Best SubTree Genetic Programming

Oana Muntean, Laura Diosan, and Mihai Oltean
Department of Computer Science
Faculty of Mathematics and Computer Science
Babes-Bolyai University
Kogalniceanu 1, Cluj-Napoca, 400084
Romania.
oana_muntean85@yahoo.com,
lauras, moltean@cs.ubbcluj.ro

ABSTRACT

The result of the program encoded into a Genetic Programming (GP) tree is usually returned by the root of that tree. However, this is not a general strategy. In this paper we present and investigate a new variant where the best subtree is chosen to provide the solution of the problem. The other nodes (not belonging to the best subtree) are deleted. This will reduce the size of the chromosome in those cases where its best subtree is different from the entire tree. We have tested this strategy on a wide range of regression and classification problems. Numerical experiments have shown that the proposed approach can improve both the search speed and the quality of results.

Categories and Subject Descriptors

I.2.6 [**Learning**]; I.2.8 [**Problem Solving, Control Methods and Search**]

General Terms

Algorithms

Keywords

Genetic Programming, Subtree, Regression, Classification

1. INTRODUCTION

Genetic Programming (GP) [6] is an evolutionary technique used for generating computer programs based on a high level description of the problem to be solved. GP chromosomes are usually trees which are manipulated by using some specific genetic operators.

In this paper we suggest a new variant of Genetic Programming (GP), where the output is not given by the root of the tree as in the case of standard GP. Instead, the best subtree is chosen in order to provide the result of a chromosome. This is why the proposed variant is called Best SubTree Genetic Programming (BSTGP).

There are 2 features which makes BSTGP different from standard GP:

- The best subtree is selected for providing the solution of the problem. This is different from the standard GP where the fitness of a chromosome (tree) is given by its root node.
- Nodes not belonging to the best subtree are deleted. In this way the BSTGP trees may be smaller compared to the GP trees.

All these operations are performed during the fitness computation process. All other elements of a standard GP algorithm [6] remain unchanged.

The proposed approach has been tested against 23 symbolic regression and classification problems. Numerical results have shown that the proposed approach performs very well compared to the standard Genetic Programming.

Numerical experiments show that the number of nodes of the BSTGP trees is smaller than the number of nodes explored by standard GP. This means that BSTGP is usually faster than standard GP.

The paper is organized as follows: Related work is briefly reviewed in Section 2. The proposed approach is described in Section 3. The most important aspect is given in Section 3.1 where the fitness assignment process is detailed. The test problems are briefly introduced in Section 4.1. The results of the numerical experiments are presented in Section 4.3. Section 5 discusses the strengths and weaknesses of the proposed approach. Finally, Section 6 concludes our paper and summarizes the further work directions.

2. RELATED WORK

Different techniques employ different strategies for selecting the sequence of instructions which will provide the solution of the problem.

The most prominent example is Cartesian GP (CGP) [10] where the output node is evolved like all other genes. This means that the graph providing the solution might not contain all nodes from the CGP chromosome.

Linear Genetic Programming (LGP) has also been subject to code optimization. In [3] the authors have removed the instructions that do not participate to the solution. Note that this is different from our approach since, in our case, all nodes of the tree are effective [3], but not all of them belong to the best subtree.

Introns removal is a common operation in GP [1, 8, 12, 13]. Many papers have investigated this issue. Some researchers have argued that introns are beneficial to GP because they protect the code from the destructive effect of crossover.

3. PROPOSED APPROACH

It can be easily seen that each tree has a number of subtrees equal to the number of nodes. For instance, the tree depicted in Figure 1 has 11 subtrees. The distinct ones are depicted in Figure 2.

Usually the result of a GP tree is given by its root node (subtree ST_5 from Figure 2). We have already seen in Section 2 that this is not a general strategy. Various GP techniques choose different subtrees for encoding the solution.

In our approach we do not choose a fixed subtree for providing the solution. Instead, we compute the quality of each subtree and we select the best of them for providing the solution of the problem.

More than that, after fitness computation we delete all nodes not belonging to the best subtree. In this way the size of the entire chromosome might decrease which will lead to a faster search process.

Remark By best subtree we understand the subtree which has the best fitness. For instance, in the case of symbolic regression the fitness has to be minimized. Thus the best subtree is the one with the smallest fitness.

3.1 Fitness assignment

We focus our explanation on symbolic regression problems.

