Empirical study on the efficiency of search based test generation for EFSM models

Ruilian Zhao*, Mark Harman† and Zheng Li†
*Department of Computer Science, Beijing University of Chemical Technology, Beijing 100029, China
†CREST, Department of Computer Science, King’s College London, Strand, London, United Kingdom

Abstract—Experimental work in software testing has generally focused on evaluating the effectiveness and efficiency on various source code programs. However, an important issue of testing efficiency on the model level has not been sufficiently addressed, and hitherto, no empirical studies exist. This paper presents an automated test data generation system for feasible transition paths (FTP) on Extended Finite State Machines (EFSM) models and investigates the statistical properties of testing efficiency using statistical tests for correlation and formalisation according to the test data generated by applying the system on four widely used EFSM models. An important and encouraging finding is a close positive correlation between test generation cost and the number of numerical event variables (NNEV) or the number of numerical equal operators in conditions (NNEOC) on a FTP. In addition, as the NNEOC increases, there is a raising correlation between the test generation cost and the length of path with events variables (LPEV) or the number of numerical event variables on a path (NNEV), and NNEV increases linearly with the LPEV. Furthermore, empirical study shows that there is a very strong exponential relationship between test generation cost and NNEV or LPEV only when NNEOC is considerable. The results provide a significant guide to predict the testing efficiency for EFSM models.

Keywords—Search; Test generation; EFSM

I. INTRODUCTION

Testing from formal specifications offers a simpler and more rigorous approach to the development of functional testing than standard testing techniques. Finite-state machines (FSMs) and extended finite-state machines (EFSMs) are among the most popular formal specifications. They are widely used in a number of industrially significant specifications, such as SDL, Estelle, Statecharts, UML, and so on. Although automated test sequence generation methods made substantial contributions toward test generation on systems specified as FSMs [6], [9], [13], [16], [21], [27], test generation for the EFSM models remains an open research problem. The difficulty of automating test generation for the EFSM models arises from the fact that, in general, an EFSM model contains infeasible paths due to the existence of the context variables. Moreover, finding a set of test data to trigger a given feasible path in an EFSM is a hard task, too [18]. Many techniques producing a set of paths for generating test sequences from an EFSM have already been reported in the literature [5], [7], [8], [10], [18]. However, there is comparatively little work on producing real test data for feasible paths in EFSMs.

On the other hand, experimental work in software testing has generally focused on comparing and evaluating the effectiveness and efficiency of different coverage criteria on various source code levels [3], [12], [15], [17], [19]. Gallagher et al. [14] reported the factors, the number of test data variables being generated and the length of test path, which affect the performance of the test data generator for Ada software system. However, the paucity of the efficiency analysis on test data generation at the model level of abstraction means that the software tester has little knowledge on potential factors that affecting the efficiency of test data generation in EFSM models.

Thus, this paper first aims to develop the infrastructure of automatic test data generation for EFSM models that produce real data to trigger feasible transition paths. Secondly, this paper provides empirical results on efficiency analysis of test data generation for a set of state-based models to address the following questions:

• Which factors affect the performance of test data generation in EFSM models?
• Which is a decisive factor?
• What correlation exists between the test generation efficiency and these factors?
• Which underlying regression model is better for predicting, linear or exponential?

The primary contributions of the paper are as follows:

1) The paper presents a genetic algorithm-based test data generation system for feasible transition paths in EFSM models, and empirically validates the efficiency of the system by applying the system to a set of EFSM models, with or without an EXIT state.

2) The paper also empirically confirms that the number of numerical equal operators in conditions (NNEOC) plays a key effect on the efficiency of test data generation. Furthermore, empirical study shows that there is a very strong linear correlation between the number of numerical event variables (NNEV) and the length of path with event variables (LPEV) on a feasible transition path, and the test generation cost grows exponentially along with the increasing of LPEV or NNEV when NNEOC is considerable.

