A Rigorous Evaluation of Crossover and Mutation in Genetic Programming

David R. White and Simon Poulding

{drw,smp}@cs.york.ac.uk
Dept. of Computer Science, University of York,
Heslington, York, YO10 5DD, UK

Abstract. The role of crossover and mutation in Genetic Programming (GP) has been the subject of much debate since the emergence of the field. In this paper, we contribute new empirical evidence to this argument using a rigorous and principled experimental method applied to six problems common in the GP literature. The approach tunes the algorithm parameters to enable a fair and objective comparison of two different GP algorithms, the first using a combination of crossover and reproduction, and secondly using a combination of mutation and reproduction. We find that crossover does not significantly outperform mutation on most of the problems examined. In addition, we demonstrate that the use of a straightforward Design of Experiments methodology is effective at tuning GP algorithm parameters.

1 Introduction

The role of crossover and mutation in Genetic Programming (GP) has long been the subject of debate in the GP community, particularly whether GP's use of crossover results in a more effective algorithm than approaches that do not, such as local search. Several papers comparing the effectiveness of the two variation operators have been published in the past [1–3]. In this paper, we carry out a principled empirical study, comparing the exclusive use of mutation against the exclusive use of crossover. Thus, we do not aim to provide the optimal combination of the two, but hope that our experiments may provide some illuminating evidence to this debate. The application of rigorous experimental method allows us to make a fair comparison between the two, by applying an equivalent amount of effort to the tuning of parameters for both cases across six example problems that are standard in the literature.

It has been stated that both the evolutionary computation and wider heuristic search communities are sometimes less rigorous in their experimental method and analysis than their counterparts in other experimental disciplines, such as the natural sciences. Effective and efficient Design of Experiments (DoE) [4] methods that are applied as a matter of course in those other disciplines have yet to be fully adopted, and often inferior trial-and-error methods are used instead. In this paper, we will demonstrate the ease of application of DoE methods in the hope that they may be more widely employed in subsequent research. The

adoption of a DoE methodology enables us to make quantified statements about our findings with a strong degree of confidence.

Johnson [5] commented on the weaknesses of experimental method in previous research and outlined a series of practical guidelines to practitioners in heuristic search. We hope to follow Johnson's excellent guidance, and in particular we are promoting the repeatability of our experiments by adopting as simple a methodology as possible, by selecting tools widely used within the community, and by providing both raw results data, code and scripts via the web [6].

2 Previous Work

2.1 Crossover and Mutation

Luke and Spector [1] published a comparison of crossover and mutation over four problems, which was subsequently revised [2] with some improved statistical analysis. The original paper examined both an exclusive choice between crossover and mutation, and secondly a mixed approach or "blend", whereas the second paper examined only the exclusive choice. We hope to add to their results. Their work across the two papers included the equivalent of 36,000 runs of a typical GP system, and one way we can improve upon their work is by taking advantage of Moore's Law: our results are based on millions of runs. We have also increased the number of parameters evaluated from 4 to 10.

Luke and Spector found little difference in performance between exclusive use of crossover or mutation, and often there was no statistically significant difference. In both papers they note that the situation is more complex than it may appear, and that their results are dependent on parameter settings.

Angeline [3] compared crossover to macromutation, arguing that the crossover operator may in fact function as a mutation operator of sorts. He borrowed the phrase *headless chicken crossover* from the Genetic Algorithms community, where crossover is made with a randomly generated tree rather than a second parent. Angeline compared standard crossover to two variants of headless chicken crossover and found that there was little difference in performance across 3 problems, whilst maintaining fixed settings for the remaining parameters.

Koza includes a section on a simple comparison of genetic programming with and without crossover, using mutation in both cases, in his first book [7]. Very little detail of the comparison is provided and it does not appear to employ statistical methods, but within the instance studied, Koza found crossover was a strongly beneficial addition to the algorithm.