Finding the best subtree of a tree is an easy and inexpensive task. We know that when the fitness of an expression (encoded into a GP tree) is computed, the values of all expressions (encoded by the subtrees of the original tree) are also computed. This operation is done in $O(n)$ steps, where n is the number of nodes of the GP tree. Thus, computing the value of an expression and of all expressions (encoded by subtrees) requires the same number of operations as the computation of the value of the entire expression. Once we have the values of each sub-expression it is very easy to compute their fitness. We only have to take the difference (in absolute value) between the actual and expected output and then we sum these results for all fitness cases.

For finding the best subtree we employ a bottom-up approach: first of all we compute the fitness of the smallest expressions (subtrees) and then we compute the fitness of increasingly bigger sub-expressions (trees). The last expression whose fitness is computed is the one encoded by the entire tree.

The lowest fitness indicates which the best subtree of the chromosome is. Therefore, that subtree will become the new chromosome. All other nodes will be deleted. If more than one subtree having the same lowest fitness, the smallest subtree will be chosen.

3.2 Example

Let's consider a regression problem with two inputs and one output (see Table 1) and a GP chromosome with 5 node levels (depicted in Figure 1). We compute the fitness of each subtree (Figure 2) as described in section 3.1 and we cache the results. A possible order for fitness computation is the following: ST_6, ST_7, ST_1, ST_2, ST_3, ST_4, ST_5. Note that there are more than one order in which the fitness can be computed. For instance ST_6, ST_4, ST_7, ST_1, ST_2, ST_3, ST_5 is also a valid order.

Fitness values for each subtree are given in Table 2.

Table 1: 3 fitness cases for a regression problem with two inputs and one output $f(a, b) = a \times b \times b$

Fitness cases	a	b	f(a,b)
i	1	3	9
ii	2	5	50
iii	1	-2	4

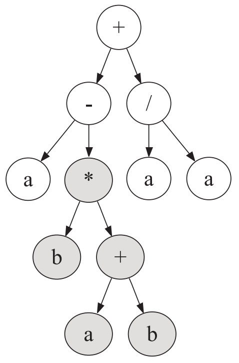

Figure 1: A GP chromosome encoding the expression $a - b \times (a + b) + a/a$. The grey filled nodes form the best fitted expression encoded in this tree.

Table 2: Fitness of each subtree. First of all the errors for each fitness case are computed. The fitness is given as the sum of errors.

Fitness case	ST_6	ST_7	ST_1	ST_4	ST_2	ST_3	ST_5
i	8	6	5	8	3	20	19
ii	48	45	43	50	15	83	82
iii	3	6	5	4	2	5	4
Fitness	59	56	53	62	20	108	105

In this example ST_2 has the best quality. Therefore, it will become the new chromosome. All other nodes of the original tree are deleted.

4. EXPERIMENTS

Several numerical experiments with GP and BSTGP are carried out in this section.

4.1 Test problems

Twenty-three test problems are chosen for these experiments: 13 of them are regression problems and the other 10 problems are binary classification problems.

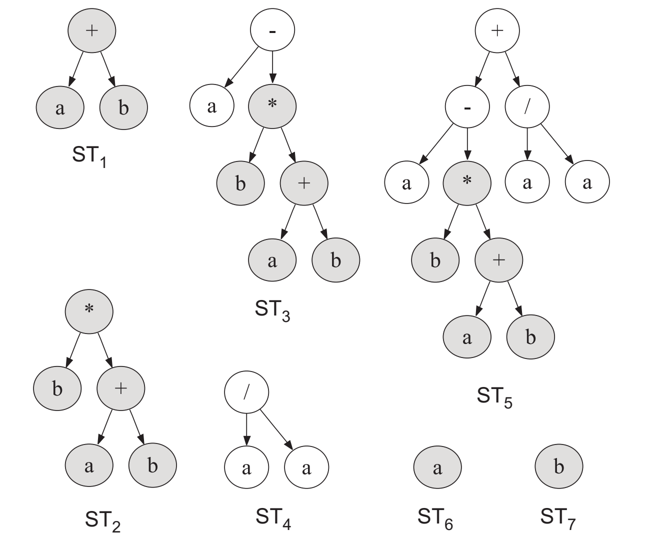

Figure 2: All possible distinct subtrees of the GP chromosome from Figure 1. Grey filled nodes belong to the best subtree.

