GUIDE User's Manual

Luis Da Costa and Marc Schoenauer

April 15, 2008

This document constitues is the User's Manual for GUIDE. An updated copy is kept on GUIDE's forge website, `http://gforge.inria.fr/projects/guide/`, under the *Documents* tab.

GUIDE is a software layer providing a common interface to different evolutionary libraries. The whole product comes accompanied with a Graphic User Interface (GUI) that extends the library's facilities to a wide range of users, expert practitioners of Evolutionary Computation (EC) or not. We think this is a very important point that deserves to be discussed and pinpointed, as we did in our latest communication[1] [1]. The center of our argument is presented in the following paragraphs.

One of the important research motivations in our research group (TAO, `tao.lri.fr`) is based on the idea of bringing the methods of our domain, Evolutionary Computation (EC), closer to general, *non-expert*, users. Indeed, despite impressive accomplishments, Evolutionary Algorithms (EAs) have still not been massively adopted as part of a general "*problem-solving toolkit*". We think that one of the main reasons for such "failure" is the lack of a clear, sensible unifying framework for the field. A proposal of a solution for this state of affairs gave birth to GUIDE.

There are (at least) two reasons for confusion among general users: first, until the publication of two books, one by Michalewicz [2] and the other by Eiben and Smith [3], it was very difficult to get a clear picture of the EC field, as there was not even a common terminology between papers in the domain. This terminology is very influenced by the history of the models that have been proposed over the years and, even though all models share a common structure, no existing software package allows the user to actually shift from one model to another [4].

The second problem is related to the specific tuning of this kind of algorithms. The description of a specific EA contains its components (the choice of representation, selection, recombination, and mutation operators) thereby setting a framework, while still leaving quite a few items undefined [5]. For instance, in a simple Genetic Algorithm (GA), while the representation is a simple bitstring with its associated operators, further details have to be given (population size and probability of application of these operators) before having a running version of the algorithm. These data, called the *algorithm parameters* or *strategy parameters*, greatly determine whether the algorithm will find an optimal or near-optimal solution, and whether it will find such a

[1] Latest GUIDE paper was sent on March 2008.

solution efficiently [5]. However, choosing such parameters is difficult, and a great deal of knowledge about EC is needed to do a sensible choice.

What would be needed for a "newcomer" to the field to use with ease an Evolutionary Algorithm? A user, advanced or not, is mainly concerned about the best possible way to solve their problem, so the historical differences that exist in the terminology will not matter at this stage [4]. Rather, as they are trying to solve a problem, they have the knowledge about two important points: first, they should be able to decide on an *encoding* for their problem. In other words, the problem to be solved needs to be adequately *represented*. Second, they have to somehow encode and define *what is a good solution for their problem*: this is what is called the definition of the *fitness function*. Other than that, they shouldn't be concerned with any of the details that we, practitioners in the field of EC, have to know about. GUIDE provides such a facility.

In this document you will find the description of GUIDE, from a designer's point of view. Writing this document was a great opportunity for us to revisit design choices we took for the implementation, so ideas and hints for improving them are added on the margins (like the one you see here), or as footnotes[2]. This document contains two big parts: one explaining the basis of GUIDE: it is what we called the *kernel*, and it is presented in Chapter . The GUI based on that kernel is in Chapter .

Hint-Hint!!

[2]This is a footnote - just in case you had forgotten.

Contents

List of Figures

List of Tables

Chapter 1

How to use this guide

1.1 Purpose and Audience

This document is targeted to any user that wishes to understand the internal functioning of GUIDE.

1.2 Scope

1.3 Suggested Reading Order

This document is organized in a logical understanding order: we will explain what is our view of an evolutionary algorithm at the beginning of the document, and then proceed examining the different parts needed to present this view to an user. So the best way of reading this document is to follow the logical structure. However, the different parts been presented are also differentiated by Sections, contained in 2 main Chapters: chap. explains the *kernel* of the application, and chap. explains the Graphical User Interface associated with this kernel.

1.4 Notational Conventions

In general, the following conventions are used in this guide:

- **Java**. The term "Java" refers to any general version of the Java compiler. When functionality differs between compilers, the specific terms

will be used (for example, *Java 1.5* for Java v. 1.5).

