Preface

Any sufficiently advanced technology is indistinguishable from magic.

Sir Arthur Charles Clarke

μGP is a computational approach for autonomously pursuing a goal defined by the user. To this end, candidate solutions for the given task are repeatedly modified, evaluated and enhanced. The alteration process mimics some principles of the Neo-Darwinian paradigm, such as *variation*, *inheritance*, and *selection*. μGP has been developed in *Politecnico di Torino* since 2000. Its original application was the generation of assembly-language programs for different types of microprocessors, hence the Greek letter *micro* in the name. Its name is sometimes spelled *MicroGP* or `uGP` due to typographic limitations. μGP is free software: it can be redistributed and modified under the terms of the GNU General Public License[1].

μGP is ordinarily utilized to find the optimal solution of hard problems, and it has been demonstrated able to outperform both human experts and conventional heuristics in such a task. In order to exploit the approach, the user describes the *appearance* of the solutions to his problem and provides a program able to evaluate them. The tool implementing the approach fosters a set of random solutions, and iteratively refines them in discrete steps. Its heuristic local-search algorithm uses the result of the evaluations, together with other internal information, to focus on the regions of the search space that look more promising, and eventually to produce an optimal solution.

μGP is an *evolutionary algorithm*. Different candidate solutions are considered in each step of the search process, and new ones are generated through mechanisms that ape both sexual and asexual reproduction. New solutions *inherit* distinctive traits from existing ones, and may coalesce the good characteristics of different *parents*. Better solutions have a greater chance to reproduce, and to succeed in the simulated struggle for existence.

Candidate solutions are internally encoded as graphs, or, more precisely, as directed multigraphs[2]. During the search process, multigraphs are constrained by a user-defined set of rules to conform to sensible structures. They are transformed

[1] For more information, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`

[2] A directed multigraph is a graph where a direction is assigned to each edge, and the same pair of vertices may be joined by more than one edge

to text files according to user-defined rules, and fed to a user-defined evaluation program. Thus, no knowledge about the problem being solved is included in μGP itself.

For an industry practitioner, μGP is a versatile optimizer able to tackle almost any problem with a limited setup effort. All the configuration is contained in XML[3] files, that can be created with simple text editors or powerful ad-hoc tools. μGP routinely handles problems that require solutions in the form of full-fledged assembly programs, including functions, interrupt handlers and data. But a much wider range of different problems can be tackled, including optimization of mathematical functions represented as trees, integer and combinatorial optimization, and real value optimization. While μGP is theoretically able to work with a problem that requires a simple unstructured solution, it may not be the best option in such cases, except, perhaps, for the easiness of set up. On the contrary, it should be exploited on tasks that involve the concurrent optimization of different data types, and when solutions exhibit quite complex structures.

This book shows how to effectively use μGP to solve an industrial problem. For this purpose, the text assumes that the user is competent in the application domain, but requires only a basic understanding of information technology. Moreover, only limited knowledge of the evolutionary computation field is required. The practitioner is guided through a list of easy steps to complete the setup. Moreover, an extensive discussion on the meaning and effect of the various parameters that can be tuned to increase the overall performance is provided.

For an evolutionary computation scholar, μGP may be regarded as a platform where new operators and strategies can be easily tested. Additionally, it presents some features that may be considered of interest: the possibility to shape the behavior smoothly from steady-state to generational, including several degrees of elitism; self adaptation of operator strength, operator activation probability, tournament size, population size and number of applied operators; diversity protection, trough population entropy and delta-entropy of individuals; fitness holes; clone detection, with optional scaling or extermination; support for different population topologies, from panmictic to lattice; multiple populations, including support for migrations; support for dynamic fitness functions; support for parallel fitness evaluation; multiple fitness, either priority-based or multi-objective.

Considering this latter goal, the book details the conceptual architecture and the implementation of the tool. While the text aims at self-containment, a basic knowledge of the evolutionary computation theory may be useful. Indeed, to fully understand the implementation details, a good knowledge of the C++ language is also required.

The book is organized in several broad sections. Chapters 1 and 2 introduce the reader to the field of evolutionary computation and provide a rationale for the whole book. Chapters 3 and 4 outline the main features of μGP from a theoretical point of

[3] XML stands for *extensible markup language*. It was developed by the World Wide Web Consortium (W3C) in the late 1990s, and defines a set of rules for encoding generic documents electronically.

view. Chapter 5 introduces the complete work flow for using μGP, and may provide a quick start for the impatient reader. Chapters 6, 7, 8, 9 and 10 cover the gory details of configuring the tool, running it and tweaking its many parameters. Chapter 11 illustrates the details of the μGP architecture and its implementation. Chapter 12 provides several examples of use of the tool, showing both the effect of tuning the evolution parameters, and several possible ways to approach problems to which the tool does not seem directly applicable. Finally, a couple of appendixes list all the possible options, parameters and special values that the tool recognizes inside its configuration files, together with a brief explanation of their use.

We would like to acknowledge some colleagues and friends who helped us in this project over the past ten years: Alessandro Aimo, Antonio Casaschi, Paolo Bernardi, Fulvio Corno, Gianluca Cumani, Davide Decicco, Sonia Drappero, Paolo Ferretti, Michelangelo Grosso, Germán Labarga, William Lindsay, Marco Loggia, Giuseppe Macchia, Onofrio Mancuso, Luca Motta, Zul Nazdri, Danilo Ravotto, Tommaso Rosato, Alessandro Salomone, Fabio Salto, Matteo Sonza Reorda, Luca Sterpone, Antonio Tomasiello, Giuseppe Trovato, Pier Paolo Ucchino, Massimo Violante, Gianluca Zaniolo.

We must dedicate a special thank to Alberto Tonda, who spent his Ph.D. enhancing and tweaking μGP. This book would not have been possible without his work and passion.

Torino,
Winter 2010

Ernesto Sanchez
Massimiliano Schillaci
Giovanni Squillero

http://www.springer.com/978-0-387-09425-0

Evolutionary Optimization: the µGP toolkit
Sanchez, E.; Schillaci, M.; Squillero, G.
2011, XIII, 178 p., Hardcover
ISBN: 978-0-387-09425-0