Fitness Predictors in Genetic Programming

Jan Mayer

Abstract

We present two existing methods to find fitness predictors in genetic programming, both based on coevolution, and compare them. We implement one of the methods and compare the results. We show that coevolved predictors can find better solutions while using same effort.

1 Genetic Programming

Genetic programming (GP) is a collection of evolutionary computation techniques that allow computers to solve problems automatically[6]. It is a technique inspired by the biological evolution. The main idea is to start with usually random population of individuals, apply genetic operations on them and repeat this process until we find an individual that solves our problem sufficiently. The stages of the evolution are as follows:

1. Selection - Select individuals from the current generation, which will be used as parents for the next generation. There are many ways to do this, such as tournament selection or deterministic crowding selection[5] used in this report. To select the individuals it is usually needed to evaluate the whole population, which can often be very computationally demanding.

2. Crossover - This operations takes two parents and combines them together to create an offspring. The exact way depends on representation of the individuals.

3. Mutation - Make a small random changes to the individuals.

2 Symbolic Regression

Symbolic regression is the problem if identifying the analytical mathematical description of a hidden system from experimental data [1], [4]. Unlike polynomial regression or related machine learning data fitting methods it doesn't need any prior knowledge about the data. Here, the individuals are represented by binary trees, where in leaves there are constants and inputs and in remaining nodes there mathematical operations. This way, we can get the value of the represented expressions on every point of our dataset. Here the mutation is done by selecting random node of the individual and replacing it by a random tree

of given length. Crossover is done by selecting random point in two individuals and exchanging the corresponding subtrees between the two individuals. The goal of symbolic regression is to find an expression that fits the training data the best. There are more ways to measure this, in this report we use the mean absolute error:

$$fitness(s) = \frac{1}{N}\sum_{i=1}^{N}|s(x_i) - y_i|$$

where $s(x_i)$ is value of candidate in i-th input in training data and y_i is the i-th output in training data and N is the size of the training dataset.

3 Fitness Predictors

As we mentioned earlier, evaluating all individuals in all training dataset points can be very computationally demanding. That is why we try to find ways to get similar results using just a subset of training dataset. The used subsets are called fitness predictors. Using the predictors results in a slight change of the evolution goal. Instead of mean absolute error on all training data shown in the previous section, we compute the values of the expressions in points of the current predictor:

$$s^* = \arg\max_{s \in S} p(s)$$

where p is the fitness predictor used in current generation and $p(s)$ is predicted fitness of an individual s using predictor p. The problem is which p to use to get the best results. In the following section we will discuss some known methods of constructing the predictors.

4 Coevolution of Fitness Predictors

4.1 Approach by Michal D. Schmidt and Hod Lipson

In this section we talk about method introduced in [7] by Michal D. Schmidt and Hod Lipson. This method constructs and selects the fitness predictors using coevolution. During the evolution there are three populations present in the algorithm:

1. Solutions - The main population of candidate solutions, the best individual is the one with best predicted fitness

$$s^* = \arg\max_{s \in S} p_{best}(s)$$

2. Trainers - Some solutions used for training the fitness predictors, the best trainer is the one with highest variance of predicted fitness among current generation of predictors

$$t_* = \arg\max \frac{1}{N}\sum_{p \in P_{cur}} (p(s) - \overline{p(s)})^2$$

3. Predictors - Population of the fitness predictors, each generation of the main population, the best predictor from this population is chosen and used to evaluate the solutions. The best predictor is the one, which can predict the fitness of individuals in current trainer population the most accurately

$$p^* = \arg\min \frac{1}{N} \sum_{t \in T_{cur}} |fitness(t) - p(t)|$$

The solutions and predictors are evolved using standard GP. At the beginning of the algorithm, all populations are randomized. It is important that the the evolution of predictors doesn't take too much effort (where effort is measured in point evaluations), otherwise there wouldn't be any advantage of using them. In this paper they set the threshold to 5% all computation effort measured in point evaluations. Every 100 predictor generations, new trainer is chosen from the solution population using the above criteria. Also, solution population and predictor population are run on two threads.

