Co-evolving Intertwined Spirals

Hugues Juillé **Jordan B. Pollack**
Computer Science Department
Volen Center for Complex Systems
Brandeis University
Waltham, MA 02254-9110
<{hugues, pollack}@cs.brandeis.edu>

Abstract

We recently solved the two spirals problem, a difficult neural network benchmark classification problem, using the genetic programming primitives set up by [Koza, 1992]. Instead of using absolute fitness, we use a relative fitness based on a competition for coverage of the data set. This is a form of co-evolutionary search because the fitness function changes with the population. Because niches are opened by proportionate reproduction, rather than crowded out, and because of the crossover operator, we find solutions which have a nice modular structure. Our experiments used our Massively Parallel Genetic Programming (MPGP) system running on a SIMD machine of 4096 processors, the Maspar MP-2.

1 Introduction

This paper presents how a difficult classification problem, usually defined as the optimization of an absolute fitness function, can be converted to a co-evolutionary problem involving relative or competitive fitness. This is related to work on sorting networks [Hillis, 1992], on self-playing Backgammon learner [Tesauro, 1992], on evolving life-forms [Sims, 1994], and on co-evolving Tic-tac-toe players[Angeline & Pollack, 1993], among others. In the current situation, our construction leads to a case of co-evolution in which there is only one species.

Our experiments using genetic programming (GP) show that this approach can be more effective than the canonical GP implementation using absolute fitness, and moreover, that the co-evolution leads to an interesting modularization of the solution to classification problems.

We have used the intertwined spiral problem as a benchmark for our experiments. This learning problem, originated by Alexis Wieland, perhaps based on the cover of Perceptrons, has been a challenge for pattern classification algorithms and has been subject of much work in the AI community, in particular in the neural network field (e.g., [Lang & Witbrock, 1988, Fahlman & Lebiere, 1990, Carpenter et al., 1992]). In neural network classification systems based on linear, quasi-linear, radial, or clustering basis function, the intertwined spirals problem leads to difficulty. When it is solved, the neural net solution often has a very "expansive" description of the spiral, i.e., the conjunction of many small regions, does not generalize outside the training regions, and is thus not particularly satisfying.

This paper is organized as follows: First, section 2 presents a survey of the implementation of our Massively Parallel Genetic Programming (MPGP). This will help to understand the techniques that have been used in the following sections. Then, the intertwined spiral problem is described in section 3, along with its representation in the co-evolutionary framework of simulated competitive evolution. Results and discussion are presented in section 4.

2 Massively Parallel GP

2.1 Parallel Evaluation of S-expressions

The individual structures that undergo adaptation in GP are represented by expression trees composed from a set of primitive functions and a set of terminals (either variables or functions of no argument). Usually, the number of functions is small, and the size of the expression trees are restricted, in order to restrict the size of the search space.

In our parallel implementation, each of the 4096 processor elements (PEs) simulates a virtual processor. Following Perkis ([Perkis, 1994]), this virtual processor is a *Stack Machine* and takes the postfix representation of an S-expression as its input.

To be able to evaluate a GP expression, the following instructions are supported by the abstract machine:

- one instruction for each primitive function of the function set. At execution time, arguments for these instructions are popped from the stack into general purpose registers, the function is computed, and the result is pushed on the top of the stack.
- a `PUSH` instruction which pushes on the top of the

stack the value of a terminal,

- an IFGOTO and a GOTO instruction which are necessary for branching if conditional functions are used,
- a STOP instruction which indicates the end of the program.

This architecture allows each PE to process efficiently a different genetic program in a MIMD-like way. The parallel interpreter of the SIMD machine reads the current postfix instruction for each virtual processor and sequentially multiplexes each instruction, i.e., all processors for which the current instruction is a PUSH become active and the instruction is performed; other processors are inactive (*idle* state). Then, the same operation is performed for each of the other instructions in the instruction set in turn. Once a STOP instruction is executed for a processor, that processor becomes idle, leaving the result of its evaluation on the top of the stack. When all processors have reached their STOP instruction, the parallel evaluation of the entire population is complete.

