# Automatic Modularization by Speciation

**Paul Darwen**
Computational Intelligence Group
School of Computer Science
University College, UNSW
Australian Defence Force Academy
Canberra ACT 2600 AUSTRALIA
darwen@cs.adfa.oz.au

**Xin Yao**
Computational Intelligence Group
School of Computer Science
University College, UNSW
Australian Defence Force Academy
Canberra ACT 2600 AUSTRALIA
xin@cs.adfa.oz.au

***Abstract*— Real-world problems are often too difficult to be solved by a single monolithic system. There are many examples of natural and artificial systems which show that a modular approach can reduce the total complexity of the system while solving a difficult problem satisfactorily. The success of modular artificial neural networks in speech and image processing is a typical example. However, designing a modular system is a difficult task. It relies heavily on human experts and prior knowledge about the problem. There is no systematic and automatic way to form a modular system for a problem. This paper proposes a novel evolutionary learning approach to designing a modular system automatically, without human intervention. Our starting point is speciation, using a technique based on fitness sharing. While speciation in genetic algorithms is not new, no effort has been made towards using a speciated population as a *complete* modular system. We harness the specialized expertise in the species of an entire *population*, rather than a single individual, by introducing a gating algorithm. We demonstrate our approach to automatic modularization by improving co-evolutionary game learning. Following earlier researchers, we learn to play iterated prisoner's dilemma. We review some problems of earlier co-evolutionary learning, and explain their poor generalization ability and sudden mass extinctions. The generalization ability of our approach is significantly better than past efforts. Using the specialized expertise of the entire speciated population though a gating algorithm, instead of the best individual, is the main contributor to this improvement.**

## 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 work well: separate parts of the brain deal with different aspects of vision [12], and artificial neural networks emulate this modular approach [8] [21].

Until now, designing a modular system has relied on human expertise (usually a committee) to manually divide a system into specialized parts. Also, manual modularization cannot be used on "knowledge-lean" or "black box" problems, for which little or no human expertise exists. There is currently no automatic and systematic method to modularize a given problem.

In this paper, we propose and evaluate a novel evolutionary learning method for automatically designing a modular system. Species automatically form to deal with separate parts of the problem, and create specialized modules. A gating algorithm manages these modules. This is different from evolutionary optimization, as the species are complementary, not competitive. Our system learns from a "black box" simulation of the problem, without *a priori* knowledge.

Our modular system performs better than past attempts at evolutionary learning, because it uses the varied expertise of the entire speciated population, instead of only the best single individual.

### A. Learning to Play Games

Co-evolution is a promising method for learning to play games. By continuously evolving novel answers to existing strategies, co-evolution should cause an evolutionary "arms race" of innovation [5, pages 179-193].

Unfortunately, previous co-evolutionary learning methods can have poor generalization ability. The reason for this is simple: a canonical GA converges to a single optimum (due to genetic drift), making it over-specialize to a single strategy. This stalls the co-evolutionary "arms race", and makes those over-specialized individuals vulnerable to novel opponents [4]. Sudden mass extinctions [14] are caused when mutation eventually produces a novel opponent [4].

Our modular approach not only solves this problem, but creates more effective strategies for the game. The actual game we study is iterated prisoner's dilemma. It has been widely used as a test for evolutionary learning [2] [4] [7] [14]. 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.

## II. Past Attempts at Evolving Intelligent Game Players

Figure 1 shows our variant of the payoff matrix of the two-player prisoner's dilemma. For the general definition of prisoner's dilemma, see [23, page 91]. In two-player *iterated* prisoner's dilemma (2IPD), the players iterate through many rounds of prisoner's dilemma, each remembering previous actions.

Axelrod [2, page 38] and others [4] [7] [14] showed that a genetic algorithm (GA), playing members of its own population against each other in 2IPD, could invent high-scoring strategies.

Figure 2 shows such a co-evolving GA learning 2IPD. From a random start, it is quickly dominated by high-scoring cooperative strategies. Then, evolution over-specializes to the current cooperative environment. Without diversity, the hoped-for "arms race" of innovation stagnates. These cooperators become so over-specialized to playing each other, that non-cooperative strategies can exploit them [4]. Mutation eventually produces a non-

**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 our variant of the 2-player prisoner's dilemma game. Each player can either cooperate (C) or defect (D). Payoff is symmetric for both players.

