AN EVOLUTIONARY APPROACH FOR ROBUSTNESS TESTING

Thaise Yano, Eliane Martins
Institute of Computing, State University of Campinas, P.O.Box 6176, 13083-970, Campinas, SP, Brazil

Fabiano L. de Sousa
National Institute for Space Research, São José dos Campos, SP, Brazil

Keywords: Robustness testing, Protocol testing, Evolutionary testing.

Abstract: In this paper we present an evolutionary testing approach to automatically generate robustness test sequences to test a communication protocol, modeled as an extended finite state machine (EFSM). The model represents the normal situation as well as in presence of faults, which makes the model too large to be treated by conventional test case generation approaches, because of the risk of combinatorial explosion. To cope with this problem, we use a testing approach based on test purposes. To search sequences that satisfy the test purposes, we use two evolutionary algorithms: the Generalized Extremal Optimization (GEO) and a Genetic Algorithm (GA). For the moment, only the control flow part of the model is taken into account. Results show that the approach is viable and potentially useful to consider data flow part of complex EFSM models.

1 INTRODUCTION

Robustness testing intends to determine whether a system has an acceptable behavior in presence of invalid inputs or stressful environmental conditions. We consider the model based testing approach for the robustness testing of communication protocol modeled as an extended finite state machine (EFSM).The model represents the protocol behavior in the normal situation (nominal specification) as well as in the presence of faults. The nominal specification is extended in order to incorporate the faults and stressful conditions. The input and output domain of the model are increased, respectively, by events representing invalid or inopportune inputs and extra output, such as error messages or exceptions. After this integration, the size of the increased specification becomes very large. In conventional test generation approaches, large state space can cause the combinatorial explosion problem. To cope with this problem, we test the robustness using an approach based on test purposes. A test purpose defines specific parts of a system to be tested. A test purpose consists of a set of events to be performed or a set of states to be reached. They are determined by practical needs or by system certification requirements. In this paper, we propose applying evolutionary algorithms (EA) to find test sequences that satisfy the test purposes which represent the robustness aspects of a protocol. The test cases are derived from the protocol specification modeled as an EFSM and, for the moment, only the control flow part is taken into account. The automated test data generation from an EFSM is an open research problem.

In the context of robustness testing, evolutionary algorithms have been applied to derive test scenarios that would violate critical requirements of the system (Schultz et al., 1993; Briand et al., 2005; Del Grosso et al., 2005). In contrast to these works, we derive the test cases from the simulator of a communication protocol and do not use any reachability technique.

Two evolutionary algorithms are used: Genetic Algorithm (GA) and Generalized Extremal Optimization algorithm (GEO) (De Sousa et al., 2003). GEO is a recently proposed EA that has been shown to be a competitive alternative to the GA in many applications (De Sousa et al., 2007; Abreu et al., 2007), being at the same time much easier to set to be applied on a given problem than the GA. This work presents a new application of GEO for software testing. In contrast to most works, a discrete encoding of the design variables is used in our implementation, instead of binary values. The performance of these EAs are compared with a "blind" random search approach.

The remainder of this paper is organized as follows. Section 2 presents the EAs applied in this work. Section 3 presents our approach for robustness testing and some experiments. Section 4 remarks some conclusions and future works.

2 EVOLUTIONARY ALGORITHMS

Genetic Algorithm (GA) is one of the evolutionary algorithms most used and known nowadays. The main steps of a generic GA consist of selecting individuals with higher fitness for the next population and evolving individuals with crossover and mutation operators during a given number of generations. We use a GA with one-point crossover (a single point to recombine two individuals), simple mutation (one value that represents an individual is changed), selection by the roulette wheel scheme (individuals are chosen proportional to its fitness) and integer codification.