The remainder of this paper is organized as follows. Section II defines two types of complete path, and introduces the main principle of test data generation algorithm using
GA on EFSM models. Section III briefly describes the test generation implementation, and gives some metrics used in test generation efficiency analysis. Section IV reports the experimental results and discussion. Section V reviews related work. Finally, conclusions and future work are given in Section VI.

II. TEST DATA GENERATION ON EFSM MODEL

One of the challenges with testing EFSM models is how to correctly account for path. This is because EFSM models can be non-terminating (i.e. without an EXIT state), which breaks traditional path used in testing.

In this section, we first introduce the syntax of EFSM models. Then we define complete transition path and potential feasible path which are used in the research reported in this paper. Finally, we describe how to generate test data using genetic algorithm for a feasible transition path from EFSM model.

A. Extended Finite State Model

An extended finite state machine (EFSM) is a 6-tuple \((S, V, I, O, T, s_0)\), where \(S\) is a nonempty finite set of states, \(V\) is a nonempty set of internal/context variables, \(I\) is a nonempty set of input interactions, \(O\) is a nonempty set of output interactions, \(T\) is a nonempty set of transitions, and \(s_0\) is an initial state [18]. Each member of \(I\) is expressed as event(inputlist) meaning the interaction of event occurs with a list of input parameters inputlist, which is disjointed from \(V\). Each member of \(O\) is expressed as action(outlist) meaning action occurs with a formal list of parameters outlist. Each parameter in outlist can be replaced by a suitable variable from \(V\), an input interaction parameter, or a constant. Each element \(t\) of \(T\) is a 5-tuple \((source, target, input, condition, action)\). Here, source and target are the states in \(S\) representing the source state and the target state of \(t\), respectively. input is either an event from \(I\) or empty. condition is a predicate as a set of logical expressions in terms of the variables in \(V\), the parameters of the event and some constants. action is a sequence of actions which consists of statements such as output statements and assignment statements and so on (we assume a standard expression language including assignments). All parts of a \(t\) are optional.

A state transition \(t\) occurs when one of the machine’s transitions is taken. If a transition \(t\) has a condition \(c\) on the internal variables and input parameters, then \(c\) must be satisfied in order for \(t\) to be taken. A self-looping transition is a transition \(t\) where the source of \(t\) is the same as the target of \(t\). A set of distinct transitions may have an identical source and an identical target. A final transition is one whose target is an EXIT state that has no outgoing transitions in a terminating EFSM model, or is one whose target is a START state in a non-terminating EFSM model.

B. Complete Paths in EFSM Model

A path is usually presented as a sequence of nodes or edges. By path of an EFSM we mean a sequence of adjacent transitions of an EFSM. Because EFSM can be non-terminating, this has led to two types of complete path definitions.

Definition 1 (Complete transition path). A complete transition path is any path \(\pi = t_1 t_2 \cdots t_i \cdots t_n\) that source\((t_1) = \text{START state}, \) target\((t_n) = \text{EXIT state}\) and target\((t_i) = \text{source}(t_{i+1})\) \((1 \leq i < n)\) in a terminating EFSM model.

Definition 2 (K Complete transition path). A K complete transition path is any path \(\pi = t_1 t_2 \cdots t_i \cdots t_n\) that source\((t_1) = \text{START state}, \) target\((t_n) = \text{START}, \) and target\((t_i) = \text{source}(t_{i+1})\) in a non-terminating EFSM model, and there is \(K-1\) transition \(t_e\) in path \(\pi\) whose target\((t_e) = \text{START state}\) \((1 \leq i, e < n)\).

Definition 3 (Condition conflict). A transition may not be traversed if there are conflicting conditions in the paths. A given path has a condition conflict if there exists a variable \(v\) and a pair of transition \((t_i, t_j)\), such that the current values of the variable \(v\) makes the condition of \(t_i\) to be True(False), but results False(True) in the condition of \(t_j\).