These prior comparisons only considered the importance of parameter settings on the comparison to a limited extent and thus their conclusions cannot be generalised. Whilst t-tests were employed, more sophisticated statistical techniques were not used. Our work addresses this issue, through systematic parameter tuning and large-scale experimentation along with extensive analysis.

2.2 Previous Applications of Rigorous Experimental Method

Feldt et al. [8] investigated the significance of various GP parameters within classification problems by applying fractional factorial designs. A large number of parameters, 17, are considered, many of them specific to the example problems. Their work applied the initial stages of DoE screening methods, to determine which parameters are significant.

Coy et. al [9] also applied two-level factorial designs to optimise parameters, although their work does not involve Genetic Programming. Having run a fractional factorial on a small number of problems, they hillclimb a linear approximation of the response surface to optimise the parameters, and finally average the parameter settings across problems. As described below, our pilot study suggests that such response surface techniques may not be consistently effective for GP algorithm parameters. Generalising from one problem to another is also difficult, and for this reason we apply optimisation separately to each individual problem. Interestingly, they also conclude their paper with a comparison between algorithms, though whether this comparison involves parameter tuning for the other algorithms involved is unclear.

3 Hypotheses

The empirical work is driven by the two hypotheses below. Three GP algorithms are used for the empirical investigation described in this paper, and for clarity within the hypotheses we denote these as follows:

$\mathbf{A_c}$ a GP algorithm that employs two genetic operators: crossover and reproduction. We will tune the parameters of the algorithm, and identify the tuned algorithm using the notation $\mathbf{A_c^*}$.

$\mathbf{A_m}$ a GP algorithm that employs two genetic operators: mutation and reproduction. We identify the tuned algorithm as $\mathbf{A_m^*}$.

$\mathbf{A_d}$ a GP algorithm that employs two genetic operators: crossover and reproduction. This algorithm is not tuned during our investigation; instead the algorithm parameters are set to values that are established within the GP community as 'defaults'.

Hypothesis 1: Crossover and Mutation There is a significant difference between the performance of algorithms A_c^* and A_m^* for a given problem instance.

In other words, after tuning each algorithm, the performance of the algorithm employing crossover as a genetic operator is significantly different from the performance of algorithm employing mutation. We use this hypothesis to determine the relative effectiveness of crossover and mutation as genetic operators for each of the problems.

Hypothesis 2: Parameter Tuning There is a significant difference between the performance of A_c^* and A_d for a given problem instance.

In other words, tuning the crossover algorithm results in a significantly different performance from the algorithm using default values. (The 'default' values for a mutation-only GP algorithm are not clearly defined, so we do not make an equivalent comparison for A_m.) We use this hypothesis to demonstrate the effectiveness of the methodology we apply to tune the algorithms.

4 Experimental Design

4.1 Problem Instances

In order to test our hypotheses, it was necessary to select a set of problems. We chose a problem set from the examples distributed with version 16 of the popular ECJ system [10], listed in Table 1. This set includes at least one problem from each category of the examples provided with ECJ. One regression problem includes Ephemeral Random Constants (ERCs), one does not. Note that we have not considered examples using Automatically Defined Functions (ADFs).

Given this selection of problem instances, it was a natural choice to use ECJ as the system to run the algorithm trials described in this paper.

Table 1. ECJ Problems Selected as Problem Instances

Number	Problem
1	Symbolic Regression of $x^4 + x^3 + x^2 + x$ with no ERCs
2	Symbolic Regression of $x^5 - 2x^3 + x$ with ERCs
3	Two Box Problem
4	Santa Fe Ant Trail
5	Boolean 11 Multiplexer
6	Lawnmower

4.2 Response Measure

In comparing the performance of different parameter settings we must select a performance measure. In this work we have used the *fitness of the best individual in the final population* as our response measure. For some of the problems, ECJ will terminate a run as soon as an "ideal" individual is found.

4.3 Parameter Selection

Using ECJ's parameter file system, we selected 9 independent parameters for algorithm A_c and 10 for A_m, as listed in Table 2. A_m has an extra parameter, x_{10}, controlling the expression grown to replace a selected subtree.

