A New Approach to Evaluate Path Feasibility and Coverage Ratio of EFSM Based on Multi-objective Optimization*

Rui Yang[1,2], Zhenyu Chen[1], Baowen Xu[1,2+] and Zhiyi Zhang[1]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China
[2] Computer Science and Technology, Nanjing University, Nanjing 210046, China
[+] Corresponding author: bwxu@nju.edu.cn

***Abstract*—Extended Finite State Machine (EFSM) is a popular model for software testing. Automated test generation on EFSM models is still far from mature. One of the challenges is test case generation for a given path and the path may be infeasible. This paper proposes a novel approach to solve the path ordering problem by the Multi-objective Pareto optimization technique. Two fitness functions are designed to obtain the Pareto-optimal solutions of path sequence, such that test cases could be generated more effectively. The optimization process is to find the path set, with trade-off path length, coverage criterion and path feasibility. An experiment was designed with four popular EFSM models. The experimental results show that our approach is more effective by comparing it to previous techniques.**

Keywords- path feasibility evaluation, Multi-objective optimization, test case generation, EFSM model-based testing

I. INTRODUCTION

Automated testing usually derives the test case via creating a model of the software which represents the real system and employs the model to generate test case that can be applied to the system implementation. The generated test cases are utilized to verify whether the implementation meets the specification.

In model-driven testing, Finite State Machine (FSM) and Extended Finite State Machine (EFSM) are widely used for the purpose of generating test case. There are several issues that limit the application of FSM-based testing. The main issue is that FSM can only represent the control part of a system. However, Many complex systems include not only control parts but also data parts. Extended Finite State Machine (EFSM),which consists of states, variables and transitions among states can express both control flow and data flow of system, can be considered as an enhanced model of FSM [1]. There have been lots of research works on FSM-based test sequence generation, whereas test case generation for the EFSM model is still a challenge .

In general, the steps of automated test case generation from an EFSM model including: (1)Generate state identification sequence. (2)Generate test paths for specified coverage criterion. (3)Generate test data to trigger the test paths. (4)Create test oracle.

The difficulty of automated testing on EFSM models since the fact that the EFSM models may contain infeasible paths due to the existence of conflict between predicates and assignment statements in transitions. Moreover, the detection of an infeasible path is generally undecidable [2].

In terms of state identification sequence generation, many methods have already been proposed in the literature [3][4][5]. There also exist a few studies address the problem of infeasible path even test data generation of EFSM models[6][7][8][9][10]. However, automated EFSM-based testing is still far from mature. In our previous work[11], an approach that combines static analysis and dynamic analysis techniques is presented to address the problem of path infeasibility in the test case generation process on EFSM model. A metric is presented in order to find a path set that has fewer paths, longer path length and goodness feasibility to meet specified coverage criterion since previous study shown that the test suite with small number of longer test cases improves fault-detection efficiency compared to shorter ones[12]. This metric is designed carefully to trade-off path length and feasible evaluation value, since a longer path has higher probability of infeasible due to the fact that a longer path may contains more conflict conditions among transitions. However, the formula is still not precise enough.

Therefore, this paper presents a new approach that overcomes aforementioned problems by Multi-objective Pareto optimization technique to explore the relationship between path length and path feasibility. Multi-objective optimization is appropriate for solving problems when the solutions need to meet several conflicting objectives. By using this technique, we transform the aforementioned problem into a bi-objective optimization problem. The optimization process is given to find a path set to achieve the trade-off between path length and path feasibility to meet the specified test criterion(longer paths may contain more transitions to achieve specified coverage criterion). In addition, we present detailed experimental study to evaluate the efficiency by applying our approach to four popular EFSM models.

The main contributions of this article are embodied in the following three aspects:

1. A new Multi-objective approach is presented to find a optimized path set with small number of longer feasible paths to generate test data and meet the test criterion.
2. We present two fitness functions of an individual to solve the path ordering problem of EFSM models. The idea also can be extended to other path ordering problems in path-oriented testing.
3. An experiment is designed and we apply the proposed approach to four popular EFSM models. The experimental results confirm that the approach is more effective by comparing it to our previous metric.

The rest of this paper is organized as follows: Section II introduces the basic concept of an EFSM model; Section III

*The work described in this article was partially supported by the National Natural Science Foundation of China (90818027, 61003024, 61170067).

introduces our previous research of test case generation process; The experiment is described in Section IV; Section V reviews related work; Section VI concludes this paper and provides some possible opportunities for future research.

