A Search-Based Technique for Automatic Test Generation from an Extended Finite State Machine

Abdul Salam Kalaji
Robert M. Hierons
Stephen Swift.
School of Information Systems, Computing and Mathematics
Brunel University
Uxbridge, UB83PH, UK.

Abstract

Extended finite state machines (EFSMs), and languages such as state-charts that are similar to EFSMs, are widely used to model state-based systems. When testing from an EFSM *M* it is common to aim to produce a set of test sequences (input sequences) that satisfies a test criterion that relates to the transition paths (TPs) of *M* that are executed by the test sequences. For example, we might require that the set of TPs triggered includes all of the transitions of *M*. One approach to generating such a set of test sequences is to split the problem into two stages: choosing a set of TPs that achieves the test criterion and then producing test sequences to trigger these TPs. However, the EFSM may contain infeasible TPs and the problem of generating a test sequence to trigger a given feasible TP (FTP) is generally uncomputable. In this paper we present a search-based approach that uses two techniques: (1) A TP fitness metric based on our previous work that estimates the feasibility of a given transition path; and (2) A fitness function to guide the search for a test sequence to trigger a given FTP. We evaluated our approach on five EFSMs: A simple in-flight safety system; a class II transport protocol; a lift system; an ATM; and the Inres initiator. In the experiments the proposed approach successfully tested approximately 96.75 % of the transitions and the proposed test sequence generation technique triggered all of the generated FTPs.

***Keywords*: Search-based testing, evolutionary testing, EFSM, automatic test derivation, test sequence generation.**

1. Introduction

Testing is an important stage of the software development process. However, manual testing is error-prone, expensive and time consuming and so there has been much interest in automated test data generation (see, for example, [1-4]). In this paper we are interested in conformance testing in which testing tries to find any differences between the behavior of an implementation under test (IUT) and its specification. Conformance testing treats the IUT as a black-box and so a tester has no information about the internal system structure and only input/output behavior is available.

In black-box testing, we apply a sequence of inputs, called a *test sequence*, and observe the resultant outputs. In order to automate test sequence generation we require a model of the IUT or of the aspect of the IUT to be tested. Finite state machines (FSMs) and extended finite state machines (EFSMs) are commonly used for the purpose of test sequence derivation [5]. However, an FSM can only model the control part of a system; an extension is needed in order to model a system with control and data parts, e.g., communication protocols. Such systems are usually represented by using an EFSM, possibly expressed using a language such as state-charts or SDL [6]. Many approaches to generating test sequences from an EFSM operate by first devising a set of transition paths (TPs) through the EFSM and then generating test sequences to trigger these paths.

However, in order for such approaches to be automated, two challenges must be overcome: producing feasible TPs and generating test sequences to trigger the feasible TPs.

Since an EFSM's transitions may have guards (preconditions) and operations, a given TP may be infeasible. For example, one transition's operation may assign the value 0 to a variable *x* while a later transition's guard requires $x>0$ despite the value of *x* not having changed between these transitions. Such a path is infeasible and so it is impossible to find test data to trigger it. However, the problem of determining whether a given path is feasible is undecidable and the development of good methods is an open research problem [7, 8]. If the path is feasible, then a test sequence is required to trigger (exercise) this path. Nevertheless, it can be difficult to find such a set since the input domain is usually quite large but the required input values might constitute just a small subset of this domain [9]. For example, a machine variable *x* can be of integer data type, but the required values to exercise a guard over *x* can be within a tiny range.

The approach of producing TPs and then generating test sequences to trigger these paths can be seen as one of initially converting the EFSM to an FSM by abstracting out the data and then using one of the many methods for testing from an FSM (see, for example, [10-13]). However this conversion approach does not guarantee that paths taken from the resultant FSM are feasible in the corresponding EFSM. An alternative approach to converting an EFSM to an FSM, is to expand out the data [14] but this can easily lead to the number of states in the resultant FSM being prohibitively large.

Although optimization algorithms have proven efficient for testing purposes [2], very little attention has been paid towards investigating their application to EFSM testing. This paper proposes a novel search-based approach to test from EFSMs. The proposed approach uses two techniques: (1) A TP fitness metric, based on our previous work [15], estimates the feasibility of a given path by analyzing the dataflow dependence among the operations and guards in the path's transitions and this guides the search for TPs with the aim of producing feasible TPs that satisfy the test criterion. (2) A fitness function guides the search for a test sequence that can trigger a given TP. The proposed search-based approach utilizes the first technique to generate paths that are likely to be feasible and satisfy a given test criterion, such as transition coverage, and then the second technique is used to try to trigger the resultant TPs. Potentially, these could be combined into an iterative algorithm in which additional TPs, with good fitness, are generated if we failed to produce test sequences to trigger the original TPs.

The main contributions of this paper are the following:

1- It describes a search-based method that directs the automatic generation of TPs from EFSMs models with the intention that the resultant TPs are feasible and relatively easy to trigger.

2- It proposes a search-based method for automatically generating a test sequence for a given TP.

3- The paper is the first to propose an integrated search-based approach for testing from an EFSM.

4- The paper empirically validates the efficiency of the proposed EFSM testing approach by using it with five EFSM case studies.

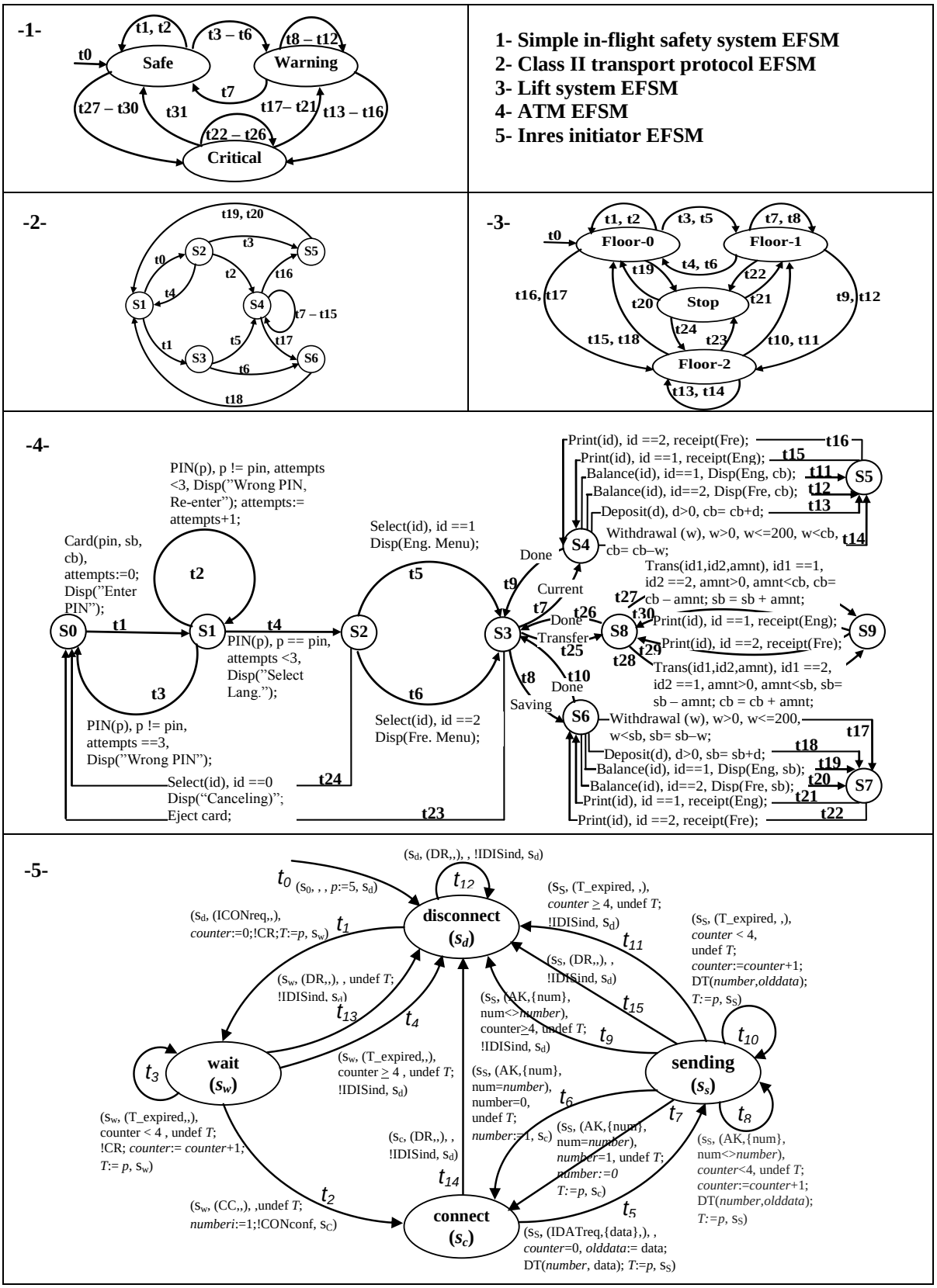


Figure 1. EFSM case studies

The rest of the paper is organized as follows: Section 2 provides background information, including a description of evolutionary algorithm (EA) and evolutionary testing (ET). In Section 3, the proposed approach is described. Experimental results are provided in Section 4 and related work is described in Section 5. Concluding remarks and future work are in Section 6.

2. Preliminaries

2.1. The model

A finite state machine (FSM) is a Mealy machine (or transducer), which has a finite set of states, inputs, and outputs. An output is produced upon state transition and this occurs when applying an input to the machine. An FSM model can successfully represent the control part of a system, such as a telephone device. However, an extension is needed in order to model a system, such as a communications protocol, that has control and data parts. When extending a Mealy machine with internal variables, predicates, and operations we get an extended finite state machine (EFSM). An EFSM is a 6-tuple [16] (S, s_0, V, I, O, t) where: S is the finite set of *logical states*, s_0 is the *initial state*, V is the finite set of *internal variables*, I is the finite set of *input declarations*, O is the finite set of *output declarations* and t is the finite set of *transitions*.

The transition $t \in T$ is represented by the 5-tuple (s_s, i, g, op, s_e) in which: s_s is the *start state* of t, i is the *input* where $i \in I$ and i may have associated input parameters, g is a logical expression called the *guard*, op is the *sequential operation* which consists of simple statements such as output statements and assignment statements and s_e is the *end state* of t. In an EFSM model, there is a set of variables. One variable in particular is used to represent the machine state[1] and is called *state* or *major state* in order to differentiate it from the other variables called *context variables*. The state variable is used to represent the logical state, such as idle, wait for connection and so on, whereas other machine data such as port number and sequencing numbers are stored in context variables. A state transition occurs when one of the machine's transitions is taken. If the state is s_s then transition t = (s_s, i, g, op, s_e) can be taken if input i is received and the guard g is satisfied. If this happens then the operations in op are executed and the logical state becomes s_e. Both g and op can refer to input parameters and context variables. An EFSM is deterministic if for any group of transitions with the same input that leave a state, it is not possible to satisfy the guards of more than one transition in this group at the same time [17]. In this paper, we only consider deterministic EFSMs.

2.2. Examples

In this paper we use the following five EFSMs, shown in Fig. 1, in the experiments:

1- **Simple in-flight safety system:** A synthesized simple system that functions as a monitor of the craft's cabin in terms of four factors: vibration, pressure, temperature and smoke. There are three states: (1) Safe when the values of these four factors are within a set of pre-defined ranges. (2) Warning when the value of one or more factors is within another set of pre-defined ranges. Here the pilot should take one or more actions according to a pre-defined list and the system can

[1] The state variable may be a tuple of values.

respond with some necessary actions i.e. when the air pressure is low, oxygen masks are released automatically. (3) Critical when the value of one or more factors is in a critical range and the pilot has to directly intervene. For example, if the pressure cannot be brought back to normal, an emergency landing might be taken. The EFSM has five context variables *V*= {VarsRead, Vb, Pr, Sm, Tm} and 31 transitions. Fig. 1-1 shows the EFSM and Table 1 lists the transitions specifications.

2- **Class II transport protocol:** This EFSM is a major model based on the *AP-module* of the simplified version of a class 2 transport protocol. The EFSM model represents the core protocol transitions as described in [16] and [18]. This EFSM has two interaction points *U* and *N* for connecting to transport service access point and a mapping module respectively. The EFSM is involved in connection establishment, data transfer, end-to-end flow control and segmentation. This EFSM has seven states $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$, five context variables *V* = {opt, R-credit, S-credit, TRsq , TSsq} and 21 transitions. The model is shown in Fig. 1-2 and the transitions are described in Table 2.

3- **Lift system:** A synthesized lift system for a building with three floors. In order to open or close the lift cabin's door, the lift should be situated in the specified place within a margin that does not exceed 15%. The lift provides three operations: Request a lift from a specified floor, Service from a floor to another floor and Stop when there is a request. When a door is closed, the cabin load's weight is read and stored. In order for the cabin to move, the temperature and smoke level inside the cabin should be within pre-defined ranges. The lift does not provide a service if the cabin load is less than or equal to 15 KG so that a small child cannot operate the lift alone. The lift EFSM has four states S= {$Floor_0$, $Floor_1$, $Floor_2$, Stop}, three context variables *V*= {Drst, w, Floor} and 24 transitions. The EFSM is shown in Fig. 1-3 and the transitions are described in Table 3.

4- **ATM:** This represents an extension of the machine described in [19]. The machine offers the option of English or French menu and provides three services: Deposit, Withdrawal and Transfer between two accounts (Current and Saving). In order for a transaction to occur, a user must provide a valid PIN within three tries otherwise the machine will cancel the operation. The ATM EFSM has ten states S= {s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9}, four context variables *V*= {PIN, cb, sb, attempts} and 30 transitions. Fig. 1-4 shows the ATM EFSM and its transitions specifications.

5- **Inres initiator:** The Inres [20] protocol is connection-oriented and comprises the initiator, which establishes a connection and sends data, and the responder which receives data and terminates connections. The Inres protocol was designed to be similar to real protocols and yet small enough to allow experiments to be conducted for research purposes. The Inres initiator has five states $S = \{s_0,$ disconnect, wait, connect, sending}, four context variables *V* = {counter, number, T, p} and 15 transitions. Fig. 1-5 shows the Inres initiator EFSM together with the transitions specifications.

In these five EFSMs, all the input parameters are of integer data type. When used, the symbol '?' indicates a request for an input whereas the symbol '!' indicates an output.