Additional ECJ parameters are derived from the set of 10 independent parameters and are listed in the bottom half of Table 2. For example, the probability of reproduction is derived as $1 - x_8$ where x_8 is the probability of crossover or mutation. Similarly, we maintain the product of the population size and number of generations at (approximately) 52500 to ensure the same number of fitness evaluations are made by all algorithms.

Note that tournament selection was chosen as the selection mechanism, hence roulette-wheel, rank selection and other alternatives were not considered. Koza's ramped half-and-half method was used for initialisation and its parameters were treated as experimental parameters, that is, factors to be examined within our experimentation. The justification for these decisions was simply that the number of possible alternatives was in effect infinite.

ECJ provides a series of default parameter values that are the same for all problems. These defaults are based on the work of Koza [7, 11], and no attempt has been made by the ECJ team to optimise these parameters for the individual problems concerned. These defaults are used to provide the parameter values for the default algorithm, A_d, and are listed in the right-hand column of Table 2.

Table 2. Algorithm Parameters and their Ranges

	Description	**Range Low**	**Range High**	**ECJ Default**
x_1	Grow prob. initialisation	0.1	0.5	0.5
x_2	Max. depth initial tree	4	10	6
x_3	Min. depth initial tree	$0.34 \times x_2$	$0.75 \times x_2$	2
x_4	Prob. of selecting root	0	0.5	0
x_5	Prob. of selecting terminal	$0.1 \times (1 - x_4)$	$0.5 \times (1 - x_4)$	0.1
x_6	Max. depth child tree	5	19	17
x_7	Population size	30	1500	1024
x_8	Prob. of crossover/mutation	0.1	1.0	0.9
x_9	Tournament selection size	2	9	7
x_{10}	Min. depth mutated subtree	3	9	5
Dependent Parameters				
x_{11}	Max. depth mutated subtree	x_{10}		5
x_{12}	Max. number of generations	$52500/x_7$		51
x_{13}	Prob. of selecting non-terminal	$1 - (x_4 + x_5)$		0.9
x_{14}	Prob. of reproduction	$1 - x_8$		0.1

4.4 Strategy

The performance of the crossover and mutation algorithms, A_c and A_m, depends on the values of the 10 independent parameters identified above. At one choice of parameter values, A_c may perform better than A_m, and the reverse may be true at a different parameter setting. In this way, an arbitrary choice of parameter settings could be a source of significant bias when comparing the two algorithms.

Therefore, a fair and principled comparison of the two algorithms must establish parameter values in an objective and unbiased manner. A sensible choice (which is also consistent with how the algorithms would be used in practice) is to identify, separately for each algorithm, parameter values at which it operates at its optimum performance and then compare these tuned algorithms.

Locating the *absolute* optimal parameter values for stochastic algorithms is often difficult and can require extensive computing resources for experimentation, especially when there are a large number of parameters as in this case. A compromise is to locate *approximations* to the optimal parameter values using a simpler method that requires computing power of a more realistic scale. If an equivalent amount of effort—in terms of both the computing power used and the data analysis performed—is spent in applying this method to each of the algorithms, it is reasonable to expect the approximations to be similarly close to the absolute optimum for each algorithm, and so the comparisons to be fair. This is the approach we take for the experimental work in this paper.

The experimental strategy therefore consists of two phases:

Parameter Tuning Identify approximations to the optimal parameter values for each algorithm. We tune the algorithms separately for each problem instance, rather than assess a single set of parameter values that give the best performance when averaged over all problem instances.

Performance Comparison At the optimal parameter values located in the first phase, compare the performance of both the crossover and mutation algorithms in order to test Hypothesis 1. In addition, we compare the performance of the crossover algorithm at its optimal parameter values with the algorithm using default ECJ parameters to test Hypothesis 2.

These phases are described in the next two sections. Each section contains a description of the experimental method for that phase and analysis of the results.