Seven of them are artificially constructed. The other 16 problems contain real-world data. The data for these problem have been taken from:

- PROBEN1 [14] (which have been adapted from UCI Machine Learning Repository [11]) - T_9, T_{11}, T_{12}, T_{16}, T_{17}, T_{19}, T_{20}, T_{22}, T_{23}, T_{24}, T_{25},
- directly from UCI Machine Learning Repository [11] - T_2, T_8, T_{13}, T_{18},
- from Delve (Data for Evaluating Learning in Valid Experiments) [15] - T_3, T_4, T_5, T_6, T_7, T_{10}, T_{20}, T_{21}.

4.1.1 Symbolic regression problems

T_1 *Quadratic polynomial.* Find a function that best satisfies a set of fitness cases generated by the quartic polynomial [6] function:

$$f_1(x) = x^4 + x^3 + x^2 + x.$$

T_2 *Abalone.* Predict the age of abalone from physical measurements [11].

T_3 *Bank.* Predict the fraction of bank customers who leave the bank because of full queues [15].

T_4 *Puma.* This is a family of datasets synthetically generated from a realistic simulation of the dynamics of a Unimation Puma 560 robot arm [15]. The task is to predict the angular acceleration of one of the robot arm's links. The inputs include angular positions, velocities and torques of the robot arm.

T_5 *Kin.* This is a family of datasets synthetically generated from a realistic simulation of the forward kinematics of an 8 link all-revolute robot arm [15]. The task is to predict the distance of the end-effector from a target. The inputs are things like joint positions, twist angles, etc.

T_6 *Add.* A synthetic function suggested by Jerome Friedman in his "Multivariate Adaptive Regression Splines" paper [5]. The function is:

$$f(x_1, \ldots, x_{10}) = 10 \times \sin(\pi \times x_1 \times x_2) + 20 \times (x_3 - 0.5)^2 + \\ +10 \times x_4 + 5 \times x_5 + n,$$

where n is a random noise (a normally distributed one-dimensional random number with mean 0 and standard deviation 1). The inputs $x_1, \ldots x_{10}$ are sampled independently from a $[0, 1]$ uniform distribution.

T_7 *CpuSmall.* The purpose of this problem is to predict a computer system activity from system performance measures by using a restricted number of attributes (excluding the paging information) [15].

T_8 *Boston.* Prediction of the housing values [11].

T_9 *Building.* Predict the electric power consumption in a building [14].

T_{10} *Cpu.* The purpose of this problem is to predict a computer system activity from system performance measures by using all attributes [15]. This is a generalization of problem T_7.

T_{11} *Flare.* Guess the number of solar areas (of small, medium, and large size) that will happen during the next 24-hour period in a fixed active region of the sun surface [14].

T_{12} *Heartac.* The purpose of this problem is to predict heart disease [14].

T_{13} *Ticdata.* This data set used in the CoIL 2000 Challenge contains information on customers of an insurance company [11].

4.1.2 Classification problems

T_{14} *Diabets.* Diagnose diabetes of Pima Indians [14].

T_{15} *Cancer1.* Classify a tumour as either benign or malignant based on cell descriptions gathered by microscopic examination [14].

T_{16} *Cancer2.* Breast cancer classification into benign and malignant classes [11].

T_{17} *Glass.* The aim of this problem is to classify glass types [14].

T_{18} *Ring.* Leo Breiman's ringnorm example [15]. Classify cases as coming from one of two overlapping normal distributions. Leo Breiman's ringnorm example [4] is a 20 dimensional, 2 class classification example. Each class is drawn from a multivariate normal distribution.

T_{19} *Twonorm* Leo Breiman's two normal example [15]. Classify a case as coming from one of 2 normal distributions. Leo Breiman's twonorm example [4] is a 20 dimensional, 2 class classification example. Each class is drawn from a multivariate normal distribution with unit variance.

T_{20} *Thyrod* Diagnose thyroid hyper- or hypofunction based on patient query data and patient examination data [14].

T_{21} *Hearta.* Predict heart disease [14]. The purpose is to decide whether at least one of four major vessels is reduced in diameter by more than 50%.

T_{22} *Horse* Predict the fate of a horse that has a colic [14] based on the results of a veterinary examination.

T_{23} *Mushroom.* The aim of this problem is to discriminate edible from poisonous mushrooms [14]. The decision is made based on a description of the mushroom's shape, color, odor, and habitat.

