RGP: An Open Source Genetic Programming System for the R Environment

Oliver Flasch
Department of Computer Science and Engineering Science
Cologne University of Applied Sciences
51643 Gummersbach, Germany
oliver.flasch@fh-koeln.de

Olaf Mersmann
Department of Statistics
TU Dortmund University
44221 Dortmund, Germany
olafm@statistik.tu-dortmund.de

Thomas Bartz-Beielstein
Department of Computer Science and Engineering Science
Cologne University of Applied Sciences
51643 Gummersbach, Germany
thomas.bartz-beielstein@fh-koeln.de

ABSTRACT

RGP is a new genetic programming system based on the R environment. 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. It strives for high modularity through a consistent architecture that allows the customization and replacement of every algorithm component, while maintaining accessibility for new users by adhering to the "convention over configuration" principle. Typical GP applications are supported by standard R interfaces. For example, symbolic regression via GP is supported by the same "formula interface" as linear regression in R. RGP is freely available as an open source R package.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Design, Documentation, Languages

Keywords

Late Breaking Abstract, Genetic Programming, Software, Symbolic Regression

1. INTRODUCTION

Genetic programming (GP) is a collection of techniques from evolutionary computing (EC) for the automatic generation of computer programs that perform a user-defined task [4, 1]. Starting with a high-level problem definition, GP creates a population of random programs that are progressively refined through variation and selection until a satisfactory solution is found.

An important advantage of GP is that no prior knowledge concerning the solution structure is needed. Another advantage is the representation of solutions as terms of a computer language, i.e. in a form accessible to human reasoning. The main drawback of GP is its high computational complexity, due to the potentially infinitely large search space of computer programs. On the other hand, 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 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 [5] already provide solutions for the more general tasks, greatly simplifying the development of GP systems based on these environments.

RGP[2] is based on the R environment for several reasons. Firstly, there seems to be a trend towards employing statistical methods in the analysis and design of evolutionary algorithms, including modern GP variants [7, 3]. Secondly, 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. Thirdly, 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 [8].

The remainder of this extended abstract gives a very short overview of RGP's design and functionality.

GECCO'10, July 7–11, 2010, Portland, Oregon, USA.
ACM 978-1-4503-0073-5/10/07.

[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 available at `rsymbolic.org`.

2. RGP OVERVIEW

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.

2.1 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 [6].

Besides classical untyped GP, strongly typed GP is supported by a type system based on simply typed lambda calculus [2]. 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 as part of 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.

2.2 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.

2.2.1 Initialization

Individual initialization can 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-typed expressions. Initialization strategies may be freely combined, e.g. to implement the well known *ramped-half-and-half* strategy.

2.2.2 Variation

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

2.2.3 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[3] for implementing a Pareto GP algorithm. 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. ACKNOWLEDGMENTS

This work has been supported by the Bundesministerium für Forschung und Bildung (BMBF) under the grant FIWA (AIF FKZ 17N2309) and by the Cologne University of Applied Sciences under the research focus grant COSA. We are grateful to Mark Kotanchek (Evolved Analytics) for making the DataModeler GP system available to us and to Wolfgang Kantschik (DIP GmbH) for many helpful discussions on real-world applications of GP.

References

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

[2] H. Barendregt, S. Abramsky, D. M. Gabbay, T. S. E. Maibaum, and H. P. Barendregt. Lambda calculi with types. In *Handbook of Logic in Computer Science*, pages 117–309. Oxford University Press, 1992.

[3] T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors. *Empirical Methods for the Analysis of Optimization Algorithms.* Springer, Berlin, Heidelberg, New York, 2009. In Press.

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

[5] R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2009. ISBN 3-900051-07-0.

[6] M. Schmidt and H. Lipson. Comparison of tree and graph encodings as function of problem complexity. In D. T. et. al., editor, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, volume 2, pages 1674–1679, London, 7-11 July 2007. ACM Press.

[7] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Efficient natural evolution strategies. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 539–546, New York, NY, USA, 2009. ACM.

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

[3]The EMOA Evolutionary Multiobjective Optimization Algorithm toolbox for R is available at `http://git.datensplitter.net/cgit/emoa`.