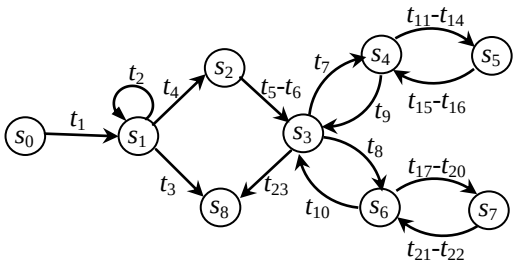

Figure 1. EFSM model of Automated Teller Machine [15]

II. PRELIMINARIES

An EFSM can be formalized as a 6-tuple $M=(S, s_0, V, I, O, T)$, where S represents a finite set of state; $s_0 \in S$ represents the initial state of the EFSM; V represents a finite set of context variables; T represents finite set of transition; I represents a set of inputs of transition and O represents a set of outputs. Each transition $t \in T$ is also a 6-tuple $T= (s_i, s_j, P_t, A_t, i_t, o_t)$, where s_i represents the start state of transition t; s_j represents the end state of transition t; P_t represents the predicate operation on context variables(known as Guard) and A_t represents the operation on current variables(known as Action); $i_t \in I$ represents an input parameter and $o_t \in O$ represents output results.

At first, the EFSM model is at an initial state s_0 with vector x_0 of initial variable values. After some transitions occurred, the EFSM move to the state s_i with the current vector x_i of variable values. If EFSM model receiving input event or input parameter i_t, and x_i is valid for predicate operation P_t, then the transition t will be triggered and state of EFSM moves to s_j. A self-loop transition is a transition that has the same start state and the end state (see **Figure 1**). When a transition is triggered, action A_t changes the current variable values, and output events or output results may also be produced. Some transitions are difficult to be satisfied since it may have complex conditions, while some transitions may have no predicate conditions. A transition path is infeasible if associated predicate conditions of the transition never be satisfied. The existence of infeasible paths creates difficulties for automated test case generation for EFSM. In addition, some models are free of exit state, while others contain both initial state and exit state.

If any group of transition has the same input that transforms a state, and the guards of more than one transition cannot be satisfied at the same time in this transition group, then the EFSM is deterministic[13]. Otherwise the EFSM model is non-deterministic. In this paper, we assume that the initial state of EFSM is always reachable from any state through Reset operation, and only deterministic EFSM model is considered.

III. IMPLEMENTATION

3.1 Test Case Generation Process of Previous Research

For the sake of clarity, we first describe our previous work[11][29], the process of our approach to generate test cases(including feasible paths, test data and oracle information) from an EFSM can be split into following steps:

Step1: Candidate Path Set Generation Algorithm

Firstly, a candidate path set is generated from the initial state to other states on an EFSM model in order to satisfy the test adequacy criterion. Then, some paths which have highest likelihood of feasibility are selected to generate test data. The transition paths in candidate path set contain loops (including self-loops). The generated paths would infinite if the infinite loops are contained in transition paths. Therefore, a constraint condition that just contains loops or self-loops only one time is imported to generate paths by means of candidate path set generation algorithm named *Path-Gen*(detail described in [11]). The paths that contain a certain number of loops (self-loops) can be also generated conveniently by expanding aforementioned algorithm.

Step2: Feasibility Evaluation Strategy

The existence of infeasible paths on EFSM model, which is due to the variable interdependencies among the actions and conditions, has become a challenging problem of path-oriented test case generation. In order to evaluate the feasibility of paths that generated in the *Step1*, a metric is presented through static analysis to evaluate the path feasibility of the EFSM and achieve all-transitions coverage criterion. Afterwards, paths in the candidate set are sorted by the evaluation value to assist test data generation in *Step3*. However, in some cases, static analysis cannot detect the infeasible path thoroughly. Therefore, we present a dynamic analysis method in order to detect infeasible paths in the test data generation process. In order to find a path subset with the property that has fewer paths, longer path length and goodness feasibility to achieve all-transitions criterion, the metric is presented as follows:

$$f = \begin{cases} \dfrac{\sum_{i=0}^{k} v(df_i)}{|TP|^d} & If \quad \sum_{i=0}^{k} v(df_i) \neq 0 \\ 0 - |TP| & If \quad \sum_{i=0}^{k} v(df_i) = 0 \end{cases} \qquad (1)$$

where k represents the number of definition-p-use transition pairs in a path; df_i represents the definition-p-use transition pair in a path and $v(df_i)$ represents its corresponding penalty value; $|TP|$ represents the path length, and d is a value used to tune the weight of path length in the metric(detail described in [11]). Denominator $|TP|$ is the trade-off between path length and penalty value since a longer path may contains more definition-p-use transition pairs, which has higher probability that it can result in a larger penalty value of a transition path. However, if

