Contents

Chapter 1

A FRAMEWORK FOR THE EMPIRICAL ANALYSIS OF GENETIC PROGRAMMING SYSTEM PERFORMANCE

Oliver Flasch and Thomas Bartz-Beielstein
Cologne University of Applied Sciences, Germany

Abstract This chapter introduces a framework for statistical sound, reproducible empirical research in Genetic Programming (GP). It provides tools to understand GP algorithms and heuristics and their interaction with problems of varying difficulty. Following an approach where scientific claims are broken down to testable statistical hypotheses and GP runs are treated as experiments, the framework helps to achieve statistically verified results of high reproducibility.

Keywords: Genetic Programming, Symbolic Regression, Design of Experiments, Sequential Parameter Optimization, Reproducible Research, Multi-Objective Optimization

1. Introduction

This chapter introduces a framework for empirical analysis of Genetic Programming (GP) system components and their influence on GP performance. Main ideas are borrowed from the empirical approach to research in evolutionary computation described in Bartz-Beielstein, 2006. Performing a well-founded experimental analysis provides valuable insight into GP components.

In current GP research, much repeated work in experimental planning, setup and result analysis is required when proposing improvements in GP system components such as selection and variation operators, individual representations, or search heuristics. To measure the performance benefit of an improved GP system component, a set of test functions has to be implemented, GP system parameters have to be chosen, experiments have to be designed and a statistical result analysis has to be

conducted. As obtaining results of statistical significance often requires many independent runs, distributed execution is necessary.

While this effort is necessary for GP research, it might constitute a barrier of entry to newcomers. Even worse, time spent re-implementing GP research infrastructure is lost for working on core issues. As each group working in GP research has to provide not only their own GP system, but also supporting infrastructure often tightly coupled to local conditions, reproducing results is often difficult. Fortunately, much of the infrastructure for reproducible research in GP can be provided as a framework and implemented as open-source software.

Today, several high quality GP systems exist that could constitute the basis of this framework. For instance, DataModeler is a flexible high-performance GP system geared to real-world symbolic regression. (Vladislavleva, 2008; Kotanchek et al., 2007; Smits and Vladislavleva, 2006) RGP[1] is an open-source GP system based on the statistical software environment R. (Flasch et al., 2010; R Development Core Team, 2011) Compared to DataModeler system, RGP is still quite immature, yet basic features for multi-objective typed GP are implemented efficiently.

The framework introduced in this chapter is applicable to most GP systems. Presently, a prototype hosted by the statistical software environment R exists, supporting the RGP system. Currently, the framework has the following features:

- a modular GP system fast enough for real-world applications (RGP)
- a set of test problems of scalable difficulty
- tools for automatic parameter tuning based on the sequential parameter optimization toolbox (SPOT) (Bartz-Beielstein, 2010)
- tools for statistical analysis of GP results
- tools for result visualization and automated result reporting
- support for distributed execution compute clusters

The framework's implementation is under active development and will be published as a supplemental part of the first author's PhD thesis. A preliminary version in form of an R package will be available on the RGP website.

This chapter will introduce the framework by means of an example study. To provide baseline results, the well-known TinyGP system (Poli, 2004) is examined on a small set of scalable symbolic regression test problems. Then, the performance benefits of several improvements to a still simple GP system, Generational Multi-Objective GP (GMOGP), are

[1]The RGP package is available via the Comprehensive R Archive Network (CRAN) at `http://cran.r-project.org/` or directly at `http://rsymbolic.org/projects/show/rgp`.

studied empirically. Thus, the remainder of this chapter is structured as follows: Section 2 introduces the different GP search heuristics considered in the example study. Scalable test problems are described in Section 3. Section 4 introduces the experimental setup and formulates research questions. Results are described in Section 5, which is followed by a discussion and conclusions in Section 6. The chapter closes with an outlook to further work in Section 7.

2. GP Search Heuristics

In this work, the term GP *search heuristic* denotes the concrete search strategy used in a GP system, which is in principle independent of the concrete GP search space. Classical GP uses a steady state Genetic Algorithm (GA) with tournament selection to search genotype space. In the GP literature, many different concrete variants have been described. It is possible to de-couple the search heuristic from the search space, giving rise to a wide variety of possible hybrid algorithms.

