A Simulator-Independent Optimization Tool based on Genetic Algorithm Applied to Nuclear Reactor Design

Cláudio Márcio do Nascimento Abreu Pereira[1,2], Roberto Schirru[3] and Aquilino Senra Martinez[3]

Abstract

Here is presented an engineering optimization tool based on a genetic algorithm, implemented according to the method proposed in recent work that has demonstrated the feasibility of the use of this technique in nuclear reactor core designs. The tool is simulator-independent in the sense that it can be customized to use most of the simulators which have the input parameters read from formatted text files and the outputs also written from a text file. As the nuclear reactor simulators generally use such kind of interface, the proposed tool plays an important role in nuclear reactor designs. Research reactors may often use non-conventional design approaches, causing different situations that may lead the nuclear engineer to face new optimization problems. In this case, a good optimization technique, together with its customizing facility and a friendly man-machine interface could be very interesting. Here, the tool is described and some advantages are outlined

Introduction

Genetic algorithms (Holland, 1975; Goldberg, 1989; Davis, 1991) has been successfully applied in many areas of the nuclear engineering, such as reactor physics designs (Pereira, 1999), refueling optimization (Chapot, 1999; DeChaine, 1995), nuclear power plant status (Pereira, 1998) and preventive maintenance scheduling optimization (Lapa, 1999; Muñoz, 1997). Motivated by that, it was developed a generic tool, based on genetic algorithm to be applied in nuclear engineering designs - SIGA (Simulator-Independent Genetic Algorithm). Such tool is simulator independent in the sense that it can work together with most of the simulators in which the input parameters are read from formatted files and the outputs are written in text files. Another fact that has motivating the development of the proposed tool is that most of the nuclear engineering simulators, specially in the field of reactor physics and thermal hydraulic, use formatted files for modeling.

The SIGA-Simulator Interface

The role of a design optimization tool is to evaluate design parameter configurations, finding the best one. Whence, many inputs must be written, many simulations must be done and many outputs must be read. So, for each evaluation, the optimization tool must realize four main steps:

- write the parameters into the input file for the simulator;
- run the simulator with that input file;
- read the important variables involved in the evaluation process;
- evaluate the configuration according to an objective function.

[1] *Instituto de Engenharia Nuclear, Comissão Nacional de Engenharia Nuclear, Coordenação de Reatores, CaixaPostal 68550, Rio de Janeiro - RJ – Brasil.*
[2] *Universidede Iguaçu, FACET - Departamento de Ciência da Computação. Av. Abílio Augusto Távora, 2134, Nova Iguaçu, RJ – Brasil.*
[3] *Universidade Federal do Rio de Janeiro, Programa de Engenharia Nuclear, Caixa Postal 68509, Rio de Janeiro - RJ – Brasil.*

Figure 1 shows the relationship between the GA and the Simulator.

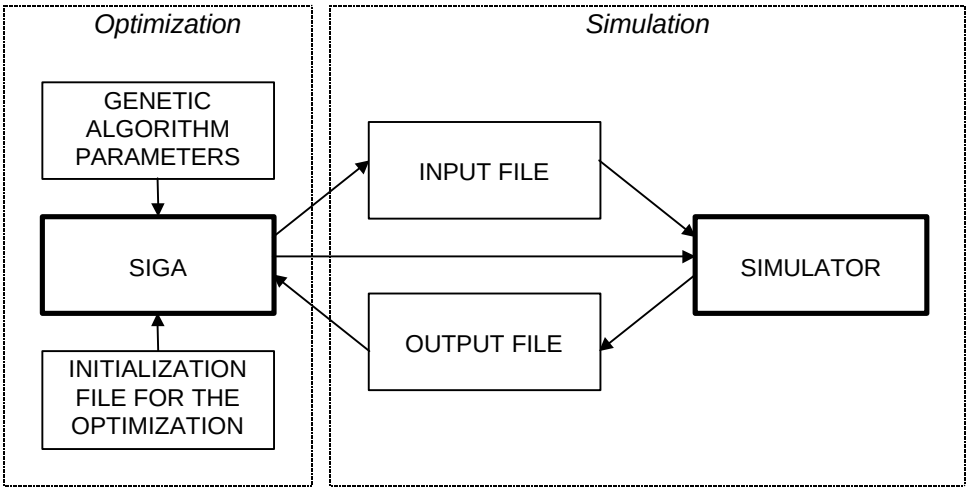

Figure 1 - Schematic diagram of the GA-Simulator interface

SIGA reads the genetic parameters file that contains the settings of the genetic algorithm such as crossover rate, mutation rate and population size, and it also reads the initialization file that will tell it everything about the optimization, such as:

- the parameters that may change during the optimization process;
- the allowed range for the parameters;
- the output variables that take part of the objective function;
- the status of the variable optimization (maximize, minimize, constraint range etc).

as well as:

- name and path for the executable code of the simulator;
- name and path for the input file for the simulator;
- name and path for the output file for the simulator;

The parameter to be written into the input file are defined as a position (row, column) in the file, and its format. The superior and inferior limit of the ranges for each one is read as optimization input parameters.