Definition 4 (Potential feasible path). A path that is free of condition conflict is called a potential feasible path.

C. GA based Test Data Generation

Genetic Algorithms work with populations of candidate solutions to a problem. Our specific problem is to use a genetic algorithm to search a set of input data that can traverse a potential (K) complete FTP in EFSM models. More generally, given a particular (K) complete path \(\pi\) in an EFSM, \(\pi = s_1 e_1/c_1/a_1 \rightarrow s_2 e_2/c_2/a_2 \rightarrow s_3 \cdots e_m/c_m/a_m \rightarrow s_{m+1}\), where \(e_i\) is an event, \(c_i\) is a condition, and \(a_i\) is a sequence of actions, an individual is a list of input values, \(x = (x_1, x_2, \cdots, x_n)\), corresponding to all parameters of the events \(e_1, e_2, \cdots, e_m\) in the order they appear. If the sequence of events, having the parameter values \(x_1, x_2, \cdots, x_n\), determines the transitions on the path \(\pi\) and validates the condition \(c_i\) of each transition, then \(x\) is a solution for path \(\pi\). This means that each reached restrictions imposed by the path must be solved and previous predicates must remain solved, despite the changes on input values in the search process. Figure 1 shows a simple transition path with three transitions. The test input is the sequence of \((e1(a, b), e2, c3)\), where the variable \(a\) and \(b\) are replaced by real values. For example, \((e1(2, 3), e2, c3)\) is a validated test data that can traverse the path while \((e1(-1, 3), e2, c3)\) is not, as the condition on T2 is failed. Therefore, \((2, 3)\) and \((-1, 3)\) are individuals but only \((2, 3)\) is a solution for this example.
The genetic algorithm evaluates each individual by executing each transition on a potentially complete FTP with the values encoded in the chromosome’s genes. A fitness function, assigning a score (fitness) to each chromosome in the current population, is used to compare the individuals and to differentiate their performance in each population. The fitter individuals are the ones which follow more transitions from the given path.

III. EXPERIMENTAL SETUP

The experimental approach is straightforward. Firstly, potentially complete feasible paths with different lengths, varying from 3 to 50 depending on EFSM models, are produced by employing Breadth-First search. Secondly, for each path length, 5 paths are picked up to develop test data. For each path, ten test cases are generated by applying the GA. Finally, the test generation efficiency is analyzed in detail by using the statistical analysis tool SPSS.

A. Subjects

The study concerns 4 EFSM models which come from previous model-based studies [1], [20]. Each model has two versions. One includes EXIT state and other is free of EXIT state, denoted by the corresponding model name following _noexit. Table I presents summary information concerning the subjects, including the model’s size in terms of the number of states, the number of transitions and a brief description.

<table>
<thead>
<tr>
<th>Models</th>
<th>Number of States</th>
<th>Number of Transitions</th>
<th>EXIT State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATM</td>
<td>9</td>
<td>23</td>
<td>Yes</td>
<td>Automated</td>
</tr>
<tr>
<td>ATM_noexit</td>
<td>9</td>
<td>24</td>
<td>No</td>
<td>Teller Machine</td>
</tr>
<tr>
<td>Cashier</td>
<td>12</td>
<td>21</td>
<td>Yes</td>
<td>Cashier</td>
</tr>
<tr>
<td>Cashier_noexit</td>
<td>12</td>
<td>22</td>
<td>No</td>
<td>Machine</td>
</tr>
<tr>
<td>CruiseControl</td>
<td>5</td>
<td>17</td>
<td>Yes</td>
<td>Cruise Control</td>
</tr>
<tr>
<td>CruiseControl_noexit</td>
<td>5</td>
<td>18</td>
<td>No</td>
<td>System</td>
</tr>
<tr>
<td>FuelPump</td>
<td>13</td>
<td>23</td>
<td>Yes</td>
<td>Fuel Pump</td>
</tr>
<tr>
<td>FuelPump_noexit</td>
<td>13</td>
<td>26</td>
<td>No</td>
<td>System</td>
</tr>
</tbody>
</table>

B. Test generation system

In order to achieve the automatic test data generation and evaluate the efficiency for feasible transition paths in EFSM models, we develop a test data generation system for EFSM models using GA. The system supports not only the test generation of integer and real data types, but also non-numerical types such as Boolean and characters types.

