Clustering Based Niching for Genetic Programming in the R Environment

Oliver Flasch, Thomas Bartz-Beielstein, Patrick Koch, and Wolfgang Konen

Fakultät für Informatik und Ingenieurwissenschaften, Fachhochschule Köln
E-Mail: {oliver.flasch | thomas.bartz-beielstein | patrick.koch | wolfgang.konen}@fh-koeln.de

Abstract

In this paper, we give a short introduction into RGP, a new genetic programming (GP) system based on the statistical package R. The system implements classical untyped tree-based genetic programming as well as more advanced variants including, for example, strongly typed genetic programming and Pareto genetic programming. The main part of this paper is concerned with the problem of premature convergence of GP populations, accompanied by a loss of genetic diversity, resulting in poor effectiveness of the search. We propose a clustering based niching approach to mitigate this problem. The results of preliminary experiments confirm that clustering based niching is effective in preserving genetic diversity in GP populations.

1 Introduction

The goal of this paper is twofold: first, it provides a short introduction into RGP, a new open source genetic programming system implemented as an extension package for the statistical language and software package R [1]. Second, it reports on our ongoing research in diversity preservation methods for tree-based genetic programming (GP).

GP is a class of evolutionary algorithms for the automatic generation of computer programs from high-level problem definitions [2, 3, 4]. In our work, we apply a strongly-typed tree-based multi-objective GP variant for symbolic regression to solve real-world time series regression problems from water resource management and from finance [5, 6]. We recently developed RGP, a modular GP system based on the R statistical package. By building on R, our system can leverage extensive tools for statistics, data handling, and visualization. Because GP individuals are represented as R functions, arbitrary R functions may be used as GP building blocks. Furthermore, GP individuals can be used and analyzed by all available R-based tools. By distributing our system as open source software, we enable others to verify the results of our experiments and to apply our algorithms and techniques to their problems and datasets.

In our application of GP, we experienced the common problem of premature convergence into local optima, accompanied by a loss of genetic diversity, resulting in poor effectiveness of the search. This problem is perhaps more severe in GP for symbolic regression than in other evolutionary algorithms or GP applications because of the highly difficult structure of the symbolic regression GP search space: depending on the set of GP building blocks (the function set), the fitness landscape is highly multi modal and rough, making it especially difficult to escape from local optima.

A typical means to circumvent the impact of premature convergence is to use multiple independent runs in parallel, combining their results only after a certain number of fitness evaluations m have passed [7]. The hyperparameter m can be fixed at the start of a run or controlled dynamically. This algorithm is a form of static niching, where genetic diversity is preserved by isolating possibly structurally, i.e., genotypically, different solutions from direct competition [8, 9]. Multi-objective GP methods, such as Pareto GP, also result in an implicit form of niching by maintaining a set of Pareto-optimal solutions that may be structurally different. Furthermore, explicit static niching schemes based on trivial geographies, demes or islands, or on individual age have shown promising potential in several GP applications [10, 3, 11, 12].

The remainder of this paper is organized as follows: Section 2 gives a high-level introduction into the design and features of the RGP system. Section 3 motivates and describes clustering based niching in the abstract and, building on this description, presents our algorithm. Section 4 explains preliminary experiments and gives first results. Section 5 concludes this paper with a short summary and an outlook to further research.

2 Genetic Programming in the R Environment

The recent availability of fast multi-core systems has enabled the practical application of GP in many real-world application domains. This has lead to the development of software frameworks for GP, including DataModeler, Discipulus, ECJ, Eurequa, and GPTIPS[1].

All of these systems are complex aggregates of algorithms for solving not only GP specific tasks, such as solution creation, variation, and evaluation, but also more general Evolutionary Computation (EC) tasks, like single- and multi-objective selection, and even largely general tasks like the design of experiments, data pre-processing, result analysis and visualization. Packages like Matlab, Mathematica, and R already provide solutions for the more general tasks, greatly simplifying the development of GP systems based on these environments [1].

RGP[2] is based on the R environment for several reasons. First, there seems to be a trend towards employing statistical methods in the analysis and design of evolutionary algorithms, including modern GP variants [13, 14]. Second, R's open development model has led to the free availability of R packages for most methods from statistics and many methods from EC. Also, the free availability of R itself makes RGP accessible to a wide audience. Third, the R language supports "computing on the language", which greatly simplifies symbolic computation inherent in most GP operations. In addition, parallel execution of long-running GP experiments is easily supported by R packages such as Snow [15].

[1] DataModeler is a commercial Mathematica-based GP system focused on symbolic regression in industrial applications (`evolved-analytics.com`). Discipulus is a commercial linear GP system (`www.rmltech.com`). ECJ is an open source framework for evolutionary computation (`cs.gmu.edu/~eclab/projects/ecj`). Eurequa is a graph GP system optimized for symbolic regression (`ccsl.mae.cornell.edu/eureqa`). GPTIPS is an open source Matlab toolbox for symbolic regression by GP (`sites.google.com/site/gptips4matlab`).

[2] The RGP package and documentation is freely available at `rsymbolic.org`.

2.1 RGP Features

RGP was mainly developed as a research tool for exploring time series regression and prediction problems with GP. Nevertheless, the system is modular enough to be easily adapted and extended to new application domains.

