Development of Software System for State Machine Generation Using Genetic Algorithms

Evgeny Andreevich Mandrikov Vladimir Anatolievich Kulev

***Abstract*—As a part of research works [1], [2], the development of a software system for state machine generation using genetic algorithms [3] is being done. This software system increases level of automation for designing state machine programs. It was named GAAP (Genetic Algorithms for Automata-based Programming).**

I. Introduction

For now many genetic algorithms realizations have been offered, and also genetic operators and chromosome representations [1]. Many of the existing genetic algorithms are only theoretical evidence which is not supported by any software solutions. Others ones are proved on test problems but hard to be applied for solving practical tasks due to lack of corresponding software systems flexibility.

II. Existing systems limitations and requirements for new ones

There are several common limitations of existing software systems implementing genetic algorithms:

1) Developed software system is strictly binded to a specific problem at point of individual representation.
2) Software implementation is done from scratch.
3) Software system is restricted to modification and integration with other software (particularly closed-source).
4) Search result and intermediate population states cannot be saved and therefore analyzed in future.

As proved in NFL (No Free Lunch) theorem [4], all extremum search algorithms save the same efficiency averaged for all possible fitness functions. Practical meaning of this theorem is there is no silver bullet and that absolute success of one optimization method in one field does not guarantee such success in other field. This indicates that for every specific field additional research is needed to find most suitable optimization method. All this implies following characteristics of software systems based on genetic algorithms:

1) Fitness function is built for each optimization task upon a specific problem [1, sec. 2.3 and 2.4]. Hence software system should allow usage of different fitness function.
2) Software system should contain genetic operators (creation of random individual, mutation, crossover, reorder). It is important to remember that those operators usually depends on specific chromosome representation.
3) Automatic or automated configuration upon a subject of research is necessary condition for effective usage of software systems using genetic algorithms.

On the basis of the foregoing, a number of requirements can be formulated for the developed software system supporting state machine generation using genetic algorithms:

1) Built-in chromosome representations and genetic operators library.
2) Module for fitting and adjustment of genetic algorithm and genetic operators parameters against a problem being solved.
3) Extensibility with external genetic operators, chromosome representations and fitness functions.
4) Ability to visualize found solutions.
5) Software system modularity with possibility to use nothing but modules needed for a specific problem.

III. Description of the developed software system and plans for future work

For implementation of the software system Java[1] programming language was choosed. Integration simplicity is ahieved by using Maven[2] project management and build automation tool.

The software system consists of the following modules:

1) Genetic algorithms module, including:
   a) Individuals repository for saving intermediate and final results. Repository can be used as a common individuals store when implementing composite genetic programming method [5]. Usage of several repositories allows to implement island genetic algorithm model [1] with different topology and rules of individuals interchange. Island model implementation is scheduled for inclusion into this module.
   b) Implementation of a standard genetic algorithm with option for fitting and adjusting of genetic algorithm and genetic operators parameters against a problem.

E. A. Mandrikov is BSc student at the Computer Technologies Department, Saint-Petersburg State University of Information Technologies, Mechanics and Optics, Saint-Petersburg, Russia (e-mail: mandrikov@rain.ifmo.ru).

V. A. Kulev is BSc student at the Computer Technologies Department, Saint-Petersburg State University of Information Technologies, Mechanics and Optics, Saint-Petersburg, Russia (e-mail: kulev@rain.ifmo.ru).

The research is supervised by A. A. Shalyto, PhD, professor at the Computer Technologies Department, Saint-Petersburg State University of Information Technologies, Mechanics and Optics, Saint-Petersburg, Russia (e-mail: shalyto@mail.ifmo.ru).

[1] http://java.sun.com/

[2] http://maven.apache.org/

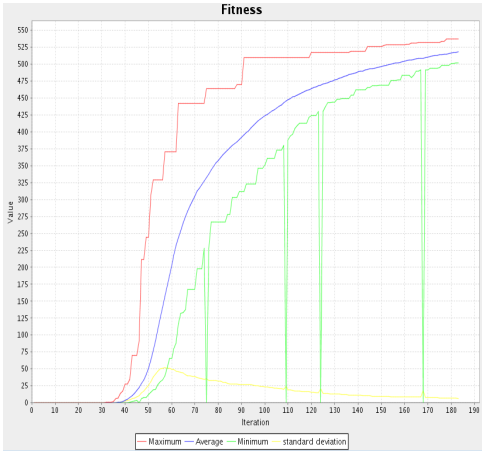

Figure 1. Genetic algorithm visualization

c) Tools for visualization and debugging. Example of visualization is on Figure 1.
d) Various genetic operators and chromosome representations (binary and real-valued encoding [6]) for numerical optimization tasks.