Table 3: **The characteristics of the test problems**

	Problem	# Instances	# attributes	Type of attributes
T_1	Quartic	100	1	all Real
T_2	Abalone	4,177	8	one-Binary, seven-Real
T_3	Bank	8,192	8	all Real
T_4	Puma	8,192	8	all Real
T_5	Kin	4,096	8	all Real
T_6	Add	8,192	10	all Real
T_7	CpuSmall	4,096	12	all Real
T_8	Boston	506	13	all Real
T_9	Building	2,104	14	all Real
T_{10}	Cpu	4,096	21	all Real
T_{11}	Flare	533	24	all Real
T_{12}	Heartac	460	35	all Real
T_{13}	Ticdata	5,000	85	all Real
T_{14}	Diabets	400	8	all Real
T_{15}	Cancer1	350	9	all Real
T_{16}	Cancer2	683	9	all Real
T_{17}	Glass	214	9	all Real
T_{18}	Ring	5,000	20	all Real
T_{19}	Twonorm	5,000	20	all Real
T_{20}	Thyroid	4,000	21	all Real
T_{21}	Hearta	460	35	all Real
T_{22}	Horse	364	58	all Real
T_{23}	Mushroom	1,000	125	all Binary

The main characteristics of the test datasets used in the experiments are presented in Table 3.

4.2 Setup for GP methods

The same function set has been used for both GP methods:

$$F = \{+, -, \times, \div\},$$

where $\div$ is the protected division which returns the numerator if a division by zero is attempted, and, otherwise, returns the normal result.

Terminal set T consists of the problem inputs.

We used a steady-state evolutionary model [16, 17] as underlying mechanism for our implementations of all GP methods. The algorithm starts by creating a random population of individuals. The following steps are repeated until a termination criterion is met: Two parents are selected using a standard selection procedure. The parents are recombined in order to obtain two offspring which are then considered for mutation. The best offspring O replaces the worst individual W in the current population if O is better than W.

The following standard genetic operations are performed [6]:

- initialization - ramped half and half,
- selection - binary tournament,
- crossover - both models use a one-cutting point crossover. Having two parent trees, we randomly chosen a one-cutting point in the first parent, another cutting-point in the second parent and we exchanged the subtree rooted at the cutting-point in first chromosome with the sub-tree rooted at the cutting-point in the second individual. Two new individuals are obtained by crossover.
- mutation - a cutting point is randomly chosen: the subtree belonging to that point is deleted, and a new subtree is grown there using the same growth process that was used to generate the initial population. Note that the growth process is limited by a maximal height allowed for the GP trees.

General parameter settings for both methods are given in Table 4. Note that, in the steady-state model, a generation is considered as being complete when 50 new individuals are generated.

Table 4: General parameters of the GP and BSTGP algorithms for solving regression and classification problems.

Parameter	Value
Number of generations	51
Number of individuals	50
Selection	Binary tournament
Crossover	One cutting point
Crossover probability	0.8
Mutation	One point
Mutation probability	0.1
Maximal tree depth	10

Chromosome quality. For the regression problems the quality of a chromosome is given by the mean absolute error (MAE) computed using the formula:

$$MAE = \frac{1}{n} \times \sum_{k=1}^{n} \left| o^k - f^k \right| \qquad (1)$$

where:

- f^k is the expected output value (the value that must be predicted),
- o^k is the obtained value (the value obtained by the best individual) for the k^{th} fitness case and
- n is the number of examples.

For the classification problems the quality of a chromosome is equal to the number of incorrectly classified cases over the total number of fitness cases.

For all problems, we must find the minimal value of the fitness function.

4.3 Numerical results

The results obtained by running each method for all test problems are given in Table 5. The average of 30 different runs is performed.

Taking into account the average results we can see that BSTGP techniques is able to find better solutions compared to GP for all test problems.

Table 5: Results obtained by GP and BSTGP for all test problems. *Avg* and *StdDev* are the mean and the standard deviation of the error for the best individual (in each run) over 30 runs.