In the output file, the position of the interesting variables as well as the length of the string to be read are entered in the initialization file for the optimization. Because some times the output file is not exactly formatted, the position is referenced to an identification string.

It is also possible to correct others variables values that must change together with the optimization parameters, using an evaluation expression. For example, it is needed to change the fuel radii of a reactor fuel cell, and the cladding thickness must remain constant, it is necessary to update the external radii of the cladding (but it is not an optimization parameter - it is consequence). In this case it is possible to write the cladding external radii as a function of the fuel radii (ex.: external cladding radii = fuel radii + constant).

Figure 2 shows an example of initialization file for the optimization of nuclear reactor parameters using the hammer code (Suich, 1967)

Line	
1	
2	******************************
3	SIGA INITIALIZATION FILE
4	******************************
5	
6	[Files]
7	e:\SIGA\hammer32.exe
8	e:\SIGA\hammer.dat
9	e:\SIGA\hammer.out
10	
11	[Parameters]
12	2
13	1 Rfuel 9 21 %10.8f
14	2 Req 17 21 %10.8f
15	
16	[Update]
17	1
18	1 Rclad 16 21 %10.8f P000+0.05
19	
20	[Output]
21	2
22	1 Flux 17 45 10 M KEFF
23	2 Keff 0 5 9 B 0.99 1.01 1.0 KEFF
24	
25	[End]
26	

Figure 2 - An example of initialization file for SIGA

The example of Figure 2, can be translated as follow:

using the simulator *e:\SIGA\hammer32.exe*;

maximize the variable called *Flux* (line 22) to be read from file *e:\SIGA\hammer.out* (line 9) *17* lines and *45* columns after the identification *KEFF* (line 22);

subject to value of variable called *Keff* (line 23), to be read from file *e:\SIGA\hammer.out* (line 9) *0* lines and *5* columns after the identification *KEFF* (line 22), between *0.99* and *1.01*, with penalization factor *1.0* (line 23);

varying the *2* parameters, *Rfuel*, to be written in the simulator input file *e:\SIGA\hammer.dat* (line 8) in line *9* and column *21*, with format *%10.8f* and *Req*, also to be written in the simulator input file *e:\SIGA\hammer.dat* in line *17* and column *21*, with format *%10.8f,* between range specified in the GENESIS template file.

updating the variable *Rclad* to be written in the simulator input file *e:\SIGA\hammer.dat* (line 8) in line *16* and column *21*, with format *%10.8f* according to the expression P000+0.05 that means, first parameter (P000) plus constant 0.05.

The Genotype and Fitness used in the GA

Both genotype and fitness used are the ones specified in Pereira (1999). The genotype is formed by the binary codification of the value of each optimization parameter. The generalized fitness concept can be written as:

$$f = O - \sum_{i=1}^{N} \Delta C_i \cdot k_i \qquad (1)$$

where O is the objective variable that must be maximized or minimized, ΔC_i is how far is variable C_i from the constraint range, and k_i is the importance factor that emphasizes more or minus the penalty.

The Tool Implementation

Initially, the GA code was developed based on the Genetic Search Implementation System - GENESIS (Grefenstette, 1990) version 5.0, and all the interface font code was written as functions inside the genesis code. The new version of SIGA (under development) is not based on GENESIS.

In order to provide a friendly man-machine interface, a graphical application, called Visual SIGA was developed to work under Windows 95. Visual SIGA provides a man-machine interface to SIGA as well as file management allowing the user to easily configure the optimization problem with no contact with the optimization initialization file or the genetic parameters file. Figure 3 shows the block diagram of the Visual SIGA system.