Historically, every GP system implemented its own search heuristics, while exhaustive comparisons of GP search heuristics, isolated from the concrete GP search space, are scarce. Many modern GP systems often employ multi-objective evolutionary algorithms (EMOAs) as search heuristic. Steady state algorithms with Pareto tournament selection seem to be the predominant EMOA variants used in modern GP systems. In simple GP systems mainly designed for research and teaching, single-objective steady state evolutionary algorithms (EAs) with tournament selection are still widespread.

This section describes the two GP search heuristics examined in this example study. The first GP search heuristic selected for this study is the straight forward single-objective tournament selection-based TinyGP search heuristic. The second GP search heuristic is based on a generational multi-objective EA and adds modern means of controlling bloat and preserving population diversity.

Common Components

The components of a GP system responsible for individual initialization and variation (i.e. mutation and recombination) are in principle independent of the search heuristic. The same applies to the concrete individual representation and the means used for individual evaluation. As the main focus of the study lies on comparing the performance of different search heuristics, classical tree GP variants of these common components were used. (Koza, 1992) Parameters of these common com-

Table 1-1. Parameters of the RGP system independent of the search heuristic used.

	Variable (Symbol)	*Domain*	*Default*
Subtree Mutation Probability	`mutationSubtreesP` (p_{mst})	$[0, 1]$	$\frac{1}{3}$
Subtree Mut. Insert/Delete Prob.	`mutationSubtreesPinsertDelete`	$[0, 1]$	0.5
Subtree Mut. Subtree Prob.	`mutationSubtreesPsubtree`	$[0, 1]$	0.9
Subtree Mut. Constant Prob.	`mutationSubtreesPconstant`	$[0, 1]$	0.5
Subtree Mut. Constant Minimum	`mutationSubtreesConstantMin`	$\mathbb{R}$	-1
Subtree Mut. Constant Maximum	`mutationSubtreesConstantMax`	$\mathbb{R}$	1
Subtree Mut. Depth Maximum	`mutationSubtreesDepthMax`	$\mathbb{N}$	2
Function Mutation Probability	`mutationFunctionsP` (p_{fun})	$[0, 1]$	$\frac{1}{3}$
Constant Mutation Probability	`mutationConstantsP` (p_{con})	$[0, 1]$	$\frac{1}{3}$
Constant Mut. Mean	`mutationConstantsMu`	$\mathbb{R}$	0
Constant Mut. SD	`mutationConstantsSigma`	$\mathbb{R}$	1
Individual Size Limit	`individualSizeLimit`	$\mathbb{N}$	64
Error Measure	`errorMeasure`	$\mathcal{G} \to \mathbb{R}$	sample `rmse`

ponents are shown in Table 1-1. Default parameters were used for all experiments.

TinyGP

TinyGP is a popular small GP implementation mainly used in teaching. It implements a simple steady-state single-objective search heuristic with tournament selection that is loosely based on Koza's work on GP. (Koza, 1992) The TinyGP search heuristic can be seen as a deliberately minimal single-objective example for the popular class of steady-state GP search heuristics. For this reason, it was implemented in RGP and included in this study as a reference.

Selection Strategy. Tournament selection in TinyGP proceeds as follows: First, an individual is selected from the population by uniform random sampling as the current winner. This individual's fitness is then compared to competitors $s_{\text{tournament}}$ times. Each time a competitor has a better (smaller) fitness value, it takes the place as the current winner. Competitors are chosen by uniform random sampling from the entire population. Therefore, there is a non-zero probability that the same individual enters the same tournament multiple times.

The negative tournament selection operator employs the same strategy, its only difference being that the order relation $<$ is being replaced by its converse $>$, so that the worst individual taking part in the tournament is returned as result.

Algorithm Structure. In the first step of the algorithm, a population $\texttt{pop}(0)$ of μ random individuals is created. Next, the steady-state evolution process starts by randomly selecting either a recombination or a mutation operator. The probability for selecting the recombination

Table 1-2. Parameters of the TinyGP search heuristic.