- **C++**. The term "C++" refers to any general version of the C++ compiler.
- **Eclipse** refers to the working environment of the same name. IDEs for different languages are found here: `http://www.eclipse.org/home/categories/languages.php`

1.5 Associated Documents and Electronic Resources

GUIDE's Javadoc Documentation is in `http://www.lri.fr/~ldacosta/guideDoc/` and was developped in the TAO team (`http://tao.lri.fr`); now is it hosted at `http://gforge.inria.fr/projects/guide/`. There are 2 main papers, to this date, associated with it: the first was published in 2003, under the title *GUIDE: Unifying Evolutionary Engines through a Graphical User Interface*, written by Pierre Collet and Marc Schoenauer [4]. The second is currently being evaluated for publication in the Journal of Artificial Intelligence Tools, and is identified in the biliography as [1].

GUIDE generates code for two evolutionary libraries: EO and ECJ. EO's website is `http://eodev.sourceforge.net/` and a main communication describing its functioning was published by Keizer *et al.* and corresponds to reference [6]. ECJ's website is `http://www.cs.gmu.edu/~eclab/projects/ecj/`

GUIDE is actively used as product in the Evotest European Project `http://evotest.iti.upv.es/`

Chapter 2

An Initial Example

In this Chapter we present an easy example of what can be done with GUIDE. The problem to be solved is presented in Section 2.1, followed by the description of how to describe it to the environment, in Section 2.2. Finally, we present the code generation that the software does, with a run of it (Section 2.3)

2.1 The Problem to be Solved

For this demonstration, let's suppose that we want to use an Evolutionary Algorithm (EA) to look for the global minimum of the so-called *Rastrigin's function*. For two independent variables, Rastrigin function is defined as

$$R(\vec{x}) = 20 + x_1^2 + x_2^2 - 10 * (\cos(2\pi x_1) + \cos(2\pi x_2))$$

A plot of the function is shown in Fig. 2.1. On it, we can observe how the function has a large number of local minima, and a global minimum at $(0, 0)$ This problem is often used to test the performance of an evolutionary algorithm, because its many local minima make it difficult for standard, gradient-based methods to find the global minimum. On Fig. 2.2 the alternating minima can be appreciated.

This is the problem we'll try to solve with the help of GUIDE as the EA generator. In the next Section we'll show you how to specify this problem, using GUIDE's GUI.

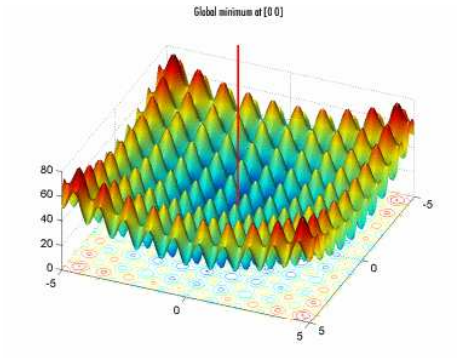

Figure 2.1: 2D Rastrigin's Function

Image from `http://www.mathworks.com/access/helpdesk/help/toolbox/gads/index.html?/access/helpdesk/help/toolbox/gads/f14773.html`

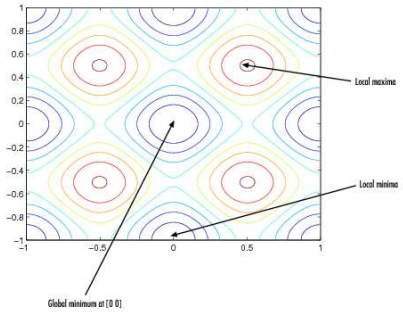

Figure 2.2: 2D Rastrigin's Function Contour Lines
Image from `http://www.mathworks.com/access/helpdesk/help/toolbox/gads/index.html?/access/helpdesk/help/toolbox/gads/f14773.html`

2.2 Problem Specification in GUIDE

First, we need to run GUIDE. You will then see the screen shown in Fig. 2.3

We want to tell GUIDE that we want to generate an EA to solve the following problem:

> Find the pair of reals $(x_1, x_2), x_i \in [-5, +5]$ minimizing the expression $20 + x_1^2 + x_2^2 - 10 * (\cos(2\pi x_1) + \cos(2\pi x_2))$

