Chapter 8

COEVOLVING FITNESS MODELS FOR ACCELERATING EVOLUTION AND REDUCING EVALUATIONS

Michael D. Schmidt[1] and Hod Lipson[2]

[1] *Computer Science, Cornell University.*
[2] *Mechanical & Aerospace Engineering, Cornell University.*

Abstract Fitness models are used to reduce evaluation frequency and cost. There are three fundamental challenges faced when using fitness models: (1) the model learning investment, (2) the model level of approximation, and (3) the lack of convergence to optima. We propose a coevolutionary algorithm to resolve these problems automatically during evolution. We discuss applications of this approach and measure its impact in symbolic regression. Results show coevolution yields significant improvement in performance over other algorithms and different fitness modeling approaches. Finally we apply coevolution to interactive evolution of pen stroke drawings where no true fitness function is known. These results demonstrate coevolution's ability to infer a fitness landscape of a user's preference while minimizing user interaction.

Keywords: Coevolution, fitness modeling, symbolic regression

1. Introduction

Fitness modeling is the technique of replacing the fitness evaluation in evolutionary algorithms with an approximate calculation. A fitness model is often used to reduce the computational cost of evolution e.g. (Jin et al., 2001; Luke and Wiegand, 2002) where the exact fitness must be measured by an expensive physical simulation or experimentation. Fitness modeling has other advantages: The approximation can smooth rugged fitness landscapes, map discrete fitness values to continuous values, and diversify populations through ambiguity.

Recent work in fitness modeling has focused on approximation methods, model incorporation, and the management/control of model evaluation (Jin,

2005). This research has made significant strides in accelerating evolution and applying evolutionary computation to domains that were formerly infeasible.

There are three fundamental difficulties that remain to be addressed when utilizing fitness modeling:

1 **Model Training:** Often great computational effort is required ahead of time to train the desired fitness model.

2 **Level of approximation:** It is often unclear what (possibly adaptive) level of approximation to use for efficiency while still maintaining evolutionary progress.

3 **Convergence to optima:** Due to different levels of approximation, the global maximum of the fitness model may not correspond to the global maximum of the exact fitness.

The goal of this paper is to address these issues through coevolution. In the general framework, there are three populations: (1) solutions to the original problem, (2) fitness models of the problem, and (3) trainers, used to improve the models. Solutions are evolved to maximize their modeled fitness using a model from the model population. Fitness models are evolved to minimize predicted fitness error of select individuals from of the solution population. Both populations start with random solutions and random fitness models which evolve in parallel, using trainers.

We introduce a coevolutionary algorithm based on this general framework and discuss its application in example domains. We then apply it to a benchmark problem in genetic programming, symbolic regression, to measure its impact. We compare performance with three other symbolic regression fitness model algorithms on various problem difficulties. We duplicate experiments in recent symbolic regression publications and compare results. Finally, we apply coevolution of fitness models to human interactive evolution of pen-stroke drawings where no fitness function exists to show how models can be coevolved to infer a fitness landscape.

2. Preliminaries

Coevolution

Coevolution holds the potential of perpetually refining solutions without the need to manually restate the problem. As one individual improves, newer fitness objectives are imposed on another, and vice versa. Ideally, the separate individuals or populations continually encourage each other to improve. In general, coevolution occurs when the fitness of an individual is defined by another individual. More formally, in a coevolutionary process, one individual

can determine the relative ranking between two other individuals (in the same or in a separate population) (Hillis, 1992; Ong et al., 2003).

Much research has been done on the use and application of coevolution to enhance problem solving (Jong and Pollack, 2004; Ficici, 2004; Ficici and Pollack, 2001; Potter and De Jong, 2000; Rosin and Belew, 1996; Stanley and Miikkulainen, 2004; Zykov et al., 2005), with the main goal of controlling coevolutionary dynamics that often result in lack of progress or progress in unanticipated directions (Cliff and Miller, 1995; Luke and Wiegand, 2002; Watson and Pollack, 2001). Here we use a specific form of coevolution which addresses many of these challenges (Bongard and Lipson, 2004; Bongard and Lipson, 2005).

Fitness Modeling

Fitness modeling has become an active area in evolutionary computation with many varying approaches and results (Jin, 2005). Here we discuss the motivations, methods, and challenges tied to fitness modeling.

Methods

The technique of fitness modeling falls most naturally in the field of machine learning. Depending on the structure of solution encodings, many different machine learning approaches such as neural nets, support vector machines, decision trees, Bayesian networks, k-nearest-neighbor, and polynomial regression can be learned to map individuals to approximate fitnesses very efficiently. Modern approaches utilize boosting, bagging, and ensembles learning to produce very accurate models. A major drawback however is it is often unclear which machine learning approach will work best (Jin, 2005).