Individual Representation RGP represents GP individuals as R expressions that can be directly evaluated by the R interpreter. This allows the whole spectrum of functions available in R to be used as building blocks for GP. Because R expressions are internally represented as trees, RGP may be seen as a tree-based GP system. However, the individual representation can be easily replaced together with associated variation and evaluation operators, if an alternative representation is found to be more effective for a given application [16].

Besides classical untyped GP, strongly typed GP is supported by a type system based on simply typed lambda calculus [17]. A distinctive feature of RGP's typed tree representation is the support for *function defining subtrees*, i.e. anonymous functions or lambda abstractions. In combination with a type system supporting function types, this allows the integration of common higher order functions like folds, mappings, and convolutions, into the set of GP building blocks.

RGP also includes a rule based translator for transforming R expressions. This mechanism can be used to simplify GP individuals during the evolution process as a means the reduce bloat, or just to simplify solution expressions for presentation. The default rule base implements simplification of arithmetic expressions. It can be easily extended to simplify expressions containing user-defined operators and functions.

GP Operators RGP provides default implementations for several initialization, variation, and selection operators. The system also provides tools for the analysis and visualization of populations and GP individuals.

Initialization Individual initialization can be performed by the conventional *grow* and *full* strategies of tree building. When using strongly-typed GP, the provided individual initialization strategies respect type constraints and will create only well-defined expressions. Initialization strategies may be freely combined, e.g. to implement the well known *ramped-half-and-half* strategy [3].

Variation RGP includes classical and type-safe subtree crossover operators. Also, several classical and type-safe mutation operators are provided. The variation step can be freely configured by combining several mutation and recombination operators to be applied in parallel or consecutively, with freely configurable probabilities.

Selection The system provides an implementation of single-objective tournament selection with configurable tournament size. Other selection strategies can be easily added and will be provided in later versions. Additionally, multi-objective selection is supported via the EMOA package by implementing the Pareto GP algorithm [7]. This algorithm optimizes solution quality while, at the same time, controlling solution complexity. For this purpose, RGP implements multiple complexity measures for GP individuals.

3 Clustering Based Niching

Preserving genetic diversity in the population of an evolutionary algorithm (EA) is important for reaching two related goals: first, high diversity is a resource for exploratory crossover, helping in the discovery of multiple optima in multi-modal search spaces. Second, high diversity decreases the probability of the whole population converging to a local optimum.

Most diversity-preserving EA are based on altering the selection operator to prevent premature convergence to a local optimum, examples include Fitness Sharing, Crowding, and Tagging [18, 19, 20]. A different approach is to use multiple independent populations or niching, like in Multinational GA and Forking GA [21, 22].

The underlying idea of niching in evolutionary algorithms is to apply the biological concept of isolated non-interbreeding species living in separate ecological niches to preserve genetic diversity in EA populations. Individuals of different species do not interbreed and individuals living in different niches do not compete for the same resources. Each niche is implemented as an independent EA run, convergence of the species in a single niche has no influence on the genetic diversity of other species in other niches. Inside each niche, individuals breed and compete like in a traditional EA, converging to a local optimum. This has the benefit that all possible EA extensions and specialized EA operators can still be applied at the niche level.

Given a technique for dynamically creating and merging niches, speciation can occur. By merging species, the higher genetic diversity of the resulting species can be exploited as a resource for exploratory crossover. By splitting a species into separate niches, the now isolated sub-species can evolve independently, creating new genetic diversity. In clustering based niching, clustering algorithms are used to divide an initial global population into species which are then distributed to niches. Clustering algorithms group individuals into species so that the individuals within the same species are relatively similar, while individuals in different species are relatively distinct. The similarity of individuals can be measured in several different ways, leading to different clusterings and different algorithm behaviour.

Clustering based niching methods have shown to be effective in preserving genetic diversity in Evolution Strategy (ES) populations [23]. In this work, we apply clustering based niching to GP for the first time. This leads to our main hypotheses:

H1 Niching is effective in preserving genetic diversity in GP populations.

H2 Clustering based niching yields significantly better results than static niching when applied to GP for symbolic regression.

If these hypotheses can be accepted is not clear, mainly because of the difficult nature the GP search space for symbolic regression. Table 1 shows the results of an experiment designed to highlight this difficulty. The phenotypic distance (see Section 3.1) between a GP individual and its mutated variant after a single mutation step of three different standard GP mutation operators is given. The phenotypic effect of a single mutation is highly random. Mutating constants in an GP individual can change its behaviour and therefore its fitness significantly, the effect of mutating function nodes or replacing subtrees is even

stronger. These observations tell us that GP individuals might be unrelated phenotypically, even if they have very similar genotypes. We study empirically if clustering based niching is effective under these conditions.

Table 1: Effect of standard genetic programming mutation operators: in each experiment, a random full tree T_A of depth 6 is generated and the phenotypic distance to another tree T_B is measured. In mutation type (Mut. Type) *Random*, T_B is another random full tree of depth 6. In *Const. Mut.*, T_B is created by mutating constants in T_A. In *Func. Mut.*, T_B is created by mutating function labels in T_A. In *Subtree Mut.*, T_B is created by replacing random subtrees of T_A. The results shown are summaries of 500 experiments for each mutation type.

