GA-lapagos, an Open-Source C Framework including a Python-based System for Data Analysis

Peter Jamieson
Department of Electrical and Computer Engineering, Miami University
Oxford, OH, USA
jamiespa@miamioh.edu

Ricardo Ferreira
Departament of Informatics, Universidade Federal de Viçosa
Viçosa, Minas Gerais, Brazil
ricardo@ufv.br

José Augusto M. Nacif
Science and Technology Institute, Universidade Federal de Viçosa
Florestal, Minas Gerais, Brazil
jnacif@ufv.br

ABSTRACT

In this work, we introduce, GA-lapagos, an open-source genetic algorithm framework written in 'C' for 3 exemplar optimization problems, and an accompanying Python-based data analysis scripts to extract and produce results. We created this system to help researchers implement a fast GA solving system for their problems allowing them to implement and leverage many ideas implemented in the research literature. Additionally, we provide a number of compilation paths to parallel computational systems such as multi-core, GPUs, and FPGAs. By building an executable framework and outputting results in comma separated values (CSV) format, we create a set of Python scripts to read the data and create graphs.

CCS CONCEPTS

• **Mathematics of computing → Combinatorial algorithms**;

KEYWORDS

Genetic Algorithms, Software Framework

ACM Reference Format:
Peter Jamieson, Ricardo Ferreira, and José Augusto M. Nacif. 2020. GA-lapagos, an Open-Source C Framework including a Python-based System for Data Analysis. In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377929.3398113

1 INTRODUCTION

There exist a number instances of software frameworks and libraries for executing genetic algorithm (GA) to solve optimization problems. Our framework, GA-lapagos, is created with the goals of being fast, flexible, open-source, and targetable to multiple computation substrates. GA-lapagos satisfies these base goals in a system that can target three exemplar problems: the travelling salesman problem (TSP), the Multi-dimensional Knapsack Problem (MKP) [8], and the Public Safety Network Mobile Station Placement Problem (PSNPP) [5]. Each of these problems has different ways of encoding the genomes that represent problem solutions, and therefore, were, initially, included to demonstrate the flexibility of our tool.

In particular, GA-lapagos is written in the 'C' programming language, is released under the MIT open source license, and is compiled and tested on standard Ubuntu Linux distributions. The Github repository contains all the benchmarks, source code, and build files to create and test the tool for the three described targeted problems. In addition to the software framework, we include a number of Python scripts to make it easy for researchers to execute and analyze their problems by executing benchmarks and collecting data. This data can be incorporated into graphs that show the performance of these algorithms via Python scripts. This allows us to get a detailed snapshot of the quality of each of our TSP GA instances of the problems, thus demonstrating the capabilities of our open-source tool that can be easily reproduced by others.

GECCO '20 Companion, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7127-8/20/07…$15.00
https://doi.org/10.1145/3377929.3398113

2 EXISTING GA LIBRARIES

Filho *et. al.* [3] wrote and early paper on "GA Programming Environments" and include a taxonomy to quantify these systems. This work is from 1994 and they provide a review of many of the systems available at that time. GA-lapagos, is classified as a GA Tool-kit for General-purpose systems based on their taxonomy.

More modern software has been produced since then including the popular GALib [7], which still exists as a C++ library implementation. ECJ [1] is a Java based library that is maintained and Gagné *et. al.* looked at these two libraries and their own, Open BEAGLE (C++), in terms of flexibility of software packages. More modern software systems have eclipsed the idea of just GAs. For example, HeuristicLab [6], which is a .NET software tool, includes GAs as part of its feature set, but GAs are only a small portion of the entire software framework.

3 GA-lapagos FRAMEWORK

GA-lapagos is a "C" software framework for experimenting with GAs. We will look at the basic features included in the software and the exemplar GA problem implementations, noting that we plan to expand GA-lapagos in the coming years.

The basic structure of a GA is shown in algorithm 1. Within GA-lapagos we have implemented most of these functions as "C" function pointers so that it is easy to implement new functions that perform the action. The function pointers allow each GA algorithmic steps to be re-implemented (we include the listing number in 1 for reference): Population initialization (Line 2), Exit condition (Line

Algorithm 1 Genetic Algorithm

```
1:  function Genetic_Algorithm()
2:  pop = initialize_population(genome, pop_size)
3:
4:  while exit_condition() == FALSE do
5:    rank = evaluate_and_rank_population(pop)
6:
7:    pop = create_new_population(mutation,
8:                                crossover,
9:                                selection(rank))
10: end while
11: end function
```

4), Cost function (Line 5), Mutation operator (Line 7), Crossover operator (Line 8), and Selection operator (Line 9).

At present, GA-lapagos includes three exemplar optimization problems. We chose these three problems because of their unique genome encoding and problem spaces, and the problems and respective encoding are: Traveling Salesman Problem (TSP), Multidimensional Knapsack Problem (MKP) [8], and Public Safety Network Mobile Station Placement Problem (PSNPP) [5].

