A Cooperative Coevolutionary Approach to Function Optimization

Mitchell A. Potter and Kenneth A. De Jong

Computer Science Department, George Mason University, Fairfax, VA 22030, USA
mpotter@gmu.edu

Abstract. A general model for the coevolution of cooperating species is presented. This model is instantiated and tested in the domain of function optimization, and compared with a traditional GA-based function optimizer. The results are encouraging in two respects. They suggest ways in which the performance of GA and other EA-based optimizers can be improved, and they suggest a new approach to evolving complex structures such as neural networks and rule sets.

1 Introduction

Genetic algorithms (GAs), originally conceived by Holland [10], represent a fairly abstract model of Darwinian evolution and biological genetics. They evolve a population of competing individuals using fitness-biased selection, random mating, and a gene-level representation of individuals together with simple genetic operators (typically, crossover and mutation) for modeling inheritance of traits.

These GAs have been successfully applied to a wide variety of problems including multimodal function optimization, machine learning, and the evolution of complex structures such as neural networks and Lisp programs. At the same time, difficulties can and do arise in forcing every problem domain into this traditional GA model. One of the earliest examples of this is the application of GAs to rule learning. Evolving rule sets of varying length and complexity doesn't map neatly into the traditional GA paradigm, resulting in a variety of extensions including Holland's classifier system, Smith's LS system, and Grefenstette's SAMUEL system [11, 14, 7].

In this paper we present an extension of the traditional GA model which appears to have considerable potential for representing and solving more complex problems by explicitly modeling the coevolution of cooperating species. We provide some initial insight into this potential by illustrating its behavior on the well-studied domain of function optimization. We conclude with a brief discussion of work in progress on more complex domains.

2 Cooperative Coevolutionary Genetic Algorithms

The hypothesis underlying the ideas presented here is that, in order to evolve more and more complex structures, explicit notions of modularity need to be

introduced in order to provide reasonable opportunities for complex solutions to evolve in the form of interacting co-adapted subcomponents. Examples of this show up in the need for rule hierarchies in classifier systems and subroutines in genetic programming.

The difficulty comes in finding reasonable computational extensions of the GA paradigm for which such substructure "emerges" rather than being prespecified by the user. At issue here is how to represent such substructures and how to apportion credit to them for their contributions to the problem solving activity. Classifier systems attempt to accomplish this via a single population of interacting rules whose individual fitnesses are determined by their interactions with other rules via a simulated micro-economy [11]. Other extensions have been proposed to encourage the emergence of niches and species in a single population [4, 3], in which individual niches represent competing (rather than cooperating) solutions to the problem.

The use of multiple interacting subpopulations has also been explored as an alternate mechanism for representing the coevolution of species, but has focused primarily on a fixed number of subpopulations each evolving competing (rather than cooperating) solutions (e.g. [8, 2, 15, 16]). The previous work that has looked at coevolving multiple cooperative species in separate subpopulations has involved a user-specified decomposition of the problem into species (see, for example, [12] or [9]).

The system we envision combines and extends these ideas in the following ways: 1) a species represents a subcomponent of a potential solution; 2) complete solutions are obtained by assembling representative members of each of the species present; 3) credit assignment at the species level is defined in terms of the fitness of the complete solutions in which the species members participate; 4) when required, the number of species (subpopulations) should itself evolve; and 5) the evolution of each species (subpopulation) is handled by a standard GA. We call such systems *cooperative coevolutionary genetic algorithms* (CCGAs).

As a first step we have chosen the domain of function optimization as the test bed for our ideas. The choice has several advantages. It is a well-studied area with respect to the use of evolutionary algorithms providing us with a solid frame of reference. It is also the case that there is a natural decomposition of the problem into a fixed number of individual subcomponents, namely, the N parameters of the function to be optimized. This allowed us to defer the most difficult of the five issues listed above (the birth and death of species) and concentrate on designing and testing the remaining four features of the proposed system.

3 Cooperative Coevolutionary Function Optimization

If we think of a solution to a function optimization problem as consisting of specifying the value of N parameters (variables), a natural decomposition is to maintain N subpopulations (species) each of which contains competing values for a particular parameter. One can then assign fitness to a particular value (member) of a particular subpopulation by assembling it along with selected

members of the other subpopulations to form one or more N-dimensional vectors whose fitness can be computed in the normal fashion, and using those results to assign fitness to the individual component being evaluated. That is, the fitness of a particular member of a particular species is computed by estimating how well it "cooperates" with other subspecies to produce good solutions.