Table 1. The transitions specifications of the in-flight safety system EFSM

t $s_s \rightarrow s_e$	Input declarations	Guards	Transition atomic operations
t_0 $s_0 \rightarrow s_1$	reset	-	VarsRead= False; SetWarningLights(all, off); Sounds are switched off;
t_1 $s_1 \rightarrow s_1$ t_8 $s_2 \rightarrow s_2$ t_{22} $s_3 \rightarrow s_3$	?Read(Pvb, Ppr, Psm, Ptm)	VarsRead == False	Vb = Pvb; Pr = Ppr; Sm= Psm; Tm = Ptm; VarsRead = True;
t_2 $s_1 \rightarrow s_1$ t_7 $s_2 \rightarrow s_1$ t_{31} $s_3 \rightarrow s_1$	MainCheck1 ()	VarsRead == True **&** Vb ≥ 0 **&** Vb ≤10 Pr ≥ 86 **&** Pr ≤ 100 **&** Sm ≥0 **&** Sm ≤ 10 Tm ≥ 11 **&** Tm ≤ 35	VarsRead= False; SetWarningLights(all, off); Sounds are switched off;
t_3 $s_1 \rightarrow s_2$ t_9 $s_2 \rightarrow s_2$	CheckVb1()	VarsRead == True **&** Vb ≥ 11 **&** Vb ≤25	VarsRead= False; SetLight(Seatbelt, on);
t_4 $s_1 \rightarrow s_2$ t_{10} $s_2 \rightarrow s_2$	CheckPr1()	VarsRead == True **&** Pr ≥ 50 **&** Pr ≤ 85	VarsRead= False; Release(masks); SetLight(Seatbelt, on);
t_5 $s_1 \rightarrow s_2$ t_{11} $s_2 \rightarrow s_2$	CheckSm1()	VarsRead == True **&** Sm ≥ 11 **&** Sm ≤ 25	VarsRead= False; SetSound(Sm, off);
t_6 $s_1 \rightarrow s_2$ t_{12} $s_2 \rightarrow s_2$	CheckTm1()	VarsRead== True **&** (Tm ≥ 36 **&** Tm ≤ 46) **V** (Tm ≥ 3 **&** Tm ≤ 10)	VarsRead= False; SetLight(Tm, on);
t_{13} $s_2 \rightarrow s_3$ t_{23} $s_3 \rightarrow s_3$ t_{27} $s_1 \rightarrow s_3$	CheckVb2()	VarsRead == True **&** Vb >25	VarsRead= False; SetLight(Seatbelt, on);
t_{14} $s_2 \rightarrow s_3$ t_{24} $s_3 \rightarrow s_3$ t_{28} $s_1 \rightarrow s_3$	CheckPr2()	VarsRead == True **&** Pr ≥ 0 **&** Pr ≤ 49	VarsRead= False; Release(masks); SetLight(Seatbelt, on); SetSound(Pr, off);
t_{15} $s_2 \rightarrow s_3$ t_{25} $s_3 \rightarrow s_3$ t_{29} $s_1 \rightarrow s_3$	CheckSm2()	VarsRead == True **&** Sm > 25	VarsRead= False SetSound(Sm, off);
t_{16} $s_2 \rightarrow s_3$ t_{26} $s_3 \rightarrow s_3$ t_{30} $s_1 \rightarrow s_3$	CheckTm2()	VarsRead= True **&** (Tm >46) **V** (Tm ≤2)	VarsRead= False SetLight(Tm, on); SelLight(AC, on);
t_{17} $s_3 \rightarrow s_2$	MainCheck2()	VarsRead == True **&** Vb ≥ 11 **&** Vb ≤25 **&** Pr ≥ 50 **&** Pr ≤ 85 **&** Sm ≥ 11 **&** Sm ≤ 25 **&** (Tm ≥ 36 **&** Tm ≤ 46) **V** (Tm ≥ 3 **&** Tm ≤ 10)	VarsRead= False SetWarningLights(all, on); SetWarningSounds (all, off); Release(masks);
t_{18} $s_3 \rightarrow s_2$	MainCheck2()	VarsRead == True **&** Vb ≥ 11 **&** Vb ≤25 **&** Pr ≥ 86 **&** Pr ≤ 100 **&** Sm ≥0 **&** Sm ≤ 10 **&** Tm ≥ 11 **&**Tm ≤ 35	VarsRead= False; SetLight(Seatbelt, on);
t_{19} s3 $\rightarrow s_2$	MainCheck2()	VarsRead == True **&** Vb ≥ 0 **&** Vb ≤10 **&** Pr ≥ 50 **&** Pr ≤ 85 **&** Sm ≥0 **&** Sm ≤ 10 **&** Tm ≥ 11 **&**Tm ≤ 35	VarsRead= False; Release(masks); SetLight(Seatbelt, on); SetSound(Pr, off);
t_{20} $s_3 \rightarrow s_2$	MainCheck2()	VarsRead == True **&** Vb ≥ 0 **&** Vb ≤10 **&** Pr ≥ 86 **&** Pr ≤ 100 **&** Sm ≥ 11 **&** Sm ≤ 25 **&** Tm ≥ 11 **&**Tm ≤ 35	VarsRead= False; SetSound(Sm, off);
t_{21} $s_3 \rightarrow s_2$	MainCheck2()	VarsRead == True **&** Vb ≥ 0 **&** Vb ≤10 **&** Pr ≥ 86 **&** Pr ≤ 100 **&** Sm ≥0 **&** Sm ≤ 10 **&** (Tm ≥ 36 **&** Tm ≤ 46) **V** (Tm ≥ 3 **&** Tm ≤ 10)	VarsRead= False SetLight(Tm, on); SelLight(AC, on);

2.3. Program data flow dependence

Given a program and a variable *x* within this program, a statement at which *x* appears can be an assignment to *x* or a use of *x* (or both). An assignment statement defines or updates the value of *x* and so *x* is said to be *defined* at such a statement. A use of *x* occurs when *x* is referenced in a predicate (a predicate use/p-use) or *x* is referenced in a computation that either updates the value of a variable or is produced as output (a computation use/c-use). Give a program path between two statements n_1 and n_2, if *x* is not defined after n_1 and before n_2 then the path from n_1 to n_2 is a definition clear path for *x* [21]. If, in addition, n_1 is a definition of *x* and n_2 is a use of *x*, then statements n_1 and n_2 form a definition-use (*du*) pair for *x* and there is dataflow dependence between n_1 and n_2 [22]. In this paper we utilize dataflow information in EFSMs to define the proposed TP fitness metric.

Table 2. The core transitions in the class II transport protocol EFSM

t $s_s \rightarrow s_e$	Input declarations	Guards	Transition atomic operations
t_0 $s_1 \rightarrow s_2$	U?TCONreq(dst_add, prop_opt)	-	opt = prop_opt; R_credit =0; N!TrCR
t_1 $s_1 \rightarrow s_3$	N?TrCR(peer_add, opt_ind, cr)	-	opt= opt_ind; S_credit=cr; R_credit=0; U!TCONind
t_2 $s_2 \rightarrow s_4$	N?TrCC(opt_ind, cr)	opt_ind < opt	TRsq=0; TSsq=0; opt=opt_ind; S_credit=cr; U!TCONconf
t_3 $s_2 \rightarrow s_5$	N?TrCC(opt_ind, cr)	opt_ind > opt	U!TDISind; N!TrDR
t_4 $s_2 \rightarrow s_1$	N?TrDR(disc_reason, switch)	-	U!TDISind; N!terminated
t_5 $s_3 \rightarrow s_4$	U?TCONresp(accpt_opt)	accpt_opt < opt	opt= accpt_opt; TRsq=0; TSsq=0; N!TrCC
t_6 $s_3 \rightarrow s_6$	U?TDISreq()	-	N!TrDR
t_7 $s_4 \rightarrow s_4$	U?TDATAreq(Udata, E0SDU)	S_credit > 0	S_credit= S_credit -1; TSsq = (TSsq +1)mod128; N!TrDT
t_8 $s_4 \rightarrow s_4$	N?TrDT(Send_sq, Ndata, E0TSDU)	R_credit != 0 **&** Send_sq== TRsq	TRsq=(TRsq+1)mod128; R_credit=R_credit -1; U!DATAind; N!TrAK
t_9 $s_4 \rightarrow s_4$	N?TrDT(Send_sq, Ndata, E0TSDU)	R_credit == 0 **V** Send_sq != TRsq	U!error; N!error
t_{10} $s_4 \rightarrow s_4$	U?U READY(cr)	-	R_credit= R_credit + cr; N!TrAK
t_{11} $s_4 \rightarrow s_4$	N?TrAK(XpSsq, cr)	TSsq > XpSsq **&** cr + XpSsq – TSsq ≥ 0 **&** cr +XpSsq – TSsq ≤ 15	S_credit = cr + XpSsq – TSsq
t_{12} $s_4 \rightarrow s_4$	N?TrAK(XpSsq, cr)	TSsq ≥ XpSsq **& (**cr + XpSsq – TSsq < 0 **V** cr +XpSsq – TSsq >0**)**	U!error; N!error
t_{13} $s_4 \rightarrow s_4$	N?TrAK(XpSsq, cr)	TSsq < XpSsq **&** cr + XpSsq – TSsq – 128 ≥ 0 **&** cr + XpSsq – TSsq – 128 ≤ 15	S_credit= cr+ XpSsq –TSsq – 128
t_{14} $s_4 \rightarrow s_4$	N?TrAK(XpSsq, cr)	TSsq < XpSsq **& (**cr + XpSsq – TSsq – 128 < 0 **V** cr + XpSsq – TSsq – 128 > 15 **)**	U!error; N!error
t_{15} $s_4 \rightarrow s_4$	N?Ready()	S_creidit > 0	U!Ready
t_{16} $s_4 \rightarrow s_5$	U?TDISreq()	-	N!TrDR
t_{17} $s_4 \rightarrow s_6$	N?TrDR(disc_reason, switch)	-	U!TDISind; N!TrDC
t_{18} $s_6 \rightarrow s_1$	N?terminated()	-	U!TDISconf
t_{19} $s_5 \rightarrow s_1$	N?TrDC()	-	N!terminated; U!TDISconf
t_{20} $s_5 \rightarrow s_1$	N?TrDR(disc_reason, switch)	-	N!terminated

2.4. Evolutionary algorithms (EAs)

Evolutionary algorithms are optimization techniques that adapt the evolution notion as a search mechanism. Genetic Algorithms (GAs) are a class of EA inspired by natural selection and have been found to be powerful, simple, and sturdy. In order to apply a GA to an optimization problem, a solution representation (encoding) is required. When solutions are encoded, each is called a *chromosome* and consists of components that are called *genes* [23]. For example, let the initial set of solutions be integer values such as {7, 6, 8}. If binary encoding is performed, then {0111, 0110, 1000} represents the chromosomes. Any bit of a chromosome represents a gene with a value of either 0 or 1.

The GA cycle starts by evaluating the fitness of each individual which is a positive value that measures how ‘fit’ this individual is and influences its chance of being selected as a parent. Then *selection* based on fitness is made to perform ‘breeding’. There are many selection methods, such as roulette wheel and ranking, that can be used [24]. Through breeding new individuals are introduced. This is accomplished by applying a *crossover* operator that acts on two individuals to produce two new individuals. There are several

Table 3. The transitions specifications of the Lift system EFSM

t $s_s \rightarrow s_e$	Input declarations	Guards	Transition atomic operations
$t_0 \rightarrow s_0$	reset		Floor = 0; DrSt = 0; w = 0;
t_1 $s_0 \rightarrow s_0$	?DrOp(Pos)	DrSt == 0 & Pos ≥ 0 & Pos ≤15	DrSt = 1;
t_2 $s_0 \rightarrow s_0$	?DrCl(Pos, Pw)	DrSt == 1 & Pos ≥ 0 & Pos ≤15	DrSt = 0; w = Pw
t_3 $s_0 \rightarrow s_1$	?Srv(Pf, Ph, Ps)	DrSt == 0 & Pf == 1 & w ≥15 & w ≤ 250 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 1; !Display(Floor);
t_4 $s_1 \rightarrow s_0$	?Srv(Pf, Ph, Ps)	DrSt == 0 & Pf == 0 & w ≥15 & w ≤ 250 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 0; !Display(Floor);
t_5 $s_0 \rightarrow s_1$	?Req (Pf, Ph, Ps)	DrSt == 0 & Pf == 1 & w =0 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 1; !Display(Floor);
t_6 $s_1 \rightarrow s_0$	?Req (Pf, Ph, Ps)	DrSt == 0 & Pf == 0 & w =0 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 0; !Display(Floor);
t_7 $s_1 \rightarrow s_1$	?DrOp(Pos)	DrSt == 0 & Pos ≥ 0 & Pos ≤15	DrSt = 1;
t_8 $s_1 \rightarrow s_1$	?DrCl(Pos, Pw)	DrSt == 1 & Pos ≥ 0 & Pos ≤15	DrSt = 0; w = Pw
t_9 $s_1 \rightarrow s_2$	?Srv(Pf, Ph, Ps)	DrSt == 0 & Pf == 2 & w ≥15 & w ≤ 250 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 2; !Display(Floor);
t_{10} $s_2 \rightarrow s_1$	?Srv(Pf, Ph, Ps)	DrSt == 0 & Pf == 1 & w ≥15 & w ≤ 250 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 1; !Display(Floor);
t_{11} $s_2 \rightarrow s_1$	?Req (Pf, Ph, Ps)	DrSt == 0 & Pf ==1 & w =0 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 1; !Display(Floor);
t_{12} $s_1 \rightarrow s_2$	?Req (Pf, Ph, Ps)	DrSt == 0 & Pf == 2 & w =0 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 2; !Display(Floor);
t_{13} $s_2 \rightarrow s_2$	?DrOp(Pos)	DrSt == 0 & Pos ≥ 0 & Pos ≤15	DrSt = 1;
t_{14} $s_2 \rightarrow s_2$	?DrCl(Pos, Pw)	DrSt == 1 & Pos ≥ 0 & Pos ≤15	DrSt = 0; w = Pw
t_{15} $s_2 \rightarrow s_0$	?Srv(Pf, Ph, Ps)	DrSt == 0 & Pf == 0 & w ≥15 & w ≤ 250 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 0; !Display(Floor);
t_{16} $s_0 \rightarrow s_2$	?Srv(Pf, Ph, Ps)	DrSt == 0 & Pf == 2 & w ≥15 & w ≤ 250 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 2; !Display(Floor);
t_{17} $s_0 \rightarrow s_2$	?Req (Pf, Ph, Ps)	DrSt == 0 & Pf ==2 & w =0 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 2; !Display(Floor);
t_{18} $s_2 \rightarrow s_0$	?Req (Pf, Ph, Ps)	DrSt == 0 & Pf == 0 & w =0 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 0; !Display(Floor);
t_{19} $s_0 \rightarrow s_s$	?Stp (Pf, Ph, Ps)	DrSt == 0 & Pf == 100 & w ≥15 & w ≤ 250 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 100; !Display(Floor);
t_{20} $s_s \rightarrow s_0$	?Srv(Pf)	DrSt == 0 & Pf == 0	Floor = 0; !Display(Floor);
t_{21} $s_s \rightarrow s_1$	?Srv(Pf)	DrSt == 0 & Pf == 1	Floor = 1; !Display(Floor);
t_{22} $s_1 \rightarrow s_s$	?Stp (Pf, Ph, Ps)	DrSt == 0 & Pf == 100 & w ≥15 & w ≤ 250 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 100; !Display(Floor);
t_{23} $s_2 \rightarrow s_s$	?Stp (Pf, Ph, Ps)	DrSt == 0 & Pf == 100 & w ≥15 & w ≤ 250 & Ph ≥ 10 & Ph ≤ 35 & Ps ≥ 0 & Ps ≤25	Floor = 100; !Display(Floor);
t_{24} $s_s \rightarrow s_2$	?Srv(Pf)	DrSt == 0 & Pf == 2	Floor = 2; !Display(Floor);

approaches to crossover including one-point crossover, which operates by choosing a random position on the chromosome's bit string, and then the substrings before that position are kept while the tails are swapped [25]. For example, if the two parents' chromosomes are P_1 and P_2 with crossover point at position 4, then C_1 and C_2 are the offspring chromosomes.

$$P_1\ \{011|00\} \qquad \Longrightarrow \qquad C_1\ \{011|11\}$$
$$P_2\ \{101|11\} \qquad\qquad\ \ C_2\ \{101|00\}$$

In order to maintain population diversity, new characteristics are infrequently injected by applying *mutation*. Mutation acts on one chromosome at a time, where it randomly changes the values of some of the chromosome's genes [25]. For example, the chromosomes C_1 above might become C_1' after mutating the bits on positions 1 and 5.

$$C_1\ \{01111\} \Longrightarrow C_1'\ \{11110\}$$

These operators yield new individuals and selection is used to obtain a new generation (population) from the previous population and the new individuals. The population undergoes a number of updates until satisfying one of the stopping criteria such as finding the best solution or reaching a maximum number of generations [26].

