Research Repository UCD

Title	GEVA : grammatical evolution in Java
Authors(s)	O'Neill, Michael, Hemberg, Erik, Gilligan, Conor, Bartley, Elliott, McDermott, James, Brabazon, Anthony
Publication date	2008
Publication information	O'Neill, Michael, Erik Hemberg, Conor Gilligan, Elliott Bartley, James McDermott, and Anthony Brabazon. "GEVA : Grammatical Evolution in Java" 3, no. 2 (2008).
Publisher	ACM
Item record/more information	http://hdl.handle.net/10197/2578
Publisher's version (DOI)	10.1145/1527063.1527066

Downloaded 2023-04-07T10:52:29Z

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information

GEVA: Grammatical Evolution in Java*

Michael O'Neill[†] Erik Hemberg Conor Gilligan
Elliott Bartley James McDermott
Anthony Brabazon

Natural Computing Research & Applications Group
Complex & Adaptive Systems Lab
University College Dublin
Ireland
m.oneill@ucd.ie

We are delighted to announce the release of **GEVA** [1] an open source software implementation of Grammatical Evolution (GE) in Java. Grammatical Evolution in Java (GEVA) was developed at UCD's Natural Computing Research & Applications group (<http://ncra.ucd.ie>).

1 What is GE?

Grammatical Evolution (GE) (e.g., [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]) is a grammar-based form of Genetic Programming [15]. It marries principles from molecular biology to the representational power of formal grammars. GE's rich modularity gives a unique flexibility, making it possible to use alternative search strategies, whether evolutionary, or other heuristic, be it stochastic or deterministic, and to radically change its behaviour by merely changing the grammar supplied. As a grammar is used to describe the structures that are generated by GE, it is trivial to modify the output structures by simply editing the plain text grammar. This is one of the main advantages that makes the GE approach so attractive. The genotype-phenotype mapping also means that instead of operating exclusively on solution trees, as in standard GP, GE allows search operators to be performed on the genotype (e.g., integer or binary chromosomes), in addition to partially derived phenotypes, and the fully formed phenotypic derivation trees themselves.

*Published in SIGEVOLution Vol. 3 Issue 2 pp. 17-22

[†]Corresponding Author

Figure 1: A screenshot of the GEVA GUI. When opened for the first time it adopts the parameter settings for the pattern matching example problem where the goal is to rediscover the string “*geva*”.

2 Features

GEVA has been released under GNU GPL version 3, and uses Java 1.5.

As well as providing the characteristic genotype-phenotype mapper of GE, an evolutionary search engine and a GUI are also provided.

GEVA comes out-of-the-box with a number of demonstration problems that can be easily switched between from the GUI or command line. Sample problems include simple String Pattern Matching, an LSystem generator, the Paint problem, and a number of classic Genetic Programming problems such as an example of Symbolic Regression, the Santa Fe ant trail and Even Five Parity.

A screenshot of the default GUI screen showing settings for the pattern matching problem can be seen in Fig. 1. The goal of the problem is to rediscover the string “*geva*”. Simple graphing support is also provided, which allows the user to observe various attributes of the population live during the course of

Figure 2: An example graph produced while running GEVA on the pattern matching example problem. Displayed are the best fitness and average fitness with error bars. It is also possible to observe the average codon length of individuals, the average expressed codon length and the number of invalid individuals in the population.

a run. The resulting graphs can then be saved for later use. Attributes that can be plotted include the best fitness, the average fitness (with error bars), the number of invalid (incompletely mapped individuals), the average number of codons in each individual, and the average number of expressed codons in the population. For an example see Fig. 2.

A number of tutorials have been developed to help the novice user get up to speed: from running the software out-of-the-box, to using the command-line parameters, writing your own grammars and fitness functions, to developing your own search engine. These include a tutorial describing a bonus demo problem, Battleship, that the interested user can add to a GEVA installation.