As a first test of these ideas, the traditional GA shown in figure 1 was extended to the CCGA-1 model given in figure 2.

```
gen = 0
Pop(gen) = randomly initialized population
evaluate fitness of each individual in Pop(gen)
while termination condition = false do begin
        gen = gen + 1
        select Pop(gen) from Pop(gen − 1) based on fitness
        apply genetic operators to Pop(gen)
        evaluate fitness of each individual in Pop(gen)
        end
```

Fig. 1. Traditional GA

```
gen = 0
for each species s do begin
    Pop_s(gen) = randomly initialized population
    evaluate fitness of each individual in Pop_s(gen)
    end
while termination condition = false do begin
        gen = gen + 1
        for each species s do begin
            select Pop_s(gen) from Pop_s(gen − 1) based on fitness
            apply genetic operators to Pop_s(gen)
            evaluate fitness of each individual in Pop_s(gen)
            end
        end
```

Fig. 2. CCGA-1

CCGA-1 begins by initializing a separate population of individuals for each function variable. The initial fitness of each subpopulation member is computed by combining it with a random individual from each of the other species and applying the resulting vector of variable values to the target function.

After this startup phase, each of the individual subpopulations in CCGA-1 is coevolved in a round-robin fashion using a traditional GA. The fitness of a subpopulation member is obtained by combining it with *the current best* subcomponents of the remaining (temporarily frozen) subpopulations. This is certainly

the simplest form of credit assignment one could imagine. It has some potential problems (such as undersampling and "greediness"), but gives us a starting point for further refinements.

Although this sequential version of the algorithm could be characterized more accurately as quasi-coevolutionary, a fully coevolutionary implementation is also possible (and on our list to explore at a later date) in which each species only occasionally communicates with the other species. Such an asynchronous version of the algorithm would be particularly well suited to a parallel implementation in which each species is evolved on a separate processor.

4 Experimental Results

We evaluated CCGA-1 by comparing its performance with the performance of a standard GA on several function optimization problems. The coevolutionary and standard GA differ only as to whether they utilize multiple species as described in the previous section. All other aspects of the algorithms are equal and are held constant over all experiments. Specifically, the algorithms have the following characteristics:

representation:	binary (16 bits per function variable)
selection:	fitness proportionate
fitness scaling:	scaling window technique (width of 5)
elitist strategy:	single copy of best individual preserved
genetic operators:	two-point crossover and bit-flip mutation
mutation probability:	1/chromlength
crossover probability:	0.6
population size:	100

All the functions used in these experiments have been defined such that their global minimums are zero. The primary performance metric used in evaluating the algorithms is the minimum function value found after a fixed number of function evaluations. Each of the results reported for this metric represents an average computed over fifty runs.

The first set of experiments is performed on four highly multimodal functions that have been used in other experimental comparisons of evolutionary algorithms [13, 6, 1]. We will refer to these functions by the names of the researchers who first proposed them—Rastrigin, Schwefel, Griewangk, and Ackley. The Rastrigin function is defined as

$$f(\boldsymbol{x}) = 3.0n + \sum_{i=1}^{n} x_i^2 - 3.0\cos(2\pi x_i),$$

where $n = 20$ and $-5.12 \leq x_i \leq 5.12$. The global minimum of zero is at the point $\boldsymbol{x} = (0, 0, \cdots)$. The primary characteristic of this function is the existence of many suboptimal peaks whose values increase as the distance from the global

optimum point increases. The Schwefel function is defined as

$$f(\boldsymbol{x}) = 418.9829n + \sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right),$$

where $n = 10$ and $-500.0 \leq x_i \leq 500.0$. The global minimum of zero is at the point $\boldsymbol{x} = (420.9687, 420.9687, \cdots)$. The interesting characteristic of this function is the presence of a second-best minimum far away from the global minimum—intended to trap optimization algorithms on a suboptimal peak. The Griewangk function is defined as

$$f(\boldsymbol{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right),$$

where $n = 10$ and $-600.0 \leq x_i \leq 600.0$. The global minimum of zero is at the point $\boldsymbol{x} = (0, 0, \cdots)$. This function has a product term, introducing an interdependency between the variables. This is intended to disrupt optimization techniques that work on one function variable at a time. The Ackley function is defined as

$$f(\boldsymbol{x}) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right),$$

where $n = 30$ and $-30.0 \leq x_i \leq 30.0$. The global minimum of zero is at the point $\boldsymbol{x} = (0, 0, \cdots)$. At a low resolution the landscape of this function is unimodal; however, the second exponential term covers the landscape with many small peaks and valleys.

The graphs in figure 3 show the minimum value found (best individual) as a function of the number of function evaluations averaged over fifty runs using the Rastrigin, Schwefel, Griewangk, and Ackley functions. Both algorithms were terminated after 100,000 function evaluations. In all cases CCGA-1 significantly outperformed the standard GA both in the minimum value found and in the speed of convergence to zero. The statistical significance of these results was verified using a two sample t test.

