Evolving Complex Structures via Cooperative Coevolution[*]

Kenneth A. De Jong and Mitchell A. Potter

Abstract

A cooperative coevolutionary approach to learning complex structures is presented which, although preliminary in nature, appears to have a number of advantages over non-coevolutionary approaches. The cooperative coevolutionary approach encourages the parallel evolution of substructures which interact in useful ways to form more complex higher level structures. The architecture is designed to be general enough to permit the inclusion, if appropriate, of *a priori* knowledge in the form of initial biases towards particular kinds of decompositions. A brief summary of initial results obtained from testing this architecture in several problem domains is presented which shows a significant speedup over more traditional non-coevolutionary approaches.

1 INTRODUCTION

For both natural and artificial systems the ability to evolve complex structures is desirable, but difficult to achieve. Our conventional evolutionary algorithms typically provide performance-oriented feedback and as a consequence place little if any selection pressure on the many possible forms identically performing structures may take. As a result "pleasing" structures involving modularity, hierarchies, and so on seldom evolve unless special additional steps are taken such as including a structural evaluation component in the feedback function, restricting the representation and operators to only produce desirable structures, or manually decomposing problems and evolving the desired subcomponents independently. In such approaches, the designer plays a significant role in explicitly introducing his/her notion as to

[*]Accepted for publication in the Fourth Annual Conference on Evolutionary Programming, March 1-3, 1995, San Diego, CA

the form such structures should take, appropriate decompositions, the sequence in which substructures are evolved, and so on.

In this paper we present a cooperative coevolutionary approach to learning complex behaviors which, although preliminary in nature, appears to require much less explicit control by the designer. In our approach shown in figure 1, multiple instances of an evolutionary algorithm (EA) are run in parallel, each instance of which evolves a species of individuals which represent possibly useful substructures. The form of these substructures and their interaction with other substructures is not enforced explicitly by the designer. He/she may seed initial populations with potential candidates for useful substructures; however, once the populations are seeded, the evolution of useful substructures and superstructures proceeds without further intervention.

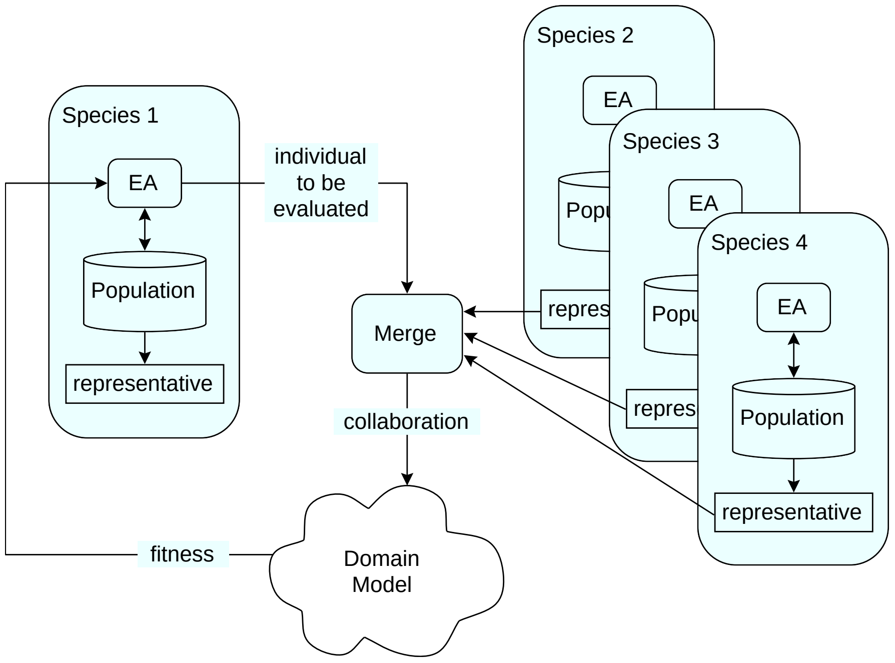

Figure 1: Cooperative coevolutionary architecture from the perspective of species number one

This is accomplished by selecting representatives from each of the EA populations (the species), and combining them into a single composite structure capable of being evaluating on the top level goal task. Credit from evaluating the composite structures flows back to the individual subcomponents reflecting how well they collaborate with the other subcomponents to achieve the top level goal. This credit is then used by the local EAs to evolve better subcomponents. We call such systems *Cooperative Coevolutionary Algorithms* (CCAs).