Sub-sampling of training data is also a very common way to reduce the cost of fitness evaluation (Albert and Goldberg, 2002). In many problems, fitness is calculated by evaluating individuals on random training cases and combining the total error. With a sub-sample, only a fraction of the training data is evaluated.

Evolutionary specific fitness modeling methods include fitness inheritance and fitness imitation. In fitness inheritance (Chen et al., 2002), fitness values are transferred from parents to child during crossover (similar to parent passing on a legacy or education). In fitness imitation (Jin and Sendhoff, 2004), individuals are clustered into groups based on a distance metric. The fitness of the central individual of each cluster is then evaluated in full and assigned to all individuals in that cluster.

Once a fitness model has been learned, there are many ways to incorporate it into evolution. It can be used simply to initialize the population, guide crossover and mutation, or replace (some) fitness evaluations (Jin, 2005). For this paper

however, we focus only on replacing actual fitness evaluations with the fitness model.

Challenges

The use of an approximate fitness model comes at a cost and potentially unacceptable consequences.

Training the model: Fitness models like neural nets, SVMs, and Gaussian processes require significant overhead to train. When advanced methods like bagging, boosting, and ensemble methods are used, this investment becomes significantly larger. In addition, a significant amount of exact fitnesses must be calculated for training and validation data to effectively learn any type of model ahead of time.

By using coevolution, we can learn these models in parallel with the problem solutions. As seen in (Yang and Flockton, 1995), early stages of evolution only require coarse fitness models. As the solution population progresses, so do the fitness models. In this fashion, coevolution encompasses automatic coarseness adjustment without the need to train several different approximations before hand.

Level of approximation: How powerful must the fitness model be to facilitate progress throughout evolution? If a single fitness model is used, it likely must be very complex in order to model all possible solutions in the fitness landscape.

When fitness models are coevolved, the models can be optimized for only the individuals in the current population. The models do not need to encapsulate the entire landscape, only a subset, so the method chosen can be significantly less complex.

Convergence to optima: Often the global maximum of the fitness model is not the same as that of the exact fitness function. In this case, the model was either too coarse or the training data given was insufficient to identify the true maxima. The consequence is convergence to local maxima and suboptimal solutions.

By using coevolution, we can intelligently select individuals for training data (active learning). As the solution population evolves, training data for the fitness models can focus heavily on recent solutions which have high predicted fitness. Therefore, optimizing fitness models to the current population can also optimize the models to promising areas of the fitness landscape refining true fitness maxima.

Symbolic Regression

Symbolic Regression (SR) is the challenging problem of identifying the exact mathematical description of a hidden system from experimental data (Augusto

and Barbosa, 2000; Duffy and Engle-Warnick, 2002). Unlike polynomial regression or machine learning methods which minimize error, SR is a system identification method. SR typically has strong advantages in resisting overfitting, extrapolating future points, and providing descriptive models at the cost of greater complexity.

For experiments in this paper, we represent functional expressions as a binary tree of primitive operations (Duffy and Engle-Warnick, 2002; Ferreira, 2003; Keijzer, 2003; Soule and Heckendorn, 2001). The operations can be unary operations such as abs, exp, and log, or binary operations such as add, mult, and div. If some a priori knowledge of the problem is known, the types of operations available can be chosen ahead of time (Augusto and Barbosa, 2000; Dolin et al., 2002; Duffy and Engle-Warnick, 2002; Eggermont and van Hemert, 2001; Hoai et al., 2002; Koza, 1992). The terminal values available consist of the function's input variables and the function's evolved constant values (Ferreira, 2003).

The fitness objective of a SR solution is to minimize error on the training set (Augusto and Barbosa, 2000; Koza, 1992). There are many ways to measure the error such as squared error, absolute error, log error, etc. Though the choice is not critical to the optimal solution, different metrics work better on different problems. For experiments in this paper, we simply use the mean absolute error for fitness measurement.

We generate training data by randomly sampling from the hidden function. The training set is used for fitness evaluation and convergence testing. A large randomly sampled test set is held out for final solution performance evaluation.

3. Coevolved Fitness Models

In this section we give a simple framework before presenting our implementation. A conventional evolutionary algorithm can be viewed as an optimization function with objective:

$$s^* = arg\ max\ fitness(s) \qquad s \varepsilon S \tag{8.1}$$

where $s \varepsilon S$ is an individual solution to the problem and *fitness*(s) is the exact computed fitness of solution s. Here we are interested in replacing all fitness evaluations with a fitness model. In this instance, the objective becomes:

$$s^* = arg\ max\ m(s) \qquad s \varepsilon S \tag{8.2}$$

where $m \varepsilon M$ is the fitness model used. For coevolution, both solutions and models are being evolved. There are thus two objectives:

$$s_t^* = arg\ max\ m_t(s) \qquad s \varepsilon S \tag{8.3}$$

$$m_t^* = arg\ min |fitness(S_t) - m(S_t)| \qquad m \varepsilon M \tag{8.4}$$

where the t subscripts denote time dependence. The solution population evolves to maximize the fitness of the current fitness model. The fitness model population evolves minimize the difference between exact and modeled fitnesses of the current population.

