A case study on the use of genetic algorithms to generate test cases for temporal systems[⋆]

Karnig Derderian[1], Mercedes G. Merayo[2],
Robert M. Hierons[1], Manuel Núñez[2]

[1] Department of Information Systems and Computing, Brunel University
Uxbridge, Middlesex, UB8 3PH United Kingdom,
derderian@karnig.co.uk, rob.hierons@brunel.ac.uk
[2] Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Madrid, Spain,
mgmerayo@fdi.ucm.es, mn@sip.ucm.es

Abstract. Generating test data for formal state based specifications is computationally expensive. In previous work we presented a framework that addressed this issue by representing the test data generation problem as an optimisation problem. In this paper we analyze a communications protocol to illustrate how the test case generation problem can be presented as a search problem and automated. Genetic algorithms (GAs) and random search are used to generate test data and evaluate the approach. GAs show to outperform random search and seem to scale well as the problem size increases. We consider a very simple fitness function that can be used with other evolutionary search techniques and automated test case generation suites.

1 Introduction

As computer technology evolves the complexity of current systems increases. Critical parts/aspects of some system are specified using formal specifications in order to better understand and model their behaviour. Communication protocols and control systems, amongst others, have used formal specifications like finite state machines. Unfortunately, in most cases it cannot be guaranteed that system implementations fully comply to the specifications.

Even though testing [1, 2] is an important part of the system development process that aims to increase the reliability of the implementation, it can be very expensive. This motivates the research in the combination of formal methods and testing [3, 4] since progress in this line of work helps to (partially) automatize the testing process.

In previous work [5] we addressed the issues related to generating test sequences for temporally constrained Extended Finite State Machine (TEFSM) based systems. We focused on generating timed feasible transition paths (TFTPs)

[⋆] Research partially supported by the Spanish MEC project TESIS (TIN2009-14312-C02-01).

with specific properties that can in turn be used to generate test input. The problem of generating these paths is represented as a search problem and Genetic Algorithms (GA) can be used to help automate the test data generation process. In short, a GA is a heuristic optimisation technique which derives its behaviour from a metaphor of the processes of evolution in nature [6, 7]. GAs have been widely used in search optimisation problems. GAs are known to be particularly useful when searching large, multimodal and unknown search spaces since one of its benefits is their ability to escape local minima in the search for the global minimum. In particular, GAs and other meta-heuristic algorithms have been also used to automate software testing [8–12]. In this paper we present a case study to evaluate our theoretical framework. We consider the Class 2 transport protocol [13] and compare the performance of two GAs and a random algorithm when looking for TFTPs.

The rest of the paper is organized as follows. In Section 2 we introduce the main definitions and concepts that will be used during the presentation of our case study. In Section 3, what constitutes the bulk of the paper, we present our case study. Finally, in Section 4 we present our conclusions and some lines for future work.

2 Preliminaries

In this section we review the concepts that will be used along the paper. In particular, we will introduce the notions of timed extended finite state machine and timed feasible transition path, and discuss on the fitness function that will guide our GAs. This part of the paper was already presented in our previous work [5], where the reader is refered to find further explanations.

We assume that the number of different variables is m. If we assume that each variable x_i belongs to a domain D_i thus the values of all variables at a given point of time can be represented by a tuple belonging to the cartesian product of $D_1 \times D_2 \times ... \times D_m = \Delta$. Regarding the domain to represent time we define that time values belong to a certain domain **Time**.

Definition 1. *A TEFSM M can be defined as $(S, s_0, V, \sigma_0, P, I, O, T, C)$ where S is the finite set of logical states, $s_0 \in S$ is the initial state, V is the finite set of internal variables, σ_0 denotes the mapping from the variables in V to their initial values, P is the set of input and output parameters, I is the set of input declarations, O is the set of output declarations, T is the finite set of transitions and C is such that $C \in \Delta$.*

*A transition $t \in T$ is defined by $(s_s, g_I, g_D, g_C, op, s_f)$ where s_s is the start state of t; g_I is the input guard expressed as (i, P^i, g_{P^i}) where $i \in I \cup \{NIL\}$; $P^i \subseteq P$; and g_{P^i} is the input parameter guard that can either be NIL or be a logical expression in terms of variables in V' and P' where $V' \subseteq V$, $\emptyset \neq P' \subseteq P^i$; g_D is the domain guard and can be either NIL or represented as a logical expression in terms of variables in V' where $V' \subseteq V$; $g_C : \Delta \to$ **Time** is the time the transition needs to take to complete; op is the sequential operation which is made of simple output and assignment statements; and s_f is the final state of t.*

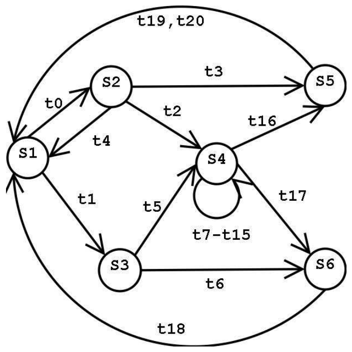

Fig. 1. Class 2 transport protocol TEFSM M_1. The transition table is on Fig. 2.