2.2 Models for Fitness Evaluation, Selection and Recombination

The MasPar MP-2 is a two-dimensional wrap-around mesh architecture. In our implementation, the population has been modeled according to this architecture: an individual or a subpopulation is assigned to each node of the mesh and, therefore, has four neighbors. This architecture allows for the implementation of different models for fitness evaluation, selection, and recombination, using the kernel of the parallel GP described in the previous section.

In this paper, only a tournament style of competitive evolution has been used and compared to canonical GP. A more general presentation of the different strategies that have been implemented can be found in [Juillé & Pollack, 1995].

3 The Spiral Problem and the Competitive Evolution Paradigm

Experiments were conducted to compare canonical GP evolution to competitive evolution for the intertwined spiral problem. This learning problem consists in classifying points into two classes according to two intertwined spirals. The data set is composed of two sets of 97 points, on the plane between -7 and +7. These two intertwined spirals are shown as "$\times$" and "$\circ$" in figures 7 and 8.

[Koza, 1992] and [Angeline, 1995] also investigate this problem using genetic programming. The same basic form was used here to define the problem and to perform the experiments. That is, the function set is composed of: $\{+, -, *, \%, iflte, sin, cos\}$, and the terminal set is composed of: $\{x, y, \Re\}$, where $\Re$ is the ephemeral random constant.

With a population of 4096 individuals, two different approaches were taken. In the first experiment, following Koza and Angeline, the fitness function was defined as the number of hits out of 194. In the second experiment, the fitness was defined as the result of a competition among the individuals. The fact that the absolute fitness function was known was in fact ignored, and a "game" was set up in which only relative fitness was used as the basis for reproduction. The trivial idea would be to simply compare the absolute score of each individual and the winner would be the individual with the larger score. However, no useful information can be expected from this kind of competition since the individual with the largest absolute fitness will always win.

Instead, we only counted a player's ability to classify those test cases which are NOT classified by its opponent. As more or fewer copies of a player spread through the population, their scores may rise or fall depending on how many other members of the population also "cover" the test cases. As a simplified view, consider a full pairwise evaluation between one weak but unique player with 25 novel hits, against four identical players all with the same 50 hits. Although they would reproduce twice as fast in an absolute fitness competition, in this modified tournament, they will only receive their 50 points for playing the weak player, who will actually receive 4*25!

In section 4, a simplified model is presented to study the dynamics of the population evolution when an absolute fitness or a competitive fitness is used to control interactions between species. We do not play all-against-all, but several rounds of a more limited tournament competition, and compute the final relative fitness of each individual as the sum of all its scores during the competition. We can of course track the absolute fitness of a population even though it is not used otherwise.

Our hypothesis is that the competitive evolution would work better because it would promote more diversity in the population, and allow subpopulations which covered different subproblems to emerge. As copies of individuals which perform well on parts of the spiral spread through the population, they will start to meet themselves in competition, and get a score of 0. This allows other individuals which may have fewer total hits, but cover other parts of the spiral to survive. From the recombinations between individuals of those two subpopulations one might expect the emergence of a better individual that combine the "advantages" of both.

Several approaches may be used when simulating a competitive evolution ([Sims, 1994]). In this work, each generation is composed of a sequence of competition rounds in which individuals are "randomly" paired up. In fact, because the architecture of the MP-2 does not have any fast-access shared-memory, this random pair-

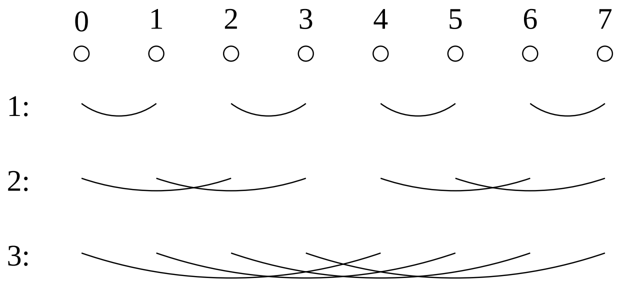

Figure 1: Divide-and-conquer communication pattern.