	Phenotypic Distance (RMSE)					
Mut. Type	*Min.*	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*
Random	0	30.4959	197.2396	1.9161×10^6	1.3951×10^3	5.8326×10^8
Const. Mut.	0	0.0030	1.1040	5.5765×10^3	17.2065	7.8703×10^5
Func. Mut.	0	0.4273	19.9414	3.6973×10^7	408.7947	1.6742×10^{10}
Subtree Mut.	0	6.8196	78.2556	2.7572×10^5	1.1684×10^3	4.5585×10^7

What is the expected advantage of clustering based niching in contrast to a conventional static niching approach? In contrast to fixed niching, in clustering based niching individuals inside a single niche can be expected to be more similar than individuals of two different niches. This could be beneficial because successful crossover is much more likely between similar individuals, i.e. individuals which share homologous subtrees.

The general scheme of clustering based niching GP consists of four steps:

1. Cluster the GP population P_i into N_i niches.
2. Perform parallel GP passes in each of the N_i niches until the pass stop criterion is met at each niche.
3. Join the N_i populations into the GP population P_{i+1}, adding the best performing individuals into an elite set E.
4. Unless the run stop criterion is met, set $i := i + 1$ and return to step 1, otherwise return the resulting population P_{i+1} and the elite set E.

3.1 Distance Measures for Clustering GP Populations

In order to perform a clustering of a GP population into niches, a distance measure for GP individuals is needed. This distance measure is then used to calculate a distance matrix which can be used as an input for several hierarchical or partitioning clustering algorithms. There are basically three classes of distance measures that are suitable for this purpose: *Fitness distance*, *phenotypic distance*, and *genotypic distance*. Each of these measures has distinct requirements, advantages, and drawbacks.

Fitness Distance Given two GP individuals T_A and T_B, a (possibly multi-objective) fitness function f and a distance measure d, the Fitness Distance d_{fitness} between T_A and T_B is given by the formula $d_{\text{fitness}} := d(f(T_A), f(T_B))$. Trivially, d_{fitness} is a metric iff d is a metric. Typically, the Euclidean metric is used for d, but other choices are possible. For example, the Euclidean Squared metric is sometimes used for efficiency reasons.

The main benefit of d_{fitness} is that it is comparatively efficient to calculate if the values of the fitness function f are already known for most individuals in a population, as it is the case in later stages of a GP run. At the start of a GP run, fitness values must be expected to be mostly random, with a high (when fitness values are minimized) mean, making d_{fitness} unsuitable for meaningful clustering in this stage of a run.

Phenotypic Distance The phenotypic distance $d_{\text{phenotype}}$ measures the behavioral difference between two GP individuals T_A and T_B. Its definition is highly dependent on the interpretation of GP individuals, i.e. the GP application. In symbolic regression, given an error measure ϵ and a set of fitness cases $F \subseteq D$, we define $d_{\text{phenotype}} := \epsilon([\![T_A]\!](F), [\![T_B]\!](F))$. The operator $[\![\cdot]\!]$ interprets a GP individual tree as a function defined on a domain D. In practice, typically the mean square error (MSE) or the root mean square error (RMSE) are used as an error measure ϵ.

Phenotypic distance can give meaningful clusterings in every stage of a GP run. Its main drawback is its possibly high computational effort, depending on the mean computational cost of calculating $[\![\cdot]\!](F)$.

Genotypic Distance As a GP population is a set of expression trees, there is no "natural" genotypic distance measure. To give meaningful and stable clusterings, a genotypic distance measure d_{genotype} should reflect the impact of the GP variation operators in the (informal) sense that $d_{\text{genotype}}(T_A, T_B) \leqslant d_{\text{genotype}}(T_A, T_C)$ iff $P(\text{variate}(T_A) = T_B) \geqslant P(\text{variate}(T_A) = T_C)$: an individual T_A is closer to an individual T_B than it is to an individual T_C iff the probability of arriving at T_B by a fixed number of stochastic variation steps from T_A is higher than the probability of arriving at T_C. Finding an appropriate d_{genotype} that is also efficiently computable is a difficult problem, particularly in the presence of a crossover operator.

We employ *norm-induced tree distance metrics* as genotypic distance measures because of their flexibility. A norm-induced tree distance metric $\delta_{(p,d)}$ uses a norm p defined on expression trees and a metric d on tree node labels to induce a metric on expression trees T_A and T_B: if both T_A and T_B are empty trees, $\delta_{(p,d)}(T_A, T_B) := 0$. If exactly one of T_A and T_B is empty, assume without loss of generality T_A is empty, then $\delta_{(p,d)}(T_A, T_B) := p(T_B)$. If neither T_A or T_B is empty, the difference of their root node labels $d(\text{root}(T_A), \text{root}(T_B))$ is added to the sum of the differences of the children as measured by $\delta_{(p,d)}$. The children lists are padded with empty trees to equalize their lengths.

In the remainder of this paper, we use $\delta_{(p_{\text{vl}}, d_{\text{discrete}})}$ as our implementation of d_{genotype}, where d_{discrete} is the discrete metric and p_{vl} is the expression visitation length norm [24, 25]. The expression visitation length is a fine-grained expression complexity measure which is calculated for each (sub-)expression as part of the Pareto GP algorithm. When using Pareto GP, we are therefore able to calculate $\delta_{(p_{\text{vl}}, d_{\text{discrete}})}$ more efficiently by reusing individual complexity information we already calculated.