	Variable (Symbol)	*Domain*	*Default*
Population Size	`mu` (μ)	$\mathbb{N}$	300
Tournament Size	`tournamentSize` ($s_{\text{tournament}}$)	$\mathbb{N}$	2
Recombination Probability	`recombinationProbability` ($p_{\texttt{rec}}$)	$[0, 1]$	0.9

operator is given by the parameter $p_{\texttt{rec}}$. In case of recombination, the algorithm selects two parents via two independent tournaments of size $s_{\text{tournament}}$. In case of mutation, a single parent is chosen in a single tournament. In both cases, a single child is creating by applying the chosen variation operator to the parent(s). Next, the algorithm chooses an individual to replace by this child in a single negative tournament of size $s_{\text{tournament}}$. This process is repeated until a predefined termination criterion is met.

Diversity Preservation. The TinyGP system does not implement any internal means of diversity preservation.

Parameters. Table 1-2 gives all parameters of the TinyGP search heuristic.

Generational Multi-Objective GP

Generational Multi-Objective GP (GMOGP) is a GP search heuristic that combines ideas of multi-objective GP search heuristics with ideas of generational multi-objective evolutionary algorithms.

Selection Strategy. The selection operator is based on non-dominated sorting (NDS) of the selection pool based on the three criteria goodness of fit on training data, genotypic complexity, and *genotypic age*. Ties in the NDS are broken by crowding distance, as in the well-established NSGA-II EMOA. (Deb et al., 2000)

The additional selection criteria can be enabled individually, yielding three configurations:

- single-objective selection based on goodness of fit (GMOGP-F)
- multi-objective selection based on goodness of fit and individual age (GMOGP-FA)
- multi-objective selection based on goodness of fit and individual complexity (GMOGP-FC)
- multi-objective selection based on goodness of fit, individual age, and genotypic individual complexity (GMOGP-FCA)

Table 1-3. Parameters of the GMOGP search heuristic.

	Variable (Symbol)	*Domain*	*Default*
Population Size	`mu` (μ)	$\mathbb{N}$	300
Children per Generation	`lambda` (λ)	$\mathbb{N}$	20
New Individuals per Generation	`nu` (ν)	$\mathbb{N}_0$	1
Enable Complexity Criterion	`complexityCriterion`	$\mathbb{B}$	`true`
Enable Age Criterion	`ageCriterion`	$\mathbb{B}$	`true`
Recombination Probability	`recombinationProbability` ($p_{\mathtt{rec}}$)	$[0, 1]$	0.1

Algorithm Structure. GMOGP is based on a generational $(\mu+\lambda)$ strategy. After creating an initial population $\mathtt{pop}(0)$ of μ random individuals, the iterative evolution process starts by choosing λ pairs of parents by uniform random sampling without replacement. Pairwise recombination is applied before mutation, yielding λ children. Next, μ individuals are chosen from the $(\mu + \lambda + \nu)$-sized set union of parents, children and ν newly initialized individuals by the Pareto selection operator detailed in the next paragraph, replacing the parent population. This iterative process is stopped when a predefined termination criterion is met.

Diversity Preservation. GMOGP implements elements of the Age-Fitness Pareto Optimization (AFPO) algorithm for preserving genotypic diversity. (Schmidt and Lipson, 2010) This diversity preservation mechanism can be disabled by setting the parameter *Age Layering* to `false`.

Parameters. Table 1-3 presents all parameters of the GMOGP search heuristic. These parameters are subject to the following constraint on the children set size:

$$\lambda \leq \left\lfloor \frac{\mu}{2} \right\rfloor$$

3. Scalable Test Functions

Test problems of controllable difficulty, i.e. *scalable test functions*, are a flexible tool for assessing the relative performance benefits of different GP system components under varying conditions. This study focuses on symbolic regression. Unfortunately, defining finely scalable test functions for symbolic regression holds many challenges. Conventional measures of problem difficulty, such as information criteria (Kiesepp, 1997), that are good predictors for the performance of fixed-structure modelling approaches like modern statistical regression techniques, often fail to predict the performance of symbolic regression runs. See Section 4 for definitions of performance measures for symbolic regression. Symbolic

regression on test functions of comparatively simple structure can prove extremely difficult for state-of-the-art symbolic regression systems. (Korns, 2011)