The Generalized Extremal Optimization (GEO) algorithm (De Sousa et al., 2003) is an evolutionary algorithm based on a model of natural evolution (Bak and Sneppen, 1993). The main steps of GEO are presented below. Firstly, the population of N design variables is initialized randomly with uniform distribution. In the second step, for each variable is associated a fitness value, given by $\Delta_i = F_i - ref$. F_i is the value of the objective function when the variable is mutated to other value mut_i of the variable domain and ref is a given value of reference (e.g., zero). After the Δ_i calculation, the value of the variable i returns to its original one. This process is repeated to all variables. In the next step, all variables are ranked by their fitness value. The first position ($k = 1$) of the ranking belongs to the least adapted variable and the last position ($k = N$) to the best adapted one. For a maximization problem, the highest value of Δ_i means $k = 1$ and the lowest value means $k = N$. A variable j is selected to be mutated according to the probability distribution $P \sim k^{-\tau}$, where k is the rank of the variable j and τ a free control parameter. If the stopping condition is not achieved, the algorithm returns to second step.

3 THE PROPOSED APPROACH

Robustness testing in this paper consists in verifying whether the protocol has an acceptable behavior in presence of unexpected inputs that can be caused by bugs in parts of the host systems or links. We aim to test whether the error detection and recovery mechanisms were implemented according to the specification. The evolutionary approach proposed here for robustness testing is described below. According to the specification, a model is built to represent the protocol behavior. For the test generation process, the model is simulated given a test sequence. Note that it is not the implementation under test but only an executable version of the protocol model (simulator). The simulator code is manually instrumented in order to track the triggered transitions by a test sequence. In order to reduce test effort and costs, the test selection is based on test purposes. As we focus on robustness testing, the test purposes are the transitions corresponding to unexpected inputs (T_{target}). The EA generates test sequences seq trying to cover T_{target}. The instrumented simulator takes seq as input and produces the transitions triggered by seq. The test purposes coverage is evaluated by the EA through the objective function that encodes the test criterion. After the stopping criterion of the evolutionary algorithm is achieved, the best test sequence found during the search is returned.

The behavior model used in our approach is an EFSM composed of states, input events, output events, variables, parameters and a state transition function. The function takes the current state and an input event, verifies if the associated guard is satisfied and, in affirmative case, the transition is triggered, returning output events and bringing the machine to the next state. The guard is a logical expression involving conditions on parameters and variables.

Each individual of the population in GA is represented by a test sequence defined as $seq = \{e_1, e_2, \ldots, e_N\}$, where e_i is a input event and N is the sequence size, considering EFSM as the protocol model. While in GEO, the test sequence represents the entire population. Note that each e_i represents a design variable. The set of transitions to satisfy the test purposes is $T_{target} = \{t_1, t_2, \ldots, t_w\}$, where $t_i \in T$. The objective function to be optimized is defined as:

$$Maximize \quad F(seq) = c/|T_{target}| \tag{1}$$

where $|T_{target}|$ is the cardinal number of T_{target} and $c = |T_{triggered} \cap T_{target}|$ is the number of triggered transitions $(T_{triggered})$ in common with T_{target}.

As an example of how a generated sequence could trigger a set of transitions in a EFSM and how the value of the objective function is calculated, consider the machine M_1 shown in Figure 1. If the test sequence is chosen to have size 4, there will be 4 design variables to be operated by GEO or GA. If the test sequence generated by one of these algorithms is $seq = \{e, a, d, c\}$ it will trigger the set $T_{triggered} = \{t_1, t_5, t_7\}$. Considering c the unexpected input, the set of transitions to satisfy the test purposes is $T_{target} = \{t_3, t_4, t_7\}$ and the objective function value associated with the test sequence seq will be 1/3 = 0.3333.

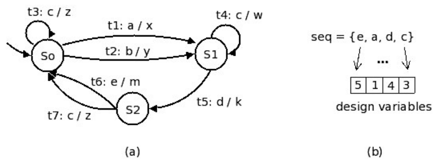

Figure 1: Example: (a) EFSM M_1; (b) test sequence.