ing can only be approximated. This approximation has been achieved by using a fixed communication pattern (the divide-and-conquer communication pattern) but individuals are not assigned to the same processor all the time: an exchange can occur between paired processors. This results in a "random walk" performed by each individual in the population which makes them meet a representative sample of opponent.

More precisely, for a given generation, all the processors are paired according to the divide-and-conquer communication pattern (such a pattern is presented in figure 1, in the case of eight processors). Each possible pairing corresponds to one competition round. Operations performed in such a round are the following:

- each processor computes the score of the individual associated to it: the number of hits that his opponent doesn't get (i.e., the number of test cases that the individual correctly classifies and that his opponent doesn't).
- for each pair of processor, one is arbitrarily selected. Then, the individual that corresponds to this processor is assigned to the left processor and the second one's individual is assigned to the right one. This way, using divide and conquer, each individual will perform a "random walk" in the population.

By performing this sequence of pairing according to the communication pattern in the same generation several times, we approximate an evaluation of each individual against the whole population. At the end of those competitions, each individual's fitness is calculated by summing all its scores in the competition.

The objective of this strategy is to make each individual in the population meet a representative sample of individuals in order to refine their relative fitness. This kind of "tournament" strategy (with no elimination) allows us to achieve this goal since, by randomly picking a winner at each round, paired individuals are very likely to be different from one tournament to the other.

Once individual fitness is evaluated, selection and recombination are performed according to a fitness proportionate rule. Details of the implementation of selection and recombination procedures for MPGP can be found in [Juillé & Pollack, 1995].

If $(4 * x^2 - y^2) < 0.0$ **then**
  return $(sin(-3.0 * y))$;
else
  return $\left(sin(\frac{0.3214 * x}{0.04762 - cos(sin(\frac{y}{x} * 0.7874))}) \right)$;
endif

Figure 4: Interpretation of the solution for the intertwined spiral problem.

4 Results and Discussion

For the two classes of experiments, we performed 25 runs and each run was stopped after 300 generations. At each generation, 90% of the population was replaced by offspring resulting from recombination and the remaining 10% was the result of fitness proportionate reproduction. In order to make each individual meet a significant number of opponent, eight successive tournaments were performed at each generation. Thus, each individual met 96 opponents (there are 12 rounds in a tournament with a population size of 4096).

Our results concerning the performance of these 50 runs, shown in figure 2, illustrate that under similar reproductive parameter conditions, competitive evolution statistically outperforms the absolute fitness approach. This is clearly not a strong result over all settings of the parameters, which will require many further simulations, but a demonstration of an interesting effect: The absolute fitness strategy works better for the first 40 generations, but, because of its rapid convergence, improves its solution very slowly thereafter. On the other hand, competitive evolution, which allows more diversity in the population, has a slower initial improvement in fitness, but ultimately has better overall performance.

Only a few runs of competitive fitness have provided us with a perfect (194 hit) solution for the intertwined spiral problem within 300 generations. We harvested some of the perfect classification solutions; one of the shortest of these S-expressions has 52 atoms and is shown in figure 3.

Because of the relatively small size of this result we were able to analyze it and simplify it mathematically, by collapsing constant calculations, removing insignificant digits, algebraic simplification, and elimination of redundant "introns". This analysis resulted in the conditional function presented in figure 4.

Basically, this solution splits the geometric plane into two domains and a different function is used for each domain. Figure 5 displays the $4x^2 - y^2$ function which multiplexes the two other functions, shown in figure 6, to create the spiral.