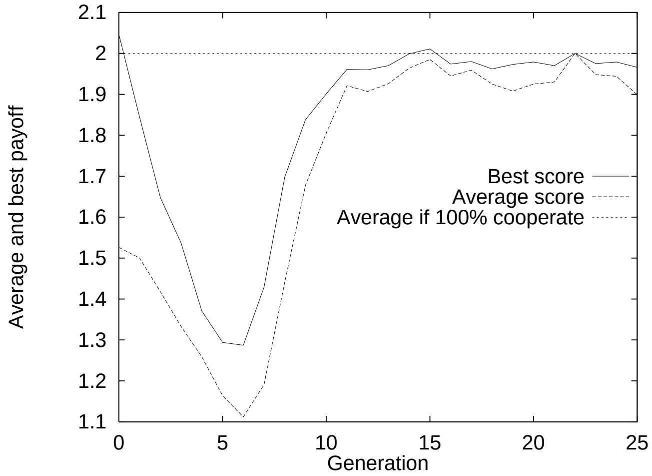


Fig. 2. A co-evolving non-speciated GA learning 2IPD. Starting from a random initial population, it quickly converges to high-scoring cooperative strategies.

cooperative strategy. As shown in Figure 3, the new defectors exploit the naïve cooperators. The naïve cooperators become extinct, quickly followed by the new defectors. Eventually, cautious cooperation is restored as the dominant strategy [4]. Spectacular mass extinctions as in Figure 3 have been observed in other co-evolutionary systems [14] [16].

One way to prevent this convergence to a population of near-identical individuals, and thus avoid over-specialized strategies, is to use speciation. Speciation maintains diverse high-quality strategies, keeping the co-evolutionary "arms race" alive, and preventing over-specialization.

The diverse strategies thus found are most effective against certain opponents, and not necessarily against all possible opponents. Smith and Gray [20] 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. Rosin and Belew [18] also used a speciated, co-evolutionary GA to learn games. However, they hoped that an individual in the GA would be well-rounded enough to play against all possible opponents. It is worth noting that in Rosin and Belew [18], the test games were simple enough for a perfect strategy to exist.

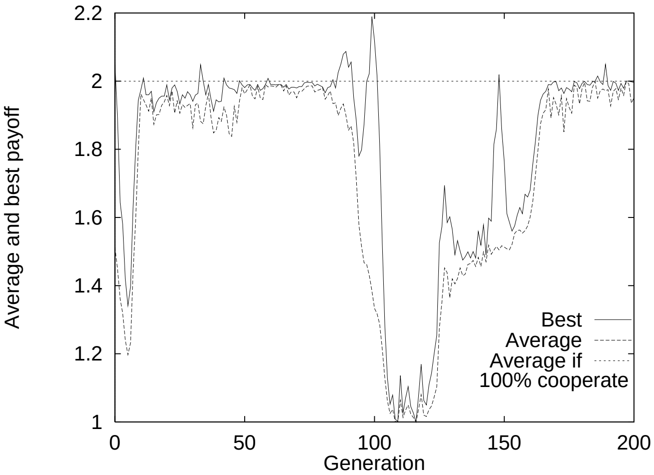


Fig. 3. A co-evolving GA learning iterated prisoner's dilemma. After the GA converges, strategies over-specialize to the cooperative environment, and become vulnerable to a new strategy that defects. Mutation eventually produces such a strategy, causing mass extinctions [4].

Although merely adding speciation prevents over-specialization and mass extinctions, using only one individual from the population effectively wastes much of the *specialized* expertise found by speciation. Instead of ignoring this specialized expertise, we keep all species and use a gating algorithm to choose from their separate expertise. This allows more specialized responses to a wide range of opponents than a single individual can manage. We show that the best "all-rounder" individual does not perform as well as our modular method.

## III. Speciation as Automatic Modularization

Automatic modularization by speciation can improve evolutionary learning. In particular, it solves the flaws in co-evolutionary learning of games. A speciated GA finds several different high-quality strategies for a given game. This lets us find separate answering strategies. To combine these answering strategies, we present a suitable gating algorithm.

### *A. Speciation with Implicit Fitness Sharing*

Speciation allows a GA to find multiple optima. Fitness sharing [6] is one GA speciation method. It reduces the fitness function at heavily-populated regions of the search space, during the course of a run. It has achieved spectacular results in difficult search problems [11]. However, fitness sharing is not without flaws [3] [19].