2.5. Evolutionary Testing (ET)

Evolutionary testing (ET) is a technique that employs EA to automatically generate test data. Test data generation is represented as a minimization problem where the lower the fitness of a solution the better it is and the optimal solution(s) will have a fitness equal to zero. When applying ET to generate test data we need the fitness function to correspond to the test adequacy criterion (a property that a test must satisfy in order to be considered sufficient). Many test adequacy criteria require that a set of structures in the code or model are covered in testing [4]. For example, we might require that all of the statements in the code are exercised (covered) in testing (statement coverage). If we consider the branch coverage criterion, then all the branches in the subject program need to be taken (covered). Test generation can involve a sequence of phases where in each phase we consider a single branch. In this case, an objective (fitness) function that depends on branch distance can be used in order to evaluate an input value. Branch distance measures how close a particular input was to executing the target branch that is missed. For example, $|x\text{-}y|$ is the branch distance for the predicate ($x < y$): the lower $|x\text{-}y|$ is the closer x is to y and the closer the test is to taking the branch. A full list of different types of conditions and their branch distance computations is provided by Tracey et al. [27].

Often, programs have nested predicates, for example an IF statement could be contained in a loop. In this case, an objective function which only employs branch distance is not sufficient and we require extra information to guide the search. This is given in terms of approach level [28] which measures how close an input was to executing the structure under test. A central notion to approach level calculation is a *critical node* which is a branching node at which the control flow may divert. Approach level is calculated by subtracting 1 from the number of critical nodes away from the target node at which the computation diverges (Equation 2). Since it is necessary that the branch distance of the upper IF statement is always greater than the ones in a lower level, the branch distance of each IF statement is normalized, using the *norm* function, to a value in the range of [0..1] (Equation 1). The normalized branch distance is then added to the approach level of that branch to form the fitness value of the test case (Equation 3). As a result, a test input that achieves more conditions (longer path) will have a better (lower) fitness than a test input that achieves fewer conditions.

$$norm\ (branch\_distance) = 1 - 1.05^{-\ branch\_distance} \quad (1)$$

$$approach\ level = numOfCriticalNodesAwayFromTarget - 1 \quad (2)$$

$$fitness = approach\ level + norm\ (branch\_distance) \quad (3)$$

A recent survey [2] has focused on evolutionary test data generation. The EFSM path test data generation technique presented in this paper adapts the notion of branch distance and approach level in order to construct a new fitness function as described in Section 3.

3. The Proposed Approach

In this section, we describe the two techniques that we use in our approach: feasible transition path generation and test sequence generation.

3.1. Feasible transition path (FTP) generation

In this subsection we describe the FTP generation approach which is based on our previous work [15]. First, we introduce some definitions.

Definition 1: A *transition path* (*TP*) of length *n* is a sequence of *n* consecutive transitions $t_1, t_2, .., t_n$.

Definition 2: A TP is an *FTP* if it is possible to trigger each transition t_i, $1 \leq i \leq n$, and in the sequential order that it appears in this TP.

Any path from the initial state of an EFSM defines a TP but only some of these paths may be FTPs. For example, in the Inres initiator (Fig. 1-5), the path $t_1t_3t_3$ is an FTP but t_1t_4 is not since t_1 sets the value of *counter* to 0 and then t_4 requires that *counter* is at least 4.

A transition's guard has the form of *(e gop e')* where *e* and *e′* are expressions and *gop* $\in \{<, >, \neq, =, \leq, \geq\}$ is the guard operator. Given an expression *e*, we let *Ref(e)* to denote the set of variables that appear in *e*. According to *e* and *e′* a transition's guard can be classified into the following types:

1. g^{pv}: a comparison involving a parameter and one or more context variables; *Ref(e)* $\cup$ *Ref(e')* contains a parameter and also context variables. An example is the transition t_2 in the ATM (Fig. 1-4) since it inputs a PIN *p* and then compares this with the correct PIN.
2. g^{vv}: a comparison among context variables' values; every element of *Ref(e)* $\cup$ *Ref(e')* is a context variable.
3. g^{vc}: a comparison between a constant and an expression involving context variables; all elements of *Ref(e)* $\cup$ *Ref(e')* are context variables and either *e* or *e′* is a constant. An example is the transition t_3 in the Inres initiator (Fig. 1-5) since its guard references a context variable *counter*, compares it to a constant and does not reference an input parameter.
4. g^{pc}: a comparison between a constant and an expression involving a parameter; there exists a parameter $p \in$ *Ref(e)* $\cup$ *Ref(e')* and either *e* or *e′* is a constant. Transition t_2 in the ATM (Fig. 1-4) would be an example of this if we considered the correct PIN to be a constant rather than a context variable.
5. g^{pp}: a comparison between expressions involving parameters; there exists a parameter $p \in$ *Ref(e)* $\cup$ *Ref(e')*.

An assignment that occurs in a transition *t* has the form of *v=e*, where *v* is a context variable and *e* is an expression. An assignment to a context variable *v* can be classified as one of the following types:

1. op^{vp}: it assigns to *v* a value that depends on the parameter and so there is a parameter $p \in$ *Ref(e)*. An example is the transition t_2 in the Lift system (Fig. 1-3) since it inputs the cabin's load weight *pw* and updates the value of the context variable *w* on the basis of this.

2. op^{vv}: it assigns to v a value that depends on the context variable(s) and so all the elements of *Ref(e)* are context variables. An example is the transition t_2 in the ATM (Fig. 1-4) since it updates the value of the context variable *attempts* by using the value of the context variable *attempts*.
3. op^{vc}: it assigns to v a constant value and so e is a constant. An example is the transition t_1 in the Inres initiator (Fig. 1-5) since it defines the value of the context variable *counter* by a constant.

Based on the classifications of guards and assignments, we can distinguish two types of transitions: affecting and affected-by transitions.

Definition 3: In a TP $t_1, t_2, .., t_n$, t_i is an *affecting* transition if t_i has an assignment $op \in \{op^{vp}, op^{vc}, op^{vv}\}$ to v and there exists a guarded transition $t_j \in$ TP, where $1 \leq i < j \leq n$, t_j has a guard $g \in \{ g^{pv}, g^{vv}, g^{vc}\}$ over v and the path from t_i to t_j is definition clear for v. We also say that t_j is an *affected-by* transition.

For example, in the Inres initiator, in Fig. 1-5, the transition t_1 assigns a value to the context variable *counter* and t_3 guard references this variable, and there is a definition clear path t_1t_3 from t_1 to t_3 and so t_1 is an affecting transition and t_3 is an affected transition.

Definition 4: For variable v, assignment op of type op^{vc} is *opposed* to guard g of type g^{vc} when the path from op to g is definition clear for v and either the constants that appear in op^{vc} and g^{vc} are the same and $gop \in \{<, >, \neq\}$ or are different and $gop \in \{=\}$.

If we again consider the Inres initiator, we find that the assignment to *counter* in transition t_1 is opposed to the guard in t_4 since t_1 sets *counter* to 0 and t_4 requires *counter* to be at least 4. As a result any path that contains the subsequence t_1t_4 must be infeasible.

Definition 5: For variable v, guards g_1 and g_2 of type g^{vc} are *opposed* when the path from g_1 to g_2 is definition clear for v and either the constants that appear in $g_1{}^{vc}$ and $g_2{}^{vc}$ are the same and ($g_1op \in \{\neq, >, <\}$ and $g_2op \in \{=\}$ or $g_1op \in \{>, \geq\}$ and $g_2op \in \{<\}$ or $g_1op \in \{<, \leq\}$ and $g_2op \in \{>\}$) or the constants are different and $g_1op, g_2op \in \{=\}$.

By Definitions 3-5, we can define two cases where a TP is clearly infeasible:

Definition 6: A TP $t_1, t_2, .., t_n$ with length $n >1$ is *definitely infeasible* if there exists a variable v and a pair of transitions (t_i, t_j) where $1 \leq i < j \leq n$, t_i is an affecting transition of type op^{vc}, t_j is an affected-by transition of type g^{vc} and op^{vc} opposes g^{vc}. An example is the transition sequence t_1t_4 in the Inres initiator (Fig. 1-5).

Definition 7: A TP $t_1, t_2, .., t_n$ with length $n >1$ is *definitely infeasible* if there exists a variable v and a pair of transitions (t_i, t_j) where g_i and g_j are of type g^{vc} and g_i opposes g_j. An example of this is the transition subsequence t_4t_5 in the Lift system (Fig. 1-3) since t_4 requires the value of the context variable w to be in [15..250] while t_5 requires the value of the same context variable w to be 0 and the path t_4t_5 is definition clear for w. Thus any path that contains the subsequence t_4t_5 must be infeasible.

3.1.1. Dependencies representation and penalties

In this subsection we describe a TP fitness metric that aims to estimate the 'feasibility' of a TP without executing it. In order to estimate the feasibility of a TP, we perform an analysis of all dependencies among the affecting and affected-by transitions in this TP. The aim is to have a fitness metric that can be used in search and so we need this to be

Table 4. The suggested penalty values

Guard & operator	Assignment			
	(nop)	**(op^{vp})**	**(op^{vv})**	**(op^{vc})**
$g^{pv}(=)$	4	8	16	24
$g^{pv}(<, >)$	3	6	12	18
$g^{pv}(\leq , \geq)$	2	4	8	12
$g^{pv}(\neq)$	1	2	4	6
$g^{vv}(=)$	16	20	40	60
$g^{vv}(<, >)$	12	16	32	48
$g^{vv}(\leq , \geq)$	8	12	24	36
$g^{vv}(\neq)$	4	8	16	24
$g^{vc}(=)$	40	30	60	10000 if *c* is different and 0 otherwise
$g^{vc}(<, >)$	32	24	48	0 if *c* is different and 10000 otherwise
$g^{vc}(\leq , \geq)$	24	18	36	10000 if *c* is different and 0 otherwise
$g^{vc}(\neq)$	16	12	24	0 if *c* is different and 10000 otherwise
$g^{pc}(=)$	4	-	-	-
$g^{pc}(<, >)$	3	-	-	-
$g^{pc}(\leq , \geq)$	2	-	-	-
$g^{pc}(\neq)$	1	-	-	-
$g^{pp}(=)$	4	-	-	-
$g^{pp}(<, >)$	3	-	-	-
$g^{pp}(\leq , \geq)$	2	-	-	-
$g^{pp}(\neq)$	1	-	-	-
g_i opposes g_j	10000	-	-	-

computationally simple. Our TP feasibility metric is therefore based on a set of approximate penalty values that are determined in advance.

The penalty value is a numerical estimate of how easily a given guard can be satisfied. Since a guard can be affected by a previous operation, we consider three factors when assigning a penalty value to a pair of (affecting, affected-by). First, we consider the guard type. For example, a guard of type g^{vc} can be classified as the hardest since the option of selecting the values of either *c* or *v* is not available. In contrast, g^{pv} is typically easier to satisfy since we can choose the value of the parameter. Secondly, we consider the guard operator. For example, the operator = is normally the most difficult to satisfy and ≠ is the easiest. Finally, we consider the operation of an affecting transition. For example, an operation of type op^{vp} is potentially useful since the parameter provides an opportunity to try to select a suitable value while op^{vc} is the worst since it is not possible to select the value of *c*. In addition to the penalty between a pair of (affecting, affected-by), it is possible to have a guard that is not affected by any operation (e.g. g^{pc}) and for such a case, we consider only the first two factors when assigning a penalty value. Table 4 shows the suggested penalty values for all possible combinations among affecting and affected-by transitions; for cases where there are no affecting transitions we use '–' to indicate that the choices op^{vp}, op^{vv} and op^{vc} are irrelevant. In the case where a TP is definitely infeasible we give a penalty of 10000 and this value has been chosen for the path length 10 used in the experiments. If a different path length is used we suggest that the penalty should be 1000 times the path length.

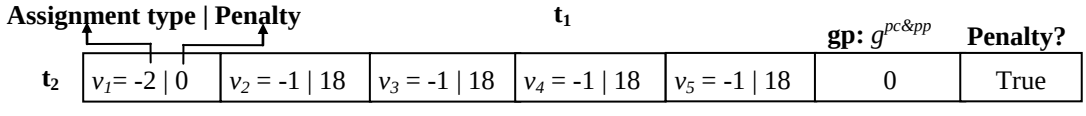
A guard can be given using nested IFs or predicates linked by AND and OR. For guards that are represented as nested IF or linked by AND, the sum of penalties is applied, however, the minimum penalty is considered when an OR operator is present.

Table 5. Assignment's types representation

op	Representation	Meaning
op^{vp}	-1	An assignment to *v* that references a parameter and no context variables
op^{vc}	-2	An assignment of a constant to *v*.
op^{vv}	$v_{1..}v_n$	An assignment to *v* that references context variables
nop	0	There is no assignment and so no dependency or open ended dependency

The dependency between affecting and affected-by transitions can occur on the basis of one or more context variables and an affected-by transition can be affected by one or more transitions in a given TP. Therefore, we record each dependency between a pair of (affecting, affected-by) transitions and the context variable at which the dependency occurs. There are three types of assignments and we represent each type by an integer: -2 and -1 mean an assignment of a constant value (op^{vc}) and an assignment of a parameter value (op^{vp}) respectively while an assignment that references a context variable (op^{vv}) is represented by a positive integer in [1..m] (*m* context variables). A number in [1..m] represents the corresponding context variable appearing on the right-hand side of the assignment. If an assignment of type (op^{vv}) references more than one context variables, we simplify the calculation by using only one of these. We observe that if we can easily set (choose) the value of one of these context variables then it may be less important whether we can set the values of the others. Consider, for example, the problem of satisfying a guard $v=v'$ for context variables v and v'. If we can easily set the value of v using a parameter p then we may be able to choose values for the other parameters, note the value of v' and then decide the value of p. As a result we choose the variable v_j referenced by considering the chain of previous assignments that compute the value of v_j in the TP and the following preference: (1) the first (earliest) assignment references a parameter, (2) the first assignment references a constant; and (3) v_j has the shortest chain of assignments that update its value. Having chosen the v_j we compute the value as above. It is possible that there is no assignment (*nop)* and so no dependency between the transitions, or there is an open-ended dependency (a variable references another variable which is not defined). We represent such cases by 0. Table 5 lists the dependency types and their representation.