The framework described in this chapter implements multiple scalable test problem classes for symbolic regression, including test functions with spurious variables. This class of scalable test problems is based on the simple yet very effective idea of adding *spurious variables* to the set of fitness cases to conceal the true functional dependency between driving variables and output for arbitrary test functions. (Kotanchek et al., 2006) To increase the difficulty of a given function, additional function arguments (spurious variables) are added that do not influence the function value. As this fact is hidden from the symbolic regression system, and spurious variables might be correlated with the function value by chance, problem difficulty can be gradually increased in fine-grained steps by increasing the number of spurious variables.

Spurious Variable Test Functions

In the experiments conducted for this study, the three scalable test functions described below were used. For each test function, 1 to 10 spurious variables were added to gradually increase problem difficulty. Fitness cases were created by uniform random sampling in the indicated training and test intervals. Both training and test set consisted of the number of fitness cases indicated below.

P1 (Simple Sine). Discovering the input-output relation of the simple sine function should be trivial even for a very simple symbolic regression system, if no spurious variables are introduced.

$$f_{\mathrm{P1}}(x_1) := \sin(\pi x_1)$$

The factor π has been introduced to test the GP system's capability of fitting constants. The training interval of the Simple Sine test function is fixed to $[-\pi, \pi]$, the test interval is fixed to $[-\frac{3}{2}\pi, \frac{3}{2}\pi]$, and the number of fitness cases, i.e. the test function sample size, is fixed to $N_{\mathrm{P1}} := 32$. The GP function set used with this test function is $\{+, -, *, /, \sin\}$.

P2 (Newton Problem). Compared to the Simple Sine, the Newton Problem poses a slightly harder test case, as the true expression has slightly larger genotypic size. It was included in this study as a more practical example based on a well known natural law.

$$f_{\mathrm{P2}}(x_1, x_2, x_3) := \frac{x_1 x_2}{x_3^2}$$

The training interval of the Newton Problem test function is $(0, 1]$, the test interval $(0, 2]$, and the number of fitness cases is fixed to $N_{\mathrm{P2}} := 64$. The GP function set used with this test function is $\{+, -, *, /\}$.

P3 (Sine Cosine). This test function is a simplified variant of test problem *(P12)* given in Michael F.Korn's work on accuracy in symbolic regression. (Korns, 2011) It is considered difficult because it constraints constants as arguments to non-linear functions.

$$f_{\mathrm{P3}}(x_1, x_2) := 6 \sin(x_1 - 3) \cos(x_2 - 3)$$

The training interval of the Sine Cosine test function is $[-\pi, \pi]$, the test interval $[-\frac{3}{2}\pi, \frac{3}{2}\pi]$, and the number of fitness cases is fixed to $N_{\mathrm{P3}} := 128$. The GP function set used is $\{+, -, *, /, \sin, \cos\}$.

4. Experimental Setup

Following the experimental framework presented by Bartz-Beielstein, 2006, a sound experimental setup requires specification of the (1) optimization problem, (2) performance measure, (3) initialization method, and (4) termination method.

Optimization Problems in GP

Optimization problems in GP have two components: (a) a test function, say f, and (b) a measure, which determines the distance between the true function f and its GP approximation, say $\hat{f}$. Test functions were presented in Section 3. The measure, which will be used in our comparisons can be described as follows: Consider an input value, say x_i, a prediction model $\hat{f}(x)$, and the true value y_i. The prediction model $\hat{f}$ is estimated by GP from a training sample. The *loss function*, which measures the errors between the y_i's and the $\hat{f}(x_i)$'s, is defined as

$$L(y, \hat{f}(x)) = (y - \hat{f}(x))^2.$$

The *training error* is the square root of the average loss over the training sample

$$\sqrt{\frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i))},$$

where N denotes the number of fitness cases as introduced in Section 3. This value depends on the test function, e.g., $N = 32$ was chosen for f_{P1}.