The resulting function is shown in figure 7, which plots the function (above/below 0) along with the training data on the range -10 to 10. Although it does not form a perfect spiral, it does continue to simulate a spiral way outside the original training range. In another set of ex-

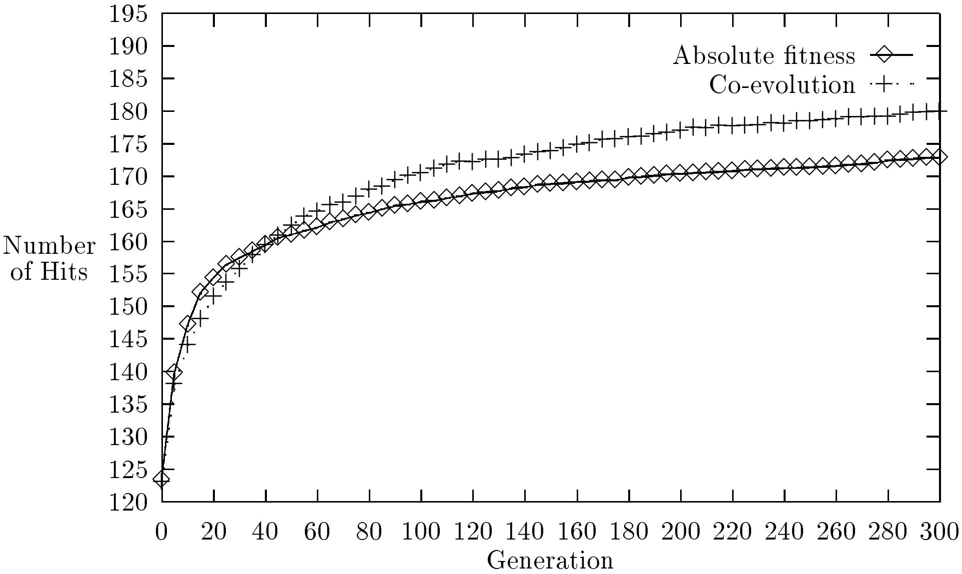

Figure 2: Absolute fitness versus competitive evolution for the intertwined spiral problem.

```
(sin (% (iflte (- (- (- (* _A _A) (sin (% (iflte  -0.52381
                                                    _B
                                                    (sin -0.33333)
                                                    -0.33333)
                                          -0.33333)))
                     (* _B _B))
                  (% _A (% -0.33333 _A)))
               -0.80952
               _B
               (sin (% (% _A
                          (- (cos (sin (* (cos (sin -0.52381))
                                          (% _B (% _A (- (cos -0.33333) 0.04762))))))
                             0.04762))
                       (sin (sin -0.33333)))))
        -0.33333))
```

Figure 3: A 52-atom S-expression scoring 194 for the intertwined spiral problem.

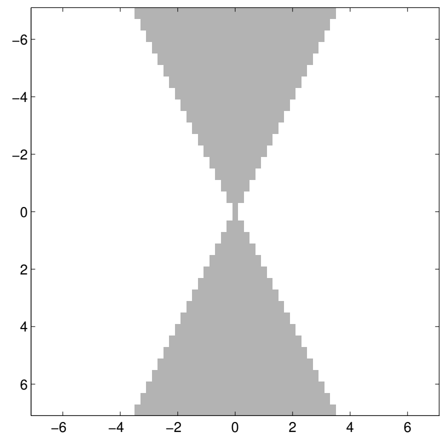

Figure 5: $4x^2 - y^2 < 0$, used to divide the plane into two domains.