Algorithm

Overview

The algorithm presented in this paper evolves two populations simultaneously: the individuals and their fitness models. This section outlines the basics needed to implement this coevolutionary approach. A high-level algorithm overview is given in Figure 8-1.

At the start, both solutions and fitness models are randomly initialized. The algorithm then chooses an individual from the solution population to measure its exact fitness for use in training the fitness models (more on this in next section). The algorithm then evolves the solution population using the most fit fitness model of the model population, and evolves the models using the known exact fitnesses. Finally, the highest modeled fitness individual is tested for convergence, and the algorithm completes if successful.

Evaluating an Exact Fitnesses

The objective of this step is to select an individual from the solution population that will improve the fitness models' ability to model the current solutions the most. Therefore, we want to choose the individual with the least certainty in modeled fitness. To do this efficiently, we select the individual that has the highest variance in modeled fitness among models in the model population such as done in (Bongard and Lipson, 2004).

A key goal of the fitness models is to optimize to the current population. In many model types, it is often beneficial to "forget" past solution fitness information to do this. For our SR experiments, we discard the exact fitness calculations that are older than 1000 generations.

Evolving the Populations

In the framework, we evolve both populations until a new exact fitness calculation can be done. Alternatively, the populations could be evolved until they plateau. For our experiments however, we simply evolve at regular intervals of 100 generations.

Although the two populations can be evolved in parallel, we manage the effort spent in each for performance measurement. To do this, we throttle back evolution on either population based on the total number of training example evaluations to maintain some ratio. Figure 8-2 (left) and 8-2 (right) show the

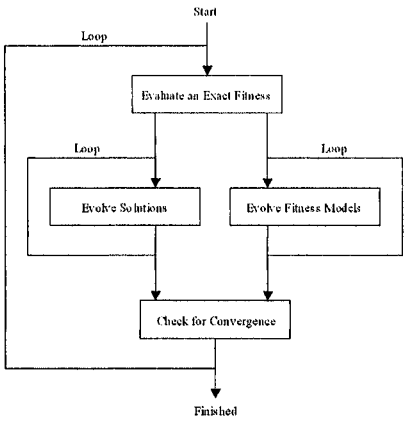


Figure 8-1. The high-level overview of the coevolution of solutions and fitness models algorithm

impact this ratio has on performance on the function $f_2(x)$ experimented on later in the paper.

The choice of the ratio is surprisingly tolerant as seen in Figure 8-2 (left). Ratios in the range 5% to 30% of effort spent training the fitness model population all yield approximately optimal convergence time. If extremely little effort is given to fitness models, performance suffers greatly since the models never adapt.

If an extremely high amount of effort is spent the fitness model population plateaus and makes no further improvement even with the extra effort. The ratio also has a generally weak impact on the number of generations before convergence as seen in Figure 8-2 (right). Over-training the fitness models does not reduce the number of generations required to find the hidden function.

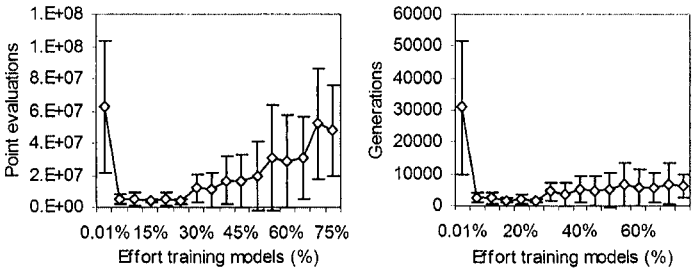


Figure 8-2. Left: The expected number of point evaluations before convergence versus the effort (percent of point evaluations) while training the fitness models. Right: The expected number of generations before convergence versus the effort (percent of point evaluations) while training the fitness models

Note that in practice, the ratio is not vital to the algorithm's performance. Each population could be evolved in parallel provided appropriate hardware. For experimentation however, we want to show that not only can we parallel out some computation, but that overall effort as well can be reduced through coevolution. In our SR experiments, we use 5% which is near the minimum while still enhancing performance.

Convergence Test

The convergence test determines when the algorithm terminates based on the solution with the highest modeled fitness. For symbolic regression, we define convergence as having near zero error on the training set (less than epsilon). If the solution has not converged, a new exact fitness is calculated and the algorithm continues. The final performance test however measures the fitness using a held out test set.

4. Training Data Sample Fitness Model

In our SR experimentation, we use a general fitness model which is simply a small subset of the training data. So the total error is measured by evaluating functions on only a few training examples. First we examine the importance of the number of samples in the training subset fitness model.