We use an extension to the original fitness sharing, implicit fitness sharing [9] [19]. Like the original, implicit fitness sharing also modifies the fitness function. It does this with the algorithm in Table 1. At first glance, Table 1 looks completely different to ordinary fitness sharing. Surprisingly, they share the same theoretical basis [19].

Although we use implicit fitness sharing, we could use any speciation method. Our focus is to evaluate our modular approach, which makes better use of the specialized expertise embodied in the different species. Other speciation details include:

For each strategy $i$ in the GA population, do the following $C$ times:

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

[a] Or the largest winning margin, if you prefer.

Table 1. Payoff function for implicit fitness sharing [19].

1. The best-in-sample receives a fixed credit, instead of their score as in the original implementation [19] [9]. Justification of this choice will be discussed in the longer version of this paper. In short, rewarding the best-in-sample with a fixed credit maintains all strategies that are the best reply to a particular opponent, no matter how high or low that best score actually is. Rosin and Belew [18] also use a fixed credit, but without explaining why.
2. As well as random sample selection, we evaluate reverse-assortative sample selection — the next individual to be included in the sample is the one most different from those already in it, using the Hamming distance as the measure of how different they are. By making each sample more diverse, this reduces sampling error and makes implicit fitness sharing work better, at the cost of more computation time.
3. To prevent parasite species from driving useful species to extinction, our system *penalizes* (with a small negative payoff) the single test strategy if the best-in-sample gets a higher score.

### *B. The Gating Algorithm*

In a modular system, a gating algorithm decides which module to pass the current problem to. In our game-playing problem, a gating algorithm must decide what strategy is best against the opponent's current strategy. As far as we know, this has not been done in the field of evolutionary computation. We now describe a simple gating algorithm.

The final generation of the speciated GA contains diverse expert strategies. There are several ways to produce separate modules from these. One could run a second GA for each individual, to find the single best counter-strategy for that individual. However, to keep things simple, we will presume that the final GA population contains suitable counter-strategies to its own members. So our separate modules will simply be the individuals in the last generation of the speciated GA. Even though this is the simplest possible way to obtain modules from a speciated GA, we will see below that even this gives better results.

The gating algorithm does the following:

1. Read in the final generation of the GA, excluding poor performers (the worst 25%).
2. From the most recent rounds of the game, the gating algorithm compares 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 GA individual plays against every imitator, starting from the current history of moves against the actual opponent. Those that score the equal highest against the imitators are called the *imitator-answers*. This can be pre-calculated for greater speed.
4. Finally, the imitator-answers vote on what action (co-operate or defect) to take in the current situation against the actual opponent. The very rare tied votes are resolved randomly.

The gating algorithm finds what high-quality strategy our opponent is currently following, and finds another high-quality strategy which is specialized against the one our opponent uses. This is shown conceptually in Figure 4.

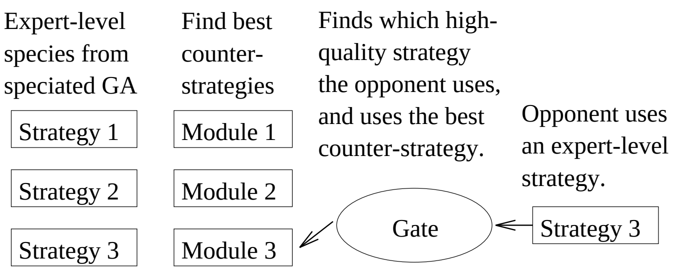


Fig. 4. The modular approach to harnessing the specialized expertise in a speciated GA's final generation.

We rely on the speciated GA to find all the high-quality strategies for the game. If the speciated GA has *not* found all the high-quality strategies, then it will overspecialize to the species it has found, causing vulnerability to unseen strategies. However, this is also true if we use the best individual instead of the modular approach. If the speciated GA finds two or more different high-quality strategies for the game, then our modular approach should work better than using the best individual.

How do we begin the game before any previous moves tell us what the opponent is like? There is no way to know what strategy our opponent is using before the first move. Our gating algorithm takes a vote of the final GA generation for the presumed pre-game moves (included in the genotype, described in Section III.C). If the vote for the first move is tied, we err on the side of caution and resolve it as a defection. We rely on the cautiousness of each strategy's initial moves.

It may not be surprising that a variety of high-quality strategies will generalize better than a single high-quality strategy. But it is surprising that so far this has been overlooked in GA learning of game strategies.