Example 1. The EFSM in Fig. 1-1 has five context variables VarsRead, Vb, Pr, Sm, Tm which we will refer to henceforth by v_1, v_2, v_3, v_4, v_5 respectively. Consider transitions t_1 and t_2 from Table 1: t_2 is an affected-by transition of type g^{vc} and t_1 is an affecting transition of type op^{vc} at v_1 and of type op^{vp} at v_2, v_3, v_4 and v_5. From Table 4, the penalty value is 0 + 18 + 18 + 18 + 18 = 72. Dependencies between t_1 and t_2 occur at all context variables so we represent the dependencies as a seven-tuple. The first five fields record the dependency and penalty which occur at each context variable and the sixth, *gp*, records the sum of penalties of guards that do not involve context variables. The last field is a Boolean and used to record whether there is a penalty between the considered two transitions. The first five fields have two parts: the dependency type and the associated penalty value:



The information in the above tuple can be read by the help of Tables 4 and 5 as: there is a dependency between transitions t_1 and t_2 at v_1 where the earlier chain of dependencies

starts with an assignment of a constant value and the associated penalty is 0. Similarly there are dependencies at v_2, v_3, v_4 and v_5 that end (when working backwards) with an assignment that references parameter values and the penalty is 18 points for each. Also, all guards of t_2 involve context variable and so the *gp* field has the value of 0.

The mentioned tuple of information is stored in an array, a *relation array*, to represent the dependencies and penalties among all the transitions in a given EFSM. The array has the size of $n \times n$ where n is the number of transitions in the considered EFSM. Affected-by transitions are rows whereas columns represent affecting transitions. Each cell in this array has the form of the mentioned tuple.

3.1.2. The fitness metric

Fig. 2 shows a high-level description of the algorithm that calculates the TP fitness metric. The inputs are the transition relation array and a TP with length $n > 1$. The algorithm first considers the penalty of any guards that do not involve context variables (Line 10). It then treats the last transition as a potential affected-by transition and determines which previous transition are affecting (Line 13). If the current pair of transitions (t_{n-1}, t_n) forms a pair of (affecting, affected-by) then a loop is entered (Line 15) to decide which context variables provide a dependency or a penalty. There are two cases: (1) The dependency type is in [-2..0], the related variable is set to be checked (Line 19) and the corresponding penalty is accumulated. (2) The dependency type is > 0 which means that the dependency continues by an assignment referencing context variables, the related variable is set to be checked, the corresponding penalty is accumulated and a call is made to a subroutine *check* to detect all the previous assignments that are propagated to the current context variable.

The recursive *check* subroutine performs data dependency analysis by starting from the context variable and affecting transition passed to the call and working backwards to find all previous transitions that may affect the context variable (Line A9). If an earlier transition t_p is found to affect the context variable, then the subroutine finds the type of the assignment (Line A10). If the assignment type is found to be < 0 then the context variable is assigned either a constant or a parameter value. Then the subroutine penalizes referencing to a constant with 60 points and to a parameter with 20 points and stops (no earlier assignments affect this assignment). If the assignment type is > 0, the assignment references a context variable v'. Here, the subroutine penalizes this referencing by 40 points and repeats the process by calling *check* with t_p and v' (Line A15). If the dependency is open ended (depends on an undefined the initial value of a variable) then 60 points is added (Line A22). When the subroutine stops (Line A21 or A22) it returns the sum of penalties. After the current pair of transitions (t_{n-1}, t_n) is scanned, another cycle starts to detect any possible relation and penalty between the next pair (t_{n-2}, t_n) (Line 12) and so forth.

3.1.3. The GA encoding for FTP generation

The proposed FTPs generation approach uses the encoding technique from [29] in which a TP is represented by a sequence of integers where each number defines a transition. Given an EFSM with k states, let n_1, n_2.. n_k be the number of transitions leaving each state. Then, the method calculates the lowest common multiplier *LCM* of n_1, n_2.. n_k. The last step is to define the ranges r_1, r_2.. r_k for each state as $r_i = LCM / n_i$. A chromosome is a sequence of

A TP fitness metric

```
1. begin
2. input: TP, EFSM analysis array
3. output: non negative integer value
4. goal: evaluate a TP complexity
5. initialize variable result = 0; , bool array [1..vk]
6. for i = n downto first_transition
7. begin
8.   bool array [1..vk]= false;
     // reset bool array so there is currently no penalty recorded at any Var.
9.   j= i;
10.  result := result + [ti,tj].gp;
      // get the penalty of guards that do not have context Vars.
11.  repeat
12.     j = j -1;
13.     if [ti,tj].penalty == true then
     // if there is a penalty between these two Trans.
14.       begin
15.          for vi = v1 to vk do
     // check at which context Var. the dependency occurs
16.           begin
17.             (if [ti,tj].vi ≤ 0) &&(not bool[vi]) then
     // the dependency ends by a Param., const or no dependency
18.              begin
19.                bool[vi] = true;
     / /don't check the penalty at this Var next time
20.                   result := result + [ti,tj].vi(penalty)
21.              end;
22.             (if [ti,tj].v > 0) &&(not bool[vi]) then
     // the dependency continues by referencing a context Var.
23.              begin
24.                 bool[vi] = true;
     / /don't check the penalty at this Var next time
25.                    result := result + [ti,tj].vi(penalty) + check(ti,tj,vi);
     // call check function to trace back all the dependencies that propagated
     // at this Var.
26.              end;
27.          end;
28.       end;
29.    until j = first_transition
30.  end;
31. return result;
32. end
```

Function *check* all of a transition dependencies

```
A1. begin
A2. input: ti,tj,v
A3. output: non negative integer value
A4. goal: trace back a flow dependence on variable v
A5. initialize variable result = 0; found = false;
A6. begin
A7.   p = j + 1;
A8.   repeat
A9.     p = p – 1;
A10.    if [ti,tp].vi ≠ 0 then
A11.     begin
A12.       case [ti,tp].vi of
       // check the type of dependency
A13.       -2   : result = result + 60;
       // Assignment to a constant
A14.       -1   : result = result + 20;
       // Assignment to a Param.
A15.       1..k : result = result + 40 + check(tp, tp-1, v1..k)
       // Assignment to a context Var. recall check
       // function to trace back all the dependencies
       //propagated at this context Var.
A16.       end;
A17.       found = true;
A18.     end;
A19.   until P = first_transition or found;
A20. end;
A21. if found then return result
A22. else return result + 60;
       // the dependency is left open ended
A23. end.
```

Figure 2. High level description of the algorithm that calculates the TP fitness metric

integers i_1, $i_2..i_n$, each in the range [1..*LCM*]. Each number i_i is divided by the corresponding r_j to determine the transition it defines. By using this method of encoding, every sequence defines a TP.

Example 2. The EFSM in Fig. 1-1 has k = 3 states, n_1 = 10, n_2 = 10 and n_3 = 11. Thus *LCM* = 110 and r_1 = 11, r_2 = 11 and r_3 = 10. If a sequence of integer is generated in the range [1..110] i.e. <5, 55 , 99> then by starting from the first state, the first number represents t_1. Since t_1 ends at the same state, we use r_1 and the second number represents t_5. Similarly, t_5 ends at the second state and so we use r_2 and the last number represents t_{15}. The final TP is: $t_1 \rightarrow t_5 \rightarrow t_{15}$.

3.2. Test sequence generation

Once there is a set of generated TPs that satisfy a test criterion such as transition coverage, there is a need to trigger each TP in this set and so a method is required to generate test sequences to exercise these TPs. Any EFSM transition can be treated as a function where the function name and input parameters are taken from the corresponding transition name

```
Double fun1( int x, int y)
 {
  if x >=10
   {if x <=20
    {if y >=0
     {if y <=10
      //result = 0 //Target achieved
 }}}}
```

Figure 3. A transition with nested IF statements

and input parameters [30]. Thus, a TP defines a sequence of function calls and a fitness function is required to guide search for inputs that can trigger the TP.

The fitness calculation method proposed by Wegener et al. [28] and described in Subsection 2.5 is effective in structural testing where the test target is represented as a single node in the main body of a function or a program and this method can be applied to one function at a time. The approach of Tracey et al. [27] can also be used with one function at a time. However, the work of Wegener et al. [28], as argued in [2], is more efficient than the Tracey et al. [27] approach since depending merely on branch distance to calculate the fitness can cause the search to become stuck in a local minimum. For example, consider an arbitrary transition (fun_1) shown in Fig. 3 which requires two suitable input values to achieve four nested IF statements. Fig. 4 shows the two fitness function landscapes of (fun_1): the first is calculated by using Wegener et al. [28] approach and the second is calculated by using Tracey et al. [27] approach. The unnecessary plateaux in the landscape of Tracey et al. [27] can make search more difficult.

In a path, there is more than one function and so a new fitness function is required. We propose a fitness function that comprises two components: *function distance* and *function approach level* [30]. The *function distance* can be calculated by using Wegener et al. [28] method and so it is equal to zero when the function is taken or it reflects how close a given input was to executing this function. Each guarded transition in a path is considered as a *critical function* at which the execution flow may divert. Therefore, we calculate *function approach level* by subtracting 1 from the number of critical functions away from the target. The *function approach level* is similar to the *approach level* (see Equation 2) and used to determine how close a test sequence was to triggering an extra transition in a path. Based on this description the proposed fitness function can be given as:

$$path\ fitness = norm\ (function\ distance) + function\ approach\ level \quad (4)$$

$$function\ approach\ level = NumOfCrticalTransAwayFromTarget - 1 \quad (5)$$

$$function\ distance = norm(branch\ distance) + approach\ level \quad (6)$$

Example 3. Consider a path with transition sequence: $fun_1(x_1,y_1) \rightarrow fun_1(x_2,y_2) \rightarrow fun_1(x_3,y_3)$. In applying the proposed fitness function (Equation 4), we first calculate the *function distance* for each transition by using Wegener et al. [28] approach. Since (fun_1) is a guarded transition (see Fig. 3), we consider each transition in a path as an IF statement and so we calculate the *function approach level* at each false exit. Fig. 5 shows the fitness calculation for the path: $fun_1(x_1,y_1) \rightarrow fun_1(x_2,y_2) \rightarrow fun_1(x_3,y_3)$ using the proposed method.

The manipulation of a path in this way is similar to the structure of nested IF statements where each IF statement compares the associated function's return value with 0. A similar notion of calculating a path fitness is introduced in [31], however, they calculate the *function distance* by using the approach of Tracey et al. [27]. As a result, this method

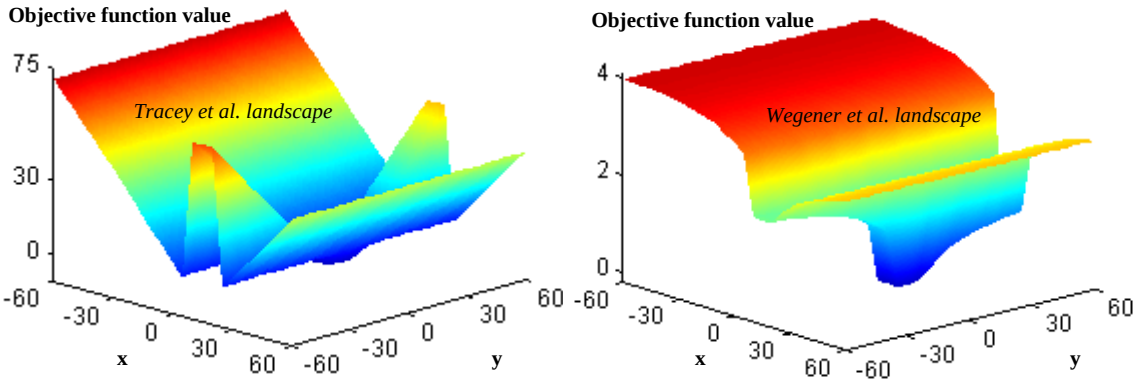


Figure 4. Objective function landscapes of (fun_1) to compare Tracy et al. and Wegener et al. fitness calculations.

can experience difficulties in the presence of nesting. Later we use experiments to compare the approach proposed in this paper by that of [31].

Naturally, transitions' guards can be sequenced as nested IF statements or linked with logical operators AND and OR. In order to apply the proposed fitness metric, guards linked with AND operator are represented as nested IF statements when calculating *function distance*.

If guards are linked with an OR, we split the transition into a number of transitions equal to the number of OR operators + 1. One benefit of this is that we test each predicate/condition in a guard, however, the alternative would be to use the minimum fitness value for a set of conditions linked with OR operator [27]. We will refer to our proposed test sequence generation approach by (ET-1) and that of [31] by (ET-2).

3.2.1. GA encoding for test sequence generation

An encoding is required and this can be selected on the basis of the machine input parameter types. It is possible to use *binary* or *integer* encoding when all of the considered machine input parameters are of *integer* data type; however, if some of the input parameters are of *double* data type then real valued encoding can be used. A candidate solution that represents a test sequence consists of components where each component represents one input parameter. For example, a possible solution encoding of the path shown in Fig. 5 consists of six components of type *integer* $<C_0, C_1, C_2, C_3, C_4, C_5>$.

4. Empirical evaluation

4.1. Experimental design

In designing our experiment, we aimed to evaluate the effectiveness of the proposed approach which consists of the TP fitness metric and the fitness function for test sequence generation. In order to achieve this, there are three factors to be considered.

The first relates to the length of TPs used. A short TP is likely to have a low fitness metric value and be easy to trigger since it has few guards and operations. Therefore, we want TPs that are relatively long. Since the EFSMs had 15..31 transitions, we considered TPs of length ten to be sufficient to avoid the impact of this factor. The second factor

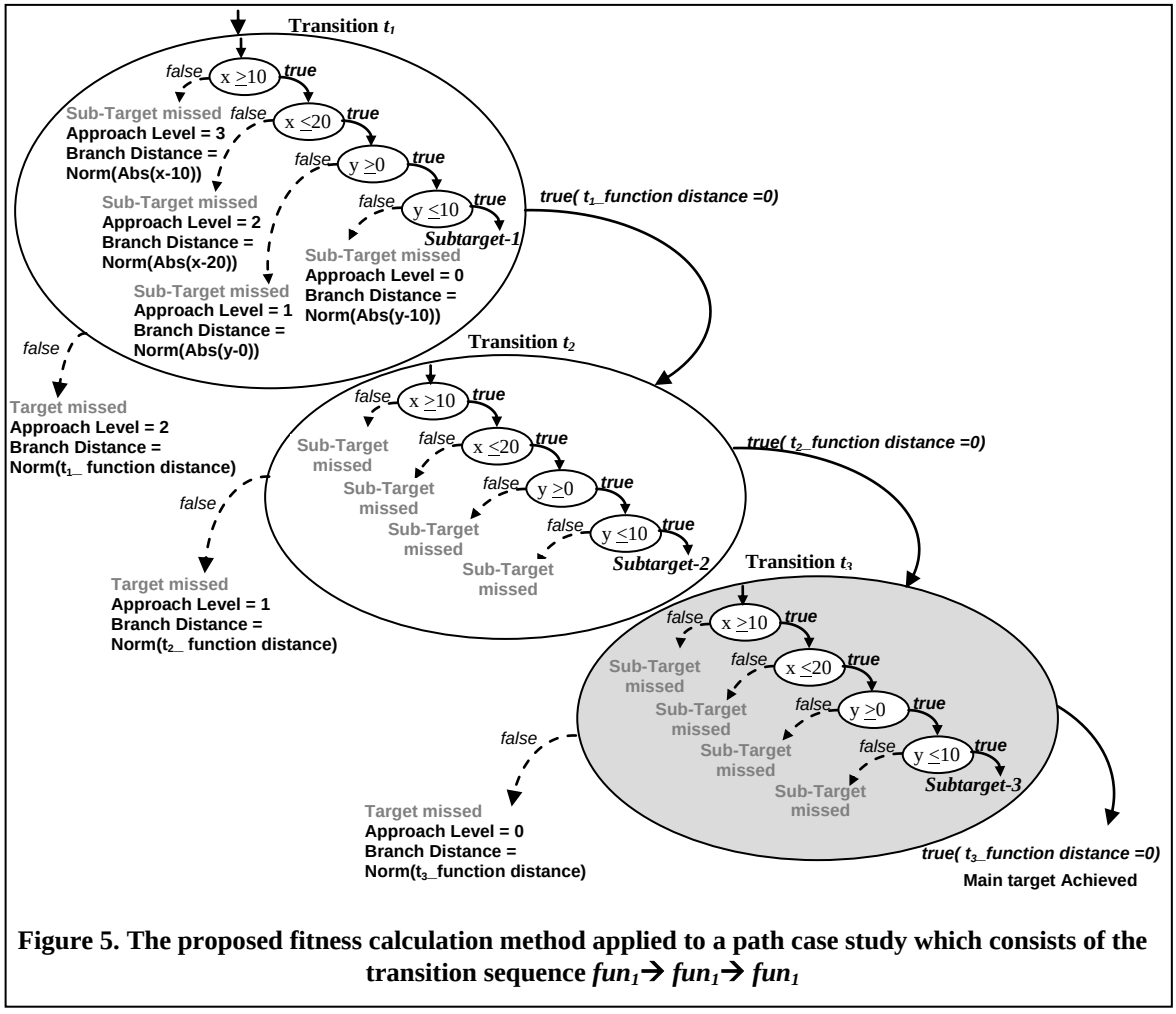


Figure 5. The proposed fitness calculation method applied to a path case study which consists of the transition sequence *fun*$_1$→ *fun*$_1$→ *fun*$_1$

relates to the number of input parameters required to trigger a TP. We might expect it to normally be harder to use search to trigger a TP that has many input parameters since it defines a larger and potentially more complex search space. We used TPs that required between 0 to 25 parameters.