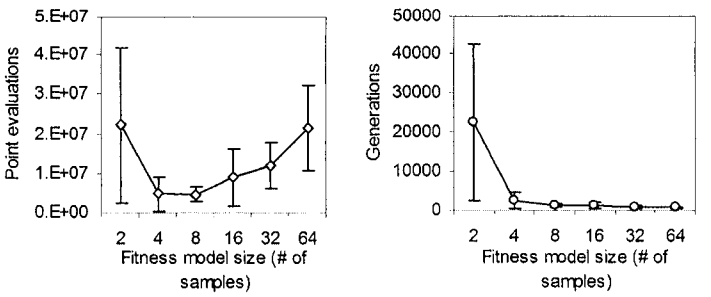


Figure 8-3. Left: The expected point evaluations before convergence versus the number of samples in the fitness model. Error bars show the standard deviation. Right: The expected generations before convergence versus the number of samples in the fitness model. Error bars show the standard deviation.

The sample size choice is fairly tolerant in as seen in Figure 8-3 (left). Sample sizes of four through eight have very similar performance. Even when using only two samples in the subset performance is acceptable.

When the fitness model only has two samples, fitness evaluations are extremely light-weight but the evolution requires many more generations as evi-

dent in Figure 8-3 (right). The larger subsets are sufficiently large for accurate modeling and do not reduce the number of generations greatly.

In our SR experiments, we use an 8-sample subset for all experiments. Although it may not be the optimal choice for all target functions, these results suggest it will not have a dramatic impact on final performance unless the target function is extremely simple or extremely complex.

5. Experiments in Symbolic Regression

Here we use symbolic regression (SR) as an example genetic programming problem for experimentation. For each run, all parameters are held constant. We use a solution population size of 128, a fitness model population size of 8. For both populations we use deterministic crowding selection, 0.1 mutation probability, and 0.5 crossover probability. The operator set is $+, -*, /, exp, log, sin, cos$ and the terminal set consists of the input variable and one evolved constant.

Fitness Model Experiments

We compare four fitness model algorithms in symbolic regression (SR). All fitness models are 8-sample subsets of the training set except for the exact fitness algorithm which evaluates fitness on the full training set. The differences in these algorithms are summarized in Table 8-1.

Table 8-1. Summary of the Compared Algorithms

Fitness model	Sample Size	Sample Selection
Coevolved Fitness Model	8	Evolved Subset
Static Fitness Model	8	Random subset chosen at runtime
Dynamic Fitness Model	8	Changing random subset
Exact Fitness	200	Use all training data

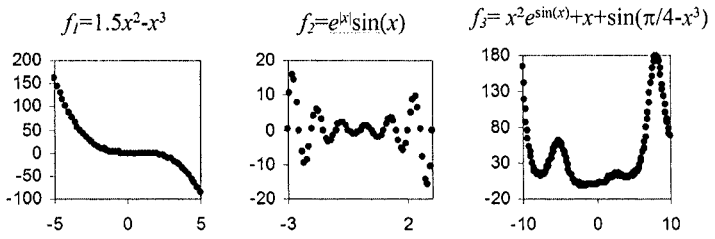


Figure 8-4. The training data of the three target functions experimented on.

We test on three different target functions of varying difficulty shown in Figure 8-4. Each test is repeated 50 times, and the average fitness for each

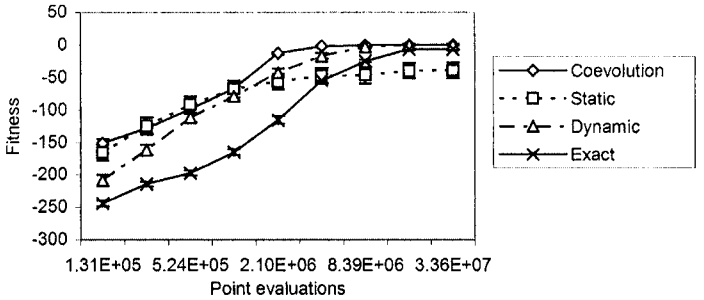


Figure 8-5. The test set fitness during evolution for target function $f_1(x)$. Results are averaged over 50 test runs. Error bars show the standard error.

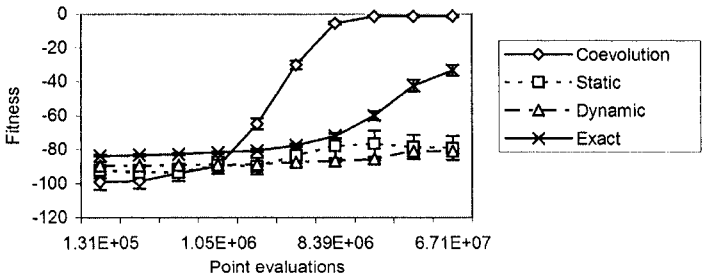


Figure 8-6. The test set fitness during evolution for target function $f_2(x)$. Results are averaged over 50 test runs. Error bars show the standard error.