$\sum_{i=0}^{k} v(df_i) = 0$, it means that there is no definition-p-use interdependence among transitions, and this path has best feasible probability. In this case, the penalty value is assigned $0-|TP|$ due to a longer path should get a lower penalty value to cover more transitions. Thereafter, the transition path set is sorted by f in ascending order. However, the final feasible path subset which attempt to achieve all-transitions criterion or other coverage criteria will be generated in the test data generation process dynamically.

Step3: Test Data Generation Process

In this step, a dynamic analysis method, which is based on meta-heuristic search algorithm, is proposed to detect infeasible paths in the test data generation process. In general, a common strategy of dynamic analysis is to limit the depth or iterations of search. We build an executable model by expressions semantic analysis technique, therefore the dynamic run-time feedback information can be collected as a fitness function and scatter search algorithm is introduced to guide the test data generation process directly. In addition, the corresponding expected outputs of the generated test data are also calculated to construct the oracle information automatically. Finally, feasible path subset, test data and oracle information are combined into complete test cases.

3.2 Multi-objective Optimization Approach

The formula 1 is designed carefully to trade-off path length and feasible evaluation value, since a longer path has higher probability that it contains more conflict Guards and Actions which can result in a path infeasible. However, the formula 1 is still not precise enough. Therefore, in this study, the multi-objective optimization approach is presented to find a path set which achieves the trade-off between path length and path feasibility to meet the specified coverage criterion(longer paths may contain more transitions to achieve specified coverage criterion). In brief, we present a Multi-objective Optimization approach to improve the path feasibility evaluation strategy.

3.2.1 Solution Encoding and Fitness Functions Design

The important issues of multi-objective optimization problem are the encoding of the solution and the design of the fitness function. Before a Multi-objective optimization algorithm can be applied to solve the problem, we need to encode potential solutions to that problem in a certain form which can be processed. In addition, the fitness function has an important influence on finding the final solution to meet the requirements. The fitness value of a solution is evaluated by the fitness function. A solution with better fitness value is selected for next iteration in the Pareto evolution process. In context of our approach, the fitness values is the measurement that whether a path set with certain path ordering can facilitate test data and oracle information generation.

In this study, we use the permutation encoding method, therefore, the paths in a set are encoded as a sequence of integer. In permutation encoding, the individuals in the population are the string of numbers. Thus, for a given transition path set, every transition path is encoded as a different number(named path ID). Please note that, when using permutation encoding, the evolution process cannot generate any new number in the individual but change its ordering. In addition, the property of a transition path, such as path length and count of distinct transitions, can be obtain by its ID. The individual structure is described in **Figure 2**.

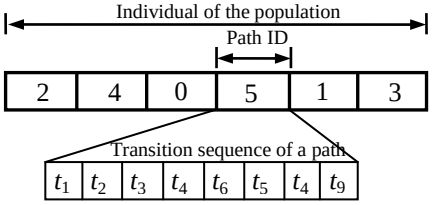

Figure 2. Individual structure of a population

For ordering the path set that generated by candidate path set generation algorithm to obtain Pareto front, we designed two fitness functions f_1 and f_2, where f_1 is utilized to find the path ordering to obtain a path subset with small number of longer paths and higher coverage ratio, and f_2 is utilized to find the path ordering to obtain a feasible path subset more efficiently in dynamic process(Step 3 in section 3.1). the fitness functions are designed as follows:

$$f_1 = \sum_{j=0}^{n-1} \frac{C_1}{(\frac{C_2}{\gamma} + 1) \times (n - j + 1)} \qquad (2)$$

$$f_2 = \sum_{j=0}^{n-1} \frac{C_1}{(\sum_{i=0}^{k} v(df_i) + 1) \times (n - j + 1)} \qquad (3)$$

where n is the count of the paths, j represents the position of a path in path set, γ is the count of the distinct transitions in a path, $\sum_{i=0}^{k} v(df_i)$ is same as formula 1. Number C_1 and C_2 as numerator are constant value since smaller fitness value means better solution in selection process. In our study, C_1 and C_2 are assigned 1000 and 100, respectively. In general, according to the fitness function f_1, the longer paths with more distinct transitions will be arranged in the front of path set, this kind of path ordering has better fitness. In terms of f_2, the paths with better feasible probability will be arranged in the front of path set, this kind of path ordering has better fitness.