4.2 Approach by Michaela Šikulová and Lukáš Sekanina

The disadvantage of the approach introduced in [7] is that the predictors are of fixed size. Many experiments have to be therefore conducted to find the optimal size to use in each task. [3] introduces adaptive fitness predictors. These predictors change their size depending on the *phase of evolution*. This way an optimal size of predictors is found during the run of the coevolution. The main idea is as follows:

- Increase the predictor size if:
  1. Current predictions are too inaccurate
  2. Objective fitness of the best individual in the population is increasing
- Decrease the predictor size if:
  1. Objective fitness of the best individual in the population is stagnating or decreasing

Another difference from the method introduced in the previous subsection is which solutions are selected as new trainers. Here the solution replaces the oldest trainer if it's subjective fitness is better than subjective fitness of the top-ranked trainer.

5 Experiments

In this section we implement approach introduced in [7] and try to mimic published results. As a function for symbolic regression we use $f_2(x) = e^{|x|}sin(2\pi x)$ (Figure 1) from the article. In the article it is said, that the training set had 200 evenly spaced samples and testing set contains additional 200 samples. In

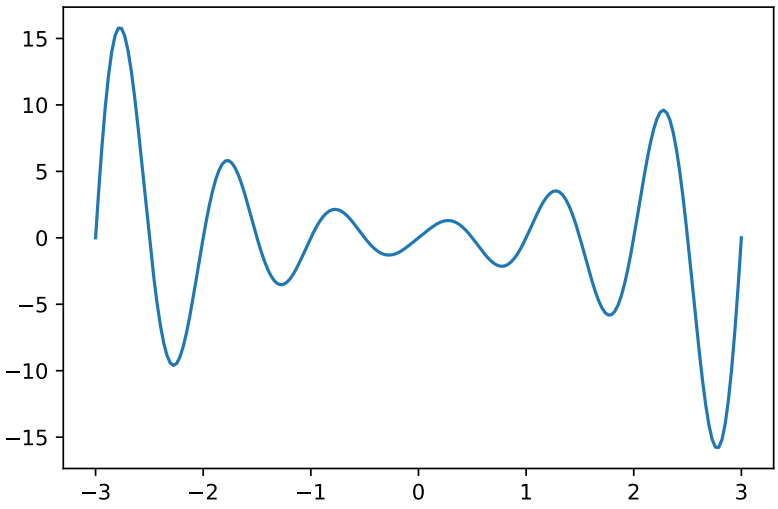

Figure 1: Target function f_2

following experiments, we choose 200 random samples from the same interval and together with samples from training set we use them as a testing set.

5.1 Implementation

For GP we chose to use DEAP: A Python framework for Evolutionary Algorithms [2]. Because it is written in Python, using more threads is problematic and we didn't do it. On the other hand, using multiple threads speeds up the process but the number of point evaluations stays the same.

5.2 Results

5.2.1 Experiment 1

First we set the maximal number of point evaluations to 1×10^7, ran the program 50 times and averaged the test set fitness. We tried different sizes of the fitness predictor and 32 turned out to be the best performing one. This is a surprising result, because in [7] size 8 is used. In our experiment, predictors with size 8 did not perform well at all, as can be seen in the Figure 3 and suffered greatly from over-fitting (Figure 4). Quality of best solutions at the end of the run can be seen in the Figure 5.