1 During the pre-experimental planning phase, a large number of function evaluations n_0, the so-called *budget*, is used the determine an adequate percentage of GP runs that reach a pre-defined (small) distance to the optimum.
2 *Run length distributions* (RLDs), as introduced by Parkes and Walser, 1996, are generated to estimate a suitable budget $n \leq n_0$ for the actual experiments.

To avoid floor and ceiling effects, we determine an adequate problem design, e.g., number of function evaluations as well as reasonably chosen training, validation, and test sets.

Performance Measures

To avoid overfitting, the *test error* $E[L(y_t, \hat{f}(x_t)]$, i.e., the expected prediction error over an independent test sample x_t, is preferred. The comparison of the algorithm designs is based on these test-set data. In addition to our considerations, we will sketch out further enhancements of our testing framework: Typically, models will have tuning parameters, say α, so we can describe predictions as $\hat{f}_\alpha(x)$. In forthcoming steps of our experimental analysis, we are interested in finding good α values while avoiding overfitting, i.e., two different problems are to be solved: First, estimating the performance of different models $\hat{f}_\alpha$ in order to chose the best model, and second, estimating its prediction error on new data. If sufficient data is available, the data set can be partitioned into a *training set* x_{train}, a *validation set* x_{val}, and a *test set* x_{test}.

Initialization and Termination Methods

Classical ramped half-and-half initialization was used in all experiments. (Koza, 1992) To determine suitable termination methods, RLDs were generated. The number of fitness case evaluations, i.e., the number of fitness cases times the number of test function evaluations was chosen as the termination criterion.

Research Questions

TinyGP is considered as a reference algorithm, which implements essential GP features. A common belief in the GP community can be stated as follows:

Scientific Claim 1 *Complex problems require complex algorithms.*

Or, stated differently: TinyGP can solve tiny problems whereas hard problems require more complex algorithms.

From a naive point of view, Claim 1 goes without saying—one may simply consider that the opposite assumption is true. Therefore, it can be taken as a guideline for performing experiments and present results that are based on solid scientific and statistical assumptions. We will proceed as follows: In order to perform a sound statistical analysis, we will define a hierarchy of complex (hard) problems. Scalable test functions as introduced before are well suited for our goals: increasing the number of spurious variables should decrease the success rate of the GP system.

Then we will define a reference GP implementation that comprehends the essential features of GP systems in a very simple and understandable manner. The TinyGP system is considered as an ideal candidate, because it is well-known, easy to obtain, and easy to implement.

Pre-Experimental Planning

During the first stage of our experiments, no parameter tuning will be performed. Our main goal is to discover (positive) correlations between algorithm complexity and problem difficulty using default algorithm parameter settings.

First experiments were set up to calibrate the reference algorithm, i.e., TinyGP, to the problem.

Statistical Hypothesis 1 (H-1) *TinyGP requires more function evaluations to reach the same success rate if the problem complexity increases.*

RLDs are suitable means to measure performance and to determine an adequate budget. A large number of function evaluations, i.e. a budget of $n_0 = 1e6$, was chosen to generate the RLDs. The set of test functions consists of $\{f_{\text{P1}}, f_{\text{P2}}, f_{\text{P3}}\}$.

The investigation of H-1 has two significant results: First, we ensure that the reference algorithm is able to solve this problem (at least it should be able to improve an existing candidate solution). Second, the correct number of test function evaluations for comparisons is determined. We are able to detect test functions which are far too easy (or too hard) for the reference algorithm. The case, that the reference algorithm is unable to solve the test function needs further considerations.

Next, we introduce the baseline variant of the new algorithm, i.e., GMOGP-F, which should be analyzed.

Statistical Hypothesis 2 (H-2) GMOGP-F *is competitive with the reference algorithm.*

To analyze H-2 the same setup as for H-1 was used.

These two series of experiments belong to the pre-experimental planning phase, because they are needed to set up fair comparisons. If one hypothesis has to be rejected, the experimental set up should be reconsidered, e.g., the set of test function should be modified or the computational budget should be increased.

Experiments

The third series of experiments is performed to analyze the influence of new GP components on the algorithm's performance. Here, we will analyze new selection schemes, namely aging, complexity, and a combination of these two.