3.1 Experiments

To illustrate the application of the proposed approach for robustness testing, the Wireless Transaction Protocol (WTP) is used as the software under test. The protocol simulator is generated by the SMC[1] (State Machine Compiler) tool. SMC takes an EFSM and, in turn, generates a source code of the machine in Java.

The performance of GEO, GA and random testing (RT) on generating test sequences to cover the transitions corresponding to the unexpected inputs of the protocol are compared. In order to adequate the EAs for the problem being tackled, it is necessary to tune their free parameters. To set the parameters of GEO and GA, 10000 function evaluations were used as stopping criterion. Note that the RT has no free parameters to be tuned. Independently of the test sequence size, the performance of GEO does not present much differences for $\tau \geq 4$, then $\tau = 4$ is used to perform the complete experiments. For the genetic algorithm implementation, as described in Section 2, we use a Java framework called JGAP[2]. The best results were obtained with *pop* of 100, p_c of 0.4 and p_m of 0.01. The tuning process of GA took approximately 20 minutes for each sequence size. It demanded more time than GEO that took around 6 minutes for each sequence size. It was used a Pentium 4 with 3.00 GHz and 1 GB of RAM memory.

Two experiments were performed to: *i*) determine the effectiveness of the approach for test case generation; *ii*) determine the effect of T_{target} and test sequence size on the algorithms performance.

3.1.1 Experiment 1

For robustness testing, the effectiveness of the algorithms is given in terms of the number of executions of the objective function necessary to cover the test purposes ($T_{exception}$) that represent the robustness properties to be tested. $T_{exception}$ is the 25 transitions that correspond to the exceptions specified in the WTP documentation. In other terms, we count the number of executions of the model necessary to satisfy the test purposes. Measures of the effectiveness were taken for different test sequence sizes: 32, 64 and 128. The average of the best results was obtained in the 20 runs and the maximum number of function evaluations was 100000 for each run. The computing time of each run was less than 5 seconds for both GA and GEO.

Figure 2 shows the comparison among GEO, GA and RT for the coverage of $T_{exception}$ with sequence size of 64 and 128. From Figure 2 it can be seen that GEO and GA have a similar performance and they clearly outperform RT.

[1]Available in http://smc.sourceforge.net.

[2]Available in http://jgap.sourceforge.net/

In this experiment, RT did not completely cover $T_{exception}$. The evolutionary algorithms have better performance to satisfy the test purposes with sequence size of 128.

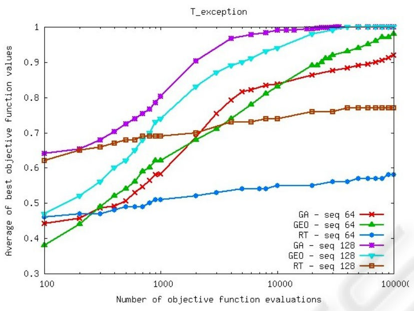

Figure 2: $T_{exception}$ coverage with different sequence sizes.

3.1.2 Experiment 2

Another experiment was performed to verify the effect of test sequence size and target transitions set size on the performance of the algorithms. Different sets of transitions T_{target} were randomly selected, differently from $T_{exception}$. The cardinal number of T_{target} varied from 5 to 25, with increment of 5. The experiment used sequences of size 32, 64 and 128. The maximum number of function evaluations was 100000 for each run. Twenty runs of each algorithm were performed for each combination of sequence size and target transitions set. Figures 3 and 4 show some results.

The results showed that the cardinality of T_{target} influences directly the performance of the algorithms. For higher values of $|T_{target}|$, the algorithms present more difficulty to generate test sequence. This occurs because the objective function becomes more difficulty to be satisfied with more transitions to be covered, since its definition is related to $|T_{target}|$.

The performance of all algorithms improves as the test sequence size increases. Higher test sequence size increases the probability of an uncovered transition to be found as new paths are generated since the machine returns to the initial state (reversible).