3 Design Overview

GEVA takes advantage of GE’s modular structure as outlined in Fig. 3. This allows us to create a framework in which any search engine algorithm can be used to generate the genotypes (the **GEChromosome** class) that are used to direct the GE Mapper’s use of the Grammar during the development of the *output* solution. In recent years this approach has included the adoption of a Particle Swarm algorithm and Differential Evolution as alternative search engines resulting in Grammatical Swarm [14] and Grammatical Differential Evolution [16] variants. GEVA facilitates the adoption of alternative search engines through the provision of an **Algorithm** interface. This will work correctly as long as a GE Mapper object is provided with a legal **GEChromosome** object, so any alternative

Figure 3: Modular components of Grammatical Evolution.

algorithm must ensure to map its search engine’s individual representation to a **GEChromosome** to generate an output solution. In this first release a standard Genetic Algorithm engine is provided with plans to add alternative engines in future releases. The current version uses individuals with (32-bit) integer codon values and adopts a corresponding integer mutation operator.

Grammars are made available to GEVA through plain text files that adopt BNF notation. A simple parser is provided which handles standard BNF and can also recognise special symbols including **GE_CODON_VALUE**, which returns the current codon’s numerical value as a terminal symbol to the developing output phenotype sentence.

Simply by altering the contents of a BNF text file you can radically change the output generated by GEVA. A number of studies have illustrated this flexibility: for example, grammars have been used to represent a diverse array of structures including binary strings, code in various programming languages (e.g., C, Scheme, Slang, Postscript), music, financial trading rules, 3D surfaces, and even grammars themselves (examples include [18, 4, 19, 20, 21, 22, 23, 24]). A number of demonstration grammars are provided in the example problems and are available through the GUI and from the command line.

4 How to find out more

GEVA is available for download from the UCD NCRA group website (<http://ncra.ucd.ie/geva>) or <http://www.grammatical-evolution.org>. Included in the release are instruc-

Figure 4: GEVA includes an interactive form of GE as an example problem that allows users to generate interesting LSystems. A similar approach was used to evolve the logo for the Natural Computing Research & Applications Group at UCD (see Fig. 5).

Figure 5: The UCD NCRA group logo was evolved using Grammatical Evolution. See [17] for more details.

tions on how to run GEVA out-of-the-box, and more detailed tutorials for those who wish to modify the software for new purposes. We also welcome feedback on the software as we plan to actively maintain the code, releasing new versions as features are added. A GEVA Google group has been set up to facilitate communication amongst the GEVA community [25]. We hope that GEVA will be a useful resource for the EC community and beyond.

5 Acknowledgments

We would like to thank past and present members of the UCD Natural Computing Research & Applications group, especially Tiberiu Simu, Jonathan Hugoson, and Jeff Wright for initial testing and some additions to the code. GEVA was beta-tested by the students of *COMP30290 Natural Computing*, an elective offered by UCD School of Computer Science & Informatics [27], from September to November 2007, and by Patrick Middleburgh who developed the seed of the LSystem interface as part of his Final Year Computer Science project at UCD. We would also like to thank Miguel Nicolau for many interesting discussions that helped inform the design of GEVA, in particular the genotype-phenotype mapper which draws upon the design adopted in Miguel’s libGE C++ library. This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grants No. 08/IN.1/I1868 and No. 06/RFP/CMS042, and UCD Research Seed Funding.

References

- [1] GEVA - Grammatical Evolution in Java. (2008). <http://ncra.ucd.ie/geva.html>
- [2] O’Neill, M., Ryan, C. (2003). *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers.
- [3] O’Neill, M. (2001). Automatic Programming in an Arbitrary Language: Evolving Programs in Grammatical Evolution. PhD thesis, University of Limerick.
- [4] Brabazon, A., O’Neill, M. (2006). *Biologically Inspired Algorithms for Financial Modelling*. Springer.
- [5] Dempsey I. (2007). Grammatical Evolution in Dynamic Environments, PhD Thesis, University College Dublin.
- [6] Dempsey I., O’Neill M., Brabazon A. (2007). Constant Creation with Grammatical Evolution. *International Journal of Innovative Computing and Applications*, pp.23-38 Vol.1 No.1