Statistical Hypothesis 3 (H-3) *Introducing multi-objective selection based on goodness of fit and individual age improves the* GMOGP-F *performance.*

To investigate H-3, the following setup was used. The set of test functions consists of $\{f_{\text{P1}}, f_{\text{P2}}\}$, and a budget of $n_0 = 250,000$ was used. The age criterion as described in Table 1-3 was set to `true`. With respect to the experimental setup for hypothesis H-2, the remaining parameters remained unmodified.

Statistical Hypothesis 4 (H-4) *Introducing multi-objective selection based on goodness of fit and individual complexity improves the* GMOGP-F *performance.*

To investigate H-4, a similar setup as for hypothesis H-3 was used. The complexity criterion as described in Table 1-3 was set to `true`. With respect to the experimental setup for hypothesis H-2, the remaining parameters remained unmodified.

Statistical Hypothesis 5 (H-5) *Introducing multi-objective selection based on goodness of fit, individual age and individual complexity improves the* GMOGP-F *performance.*

To investigate H-5, a similar setup as for hypothesis H-3 was used. The complexity as well as the age criterion (Table 1-3) was set to `true`. With respect to the experimental setup for hypothesis H-2, the remaining parameters remained unmodified.

5. Results

To generate the experimental data needed to test the statistical hypotheses established in Section 4, experiments for each hypothesis were performed.

Table 1-4. Run length distributions at 250,000 fitness function evaluations.

Search Heuristic	*Test Function*	*Difficulty (# Spurious Variables)* 1	2	3	4	5	6	7	8	9	10
TinyGP	Simple Sine	95	75	70	65	55	55	50	50	55	30
	Newton Problem	80	80	60	70	60	35	40	45	35	10
	Sine Cosine	10	20	20	15	10	0	5	15	5	5
GMOGP-F	Simple Sine	100	100	100	100	100	100	100	95	100	95
	Newton Problem	90	90	90	80	50	75	65	55	40	30
GMOGP-FA	Simple Sine	100	100	100	100	100	95	100	100	100	100
	Newton Problem	100	100	85	85	80	95	80	75	65	55
GMOGP-FC	Simple Sine	100	85	100	80	65	85	75	90	75	70
	Newton Problem	65	60	30	30	15	10	20	15	0	5
GMOGP-FCA	Simple Sine	100	100	100	100	100	95	95	95	100	90
	Newton Problem	95	80	75	50	30	50	60	55	35	35

Statistical Hypothesis H-1. Table 1-4 and Figure 1-1 show that TinyGP's success rates decrease as problem complexities increase. As the algorithm can get stuck in local optima and basing RLD calculation on test data instead of training data adds additional noise, this decrease is not strictly monotonic. Nonetheless, data do not indicate that H-1 has to be rejected. During pre-experimental planning, the Sine Cosine test problem proved too difficult for the reference algorithm (TinyGP), and was therefore excluded from the main experiments.

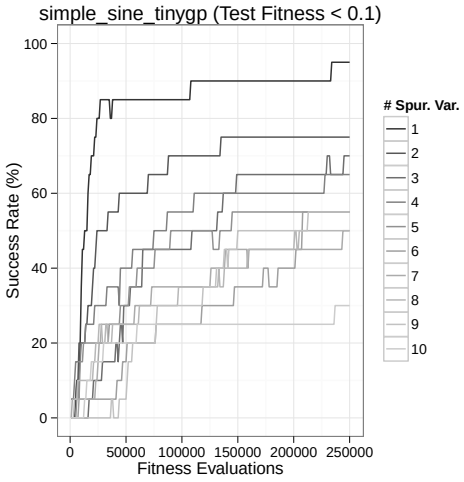
Statistical Hypothesis H-2. On both test functions studied, GMOGP-F shows slightly better performance than the reference, as visible in Table 1-4. Hypothesis H-2 does not have to be rejected. This concludes the analysis of the pre-experimental planning phase.

Statistical Hypothesis H-3. As visible in Table 1-4, as well as in Figure 1-1, introducing age as a secondary optimization criterion (GMOGP-FA) significantly increases algorithm performance on the Newton Problem test function. Hypothesis H-3 can not be rejected.

