Benefits of the Coevolution of fitness predictors and fitness maximizers

Arash Kayhani, Dalhousie university

Abstract-Surrogate modeling has become very popular strategy in the field of evolutionary computation. In this paper, I implement a modern strategy in this field called coevolution of fitness predictor to improve the performance of the fitness modeling techniques and also research on other benefits of this approach which mostly comes from its unique sampling strategy. Finally, I will compare the results with other algorithms on mathematical functions.

Keywords: Predictors, surrogate modeling, fitness prediction

I. INTRODUTION

Evolutionary computation-the big family of algorithms inspired by the biological evolution have become very popular in all areas of engineering and science. Although many types of evolutionary algorithms have been developed in different areas of engineering and science for many years, only recently, they became applicable for problems with expensive objective function. Scientists are using the new approaches especially surrogate modeling to reduce the use of the real objective function in expensive problems. In this approach (surrogate modeling), the algorithm uses, data points that have already been measured with the true fitness function to create a good approximation of the fitness for the other points and as a result to reduce the overall use of the real fitness function. Although this approach helps us with reducing the real-world evaluations, if we develop a very accurate model, in every generation the algorithm needs an extra heavy calculation for modeling the fitness. In other hand, with a very low accuracy model we might not find the best answer (stuck in local optimum). Therefore, developers usually try to find a good tradeoff between accuracy and efficiency of the surrogate modeling.

In this paper, we will follow an approach called coevolution of the fitness predictors suggested by Schmidt in [1] and [2]. In this approach, we want to find a good subset of the available training data called predictors to develop our fitness model as efficiently as possible. Moreover, we will test the algorithm on problems with different complexity to discover the benefits of this approach for evolutionary search.

Chapter 2 describes all possible benefits of surrogate modeling, chapter 3 is about some well-known subsampling strategies, in chapter 4 we describe the concept of coevolution and how we can use it to make a more advanced subsampling strategy and finally chapter 5 compares the new algorithm with other previous strategies.

II. SURROGATE MODELS

Evolutionary algorithms are not applicable when we have problems with expensive fitness functions. That's because in these kinds of algorithms we need to try as many points to find the optimum. Although, these approached are very expensive, they provide us with best performance compare to other algorithms in black box problems. Recently, new approaches in Machin learning has helped the evolutionary algorithms to significantly reduce the computational cost of the EAs by decreasing the number of evaluations on real world expensive functions and instead using an approximation of the fitness model as a part of the solution evolution.

Moreover, in many real-world problems especially in human interactive evolution there are not real fitness function available, however, we can use the available data to model the user answers and reduce the number of necessary feedbacks. In addition to that in the fitness of many real-world problems can only be calculated in the presence of noise. Especially when we have a very serve noise, evolutionary algorithm cannot progress toward the objective. Fortunately, fitness modeling can effectively reduce the effects of noise on by creating a good model of the real fitness

The other important advantages of the fitness modeling are reducing the complexity and smoothing the landscape of the fitness function and promoting diversity which will be discussed in the next sections.

In [1], researchers are using the tree based GP to model the fitness. Unlike [1], I used a linear Genetic programming that we already developed in the sandboxes for modeling. My GP algorithm contains 4 registers and 4 inputs including 3 constants (1 to 3) and the input of the function with a tournament selection and mutation rate of 0.1.

Results show that surrogate model decreases the number of the real fitness function evaluation. However, we still face the high level of computational effort for modeling. Therefore, we need to find a good tradeoff between the model accuracy and its computational cost.

A beneficial approach would be to use only a subset of the training data to reduce the computational cost of each generation, in the cost of losing accuracy. However, in the next sections we will see that by using coevolution we can increase the accuracy along with efficiency.

Another problem that a good sampling strategy can solve is the problem of overfitting of fitness models on the limited available training data. In Figure 1 shows the very complex problem of $e^{|x|} \times \sin(57.295 \times x)$. (57.295 degrees equal 1 radian). In the Figure 1 the function has very sudden changes from -20 to 20 which makes modeling very hard. Figure 1 shows the best approximation that GP modeling algorithm can make using a 200-training dataset. Figure 2 shows that model is converging to a very small value of error on the training dataset. However, figure 3 shows testing the evolving models on a diferent random test dataset show a severe increase of error as the model overfits on the training data set. (in this paper, we use point evaluation instead of generation or fitness evaluation which gives us a better comparison between different strategies).