### *C. Genetic Algorithm Details*

Some attempts at co-evolution use two GA populations: an individual's fitness is determined by how well it performs against the *other* population, with no intra-population competition [18]. We would like intra-population competition. Like Axelrod [2], we use only a single GA: an individual's fitness is found by its performance against the members of its own population.

Our representation scheme is similar to Axelrod's [2]. Each individual is a set of rules stored in a look-up table that covers every possible history, and is represented as a binary string. A game that runs for 100 iterations has too many possible histories, and only recent steps will have significance for the next move. Therefore we only consider every possible history over the most recent $l$ steps, where $l \leq 4$ steps. This means an individual can only remember the $l$ most recent rounds. We represent a history with $l$ 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, a "0" cooperation.

For example, if we are looking at $l = 3$ remembered rounds, then one of the players would see the 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 action 3 steps ago (on the right), was a "1", i.e., defection. Similarly, the other player's previous actions are represented by `111`.

As strategies are too simple to learn that the game is finite, we may ignore the "shadow of the future" of IPD [1, page 13]. All games of IPD last for 100 iterations.

There are $2^{2l}$ possible histories, so each individual's genotype must be that long to represent an action for each possibility. At the start of a game, there are no previous $l$ rounds. We have to specify them with another $2l$ bits. Hence each strategy is finally represented by a binary string of length $2^{2l} + 2l$. For $l = 3$, this comes to 70 bits, allowing $2^{70} \approx 10^{21}$ possible strategies.

The parameters below were selected after a modest amount of searching. The different parameters for $l = 4$ are to keep running times reasonable. We did 30 runs for each value of $l$, the length of remembered history.

- The population size is 50 for remembered history $l = 2, 3$, and 200 for $l = 4$.
- Mutation rate is one-fifth of the inverse of the genotype length.
- We use assortative crossover with incest prevention — crossover between similar, but non-identical partners. As this reduces crossover disruption, we set the crossover rate to 1.
- We use linear ranked selection: the best in the population expects 2 offspring, and the worst no offspring.
- We use simple elitism: the best individual always appears in the next generation.
- From Table 1, sample size $\sigma = 0.2$ of the population for $l = 2, 3$ and $\sigma = 0.1$ of the population for $l = 4$.
- From Table 1, the number of samples is $C = 10$ for $l = 2, 3$ and $C = 5$ for $l = 4$.

## IV. Results and Discussion

We compare strategies produced by different evolutionary methods. We do this by seeing how well they perform against a set of test opponents. As Wolpert [22, page 246] observes, "It may turn out that ... there is no way to measure generalization more rigorously than via test problems." To obtain this test set, we performed a partial enumerative search of all strategy genotypes, for each value of remembered history $l$. For each strategy tested, we evaluated its performance against a large number of randomly chosen strategies. Only the best 25 were kept for the test set.

Tables 2 – 4 show the results against the test sets. Each game of 2IPD, both during the GA learning and against the test set, was 100 iterations. The results are for 30 of each of the following strategies:

- Best single individual from the final generation of a co-evolutionary GA without speciation, like Axelrod [2] (denoted by **best.ns**).
- A co-evolutionary GA using implicit fitness sharing with *random* sample selection, using both the best individual from the final generation (**best.sr**) and the gating algorithm on the entire final generation (**gate.sr**).
- A co-evolutionary GA using implicit fitness sharing with *reverse assortative* sample selection — the next individual to go into the sample is the one *least* like those already in it. Again, we use the best individual from the final generation (**best.sp**)and the gating algorithm on the entire final generation (**gate.sp**). With the larger population used for $l = 4$, this sample selection scheme was so slow we did not use it.

$l = 2$

| Strategy | Wins % | Ties % | Average score Own | Average score Other's |
|---|---|---|---|---|
| best.ns | 20.8 | 15.3 | 1.055 | 1.435 |
| best.sr | 34.5 | 31.2 | 1.274 | 1.164 |
| gate.sr | 50.7 | 6.3 | 1.301 | 1.139 |
| best.sp | 64.4 | 8.5 | 1.423 | 0.986 |
| gate.sp | 56.8 | 10.5 | 1.422 | 0.971 |

Table 2. Results of 30 of each strategy against the best 25 strategies from the partial enumerative search, so each line is the average of 750 games. Remembered history is $l = 2$. Note that for this simpler game, speciation gives similar results for both the best individual or the modular approach.