A TEFSM M is deterministic if any pair of transitions t and t' initiating from the same state s that share the same input x have mutually exclusive guards. A TEFSM M is strongly connected if for every ordered pair of states (s, s') there is some feasible path from s to s'. A configuration for a TEFSM M is a combination of state and values of the internal variables V of M.

We assume that any TEFSM considered in this paper is deterministic and strongly connected. For example, consider the Class 2 transport protocol [13] represented as a TEFSM in Figure 1.

A timed feasible transition path (TFTP) for state s_i to state s_j of a TEFSM M is a sequence of transitions initiating from s_i that is feasible for at least one combination of values of the finite set of internal variables V (configuration) of M and ends in s_j.

An input sequence (IS) is a sequence of input declarations $i \in I$ with associated input parameters $P^i \subseteq P$ of a TEFSM M.

A predicate branch (PB) is a label that represents a pair of g_{P^i} and g_D for a given state s and input declaration i. A PB identifies a transition within a set of transitions with the same start state and input declaration.

An abstract input sequence (AIS) for M represents an input declaration sequence with associated PBs that triggers a TP in the abstracted M.

We use a very simple fitness function that combines a transition ranking (how likely is to take this transition) and a temporal constrain ranking (how complex the time constraint of a transition is).

Definition 2. *The fitness is a function that given a TP of a TEFSM M, sums the penalty points (assigned through the transition ranking process for M and the temporal constrain ranking for M) for each of the transition of the TP.*

We chose to give equal weight to the rankings following the conclusions of a similar experiment [14] where different weights were used for a similar multi-optimisation problem. However it is possible to consider different ways to combine the two matrices in the fitness function.

t	input	output	feasibility rank	temporal rank
t_0	ICONreq	!CR	0	0
t_1	CC	!ICONconf	0	0
t_2	T_expired	!CR	2	0
t_3	T_expired	!IDISind	1	1
t_4	IDATreq	DT	0	0
t_5	AK		6	1
t_6	AK		6	1
t_7	AK	DT	5	0
t_8	AK	!IDISind	4	0
t_9	T_expired	DT	3	0
t_{10}	T_expired	!IDISind	2	1
t_{11}	DR	!IDISind	0	2
t_{12}	DR	!IDISind	0	2
t_{13}	DR	!IDISind	0	2
t_{14}	DR	!IDISind	0	2

Fig. 2. Temporal constraint ranking and feasibility ranking for all transitions in M_1

3 Case study

The Class 2 transport protocol M_1 is presented in Figure 1 and the corresponding transition table (excluding the conditions and temporal constraints) is shown in Figure 2. This table also shows the ranked transition table for M_1. For example t_3 and t_{10} share the same temporal constraint classification and therefore are ranked lower than some other transitions however they have different feasibility ranking due to the differently classified guards they have.

The search for a TP that is likely to be feasible and yet have complex temporal constraints is represented as a fitness minimisation problem. The GA is then used to search for appropriate solutions. The same computational effort is also used with a random TP generator using the same fitness function and result verification as the GA. This search problem uses a fitness function that rewards transition sequences with higher ranked transitions and penalises invalid transitions. It produces a numerical value potentially showing how close an input sequence is to defining a valid TFTP. The fitness function represents the search for a TFTP sequence as a function minimisation problem so an AIS with a lower fitness value is considered to be more likely to form a TFTP since it is made up of more highly ranked transitions.

The fitness does not guarantee that a particular transition path can be triggered or that it contains the most complex temporal constraints in M. It makes sure that it is constructed using consecutive transitions that are highly ranked. The verification process then checks if an IS can be generated to trigger such a TP. The verification method consists in evaluating a TP by resetting M to its initial configuration and attempt to trigger the TP in the simulated implementation. The process is repeated several times and the overall result of how many times the TP was correctly triggered are counted and compared to the times it failed. Hence an estimation is derived to measure the feasibility of these TPs.