5 Parameter Tuning

5.1 Experimental Method

A possible method of tuning the algorithm parameters is to use Response Surface Methodology (RSM) [12, 4]. This is an iterative process that starts in a small region of parameter space and uses experimental results to move the region of interest towards the optimal parameters. The traditional application of RSM is the improvement of industrial processes and it has recently been applied to optimising the parameters of wireless protocols [13].

We performed a pilot study that tested RSM, as well as a more straightforward application of Design of Experiments methodology, as techniques for tuning the parameters of the algorithms A_c and A_m. Contrary to our expectations, the DoE method located parameter values that were closer to the optimum than those found using RSM. (We speculate that the relatively high dimensionality of the parameter space presents a challenge to RSM, but more rigorous experimentation is necessary before we are able to comment on the relative efficacy

of each method.) In addition, the RSM required more, potentially subjective, choices from the experimenter than did DoE. We therefore proceeded with the DoE method for the parameter tuning phase described in this section.

Our DoE methodology approximates the performance, y, of the algorithm to the parameter settings, x_i, using a second-order linear model:

$$y = \beta_0 + \sum_i \beta_i x_i + \sum_i \sum_{j>i} \beta_{ij} x_i x_j + \sum_i \beta_{ii} x_i^2 + \epsilon \tag{1}$$

This model is almost certainly a simplification of the actual relationship between the algorithm's performance and its parameters, especially over the large ranges we allow for the parameter values. However, if we keep in mind the goal of locating *approximations* to the optimal parameter values using an objective methodology, this simplification may be appropriate. The results of the pilot study provided evidence that this is indeed the case.

The β coefficients in the equation define the relationship between algorithm parameters and performance. The objective of the experiments described in this section is to estimate the value of these coefficients in order to fit the model to the actual performance of the algorithm.

The algorithm performance is also affected by the sequence of random numbers that control operations such as initialisation, crossover, mutation, reproduction and selection. We account for this affect using the 'noise' term, ϵ: it quantifies the difference between the predicted mean performance of the algorithm and the observed performance of a single run of the algorithm with a specific random number seed.

5.2 Experimental Design

In order to provide data that is used to estimate the β coefficients in the model, a series of experimental trials are run at different parameter values. The choice of these parameter values is specified by an experimental design. For second-order linear models such as equation (1), a Central Composite Design (CCD) is often chosen since good coefficient estimates can be obtained with relatively few experimental trials [4]. A further pilot study compared the use of CCD with a three-level full factorial design (FD), and suggested the latter to be more effective at locating good approximations to the optimum algorithm parameters, even when a large number of the repetitions of the CCD were used to reduce the effect of the stochastic noise, i.e. the changes in algorithm performance caused by the random seeds. (We speculate that CCD is an efficient and effective design when a second-order linear model is a good approximation of the actual response, but when the model is a necessary simplification as it is here, a three-level FD is more effective since it explores many more points in the parameter space.)

The three-level FD considers three values of each parameter: the points at each end of the range shown in Table 2 and a point in the middle of the range. The ranges are sufficiently large to incorporate most reasonable values for each parameter, but exclude extreme parameter values (e.g. a crossover probability of

0, or a tournament selection size of 1) that might significantly alter the nature of the algorithm and therefore the shape of the response surface in such regions. The full design consists of all possible combinations of the three values for each parameter, resulting in $3^9 = 19,683$ design points for A_c, and $3^{10} = 59,049$ design points for A_m.

There is redundancy inherent in the three-level factorial design: it contains many more design points than is necessary to fit a second-order model. This allowed us to use a relatively small number of repetitions—two, using a difference random seed in each case—to accommodate the stochastic noise and therefore improve the accuracy of model fitting.

5.3 Analysis

As described in Section 4.2, our response measure is the fitness of the best individual in the final population. We use the 'adjusted fitness' value returned by ECJ for this measure since it has convenient mathematical properties. It is a value in the range $(0, 1]$, with a value of 1 indicating the ideal solution was found by the algorithm.

The model of equation (1) is fitted to the response values using standard linear regression, which estimates the values of the β coefficients.