We have initially tested this CCA architecture in two quite different domains: the well-studied and well-understood domain of function optimization and the domain of learning robot task programs. In both cases we achieved considerable performance improvements over more traditional centralized architectures (Potter and De Jong 1994; Potter, De Jong, and Grefenstette 1995). In the remainder of this paper, we describe the architecture in more detail, summarize these initial case studies, and then discuss ongoing efforts to scale up to more complex problems

2 BASIC COEVOLUTIONARY FRAMEWORK

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. The difficulty comes in finding reasonable computational extensions to our current evolutionary paradigms in which such subcomponents "emerge" rather than being hand designed as in (de Garis 1990). At issue here is how to represent such subcomponents and how to apportion credit to them for their contributions to the problem solving activity such that the evolution of complex structures proceeds without a human in the loop.

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 (Holland and Reitman 1978). Other extensions have been proposed to encourage the emergence of niches and species in a single population (DeJong 1975; Deb and Goldberg 1989; Davidor 1991; Forrest, Javornik, Smith, and Perelson 1993; Giordana, Saitta, and Zini 1994) in which individual niches compete for the allocation of trials.

The use of multiple interacting subpopulations has also been explored as an alternate mechanism for coevolving niches using the so-called *island model* (Grosso 1985; Cohoon, Hegde, Martin, and Richards 1987; Tanese 1989; Whitley and Starkweather 1990). In the island model a fixed number of subpopulations evolve competing rather than cooperating solutions. In addition, individuals occasionally migrate from one subpopulation (island) to another, so there is a mixing of genetic material. The previous work that has looked at cooperating rather than competing subpopulations has involved a userspecified decomposition of the problem into species (see, for example, (Husbands and Mill 1991) or (Hills 1990)).

Cooperative coevolutionary algorithms combine and extend ideas from these earlier systems in several ways. A CCA consists of a collection of independent subpopulations, each attempting to evolve subcomponents (species) which are useful as modules for achieving more complex structures. Unlike the island model, the individuals from the separate subpopulations do not interbreed. Although nothing explicitly prevents multiple species from existing within the same subpopulation (a condition that is likely to occur during the early stages of the evolution of a subpopulation) the existence of distinct subpopulations eliminates haphazard and often destructive recombination between dominate species once niches are established.

Complete solutions are obtained by assembling representatives from each of the species present. Credit assignment at the species level is defined in terms of the fitness of the complete solutions in which the species members participate. This provides evolutionary pressure for species to cooperate rather than compete. However, competition still exists among individuals within the same subpopulation.

In the system used in this paper, the evolution of each species (subpopulation) is handled by a standard GA (although we could just as easily have chosen some other evolutionary algorithm). We emphasize this choice by adopting the more specific terminology CCGA.

3 CASE STUDY 1: FUNCTION OPTIMIZATION

For our first test of these ideas we chose the domain of function optimization. This domain has several advantages for the study of a new coevolutionary architecture. 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 focus on the mechanisms of collaboration and credit assignment, and to defer the more difficult issue of emergent problem decomposition to later studies. In the remainder of this section we briefly summarize the coevolutionary function optimization study. For more detail see (Potter and De Jong 1994).

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. A collaboration is then simply the process of assembling selected members of each species into an N-dimensional vector whose fitness can be computed in the normal fashion.

Local fitness within each species is intended to measure how well members collaborate with other species to produce useful N-dimensional vectors. Since it's clearly infeasible to sample all (or even a large percentage) of the possible collaborations, we initially tested two simple estimates of collaborative effectiveness. The first method assigned local fitness to each member of a species by having each one form a single collaboration with the current best individual from each other species, and used the global fitness measure associated with the N-dimensional vector formed in this manner as the local measure of fitness. This approach significantly outperformed a standard GA when there were relatively few interdependencies between function variables. However, this approach proved overly "greedy" when applied to the optimization of functions with highly interdependent terms.