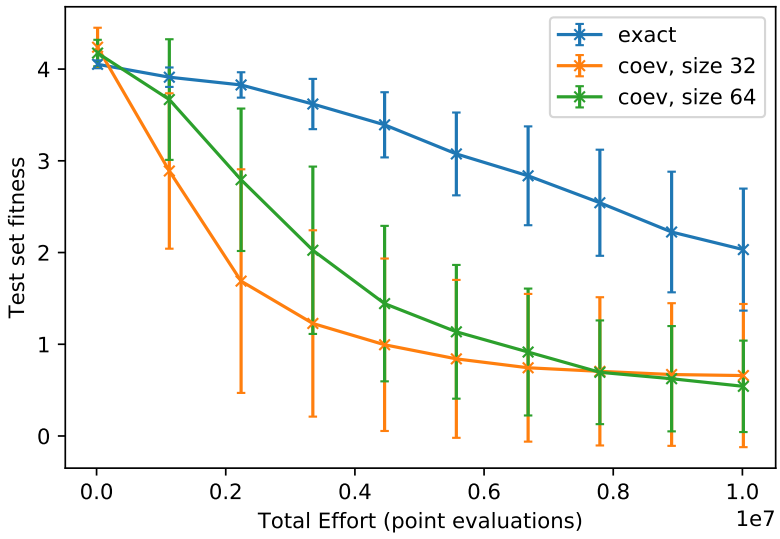

Figure 2: Test set fitness during evolution for function f_2

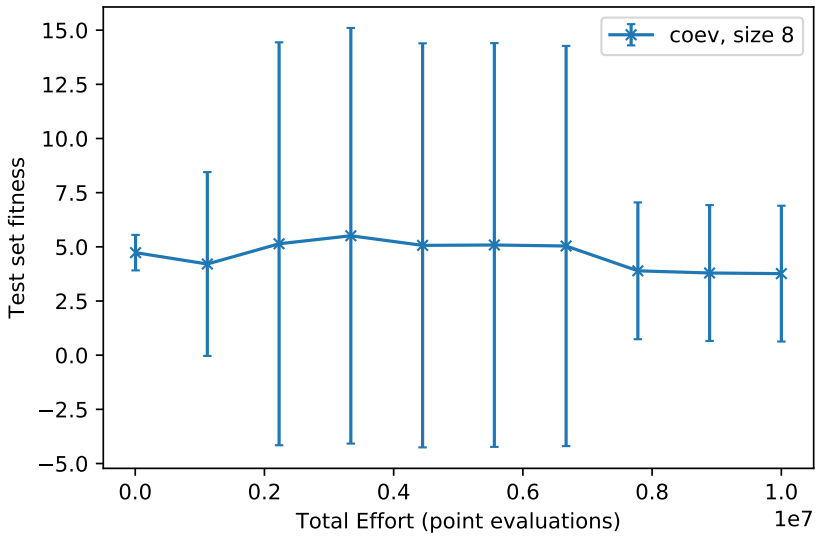

Figure 3: Test set fitness during evolution for function f_2

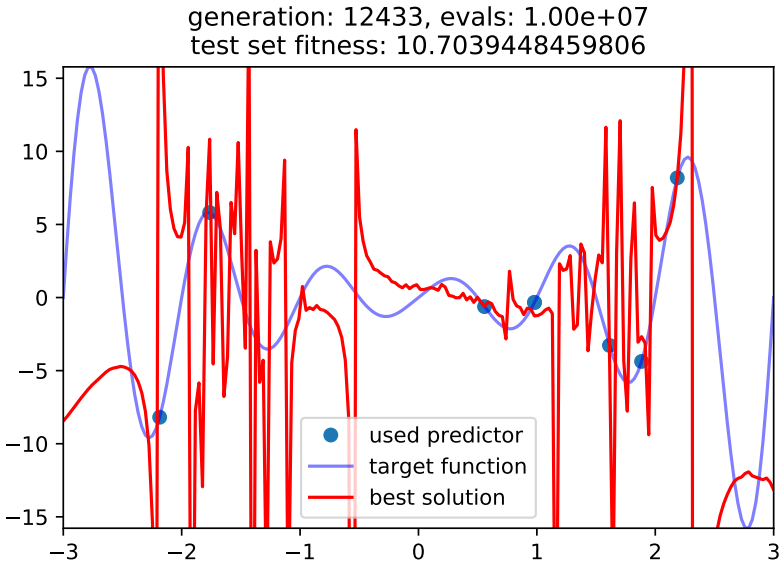

Figure 4: Example of one of the many unsuccessful runs using predictors with size 8