Because of the different genome encodings, we have implemented a number of crossover operators [2] and selection operators for the above problems. For the crossover operator, we have implemented the following crossover operators for binary encodings: One-point Crossover, Two-point Crossover, and Uniform Crossover; the following for permutation encodings: Partial Mapped Crossover, Position-based Crossover, Cycle Crossover, Modified Cycle Crossover, Ordered Crossover, and Confined Swap Crossover.

We have implemented the following selection functions for choosing the parents to crossover (where these apply regardless of encoding): Roulette, Stochastic Universal Sampling, Tournament, Rank, and Truncation.

Additionally, we have provided researchers with a means to target parallel computational substrates, and these are: Multicore via Pthreads, GPU via CUDA, and FPGA via High-Level Synthesis. Based on our experience in this space, we believe that the GPU and threading implementations can be more, tightly, integrated into the codebase, but at present we have only done this for a simple threading implementation. Both the GPU and FPGA paths are illustrated by a one-off refactoring of the code, partially because of the use of function pointers.

3.1 Flexible Parameters for Experiments

We use a configuration file in XML file format for configuration of the framework for the optimization problem to execute and the options to select including algorithmic parameters such as the percent of a new population to be mutated and which crossover and selection operator to use. This approach is similar to the one we took in our open-source FPGA software [4] , and it allows for the quick configuration of many problems to execute.

The goal of GA-lapagos is to be flexible and configurable for a wide range of experiments and target computation platforms, and to provide sample problem instances with a GA to test out problems. We have released this as open source under the MIT license and can be accessed at: https://github.com/drpaj12/GA-lapagos.

3.2 Data Extraction and Analysis

The last aspect of GA-lapagos that we believe we should describe is how we generate results, extract the information, and create tables and graphs for experiments. While we do not consider these ideas to be novel, they are useful for any research software system, and these tools and methods can be applied in a number of systems for the benefit of the researcher.

We use Python scripts to batch execute the benchmarks, aggregate the data generated by the benchmarks, and process the data as either graphs or tables depending on presentation needs. For this, the first step is to have GA-lapagos output information to a text file in Comma Separated Values (CSV) format. The reason for this, is it makes the parsing of the data by a Python script trivial by using the Pandas Library (with the API call to read_csv() into DataFrames).

Finally, to create graphs we use the matlibplot library and the plethora of available graph types. These graphs are easy to generate from the DataFrame structure used in Pandas. For Latex tables, we use the Pandas API again with pandas.DataFrame.to_latex to create the table or use iterative string outputs to create the text for the table for import into our latex documents. By using these simple techniques we can quickly generate, aggregate, and create all of the data provided in the next section.

4 CONCLUSION

In this paper, we described our open source tool and framework built for GAs, GA-lapagos, can execute, produce, and generate data from the three optimization problems to analyze aspects of GAs.

The release of both the code and experimental setup is fundamental in improving GA research. For example, in our attempt to replicate some of the results from previous work, it is impossible given the data and code provided. Frameworks, like GA-lapagos, that are released openly allow other researchers to confirm the results and find errors in ours and others methodologies.

REFERENCES

[1] Sean Luke, Liviu Panait, Z Skolicki, J Bassett, R Hubley, and A Chircop. 2009. ECJ: a java-based evolutionary computation and genetic programming research system. *Disponıvel em http://www. cs. umd. edu/projetc/plus/ec/ecj/, última visita em* 24 (2009).

[2] G Pavai and TV Geetha. 2016. A survey on crossover operators. *ACM Computing Surveys (CSUR)* 49, 4 (2016), 1–43.

[3] José L Ribeiro Filho, Philip C Treleaven, and Cesare Alippi. 1994. Genetic-algorithm programming environments. *Computer* 27, 6 (1994), 28–43.

[4] J. Rose, J. Luu, C.W. Yu, O. Densmore, J. Goeders, A. Somerville, K.B. Kent, P. Jamieson, and J. Anderson. 2012. The VTR project: architecture and CAD for FPGAs from verilog to routing. In *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*. 77–86. http://dl.acm.org/citation.cfm?id=2145708

[5] Chen Shen, Mira Yun, Amrinder Arora, and Hyeong-Ah Choi. 2019. Efficient Mobile Base Station Placement for First Responders in Public Safety Networks. In *Future of Information and Communication Conference*. Springer, 634–644.

[6] Stefan Wagner, Gabriel Kronberger, Andreas Beham, Michael Kommenda, Andreas Scheibenpflug, Erik Pitzer, Stefan Vonolfen, Monika Kofler, Stephan Winkler, Viktoria Dorfer, et al. 2014. Architecture and design of the heuristiclab optimization environment. In *Advanced methods and applications in computational intelligence*. Springer, 197–261.

[7] Matthew Wall. 1996. GAlib: A C++ library of genetic algorithm components. *Mechanical Engineering Department, Massachusetts Institute of Technology* 87 (1996), 54.

[8] H Martin Weingartner and David N Ness. 1967. Methods for the solution of the multidimensional 0/1 knapsack problem. *Operations Research* 15, 1 (1967), 83–103.