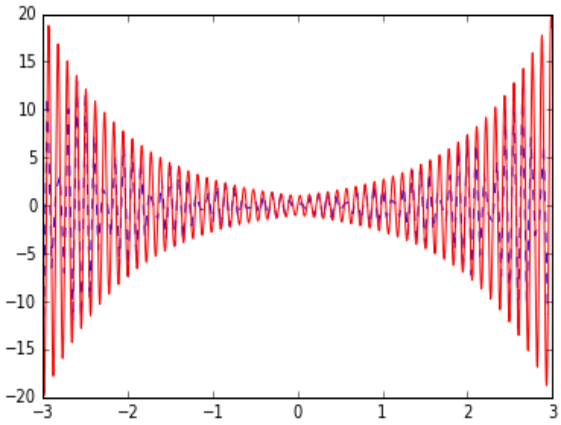


Figure1-$e^{|x|} \times \sin(57.295 \times x)$ real fitness values(red) compared to the best model coming from 200 random samples(blue)

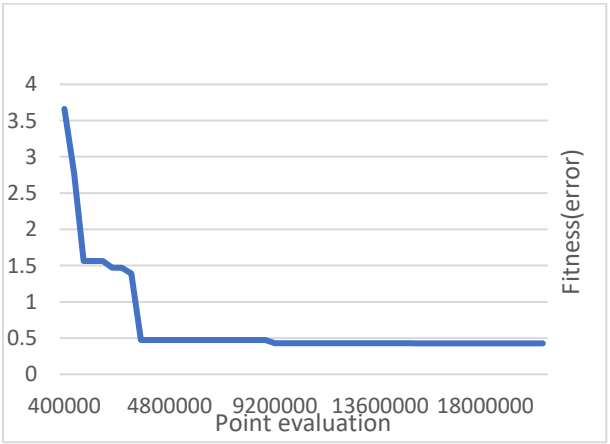


Figure 2- mean absolute fitness error of the models in 5 trials using 200 random training dataset in range [-3,3]

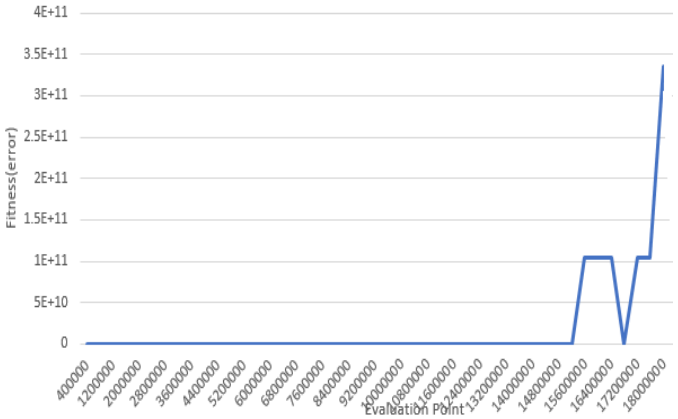


Figure 3 - mean absolute fitness error of the models in 5 trials using 200 random test dataset in range [-3,3]

III. SCALING

Sampling is a popular approach when we want to train our algorithms on very big datasets. In sandbox 2 we used 2 different main strategies including random sampling and sampling equally from all classes, with different values of sampling rate, which helped us with reducing the computational effort of the program evolution. However, they don't show the same benefits in problems like surrogate modeling where solutions belong to a continues range of classes.

Facing with the costly computation of the fitness modeling, researchers in [1] have tried different scaling techniques. In their approach, GP uses a limited number of points from the training dataset as predictors of the fitness of the model on all the training dataset. Predictors are selected using 4 different strategies:

3.1. Static Random: In this approach, a subset of the training data is selected randomly at the run time and it stays fixed during the whole program evolution.

3.2. Random sampling: Its very similar to the static random sampling, but in this approach in every generation we chose a new random sample of the training dataset.

3.3. Coevolution of the predictors: in this approach, the predictors data set coevolves with fitness models' population to make a better prediction of the objective fitness function. It will be explained in detail in the next Chapter.