Figure 2 shows the test data generation algorithm using GA. The implementation consists of two steps.

TestGeneration(efsm, popsize, Imax, Pc, Pm):
Input : efsm: EFSM model to be tested
        popsize: Population size
        Imax: Maximum iteration number
        Pc: Crossover probability
        Pm: Mutation probability
Output: a set of test data for efsm
Create feasible transition paths
Randomly generate initial population popu
repeat fitness ← evaluate(popu)
    popu ← select(popu, fitness)
    popu ← crossover(popu, fitness, Pc)
    popu ← mutate(popu, fitness, Pm)
    fitness ← evaluate(popu)
    popu ← survive(popu, fitness)
until success or iterationnumber > Imax

Step 1: generate complete transition paths

First, complete paths with a variety of lengths are created according to breadth-first search technique, and potential feasible paths are produced by deleting the paths where condition conflicts exist. The path varies in length from 3 to 50. For each potential feasible path, a sequence of events that triggers all transitions in order is extracted and the types of input parameters of the events are identified.

Step 2: test data generation using GA

In this step, for each potential feasible path, find the input parameter values that trigger the path by applying GA. The initial population is generated randomly depending on their data types. The chromosomes are real-encoded, each gene representing one input parameter. Each individual is evaluated by a fitness function. A recent survey on search-based test data generation [24] suggests the notion of approach level and branch distance in order to construct a fitness function. The approach level will evaluate how close a chromosome is to the given path. The branch distance will measure how close is the first unsatisfied pre-condition to being true. So, in our research, the following fitness function is applied.

\[
\text{fitness} = \text{approach}_\text{level} + \text{norm}(d)
\]

\[
\text{norm}(d) = 1 - 1.001^d
\]

Where \(d\) is a branch distance, and \(\text{norm}(d)\) is the branch distance value scaled between [0, 1].

In selection procedure \(\text{select}(\text{population}, \text{fitness})\), the parents are chosen according to their fitness values. This guarantees that the chromosomes with a higher fitness value have a higher likelihood of being selected. In crossover procedure \(\text{crossover}(\text{population}, \text{fitness}, Pc)\), we produce
new offsprings selected by crossover rate $P_c$ based on following computing inspired from [22].

$$y_1 = |0.05 \times (x_1 - x_2) + x_1|$$
$$y_2 = |0.05 \times (x_2 - x_1) + x_2|$$

Where $x_1$ and $x_2$ are the chosen parent individuals, $y_1$, $y_2$ are new individuals after applying the crossover operation, and a fixed 0.05 is chosen since it supplies a better score than other values randomly selected. In mutation procedure $mutate(population, \text{fitness}, P_m)$, an individual is chosen by mutation rate $P_m$, and a new gene will be generated randomly according its data type to substitute the original. After these procedures, population will be evaluated again, and a basic survive procedure $survive(popu, \text{fitness})$ is employed to pick up certain individuals of the offsprings into the next generation according to their fitness which means that better individuals (higher fitness) have a better chance of being chosen.

In the test data generation for EFSM models using GA, the below arguments are set as follows:

- Crossover probability $P_c = 0.7$
- Mutation probability $P_m = 0.08$
- Survival probability $P_s = 0.8$
- Population Size $popuSize = 20$
- Maximum iteration number $Imax = 500000$

GA-based test data generation is an heuristic process. When a new input is created, the EFSM model under the test has to be executed again in order to evaluate its fitness value. The cost of test generation algorithm depends mainly on the number of times the fitness function must be evaluated, i.e., the number of times the EFSM model is executed. In addition, the empirical experiments were done on two PCs and a SUN SPARC station, the time of test data generation varied with the different computers. Therefore, only the number of evaluation (NE) of fitness function during generating a test case for a FTP is considered as the cost of test generation. Thus, in this paper, the efficiency of the test data generation system is examined by the number of evaluations of the fitness function.