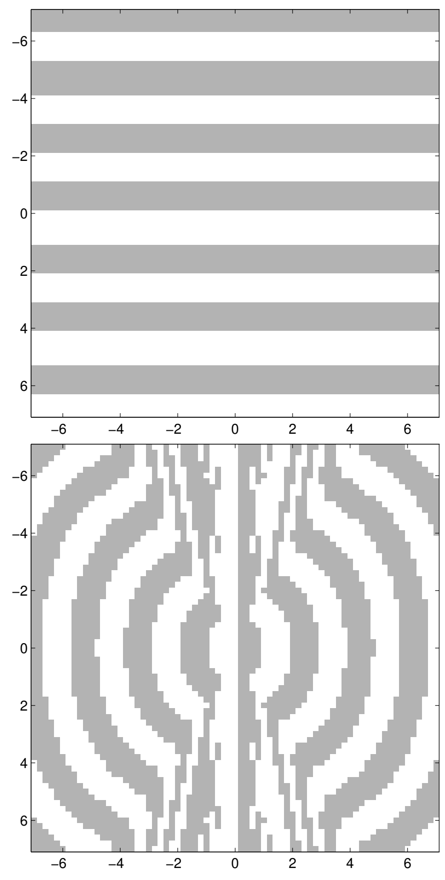

Figure 6: $sin(-3y)$ and the other function which are selectively added to make a spiral.

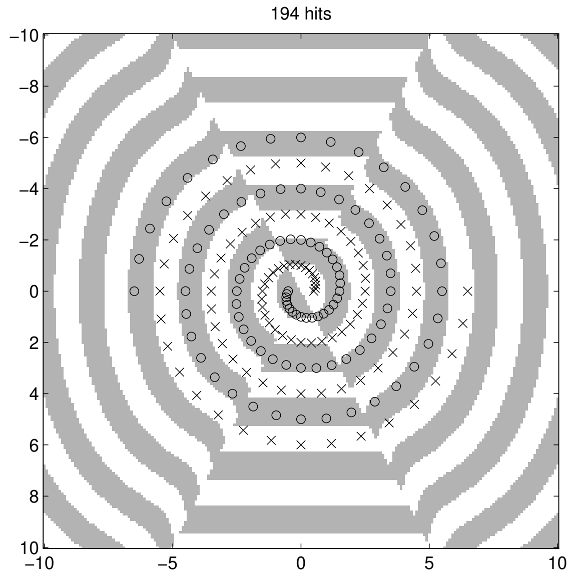

Figure 7: Perfect score generalizing classification of the two intertwined spirals.

periments (limited to 100 generations) another perfect solution has been discovered (presented in figure 8). The S-expression representing this solution is composed of 161 atoms.

Furthermore, we believe that compared to neural network solutions, which are often the composition of hundreds of clusters or decision boundaries, and some of the GP solutions shown by Koza, ours is the most perspicacious to date. The fact that the spiral is composed of a synergy of two (or more) functions which cover separate parts of the data supports the hypothesis that the relative fitness competitive evolution strategy can be more effective than an absolute fitness function.

To support the idea that competitive evolution allows subpopulations that cover different parts of the problem to survive, contrary to an absolute fitness driven evolution, we propose the following analysis. The two-intertwined problem is a classification problem. Therefore, it can be seen as a set of test cases and the population can be split up into groups (or clusters) in which individuals would cover exactly the same test cases. For the sake of clarity, let us formalize this idea. First, let us define the following terms:

- n: number of test cases,
- m: number of groups (or clusters) that compose the population,
- t_i: i^{th} test case,
- G_j: j^{th} group of individuals,
- $s_j(t)$: size (number of individuals) of group G_j at time t,

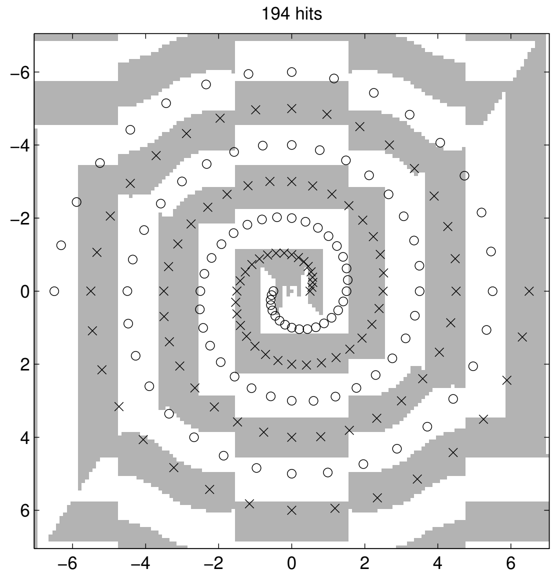

Figure 8: Another perfect score classification of the two intertwined spirals.