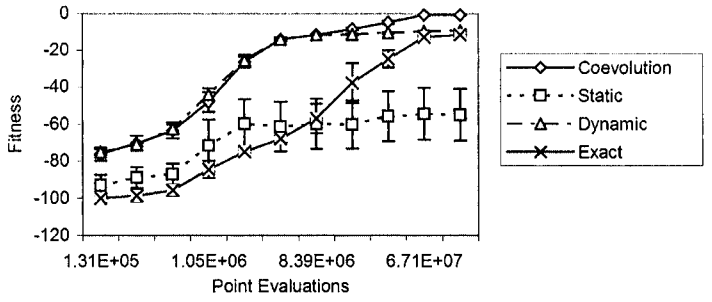


Figure 8-7. The test set fitness during evolution for target function $f_3(x)$. Results are averaged over 50 test runs. Error bars show the standard error.

run is recorded over time. The training and test data are generated by random sampling each target function uniformly.

The performances on these three functions for each algorithm are shown versus the number of point evaluations in Figures 8-5 – 8-7.

In these experiments, we see the coevolution algorithm converges first and maintains higher fitness throughout evolution. On the simple function $f_1(x)$ the difference in performance is small. On the non-linear functions $f_2(x)$ and $f_3(x)$, the difference is more pronounced.

The fitness curves of the coevolution and exact fitness algorithms are very similar in shape. The exact fitness algorithm is delayed and elongated by extra point evaluations but endures similar average evolutionary paths.

The static and dynamic random training subset fitness model algorithms are less predictable in performance as is shown by their higher standard errors and ranks on the differing functions. The static model algorithm appears to suffer if the sample is not very informative by chance or if the target function is too complex to adequately represent with eight samples. The dynamic model appears to perform well on finding simple features but poorly on detail and high-frequency terms.

Comparison to Previously Published Methods

In this section, we compare the coevolution algorithm with four recently published symbolic regression techniques: Stepwise Adaptive Weights (SAW), Grammar Guided Genetic Programming (GGGP), Tree-Adjunct Grammar Guided Genetic Programming (TAG3P), and Coevoltuion with Tractable Shared Fitness (Dolin et al., 2002; Eggermont and van Hemert, 2001; Hoai et al., 2002). Where possible, we compare with their best reported results and match performance metrics as closely as possible.

We compare performance based on point evaluations. The coevolution algorithm is stopped based on the number of function evaluations the compared algorithm made during each experiment. Since each compared algorithm differs in fitness calculation, we assume that each individual's fitness is measured by evaluating it on the training set each generation. Likewise, we force the coevolution algorithm to calculate fitness for every individual every generation also, even though different selection algorithms do not require it.

Many of these experiments are on simple functions but are stopped at a very low number of point evaluations. So, finding the target function quickly is the highest priority. The cosine identity and the Gaussian function experiments are noticeably more challenging to regress.

Significant improvements in Table 8-2 are shown in bold text. The coevolution algorithm has significantly higher convergence than the PSAW and GGGP algorithms. The TAG3P algorithm performs significantly higher in the polynomial functions. Coevolution makes a large improvement in convergence in the cosine double angle identity experiment however.

Table 8-2. Research Comparison Summary

Algorithm	Target Function	Metric	Results	Coevolved Models
PSAW	$F(x) = x^5 - 2x^3 + x$	$Convergence^{\dagger}$	85.9%	**93.9%**
	$f(x) = x^6 - 2x^4 + x^2$	$Convergence^{\dagger}$	81.8%	**86.9%**
GGGP	P2, P3, P4, P5*	$Convergence^{\dagger\dagger}$	92%, 64%	**100%, 86%**
	$f(x) = cos(2x)$**	$Convergence^{\dagger\dagger}$	20%	**76%**
TAG3P	P2, P3, P4, P5*	$Convergence^{\dagger\dagger}$	100%, 100%, **96%, 84%**	100%, 86% 62%, 52%
	$f(x) = cos(2x)$**	$Convergence^{\dagger\dagger}$	36%	**76%**
Coevolved Tractable	Gaussian	$Evaluations^{\dagger\dagger\dagger}$	3.384e7	**2.107e7**

* P2, P3, etc. refer to polynomials $(x^3 + x^2 + x, x^4 + x^3 + x^2 + x, x^5 + x^4 + x^3 + x^2 + x$, etc.
** The operator set does not include the cos function, a trigonometric identity must be found
†The percent of successful convergences from 100 test runs
††The percent of successful convergences from 50 test runs
†††The maximum number of evaluations before convergence for 100 test runs

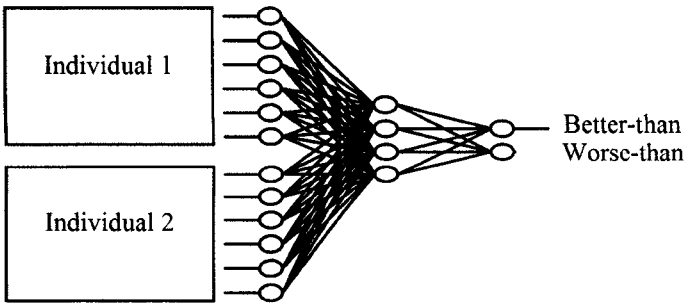


Figure 8-8. The neural network comparator fitness model.