### C. Metrics

In order to investigate which factors affect the performance of test data generation in EFSM, the following metrics are considered in this paper.

1. **Length of path (LP):** The number of transitions in a path.
2. **Number of variables (NV):** The number of variables defined or used in a path, including variables used as input parameters (defined) in events, defined in actions, used in conditions or actions on the path.
3. **Number of variables defined in event (NVDE):** The number of variables appeared as input parameters in event sequence within a path. These variables also are called event variables.
4. **Number of variables defined in actions (NVDA):** The number of variables defined in actions within a path.
5. **Number of variables used in conditions (NVUC):** The number of variables used in conditions within a path.
6. **Number of variables used in actions (NVUA):** The number of variables used in actions within a path.
7. **Number of variables defined in event and used in conditions (NVDEUC):** The number of variables defined in events of transition $t_i$, used in conditions of transition $t_i$ or $t_j$ within a path, and there are no other definitions with respect to the variables from transition $t_i$ to $t_j$, if used in $t_j$.
8. **Number of variables defined in actions and used in conditions (NVDAUC):** The number of variables defined in actions of transition $t_i$, used in conditions of transition $t_j$ within a path, and there are no other definitions with respect to the variables from transition $t_i$ to $t_j$.
9. **Number of conditions (NC):** The number of nonempty conditions in a path.
10. **Number of sub-conditions (NSC):** The number of sub-conditions in a path.
11. **Number of equal operators in conditions (NEOC):** The number of equal operators in conditions within a path, including logical equal and numerical equal. Logical equal implies equalling to True or False, and numerical equal means equal in integer, real or character value, not in logical value.
12. **Number of numerical equal operators in conditions (NNEOC):** The number of numerical equal operators in conditions within a path.
13. **Length of path with event variables (LPEV):** The number of transitions whose event is provided with nonempty input parameters within a path.
14. **Number of numerical variables (NNV):** The number of numerical variables defined or used in a path, including variables defined in events or actions, used in conditions or actions on the path. Numerical variables imply that their data type is integer or real.
15. **Number of numerical event variables (NNEV):** The number of numerical event variables within a path.

### D. Data Collection and Description

In order to investigate and illustrate the effectiveness of the test data generation system for EFSM models, we have conducted a substantial number of experiments for the 8 EFSM subjects presented in Table I. For each subject, test cases are generated for potential (K) complete FTPs with different lengths. If test generation fail on a path within maximum iteration number, another path with the same length is chosen. In addition, corresponding numbers of evaluation of the fitness function as well as above metrics...
are recorded during the test generation. Considering the
randomicity of initial population in GA, we delete the
highest and lowest number of the evaluation in the test cases
for a path. Table II provides the summary statistics of 7
factors from the above metrics due to the limitation of page
space.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we examine the association between
the number of evaluations and above metrics by using
correlation analysis to identify which factors affect test
generation efficiency for EFSM models. Then, we inspect
the potential relationship between test generation efficiency
and the factors by applying linear or nonlinear regression
analysis. Finally, the empirical results are discussed.

A. Key factors identification

As displayed in Table II, the largest set is provided with
1917 data, and the smallest is 619 for all 8 EFSM
models. The number of data points in each dataset is limited,
and Spearman’s correlation coefficient that is more robust
to atypical values and to non-linearity of the underlying
relationship, is used in this study.