The third factor is related to each EFSM; it may happen that the given EFSM is simple and so arbitrary generated TPs can be simple and easy to trigger. As a result we generated two sets of TPs for each EFSM, each set covering all of the transitions. One set was generated using search to produce TPs with low fitness and the other set was randomly generated. The first set was generated using an evolutionary algorithm that implemented the proposed TP fitness metric and these TPs are denoted by (TP-$_{c\text{-}ti\text{-}EA}$) where *c* denotes the EFSM number as it appears in Fig. 1, t_i denotes a transition that this TP is generated to cover, and *EA* means that this TP was generated by using an evolutionary algorithm that implements the proposed TP fitness metric. The alternative TPs were generated randomly and are denoted in a similar way (TP-$_{c\text{-}ti\text{-}RA}$), *RA* meaning that the TP was randomly generated. For the purpose of comparison, we measured the fitness metric value of each randomly generated TP after it was generated. For each TP, we applied three test sequence generation techniques:

1- ET-1: An ET technique that implements the proposed fitness function for test sequence generation;
2- ET-2: An ET technique that implements the fitness function proposed in [31]; and
3- Rand: A random test sequence generator.

All search techniques were implemented in the publicly available Genetic and Evolutionary Algorithm Toolbox GEATbx [32]. A detailed description of each of the GEATbx parameters used with EA search and ET-based techniques is beyond the scope of this paper. However, these parameters are fully explained at the GEATbx website [32] and we record the values used here to allow the experiment to be replicated.

An integer valued encoding was used to represent individuals and population size was 100 individuals. The selection method was linear-ranking with selective pressure set to 1.8. Discrete recombination was used to recombine individuals whereas mutate integer method was used for mutation. GEATbx allows the use of standard random approach by setting the recombination and mutation methods to 'recnone' and 'mutrandint' respectively.

For TP generation, each individual consisted of 10 integers which represented its transitions. The range of values allowed for each variable varied according to each subject EFSM as previously described in Subsection 3.1.3. The search terminated after 1000 generations or if a TP fitness metric value was zero.

For test sequence generation, each individual consists of 25 integers which represent the maximum number of input parameters. The range of values allowed for each variable was [0-1000]. Thus, the input domain used with each TP had the size of 1×10^{75} possible candidate solutions. Search terminated if the fitness value of zero was achieved or a maximum number of 1000 generations was reached. Finally, we repeated the search with each of the three test sequence generation techniques (ET-1, ET-2 and Rand) ten times for each subject TP.

4.2. Experimental results

An EA search that implemented the proposed TP fitness metric was applied to generate a transition coverage set of TPs for each subject EFSM shown in Fig. 1. Also, a random TP generator was applied to produce an alternative transition coverage set of TPs for each EFSM. Tables A-1, A-2, A-3, A-4 and A-5 (given in Appendix A) list the two sets of TPs for each subject EFSM. In these tables, each TP is given as a sequence of ten transitions together with the associated TP fitness metric and the number of input parameters.

For all the TPs that were generated by using the EA search (123 TPs), the fitness metric values were in the range [0..208]. This shows that the search found TPs that have a relatively low (better) fitness value.

The randomly generated TPs (123 TPs) can be divided into two groups: (1) TPs with fitness values that are in the range [0..910] and (2) TPs that their fitness metric values are $\geq 10^4$. In analyzing the results of the experiments we considered three cases:

1- **Case 1:** If a TP fitness metric value is $\geq 10^4$ then the TP is considered to be definitely infeasible. This is because the value of 10^4 means that the analysis has identified that the TP is infeasible TP (guards in opposition or operation and guard in opposition).
2- **Case 2:** The TP fitness metric value is in the range [0..9999] and one of the test sequence generation methods was able to trigger this TP and so it is feasible.

3- **Case 3:** The TP fitness metric is in the range [0..9999], however, none of the test sequence generation methods is able to trigger this TP. In this case, we manually determined whether the TP is feasible.

For TPs that were generated randomly, there are 76 TPs with fitness metric values $\geq 10^4$ and so, according to Case 1, are definitely infeasible. This shows that the considered EFSM models are non-trivial since almost 61.8 % of the randomly generated TPs are found to be infeasible. The remaining randomly generated TPs (47 TPs) have fitness metric values in the range [0..910], thus these TPs can belong to either Case 2 or Case 3. Similarly, the fitness metric values for TPs that were generated by using the EA search were in the range [0..208] and so these TPs belong to either Case 2 or Case 3.

The three test sequence generation techniques (Rand, ET-1 and ET-2) were applied to each generated TP. Since the complete set of results cannot fit in this paper, Tables 6, 7, 8, 9 and 10 report the performance of each test sequence generation method in terms of the average number of generations required by a particular technique in ten tries and whether the considered TP was successfully triggered.

For the rest of the randomly generated TPs (47 TPs), Tables 6, 7, 8, 9 and 10, show that 42 TPs are triggered by at least one of the test sequence generation techniques. Thus there were 5 additional potentially infeasible TPs: ($TP\text{-}_{4\text{-}3\text{-}RA}$, $TP\text{-}_{4\text{-}5\text{-}RA}$, $TP\text{-}_{4\text{-}10\text{-}RA}$, $TP\text{-}_{5\text{-}4\text{-}RA}$, $TP\text{-}_{5\text{-}12\text{-}RA}$) where the first three belong to the ATM EFSM and the last two belong to the Inres initiator EFSM. We manually inspected these and found that they are infeasible because of a guard that references a counter variable (a variable that counts the number of times that a transition has been repeated).

From Fig. 1-4, $TP\text{-}_{4\text{-}3\text{-}RA}$, $TP\text{-}_{4\text{-}5\text{-}RA}$ and $TP\text{-}_{4\text{-}10\text{-}RA}$ have the transition t_3 which requires the use of t_2 three times to increase the value of the counter variable *attempts*. The guard of t_3 requires the counter variable (*attempts* = 3). Also, both of $TP\text{-}_{5\text{-}4\text{-}RA}$ and $TP\text{-}_{5\text{-}12\text{-}RA}$ have a similar problem with the variable *counter*.

For TPs generated by EA search, Tables 6, 7, 8, 9 and 10 show that 119 TPs of the 123 TPs were triggered by at least one of the test sequence generation techniques. This leaves us with 4 TPs that were not triggered by any test sequence generation techniques. These TPs are ($TP\text{-}_{4\text{-}3\text{-}EA}$, $TP\text{-}_{5\text{-}4\text{-}EA}$, $TP\text{-}_{5\text{-}9\text{-}EA}$, $TP\text{-}_{5\text{-}11\text{-}EA}$) where the first TP belongs to ATM EFSM and the other three TPs belong to Inres Initiator EFSM. A manual inspection of these TPs showed that they are infeasible due to a guard over a counter variable. For example, $TP\text{-}_{5\text{-}4\text{-}EA}$ has the transition t_4 with a guard that checks whether (*counter*$\geq$4) and so the transition t_3 must occur four times before t_4.

From these results, we can state that the random path generator was able to produce FTPs with an accuracy rate approximately 34.15 %. In contrast, the proposed fitness metric successfully guided the EA search to generate a set of FTPs with an accuracy rate approximately 96.75 %. The remaining 3.25 % of infeasible paths belonged to the case where TPs have a guard over a counter variable. For example, if we consider the $TP\text{-}_{5\text{-}4\text{-}EA}$ in the Inres initiator EFSM, the current fitness metric algorithm will always penalize the sequence t_3, t_3, t_4 more than the sequence t_3, t_4 since t_3 has an operation of type op^{vv}, a guard of type g^{vc} and t_3 is affected by t_3. Thus, every time t_3 is followed by t_3 there is a penalty to be added. Thus, if we minimize the fitness then it is unlikely to get t_3 to occur more than once. This description also applies to all the infeasible TPs with low fitness. We are currently investigating this problem and see this as an important area of future work.

Table 6. Results of three test data generation techniques on the in-flight EFSM subject TPs

Path ID	Path Fitness	Taken(Rand)	Avg. Gen. Rand	Taken(ET-1)	Avg. Gen. ET-1	Taken(ET-2)	Avg. Gen. ET-2
$TP_{1\text{-}1\text{-}EA}$	72	No	1000	Yes	48.2	Yes	59
$TP_{1\text{-}1\text{-}RA}$	288	No	1000	Yes	954.7	No	1000
$TP_{1\text{-}2\text{-}EA}$	180	No	1000	Yes	710.7	No	1000
$TP_{1\text{-}2\text{-}RA}$	234	No	1000	Yes	803.2	No	1000
$TP_{1\text{-}3\text{-}EA}$	72	No	1000	Yes	117.3	Yes	89.7
$TP_{1\text{-}3\text{-}RA}$	40198	No	1000	No	1000	No	1000
$TP_{1\text{-}4\text{-}EA}$	72	No	1000	Yes	57.3	Yes	42.4
$TP_{1\text{-}4\text{-}RA}$	41246	No	1000	No	1000	No	1000
$TP_{1\text{-}5\text{-}EA}$	72	Yes	666.7	Yes	39.3	Yes	42.8
$TP_{1\text{-}5\text{-}RA}$	41246	No	1000	No	1000	No	1000
$TP_{1\text{-}6\text{-}EA}$	72	Yes	617.1	Yes	47.1	Yes	31.3
$TP_{1\text{-}6\text{-}RA}$	30090	No	1000	No	1000	No	1000
$TP_{1\text{-}7\text{-}EA}$	126	No	1000	Yes	288.3	No	1000
$TP_{1\text{-}7\text{-}RA}$	234	No	1000	Yes	809.2	No	1000
$TP_{1\text{-}8\text{-}EA}$	72	No	1000	Yes	77.4	Yes	50.4
$TP_{1\text{-}8\text{-}RA}$	40042	No	1000	No	1000	No	1000
$TP_{1\text{-}9\text{-}EA}$	72	No	1000	Yes	102.5	Yes	87.8
$TP_{1\text{-}9\text{-}RA}$	30108	No	1000	No	1000	No	1000
$TP_{1\text{-}10\text{-}EA}$	72	Yes	948.4	Yes	155.3	Yes	65
$TP_{1\text{-}10\text{-}RA}$	30204	No	1000	No	1000	No	1000
$TP_{1\text{-}11\text{-}EA}$	72	No	1000	Yes	81.2	Yes	44.5
$TP_{1\text{-}11\text{-}RA}$	40072	No	1000	No	1000	No	1000
$TP_{1\text{-}12\text{-}EA}$	72	No	1000	Yes	79	Yes	53.2
$TP_{1\text{-}12\text{-}RA}$	30180	No	1000	No	1000	No	1000
$TP_{1\text{-}13\text{-}EA}$	72	Yes	299	Yes	31.4	Yes	32.2
$TP_{1\text{-}13\text{-}RA}$	30228	No	1000	No	1000	No	1000
$TP_{1\text{-}14\text{-}EA}$	126	No	1000	Yes	230.8	No	1000
$TP_{1\text{-}14\text{-}RA}$	40180	No	1000	No	1000	No	1000
$TP_{1\text{-}15\text{-}EA}$	78	Yes	656	Yes	39.5	Yes	23.2
$TP_{1\text{-}15\text{-}RA}$	40102	No	1000	No	1000	No	1000
$TP_{1\text{-}16\text{-}EA}$	126	No	1000	Yes	213.1	No	1000
$TP_{1\text{-}16\text{-}RA}$	30108	No	1000	No	1000	No	1000
$TP_{1\text{-}17\text{-}EA}$	126	No	1000	Yes	157.5	No	1000
$TP_{1\text{-}17\text{-}RA}$	30222	No	1000	No	1000	No	1000
$TP_{1\text{-}18\text{-}EA}$	126	No	1000	Yes	147.2	No	1000
$TP_{1\text{-}18\text{-}RA}$	30204	No	1000	No	1000	No	1000
$TP_{1\text{-}19\text{-}EA}$	126	No	1000	Yes	518.9	No	1000
$TP_{1\text{-}19\text{-}RA}$	30186	No	1000	No	1000	No	1000
$TP_{1\text{-}20\text{-}EA}$	126	No	1000	Yes	248.2	No	1000
$TP_{1\text{-}20\text{-}RA}$	40198	No	1000	No	1000	No	1000
$TP_{1\text{-}21\text{-}EA}$	126	No	1000	Yes	172.9	No	1000
$TP_{1\text{-}21\text{-}RA}$	40276	No	1000	No	1000	No	1000
$TP_{1\text{-}22\text{-}EA}$	72	Yes	56.2	Yes	13.3	Yes	19.1
$TP_{1\text{-}22\text{-}RA}$	40036	No	1000	No	1000	No	1000
$TP_{1\text{-}23\text{-}EA}$	78	Yes	2.9	Yes	6.4	Yes	5.5
$TP_{1\text{-}23\text{-}RA}$	30132	No	1000	No	1000	No	1000
$TP_{1\text{-}24\text{-}EA}$	72	Yes	249.1	Yes	20.6	Yes	22.9
$TP_{1\text{-}24\text{-}RA}$	30162	No	1000	No	1000	No	1000
$TP_{1\text{-}25\text{-}EA}$	78	Yes	3.2	Yes	5.5	Yes	5
$TP_{1\text{-}25\text{-}RA}$	40204	No	1000	No	1000	No	1000
$TP_{1\text{-}26\text{-}EA}$	72	Yes	1	Yes	1	Yes	1.1
$TP_{1\text{-}26\text{-}RA}$	30168	No	1000	No	1000	No	1000
$TP_{1\text{-}27\text{-}EA}$	78	Yes	1	Yes	1	Yes	1
$TP_{1\text{-}27\text{-}RA}$	30222	No	1000	No	1000	No	1000
$TP_{1\text{-}28\text{-}EA}$	72	Yes	7.5	Yes	4	Yes	6
$TP_{1\text{-}28\text{-}RA}$	30180	No	1000	No	1000	No	1000
$TP_{1\text{-}29\text{-}EA}$	78	Yes	1	Yes	1	Yes	1
$TP_{1\text{-}29\text{-}RA}$	20162	No	1000	No	1000	No	1000
$TP_{1\text{-}30\text{-}EA}$	126	No	1000	Yes	192.2	No	1000
$TP_{1\text{-}30\text{-}RA}$	40198	No	1000	No	1000	No	1000
$TP_{1\text{-}31\text{-}EA}$	126	No	1000	Yes	187.4	No	1000
$TP_{1\text{-}31\text{-}RA}$	30168	No	1000	No	1000	No	1000