3.2.2 Multi-objective GA Algorithm

For implementing our method, the non-dominated sorting genetic algorithm II (NSGA-II) presented by Deb et al. [14] is utilized. NSGA-II uses an elitist approach and a crowed comparison mechanism, improving both quality and diversity of Pareto solutions. Hence, in our case, NSGA-II is used to solve the problem of multi-objective optimization.

In every generation, NSGA-II utilizes crossover and mutation to create a new population. The next generation is given by a Pareto-optimal selection from both the new offspring and their parents. Therefore, NSGA-II is elitist. In this study, NSGA-II was given two fitness function:f_1 and f_2.

At the beginning, a population P_0 is created by means of stochastic method. The population is sorted based on the

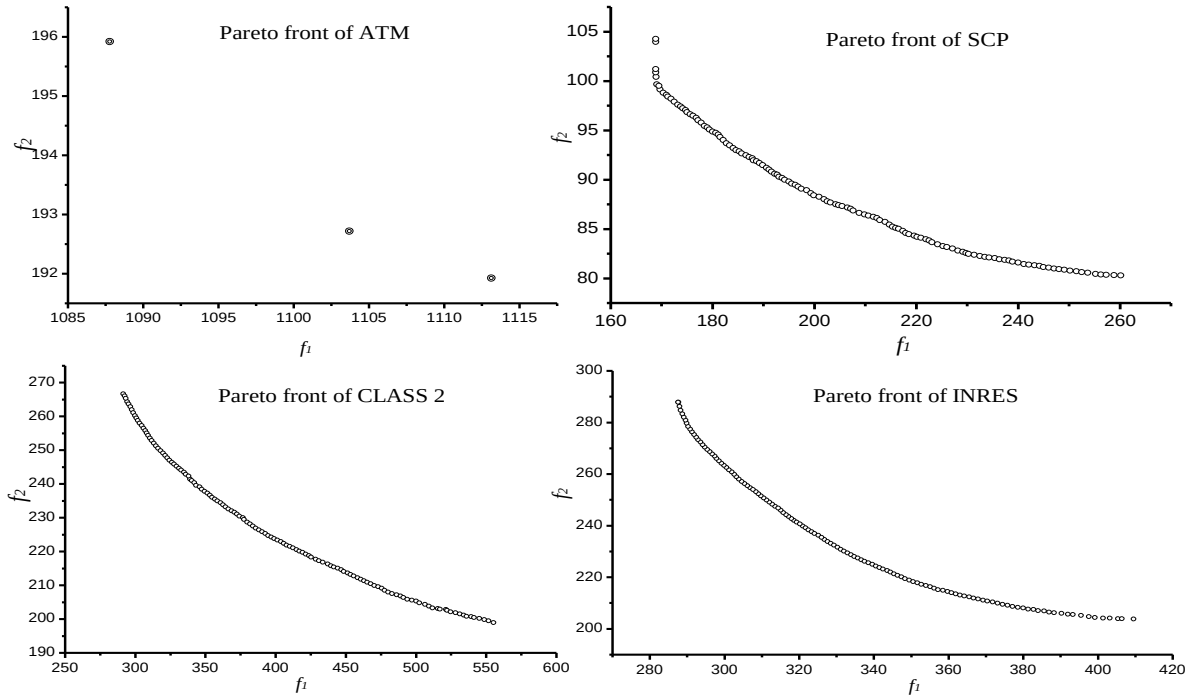

Figure 3. The front of Pareto–optimal solutions of four EFSM models

non-domination. To fit the permutation encoding method, we adopt the BinaryTournament selection, TwoPointsCrossover and SwapMutation operators to create a offspring population Q_0 (size is N).

After that, the algorithm enters into iteration process: first the population $R_t=P_t \cup Q_t$ (R_t size is $2N$) is formed by combination method. Afterwards, the population of R_t is sorted according to non-domination, and the ranks are assigned to each Pareto front with fitness F_i. Then, from Pareto front of fitness F_1, each Pareto front is added to new population P_{t+1} until it reaches the size N. The new population P_{t+1} will be used for selection, crossover and mutation to create the new population Q_{t+1}. The above process is repeated until all individuals are assigned to a pare to front.