In our example we looked at a relatively simple temporal constraints range (hence the small range of rankings on Figure 2) and it was easy to manually check the complexity of the temporal constraints for each transition. This was sufficient for our case study, but defining an automated estimation measure for the temporal qualities of a TP remains future work.

In order to compare the performance of the GA and Random algorithms TFTP generation two metrics are used. *State coverage* is the number of cases where at least one TFTP was generated for every TFTP size attempted from each state in M and *success rate* is the number of TFTPs that were generated compared to the total number of attempts it took to generate the results.

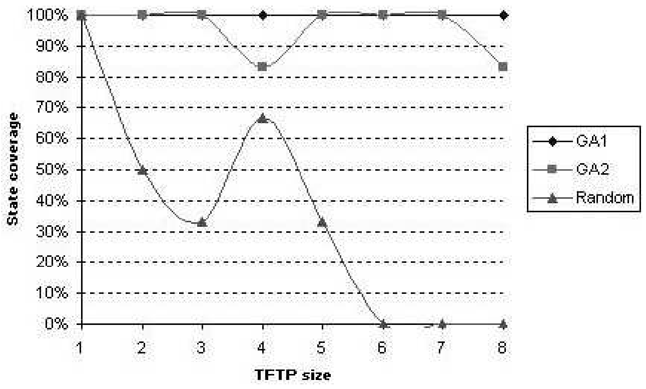

Fig. 3. State coverage for PB notation TFTPs generated using GA and Random generation algorithms for M_1 with 1-8 transitions

In this case study two slightly different GAs were used to compare their performance when applied to this problem. The first GA used a single point crossover and mutation while the second used a complex multiple point crossover and mutation. In general the second GA tended to find a solution slightly faster than the first GA, but they produced the same results.

Figure 5 represents a summary of the result averages. In general the results show that the GAs seem to perform better than the Random generation method according to both metrics. Figure 3 represents the state coverage results for all the different TFTP sizes. GA1 performs well and GA2 fails to find only one TFTP of size 4 and one of size 8, while the random generation algorithm performance peaks when generating TFTPs of size 4 and declines as the TFTP size increases. Clearly the GAs outperform the Random generation method for TFTPs of more than one transition. Figure 4 represents the success rate results for all the different TFTP sizes in our case study. The high fluctuation here can be explained by the different degree of difficulty in generating TFTPs of different sizes for some states. This relates to the guards of transition $t14$, which is one of the core transitions. Although the guard conditions are complex in the context of M_1 they are very easy to satisfy. Hence the random algorithm can easily select $t14$ in its TP search, while the GAs try to avoid it without realising that it should not. GA2 shows the best performance with an average performance of 65%. GA1 performs mostly better than the random generation algorithm, except for TFTPs of size 4 (average performance of 54%). For TFTPs of size 4 the random algorithm performs slightly better than GA1 and GA2. The random generation method did not find any TSTPs for sizes 4 to 6 while the GAs have different success rate for each size. This shows how different states have different properties. Hence future work may focus on more analysis of the guards and the temporal conditions to even out the search performance.

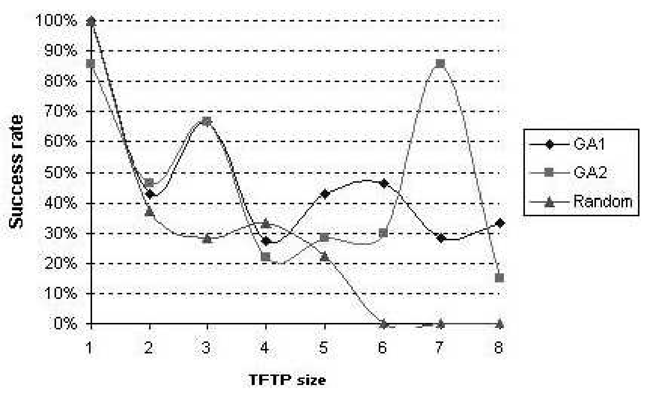

Fig. 4. Success rate for PB notation TFTPs generated using GA and Random generation algorithms for M_1 with 1-8 transitions

For both metrics the two GA search algorithms perform on average better than the random generation algorithm. This suggests that the fitness function here helps guide a heuristic search for TFTPs. In Figure 3 and Figure 4 we observe that as longer TFTPs (and possibly when larger TEFSMs) are considered, the heuristics seems to perform increasingly better than the random generation algorithm when given equal processing effort in terms of fitness evaluations and TFTP verifications. On all occasions the TFTPs generated by the GAs the had equivalent or more complex temporal constraints compared to those generated using the random TP generation method. For a TEFSM of this size, as in our case study, it is expected to have similar performance for the small TFTPs because the search space in those situations is not that big. However as the search space is increased (in or case study by increasing the size of the TFTPs) it becomes clear that a random generation approach finds it hard to generate TPs that feasible and satisfy the temporal constraints.