Let's use the environment for such a task; for this, we well use the two

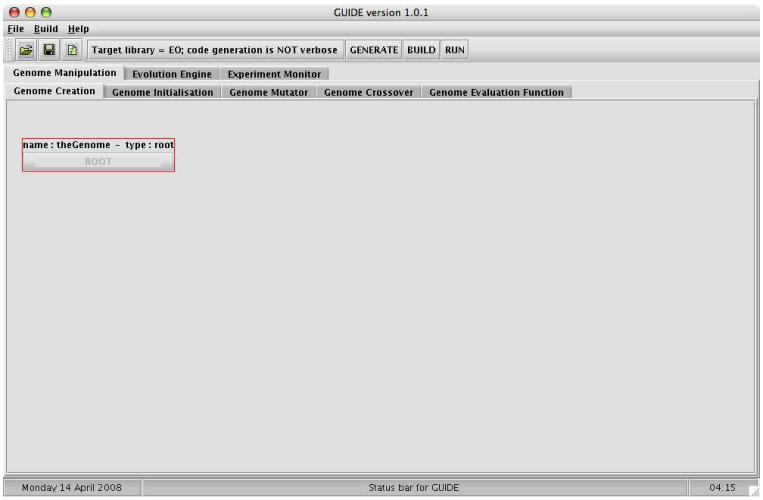

Figure 2.3: GUIDE's initial screen

rows of Tabs you can see in GUIDE's GUI: on the first row three tabs are proposed:

1. **Genome Manipulation**: this is the Tab selected by default. On it, you will be able to enter the Representation of the problem you are solving. As specified somewhere (and particularly on the our paper, refered as [1]), the Representation consists of the type of the entry data to your problem; accompanied with the Initialization and Variation Operators that can be applied on this data. That is why this Tab proposes 5 sub-options:

   (a) **Genome Creation**: in this Tab the type of the data used in the experiment is specified. It is grouped under a general type, called *root*. For specifying your data, right-click on the root node. You will see a Menu appear with a unique option available: "*insert under*". Clicking that option will present you with the pre-defined types[1] in GUIDE from which you can choose. Select a *real* type, and select an interval of $[-5.0, +5.0]$ when asked to do so. Repeat the procedure (starting from the clicking on the *root* node). The final result of your procedure should look like in Fig. 2.4.

[1] As a matter of fact, as noted and described in the System's Manual, those types are

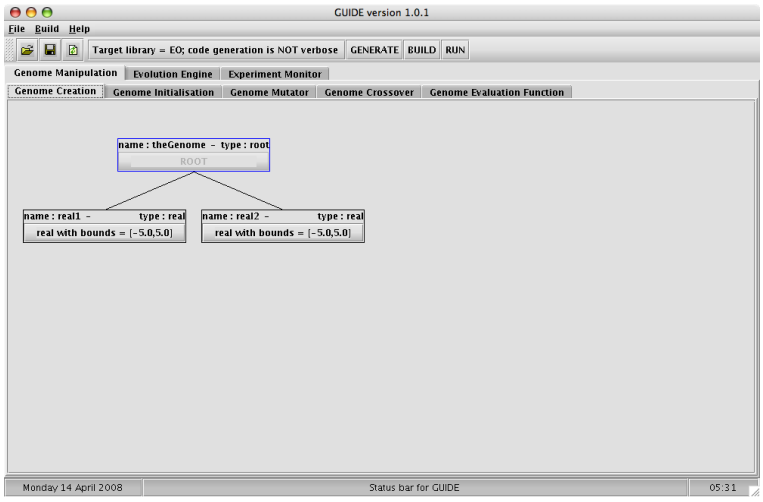

Figure 2.4: The two reals forming the entry data for our problem.

(b) **Genome Initialisation, Mutator, Crossover** and **Genome Evaluation Function**: in this Tab the initialisation, mutation, and crossover functions for the data specified are chosen and/or defined. You can build your own functions; however, GUIDE chooses sensible defaults, and for this early example we will keep them. However, you are free to wander around the Tabs defining such functions.

(c) **Genome Evaluation Function**: in this Tab the fitness function is specified. However, before specifying it, we will need to save all the information we have inserted on the environment so far. For that, go to the File Menu, click on Save and specify a location for your file experiment. Let's call this experiment *rastrigin.xml* - save it on your favourite folder (Fig. 2.5).

On the moment of Saving, GUIDE has collected all the information it needs to help you, in a very rudimentary way, to write the evaluation function. As you click on the "Genome Evaluation Function" Tab, you will see a template for writing a C++ function, with comments as to the variables you have accessible, and the bounds of them. For this case, you just need to write down

not *pre-defined*, but rather *user-defined* by means of a tree of specifications.