### *A. Discussion*

For the simplest 2IPD with history $l = 2$, where strategies are only 20 bits in length, both methods (best-in-population, and gated) give mixed performance in Table 2. This is because our simple method of finding test strategies produced a group of very similar strategies for $l = 2$. Against these non-diverse test opponents, our modular approach performs at about the same level as the best individual. This is a flaw of our test set selection, caused by the game being so simple at $l = 2$ that it lacks diverse high-quality strategies.

| | | | $l = 3$ | |
|---|---|---|---|---|
| Strategy | Wins % | Ties % | Score Own | Other's |
| best.ns | 18.5 | 5.1 | 1.016 | 1.458 |
| best.sr | 18.3 | 8.0 | 1.039 | 1.452 |
| gate.sr | 25.9 | 5.3 | 1.147 | 1.347 |
| best.sp | 19.1 | 8.9 | 1.006 | 1.470 |
| gate.sp | 39.1 | 11.3 | 1.291 | 1.125 |

Table 3. Results against the best 25 strategies from the partial enumerative search, for 2IPD with remembered history $l = 3$. This is exactly the same as used by Axelrod [2]. Note that the modular approach works better than the best individual. For both methods, improving the speciation method by using the better sample selection scheme also gives better results.

| | | | $l = 4$ | |
|---|---|---|---|---|
| Strategy | Wins % | Ties % | Score Own | Other's |
| best.ns | 34.3 | 5.7 | 1.235 | 1.595 |
| best.sr | 36.0 | 5.9 | 1.322 | 1.513 |
| gate.sr | 64.3 | 5.9 | 1.520 | 1.234 |

Table 4. Results against the best 25 strategies from the partial enumerative search, for 2IPD with remembered history $l = 4$. Reverse-assortative sample selection takes much more time for this larger population, so we only did two GA experiments (of 30 runs each).

In the more complex games of 2IPD with $l > 2$, the "all-rounder" individuals do not perform as well as our modular method, as shown in Tables 3 and 4. This validates our approach.

As expected, using a sample selection scheme that improves sample diversity gives better results than for random sample selection — the **sp** results are better in Tables 3 and 4, for both the best individual approach and our modular approach. However, the improvement is greater for the modular approach, as shown in Table 3 — against the same test of unseen opponents, the best-in-population approach increases its share of wins from 18.3% to 19.1%, while the modular approach rises from 25.9% to 39.1%. This indicates that our modular approach relies heavily on the speciated GA to maintain diversity: it works better as the speciation method works better.

In our implementation, the computation time of doing the reverse-assortative sampling is expensive, and we do not insist it is a big improvement. For many test problems, better results may be obtained by merely using random sample selection and using a larger sample size, instead of using reverse-assortative sampling. We use it to demonstrate that this small improvement to the speciation method gives dramatic improvement to our modular approach in Table 3. This indicates the need for improved speciation methods in evolutionary computation.

### *B. Future Work*

The method presented here is a very simple implementation of a modular approach. Further refinement should give improved results. For example, an earlier speciated co-evolutionary system [20] displayed a never-ending turnover of species. This suggests that sampling the GA population during a run may produce more general responses than only using the final generation.

A future project will be to apply this approach to a more complex task. The system described here is highly specific to learning game strategies, to demonstrate the modular approach. But speciation as automatic modularization can solve more realistic and complicated problems.

Modular neural networks have been effective in many problems [8] [21], and genetic algorithms have evolved non-modular neural networks for various games [10] [15]. This indicates that a modular approach to evolving neural networks should give improved results.

Some previous studies have used co-evolving "species" (actually separate GA populations), each dealing with a different aspect of the problem [13] [17]. Unfortunately, these do not use a speciation method to form the separate sub-populations — the number of sub-populations is fixed in advance. In these studies, the various sub-solutions are recombined, but modularization is still done manually.

This shows the range of problems in which good results are promised by co-evolving species as automatic modularization. We consider learning game strategies as a simple example problem, to demonstrate our approach.

## V. Conclusions

Earlier attempts at using co-evolution to learn to play games gave poor generalization ability. As a canonical GA converges to only one solution, it over-specializes to one particular strategy, becoming vulnerable to novel opponents. This can cause mass extinctions as in Figure 3 when mutation produces such a novel strategy. Such features have been observed before [14] [16], and only recently explained by us [4].

Unaware of this reason, some workers have nonetheless used both speciation and co-evolution [18] [20]. This prevents over-specialization and improves generalization ability. However, although these approaches learned diverse and specialized strategies, the authors only used one individual at a time. They did not address the question of when to use which specialized strategy. This wastes much of the diverse expertise found by speciation.