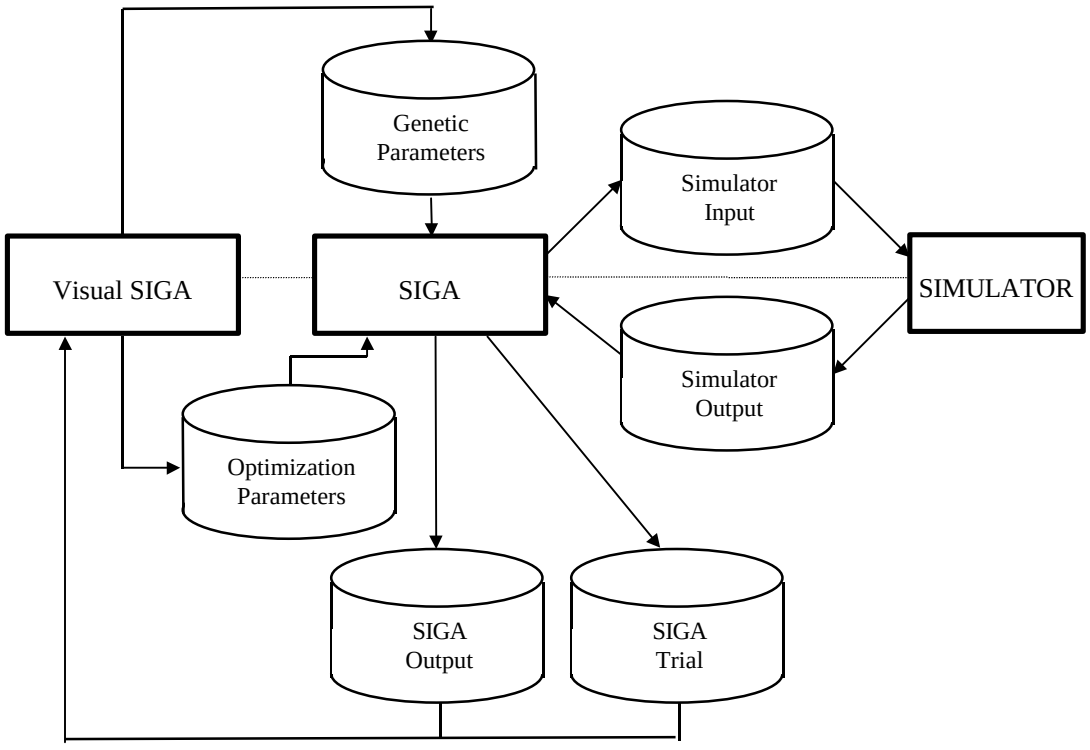

Figure 3 - The Visual SIGA system block diagram.

Visual SIGA provides a windows-based interface that allow user to write the optimization problem without having contact with the files involved. However, SIGA can be used without Visual SIGA.

SIGA files are written by Visual SIGA. Then, the Visual SIGA passes the control to the SIGA. Once SIGA is initialized, it controls the simulator reading and writing input and output files respectively. After each trial, SIGA writes the current output data into SIGA's trial file, in order to be displayed on screen by Visual SIGA in the form of numbers and graphs. In parallel, the best configuration even found is written to SIGA Output, and is as well displayed on screen.

Just for illustration, some pictures of the Visual SIGA input screens are showed in Figure 4. In Figure 4 is shown the input window that have 4 parts: (i) *Simulator*, were the user can enter the name and paths for the simulator to be used as well as the input and output files; (ii) *Optimization Parameters*, were the parameters to be optimized are described in terms of their positions in the input file, format, range etc; (iii) *Update Variables*, were the variables to be updated are described and (iv) *Outputs*, to enter the objective and constraints of the problem.