- [7] O'Neill, M., Ryan, C. (2001). Grammatical Evolution, *IEEE Trans. Evolutionary Computation*. 2001.
- [8] Ryan, C., Collins, J.J., O'Neill, M. (1998). Grammatical Evolution: Evolving Programs for an Arbitrary Language. *Proc. of the First European Workshop on GP*, pp. 83-95, Springer-Verlag.
- [9] O'Neill, M., Brabazon, A. (2005). Recent Adventures in Grammatical Evolution. In *Proceedings of CMS 2005 Computer Methods and Systems*. Vol.1, pp. 245-253, November 2005, Krakow, Poland.
- [10] Hemberg E., Gilligan C., O'Neill M., Brabazon A. (2007). A Grammatical Genetic Programming Approach to Modularity in Genetic Algorithms. In Ebner M., O'Neill M., Ekart A., Vanneschi L., and Esparcia Alcazar A. (eds.) *EuroGP (Tenth European Conference on Genetic Programming)*, Valencia, Spain. Springer.
- [11] Brabazon A., O'Neill M., Dempsey I. (2008). An Introduction to Evolutionary Computation in Finance. *IEEE Computational Intelligence Magazine*, November, pp. 42-55, IEEE Press.
- [12] Cleary R. (2005). Extending Grammatical Evolution with Attribute Grammars: An Application to Knapsack Problems, Masters Thesis, University of Limerick. Available from <http://ncra.ucd.ie/downloads/pub/thesisExtGEwithAGs-CRC.pdf>
- [13] Amarteifio S. (2005). Interpreting a Genotype-Phenotype Map with Rich Representations in XMLGE, Masters Thesis, University of Limerick. Available from <http://ncra.ucd.ie/downloads/pub/SaoirseMScThesis.pdf>
- [14] O'Neill, M. and Brabazon, A. (2006). Grammatical Swarm: The Generation of Programs by Social Programming. *Natural Computing*, pp. 443-462 Vol.5 No.4
- [15] Poli, R., Langdon, W.B., McPhee, N.F. (2008). A field guide to genetic programming. Published via <http://lulu.com> and freely available at <http://www.gp-field-guide.org.uk>.
- [16] O'Neill, M. and Brabazon, A. (2006). Grammatical Differential Evolution. *International Conference on Artificial Intelligence (ICAI'06)*, pp. 231-236 CSEA Press Las Vegas, Nevada.
- [17] O'Neill, M., Brabazon, A. (2008). Evolving a Logo Design using Lindenmayer Systems, Postscript and Grammatical Evolution. In *Proceedings of the IEEE World Congress on Evolutionary Computation*, pp. 3788-3794. Hong Kong, 1-6 June 2008. IEEE Press.

- [18] O'Reilly U-M., Hemberg M. (2007). Integrating generative growth and evolutionary computation for form exploration. *Genetic Programming and Evolvable Machines*, 8(2), pp.163-186.
- [19] Moore J.M., Hahn L.W. (2003). Petri net modeling of high-order genetic systems using grammatical evolution. *BioSystems*, 72(1-2), pp.177-186.
- [20] de la Puente A.O., Alfonso R.S., Moreno M.A. (2002). Automatic composition of music by means of grammatical evolution. *Proceedings of the 2002 conference on APL*, pp. 148-155, ACM Press.
- [21] Karpuzcu U.R. (2005). Automatic Verilog Code Generation through Grammatical Evolution. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO2005) workshop program*, pp. 394-397, ACM Press.
- [22] Tsoulos I.G., Lagaris I.E. (2006). Solving differential equations with genetic programming. *Genetic Programming and Evolvable Machines*, 7(1), pp. 33-54.
- [23] Berarducci P., Jordan D., Martin D., Seitzer J. (2004). GEVOSH: Using Grammatical Evolution to Generate Hashing Functions. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO2004) workshop program*.
- [24] Cebrian M., Alfonseca M., Ortega A. (2007). Automatic generation of benchmarks for plagiarism detection tools using grammatical evolution. *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, Vol. 2, pp. 2253-2253, ACM Press.
- [25] GEVA user group. <http://groups.google.com/group/geva>
Email: geva@googlegroups.com
- [26] <http://www.grammatical-evolution.org>
- [27] O'Neill, M. (2007). COMP30290 Natural Computing.
<http://ncra.ucd.ie/COMP30290/>