As a first step in this process, a Box-Cox transform [4] is applied to the adjusted fitness, y:

$$y' = \begin{cases} \frac{y^\lambda - 1}{y} & \text{if } \lambda \neq 0 \\ \ln(y) & \text{if } \lambda = 0 \end{cases} \tag{2}$$

The value of λ is chosen from the range $[-2, 2]$ so that the fit of the transformed responses, y', to the second-order linear model of equation (1), is maximised. The analysis of the linear model makes particular assumptions, particularly as to the distribution of the random variable ϵ, and this transform enables the responses to satisfy the assumptions as closely as possible.

To locate the optimal parameter values for the algorithm, we apply a deterministic optimisation technique, quadratic programming, to the fitted model. This technique locates the parameter values that give the largest response from the model, and therefore the best performance from the algorithm. We constrain potential parameter values to the ranges listed in Table 2.

5.4 Results

The tuned parameter values are shown in Table 3 for algorithm A_c, and Table 4 for algorithm A_m.

6 Performance Comparison

6.1 Experimental Method

The second phase of experimentation considers the tuned algorithms, A_c^* and A_m^*, and, in addition, the default algorithm A_d. To compare algorithm perfor-

Table 3. Parameter Values for Tuned Crossover Algorithm, A_c^*

	Description	**Prob 1**	**Prob 2**	**Prob 3**	**Prob 4**	**Prob 5**	**Prob 6**
x_1	Grow prob. initialisation	0.9	0.1	0.1	0.9	0.9	0.3973
x_2	Max. depth initial tree	4	8	8	4	4	4
x_3	Min. depth initial tree	3	5	4	2	1	3
x_4	Prob. of selecting root	0	0	0	0	0	0
x_5	Prob. of selecting terminal	0.2665	0.4723	0.4612	0.4299	0.5	0.1
x_6	Max. depth child tree	17	19	19	17	16	19
x_7	Population size	1221	1221	1094	1500	1500	143
x_8	Prob. of crossover	1	1	1	0.9861	1	0.9541
x_9	Tournament selection size	2	9	9	9	9	7

Table 4. Parameter Values for Tuned Mutation Algorithm, A_m^*

	Description	**Prob 1**	**Prob 2**	**Prob 3**	**Prob 4**	**Prob 5**	**Prob 6**
x_1	Grow prob. initialisation	0.8364	0.9	0.1	0.9	0.1	0.9
x_2	Max. depth initial tree	4	7	8	4	9	4
x_3	Max. depth initial tree	2	3	3	2	5	1
x_4	Prob. of selecting root	0	0	0	0	0	0
x_5	Prob. of selecting terminal	0.5	0.5	0.5	0.5	0.5	0.5
x_6	Max. depth child tree	18	18	19	14	19	19
x_7	Population size	1458	1193	1010	1419	808	30
x_8	Prob. of mutation	0.9386	0.816	1	0.8488	0.8133	1
x_9	Tournament selection size	9	9	9	8	9	9
x_{10}	Min. depth mutated subtree	3	3	9	3	9	9

mance, each combination of algorithm (A_c^*, A_m^* and A_d) and problem (1 to 6) was run 500 times and the response measured. Each algorithm run used a different random seed. The sample size of 500 is an estimate of an appropriate number of trials to demonstrate a statistically significant difference between the algorithms in the presence of stochastic noise.

6.2 Analysis

The hypotheses of Section 3 relate to a *significant* difference between the performance of the algorithms. Our analysis considers two criteria of significance:

Statistical Significance A statistical analysis of the difference in observed algorithm performance must provide evidence that the observed difference is unlikely to be a chance result.

Scientific Significance The difference in observed algorithm performance—the effect size—must be sufficiently large in comparison to the stochastic noise (arising from the choice of random seeds) to be scientifically important.

Given a sufficiently large sample size it would be possible to demonstrate a *statistically* significant difference in algorithm performance, even if the observed

difference were extremely small. Such a result would unlikely be *scientifically* important, and our second criterion guards against this situation. In particular, it moderates against the possibility that our choice of a sample size of 500 for the comparisons is too large.