IV. EXPERIMENTAL STUDY

In this section, in order to determine whether test case generation with specified coverage criterion for an EFSM can benefit from proposed multi-objective optimization technique, we present a detailed experimental study. More specifically, the study is designed to get the final Pareto fronts of the transition path ordering of the EFSM models. Thereafter, Pareto-optimal solution is used to test data generation process to find whether this approach is more effective than our previous metric.

Four popular EFSM models are utilized for our experimental study: Class 2 transport protocol[3], Automated Teller Machine (ATM)[15], INRES protocol[16], and SCP protocol[17]. Class 2, INRES and SCP model are free of exit states while ATM contains both a start and exit state. For generality, in our experiment, we do not considered that the path must end in an exit state in the model with exit state.

4.1 Experimental Setup

In order to achieve the experiment fairly, the experiment environment and the main configurations of test data generation algorithm are same as our previous work [11].

The main parameters configuration of NSGA-II algorithm we set in this study are listed as follows:

1. population size =128;
2. Max Evaluations times=20000;
3. Crossover Operator= TwoPointsCrossover;
4. Crossover probability=0.9;
5. Mutation Operator= SwapMutation;
6. Mutation probability=0.2;

Since permutation encoding method is utilized in this study to solve path ordering problem. The individuals in the population are the string of numbers that represent a path ID in a sequence. When crossover and mutation operation are used to create the new population, the operation cannot generate any new path ID in the individual but change its ordering. That is to say, the elements of a individual remain unchanged, crossover and mutation operation only change the ordering of the elements of an individual. Therefore, special crossover and mutation operator are needed. In this study, TwoPointsCrossover operator and SwapMutation operator are adopted to achieve permutation encoding method. Crossover probability is 0.9 where Mutation probability is 0.2. The population size for evolution is 128. While generation count reached Max iteration times, NSGA-II algorithm stopped.

TABLE I. TEST GENERATION RESULTS WITH DIFFERENT TECHNIQUES

Model & Method	Subject Model	Executed Path Number	FTP Number	Iteration Times	Execute Time(sec.)
Feasibility Evaluation Using Formula 1	ATM	17	16	1600	4
	CLASS 2	30	16	2016	57
	INRES	10	10	7650	2
	SCP	10	2	816	13
Multi-objective Optimization	ATM	15	14	1400	4
	CLASS 2	26	12	3456	58
	INRES	11	9	20936	9
	SCP	10	2	1126	14

4.1 Experimental Results

In this section, the experimental results of the path ordering problem on four EFSM models are presented. **Figure 3** shows the front of Pareto–optimal solutions of four EFSM models. In the figure, Y-axis represents the fitness value of the path sequence that calculated by fitness function f_2. The number of X-axis represents the fitness value of the path sequence that calculated by fitness function f_1. That is to say, a dot in the coordinate plane represents a kind of path sequence that has fitness value f_1 and f_2. The count of dot less than or equals to 128 since population size equals 128. The special case in the figure is that the dot of ATM model relative few as a result of this model has relative simple transition correlation, also the Pareto-optimal solutions is relative few.

In order to evaluate the efficiency of our approach, we need to choose a solution in Pareto–optimal front(an ordering path set), this path set is used to generate test data, meanwhile, a subset of feasible path which achieves all-transitions coverage strategies will be generated dynamically in this process. In the experiment, the solution with minimum value of f_1+f_2 is selected to generate test data.

Since the execution time of the test data generation of the same model exists tiny difference, the program is executed 30 times for every model and calculate its average execution time for the purpose of illustrating the experimental results precisely. In order to illustrate the advantage of this approach, the results of test data generation are used to compare with previous results in the[11]. The comparison results are listed in **Table I**, where Executed Path Number is the count of transition path that be chosen to generate test data. FTP Number is the feasible transition path number that are selected for test data generation to achieve all-transitions coverage. Iteration Times is the sum of the Iteration Times of every feasible path. It shows that the final feasible path number(FTP Number) reduced obviously compared with previous technique.