- $T(G_j)$: returns a list of booleans of size n in which the k^{th} entry indicates whether the test case t_k is covered by individuals in group G_j,
- $\mathcal{B}$: matrix whose rows are the $T(G_j)$.

$$\mathcal{B} = \begin{pmatrix} T(G_1) \\ \vdots \\ T(G_m) \end{pmatrix}$$

Each entry $b_{i,j}$ of $\mathcal{B}$ is a 1 (*true*) if the test case t_j is covered by the group G_i, and 0 (*false*) otherwise.

For the following, let us consider an example:

- $n = 10$,
- $m = 5$,
- $T(G_1) = (0,1,1,1,0,1,0,1,0,1)$,
  $T(G_2) = (1,0,0,0,1,0,0,0,1,0)$,
  $T(G_3) = (0,1,0,1,0,0,1,1,0,1)$,
  $T(G_4) = (0,0,1,1,0,0,0,0,1,0)$,
  $T(G_5) = (0,1,0,1,1,0,1,0,0,1)$

Now, we can define the $(m \times m)$ square matrix $\mathcal{A}$ for which each entry $a_{i,j}$ equals the number of test cases correctly classified by group G_i but that group G_j doesn't. Thus, each entry of $\mathcal{A}$ is defined as follows:

$$a_{i,j} = \sum_{l=1}^{n} (b_{i,l} \wedge \neg b_{j,l})$$

With our example, $\mathcal{A}$ equals:

$$\mathcal{A} = \begin{pmatrix} 0 & 6 & 2 & 4 & 3 \\ 3 & 0 & 3 & 2 & 2 \\ 1 & 5 & 0 & 4 & 1 \\ 1 & 2 & 2 & 0 & 2 \\ 2 & 4 & 1 & 4 & 0 \end{pmatrix}$$

Now, we can define the fitness function for the two cases of study:

- *absolute fitness* for an individual of group G_j:

  $$f_a(j) = \sum_{l=1}^{n} b_{jl}$$

  For our example:
  $f_a(1) = 6; f_a(2) = 3; f_a(3) = 5; f_a(4) = 3; f_a(5) = 5$

- *relative fitness* for an individual of group G_j:

  $$f_r(j) = \sum_{l=1}^{m} (s_l(t) \times a_{j,l})$$

  According to this definition, each individual competes once against all other individuals in the population. In our experiments, we only approximate this by making each individual compete against a sample of the population.

For the sake of simplicity, we assume there are no recombination between individuals but only fitness proportionate reproduction. Indeed, what we want to show with this simplified model is that subpopulations that cover different test cases survive when competitive evolution is involved. Therefore, we want to study the dynamics of the evolution of group size with time. A simple rule for fitness proportionate reproduction gives us:

$$s_j(t+1) = s_j(t) \times \left(1 + \alpha \times \frac{f(j) - \overline{f}}{\overline{f}}\right)$$

where:

- α is a parameter that controls the speed of the simulated evolution,
- $f(j)$ is the fitness. According to the case of study, it is replaced by $f_a(j)$ or $f_r(j)$.
- $\overline{f}$ is the average of the fitness.