Evolving Pen Stroke Drawings

In this experiment we evolve drawings produced by a series of closed pen strokes. Each individual encodes each coordinate drawn to on a 32 by 32 pixel image. Drawings are composed of six pen strokes. Here we evaluate the algorithm's ability to identify a preference for star-shaped drawings. We compare results with random search and local search techniques.

Random Search Comparison

We split the drawing space into six regions to approximate the number of possible star drawings. Note that a five point star is uniquely determined by

three of its vertices. The probability,P_{star}, and expected time to encounter a random star shape, T_{star}, is calculated:

$$P_{star} \approx \frac{5}{3888} \approx 0.128 \tag{8.5}$$

and

$$\therefore E(T_{star}) \approx 777.6 \tag{8.6}$$

Perfect Local Search Comparison

In the local search algorithm, the user is given a random individual and asked to fine tune parameters individually. The variation we compare with can be thought of as a perfect local search where the user is assumed to make optimal choices to transform the drawing into a star. The optimal expected number of prompts required is lower-bounded by:

$$< steps >= \frac{1}{9} * \sum_{x=0}^{8} x + |\sqrt{8^2 - x^2}| \approx 9 \tag{8.7}$$

Empirical Results

Figure 8-9 shows the median run out of ten to evolve an approximate star shaped drawing. The user has a consistent strategy when answering comparison prompts to prefer shapes with multiple sharp corners such as a star shape has. The user's preferred response for each comparison is shown in bold.

A well formed star is evolved as the top modeled shape after five user prompts. No prior prompts required a star shape. The next three prompts are given to show the fitness model is stable and continues to favor the consistent star-like shapes. Figure 8-10 compares the expected user prompts required by each algorithm.

Inferring the Fitness Landscape

In this experiment visualize the fitness landscape learned by the fitness models. We evolve single pen-stroke drawings appear as a clock hour hand. The user is then asked to prefer two times where the hand points to either 3:00 or 7:00. The target solution therefore has two optimal solutions.

To visualize the fitness landscape in this experiment we sum the comparisons between all possible drawings. Figure 8-11 shows four fitness landscapes of the fitness models over six user prompts. At zero prompts, the fitness landscape is flat but becomes more distinct with user responses.

The fitnesses on the xz-plan resemble an arch which peaks at approximately the target 3:00 position. The yz-plane fitnesses exhibit the same behavior for the 7:00 position. This final resulting landscape is a surprisingly friendly terrain

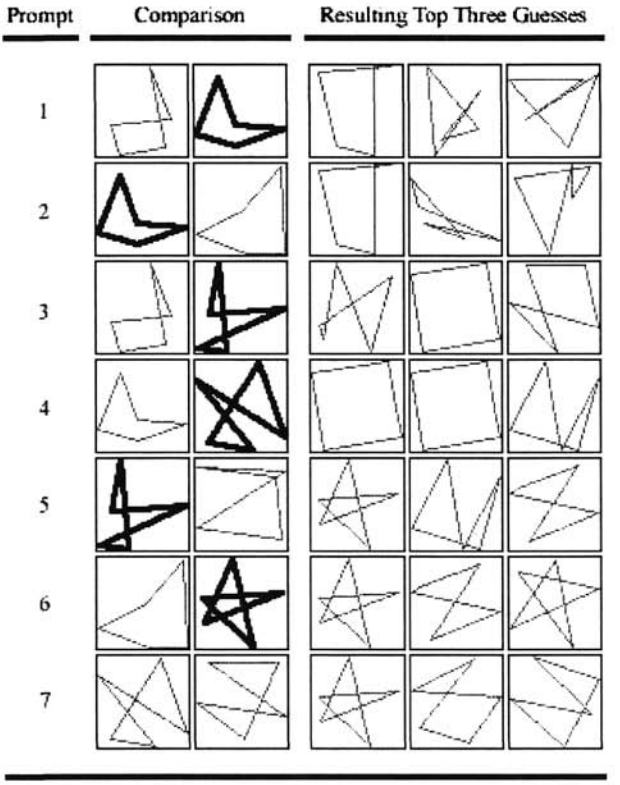


Figure 8-9. The user prompts and the resulting top three drawings evolved. User choices are shown in bold.

for the evolutionary search. Very gradual gradients exist near the target clock hand locations that should be easily descended.