IV. COEVOLUTION OF FITNESS PREDCTORS

Coevolution is a state-of-the-art strategy in evolutionary computation inspired by the bio-coevolution in nature. In coevolution two or more separate population evolve together when in some way their finesses are related. Mainly, the relation between the fitness of the populations are either in opposite path (competitive coevolution) or in the same direction (cooperative coevolution). In [2] the researchers use

coevolution to evolve three population of models, subsamples(predictors) and trainers. (note that the definition of the predictors, training dataset and other keywords are slightly different between [1] and [2])

4.1. Fitness models: (models are called fitness maximizers in [2] and solutions in [1])

Fitness models are the population of GP models of the underlying world. In this strategy [2], we use the predictors which are a subset of the training dataset to train the models. Fitness of these models comes from the average difference between the real value of the fitness and the model approximation value for that points in the best predictor subset.

$$model = \underset{model \in M}{\operatorname{argmin}} \frac{1}{N} \sum_{s \in P_{best}} |fitness(s) - model(s)| \; ((Fitness\ model\ evolution)$$

This formula shows that the absolute mean deference between the real fitness and the model approximation is calculated form all N points in the best predictor P_{best} subset. As I mentioned before we use a linear GP for the models.

4.2. Predictors:

Predictors are encoded as a subset of the available training dataset. In this algorithm, we evolve the predictor population on a subset of the models' population (called predictor training functions or trainers) with their mean absolute error on all the training dataset. In each generation of the fitness predictors the algorithm develops predictors that have a closer mean absolute error on the predictor training functions compared to the mean absolute error on all the training dataset. In fact, in this stage the algorithm wants to find the best subset of the training data that represents the whole dataset.

$$P^* = \underset{predictor \in P}{\operatorname{argmin}} \frac{1}{N} \sum_{trainer \in T} |trainer(training\ dataset) - trainer(p)| \text{(Predictor evolution)}$$

In this equation, N is the number of trainers in the trainer dataset T and P is the population of the predictors.

4.3. Trainers:

Trainers consist of several recent models in the models' population. In [1], Schmidt suggests multiple strategies for choosing trainers, according to his experiments the best trainers is a trainer that shows the highest prediction variance. The most variant models are the models that have the most different fitness value on the predictors, which provides the trainers dataset with the newest information and as a result enforces novelty to all three evolving populations.

$$t^* = \underset{model \in M}{\operatorname{argmin}} \frac{1}{N} \sum_{predictor \in P} |model(predictor) - \overline{models(predictor)}| \text{(trainer evolution)}$$

This equation choses the most variant model on the current population of the predictor suggested in [1], the other similar evaluation of trainers is proposed in [2], which only evaluates the variance of the models on the best predictor.

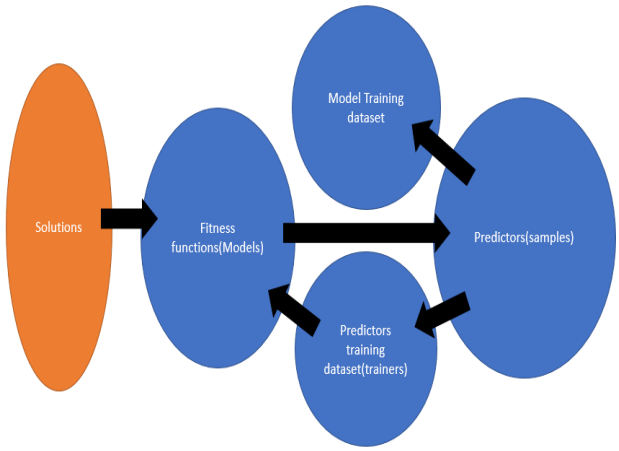


Figure 4-Populations in the predictors coevolution algorithm, the black arrows show where the fitness of each population comes from.

1. Load function training data
2. Initialize predictor training data with f(x)=0
3. Initialize the function and predictor populations
4. Execute a cycle
   a. Evolve predictor population
   b. Evolve function population
   c. Generate a predictor training data item
5. Loop back to step 4

V. EXPERIMENTS AND RESULTS

I used the same fitness functions that Schmidt used in [1] and [2] to compare the results coming from different sampling strategies including predictors coevolution. Algorithms use the parameters in table 2 and 3 which are very similar to the parameters suggested by Schmidt in [1] and [2]. In the attached code depending on the sampling strategy I used different number of function evolution per predictor evolution in a way that the predictor training ratio always stays at 0.05 of the whole point evaluations. For example, in static random we can have more predictor generation because each generation needs less evolution point, but on exact fitness we have less predictor generations because of its high computational cost. Looking at the improvement that only

0.05 predictor evolution makes shows that even with a very limited number of predictor evolution we can highly improve the accuracy of fitness models.