3.2 Dynamic Restart of GP Passes

As in each niche, we perform a standard GP run (a GP pass), the pass might fail in the sense that the niche population converges into a local optimum of unsatisfactory fitness. We implement a dynamic restart strategy to mitigate this problem [26]. Many different criteria for detecting convergence are conceivable, such as monitoring the standard deviation of the best, median, or mean fitness in a fixed time window during the run and triggering restarts once the standard deviation drops below a certain fixed limit given as an algorithm parameter.

We take a different approach to detect convergence by taking advantage of the distance measures we defined for clustering GP populations (see Section 3.1). This enables us to reuse a niche's sub-matrix of the distance matrix calculated for population clustering to derive a convergence criterion for that niche. We consider a niche to be converged if its median genotypic distance has dropped below a certain fixed limit given as an algorithm parameter. When triggering a restart, the fittest individual in a niche is saved into an elite set, then the niche is reinitialized with newly generated random individuals.

3.3 Clustering Based Niching with Dynamic Restart

After all components of our clustering based niching GP algorithm have been described in the abstract, this section shows how these components are integrated into our concrete RGP implementation. The description follows the general scheme given in Section 3.

Figure 1 shows an outline of the algorithm. After creating an initial population of GP individuals using the standard ramped half-and-half initialization strategy, the population is clustered into N_i niches using the R implementation of Ward's agglomerative hierarchical clustering algorithm "`hclust`" [3, 27]. [3] The clustering can be based on any distance measure described in Section 3.1. Due to time constraints, we only use the genotypic distance measure $\delta_{(p_{\text{vl}},d_{\text{discrete}})}$ in this paper.

In the next step, the N_i niches are distributed to a compute cluster where each compute node performs an isolated GP run (a GP pass) on its niche population. Figure 2 shows a flow diagram of a single GP pass in a niche $k \in [1, N_i]$. Starting at the initial niche population, a standard GP step consisting of tournament selection, mutation, and crossover is performed. Based on either fitness distance, phenotypic distance, or genotypic distance, a diversity measure for the niche is calculated. In this paper, we use the niche's median of the genotypic distance $\delta_{(p_{\text{vl}},d_{\text{discrete}})}$ as a diversity measure. If this measure drops below a fixed limit given as an algorithm parameter, the niche population is reinitialized with new individuals created by the same strategy as the initial population. Before, the fittest existing individual in the niche is saved to a fixed size niche local elite set E_k, replacing surplus individuals based on their relative fitness. This process is repeated until a pass stop criterion holds. When the pass ends, the best individuals of the current niche population are joined into the niche population's local elite set E_k.

After all parallel passes have met their stop conditions, the pass results, i.e. the niche populations and elite sets, are joined. All niche populations are again combined into a

[3] For practical reasons, N_i is fixed in the current implementation to simplify distribution to a compute cluster. Yet in principle a dynamic number of clusters could be used, which would lead to a different N_i for each clustering pass i.