We apply *non-parametric* statistical tests to analyse both types of significance. These types of tests make few, if any, assumptions about the distribution from which the observed responses are sampled. Equivalent *parametric* tests, such as a t-test, make specific assumptions about the distribution. While such parametric tests can be very effective, even small deviations from these assumptions can invalidate the test results [14]. By using non-parametric tests, we therefore avoid the need to verify our samples satisfy the assumptions and also ensure that the test results are robust to any undetected deviations from the test assumptions.

To analyse *statistical significance*, we apply the Mann-Whitney-Wilcoxon or rank-sum test [15]. The null hypothesis for this test is that the responses of the two algorithms have identical distributions with equal medians; the alternative hypothesis is that the distributions are different. We use a 5% significance level: a p-value of $< 5\%$ rejects the test's null hypothesis and indicates a statistically significant difference in algorithm performance.

To analyse *scientific significance*, we calculate the Vargha-Delaney A statistic as a measure of effect size compared to stochastic noise [16]. This statistic is independent of the sample size and has a range of $[0, 1]$: a value of 0.5 indicates no difference in algorithm performance; values between 0.5 and 1.0 indicate increasingly large effect sizes where the first algorithm has the better performance; values between 0.5 and 0.0 indicate increasingly large effect sizes where the second algorithm has the better performance. The choice of what constitutes a significant effect size can depend on context. In this case, we apply the guidelines of Vargha and Delaney [16] that an A statistic greater than 0.64, or less than 0.36, indicates a 'medium' or 'large' effect size. We consider any comparison demonstrating 'medium' or 'large' effect sizes as scientifically significant.

6.3 Results

The results of the comparison between the A_c^* and A_m^* are presented in Table 5, and between the A_c^* and A_d in Table 6. Highlighted in bold are p-values that show a statistically significant difference in algorithm performance at the 5% significance level. The Vargha-Delaney A statistic shows which algorithm has the better performance: values greater than 0.5 indicate that A_c^* performs better, while values less than 0.5 indicate that A_m^* and A_d perform better. Values of this statistic highlighted in bold indicate a medium or large effect size which we consider to be scientifically significant.

The results in Table 5 show that Hypothesis 1 holds for two of the six problems: for Problems 1 and 5, the tuned crossover algorithm, A_c^*, demonstrates a superior performance compared to the tuned mutation algorithm, A_m^*, that is both statistically and scientifically significant. Using a similar analysis, the results in Table 6 show that Hypothesis 2 holds for three of the six problems: for

Table 5. Comparison of Tuned Crossover Algorithm, A_c^*, to Tuned Mutation Algorithm, A_m^*

Statistic	**Prob 1**	**Prob 2**	**Prob 3**	**Prob 4**	**Prob 5**	**Prob 6**
rank-sum p-value	$< \mathbf{10^{-5}}$	0.631	**0.0266**	0.0653	$< \mathbf{10^{-5}}$	$< \mathbf{10^{-5}}$
Vargha-Delaney A statistic	**0.757**	0.491	0.541	0.533	**0.939**	0.588

Table 6. Comparison of Tuned Crossover Algorithm, A_c^*, to Default Algorithm, A_d

Statistic	**Prob 1**	**Prob 2**	**Prob 3**	**Prob 4**	**Prob 5**	**Prob 6**
rank-sum p-value	$< \mathbf{10^{-5}}$	0.8662	$< \mathbf{10^{-5}}$	$< \mathbf{10^{-5}}$	$< \mathbf{10^{-5}}$	1.00
Vargha-Delaney A statistic	**0.696**	0.503	0.612	**0.739**	**0.825**	0.500

Problems 1, 4, and 5, the tuned crossover algorithm, A_c^*, demonstrates a superior performance compared to the default algorithm, A_d, that is both statistically and scientifically significant.