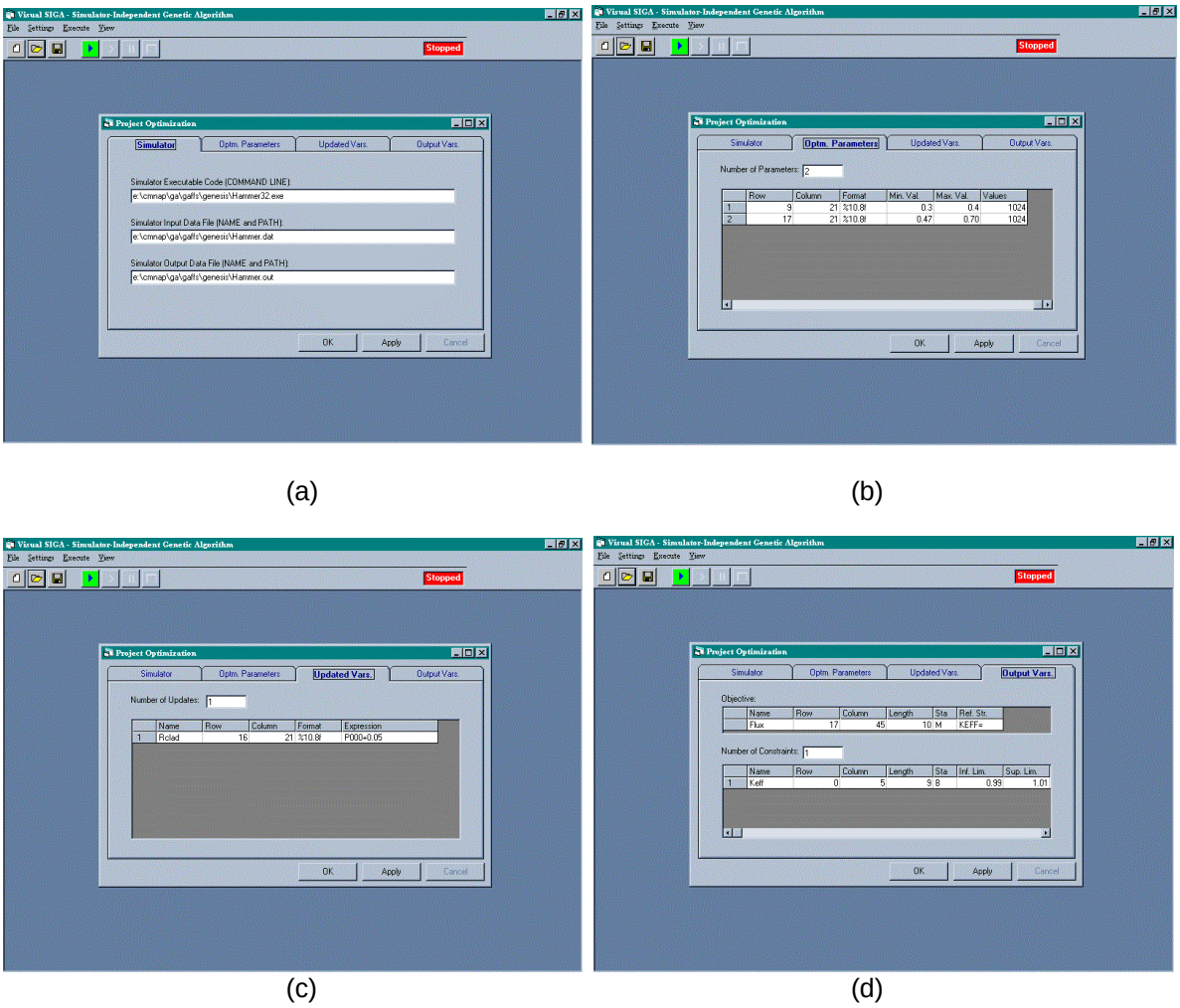

Figure 4 - Input screens of the Visual SIGA. (a) Simulator, (b) Optimization Parameters, (c) Update Variables and (d) Outputs.

The output window allow user to view the exploitation made by the GA, in the actual current trial and the best configuration ever found. It is also available a graph window that can show the best average and fitness through the generations.