In Section III-C, we introduced 15 metrics that are poten-
tial factors on affecting performance of test data generation.
However, correlation may exist between these factors. For
example, a longer path that contains more transitions may
have more variables in the events and more conditions.
Therefore, we first investigate the correlation between these
metrics. Table III and Table IV list the correlation coef-
ficients quantifying the strength of the interaction among
the metrics in ATM and CruiseControl model, respectively.
All of the values, except ones with + (meaning there is no
significant correlation at 0.01 and 0.05 significant level) and
- (denoting the correlation coefficients cannot be computed
because NNEOC is constant) in Table III, are significant at
α = 0.01 level (2-tailed). The bold-face values representing
the correlation coefficient are very high (larger than 0.9),
implying there is a strong relationship between factors.
In Table IV, most of metrics in CruiseControl model are
significantly correlated with each other indicated by many
bold-face values. However, this is not shown in Table III of
ATM model, where there are few bold-face values.

To investigate the relationship between the number of
evaluations (NE) during test generation and these metrics,
the correlation coefficients are computed and shown in the
last line in Table III and Table IV. It can be seen that there
are strong correlations between NE and various metrics ex-
cept NVDA, NVDAUC, NEOC and NNEOC (about 0.668)
in CruiseControl. Again this is not shown in ATM where
the coefficient is about 0.363, because of the low correlation
between the metrics.

Matthew et al. [14] suggested that the length of path
affects the performance of test generation for Ada software
system. It is interesting to investigate whether does it exist
in EFSMs. Figure 3 and Figure 4 display the relationships
between NE and LP on 4 models with EXIT state and
4 models without EXIT state, respectively. It is observed,
from Figure 3.(d) and Figure 4.(d), that there are no distinct
relationships between NE and LP for FuelPump_exit/noexit
models. However, Figure 3.(c) and Figure 4.(c) indicate
that NE increases approximately exponentially with LP
for CruiseControl_exit/noexit models. It can also be ob-
erved that NE increases in fluctuation as LP enlarges
for other models. To obtain a more detailed insight, we
analyze carefully the experimental results, and find that
the number of numerical equal operators in the conditions
(NNEOC) of FuelPump_exit/noexit, ATM_exit, Cashier_exit, ATM_noexit, Cashier_noexit, CruiseControl_exit
and CruiseControl_exit model, is 0, 0, 1, 1, 2, 2, 3 and 4,
respectively. The corresponding correlation coefficients between
NE and LP are -0.013+, 0.030+, 0.404, 0.323, 0.580, 0.676,
0.890 and 0.951, respectively. It suggests that the correlation
between the NE and LP rises along with the growing of
NNEOC.

On the other hand, when NNEOC is considerable, such as
CruiseControl_exit/noexit model, there is a close relationship
between NE and LP, and most of the other metrics strongly
associated with LP (see the second column of Table IV).
It seems to indicate that NNEOC has an important effect
on EV. However, this is not explored by the analysis of
the correlation between NE and NNEOC. Further inspection
of the data reveals that NNEOC may be too small and
insignificant to demonstrate the relationship. As shown in
Figure 5, NE increases along with the growth of NNEOC
on all models whose NNEOC is larger than 1. In addition,
we especially create some complete paths on ATM_noexit
model to improve NNEOC by paying special attention to
transition T4 where a numerical equal operator is required.
The largest NNEOC is equal 8 in the paths, and the corre-
lation coefficients between NE and all metrics are displayed
in Table V. It is obvious that NNEOC is significantly correlated
with EV, holding the coefficient 0.887 close to the largest
0.899.