The second approach we explored was to "soften the greediness" by constructing two collaboration vectors instead of just one. In one vector the collaborators were the current best individuals from the other species as before. The second vector was constructed from a random sample of individuals from the each of the other species. Both vectors were then applied to the target function and the better of the two results was returned as the offspring's fitness. The additional exploration provided by this second approach enabled the CCGA to perform well on functions with highly interdependent terms while only performing slightly worse than the more greedy approach on functions with relatively independent terms.

The interested reader can see (Potter and De Jong 1994) for more detail. We briefly illustrate these effects here via two functions: the Rastrigin function, a highly multimodal function with relatively little interdependency between terms and whose primary characteristic is the existence of many suboptimal peaks whose values increase as the distance from the global optimum point increases (Mühlenbein, Schomisch, and Born 1991; Gordon and Whitley 1993), and the Rosenbrock function from the original De Jong test suite (DeJong 1975), which is a function of two highly interdependent variables and is characterized by an extremely deep parabolic valley that leads to the global minimum.

The results from the Rastrigin optimization experiments are shown in figure 2. The curve labeled "coevolution 1" refers to the greedy collaborator selection approach described above while the curve labeled "coevolution 2" refers to the more exploratory approach. This plot shows that both the greedy and more exploratory approaches significantly outperform the standard GA both in the minimum value found and in the speed of convergence to the minimum. It also shows the slightly better performance of the greedy collaborator selection approach when applied to functions with relatively independent terms.

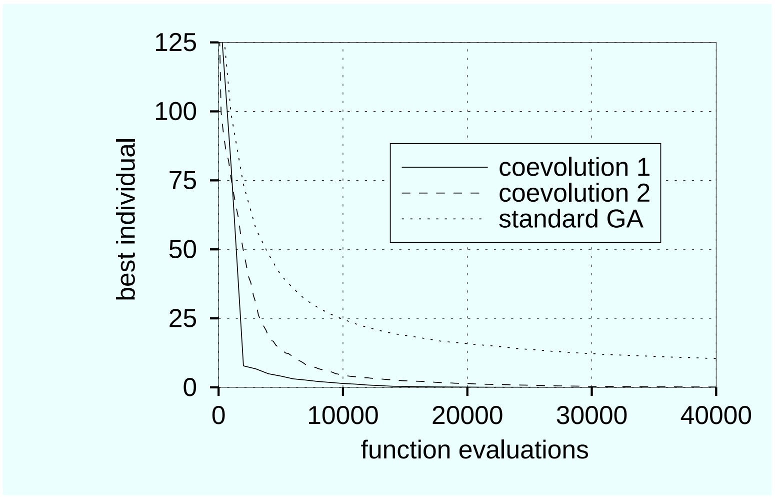

Figure 2: Comparison of CCGA and standard GA optimization rate on Rastrigin function

A similar plot from the Rosenbrock optimization experiments is shown in figure 3. The discovery of the floor of the parabolic valley where the interdependency between terms becomes an issue is clearly visible as the point where the curves abruptly flatten out. The exploratory collaborator selection approach locates this region quickly; but once found, its performance is comparable to the standard GA. The greedy approach does not perform as well on this kind of function.

As a result of the study we felt comfortable with the basic architecture, and ready to test the system in less structured domains.

4 CASE STUDY 2: ROBOT LEARNING

We chose as our second test of the system the domain of learning task programs for robots. This domain does not have generally have natural decompositions like that of the function optimization domain, allowing us to explore issues such as subcomponent formation and stability. To simplify our efforts ans to test the generality of the CCA architecture, we extended the Samuel system to allow multiple instantiations to be run in parallel and communicate with each other for the purpose of forming collaborations. Samuel is designed to evolve sets of sequential