Recall that CCGA-1 evolves each species (function variable) in a round-robin fashion using the current best values from the other species. This is quite similar in style to the family of numerical optimization techniques which proceed by optimizing one function variable at a time while holding the other variables constant. It is well known that such procedures work well on functions whose variables are reasonably independent, but have difficulties with functions with interacting variables.

On closer inspection, we noticed that CCGA-1 demonstrated slightly less of an advantage over the standard GA on the Griewangk function than on the other three functions. We hypothesize that this due to the interdependencies between the function variables introduced by the Griewangk product term, and selected an additional function (F2) from the original De Jong test suite [5] that has

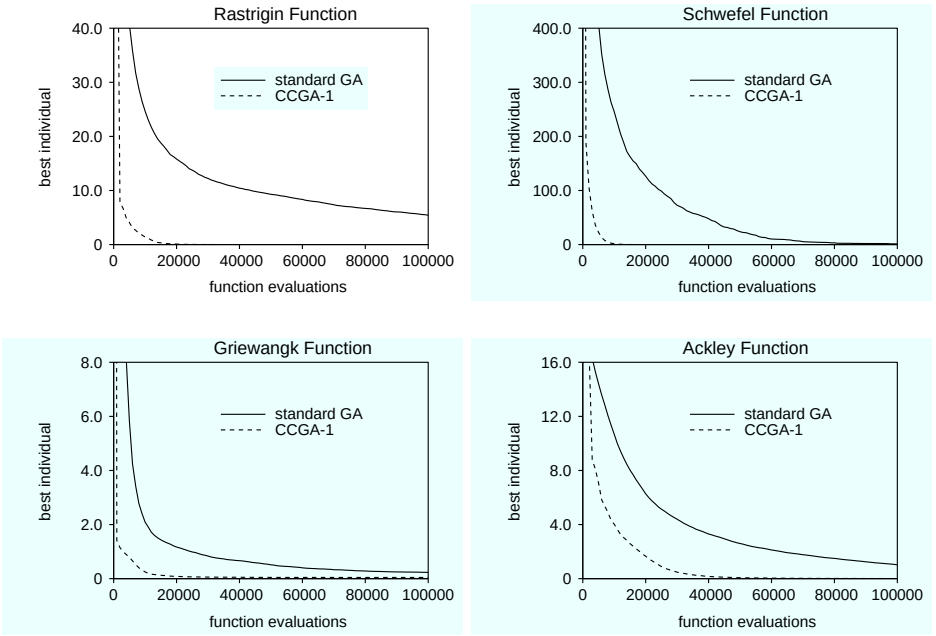

Fig. 3. Comparisons of standard GA and CCGA-1 performance

even stronger variable interactions than the Griewangk function. This function is called the Rosenbrock function and is defined as

$$f(\boldsymbol{x}) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2,$$

where $-2.048 \leq x_i \leq 2.048$. The global minimum of zero is at the point $(1, 1)$. The Rosenbrock function is characterized by an extremely deep valley along the parabola $x_1^2 = x_2$ that leads to the global minimum.

As illustrated in figure 4, CCGA-1 performed much worse than the standard GA on the Rosenbrock function, supporting our hypothesis that interacting variables (product terms) would present difficulties.

We felt that much of the source of this difficulty was due to the simple credit assignment algorithm in CCGA-1. To test this hypothesis, we modified the credit assignment algorithm as follows. Each individual in a subpopulation is evaluated by combining it with the best known individual from each of the other species and with a random selection of individuals from each of the other species. The two resulting vectors are then applied to the target function and the better of the two values is returned as the offspring's fitness.

We evaluated this variant (CCGA-2) on the Rosenbrock and the earlier Rastrigin function to see to what extent performance is improved on interacting variable problems, and to assess what performance penalty (if any) is observed on a representative of the non-interacting variable problems.

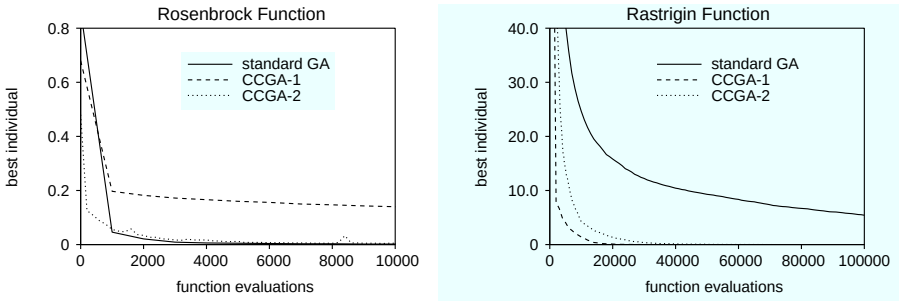

Fig. 4. Comparisons of standard GA, CCGA-1, and CCGA-2 performance

As illustrated in figure 4, CCGA-2 performed as well as the standard GA on the Rosenbrock function, but at the expense of a slight decrease in performance from CCGA-1 on the non-interacting variable problem. These results were tested as before for significance.