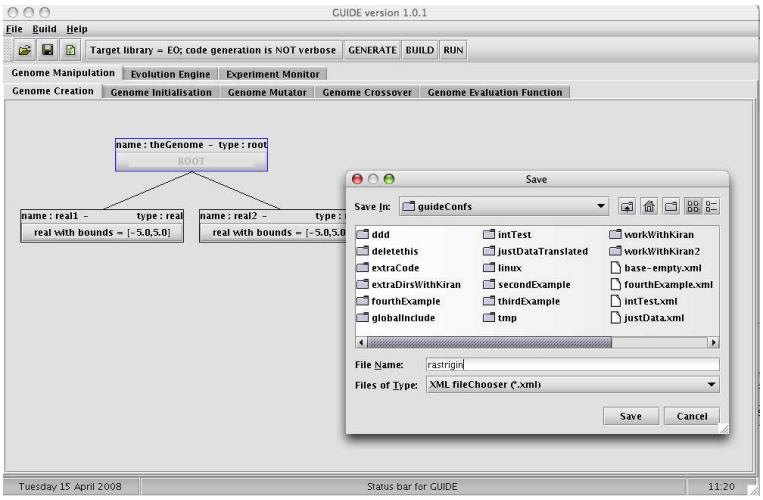

Figure 2.5: Saving Rastrigin's Data

the C++ mathematical expression:

```
fit = 20 + theGenome.var1 ^ 2 + theGenome.var2 ^ 2;
fit += -10 * cos(2 * pi * theGenome.var1);
fit += -10 * cos(2 * pi * theGenome.var2);
```

Other ways of treating the fitness function will be detailed later in this Document.

2. **Evolution Engine**: the engine is the part of the evolutionary algorithm where all parameters are defined. In this Tab you will see all the parameters displayed in logical order (although not in *beautiful* order. But this will come). See Fig. 2.6.

   The parameters of the engine have a huge influence on the performance of the generated EA. In this example we will not discuss the multiple options available to the user. For now, we will just check on the option marked "Fitness Goal" in order to make it "Minimise", and we will set a Stopping Criteria of 500 Generations.

3. **Experiment Monitor**:

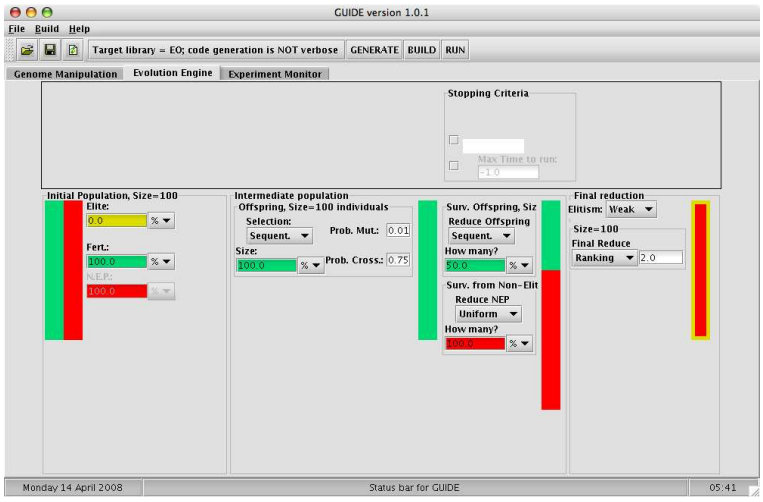

Figure 2.6: The Evolution Engine.

2.3 Code Generation in GUIDE

OK! You are ready now to generate and compile your code (that is, if you followed the steps described on the installation manual. If you didn't, please go and check it, and then come back here!). The button "**GENERATE**" will produce the C++ files needed for the experiment, while the button "**BUILD**" will compile all your code.

Acknowledgments

Bibliography

[1] Marc Schoenauer and Luis Da Costa. Representation-independent evolutionary algorithms: Principles and GUIDE, 2008. Submitted to the Journal of Artificial Intelligence Tools, March 2008.

[2] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, New-York, 1992-1996. 1st-3rd edition.

[3] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer Verlag, 2003.

[4] Pierre Collet and Marc Schoenauer. GUIDE: Unifying evolutionary engines through a graphical user interface. In Pierre Liardet, Pierre Collet, Cyril Fonlupt, Evelyne Lutton, and Marc Schoenauer, editors, *Evolution Artificielle, 6th International Conference*, volume 2936 of *Lecture Notes in Computer Science*, pages 203–215, Marseilles, France, 27-30 October 2003. Springer. Revised Selected Papers.

[5] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter Control in Evolutionary Algorithms. In F.G. Lobo, C.F. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, chapter 2, pages 19–46. Springer Verlag, 2007.

[6] M. Keijzer, J. J. Merelo, G. Romero, and M. Schoenauer. Evolving Objects: a general purpose evolutionary computation library. In P. Collet et al., editor, *Evolution Artificielle'01*, pages 229–241. LNCS 2310, Springer Verlag, 2002. URL: http://eodev.sourceforge.net/.