In the mass, the count of transition path that be chosen to generate test data(executed path number) also decreases. More infeasible paths are avoided in the process of test data generation(the value of Executed Path Number minus FTP Number of a model means the count of infeasible path that encountered in the process of test data generation). In the other hand, Iteration Times of test data generation increased in some degree, Execute Time is the same as Iteration Times. The reason is that the feasible transition paths become longer and more difficult to generate test data to traverse these paths. Through observing, we found that the longer path contains more correlation among transitions, and test data generation process becomes more difficult, even the path becomes infeasible. However, we think that the quality of test paths and test data is more important than iteration times. Hence, this approach is more effective by comparing it to previous metric.

V. RELATED WORK

In terms of EFSM-based testing, some test sequence generation methods of EFSM models have been proposed in [18][19]. In these methods, test sequence generation with data flow criteria is considered, and control flow testing is ignored or considered separately. Duale et al.[7][21] transformed an EFSM to one that has no infeasible paths, the approach converted a EFSM into consistent EFSM, and they utilized simplex algorithm to solve the infeasible problem. However, this method only can be utilized for EFSMs in which all operations and guards are linear. In addition, there are some other methods used FSM-based techniques to test an EFSM model[16][20], however, these methods may suffer the state explosion problem. In order to generate feasible test paths from EFSM models, Derderian et al.[8] gave a fitness function to estimate how easy it is to trigger a path. Kalaji et al.[9] proposed a GA-based approach to generate feasible transition paths that are easy to trigger. However, the path length should be determined in advance and all paths have same length. In terms of test data generation on EFSM, Lefticaru et al.[22] first introduced the genetic algorithm to derive test data from a FSM. The design of fitness function is based on literature [23]. Yano et al.[24] presented a multi-objective evolutionary approach for test sequence generation from an EFSM. However, the generated test data may have redundancy. Kalaji et al.[25] used a genetic algorithm whose fitness function is based on a combination of a branch distance function[23] and approach level [26] to generate test data for specified path. J. Zhang et al.[27] proposed a method to obtain a deterministic EFSM from a program written in a subset of C program, and they attempted to find test data by means of the symbolic execution technique. Ngo et al.[28] proposed a heuristics-based approach for code-based testing to detect infeasible path for dynamic test data generation by the observation of some common properties of program paths. J. Zhang et al.[29]presented the detail of test data generation from EFSM by using scatter search and execute information feedback technique. The method showed good effectiveness by the experiments of comparing with the random algorithm.

VI. CONCLUSIONS AND FUTURE WORK

This paper presents a new approach that utilizes Multi-objective Pareto optimization technique to explore the relationship between path length and path feasibility on EFSM models. Two fitness functions of an individual are designed to solve the path ordering problem. The optimization process is given to find a path set which achieves the trade-off between path length and path feasibility to meet the specified test criterion. This idea also can be extended to other path ordering problem in path-oriented testing. An experiment was designed with four popular EFSM models. The experimental results show that the approach is more effective by comparing it to previous metric.

In our future study, we intend to improve design of the fitness function to get better quality of test case by further exploring the property in the transition path. The tuning of the parameters and operators of multi-objective optimization algorithm to get higher efficiency also will be implemented.

VII. ACKNOWLEDGMENTS

The work described in this article was partially supported by the National Natural Science Foundation of China (90818027, 61003024, 61170067). The authors would like to thank anonymous reviewers for their valuable comments.

REFERENCE

[1] A. Petrenko, S. Boroday and R. Groz, "Confirming configurations in EFSM," FORTE, pp.5-24, 1999.

[2] D. Hedley and M. A. Hennell, "The Causes and Effects of Infeasible Paths in Computer Programs," ICSE, pp. 259-266, 1985.

[3] T. Ramalingom, K. Thulasiraman, and A. Das, "Context independent unique state identification sequences for testing communication protocols modelled as extended finite state machines," Computer Communications, Vol. 26(14), pp. 1622-1633, Sep 2003.

[4]C.M. Huang, M.Chiang, and M.Y.Jang. "UIO_E: A protocol test sequence generation method using the transition executability analysis (TEA), " Computer Communications, vol.21(16), pp.1462-1475, 1998.

[5] A. Petrenko, S. Boroday and R. Groz, "Confirming configurations in EFSM testing," IEEE Transactions on Software Engineering, Vol.30(1), pp.29-42, 2004.

[6] S. T. Chanson and J. Zhu, "A unified approach to protocol test sequence generation," In Proceedings of the International Conference on Computer Communications , IEEE INFOCOM , pp. 106-114, 1993.