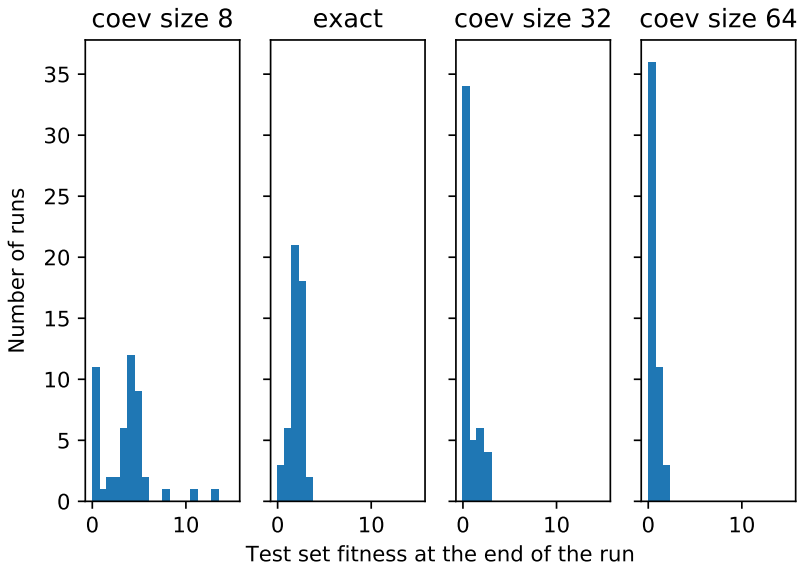

Figure 5: Histogram of fitness of the best solution at the end of the run for 1×10^7 evaluations

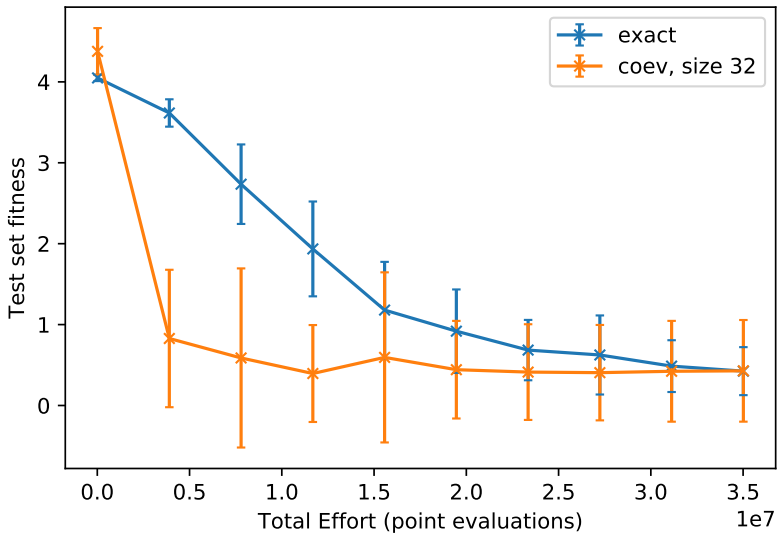

Figure 6: Test set fitness during evolution for function f_2

5.2.2 Experiment 2

On the Figure 2 we can see that using coevolved fitness predictors generate better solution at the end of run with limit of 1×10^7 evaluations. It looks like runs using exact fitness have potential to get better if we increase the limit. This time we limit the runs to 3.5×10^7 evaluations. On the Figure 6 we can see the result. Runs using fitness predictors have clearly a head start in the beginning, but runs using exact fitness catch up, if given enough time. In the Figure 7 we can see that even though the average fitness at the end of the run is similar, runs using fitness predictor generate very good solutions more often. In the Figure 8 we can see, that some samples are chosen significantly more often than others. It seems, that they are usually the points close to some of the local extremes of the function, as these are the points that define the shape of the function the most.

6 Conclusion

We implemented coevolving fitness predictors and conducted several experiments with them. Even though we weren't able to mimic the results in the published papers exactly, we proved that GP using coevolved fitness predictors produces fitter solutions while using same effort. In the future bachelor thesis, we will design our own methods and compare them with the known ones.