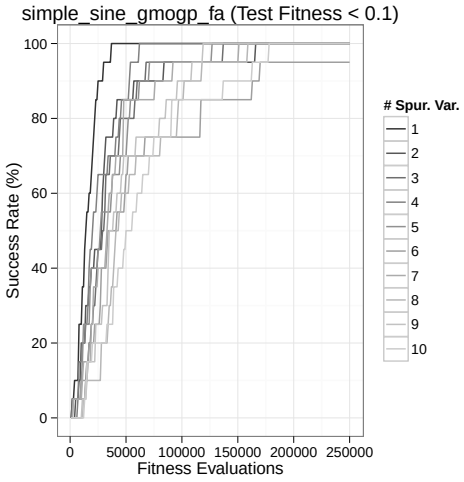
Statistical Hypothesis H-4. In contrast, introducing complexity as a secondary optimization criterion (GMOGP-FC) degrades algorithm performance in this experimental framework, as visible in Table 1-4. The hypothesis H-4 has to be rejected.

Statistical Hypothesis H-5. Introducing both complexity and age as additional optimization criteria (GMOGP-FCA) also degrades algorithm performance in comparison with the single-objective case (GMOGP-F), but not so much as GMOGP-FC. Hypothesis H-5 has to be rejected.

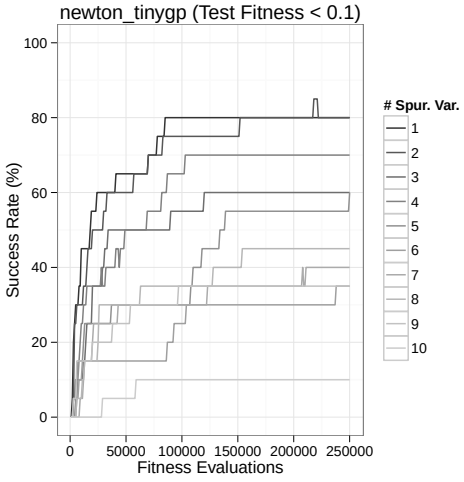
Figure 1-1. Run length distribution plots: The number of fitness function evaluations are shown on the x-axis, algorithm success rates are shown on the y-axis. Each plot shows 10 difficulty levels (number of spurious variables) of a single test function for a single search heuristic. For each combination of search heuristic, test function, and difficulty level, 20 independent GP runs were performed. The plots also illustrate that harder test functions require more function evaluations to reach the same success rates as simpler test functions.



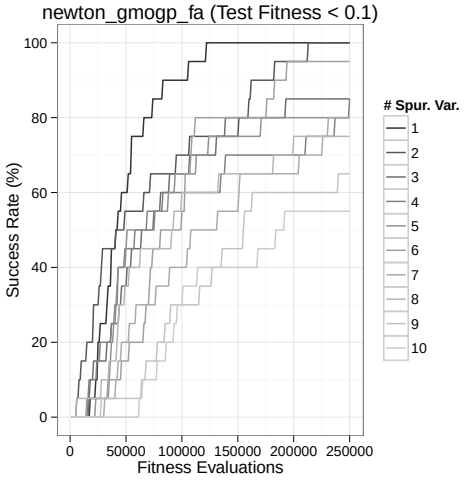
(a) Simple Sine test function, TinyGP search heuristic



(b) Simple Sine test function, GMOGP-FA search heuristic



(c) Newton Problem test function, TinyGP search heuristic



(d) Newton Problem test function, GMOGP-FA search heuristic

6. Discussion and Conclusions

In relation to hypotheses H-1 and H-2, it became clear that the scalable test function set based on the introduction of spurious variables works as expected on all studied search heuristics. TinyGP could be established as a practical reference algorithm. Therefore, a conceptual framework for statistically well-founded comparisons of the relative performance benefits of GP system components is available. An R prototype realizing this framework in software is under development.