The state coverage metric is the easier one to satisfy. Not surprisingly the GAs found at least one TFTP for every state in M_1. This measure however discards all the unsuccessful attempts to generate a given TFTP. Hence the success rate metric considers those unsuccessful attempts as well. The success rates results are lower but the GAs seem to outperform the random algorithm.

	State Coverage	Success rate
GA1	100%	48%
GA2	96%	47%
Random	35%	28%

Fig. 5. GA and Random search result averages for the Class 2 protocols for TFTPs with 1-8 transitions.

Overall both GAs performed well and generated very similar results. This indicates that the fitness function and the TP representation represent the problem of TFTP generation reasonably well. Hence future work can focus on refining the fitness function and evaluations of larger TEFSMs.

4 Conclusions and Future work

This paper reported on the application to a communications protocol of a computationally inexpensive method to address the important problem of test data generation for TEFSMs. By taking as initial step our previous work [5], we defined a fitness function that yields some positive results when GA search is applied to the problem. The GA almost fully satisfies the coverage criteria defined and increasingly outperforms random generation as the TFTP size increases. The success rate fluctuated in this case study but the average success rate of the GA generated results was almost double that of the randomly generated results. Overall the limited results suggest that the approach scales well and could possibly be applied to larger TEFSMs. As a conclusion, we claim that our computationally inexpensive fitness function may be used to aid the generation of potentially computationally expensive test input generation sequences in a computationally inexpensive way.

Future work may focus on refining the fitness function to take into account several difficulties to estimate transitions. In addition, it would be interesting to apply our methodology to larger systems. Work on the combination of other related research, like alternative approaches to test sequence generation, with feasibility estimation and temporal constraint satisfaction can also be considered to aid the generation of test input sequences for TEFSMs.

References

1. Myers, G.: The Art of Software Testing. 2nd edn. John Wiley and Sons (2004)
2. Ammann, P., Offutt, J.: Introduction to Software Testing. Cambridge University Press (2008)

3. Hierons, R., Bowen, J., Harman, M., eds.: Formal Methods and Testing, LNCS 4949. Springer (2008)
4. Hierons, R., Bogdanov, K., Bowen, J., Cleaveland, R., Derrick, J., Dick, J., Gheorghe, M., Harman, M., Kapoor, K., Krause, P., Luettgen, G., Simons, A., Vilkomir, S., Woodward, M., Zedan, H.: Using formal methods to support testing. ACM Computing Surveys **41**(2) (2009)
5. Derderian, K., Merayo, M., Hierons, R., Núñez, M.: Aiding test case generation in temporally constrained state based systems using genetic algorithms. In: 10th Int. Conf. on Artificial Neural Networks, IWANN'09, LNCS 5517, Springer (2009) 327–334
6. Goldberg, D.E.: Genetic Algorithms in search, optimisation and machine learning. Addison-Wesley Publishing Company, Reading, Mass. USA (1989)
7. Srinivas, M., Patnaik, L.M.: Genetic algorithms: A survey. IEEE Computer **27** (1994) 17–27
8. Jones, B.F., Eyres, D.E., Sthamer, H.H.: A strategy for using genetic algorithms to automate branch and fault-based testing. The Computer Journal **41**(2) (1998) 98–107
9. Michael, C.C., McGraw, G., Schatz, M.A.: Generating software test data by evolution. IEEE Transactions on Software Engineering **27**(12) (2001) 1085–1110
10. McMinn, P.: Search-based software test data generation: a survey. Software Testing Verification and Reliability **14**(2) (2004) 105–156
11. Derderian, K., Hierons, R.M., Harman, M., Guo, Q.: Automated Unique Input Output Sequence Generation for Conformance Testing of FSMs. The Computer Journal **49**(3) (2006) 331–344
12. Harman, M., McMinn, P.: A theoretical and empirical study of search-based testing: Local, global, and hybrid search. IEEE Transactions on Software Engineering **36**(2) (2010) 226–247
13. Ramalingom, T., Thulasiraman, K., Das, A.: Context independent unique state identification sequences for testing communication protocols modelled as extended finite state machines. Computer Communications **26**(14) (2003) 1622–1633
14. Derderian, K.: Automated test sequence generation for Finite State Machines using Genetic Algorithms. PhD thesis, Brunel University (2006)