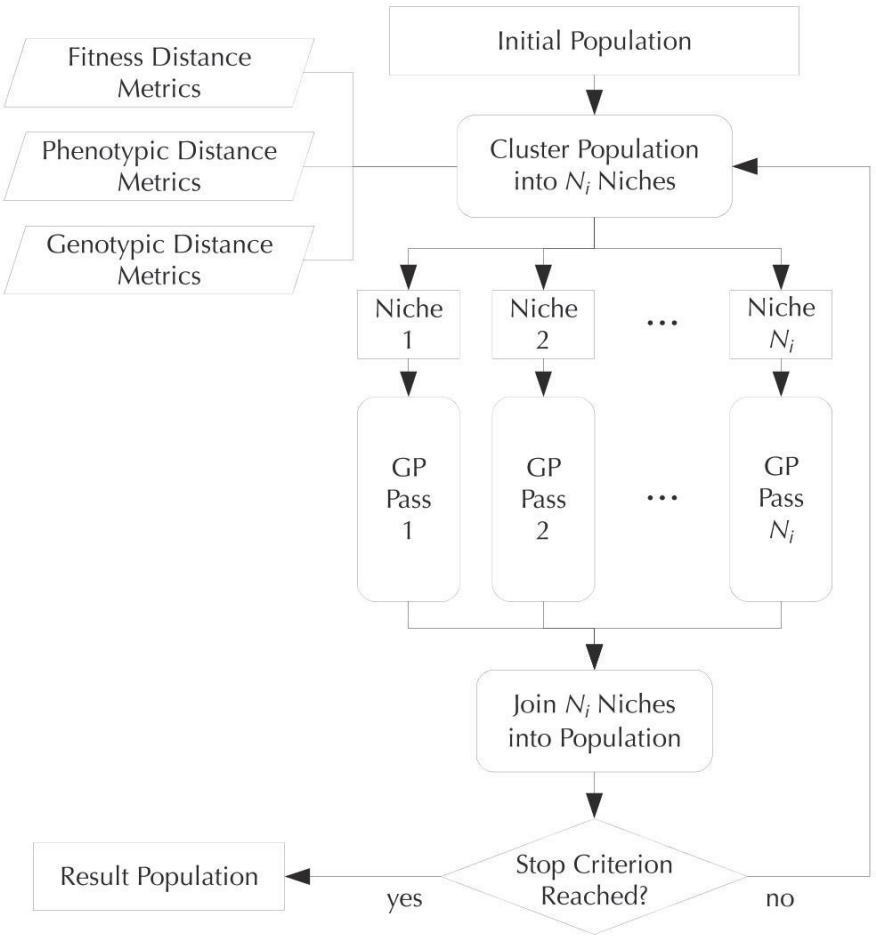

Figure 1: Outline of the distributed clustering based niching with restarts algorithm: an initial population is clustered into N_i niches, based on fitness distance, phenotypic distance, or genotypic distance. The niches are distributed to a compute cluster where each node performs an isolated GP run (a GP pass) on its niche population. The pass results are collected from the cluster nodes and joined into a single population. This process is repeated until a run stop criterion holds.

single population. All niche elite sets are joined into a fixed size global elite set E stored as a separate part of the population, where surplus individuals are replaced based on their relative fitness. If a global stop criterion holds, the current combined population and the global elite set E is returned as the algorithm's result. Otherwise, the process is repeated, starting at the clustering step.

4 Preliminary Experiments

In a set of preliminary experiments, we compared the performance of our algorithm as described in Section 3.3 (referred to as CBNGP in the remainder of this paper) with standard GP (SGP) and with a static fixed niche strategy (FNGP). SGP uses a single global population without any niching. In contrast to CBNGP, FNGP randomly divides the initial population into N_i fixed niches of equal size at the start of the run. That division is never changed until the end of the run. In this respect, FNGP resembles basic multi-deme or island strategies often employed in GP [3, 11]. All algorithms are granted a

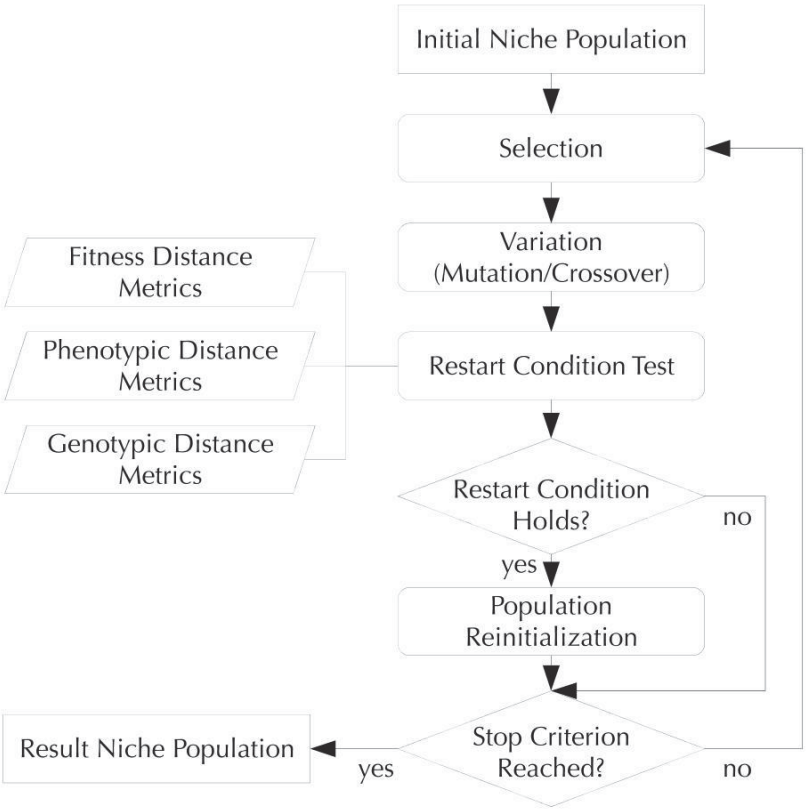

Figure 2: Outline of a single pass of the clustering based niching with restarts algorithm: Starting from an initial niche population, a standard GP step consisting of tournament selection and variation (mutation and crossover) is performed. Based on fitness distance, phenotypic distance, or genotypic distance, a diversity measure for the niche population is calculated and used to test a restart condition. If this condition holds, the niche population is reinitialized with new individuals, while the fittest existing individual is saved into a niche local elite set, otherwise the niche population is not changed. This process is repeated until a pass stop criterion holds. After the pass ends, the best individuals of the current niche population are joined into the niche population's elite set.

budget of 300,000 fitness evaluations, all algorithms use the same parameter settings as far as applicable, all algorithms use the same set of random seeds, and all algorithms use the same restart condition. Note that in our experiments FNGP and CBNGP can run 10 GP passes in parallel while SGP relies on a single sequential run, resulting in an about tenfold lower total runtime of FNGP and CBNGP in relation to SGP.

The parameter settings used in our experiment runs are shown in Table 2. Parameters that only apply to FNGP and CBNGP are marked by the phrase "FNGP and CBNGP only".

We used the following three univariate test functions, which are plotted in Figure 3, in our preliminary experiments. Descriptions of the fitness cases (equally spaced samples) for each test function are given in parenthesis:

Damped Oscillator 1D $\frac{3}{2}\,\mathrm{e}^{-\frac{x}{2}}\sin(\pi x+\pi)$ (512 fitness cases from $[1, 4\pi]$)

Salustowicz 1D $\mathrm{e}^{-x}\,x^3\sin x\cos x(\sin^2 x\cos x-1)$ (512 fitness cases from $[0, 12]$)

Unwrapped Ball 1D $\frac{10}{(x-3)^2+5}$ (512 fitness cases from $[-2, 8]$)

Table 2: GP parameters of the experiment runs: note that some parameters do not apply to all experiments, these are marked by the phrase "FNGP and CBNGP only".