Total	**FTPs**	**FTPs Taken by Random**	**FTPs Taken by ET-1**	**FTPs Taken by ET-2**
31 $TPs\text{-}_{EA}$	31	13	31	20
31 $TPs\text{-}_{RA}$	3	0	3	0

Table 7. Results of three test sequence generation techniques on the Class II EFSM subject TPs

Path ID	Path Fitness	Taken(Rand)	Avg. Gen. Rand	Taken(ET-1)	Avg. Gen. ET-1	Taken (ET-2)	Avg. Gen. ET-2
$TP_{2\text{-}0\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}0\text{-}RA}$	7	Yes	897.6	Yes	22.6	Yes	115.8
$TP_{2\text{-}1\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}1\text{-}RA}$	10070	No	1000	No	1000	No	1000
$TP_{2\text{-}2\text{-}EA}$	4	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}2\text{-}RA}$	10068	No	1000	No	1000	No	1000
$TP_{2\text{-}3\text{-}EA}$	6	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}3\text{-}RA}$	82	Yes	20.3	Yes	4.8	Yes	8.3
$TP_{2\text{-}4\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}4\text{-}RA}$	10070	No	1000	No	1000	No	1000
$TP_{2\text{-}5\text{-}EA}$	4	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}5\text{-}RA}$	34	Yes	18.9	Yes	6.2	Yes	5.9
$TP_{2\text{-}6\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}6\text{-}RA}$	4	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}7\text{-}EA}$	28	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}7\text{-}RA}$	910	No	1000	Yes	332.3	No	1000
$TP_{2\text{-}8\text{-}EA}$	40	Yes	16.4	Yes	7.3	Yes	8.1
$TP_{2\text{-}8\text{-}RA}$	10040	No	1000	No	1000	No	1000
$TP_{2\text{-}9\text{-}EA}$	10	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}9\text{-}RA}$	10496	No	1000	No	1000	No	1000
$TP_{2\text{-}10\text{-}EA}$	4	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}10\text{-}RA}$	10358	No	1000	No	1000	No	1000
$TP_{2\text{-}11\text{-}EA}$	40	Yes	718.7	Yes	18.7	Yes	448.4
$TP_{2\text{-}11\text{-}RA}$	160	No	1000	Yes	251.4	Yes	642
$TP_{2\text{-}12\text{-}EA}$	34	Yes	27.6	Yes	8.2	Yes	6.4
$TP_{2\text{-}12\text{-}RA}$	772	No	1000	Yes	251.4	No	1000
$TP_{2\text{-}13\text{-}EA}$	46	Yes	6.7	Yes	7.6	Yes	5.3
$TP_{2\text{-}13\text{-}RA}$	834	No	1000	Yes	338.5	No	1000
$TP_{2\text{-}14\text{-}EA}$	40	Yes	3	Yes	2.1	Yes	2.3
$TP_{2\text{-}14\text{-}RA}$	306	Yes	748.4	Yes	16.4	Yes	18.6
$TP_{2\text{-}15\text{-}EA}$	28	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}15\text{-}RA}$	10058	No	1000	No	1000	No	1000
$TP_{2\text{-}16\text{-}EA}$	4	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}16\text{-}RA}$	68	Yes	86.8	Yes	7.4	Yes	7.1
$TP_{2\text{-}17\text{-}EA}$	4	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}17\text{-}RA}$	10092	No	1000	No	1000	No	1000
$TP_{2\text{-}18\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}18\text{-}RA}$	4	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}19\text{-}EA}$	4	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}19\text{-}RA}$	10	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}20\text{-}EA}$	4	Yes	1	Yes	1	Yes	1
$TP_{2\text{-}20\text{-}RA}$	16	Yes	1	Yes	1	Yes	1
Total	**FTPs**	**FTPs Taken by Random**		**FTPs Taken by ET-1**		**FTPs Taken by ET-2**	
21 $TPs\text{-}_{EA}$	21	21		21		21	
21 $TPs\text{-}_{RA}$	13	9		13		10	

Table 11 provides a summary of the results in terms of the number of generated TPs for each case, TPs generation method (EA or RA), how many TPs of each set were feasible and how many FTPs from each set were triggered by each test sequence generation technique. This table also reports the success rate of each generation method (EA and RA) and for each test sequence generation method (ET-1, ET-2 and Rand).

For the EA generation method that utilizes the TP fitness metric, this was able to generate FTPs with a success rate approximately 96.75 %. Compared to Rand TP generation, Rand was able to generate FTPs with a success rate approximately 34.15 %. The Rand performance shows that generating FTPs form the considered EFSMs is not an easy task.

For TPs that were generated randomly, we can state that the proposed ET-1 technique was able to trigger all the randomly generated FTPs (42 FTPs) and so it has the success rate of 100 %. The ET-2 technique was able to exercise 31 FTPs with a success rate

Table 8. Results of three test sequence generation techniques on the Lift EFSM subject TPs

Path ID	Path Fitness	Taken(Rand)	Avg. Gen. Rand	Taken(ET-1)	Avg. Gen. ET-1	Taken (ET-2)	Avg. Gen. ET-2
$TP_{3\text{-}0\text{-}EA}$	72	No	1000	Yes	743	Yes	593.2
$TP_{3\text{-}0\text{-}RA}$	40214	No	1000	No	1000	No	1000
$TP_{3\text{-}1\text{-}EA}$	72	No	1000	Yes	679	Yes	664.9
$TP_{3\text{-}1\text{-}RA}$	30242	No	1000	No	1000	No	1000
$TP_{3\text{-}2\text{-}EA}$	72	No	1000	Yes	737	Yes	536.9
$TP_{3\text{-}2\text{-}RA}$	40214	No	1000	No	1000	No	1000
$TP_{3\text{-}3\text{-}EA}$	110	No	1000	Yes	891.9	No	1000
$TP_{3\text{-}3\text{-}RA}$	40184	No	1000	No	1000	No	1000
$TP_{3\text{-}4\text{-}EA}$	130	No	1000	Yes	969.8	No	1000
$TP_{3\text{-}4\text{-}RA}$	30204	No	1000	No	1000	No	1000
$TP_{3\text{-}5\text{-}EA}$	92	No	1000	Yes	821.3	Yes	998.3
$TP_{3\text{-}5\text{-}RA}$	30168	No	1000	No	1000	No	1000
$TP_{3\text{-}6\text{-}EA}$	112	No	1000	Yes	831	No	1000
$TP_{3\text{-}6\text{-}RA}$	30196	No	1000	No	1000	No	1000
$TP_{3\text{-}7\text{-}EA}$	92	No	1000	Yes	847.8	Yes	945.7
$TP_{3\text{-}7\text{-}RA}$	30206	No	1000	No	1000	No	1000
$TP_{3\text{-}8\text{-}EA}$	92	No	1000	Yes	904.4	No	1000
$TP_{3\text{-}8\text{-}RA}$	30232	No	1000	No	1000	No	1000
$TP_{3\text{-}9\text{-}EA}$	152	No	1000	Yes	965.4	No	1000
$TP_{3\text{-}9\text{-}RA}$	20252	No	1000	No	1000	No	1000
$TP_{3\text{-}10\text{-}EA}$	130	No	1000	Yes	924.3	No	1000
$TP_{3\text{-}10\text{-}RA}$	40214	No	1000	No	1000	No	1000
$TP_{3\text{-}11\text{-}EA}$	112	No	1000	Yes	926.6	No	1000
$TP_{3\text{-}11\text{-}RA}$	30192	No	1000	No	1000	No	1000
$TP_{3\text{-}12\text{-}EA}$	132	No	1000	Yes	960.8	No	1000
$TP_{3\text{-}12\text{-}RA}$	40222	No	1000	No	1000	No	1000
$TP_{3\text{-}13\text{-}EA}$	92	No	1000	Yes	896.2	Yes	947.7
$TP_{3\text{-}13\text{-}RA}$	40210	No	1000	No	1000	No	1000
$TP_{3\text{-}14\text{-}EA}$	92	No	1000	Yes	892.2	No	1000
$TP_{3\text{-}14\text{-}RA}$	30210	No	1000	No	1000	No	1000
$TP_{3\text{-}15\text{-}EA}$	152	No	1000	Yes	920.1	No	1000
$TP_{3\text{-}15\text{-}RA}$	40214	No	1000	No	1000	No	1000
$TP_{3\text{-}16\text{-}EA}$	110	No	1000	Yes	793.6	No	1000
$TP_{3\text{-}16\text{-}RA}$	10232	No	1000	No	1000	No	1000
$TP_{3\text{-}17\text{-}EA}$	92	No	1000	Yes	818.4	No	1000
$TP_{3\text{-}17\text{-}RA}$	40182	No	1000	No	1000	No	1000
$TP_{3\text{-}18\text{-}EA}$	112	No	1000	Yes	969.9	No	1000
$TP_{3\text{-}18\text{-}RA}$	30196	No	1000	No	1000	No	1000
$TP_{3\text{-}19\text{-}EA}$	114	No	1000	Yes	848	No	1000
$TP_{3\text{-}19\text{-}RA}$	30210	No	1000	No	1000	No	1000
$TP_{3\text{-}20\text{-}EA}$	114	No	1000	Yes	890.7	No	1000
$TP_{3\text{-}20\text{-}RA}$	20240	No	1000	No	1000	No	1000
$TP_{3\text{-}21\text{-}EA}$	114	No	1000	Yes	837.3	No	1000
$TP_{3\text{-}21\text{-}RA}$	40214	No	1000	No	1000	No	1000
$TP_{3\text{-}22\text{-}EA}$	156	No	1000	Yes	922.3	No	1000
$TP_{3\text{-}22\text{-}RA}$	30194	No	1000	No	1000	No	1000
$TP_{3\text{-}23\text{-}EA}$	134	No	1000	Yes	873.7	No	1000
$TP_{3\text{-}23\text{-}RA}$	198	No	1000	Yes	944.9	No	1000
$TP_{3\text{-}24\text{-}EA}$	114	No	1000	Yes	797	No	1000
$TP_{3\text{-}24\text{-}RA}$	20236	No	1000	No	1000	No	1000

Total	**FTPs**	**FTPs Taken by Random**	**FTPs Taken by ET-1**	**FTPs Taken by ET-2**
25 $TPs\text{-}_{EA}$	25	0	25	6
25 $TPs\text{-}_{RA}$	1	0	1	0

approximately 73.81 % whereas the Rand technique triggered only 14 FTPs and the success rate was approximately 33.34 %.

For TPs that were generated by the EA search, the proposed test sequence generation technique ET-1 was able to trigger all the FTPs with a success rate of 100%. In contrast, the ET-2 technique had a success rate approximately 73.95 % whereas the Rand technique exhibited the worst performance with a success rate approximately 46.22 %.

If we consider only the FTPs that were generated randomly and by the EA search, we can state that the proposed ET-1 technique was found to be efficient since it exercised all

Table 9. Results of three test sequence generation techniques on the ATM EFSM subject TPs

Path ID	Path Fitness	Taken(Rand)	Avg. Gen. Rand	Taken(ET-1)	Avg. Gen. ET-1	Taken(ET-2)	Avg. Gen. ET-2
$TP_{4\text{-}1\text{-}EA}$	36	No	1000	Yes	29.4	Yes	27.8
$TP_{4\text{-}1\text{-}RA}$	120	No	1000	Yes	597.5	Yes	504
$TP_{4\text{-}2\text{-}EA}$	54	No	1000	Yes	55.2	Yes	144.6
$TP_{4\text{-}2\text{-}RA}$	10238	No	1000	No	1000	No	1000
$TP_{4\text{-}3\text{-}EA}$	208	No	1000	No	1000	No	1000
$TP_{4\text{-}3\text{-}RA}$	380	No	1000	No	1000	No	1000
$TP_{4\text{-}4\text{-}EA}$	36	Yes	911	Yes	34.7	Yes	27.7
$TP_{4\text{-}4\text{-}RA}$	104	No	1000	Yes	647.8	Yes	466.7
$TP_{4\text{-}5\text{-}EA}$	36	Yes	797	Yes	20.7	Yes	26
$TP_{4\text{-}5\text{-}RA}$	414	No	1000	No	1000	No	1000
$TP_{4\text{-}6\text{-}EA}$	36	Yes	939.9	Yes	25.3	Yes	34.2
$TP_{4\text{-}6\text{-}RA}$	10222	No	1000	No	1000	No	1000
$TP_{4\text{-}7\text{-}EA}$	36	Yes	759.6	Yes	25.8	Yes	23
$TP_{4\text{-}7\text{-}RA}$	20088	No	1000	No	1000	No	1000
$TP_{4\text{-}8\text{-}EA}$	36	Yes	953.6	Yes	26.3	Yes	19.5
$TP_{4\text{-}8\text{-}RA}$	60	No	1000	Yes	140.3	Yes	207.7
$TP_{4\text{-}9\text{-}EA}$	36	Yes	927.2	Yes	30.2	Yes	29
$TP_{4\text{-}9\text{-}RA}$	60	No	1000	Yes	121.1	Yes	99.5
$TP_{4\text{-}10\text{-}EA}$	36	Yes	980.7	Yes	33.1	Yes	36.5
$TP_{4\text{-}10\text{-}RA}$	232	No	1000	No	1000	No	1000
$TP_{4\text{-}11\text{-}EA}$	60	No	1000	Yes	206.2	Yes	114.4
$TP_{4\text{-}11\text{-}RA}$	20072	No	1000	No	1000	No	1000
$TP_{4\text{-}12\text{-}EA}$	60	No	1000	Yes	105.7	Yes	137.7
$TP_{4\text{-}12\text{-}RA}$	104	No	1000	Yes	530	Yes	421.8
$TP_{4\text{-}13\text{-}EA}$	56	No	1000	Yes	76.8	Yes	55.8
$TP_{4\text{-}13\text{-}RA}$	270	No	1000	Yes	374.8	Yes	162.7
$TP_{4\text{-}14\text{-}EA}$	76	No	1000	Yes	110.9	Yes	82.4
$TP_{4\text{-}14\text{-}RA}$	242	No	1000	Yes	138.4	Yes	262.4
$TP_{4\text{-}15\text{-}EA}$	56	No	1000	Yes	72.3	Yes	60.7
$TP_{4\text{-}15\text{-}RA}$	100	No	1000	Yes	309.8	Yes	475.5
$TP_{4\text{-}16\text{-}EA}$	56	No	1000	Yes	72.9	Yes	57.8
$TP_{4\text{-}16\text{-}RA}$	124	No	1000	Yes	68.2	Yes	82.7
$TP_{4\text{-}17\text{-}EA}$	72	No	1000	Yes	87.9	Yes	166.1
$TP_{4\text{-}17\text{-}RA}$	10102	No	1000	No	1000	No	1000
$TP_{4\text{-}18\text{-}EA}$	56	No	1000	Yes	59.1	Yes	73.9
$TP_{4\text{-}18\text{-}RA}$	80	No	1000	Yes	337.5	Yes	401.7
$TP_{4\text{-}19\text{-}EA}$	60	No	1000	Yes	272.2	Yes	186.4
$TP_{4\text{-}19\text{-}RA}$	108	No	1000	Yes	669	Yes	557.7
$TP_{4\text{-}20\text{-}EA}$	60	No	1000	Yes	115.4	Yes	199.6
$TP_{4\text{-}20\text{-}RA}$	100	No	1000	Yes	592.9	Yes	364.3
$TP_{4\text{-}21\text{-}EA}$	56	No	1000	Yes	62.2	Yes	53.1
$TP_{4\text{-}21\text{-}RA}$	84	No	1000	Yes	507.7	Yes	535.5
$TP_{4\text{-}22\text{-}EA}$	56	No	1000	Yes	44.2	Yes	60.9
$TP_{4\text{-}22\text{-}RA}$	108	No	1000	Yes	800.8	Yes	674.5
$TP_{4\text{-}23\text{-}EA}$	48	No	1000	Yes	49.2	Yes	59.2
$TP_{4\text{-}23\text{-}RA}$	20066	No	1000	No	1000	No	1000
$TP_{4\text{-}24\text{-}EA}$	72	No	1000	Yes	95.8	Yes	81.7
$TP_{4\text{-}24\text{-}RA}$	20208	No	1000	No	1000	No	1000
$TP_{4\text{-}25\text{-}EA}$	36	Yes	817.6	Yes	25.6	Yes	21.8
$TP_{4\text{-}25\text{-}RA}$	116	No	1000	Yes	88.3	Yes	963.9
$TP_{4\text{-}26\text{-}EA}$	36	No	1000	Yes	24.8	Yes	24.3
$TP_{4\text{-}26\text{-}RA}$	230	No	1000	Yes	88.3	Yes	90.4
$TP_{4\text{-}27\text{-}EA}$	98	No	1000	Yes	162.8	Yes	914.8
$TP_{4\text{-}27\text{-}RA}$	264	No	1000	Yes	832	No	1000
$TP_{4\text{-}28\text{-}EA}$	98	No	1000	Yes	329.7	Yes	953.8
$TP_{4\text{-}28\text{-}RA}$	196	No	1000	Yes	731	No	1000
$TP_{4\text{-}29\text{-}EA}$	98	No	1000	Yes	334.7	No	1000
$TP_{4\text{-}29\text{-}RA}$	146	No	1000	Yes	647	No	1000
$TP_{4\text{-}30\text{-}EA}$	98	No	1000	Yes	226.9	Yes	938.1
$TP_{4\text{-}30\text{-}RA}$	254	No	1000	Yes	805.7	No	1000
Total	**FTPs**	**FTPs Taken by Random**		**FTPs Taken by ET-1**		**FTPs Taken by ET-2**	
30 $TPs\text{-}_{EA}$	29	8		29		28	
30 $TPs\text{-}_{RA}$	20	0		20		16	