Within this framework, experiments show that GMOGP-F is competitive with TinyGP, i.e. that a simple generational search heuristic is a viable replacement for steady-state heuristics more common in GP. The RLD plots of this chapter illustrate that harder test functions, i.e. test functions with more spurious variables, require more function evaluations to reach the same success rates as simpler test functions. Introducing an age criterion (GMOGP-FA) improves GP performance significantly. On the other hand, the introduction of a complexity criterion (GMOGP-FC and GMOGP-FCA) does not result in the expected performance gain, contrary to results from literature. This may indicate an implementation or parameterization problem, demanding further investigation. As all experiments were conducted with default parameters, parameter optimization via SPO should help to provide better parameter settings.

7. Outlook

Only basic features of the test framework were demonstrated in this chapter. Many extensions are available, including tools for GP system parameter optimization.

In further work, the usefulness of additional scalable test function classes will be analyzed. A study on automatic parameter tuning for TinyGP, GMOGP and DataModeler based on SPOT will be conducted to investigate the sensitivity of GP systems to their parameter settings. This statistical analysis requires enhanced experimental designs which are subject of our research.

In summary, the hypotheses-driven approach encouraged by this framework should lead to statistically validated results of high reproducibility. In the future, this framework will be applied to study other GP systems based on a larger set of realistic scalable test problem classes.

Acknowledgements. This work was supported the Bundesministerium für Bildung und Forschung (BMBF) under the grant FIWA (AiF

FKZ 17N2309). Many thanks to Boris Naujoks, Tobias Brandt, and Jörg Stork for valuable ideas and suggestions.

References

Bartz-Beielstein, Thomas (2006). *Experimental Research in Evolutionary Computation—The New Experimentalism*. Natural Computing Series. Springer, Berlin, Heidelberg, New York.

Bartz-Beielstein, Thomas (2010). SPOT: An R package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. CIOP Technical Report 05/10, Research Center CIOP (Computational Intelligence, Optimization and Data Mining), Cologne University of Applied Science.

Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T. (2000). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Schoenauer, M. et al., editors, *Proceedings of the Parallel Problem Solving from Nature VI*, pages 849–858, Berlin, Heidelberg, New York. Springer.

Flasch, Oliver, Mersmann, Olaf, and Bartz-Beielstein, Thomas (2010). RGP: An open source genetic programming system for the R environment. In Pelikan, Martin and Branke, Jürgen, editors, *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon*, pages 2071–2072. ACM.

Kiesepp, I. A. (1997). Akaike information criterion, curve-fitting and the philosophical problem of simplicity. *British Journal for the Philosophy of Science*, 48:21–48.

Korns, Michael F. (2011). Accuracy in symbolic regression. In Riolo, Rick, Vladislavleva, Ekaterina, and Moore, Jason H., editors, *Genetic Programming Theory and Practice IX*, Genetic and Evolutionary Computation, pages 129–151. Springer New York.

Kotanchek, Mark, Smits, Guido, and Vladislavleva, Ekaterina (2006). Pursuing the pareto paradigm tournaments, algorithm variations & ordinal optimization. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice IV*, volume 5 of *Genetic and Evolutionary Computation*, chapter 12, pages 167–186. Springer, Ann Arbor.

Kotanchek, Mark, Smits, Guido, and Vladislavleva, Ekaterina (2007). Trustable symoblic regression models. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, editors, *Genetic Programming Theory and Practice V*, pages 203–222.

Koza, John R. (1992). A genetic approach to the truck backer upper problem and the inter-twined spiral problem. In *Proceedings of IJCNN*

International Joint Conference on Neural Networks, volume IV, pages 310–318. IEEE Press.

Parkes, A. J. and Walser, J. P. (1996). Tuning local search for satisfiability testing. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI'96)*, pages 356–362.

Poli, R. (2004). TinyGP. See TinyGP GECCO 2004 competition at http://cswww.essex.ac.uk/staff/sml/gecco/TinyGP.html.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Schmidt, Michael D. and Lipson, Hod (2010). Age-fitness pareto optimization. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 543–544, New York, NY, USA. ACM.

Smits, G. and Vladislavleva, E. (2006). Ordinal pareto genetic programming. In Yen, Gary G. et al., editors, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 3114–3120, Vancouver, BC, Canada. IEEE Press.

Vladislavleva, Ekaterina (2008). *Model–based Problem Solving through Symbolic Regression via Pareto Genetic Programming*. PhD thesis, Tilburg University.