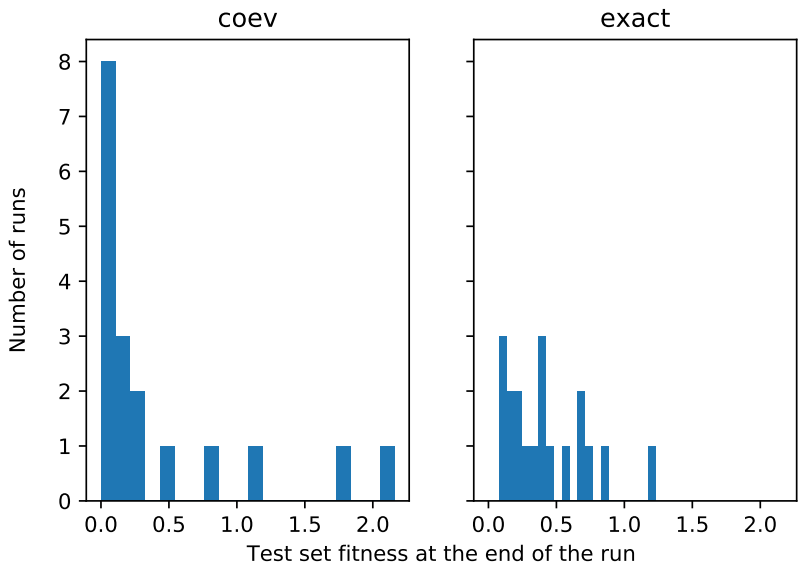

Figure 7: Histogram of fitness of the best solution at the end of the run for 3.5×10^7 evaluations

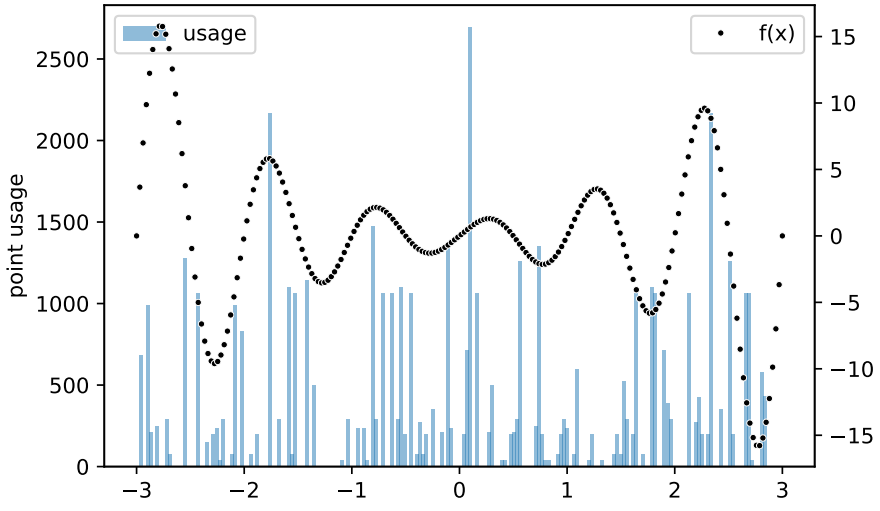

Figure 8: Histogram of used predictors in one of successful runs

References

[1] Douglas A. Augusto and Helio J. C. Barbosa. Symbolic regression via genetic programming. In *Proceedings of the VI Brazilian Symposium on Neural Networks (SBRN'00)*, SBRN '00, page 173, USA, 2000. IEEE Computer Society.

[2] François-Michel De Rainville, Félix-Antoine Fortin, M Gardner, Marc Parizeau, and Christian Gagné. Deap: A python framework for evolutionary algorithms. pages 85–92, 07 2012.

[3] Michaela Drahosova, Lukas Sekanina, and Michal Wiglasz. Adaptive fitness predictors in coevolutionary cartesian genetic programming. *Evolutionary Computation*, 27:1–27, 06 2018.

[4] John Duffy. Using symbolic regression to infer strategies from experimental data. 02 1970.

[5] Samir W. Mahfoud. Niching methods for genetic algorithms. Technical report, 1995.

[6] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).

[7] Michael D. Schmidt and Hod Lipson. Coevolution of fitness predictors. *IEEE Trans. on Evolutionary Computation*, pages 736–749, 2008.