Table 10. Results of three test sequence generation techniques on the Inres EFSM subject TPs

Path ID	Path Fitness	Taken(Rand)	Avg. Gen. Rand	Taken(ET-1)	Avg. Gen. ET-1	Taken(ET-2)	Avg. Gen. ET-2
$TP_{5\text{-}0\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}0\text{-}RA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}1\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}1\text{-}RA}$	20000	No	1000	No	1000	No	1000
$TP_{5\text{-}2\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}2\text{-}RA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}3\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}3\text{-}RA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}4\text{-}EA}$	136	No	1000	No	1000	No	1000
$TP_{5\text{-}4\text{-}RA}$	136	No	1000	No	1000	No	1000
$TP_{5\text{-}5\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}5\text{-}RA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}6\text{-}EA}$	48	No	1000	Yes	29.9	Yes	23.6
$TP_{5\text{-}6\text{-}RA}$	10000	No	1000	No	1000	No	1000
$TP_{5\text{-}7\text{-}EA}$	24	Yes	13.5	Yes	8.5	Yes	10
$TP_{5\text{-}7\text{-}RA}$	20024	No	1000	No	1000	No	1000
$TP_{5\text{-}8\text{-}EA}$	6	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}8\text{-}RA}$	10160	No	1000	No	1000	No	1000
$TP_{5\text{-}9\text{-}EA}$	142	No	1000	No	1000	No	1000
$TP_{5\text{-}9\text{-}RA}$	10006	No	1000	No	1000	No	1000
$TP_{5\text{-}10\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}10\text{-}RA}$	10000	No	1000	No	1000	No	1000
$TP_{5\text{-}11\text{-}EA}$	136	No	1000	No	1000	No	1000
$TP_{5\text{-}11\text{-}RA}$	10000	No	1000	No	1000	No	1000
$TP_{5\text{-}12\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}12\text{-}RA}$	324	No	1000	No	1000	No	1000
$TP_{5\text{-}13\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}13\text{-}RA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}14\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}14\text{-}RA}$	20000	No	1000	No	1000	No	1000
$TP_{5\text{-}15\text{-}EA}$	0	Yes	1	Yes	1	Yes	1
$TP_{5\text{-}15\text{-}RA}$	10000	No	1000	No	1000	No	1000
Total	**FTPs**	**FTPs Taken by Random**		**FTPs Taken by ET-1**		**FTPs Taken by ET-2**	
16 $TPs_{\text{-}EA}$	13	13		13		13	
16 $TPs_{\text{-}RA}$	5	5		5		5	

the FTPs, however, this is not the case with both ET-2 and Rand techniques where the rate of successfully triggering the generated FTPs were approximately 73.92 % and 42.86 % respectively.

Fig. 6 has three graphs that show the fitness metric value of each TP generated from the five EFSMs by using the EA search (123 TPs). In this Figure, each graph plots TP fitness against the average number of generations required to trigger the TP averaged over ten tries. Since the ET-1 technique triggered all the generated FTPs, the third graph can be considered as a baseline to compare with.

The first graph shows that all the FTPs that were triggered by Rand (55 FTPs) were associated with a fitness metric value that did not exceed 78. However, there are still some FTPs (28 FTPs) with TP fitness values less than 78 that were not triggered randomly.

The second graph shows that the ET-2 technique triggered all the FTPs with TP fitness values in [0...98] but failed for FTPs with TP fitness metric greater than 98. We can see that ET-2 outperformed that Rand technique performance but exhibited worse performance than the proposed ET-1 technique.

The third graph clearly illustrates that ET-1 was most effective since all the generated FTPs were successfully triggered. The third graph seems to show a trend between an FTP fitness and how easily this FTP can be triggered in terms of the required number of generations. There is also some evidence of this trend in the first and the second graphs; when the FTP fitness metric did not exceed 75, these FTPs appeared to require fewer

Table 11. A summary of the experimental results

EFSM	TPs by	TPs	FTPs	FTPs taken (Rand)	FTPs taken (ET-1)	FTPs taken (ET-2)
in-flight	EA	31	31	13	31	20
	RA	31	3	0	3	0
Class II	EA	21	21	21	21	21
	RA	21	13	9	13	10
Lift	EA	25	25	0	25	6
	RA	25	1	0	1	0
ATM	EA	30	29	8	29	28
	RA	30	20	0	20	16
Inres	EA	16	13	13	13	13
	RA	16	5	5	5	5
Totals	EA	123	119	55	119	88
	RA	123	42	14	42	31
Success rate	EA	123	≈96.75 %	≈46.22	= 100 %	≈73.95 %
	RA	123	≈34.15 %	≈33.34	= 100 %	≈73.81 %
	EA&RA	246	161	≈42.86 %	= 100 %	≈73.92 %

generations to be triggered. While this trend is relatively clear in the third graph, we cannot consider this as strong evidence but merely worth noting at this point.

The experimental results provide strong evidence that the proposed fitness metric algorithm can effectively guide an EA search towards TPs that are likely to be feasible. Furthermore, the experiments suggest that the proposed test sequence generation approach (ET-1) is effective in finding test data to trigger an FTP.

4.3. Threats to validity

In this subsection we discuss the potential threats to the validity of our study. Threats to external validity are the conditions that restrict our ability to generalize our results. The threats to external validity of this study are related to: First, the subject EFSM models, though nontrivial, they have relatively few states and transitions, and larger EFSM model might be helpful to derive more general conclusions about the validity of the proposed approach. Nevertheless, this threat has been limited by using two synthesized EFSMs and another three EFSMs that are commonly used in evaluating other EFSM testing techniques. Second, a specific path length was chosen. Paths with more transitions may be subject to a different cost e.g., test sequence generation might require more generations in order to traverse a path. Additional experiments with larger EFSMs could investigate the impact of using longer paths.

5. Related work

Many test generation approaches for systems modeled as EFSMs appear in the literature [6-9, 16, 29, 31, 33-38]. An approach to generate a unified test sequence (UTS) for EFSM models is presented in [33] based on two techniques: one to test the control part (FSM) and the other to test the data part by using data flow analysis technique. The resultant UTS is then checked for executability by using a *constraint satisfaction* method. However, some assumptions about the EFSM model i.e. the existence of *self-loop influencing* (a loop that modifies a global predicate variable) restrict its applicability.

Generating test sequence from EFSMs by employing functional program testing was studied in [38]. The approach converted the specification written in Estelle [39] into a simpler form in order to construct control and data flow graphs to be used in test sequence derivation. However, the approach restricted the use of common code constructs such as

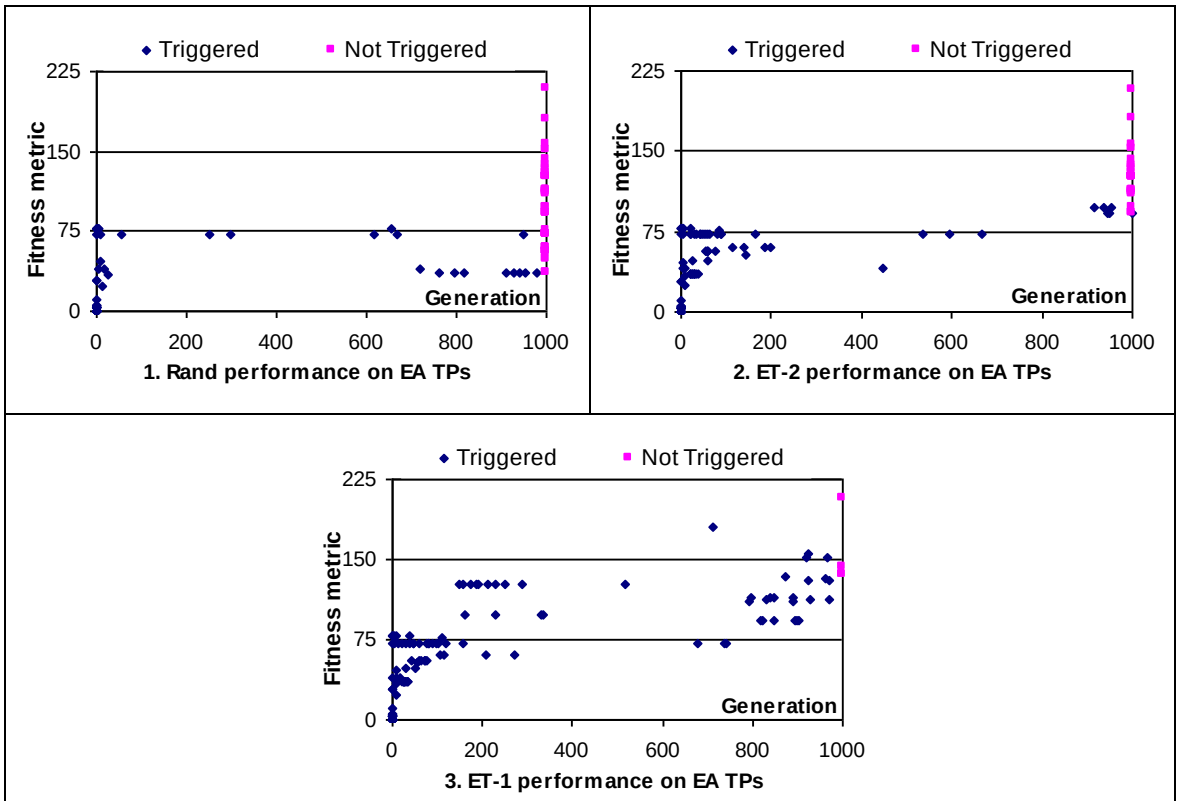


Figure 6. The performance of the three test sequence generation techniques on EA TPs

functions calls and conditional statements. Also, it did not describe how test sequences can be generated.

Other methods that test from an EFSM using FSM-based test techniques appear in [34-36] but these require an EFSM to be converted into an FSM. There are two main approaches, the first being to expand the data in the EFSM. However, the number of states in the resultant FSM can easily become prohibitively large [14]. The alternative is to abstract the data from the EFSM to produce an FSM. This approach has two limitations that motivate the work described in this paper: there is a need to produce test sequences to trigger the paths chosen and these paths are not necessarily feasible in the original EFSM.

A technique for generating unique state identification sequences for EFSM models is presented in [16]. The technique is based on computing a new type of state identification for each state called *context independent unique sequence (CIUS)*. This requires that all the paths that start from any state be context independent. That is, all the guards included in any path can be interpreted symbolically and each state must have a CIUS. This appears to limit the applicability of the approach. The authors did not consider the problem of generating test sequences for the generated paths and here the approach proposed in this paper might help.

An approach which employs software data flow testing to derive a test sequence from EFSM models is presented in [9]. The selection of each test case depends on identifying all the associations between each output and all the inputs that affects that output. However, as stated in [40], the approach might not always provide the intended coverage.

Approaches that study the path feasibility problem are introduced in [6, 7]. In [7], a method is given to convert EFSMs into EFSMs in which all paths are feasible but this requires guards and operations to be linear. In [6], the infeasible path problem was overcome through two steps. First, the *SDL* (*Specification and Description Language*) model is rewritten in order to derive a *normal form-EFSM* (*NF-EFSM*). Second, the resultant *NF-EFSM* is extended to *Expanded-EFSM* (*EEFSM*) in order to aid testability. As a result, all the paths presented in the output *EEFSM* are feasible. However, these two approaches [6, 7] did not tackle the problem of generating test sequences that trigger the resultant paths and here the approach proposed in this paper could be used.

Approaches that utilize search algorithms to test from EFSMs are introduced in [29, 31]. The approach proposed in [31] describes a fitness calculation method to find a test sequence for a path. The considered fitness function applies Tracey et al. [27] technique to each transition in a path. Path fitness is defined by considering each function in the path as a critical node. The limitation of this study is the assumption that each function does not have an internal path i.e. nested IF statements for which Tracey et al. [27] approach does not always provide a sufficient guidance as argued in [2]. Furthermore, the work did not consider the problem of choosing a path that is likely to be feasible. In [29], a GA approach to generate FTPs from EFSM model was presented. This is the only previous work that utilizes a GA to generate FTPs from a given EFSM. The approach evaluated the feasibility of a given TP according to the number and the types of guards found in that TP. However, the dependences between transitions in a path were not considered.