6. Conclusion

We have addressed three fundamental challenges faced when using fitness models in evolutionary algorithms: (1) the model training investment, (2) choosing a level of approximation, and (3) lack of convergence to global maxima.

We have introduced a coevolutionary approach where fitness models are evolved in parallel with original problem solutions. In this framework, the fitness models do not require prior training investment, can optimize to the current population, and can actively select the most informative solutions to evaluate in full for model training.

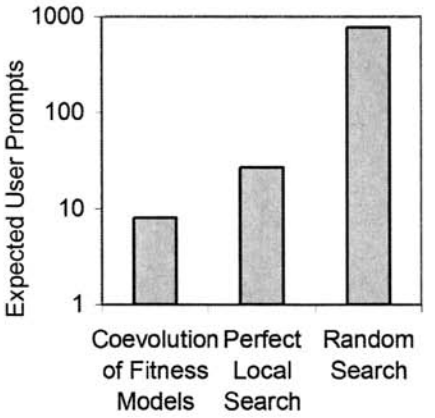


Figure 8-10. The number of user prompts expected between the compared algorithms to find the target star shape

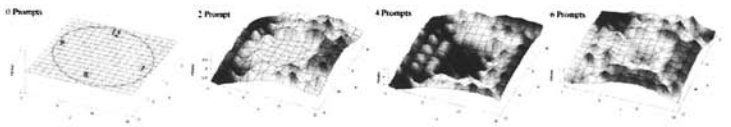


Figure 8-11. The clock time fitness landscapes learned over six user prompts.

We applied this approach in symbolic regression to measure its impact. Results show a significant improvement in both fitness and convergence rates over other fitness model and exact fitness algorithms. In the research comparison, the coevolution algorithm significantly outperforms PSAW, GGGP, and coevolved tractable fitness. The coevolution algorithm is outperformed by TAG3P on polynomial functions but makes a large improvement in the more difficult problem of deriving a trigonometric identity.

Finally, we applied the coevolution of fitness models to human interactive evolution where no fitness function is defined. Results show the fitness models are able to infer the user preference in pen-stroke drawings through very limited user feedback. Fitness models converged on trends in the user responses to identify the target drawing which was never explicitly presented to the user.

References

Albert, Laura A. and Goldberg, David E. (2002). Efficient discretization scheduling in multiple dimensions. In Langdon, W. B., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G.,

Wegener, J., Bull, L., Potter, M. A., Schultz, A. C., Miller, J. F., Burke, E., and Jonoska, N., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 271–278, New York. Morgan Kaufmann Publishers.

Augusto, D.A. and Barbosa, H.J.C. (2000). Symbolic regression via genetic programming. In *VI Brazilian Symposium on Neural Networks (SBRN'00)*, volume 1.

Bongard, J.C. and Lipson, H. (2004). Nonlinear system identification using co-evolution of models and tests. In *IEEE Transactions on Evolutionary Computation*.

Bongard, Josh C. and Lipson, Hod (2005). 'managed challenge' alleviates disengagement in co-evolutionary system identification. In Beyer, Hans-Georg, O'Reilly, Una-May, Arnold, Dirk V., Banzhaf, Wolfgang, Blum, Christian, Bonabeau, Eric W., Cantu-Paz, Erick, Dasgupta, Dipankar, Deb, Kalyanmoy, Foster, James A., de Jong, Edwin D., Lipson, Hod, Llora, Xavier, Mancoridis, Spiros, Pelikan, Martin, Raidl, Guenther R., Soule, Terence, Tyrrell, Andy M., Watson, Jean-Paul, and Zitzler, Eckart, editors, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 1, pages 531–538, Washington DC, USA. ACM Press.

Chen, Jian-Hung, Goldberg, David E., Ho, Shinn-Ying, and Sastry, Kumara (2002). Fitness inheritance in multi-objective optimization. In Langdon, W. B., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M. A., Schultz, A. C., Miller, J. F., Burke, E., and Jonoska, N., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 319–326, New York. Morgan Kaufmann Publishers.

Cliff, Dave and Miller, Geoffrey F. (1995). Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In Morán, F., Moreno, A., Merelo, J. J., and Chacón, P., editors, *Proceedings of the Third European Conference on Artificial Life : Advances in Artificial Life*, volume 929 of *LNAI*, pages 200–218, Berlin. Springer Verlag.

Dolin, B., Bennett, F., and Rieffel, E. (2002). Co-evolving an effective fitness sample: Experiments in symbolic regression and distributed robot control. In *Proceedings of the 2002 ACM Symposium on Applied Computing*.

Duffy, J. and Engle-Warnick, J. (2002). Using symbolic regression to infer strategies from experimental data. In Chen, S.H, editor, *Evolutionary Computation in Economics and Finance*. Physica-Verlag, New York.