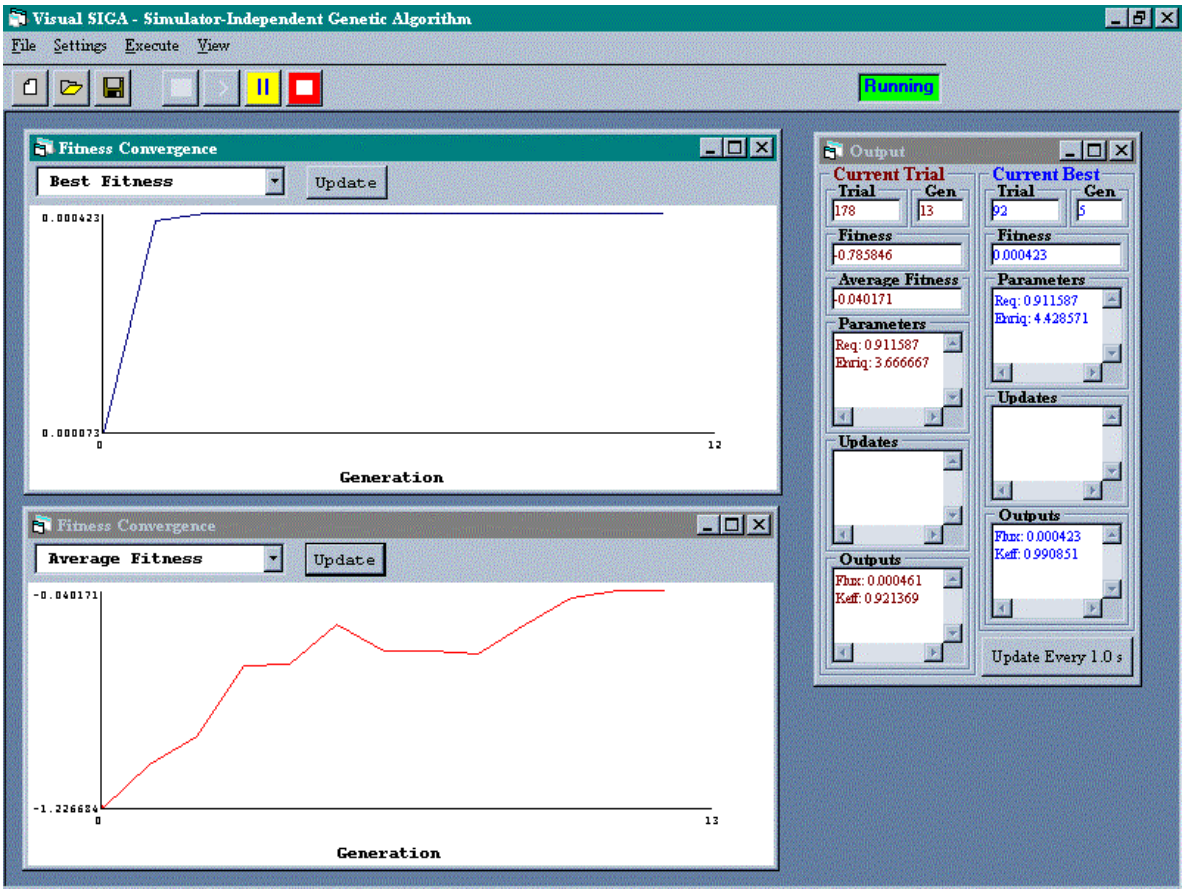

Figure 5 - Output Window of the Visual SIGA System

Tool Application and Conclusions

The tool could be efficiently applied to a nuclear reactor cell design optimization using the Hammer system, reproducing the results shown in past work (Pereira, 1999).

Once verified in past work (Pereira, 1999) that GA is useful in reactor design optimization, and due to the friendly man-machine interface together with the ability of interfacing with many different simulators, SIGA has demonstrated to be very useful as optimization tool for nuclear reactor design. SIGA intend to be a powerful optimization tool that do not need to be operated by an expert. Hence, the nuclear engineers/designers can easily optimize their designs. Although SIGA has been developed to be applied to nuclear reactor designs, it can be applied to others optimization problems, however, the feasibility of the use of GA to such problems must be investigated.

References

Chapot, J. L. C., Silva, F. C., Schirru, R., 1999. A New Approach to the Use of Genetic Algorithms to Solve Pressurized Water Reactor's Fuel Menagement Optimization Problem", Annals of Nuclear Energy, 26,7, 641-655.

Davis, L., Handbook of Genetic Algorithms, VNR, New York, 1991
DeChaine, M. D. and Feltus, M. A., 1995. Nuclear Fuel Management Optimization Using Genetic Algorithms, Nuclear Technology, 111, 109-114.
Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley.
Grefenstette, J. J.; A User's Guide to Genesis Version 5.0, 1990.
Holland, J. H., 1975. Adaptation in Natural and Artificial Systems, An Arbor, University of Michigan.
Lapa, C. M. F, Pereira, C. M. N. A. and Mol, A.C. A. (1999) Maximization of a Nuclear System Availability through Maintenance Scheduling Optimization Using Genetic Algorithm, Nuclear Engineering and Design (to appear).
Muñoz, A., Martorell, S. and Serradell, V., 1997. Genetic Algorithms in Optimizing Surveillance and Maintenance of Components. Reliability Engineering and System Safety 57, 107-120.
Pereira, C. M. N. A. , Schirru, R. e Martinez, A. S., 1999 Basic Investigations Related to Genetic Algorithms in Core Designs, Annals of Nuclear Energy, 26, 3, 173-193.
Pereira, C. M. N. A., Schirru, R. e Martinez, A. S. (1998) Learning an Optimized Classification System From a Data Base of Time Series Patterns Using Genetic Algorithm. In: Ebecken, N. F. F (ed), Data Mining, 1 ed., Computational Mechanics Publications, WIT Press, England.
Suich, J. E. and Honec, H. C., The HAMMER System Heterogeneous Analysis by Multigroup Methods of Exponentials and Reactors, Savannah River Laboratory, Aiken South Carolina, 1967.