[7] A. Y. Duale and M. Ü. Uyar, "A Method Enabling Feasible Conformance Test Sequence Generation for EFSM Models," IEEE Transactions on Computers, Vol. 53(5), pp. 614-627, 2004.

[8] K. Derderian, R. M. Hierons, M. Harman, and Q. Guo, "Estimating the feasibility of transition paths in extended finite state machines," Automated Software Engineering, Vol. 17(1), pp. 33-56, Nov 2009.

[9]A. S. Kalaji, R. M. Hierons, and S. Swift, "Generating Feasible Transition Paths for Testing from an Extended Finite State Machine (EFSM)," ICST, pp. 230-239, Apr 2009.

[10]T. Yano, E.Martins, and F.L.Sousa. "Generating Feasible Test Paths from an Executable Model Using a Multi -objective Approach," ICST Workshops, pp.236-239, 2010.

[11]R. Yang, Z. Chen, B. Xu, W. Eric Wong, and J. Zhang "Improve the Effectiveness of Test Case Generation on EFSM via Automatic Path Feasibility Analysis," IEEE HASE, pp.17-24, 2011.

[12]G. Fraser and F. Wotawa. "Redundancy Based Test-Suite Reduction," Lecture Notes in Computer Science, 4422, pp. 291-305, 2007.

[13] C. Shih, J. Huang, and J. Jou, "Stimulus generation for interface protocol verification using the non-deterministic extended finite state machine model," IEEE HLDVT, pp. 87-93, 2005.

[14]K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, Vol. 6(2), pp. 182–197, 2002.

[15]B. Korel, I. Singh, L. Tahat, and B. Vaysburg, "Slicing of state-based models," ICSM, pp. 34-43, 2003.

[16]C. M. Huang, M. Y. Jang, and Y. C. Lin, "Executable EFSM-based data flow and control flow protocol test sequence generation using reachability analysis," Journal of the Chinese Institute of Engineers, Vol. 22(5), pp. 593-615, Jul 1999.

[17]A. Cavalli, C. Gervy, and S. Prokopenko, "New approaches for passive testing using an Extended Finite State Machine specification," Information and Software Technology, Vol. 45(12), pp. 837-852, Sep 2003.

[18]H. Ural and B. Yang, "A Test Sequence Selection Method for Protocol Testing," IEEE Transactions on Communication,Vol.39(4), pp.514-523, 1991.

[19]R. Miller and S. Paul, "Generating Conformance Test Sequences for Combined Control and Data Flow of Communication Protocols," In Protocol Specifications, Testing and Verification, pp.13-27, 1992.

[20]R.M.Hierons, T.H.Kim, and H.Ural, "Expanding an extended finite state machine to aid testability," COMPSAC, pp.334-339, 2002.

[21]M. Ü. Uyar and A. Y. Duale, "Test generation for EFSM models of complex army protocols with inconsistencies," 21st Century Military Communications. Architectures and Technologies for Information Superiority, Vol 0(C), pp. 340-346, 2000.

[22]R. Lefticaru and F. Ipate, "Automatic State-Based Test Generation Using Genetic Algorithms," SYNASC 2007, pp. 188-195, Sep 2007.

[23]N. Tracey, J. Clark, K. Mander, and J. McDermid, "An automated framework for structural test-data generation," ASE, pp. 285-288, 1998.

[24]T. Yano, E.Martins, and F.L.Sousa, "MOST: A Multi-objective Search-Based Testing from EFSM, " ICST Workshops, pp.164-173, 2011.

[25]A. S. Kalaji, R. M. Hierons, and S. Swift, "An integrated search-based approach for automatic testing from extended finite state machine (EFSM) models," Information and Software Technology, Vol.53, pp.1297 - 1318, Dec 2011.

[26]P. McMinn and M. Holcombe, "Evolutionary testing of state-based programs," GECCO, pp.1013-1020, 2005.

[27] J. Zhang, C. Xu and X. Wang, "Path-Oriented Test Data Generation Using Symbolic Execution and Constraint Solving Techniques," SEFM, pp.242-250, 2004.

[28]M. Ngo, and H. Tan, "Heuristics-based infeasible path detection for dynamic test data generation," Information and Software Technology, Vol. 50(7-8), pp. 641-655, Jun 2008.

[29]J. Zhang, R. Yang, Z. Chen, Z. Zhao and B. Xu, "Automated EFSM-Based Test Case Generation with Scatter Search," ICSE Workshop AST, Accepted, 2012.