Table 2 strategies parameters

Strategy	Sample Size	Sample Selection Method
Coevolved Predictor Sample	8	Evolved subset
Static Random Sample	8	Random subset chosen at runtime
Dynamic Random Sample	8	Changing random subset
Exact Fitness	200	Use all training data

Table 3 Coevolution of the fitness predictors parameters

Parameter	Control value
Model population size	128
Predictor population size	5
Predictor training ratio	0.05
Predictor Samples	8
Selection	Tournament (4)
Operators	*, /, +, -, log, exp, sin, cos
Maximum size of individuals in GP	35
Number of registers used in GP	4
Terminals	Input and 3 constants (1 to 3)
Test dataset	200 random points
Training dataset	200 random points
F3 range	-10,10
F2 range	-3,3
F1 range	-5,5

Figure 5 shows the results of modeling on all the training dataset on $f2 = e^{|x|} \times \sin(pi/2 \times x)$, it also shows that after a while we face the overfitting problem (high error variance on test dataset).

My results on $f1 = 1.5x^2 - x^3$ function using other sampling techniques (static random and dynamic random) shows close results with exact fitness. However, on more complex problems like $f3 = x^2 e^{\sin(x)} + x + \sin(\frac{pi}{4} - x^3)$ coevolution is very beneficial by converging very fast. It is also important to note that the coevolution strategy produces more reliable result with lower value of fitness variance on test dataset compared to other strategies (figure 8) and training dataset on all except the exact fitness strategy.

Comparing my results with Schmidt result (figure 9) we can see that the rate of convergence is faster in my algorithm but at the end Schmidt finds better results, which might be because of the difference between our algorithms in selection strategy and GP approach (linear or tree). However, after looking at the Schmidt different papers and results I still cannot find out that the fitness values in Figure 9 comes from the training dataset or the test dataset.

figure 6 and 7 shows that coevolution coverage faster than the exact fitness approach on F2, too. However, the convergence rate provided by Schmidt is much higher (figure 9). (His own results are different in [1] and [2] which might be because of the difference in the parameters and selection techniques)

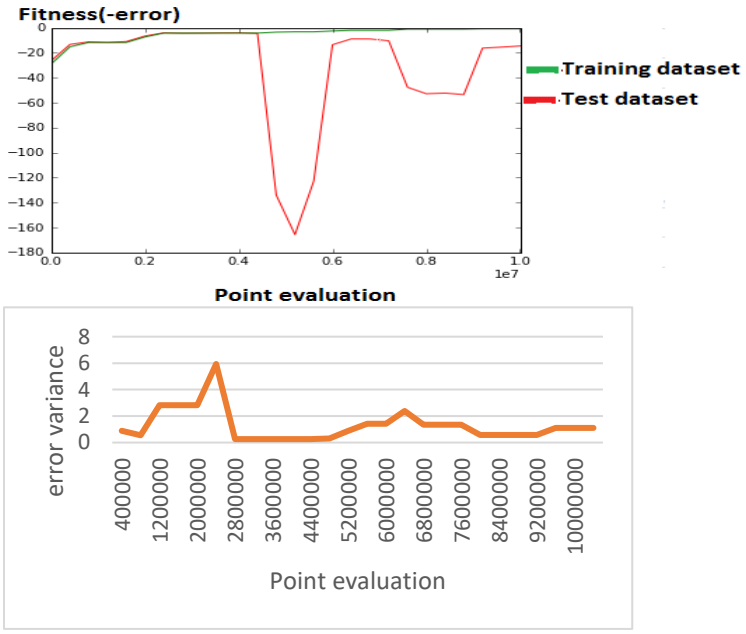
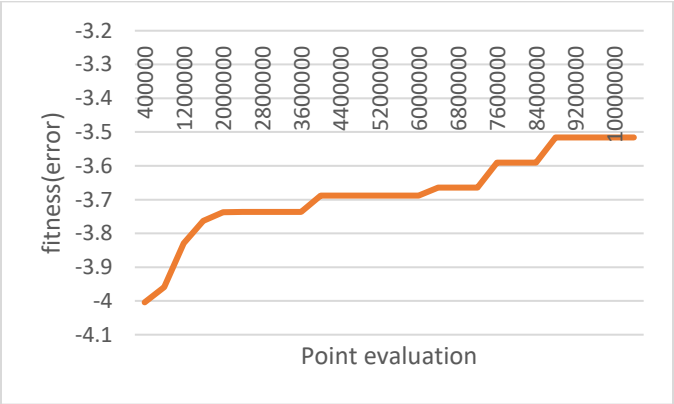


Figure 5- Exact fitness approach on f1, and its fitness variance (average 3 trials) on training dataset.