7 Conclusions & Further Work

The use of crossover has a significant effect in improving the performance of only two problems out of six; it does not significantly outperform mutation on the remaining four problems (as discussed above, we require a significant result to demonstrate both statistical and scientific significance). This is consistent with the conclusions of previous work. Moreover, since it is the outcome of rigorous experimental methodology and robust data analysis, we are able to be very confident in the validity of the result.

It is worth noting that A_m is a restricted form of mutation that will only use fixed-length subtrees, and variable-sized mutation may prove even more competitive with A_c. This lends credence to the idea that crossover is often only a macromutation operator, and a population-based paradigm using crossover may not be most effective at solving the majority of these problems.

The next step in the investigation of the role of crossover and mutation is to compare the performance of algorithms that use both genetic operators, and to explore the performance of variations of the mutation algorithm.

The results also demonstrate the effectiveness of using a Design of Experiments (DoE) methodology based on a factorial design in tuning GP algorithm parameters: for three of the six problems this technique produced tuned parameter values that were significantly better that the default values. As described above, we were surprised that a pilot study showed that this straightforward DoE approach resulted in more accurately tuned parameters than a more sophisticated technique, Response Surface Methodology, and a more efficient experimental design, the Central Composite Design.

Further work will investigate the observed superiority of the DoE methodology in tuning algorithm parameters and perform an exploration of the response surfaces of GP algorithms in order to identify a reason for this superiority.

8 Acknowledgements

This work is supported by an EPSRC grant (EP/D050618/1), SEBASE: Software Engineering By Automated SEarch.

References

1. Luke, S., Spector, L.: A comparison of crossover and mutation in genetic programming. In: Genetic Programming 1997: Proceedings of the Second Annual Conference, Morgan Kaufmann (1997) 240–248
2. Luke, S., Spector, L.: A revised comparison of crossover and mutation in genetic programming. In: Genetic Programming 1998: Proceedings of the Third Annual Conference, Morgan Kaufmann (1998) 208–213
3. Angeline, P.J.: Comparing subtree crossover with macromutation. In: EP '97: Proceedings of the 6th International Conference on Evolutionary Programming, Springer-Verlag (1997) 101–112
4. Montgomery, D.C.: Design and Analysis of Experiments. 6th edn. John Wiley & Sons, Inc. (2005)
5. Johnson, D.S.: A theoretician's guide to the experimental analysis of algorithms. In Goldwasser, M.H., Johnson, D.S., McGeoch, C.C., eds.: Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges. American Mathematical Society (2002) 215–250
6. Online Experiment Source Code and Scripts: http://www.cs.york.ac.uk/~drw/papers/eurogp2009/
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press (1992)
8. Feldt, R., Nordin, P.: Using factorial experiments to evaluate the effect of genetic programming parameters. In: Proceedings of the European Conference on Genetic Programming, Springer-Verlag (2000) 271–282
9. Coy, S.P., Golden, B.L., Runger, G.C., Wasil, E.A.: Using experimental design to find effective parameter settings for heuristics. Journal of Heuristics **7**(1) (2001) 77–97
10. ECJ: http://http://cs.gmu.edu/~eclab/projects/ecj/ (2008)
11. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press (1994)
12. Myers, R.H., Montgomery, D.C.: Response surface methodology: process and product optimization using designed experiments. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc. (2005)
13. Vadde, K.K., Syrotiuk, V.R., Montgomery, D.C.: Optimizing protocol interaction using response surface methodology. IEEE Trans. Mob. Comput. **5**(6) (2006) 627–639
14. Leech, N.L., Onwuegbuzie, A.J.: A call for greater use of nonparametric statistics. Technical report, US Dept. Education, Educational Resources Information Center (2002)
15. Wilcoxon, F.: Individual comparisions by ranking methods. Biometrics Bulletin **1**(6) (1945) 80–83
16. Vargha, A., Delaney, H.: A critique and improvement of the CL common language effect size statistics of McGraw and Wong. J. Educational and Behavioral Statistics **25**(2) (2000) 101–132