<table>
<thead>
<tr>
<th>Metric</th>
<th>LP</th>
<th>NV</th>
<th>NVDE</th>
<th>NVDA</th>
<th>NVUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE</td>
<td>0.341</td>
<td>0.890</td>
<td>0.399</td>
<td>0.311</td>
<td>0.735</td>
</tr>
<tr>
<td>Metric</td>
<td>NVU</td>
<td>NVDEUC</td>
<td>NVDAUC</td>
<td>NC</td>
<td>NSC</td>
</tr>
<tr>
<td>NE</td>
<td>0.311</td>
<td>0.838</td>
<td>0.276</td>
<td>0.821</td>
<td>0.746</td>
</tr>
<tr>
<td>Metric</td>
<td>NEOC</td>
<td>NNEOC</td>
<td>LPEV</td>
<td>NNV</td>
<td>NNEV</td>
</tr>
<tr>
<td>NE</td>
<td>0.840</td>
<td>0.887</td>
<td>0.830</td>
<td>0.990</td>
<td>0.881</td>
</tr>
</tbody>
</table>

How about the number of equal operators in the condi-
tions? Does it affects the efficiency of test generation for
EFSM models? To address these questions, we analyze the
NEOC on all EFSM models, and find there is little con-
nection between NE and NEOC. This is easy to understand
since logical equal operators are included in NEOC, which requires little effort to meet. Consequently, a conclusion can be drawn from the above analyses that NNEOC plays a key effect on the number of evaluations during test generation. That is to say, NNEOC of a path affects the performance of test data generation decisively.

### B. Regression models analysis

In Section IV-A we concluded that the number of the evaluations has a close correlation with the number of numerical equal operators in the conditions on feasible
tern or a curved pattern? How about the metrics, length of path and number of variables in events? Are there similar relationships with the number of evaluation in EFSM models as in Ada software systems? To address these questions, we apply a regression analysis to the data of all the 8 subjects. The regression analysis on the correlation between EV and NNEOC is illustrated in Figure 6 on all models that NNEOC is larger than 1. It can be seen that the exponential coefficient of determination R-Square improves from 0.330 to 0.751 when NNEOC increases from 2 to 8. That is to say, the exponential regression model works better when NNEOC is considerable.

However, what is this relationship? Is there a constant upward or downward trend that follows a straight-line pattern or a curved pattern? How about the metrics, length of path and number of variables in events? Are there similar relationships with the number of evaluation in EFSM models as in Ada software systems? To address these questions, we apply a regression analysis to the data of all the 8 subjects. The regression analysis on the correlation between EV and NNEOC is illustrated in Figure 6 on all models that NNEOC is larger than 1. It can be seen that the exponential coefficient of determination R-Square improves from 0.330 to 0.751 when NNEOC increases from 2 to 8. That is to say, the exponential regression model works better when NNEOC is considerable.

Consider a test data required to traverse a FTP, the number of transitions with event variables could be more reasonable than the number of transitions in the efficiency analysis of test generation. For example, the comparison LP with LPEV with respect to NE is given in Figure 7(a) for ATM model. It is evident that LPEV is more advisable. On the other hand, it is easy to understand that the number of numerical event variables is more reasonable than the number of variables defined in events since NVDE includes logical variables which are effortless to meet the test requirements. The advantage is clear in Figure 7(b), which displays the comparison NVDE with NNEV with respect to NE for CruiseControl_noexit model. So, we use LPEV and NNEV, instead of LP and NEV in following analysis.

In order to answer the question what relationship exists between EV and LPEV as well as NNEV, we apply regression analysis to all the models. The results are shown in Figure 8, Figure 9 and Figure 10, respectively, according to their NNEOC values. The scatterplot graphs exhibit a gradually exponential relationship between EV and LPEV (see the left hand side of the graphs) as well as NNEV (see the right hand side) when NNEOC varies from 0 to 4 depending
on models. The detailed coefficients of determination R-Square about exponential, increase from 0.053 to 0.855, and from 0.049 to 0.851 with respect to LPEV and NNEV, respectively. At the same time, we compute their R-Square by linear regression on corresponding models. It is further observed that the R-Square of linear is obviously smaller than that of the exponential on each corresponding model except FuelPump_noexit models, which NNEOC=0, shown in Figure 8(a), especially, for the CruiseControl model (see the Figure 10(b)), their R-Square of exponential are 0.855 and 0.851, whereas R-Square of linear are 0.382 and 0.383, with respect to LPEV and NNEV, respectively. That is to say, the exponential regression models work well. As a result, we can draw the conclusion that there exists strong exponential relationships between EV and LPEV as well as NNEV when NNEOC is sufficient.