Although we have emphasized the number of function evaluations as a measure of cost, the amount of computation required to perform the experiments should also be briefly mentioned. Because the standard GA represents the entire set of function variables in each of its chromosomes while the CCGA algorithms only represent a single function variable in each of their chromosomes, there is much more overhead in a standard GA associated with the genotype to phenotype mapping process. To evaluate each new individual, the standard GA performs a genotype to phenotype mapping on all function variables while the CCGA algorithms only need to apply the mapping to a single function variable. As the dimensionality of the problem increases, this additional overhead becomes considerable. As an illustration, while optimizing the Ackley function of thirty dimensions the standard GA took approximately six times longer to complete a given number of fitness evaluations than the CCGA counterpart.

5 Discussion, Conclusions and Future Work

The results presented here are preliminary in nature, but provide initial evidence of the potential problem solving capabilities of cooperative coevolutionary systems. To make any strong claims concerning their value for function optimization, further refinement of the approach is required as well as additional comparisons with other existing optimization techniques. There are, however, several aspects of this approach that deserve some attention.

First, note that any evolutionary algorithm (EA) can be used to evolve the subpopulations—GAs just happen to be our favorite choice. Hence we encourage others to explore the potential of extending their own favorite EA to a CCEA. The evidence presented here suggests that the result may be improved problem solving capabilities at lower computational costs.

A second feature of these systems is a natural mapping onto coarsely grained parallel architectures. We plan to utilize networked workstations to coevolve the species in parallel on more difficult problem classes.

Finally, we feel that the real potential of these cooperative coevolutionary systems will become apparent when applied to domains requiring the evolution of more complex structures such as neural networks and sets of rules. We hope to provide such evidence in the near future.

References

1. Bäck, T., Schwefel, H.-P.: An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation **1**(1) (1993) 1–23
2. Cohoon, J.P., Hegde, S.U., Martin, W.N., Richards, D.: Punctuated equilibria: a parallel genetic algorithm. Proceedings of the Second International Conference on Genetic Algorithms (1987) 148–154
3. Davidor, Y.: A naturally occuring niche & species phenomenon: the model and first results. Proceedings of the Fourth International Conference on Genetic Algorithms (1991) 257–263
4. Deb, K., Goldberg, D.E.: An investigation of niche and species formation in genetic function optimization. Proceedings of the Third International Conference on Genetic Algorithms (1989) 42–50
5. DeJong, K.A.: Analysis of Behavior of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan, Ann Arbor, MI (1975)
6. Gordon, V.S., Whitley, D.: Serial and parallel genetic algorithms as function optimizers. Proceedings of the Fifth International Conference on Genetic Algorithms (1993) 177–183
7. Grefenstette, J.J.: A system for learning control strategies with genetic algorithms. Proceedings of the Third International Conference on Genetic Algorithms (1989) 183–190
8. Grosso, P.B.: Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model. PhD thesis, University of Michigan, Ann Arbor, MI (1985)
9. Hills, D.W.: Co-evolving parasites improve simulated evolution as an optimization procedure. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, Artificial Life II (1990) 313–324
10. Holland, J.H.: Adaptation in Natural and Artificial Systems (1975)
11. Holland, J.H., Reitman, J.S.: Cognitive systems based on adaptive algorithms. In D.A. Waterman and F. Hayes-Roth, editors, Pattern-Directed Inference Systems (1978)
12. Husbands, P., Mill, F.: Simulated co-evolution as the mechanism for emergent planning and scheduling. Proceedings of the Fourth International Conference on Genetic Algorithms (1991) 264–270
13. Mühlenbein, H.: The parallel genetic algorithm as function optimizer. Proceedings of the Fourth International Conference on Genetic Algorithms (1991) 271–278
14. Smith, S.F.: Flexible learning of problem solving heuristics through adaptive search. Proceedings of the Eighth International Joint Conference on Artificial Intelligence (1983) 422–425
15. Tanese, R.: Distributed genetic algorithms. Proceedings of the Third International Conference on Genetic Algorithms (1989) 434–439

16. Whitley, D., Starkweather, T.: Genitor II: a distributed genetic algorithm. Journal of Experimental and Theoretical Artificial Intelligence **2** (1990) 189–214