2) State machines module, including:
   a) Various state machine representations. At the moment there are two implemented representations (fixed length strings and decision trees [7]) and one is planned for integration.
   b) Support for XML format, which allows saving and modifying state machines. Proposed format is a basis for making state machine visual representation by XSL transformation. The format can be used for translating state machine into program code in various languages.
   c) Implementation of a "state machine + controlled object = automated object" model [8]. This allows state machine to manipulate the controlled object.

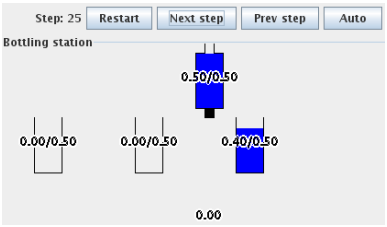

Figure 2. Visualization of a virtual "Pouring line"

   d) Tools for visualizing state machine execution step by step both forwards and backwards. Example is on Figure 2.

3) Genetic programming for state machines module. It consists of a number of genetic operators for state machine representations mentioned above.

We are also planning a distributed individuals testing support for the genetic algorithms module, because testing computational cost is very high.

After implementation of a core modules functionality, the result software system was evaluated for a following problems: "Pouring line" problem [9], "Flibs" problem [1], [10], "Ant" problem [1], [11], [12], [7].

There are ongoing software system adaptation for mobile robot programming purposes.

It ought to be noted that GAAP is being developed within the Open Project Documentation Foundation[3] and is an open source project[4]. All this encourages its further development and propagation.

IV. Related works

One previously developed software system [11] is oriented towards a single problem ("Ant" problem) research, thus difficult for further development and other applications. Other software system [9] is limited to only one state machine representation. This limitation does not allow in depth research of a specific problem. The proposed software system is more flexible and general-purpose.

References

[1] *Genetic programming technology for state machine generation for complex behavior systems control, stage I report*. SPbSU ITMO, 2007. [Online]. Available: http://is.ifmo.ru/genalg/_2007_01_report-genetic.pdf
[2] *Genetic programming technology for state machine generation for complex behavior systems control, stage II report*. SPbSU ITMO, 2007. [Online]. Available: http://is.ifmo.ru/genalg/_2007_02_report-genetic.pdf
[3] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*. New York, USA: John Wiley, 1966.
[4] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.
[5] Y. D. Bendy, "Genetic algorithms for generating cellular automatas," SPbSU ITMO, Tech. Rep., 2006. [Online]. Available: http://is.ifmo.ru/papers/genalgcelaut/
[6] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artificial Intelligence Review*, vol. 12, no. 4, pp. 265–319, 1998.
[7] V. R. Danilov and A. A. Shalyto, "Genetic programming method for generating state machines represented with decision trees," *Scientific software in education and scientific research. SPbSPU.*, pp. 175–181, 2008. [Online]. Available: http://is.ifmo.ru/download/2008-03-07-danilov.pdf
[8] N. I. Policarpova and A. A. Shalyto, *Automata programming, study guide*. SPbSU ITMO, 2007. [Online]. Available: http://is.ifmo.ru/books/_umk.pdf
[9] N. I. Policarpova, V. N. Tochilin, and A. A. Shalyto, "Development of a library for generating controlling state machines using genetic programming method," *Digest for a X international soft computations and measurements conference*, vol. 2, pp. 84–87, 2007. [Online]. Available: http://is.ifmo.ru/download/polikarpova(LETI).pdf
[10] E. A. Mandrikov, V. A. Kulev, and A. A. Shalyto, "Applying genetic algorithms to controlling state machine generation for 'flibs' problem," *Information technologies*, no. 1, pp. 42–45,89, 2008. [Online]. Available: http://is.ifmo.ru/download/2008-02-23_flibs.pdf
[11] Y. D. Bendy and A. A. Shalyto, "Applying genetic algorithms to state machine generation for 'ant' problem," SPbSU ITMO, Tech. Rep., 2007. [Online]. Available: http://is.ifmo.ru/works/_ant.pdf
[12] F. N. Tsarev and A. A. Shalyto, "Using genetic algorithms for generating state machine with minimal states count in 'ant' problem," *Digest for a X international soft computations and measurements conference*, vol. 2, pp. 88–91, 2007. [Online]. Available: http://is.ifmo.ru/download/2008-02-25_tsarev_shalyto.pdf

[3] http://is.ifmo.ru/foundation/?i0=foundation
[4] GAAP web site - http://godin.net.ru:8080/projects/gaap/