USING DESIGN PATTERNS FOR DESIGN OF SOFTWARE ENVIRONMENT FOR RESEARCHES IN GENETIC ALGORITHMS

Yuri R. Tsoy, Vladimir G. Spitsyn
Department of Computer Engineering,
Tomsk Polytechnic University,
84, Sovetskaya street, Tomsk, 634034, Russia,
Tel: +7 3822 418912,
Fax: +7 3822 419149,
neuroevolution@mail.ru

Abstract

Genetic algorithms are known as common and robust optimization concept. One of their main disadvantages is the lack of theoretical and practical research results. We are going to introduce program environment "GA Workshop" project that will have convenient tools for research purposes. This software is designed with use of design patterns approach, which guarantees further extendibility, flexibility and scalability of introduced program product. This software will be distributed free with its source codes for research and education centers.

Keywords: genetic algorithms, software design, design patterns**.**

1. Introduction

Rapid informational technologies development in 20-th century significantly extended scientific research making incredible machine resources available for modeling and complex calculations. Along with this fact an increase of experimental data requires complex programs to process these results. Moreover almost any specific problem requires specific mathematical tools and has different parameters to visualize.

The main goal of this work is to design a software environment with following characteristics:

1. Ability to research genetic algorithm properties and its performance for wide variety of algorithm models and tasks.

2. There should be ability to vary parameters of genetic algorithm without deep dependence on algorithm model and chosen task.

3. Ability to visualize and process data of experiments.

4. Architecture of program should be as flexible as possible. This feature provides support of future extensions and use of components designed by other developers.

For implementation of the last feature we decided to use design patterns. They represent in many ways common methodology for design of reusable software. All patterns names that will be mentioned in this paper are taken from [1].

2. Architecture description

Introduced program consists of 5 blocks (fig.1):

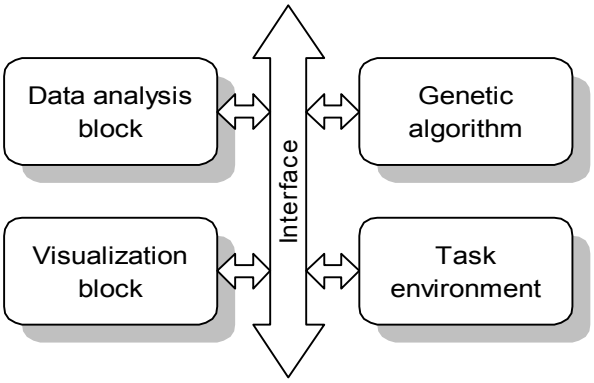

Fig. 1. Program structure

1. Interface – needed to bind and isolate all other blocks. Implements all data conversions necessary for other blocks cooperation.

2. Genetic Algorithm – implements certain algorithm and its workflow control.

3. Task environment – implements environment for the given problem, where all potential solutions, got by some algorithm are evaluated.

4. Analysis block – implements algorithms for analysis of genetic algorithm and solutions data.

5. Visualization block – implements routines necessary for output of experimental results in graphical representation.

The central part that allows significant enhance of flexibility and extendibility of introduced program system is Interface. Since it's present it gives an opportunity to isolate blocks from each other to make them as independent as possible. Thus extendibility and flexibility are provided.

Each of other blocks includes following obligatory parts (fig. 2):

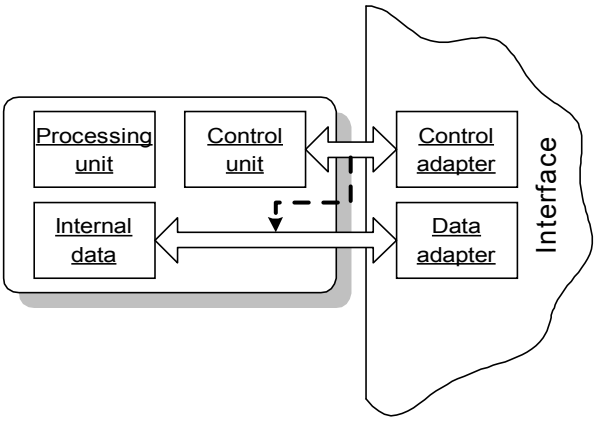

Figure 2. General block structure

1. Control unit. As blocks can be of any internal structure and possibilities they should provide the way to control their work.

2. Processing unit. Should implement all operations specific for given block.

3. Internal data. This part represents data in formats used by block for proper functioning.

To make interaction between blocks possible Interface implements adapters that convert data from one internal block representation to another and also delegate functions calls. In most cases control adapter implements only the very general interface functions such as block mount and dismount, initialization, and data transfers. Any operations specific for certain block should be implemented via control and processing unit of this block.

2.1. Genetic algorithm block description

As this software complex is designed for purposes of researches in genetic algorithms we will take a more thorough look at "Genetic algorithm" block.

There are following parameters in implementation of genetic algorithm that can vary in rather wide range:

1. Genetic representation (string of loci, array of genes, array of complex structures).
2. Initialization of population (random, user-defined, combined).
3. Selection strategy (roulette, tournament, truncation).
4. Reproduction strategy ((semi)random parent individuals, mating pairs, one or two offspring).
5. Genetic operators (different types of crossover and mutation, task specific operators).
6. Next generation formation properties (elitism, lamda+nu strategy).

Listed parameters to be implemented independently from each other. Use of FactoryMethod and TemplateMethod patterns make it possible to put them together.

This block should not implement any data processing (for example, statistical) and visualization. If there is need of them then appropriate calls to interface should be made with correspondent data transfer. An Interface since it implements Mediator pattern will then call Analysis and Visualization block routines and will provide all necessary data conversion via correspondent adapters. Thus a transparent interaction between blocks is provided. Any on-line data processing can be available via Observer pattern applied to parameters of genetic algorithm.

Collaboration with Task Environment block is implemented via pass of parameters from genetic representation of some individual from Genetic algorithm block to the Task Environment block. All necessary data conversion made in Interface just like in case of data processing. Feedback about fitness of individual passed to the Genetic Algorithm block. Evaluation of success of individuals should be implemented inside Task Environment block.

3. Flexibility and extensibility

Designed architecture allows collaboration of different blocks even if they were not designed to work together. Set of adapters implemented in interface helps to use new block easily without serious modifications in interface and without any modifications in other blocks.

One block can behave like different ones. Thus blocks can be combined. For example data processing and visualization blocks. But for other parts of introduced software there are still 2 separate blocks with different functionality and data formats.

From the other hand its easy to imagine situations when there are multiple representations of one block. In other words several independent blocks can operate like they are parts of one bigger block for example in case of complex computations.

Given architecture gives an opportunity to modify separate blocks without any modifications in other parts except appropriate adapter for Interface. Anyway structure of system does not change. Hence it's possible to modify different parts not thinking about compatibility problems because we can always write new adapter. This is much easier then trials to take into consideration all other blocks structure and data formats.

Inner organization of Genetic Algorithm block provides rather easy way to vary inner properties of different algorithms and investigate its performance. One more advantage is that different parts of algorithm are reusable for different models.

Conclusion

We should note that introduced architecture can be applied with minor modifications for research in almost any problem area.

Our work is only started. First release will likely be available in the 2-nd quarter of the next year. We are going to distribute introduced software without any charge with condition that it will be used for research purposes only. For the convenience of future users source code in C++ and documentation will be available too.

References

[1] Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*, Massachusetts: Addison-Wesley, 1995.

[2] Whitley D., A genetic algorithm tutorial, 1994, *Statistics and Computing*, 1994, n4, p. 65-85.