A modular approach harnesses this diverse expertise. Our method 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, our method performs better than the common approach of using the best individual from the GA's last generation.

The method presented here is a very simple implementation of a modular approach. This indicates that further refinement should yield greatly improved results.

This paper brings us a step closer to a machine learning system that creates expertise without human help or prior knowledge, using only the problem itself.

## Acknowledgements

We thank Dr Robert I. McKay and Mr Peter A. Whigham for comments on the manuscript.

## References

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

[2] R. M. Axelrod, "The evolution of strategies in the iterated prisoner's dilemma," in *Genetic Algorithms and Simulated Annealing*, (L. Davis, ed.), ch. 3, pp. 32–41, Morgan Kaufmann, 1987.

[3] P. Darwen and X. Yao, "A dilemma for fitness sharing with a scaling function," in *Proceedings of the 1995 IEEE Conference on Evolutionary Computation*, pp. 166–171, IEEE Press, 29 November - 1 December 1995.

[4] P. Darwen and X. Yao, "On evolving robust strategies for iterated prisoner's dilemma," in *Progress in Evolutionary Computation*, pp. 276–292, Berlin: Springer, 1995.

[5] R. Dawkins, *The Blind Watchmaker.* Longman, first ed., 1986.

[6] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proceedings of the Third International Conference on Genetic Algorithms*, (J. D. Schaffer, ed.), pp. 42–50, Morgan Kaufmann, June 1989.

[7] D. B. Fogel, "Evolving behaviours in the iterated prisoner's dilemma," *Evolutionary Computation*, vol. 1, no. 1, pp. 77–97, 1993.

[8] T. C. Folsom, "A modular hierarchical neural network for machine vision," in *International Joint Conference on Neural Networks*, pp. 897–902, IEEE Press, 1990.

[9] 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.

[10] B. Freisleben, "Teaching a neural network to play gomoku," in *Proceedings of the 1992 International Conference on Artificial Neural Networks*, (I. Aleksander, ed.), pp. 1659–1662, Sep. 1992.

[11] D. E. Goldberg, K. Deb, and J. Horn, "Massive multimodality, deception, and genetic algorithms," in *Parallel Problem Solving from Nature 2*, (R. Manner and B. Manderick, eds.), pp. 37–46, North-Holland, Sep. 1992.

[12] D. Grady, "The vision thing: Mainly in the brain," *Discover*, vol. 14, pp. 57–66, June 1993.

[13] P. Husbands and F. Mill, "Simulated co-evolution as the mechanism for emergent planning and scheduling," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, (R. Belew and L. Booker, eds.), pp. 264–270, Morgan Kaufmann, July 1991.

[14] K. Lindgren, "Evolutionary phenomena in simple dynamics," in *Artificial Life 2*, pp. 295–312, Addison-Wesley, 1991.

[15] D. Moriarty and R. Miikkulainen, "Evolving complex Othello strategies using marker-based genetic encoding of neural networks," Tech. Rep. AI93-206, The University of Texas at Austin, Sep. 1993.

[16] 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.

[17] M. A. Potter, K. A. De Jong, and J. J. Grefenstette, "A coevolutionary approach to learning sequential decision rules," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 366–372, Morgan Kaufmann, July 1995.

[18] C. D. Rosin and R. K. Belew, "Methods for competitive co-evolution: Finding opponents worth beating," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, (L. Eshelman, ed.), pp. 373–380, Morgan Kaufmann, July 1995.

[19] 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.

[20] R. E. Smith and B. Gray, "Co-adaptive genetic algorithms: An example in Othello strategy," in *Proceedings of the 1994 Florida Artificial Intelligence Research Symposium*, (D. D. Dankel, ed.), (Saint Petersburg, Florida), pp. 259–264, Florida AI Research Society, 1994.

[21] M. Van Hulle and G. Orban, "The EDANN concept: a modular artificial neural network model for biological vision and image processing," in *World Congress on Neural Networks-San Diego*, p. 320, Lawrence Erlbaum Associates, 1994.

[22] D. H. Wolpert, "A mathematical theory of generalization," *Complex Systems*, vol. 4, pp. 151–249, 1990.

[23] X. Yao and P. Darwen, "An experimental study of N-person iterated prisoner's dilemma games," in *Progress in Evolutionary Computation*, pp. 90–108, Berlin: Springer, 1995.