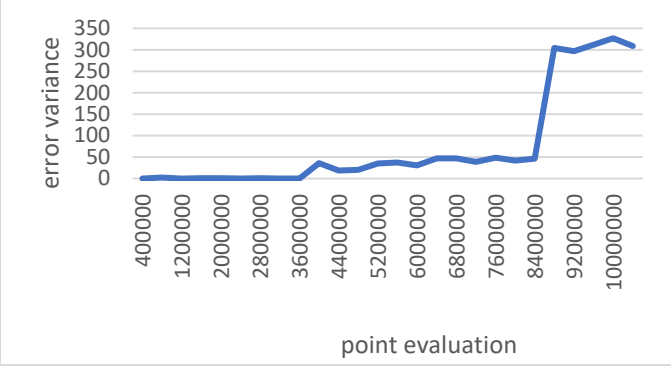


Figure 6-Exact fitness approach on f2 (average 3 trials)on training dataset and its error variance on test dataset increasing at the end probably because of overfitting with the training dataset

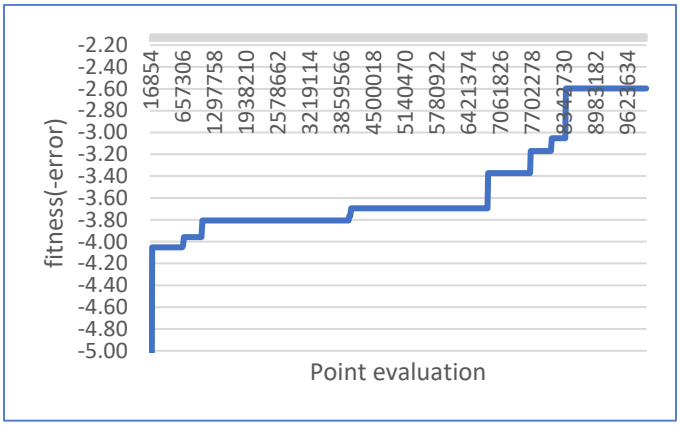


Figure 7-Best individual fitness of the Coevolution strategy on f2 on training dataset (average 2 trials)

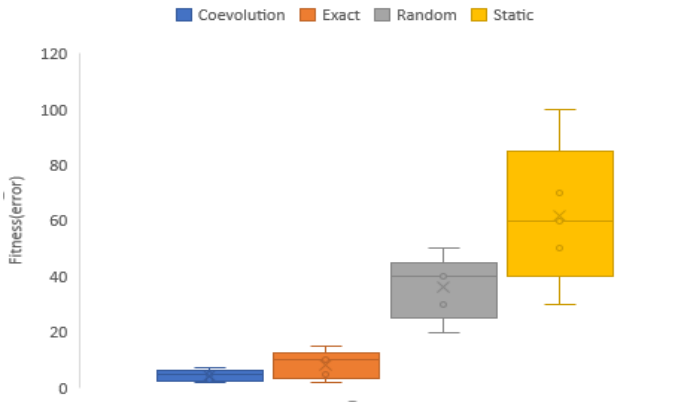


Figure 8-Best fitness values of the strategies on random test dataset (200 random points) on f3 after 10^7 point evaluations (average 5 trials)-Algorithms include: coevolution (coevolution of fitness predictors) ,exact (using exact 200 test dataset) random (dynamic random approach which is very dependent on the number of generation per changing the samples)