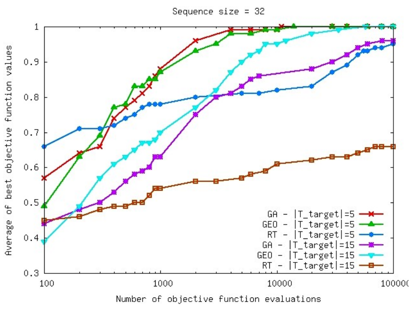

Figure 3: Different T_{target} with sequence size of 32.

The problem of finding test sequence is more complex when $|T_{target}|$ is bigger and the sequence size is smaller. The difference between the performance of the EAs and RT is very clear. Moreover, GEO has better performance than

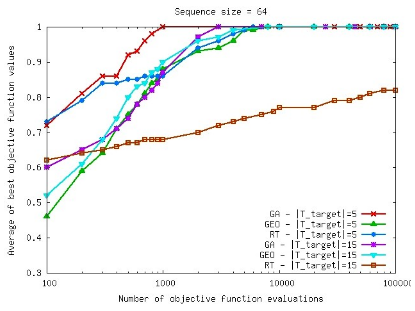

Figure 4: Different T_{target} with sequence size of 64.

GA in this case. Although we need further experiments to perform a thorough analysis of the behavior of the EAs for different models, these preliminary results let us deduce that they can substitute RT for model-based test sequence generation. The experiments shown that test sequence size is also worth to be considered for optimization.

4 CONCLUSIONS

In this paper an evolutionary approach for robustness testing was presented. Evolutionary algorithms were used to generate the test sequences to cover a test purpose, which consist of the unexpected inputs of a communication protocol. The evolutionary algorithms, GEO and GA, were efficient to resolve that problem. Both algorithms clearly outperform the random testing, mainly in more complex problems, as expected. The results indicate that GEO is competitive with GA for this problem. Of course, further experiments are necessary to assure this claim.

In addition to the control flow, the data flow inherent of the EFSM need to be considered.

The use of a multi-objective function is under way. The goal is not only to maximize the transition coverage, but also to minimize the test sequence size.

ACKNOWLEDGEMENTS

This research is supported by CAPES and CNPq.

REFERENCES

Abreu, B. T., Martins, E., and Sousa, F. L. (2007). Generalized extremal optimization: a competitive algorithm for test data generation. In *21st Brazilian Symposium on Software Engineering*, João Pessoa, Brazil.

Bak, P. and Sneppen, K. (1993). Punctuated equilibrium and criticality in a simple model of evolution. *Physical Review Letters*, 71(24):4083–4086.

Briand, L. C., Labiche, Y., and Shousha, M. (2005). Stress testing real-time systems with genetic algorithms. In *GECCO 2005: Proc. of Conf. on Genetic and Evolutionary Computation*, pages 1021–1028, New York, NY, USA. ACM.

De Sousa, F. L., Ramos, F. M., Paglione, P., and Girardi, R. M. (2003). New stochastic algorithm for design optimization. *AIAA Journal*, 41(9):1808–1818.

De Sousa, F. L., Soeiro, F., Neto, A. S., and Ramos, F. M. (2007). Application of the generalized extremal optimization algorithm to an inverse radiative transfer problem. *Inverse Problems in Science and Eng.*, 15(7):699–714.

Del Grosso, C., Antoniol, G., Penta, M. D., Galinier, P., and Merlo, E. (2005). Improving network applications security: a new heuristic to generate stress testing data. In *GECCO 2005: Proc. of Conf. on Genetic and Evolutionary Computation*, pages 1037–1043, New York, NY, USA. ACM.

Schultz, A. C., Grefenstette, J. J., and Jong, K. A. (1993). Test and evaluation by genetic algorithms. *IEEE Expert: Intelligent Systems and Their Applications*, 8(5):9–14.