Pro-	GP		BSTGP	
blem	Avg	StdDev	Avg	StdDev
T_1	0.06879	0.04933	0.03651	0.05504
T_2	2.49478	0.33221	2.12272	0.17073
T_3	0.03865	0.00558	0.02859	0.00243
T_4	3.75303	0.20149	3.48060	0.15908
T_5	0.32158	0.00978	0.29141	0.01926
T_6	8.80459	1.68633	7.12024	1.90391
T_7	59.56392	3.13729	52.92960	3.45726
T_8	6.37362	0.89921	5.53327	0.27877
T_9	0.16768	0.03031	0.12350	0.02593
T_{10}	49.37945	13.34931	26.28098	2.19767
T_{11}	0.03780	0.00041	0.03706	0.00058
T_{12}	0.24223	0.00808	0.22774	0.00633
T_{13}	0.05826	0.00011	0.05807	0.00009
T_{14}	0.27650	0.02633	0.25938	0.01632
T_{15}	0.11029	0.02657	0.10543	0.01771
T_{16}	1.45673	0.28900	1.19561	0.26543
T_{17}	0.23645	0.02064	0.22640	0.02395
T_{18}	0.00469	0.00100	0.00389	0.00061
T_{19}	0.31410	0.01274	0.29411	0.01259
T_{20}	0.01160	0.00423	0.00943	0.00076
T_{21}	0.23120	0.01960	0.21870	0.01147
T_{22}	0.29093	0.01426	0.28077	0.01377
T_{23}	0.08990	0.04577	0.07840	0.02313

In order to determine whether the differences between the compared GP techniques are statistically significant, we use a t-test with a 0.05 level of significance. Before applying the t-test, an F-test is used for determining whether the compared data have the same variance. The P-values of a two-tailed t-test with 29 degrees of freedom are given in Table 6.

Table 6 shows that the differences between the results obtained by GP and BSTGP are statistically significant ($P < 0.05$) in 19 cases (out of 23).

Table 6: The P values of F-test and t-test

Problem	F-test	t-test
T_1	0.6383	0.0582
T_2	0.0056	0.0001
T_3	0.0007	0.0000
T_4	0.3115	0.0000
T_5	0.0048	0.0000
T_6	0.6021	0.0053
T_7	0.6763	0.0000
T_8	0.0000	0.0003
T_9	0.5023	0.0000
T_{10}	0.0000	0.0000
T_{11}	0.1477	0.0000
T_{12}	0.2970	0.0000
T_{13}	0.5089	0.0000
T_{14}	0.0000	0.0294
T_{15}	0.0850	0.5004
T_{16}	0.0402	0.0041
T_{17}	0.0242	0.0185
T_{18}	0.7146	0.0051
T_{19}	0.5239	0.1634
T_{20}	0.0046	0.3223
T_{21}	0.9600	0.0000
T_{22}	0.8803	0.0274
T_{23}	0.0434	0.0180

Because BSTGP keeps only the best subtree from each tree we are interested to see if there is a reduction in the number of explored nodes. For this purpose we have counted the number of nodes explored by each technique. These values are given in Table 7.

Table 7 provides an evidence for our claim. BSTGP trees are smaller compared to the GP trees. In some cases the average number of BSTGP nodes is 1/4 of the number of GP nodes (see for instance problem T_5). This means that BSTGP is usually faster than standard GP. Again we have used the t-test and F-test for checking whether the differences between the number of explored nodes are significant. Due to space limitations and very large differences these results are not presented.

5. STRENGHTS AND WEAKNESSES

The advantages and weaknesses of the proposed method are discussed in this section.

5.1 Strengths

One of the benefits of BSTGP is the reduced size of trees compared to the standard GP. This leads to a faster search in the solution's space.

Keeping only the best subtree might be an effective way of fighting bloat [2, 7, 9].

5.2 Weaknesses

There are some small difficulties related to the proposed method:

- A (small) extra amount of memory is required for storing the fitness of each subtree. This space is equal to the number of subtrees of each tree. Thus, if the results are stored as real type (**double**) the extra amount bytes is equal to *sizeof*(**double**) $* NumberOfNodes$.

Table 7: Number of nodes explored by each method. The results are average over 30 runs.

Problem	GP	BSTGP
T_1	2,311,143.40	781,341.80
T_2	2,265,416.00	802,422.50
T_3	2,253,610.90	568,260.10
T_4	2,247,850.30	492,125.10
T_5	2,243,119.80	434,925.20
T_6	2,318,371.80	691,744.30
T_7	2,262,462.00	638,732.40
T_8	2,253,297.90	594,143.50
T_9	2,255,947.50	569,851.90
T_{10}	2,269,794.30	602,987.00
T_{11}	2,243,630.10	624,763.90
T_{12}	2,222,638.30	602,973.40
T_{13}	2,235,047.20	667,555.40
T_{14}	2,273,668.30	891,806.80
T_{15}	2,267,700.80	753,104.40
T_{16}	2,343,849.30	765,412.70
T_{17}	2,243,630.10	624,763.90
T_{18}	2,271,783.80	561,705.80
T_{19}	2,324,688.50	1,021,195.00
T_{20}	2,260,328.30	858,234.00
T_{21}	2,328,243.10	810,742.50
T_{22}	2,296,142.60	816,005.10
T_{23}	2,320,104.60	790,631.90

- There is a small overhead when the nodes not belonging to the best subtree are deleted. This overhead is not significant since BSTGP is faster than standard GP (as can be seen from Table 7).