Objective:	symbolic regression of a test function f with fitness cases F
Fitness:	$\mathrm{RMSE}(f(F), [\![\cdot]\!](F))$ (RMSE between real and predicted function values)
Terminal set:	$\{x\} \cup [0.0, 1.0]$ (function parameter x and uniform distributed random constants)
Function set:	$\{+, \times, -, \div, \sin, \cos, \tan, \sqrt{\cdot}, \exp, \ln\}$
Selection:	tournament selection with tournament size 10
Population size:	200 expression trees
Initialization:	ramped half-and-half with maximum tree depth of 6
Variation:	constant node mutation (prob. 0.1), function node mutation (prob. 0.1), replacement of node by new subtree (prob. 0.1), single-point crossover (prob. 1.0)
Restart:	restart when median of genotypic distance $\delta_{(p_{\mathrm{vl}}, d_{\mathrm{discrete}})}$ is less than 4
Clustering:	FNGP: fixed, same at every pass; CBNGP: dynamic, based on $\delta_{(p_{\mathrm{vl}}, d_{\mathrm{discrete}})}$ (FNGP and CBNGP only)
Niches:	10 (FNGP and CBNGP only)
Pass termination:	terminate after $6,000$ fitness evaluations (FNGP and CBNGP only)
Run termination:	terminate after $300,000$ fitness evaluations

By using these very simple test functions, we are able to obtain usable results in comparatively short run time. A single experiment run takes about 10 minutes on a Intel Xeon 5550 system equipped with two 2.66 GHz quad core CPUs.

4.1 Results

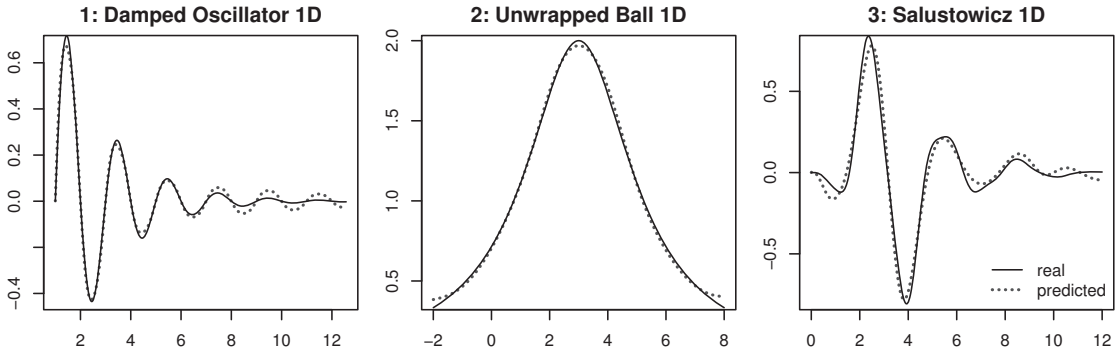

Figure 3: Plots of the univariate test functions for symbolic regression used in our experiments and best results: the test functions are plotted in solid black, the overall best symbolic regression models are plotted in dotted gray.

Table 3 shows a summary of the results of 10 runs for each algorithm (SGP, FNGP, and CBNGP) and each test function (Damped Oscillator 1D, Salustowicz 1D, and Unwrapped Ball 1D). The 10 runs for each algorithm are based on the same set of 10 random seeds. The lowest, i.e. best, RMSE values are highlighted by gray rectangles. Figure 3 shows dotted gray line plots of the overall best individuals from Table 3 overlayed over the solid black line plots of the test functions. Figure 4 shows the data of Table 3 in the form of box plots.

It is readily apparent from Table 3 that SGP results in the best minimum RMSE of 10

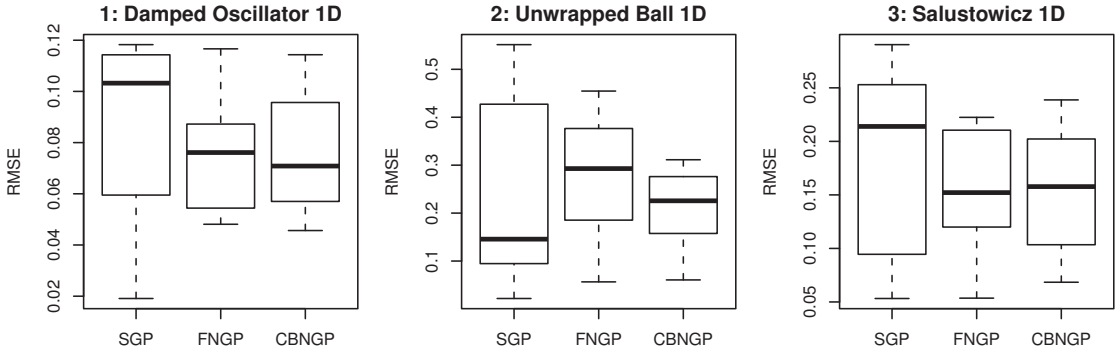

Figure 4: Box plots of our preliminary results: the plots are based 10 runs for each test function and each algorithm.