Eggermont, J. and van Hemert, J.I. (2001). Stepwise adaptation of weights for symbolic regression with genetic programming. In *Proceedings of EuroGP*.

Ferreira, C. (2003). Function finding and the creation of numerical constants in gene experssion programming. In *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 257–266. Springer-Verlag.

Ficici, S.G. (2004). *Solution Concepts in Coevolutionary Algorithms*. PhD thesis, Computer Science Dept., Brandeis University.

Ficici, S.G. and Pollack, J.B. (2001). Pareto optimality in coevolutionary learning. In *Advances in Artificial Life, 6th European Conference (ECAL2001)*, pages 316–327.

Hillis, W. Daniel (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In Langton, Christopher G., Taylor, Charles, Farmer, J. Doyne, and Rasmussen, Steen, editors, *Artificial life II*, volume 10 of *Sante Fe Institute Studies in the Sciences of Complexity*, pages 313–324. Addison-Wesley, Redwood City, Calif.

Hoai, N. X., McKay, R. I., Essam, D., and Chau, R. (2002). Solving the symbolic regression problem with tree-adjunct grammar guided genetic programming: The comparative results. In Fogel, David B., El-Sharkawi, Mohamed A., Yao, Xin, Greenwood, Garry, Iba, Hitoshi, Marrow, Paul, and Shackleton, Mark, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1326–1331. IEEE Press.

Jin, Y. (Jan 2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing - A fusion of Foundations, Methodologies and Applications*, 9(1):3.

Jin, Yaochu, Olhofer, Markus, and Sendhoff, Bernhard (2001). Managing approximate models in evolutionary aerodynamic design optimization. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 592–599, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea. IEEE Press.

Jin, Yaochu and Sendhoff, Bernhard (2004). Reducing fitness evaluations using clustering techniques and neural network ensembles. In Deb, Kalyanmoy, Poli, Riccardo, Banzhaf, Wolfgang, Beyer, Hans-Georg, Burke, Edmund, Darwen, Paul, Dasgupta, Dipankar, Floreano, Dario, Foster, James, Harman, Mark, Holland, Owen, Lanzi, Pier Luca, Spector, Lee, Tettamanzi, Andrea, Thierens, Dirk, and Tyrrell, Andy, editors, *Genetic and Evolutionary Computation – GECCO-2004, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 688–699, Seattle, WA, USA. Springer-Verlag.

Jong, Edwin D. De and Pollack, Jordan B. (2004). Ideal Evaluation from Coevolution. *Evolutionary Computation*, 12(2):159–192.

Keijzer, M. (2003). Improving symbolic regression with interval arithmetic and linear scaling. In *Sixth European Conference on Genetic Programming*, pages 70–82, Essex, UK. Springer.

Koza, J. R. (1992). *Genetic Programming: On the programming of computers by means of natural selection*. Bradford Books, Cambridge, MA.

Luke, Sean and Wiegand, R. Paul (2002). When coevolutionary algorithms exhibit evolutionary dynamics. In Barry, Alwyn M., editor, *GECCO 2002:*

Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference, pages 236–241, New York. AAAI.

Ong, Y., Nair, P., and Keane, A. (2003). evolutionary optimization of computationally expensive problems via surrogate modeling. *AJAA J.*, 41(4):678–696.

Potter, Mitchell A. and De Jong, Kenneth A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29.

Rosin, Christopher D. and Belew, Richard K. (1996). New methods for competitive coevolution. Technical Report CS96-491, Cognitive Computer Science Research Group, University of California, San Diego, CA. to appear in Evolutionary Computation 5:1.

Soule, Terence and Heckendorn, Robert B. (2001). Function sets in genetic programming. In Spector, Lee, Goodman, Erik D., Wu, Annie, Langdon, W. B., Voigt, Hans-Michael, Gen, Mitsuo, Sen, Sandip, Dorigo, Marco, Pezeshk, Shahram, Garzon, Max H., and Burke, Edmund, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, page 190, San Francisco, California, USA. Morgan Kaufmann.

Stanley, K. O. and Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100.

Watson, Richard A. and Pollack, Jordan B. (2001). Coevolutionary dynamics in a minimal substrate. In Spector, Lee, Goodman, Erik D., Wu, Annie, Langdon, W. B., Voigt, Hans-Michael, Gen, Mitsuo, Sen, Sandip, Dorigo, Marco, Pezeshk, Shahram, Garzon, Max H., and Burke, Edmund, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 702–709, San Francisco, California, USA. Morgan Kaufmann.

Yang, Dekun and Flockton, Stuart J. (1995). Evolutionary algorithms with a coarse-to-fine function smoothing.

Zykov, V., Bongard, J., and Lipson, H. (2005). Co-evolutionary variance guides physical experimentation in evolutionary system indentification. In *The 2005 NASA/DoD Conference on Evolvable Hardware.*