- There are some problems for which is quite difficult to apply this technique because of the overhead in computing the fitness values. One example is induction of grammars using GP. For this problem the fitness of each subtree should be calculated independent of the other subtrees. Hence, a very large complexity may be achieved.

6. CONCLUSION AND FURTHER WORK

In this paper a new way of selecting the subtree providing the solution was investigated.

Instead of using the root of the tree as solution provider, any other subtree was seen as a potential solution of the problem. There is neither practical nor theoretical evidence that one of these subtrees is better than the others.

Moreover, Wolpert and McReady [19, 18] proved that we cannot use the search algorithm's behaviour so far for a particular test function to predict its future behaviour on that function. Thus, fixing the subtree which provides the solution might not be the best strategy to follow. This is why we have designed a dynamic strategy for selecting the subtree providing the solution of the problem.

Nodes not belonging to the optimal subtree are removed. This has improved the efficiency of the method.

Several numerical experiments have been performed by using a wide range of regression and classification problems. The results have shown the effectiveness of the proposed approach.

Further work will be focused on several directions:

- applying the proposed strategy to more problems,
- investigating the same strategy for other GP variants.

7. REFERENCES

[1] D. Andre and A. Teller. A study in program response and the negative effects of introns in genetic programming. *Genetic Programming*, pages 12–20, 1996.

[2] T. Blickle and L. Thiele. Genetic programming and redundancy. In J. Hopf, editor, *Genetic Algorithms within the Framework of Evolutionary Computation*, pages 33–38, Saarbrücken, Germany, 1994. Max-Planck-Institut für Informatik.

[3] M. Brameier and W. Banzhaf. A comparison of linear genetic programming and neural networks in medical data mining. *IEEE-EC*, 5(1):17–26, 2001.

[4] L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, Berkeley, 1996.

[5] J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–141, 1991.

[6] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[7] W. Langdon. Quadratic bloat in genetic programming. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 451–458, 2000.

[8] J. Levenick. Inserting introns improves genetic algorithm success rate: Taking a cue from biology. *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 123–127, 1991.

[9] N. McPhee and R. Poli. *A Schema Theory Analysis of the Evolution of Size in Genetic Programming with Linear Representations.* Springer, 2000.

[10] J. F. Miller and P. Thomson. Cartesian genetic programming. In R. Poli, W. Banzhaf, W. B. Langdon, J. F. Miller, P. Nordin, and T. C. Fogarty, editors, *Proceedings of European Conference on Genetic Programming (EuroGP)*, volume 1802 of *Lecture Notes in Computer Science*, pages 121–132. Springer-Verlag, 2000.

[11] D. Newman, S. Hettich, C. Blake, and C. Merz. Uci repository of machine learning databases, 1998.

[12] P. Nordin, W. Banzhaf, and F. Francone. Introns in nature and in simulated structure evolution. *Bio-Computation and Emergent Computation, Skovde, Sweden*, pages 1–2, 1997.

[13] P. Nordin, F. Francone, and W. Banzhaf. Explicitly defined introns and destructive crossover in genetic programming. *Advances in Genetic Programming*, 2:111–134, 1996.

[14] L. Prechelt. PROBEN1 - A set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, University of Karlsruhe, Faculty of Informatics, 1994.

[15] C. E. Rasmussen, R. M. Neal, G. Hinton, D. van Camp, M. Revow, Z. Ghahramani, K. Kustra, and R. Tibshirani. Data for evaluating learning in valid experiments, 1996.

[16] G. Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 2–9, San Francisco, CA, USA, 1989. Morgan Kaufmann.

[17] G. Syswerda. A study of reproduction in generational and steady state genetic algorithms. In G. J. E. Rawlins, editor, *Proceedings of Foundations of Genetic Algorithms Conference*, pages 94–101. Morgan Kaufmann, 1991.

[18] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[19] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.