Table 3: Overview on our preliminary results: this table shows a summary of 10 runs for each algorithm and each test function. The best, i.e. lowest, RMSE values are highlighted by gray rectangles.

		RMSE					
Algorithm	*Test Fun.*	*Min.*	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*
SGP	Damp. Osc. 1D	0.01909	0.06051	0.10320	0.08644	0.11410	0.11830
	Salust. 1D	0.05322	0.09828	0.21390	0.17760	0.24520	0.29030
	Unwr. Ball 1D	0.02185	0.09492	0.14560	0.23240	0.39940	0.55140
FNGP	Damp. Osc. 1D	0.04810	0.05699	0.07612	0.07630	0.08613	0.11660
	Salust. 1D	0.05356	0.12440	0.15210	0.15630	0.20640	0.22240
	Unwr. Ball 1D	0.05653	0.19530	0.29290	0.27530	0.36890	0.45460
CBNGP	Damp. Osc. 1D	0.04564	0.05730	0.07084	0.07498	0.09282	0.11430
	Salust. 1D	0.06842	0.11040	0.15770	0.15840	0.20200	0.23870
	Unwr. Ball 1D	0.06063	0.16260	0.22560	0.20690	0.27280	0.31130

runs in all three test functions, in case of Damped Oscillator 1D and Unwrapped Ball 1D it does so by a large margin. At the same time, SGP's 10 run maximum RMSE is the worst in all three test functions. The question of which algorithm is best in the 10 run median case yields no conclusive answer. In the 10 run mean and in the 10 run maximum case, FNGP yields the best RMSE on the Salustowicz 1D test function, while CBNGP yields the best RMSE on the other two test function. Overall the differences between FNGP's and CBNGP's results are less pronounced than the differences between each of these niching GP algorithms and standard GP.

4.2 Discussion and First Conclusions

A possible explanation for the preliminary results shown is the high sensitivity of a GP run to its initial population. This explains the high standard deviation in the best fitness reached over 10 SGP runs: when starting with a favorable initial population, spending a high budget of fitness evaluations on a single run leads to good results. Naturally, the converse is also true, when starting a GP run with an unfavorable initial population, even a high budget of fitness evaluations leads to a poor result.

Both niching strategies, FNGP as well as CBNGP, result in significantly better mean results. This indicates that niching is effective in preserving genetic diversity in GP populations, leading us to accept hypothesis **H1**. Regarding hypothesis **H2**, i.e. whether clustering based niching is superior to static fixed niching in the domain of symbolic regression, our results are still inconclusive. Further experiments with more realistic test functions and real-world test cases should illuminate this question further.

5 Summary and Outlook

In this paper we introduced RGP, a new GP system for the R environment, and described our ongoing research in diversity preservation for GP populations through clustering based niching. We applied clustering based niching to symbolic regression with GP for the first time and conducted first experiments. Three classes of distance measures to be used for GP population clustering and convergence detection where introduced, while only genotypic distance was used in preliminary experiments.

In future work, we plan to study the relative benefits and drawbacks of fitness distance, phenotypic distance and genotypic distance measures for population clustering and convergence detection in a series of experiment runs. These runs will be based on a much broader set of test functions, including multivariate functions. We will also include difficult real-world regression problems from the financial and water resource management domains into our test function set [6, 5].

Regarding the further development of RGP, we are currently implementing optimized versions of the most important GP operators, which should enable RGP to solve more difficult problems in less compute time. We will also extend RGP's type system to allow the finer description of the relevant solution space. This will reduce the time spent searching infeasible regions of the solution space and will contribute to a more efficient and effective GP search.

6 Acknowledgements

This work has been supported by the Bundesministerium für Bildung und Forschung (BMBF) under the grants FIWA and SOMA (AiF FKZ 17N2309 and 17N1009, "Ingenieurnachwuchs") and by the Cologne University of Applied Sciences under the research focus grant COSA. We are grateful to Dr. Wolfgang Kantschik (DIP GmbH) for helpful discussions on niching strategies for GP.

References

[1] R Development Core Team: *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `http://www.R-project.org`. 2008.

[2] Banzhaf, W.; Francone, F. D.; Keller, R. E.; Nordin, P.: *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 1-55860-510-X. 1998.

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

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

[5] Flasch, O.; Bartz-Beielstein, T.; Koch, P.; Konen, W.: Genetic Programming Applied to Predictive Control in Environmental Engineering. In: *Proceedings 19. Workshop Computational Intelligence* (Hoffmann, F.; Hüllermeier, E., eds.), p. 101–113. Karlsruhe: KIT Scientific Publishing. 2009.

[6] Flasch, O.; Bartz-Beielstein, T.; Davtyan, A.; Koch, P.; Konen, W.; Oyetoyan, T. D.; Tamutan, M.: Comparing CI Methods for Prediction Models in Environmental Engineering. In: *Proc. 2010 Congress on Evolutionary Computation (CEC'10) within IEEE World Congress on Computational Intelligence (WCCI'10), Barcelona, Spain* (Fogel, G.; and others, eds.). Piscataway NJ: IEEE Press. 2010.