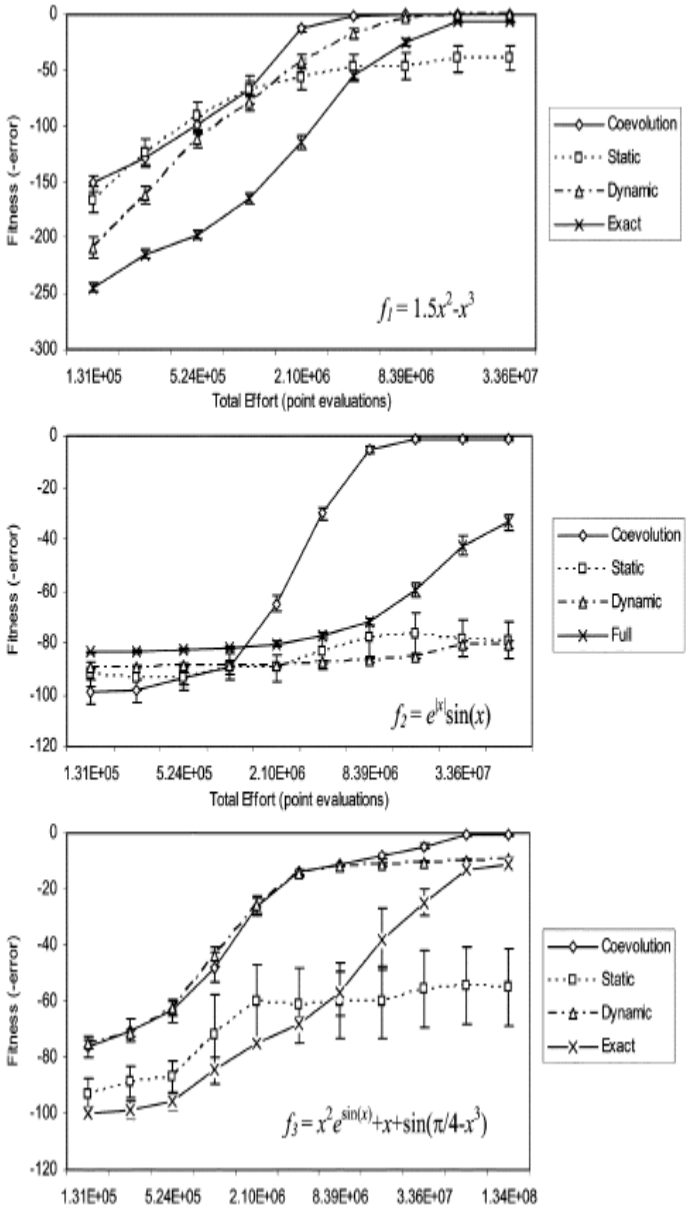


Figure 9-results in the paper [1]

VI. CONCLUSION

The results coming from all the provided experiments in this paper alongside the results from [1] and [2] shows that the predictor coevolution strategy outperforms other strategies. Their improvement comes from 3 main techniques (modeling, sampling, novelty, coevolution) implemented inside the algorithm:

Modeling: The modeling strategy reduces the number of true fitness evaluation and moderates the effect of noise over the fitness evaluation as it is being discussed earlier in section 2. Schmidt in [1] believes that better performance of the coevolution strategy for f3 which is a very complex function (figure 8 and 9) also shows that the right sampling strategy also helps with the noise problem and smoothing the landscape.

Sampling: In addition to helping with noise, comparing the results in the diagram 3,6 and7, predictors can help with the problem of overfitting or as [1] says it can reduce the number of bloated solutions

Novelty: choosing the most variant models for the trainers' population helps the algorithm by evolving the most novel sets of predictors which is very beneficial by searching over the most diverse set of solutions and not getting stuck in the local optimum, which makes the algorithm to converge faster and also to reduce the bloated solutions.

Coevolution: Coevolving the three population creates a very intelligent heuristic, the models evolve very fast with a smaller number of training dataset but the predictors evolve at the same time to the newer and more accurate but diverse models, then they push the models out of their premature convergence, and this cycle repeats until we get close enough to the most accurate fitness approximation.

ATTAHCMENTS

Attached you can find the codes used in this paper and also some of the results that I didn't add to this paper, I also attached my previous studies in evolutionary strategies to emphasize the importance of surrogate modeling

REFRENCES

[1] Schmidt, M.D. and Lipson, H., 2008. Coevolution of fitness predictors. IEEE Transactions on Evolutionary Computation, 12(6), pp.736-749.

[2] Schmidt, M. and Lipson, H., 2005. Coevolution of fitness maximizers and fitness predictors. GECCO Late Breaking Paper.