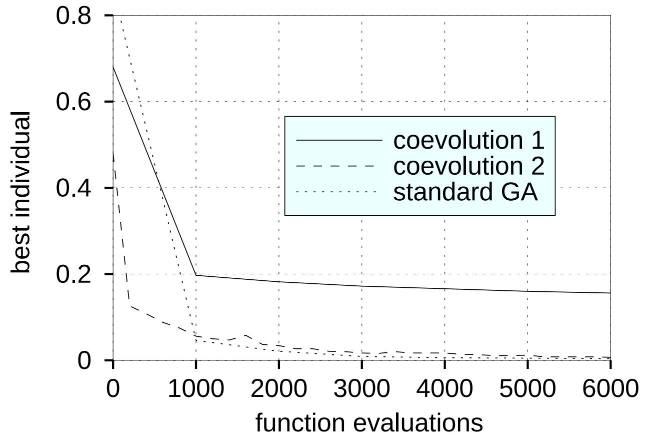

Figure 3: Comparison of CCGA and standard GA optimization rate on Rosenbrock function

decision rules to be used by decision making agents (Grefenstette, Ramsey, and Schultz 1990). The study is described in greater detail in (Potter, De Jong, and Grefenstette 1995).

The architecture of SAMUEL consists of three main components: a domain module, a production system, and a genetic algorithm. The production system and domain module evaluate rule sets while the genetic algorithm evolves new rule sets. The domain module consists of a world model, agents, and an agent critic. The agents consist of sensors and effectors and are controlled by the production system. Pattern matching is done between rule antecedents and agent sensor values. When a rule fires it modifies agent effectors. The GA sends rule sets to be evaluated to the production system which communicates with the agent critic to determine the fitness of the rules. This information is then used by the GA to evolve better rule sets.

Because we were less concerned about nonlinear interactions among rule sets and because we were concerned about the cost of evaluation, we chose to implement the "greedy" form of local evaluation. That is, in coevolutionary SAMUEL, each population makes its best rule set available to the other SAMUEL populations so that collaborations may be formed. This allows rules covering the entire spectrum of required behaviors to be coevolved across multiple GA populations rather than "shaping" behaviors in stages or evolving all behaviors in a single chromosome. More specifically, in order for any of the multiple SAMUEL instantiations to send a rule set to the production system and domain module to be evaluated, the rule set must first be merged with the best rule sets from the other instantiations. Credit from the agent critic flows back to the local rule set reflecting how well it performed with the rule sets from the other instantiations to achieve the top level goal.

We tested coevolutionary SAMUEL on a moderately complex task in which a robot had to maneuver itself around an obstacle free room in which food pellets appeared at random locations and times. The robot had to consume these food pellets to replace lost energy. Energy loss was a function of the speed and turning rate of the robot. Fitness was computed as the average energy level over multiple food gathering episodes. There was a second robot in the environment controlled by a fixed set of hand-crafted rules and which competed for the food pellets. The goal was to evolve over time a set of behaviors (rule sets) which would enable the robot to survive indefinitely in this stochastic, unfriendly environment.

SAMUEL provides a simple mechanism for initially seeding each GA population with user-supplied rules. We used that facility to provide each of the GA populations with an initial bias to develop an area of expertise different from the others. However, once evolution begins there is nothing to prevent the roles of species from changing considerably. This enabled us to study the stability of the cooperating behaviors (niches) that were formed. The particular biases we chose for this study was to evolve behavior related to food being absent and food being present.

We investigated the stability of the cooperating behaviors by running coevolutionary SAMUEL until highly fit rule sets were evolved and inspecting the best rule set from the final generation. This was repeated on five separate runs. We found that most of the rules from the final generation continued to exhibit the behavior their species was biased toward, that is, the rules in the species biased toward behavior related to food being absent would mostly fire when there was no food and the rules in the species biased toward behavior related to food being present would mostly fire when food pellets appeared.

We also discovered the appearance of a third class of behavior coexisting within the population biased toward food being absent. This unexpected behavior can be summarized as “if you are not hungry then you should not waste energy seeking food”.

We were also interested in comparing learning rates between the coevolutionary and standard SAMUEL systems. Figure 4 illustrates the typical speedups achieved by the coevolutionary approach.