A normalization step for $s_j(t+1)$ is then performed in order to keep a constant population size. If $\alpha = 1$, we get the more well-known expression for fitness proportionate reproduction:

$$s_j(t+1) = s_j(t) \times \frac{f(j)}{\overline{f}}$$

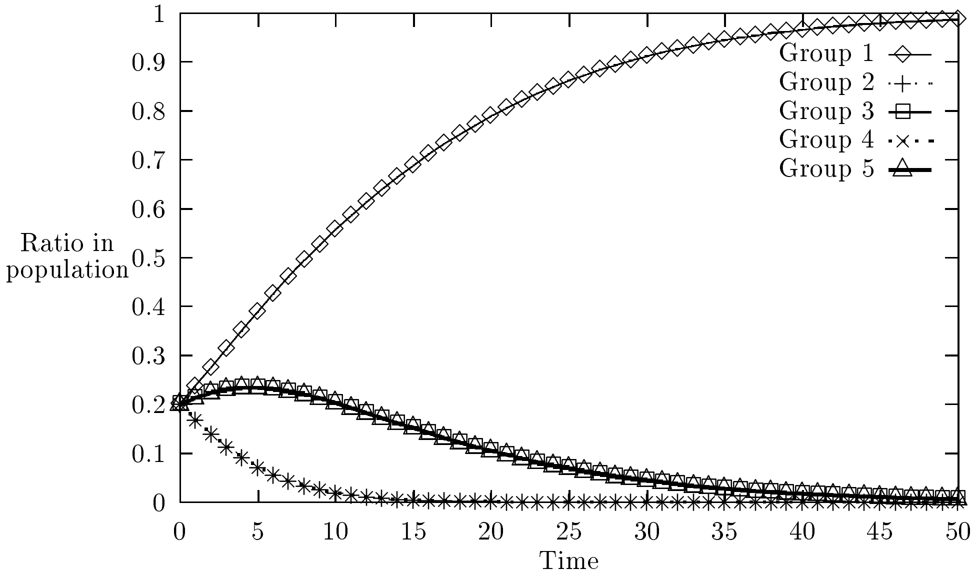

Figure 9: Evolution of the ratio for each group in the population in the case of an absolute fitness.

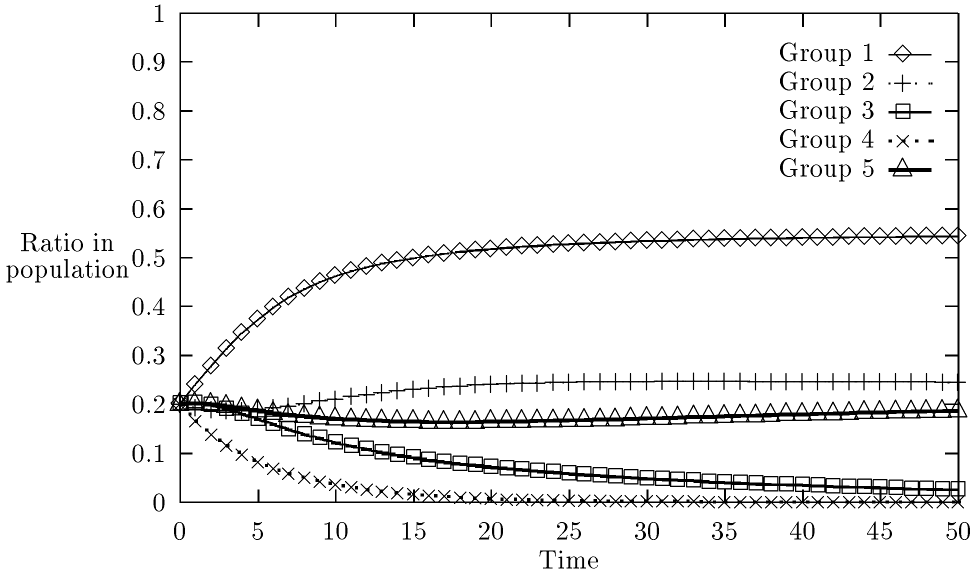

Figure 10: Evolution of the ratio for each group in the population in the case of a relative fitness.