Figure 7. LP VS LPEV and NVDE VS NNVE

Figure 8. Regression analysis on models with NNEOC=0,1

Figure 9. Regression analysis on models with NNEOC=2

Figure 10. Regression analysis on models with NNEOC=3,4

It is distinctly different from Matthew’s finding that there was an approximate linear relationship between NE with NV generated in Ada software systems. In order to further support our proposition that NNEV has an exponential effect on NE, we plot the relationship between NNEV and LPEV. As demonstrated in Fig 11, there is a very strong linear relationship between NNEV and LPEV on all models. Their coefficient of determination R-Square are about 0.929 (92.9%), which is substantial, except Cashier_noexit model (about 0.792). Therefore, NNEV and LPEV should maintain similar relationship with EV. That is, there exists an exponential connection between NNEV and NE.

V. RELATED WORK

Many test generation approaches had been studied in the literature for systems modelled as EFSMs [4], [5], [7], [8],
[10], [18], [23]. Most of them focus on the generation of test sequences. Juhan et al. [10] presented a way to generate test sequences from EFSM models using a guided model checker. Duale et al. [8] introduced a method that overcomes the feasibility problem in advance. The method converted a class of EFSMs into consistent EFSMs in which all transition paths were feasible. Derderian et al. [5] proposed a GA approach to generate feasible paths from EFSM model. The approach evaluates the feasibility of a given transition path according to the number and the types of guards found in that TP. Kalaji et al. [18] reported a fitness metric to estimate the path’s feasibility and to direct the search towards paths that are relatively easy to trigger. However, the studies do not tackle the problem of generating input test data to be used in testing the generated paths.

A number of comparisons and evaluations of testing efficiency have been carried out by researchers [2], [11], [12], [17], [25]. Ntafos [25] compared branch coverage, efficiency have been carried out by researchers [2], [11], the studies do not tackle the problem of generating input to estimate the path's feasibility and to direct the search found in that TP. Kalaji et al. [18] reported a fitness metric to estimate the path’s feasibility and to direct the search towards paths that are relatively easy to trigger. However, the studies do not tackle the problem of generating input test data to be used in testing the generated paths.

VI. CONCLUSION AND FUTURE WORK

Testing from formal specifications offers a simpler and more rigorous approach to developing test data. However, current work in specification-based automatic testing has been limited largely to FSM models in which context variables do not exist. It is hard to find a set of test data to trigger a given feasible path in EFSM models if more variables are used in the path and conditions are complex. Therefore, in this paper, we have presented a GA-based system to automatically generate test data for feasible transition paths in EFSM models.

In order to investigate the effectiveness of our test generation approach and identify the key factors affecting the efficiency of test generation in EFSM models, an empirical study has been conducted on 8 EFSM models, and the results were analyzed in detail using statistical analysis. We conclude that NNEOC plays a very important role in test generation efficiency. It is only when NNEOC is considerable that NE grows exponentially along with the increasing of NNE or LPEV. Moreover, there is a very strong linear relationship between NNEV and LPEV. The results provide a significant guide to predict the testing efficiency based on NNVE or LPEV on a FTP for EFSM models.

Future research work on this topic will concentrate on the problem of test data generation in EFSM models including string, compound data types as well as function calls etc. The questions such as what relationship exists between the test generation cost and length of string, the number of string variables, as well as the number of function calls will be explored carefully.

ACKNOWLEDGEMENT

The work described in this paper was supported by Beijing Natural Science Foundation under Grant No.4072021 and National Natural Science Foundation of China under Grant No.60903002. The authors wish to thank Lorna Anderson for help with proof reading.

REFERENCES