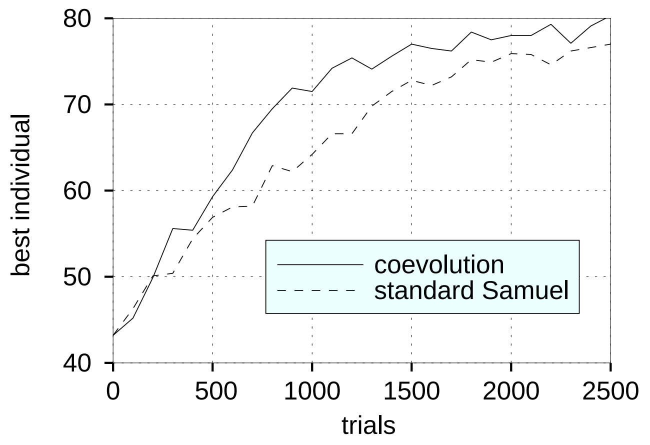

Figure 4: Comparison of coevolutionary Samuel and standard Samuel learning rate on robot domain

5 DISCUSSION AND CONCLUSIONS

Although our motivation for developing the CCA architecture was driven by the need for an effective means to evolve complex structures, the CCA architecture appears to have an interesting potential for speeding up more traditional domains such as function optimization. Our intuition here is that process of forming collaborations places context-sensitive constraints on the search process which, if not too 'greedy" can be quite effective in speeding up the search process. However, more work need to be done to understand better the dynamics of cooperating populations.

As noted earlier, we are more interested in the potential for evolving more complex structures. To that end, we are currently using the robot learning domain to test our ideas at the next level of difficulty: one which requires the dynamic creation and destruction of species, representing the birth and death of behavioral niches with the goal of evolving complex robot task programs from these coevolving and cooperating lower level behaviors. We hope to be able to report these results in the near future.

Acknowledgments

This work was supported in part by the George Mason University Department of Computer Science and by the Navy Center for Applied Research in Artificial Intelligence. We also wish to acknowledge contributions made by John Grefenstette and Bob Daley to the SAMUEL case study.

References

Cohoon, J., S. Hegde, W. Martin, and D. Richards (1987). Punctuated equilibria: A parallel genetic algorithm. In J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 148–154. Lawrence Erlbaum Associates.

Davidor, Y. (1991). A naturally occuring niche & species phenomenon: The model and first results. In R. Belew and L. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 257–263. Morgan Kaufmann.

de Garis, H. (1990). Building artificial nervous systems using genetically programmed neural network modules. In B. Porter and R. Mooney (Eds.), *Proceedings of the Seventh International Conference on Machine Learning*, pp. 132–139.

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

DeJong, K. (1975). *Analysis of Behavior of a Class of Genetic Adaptive Systems.* Ph. D. thesis, University of Michigan, Ann Arbor, MI.

Forrest, S., B. Javornik, R. Smith, and A. Perelson (1993). Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation 1*(3), 191–211.

Giordana, A., L. Saitta, and F. Zini (1994). Learning disjunctive concepts by means of genetic algorithms. In W. Cohen and H. Hirsh (Eds.), *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 96–104. Morgan Kaufmann.

Gordon, V. and D. Whitley (1993). Serial and parallel genetic algorithms as function optimizers. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 177–183. Morgan Kaufmann.

Grefenstette, J., C. Ramsey, and A. Schultz (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning 5*(4), 355–381.

Grosso, P. (1985). *Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model*. Ph. D. thesis, University of Michigan, Ann Arbor, MI.

Hills, D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. In C. Langton, C. Taylor, J. Farmer, and S. Rasmussen (Eds.), *Artificial Life II*, pp. 313–324. Addison-Wesley.

Holland, J. and J. Reitman (1978). Cognitive systems based on adaptive algorithms. In D. Waterman and F. Hayes-Roth (Eds.), *Pattern-Directed Inference Systems*. Academic Press.

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

Mühlenbein, H., M. Schomisch, and J. Born (1991). The parallel genetic algorithm as function optimizer. In R. Belew and L. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 271–278. Morgan Kaufmann.

Potter, M. and K. De Jong (1994). A cooperative coevolutionary approach to function optimization. In Y. Davidor and S. H.-P. (Eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pp. 249–257. Springer-Verlag.

Potter, M., K. De Jong, and J. Grefenstette (1995). A coevolutionary approach to learning sequential decision rules. Technical Report AIC-95-010, Navy Center for Applied Research in Artificial Intelligence, Washington DC.

Tanese, R. (1989). Distributed genetic algorithms. In J. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 434–439. Morgan Kaufmann.

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