The graphical results of the evolution of the ratio for each group in the population for the two models of evolution are presented in figure 9 and figure 10. For our analysis, all groups have the same size at $t = 0$, and we took $\alpha = 0.5$. Clearly, in the case of the absolute fitness, all the population is overcome by the first group which has the largest absolute fitness ($f_a(1) = 6$). On the contrary, in the case of the competitive evolution, once stability is reached, the first group takes only 50% of the population and groups 2 and 5 around 20%. Group 4 disappears very quickly and group 3 takes only a tiny part of the population. It is possible to prove that these results are independent of the initial size of the different groups (at the condition that no group has null size) and of the value of the non-null parameter α.

The aim of this analysis is to show that competitive evolution allows different subpopulation to survive, contrary to the canonical model of evolution, therefore keeping more diversity in the population. We also believe that in the case of the intertwined spiral problem, recombination of individuals from different subpopulation are at the origin of new solutions that cover some part of the problem that were specific to each of the two subpopulations.

5 Conclusion

Experiments presented in this paper show that the classification procedure for a challenging problem (namely, the intertwined spiral problem) can be significantly improved by using a relative fitness rather than absolute fitness approach. The heuristic behind this co-evolutionary approach is that in the classification "game", more fitness payoff is made to players which correctly classify subproblems covered by fewer other players (S-expressions in the case of GP). So a particular player may survive by covering areas of the problem which are not otherwise classified correctly by the majority population. We are thus giving these minority players an ability to survive until their niche contribution to the total solution is overtaken by, or incorporated by crossover into, new stronger players.

This approach generally can be seen as a method for fitness sharing ([Goldberg & Richardson, 1987]). However, fitness sharing explicitly controls the creation of niches by derating the fitness of population elements according to the number of individuals in niches. In contrast, co-evolution allows the niches to be formed and maintained dynamically as a consequence of the relative fitness, and prevents suboptimal niches from overcoming the population.

References

[Angeline, 1995] Angeline, P. J. (1995). Two self-adaptive crossover operations for genetic programming. In *Advances in Genetic Programming II*. MIT Press. To appear.

[Angeline & Pollack, 1993] Angeline, P. J. & Pollack, J. B. (1993). Competitive environments evolve better solutions for complex tasks. In Forrest, S. (Ed.), *The Fifth International Conference on Genetic Algorithms*, pp. 264–270. Morgan Kaufmann.

[Carpenter et al., 1992] Carpenter, G., Grossberg, S., Markuzon, N., Reynolds, J., & Rosen, D. (1992). Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3:698–713.

[Fahlman & Lebiere, 1990] Fahlman, S. E. & Lebiere, C. (1990). The cascade-correlation learning architecture. In Touretzky (Ed.), *Advances in Neural Information Processing Systems 2*. Morgan Kauffman.

[Goldberg & Richardson, 1987] Goldberg, D. E. & Richardson, J. J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J. J. (Ed.), *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pp. 41–49. Lawrence Erlbaum Associates.

[Hillis, 1992] Hillis, W. D. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In Langton, C., Taylor, C., Farmer, J. D., & Rasmussen, S. (Eds.), *Artificial Life II*, pp. 313–324. Addison Wesley.

[Juillé & Pollack, 1995] Juillé, H. & Pollack, J. B. (1995). Massively parallel genetic programming. In Angeline & Kinnear (Eds.), *Advances in Genetic Programming II*. MIT Press. To appear.

[Koza, 1992] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

[Lang & Witbrock, 1988] Lang, K. J. & Witbrock, M. J. (1988). Learning to tell two spirals apart. In *Proceedings of the 1988 Connectionist Summer Schools*. Morgan Kaufmann.

[Perkis, 1994] Perkis, T. (1994). Stack-based genetic programming. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*. IEEE Press.

[Sims, 1994] Sims, K. (1994). Evolving 3d morphology and behavior by competition. In Brooks & Maes (Eds.), *Artificial Life IV*, pp. 28–39. MIT Press.

[Tesauro, 1992] Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, 8:257–277.