[7] Smits, G.; Vladislavleva, E.: Ordinal Pareto Genetic Programming. In: *Proceedings of the 2006 IEEE Congress on Evolutionary Computation* (Yen, G. G.; and others, eds.), p. 3114–3120. Vancouver, BC, Canada: IEEE Press. URL `http://ieeexplore.ieee.org/servlet/opac?punumber=11108`. 2006.

[8] Mahfoud, S. W.: *Niching methods for genetic algorithms*. Phd Thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA. 1995.

[9] Shir, O. M.: *Niching in derandomized evolution strategies and its applications in quantum control*. Phd Thesis, Natural Computing Group, LIACS, Faculty of Science, Leiden University. 2008.

[10] Spector, L.; Klein, J.: Trivial Geography in Genetic Programming. In: *Genetic Programming in Theory and Practice III*. Springer. 2005.

[11] Carbajal, S. G.; Levine, J.; Martinez, F. G.: Multi Niche Parallel GP with a Junk-Code Migration Model. In: *EuroGP*, p. 327–334. 2003.

[12] Hornby, G. S.: ALPS: the age-layered population structure for reducing the problem of premature convergence. In: *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation* (Keijzer, M.; and others, eds.), vol. 1, p. 815–822. Seattle, Washington, USA: ACM Press. ISBN 1-59593-186-4. 2006.

[13] Sun, Y.; Wierstra, D.; Schaul, T.; Schmidhuber, J.: Efficient natural evolution strategies. In: *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, p. 539–546. New York, NY, USA: ACM. ISBN 978-1-60558-325-9. 2009.

[14] Bartz-Beielstein, T.; Chiarandini, M.; Paquete, L.; Preuss, M. (eds.): *Experimental Methods for the Analysis of Optimization Algorithms*. Berlin, Heidelberg, New York: Springer. Im Druck. 2010.

[15] Tierney, L.; Rossini, A. J.; Li, N.; Sevcikova, H.: *snow: Simple Network of Workstations*. R package version 0.3-3. 2009.

[16] Schmidt, M.; Lipson, H.: Comparison of tree and graph encodings as function of problem complexity. In: *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation* (Thierens, D.; Beyer, H.-G.; Bongard, J.; Branke, J.; Clark, J. A.;

Cliff, D.; Congdon, C. B.; Deb, K.; Doerr, B.; Kovacs, T.; Kumar, S.; Miller, J. F.; Moore, J.; Neumann, F.; Pelikan, M.; Poli, R.; Sastry, K.; Stanley, K. O.; Stutzle, T.; Watson, R. A.; Wegener, I., eds.), vol. 2, p. 1674–1679. London: ACM Press. 2007.

[17] Barendregt, H.; Abramsky, S.; Gabbay, D. M.; Maibaum, T. S. E.; Barendregt, H. P.: Lambda Calculi with Types. In: *Handbook of Logic in Computer Science*, p. 117–309. Oxford University Press. 1992.

[18] Goldberg, D. E.; Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, p. 41–49. Hillsdale, NJ, USA: L. Erlbaum Associates Inc. ISBN 0-8058-0158-8. 1987.

[19] De Jong, K.: *An analysis of the behaviour of a class of genetic adaptive systems*. Phd Thesis, University of Michigan. 1975.

[20] Bäck, T.; Fogel, D.; Michalewicz, Z.: *Handbook of Evolutionary Computation*. New York NY: IOP Publishing and Oxford University Press. 1997.

[21] Ursem, R. K.: Multinational GAs: Multimodal optimization techniques in dynamic environments. In: *Proceedings of the Genetic and Evolutionary Computation Conference* (Whitley, D.; Goldberg, D. E.; Cantu-Paz, E.; Spector, L.; Parmee, I.; Beyer, H.-G., eds.), p. 19–26. Las Vegas, Nevada, USA: Morgan Kaufmann. 2000.

[22] Tsutsui, S.; Fujimoto, Y.: Forking genetic algorithm with blocking and shrinking modes (FGA). In: *Proceedings of the 5th International Conference on Genetic Algorithms* (Forrest, S., ed.), p. 206–215. San Mateo, CA: Morgan Kaufman. 1993.

[23] Streichert, F.; Stein, G.; Ulmer, H.; Zell, A.: A Clustering Based Niching Method for Evolutionary Algorithms. In: *Genetic and Evolutionary Computation GECCO 2003* (Cantu-Paz, E.; and others, eds.), Lecture Notes in Computer Science. Springer. 2003.

[24] Barile, M.: Discrete Metric. MathWorld – A Wolfram Web Resource, created by Eric W. Weisstein. URL `http://mathworld.wolfram.com/DiscreteMetric.html`. 2010.

[25] Keijzer, M.; Foster, J.: Crossover Bias in Genetic Programming. In: *Genetic Programming* (Ebner, M.; O'Neill, M.; Ekart, A.; Vanneschi, L.; Esparcia-Alcazar, A., eds.), vol. 4445 of *Lecture Notes in Computer Science*, p. 33–44. Springer. 2007.

[26] Jansen, T.: On the Analysis of Dynamic Restart Strategies for Evolutionary Algorithms. In: *PPSN VII: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, p. 33–43. London, UK: Springer-Verlag. ISBN 3-540-44139-5. 2002.

[27] Ward, J. H.: Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association* 58 (1963), p. 236–244.