6. Conclusion

Although the EFSM is a powerful model and has been widely applied, testing from this model is a challenging task for two reasons: some paths may be infeasible and it may be difficult to produce a test sequence to execute a feasible transition path. Despite the fact that optimization algorithms have proven to be effective in automating the process of software testing, previously these have mainly been applied to white-box testing.

We can approach testing from an EFSM by first finding a set of paths that satisfy a given test criterion. It is important that these paths are feasible and so we would like to use an optimization algorithm to guide search towards paths that are likely to be feasible. It is also necessary to have a method that can guide a second search towards a test sequence to trigger a given feasible path.

This paper addressed this problem by proposing the first integrated search-based approach for automatically testing from EFSM models. The proposed approach uses two techniques: (1) a TP fitness metric that can be utilized by an optimization algorithm to guide search towards paths that are likely to be feasible and that satisfy a give test criterion such as the transition coverage. The proposed TP fitness metric is based on analyzing the data dependencies in a TP. (2) A test sequence fitness function that can guide a search for the required test sequence to exercise a given feasible TP. The proposed fitness function treats transitions in an EFSM model as a set of functions and the problem of test sequence generation is a search for a suitable test sequence to be applied to a set of functions that are called in a sequence. The fitness of a test sequence has two components: the *function distance* that measures how close a given input for a particular transition was to execute

this transition and the *function approach level* which determines how far the whole set of path inputs was to reach the target (executing the last transition in a path).

We carried out experiments using five EFSMs with the aim of evaluating the proposed approach. A total of 123 transition paths were generated using the proposed fitness metric and 123 paths were randomly generated for the purpose of comparison. For each path, three test sequence generation methods were applied: the proposed technique (ET-1), the alternative technique from the literature (ET-2) and a random generator (Rand).

Experimental results showed that the proposed fitness metric successfully guided an EA search towards paths that are feasible with an accuracy rate of approximately 96.75 %. The remaining 3.25 % of paths were found to be infeasible due to a counter. The random path generator showed that the considered EFSMs are non trivial since 61.8 % of the randomly generated paths were infeasible. Furthermore, the proposed test sequence generation technique was found to be effective and successfully triggered all of the generated feasible paths. This was not the case with the other two techniques, the success rates being approximately 73.92 % for ET-2 and 42.86 % for Rand test generation.

Further research will focus on refining the TP fitness metric approach to overcome the counter problem. This can be achieved by determining which other transitions are involved and how many times they must be called [41]. Once this can be automatically determined, the test adequacy criterion can be adapted so that it includes these required extra transitions that affect the counter variable. It would also be interesting to investigate how different penalty values can affect the TP fitness metric efficiency of guiding the search towards paths that are likely to be feasible.

References:

[1]. Korel, B., *Automated software test data generation.* Software Engineering, IEEE Transactions on, 1990. **16**(8): p. 870-879.

[2]. McMinn, P., *Search-based software test data generation: a survey: Research Articles.* Software Testing, Verification & Reliability, 2004. **14**(2): p. 105-156.

[3]. Harman, M. *Automated Test Data Generation using Search Based Software Engineering*. in *Automation of Software Test , 2007. AST '07. Second International Workshop on*. 2007.

[4]. Michael, C.C., G. McGraw, and M.A. Schatz, *Generating software test data by evolution.* Software Engineering, IEEE Transactions on, 2001. **27**(12): p. 1085-1110.

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

[6]. Hierons, R.M., T.-H. Kim, and H. Ural, *On the testability of SDL specifications.* Computer Networks, 2004. **44**(5): p. 681-700.

[7]. Duale, A.Y. and M.U. Uyar, *A method enabling feasible conformance test sequence generation for EFSM models.* Computers, IEEE Transactions on, 2004. **53**(5): p. 614-627.

[8]. Duale, A.Y., M.U. Uyar, B.D. McClure, and S. Chamberlain. *Conformance testing: towards refining VHDL specifications*. in *Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE*. 1999.

[9]. Ural, H. and B. Yang, *A test sequence selection method for protocol testing.* Communications, IEEE Transactions on, 1991. **39**(4): p. 514-523.

[10]. Hierons, R.M., *Separating sequence overlap for automated test sequence generation.* Automated Software Engineering., 2006. **13**(2): p. 283-301.

[11]. Derderian, K., R.M. Hierons, M. Harman, and Q. Guo, *Automated Unique Input Output Sequence Generation for Conformance Testing of FSMs.* The Computer Journal, 2006. **49**(3): p. 331-344.

[12]. Lai, R., *A survey of communication protocol testing.* Journal of Systems and Software, 2002. **62**(1): p. 21-46.

[13]. Lee, D. and M. Yannakakis, *Principles and methods of testing finite state machines-a survey*. Proceedings of the IEEE, 1996. **84**(8): p. 1090-1123.
[14]. Hierons, R.M. and M. Harman, *Testing conformance of a deterministic implementation against a non-deterministic stream X-machine*. Theoretical Computer Science, 2004. **323**(1-3): p. 191-233.
[15]. Kalaji, A.S., R.M. Hierons, and S. Swift. *Generating Feasible Transition Paths for Testing from an Extended Finite State Machine (EFSM)*. in *Software Testing, Verification, and Validation (ICST), 2009 2nd International IEEE Conference on*. 2009. Denver, Colorado - USA: IEEE.
[16]. Ramalingom, T., K. Thulasiraman, and A. Das, *Context independent unique state identification sequences for testing communication protocols modelled as extended finite state machines*. Computer Communications, 2003. **26**(14): p. 1622-1633.
[17]. Shih, C.-H., J.-D. Huang, and J.-Y. Jou. *Stimulus generation for interface protocol verification using the nondeterministic extended finite state machine model*. in *High-Level Design Validation and Test Workshop, 2005. Tenth IEEE International*. 2005.
[18]. Bochmann, G.V., *Specifications of a simplified transport protocol using different formal description techniques.* Computer Networks and ISDN Systems, 1990. **18**(5): p. 335-377.
[19]. Korel, B., L.H. Tahat, and B. Vaysburg. *Model based regression test reduction using dependence analysis*. in *Software Maintenance, 2002. Proceedings. International Conference on*. 2002: IEEE.
[20]. Hogrefe, D., *OSI formal specification case study: the Inres protocol and service. Technical Report IAM-91-012*. 1991, University of Bern, Institute of Computer Science and Applied Methematics. p. 5.
[21]. Tai, K.-C., *A program complexity metric based on data flow information in control graphs*, in *Proceedings of the 7th international conference on Software engineering*. 1984, IEEE Press: Orlando, Florida, United States.
[22]. Weiser, M., *Program slicing*, in *Proceedings of the 5th international conference on Software engineering*. 1981, IEEE Press: San Diego, California, United States.
[23]. Holland, J.H., *Adaptation in natural and artificial systems*. 1992, Cambridge, MA: MIT Press. 211.
[24]. Yao, X. *Global optimisation by evolutionary algorithms*. in *Parallel Algorithms/Architecture Synthesis, 1997. Proceedings. Second Aizu International Symposium*. 1997.
[25]. Srinivas, M. and L.M. Patnaik, *Genetic algorithms: a survey*. Computer, 1994. **27**(6): p. 17-26.
[26]. Baresel, A., D. Binkley, M. Harman, and B. Korel, *Evolutionary testing in the presence of loop-assigned flags: a testability transformation approach*, in *Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis*. 2004, ACM: Boston, Massachusetts, USA.
[27]. Tracey, N., J. Clark, K. Mander, and J. McDermid. *An automated framework for structural test-data generation*. in *Automated Software Engineering, 1998. Proceedings. 13th IEEE International Conference on*. 1998.
[28]. Wegener, J., A. Baresel, and H. Sthamer, *Evolutionary test environment for automatic structural testing.* Information and Software Technology, 2001. **43**(14): p. 841-854.
[29]. Derderian, K., R.M. Hierons, M. Harman, and Q. Guo, *Generating feasible input sequences for extended finite state machines (EFSMs) using genetic algorithms*, in *Proceedings of the 2005 conference on Genetic and evolutionary computation*. 2005, ACM: Washington DC, USA.
[30]. Kalaji, A.S., R.M. Hierons, and S. Swift. *A Search-Based Approach for Automatic Test Generation from Extended Finite State Machine (EFSM)*. in *Testing: Academia & Industry Conference - Practice And Research Techniques (TAIC-PART 2009)*. 2009. Windsor, UK: IEEE.
[31]. Lefticaru, R. and F. Ipate, *Functional Search-based Testing from State Machines*, in *Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation*. 2008, IEEE Computer Society.
[32]. Pohlheim, H. *GEATbx - Genetic and Evolutionary Algorithm Toolbox for Matlab*. 1994-2008 [cited; Available from: http://www.geatbx.com.
[33]. Chanson, S.T. and J. Zhu. *A unified approach to protocol test sequence generation*. in *INFOCOM '93. Proceedings.Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future. IEEE*. 1993.
[34]. Cheng, K.-T. and A.S. Krishnakumar, *Automatic generation of functional vectors using the extended finite state machine model.* ACM Transactions on Design Automation of Electronic Systems., 1996. **1**(1): p. 57-79.
[35]. Dahbura, T.A., K.K. Sabnani, and M.U. Uyar, *Formal methods for generating protocol conformance test sequences.* Proceedings of the IEEE, 1990. **78**(8): p. 1317-1326.

[36]. Petrenko, A., G.v. Bochmann, and M. Yao, *On fault coverage of tests for finite state specifications.* Computer Networks and ISDN Systems, 1996. **29**(1): p. 81-106.
[37]. Ramalingom, T., K. Thulasiraman, and A. Das. *Context independent unique sequences generation for protocol testing*. in *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*. 1996.
[38]. Sarikaya, B., G.v. Bochmann, and E. Cerny, *A Test Design Methodology for Protocol Testing.* Software Engineering, IEEE Transactions on, 1987. **SE-13**(5): p. 518-531.
[39]. Budkowski, S. and P. Dembinski, *An introduction to Estelle: a specification language for distributed systems.* Computer Networks and ISDN Systems., 1987. **14**(1): p. 3-23.
[40]. Bourhfir, C., R. Dssouli, and E.M. Aboulhamid, *Automatic Test Generation for EFSM-based Systems. Technical report*. 1996, University of Montreal, TR-1043. p. 1-59.
[41]. Kalaji, A.S., R.M. Hierons, and S. Swift. *A Testability Transformation Approach for State-Based Programs*. in *Search Based Software Engineering, 2009 1st International Symposium on*. 2009. Windsor, UK: IEEE.

Appendix –A: Subject Transition Paths

This appendix reports the subject transition paths for each EFSM case study. Each table shows the subject paths that were generated by (1) the EA search that implements the proposed TP fitness metric and (2) the random path generator.

Table A-1. Two sets of subject paths for the in-flight EFSM

Path ID	Subject paths	Fitness	Params.
$TP_{1\text{-}1\text{-}EA}$	$t_0()$,**$t_1(p_1,p_2,p_3,p_4)$**,$t_6()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{12}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{14}()$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{24}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}1\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_2()$,**$t_1(p_5,p_6,p_7,p_8)$**,$t_2()$,$t_1(p_9,p_{10},p_{11},p_{12})$,$t_2()$,$t_1(p_{13},p_{14},p_{15},p_{16})$,$t_2()$,$t_1(p_{17},p_{18},p_{19},p_{20})$	288	20
$TP_{1\text{-}2\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_2()$**,$t_1(p_5,p_6,p_7,p_8)$,$t_{30}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{18}()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_{11}()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	180	20
$TP_{1\text{-}2\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_2()$**,$t_1(p_5,p_6,p_7,p_8)$,$t_2()$,$t_1(p_9,p_{10},p_{11},p_{12})$,$t_{28}()$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{18}()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	234	20
$TP_{1\text{-}3\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_3()$**,$t_8(p_5,p_6,p_7,p_8)$,$t_9()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_9()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_9()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}3\text{-}RA}$	$t_0()$,$t_{28}()$,$t_{22}(p_1,p_2,p_3,p_4)$,$t_{18}()$,$t_{11}()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{16}()$,$t_{31}()$,**$t_3()$**,$t_8(p_9,p_{10},p_{11},p_{12})$	40198	12
$TP_{1\text{-}4\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_4()$**,$t_8(p_5,p_6,p_7,p_8)$,$t_9()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{10}()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_9()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}4\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{29}()$,$t_{24}()$,$t_{31}()$,**$t_4()$**,$t_{12}()$,$t_{13}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{21}()$	41246	8
$TP_{1\text{-}5\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_5()$**,$t_8(p_5,p_6,p_7,p_8)$,$t_9()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{16}()$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{24}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}5\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{29}()$,$t_{21}()$,$t_{13}()$,$t_{24}()$,$t_{31}()$,**$t_5()$**,$t_8(p_5,p_6,p_7,p_8)$,$t_{16}()$	41246	8
$TP_{1\text{-}6\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_6()$**,$t_8(p_5,p_6,p_7,p_8)$,$t_9()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{10}()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_{13}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}6\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_6()$**,$t_8(p_5,p_6,p_7,p_8)$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{12}()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_{12}()$,$t_{10}()$,$t_{13}()$	30090	16
$TP_{1\text{-}7\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_6()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{10}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{11}()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,**$t_7()$**,$t_1(p_{17},p_{18},p_{19},p_{20})$	126	20
$TP_{1\text{-}7\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_6()$,$t_8(p_5,p_6,p_7,p_8)$,**$t_7()$**,$t_1(p_9,p_{10},p_{11},p_{12})$,$t_2()$,$t_1(p_{13},p_{14},p_{15},p_{16})$,$t_2()$,$t_1(p_{17},p_{18},p_{19},p_{20})$	234	20
$TP_{1\text{-}8\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_6()$,**$t_8(p_5,p_6,p_7,p_8)$**,$t_{10}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{12}()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_{11}()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}8\text{-}RA}$	$t_0()$,$t_2()$,$t_{28}()$,$t_{21}()$,**$t_8(p_1,p_2,p_3,p_4)$**,$t_{13}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{25}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$	40042	16
$TP_{1\text{-}9\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_6()$,$t_8(p_5,p_6,p_7,p_8)$,**$t_9()$**,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_9()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_9()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}9\text{-}RA}$	$t_0()$,$t_2()$,$t_{29}()$,$t_{23}()$,$t_{22}(p_1,p_2,p_3,p_4)$,$t_{17}()$,$t_8(p_5,p_6,p_7,p_8)$,**$t_9()$**,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{12}()$	30108	12
$TP_{1\text{-}10\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_4()$,$t_8(p_5,p_6,p_7,p_8)$,$t_9()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_9()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,**$t_{10}()$**,$t_8(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}10\text{-}RA}$	$t_0()$,$t_2()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{28}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{31}()$,$t_{29}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{21}()$,**$t_{10}()$**	30204	12
$TP_{1\text{-}11\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_4()$,$t_8(p_5,p_6,p_7,p_8)$,**$t_{11}()$**,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{11}()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_9()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}11\text{-}RA}$	$t_0()$,$t_3()$,**$t_{11}()$**,$t_8(p_1,p_2,p_3,p_4)$,$t_{11}()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{10}()$,$t_{10}()$,$t_{11}()$,$t_8(p_9,p_{10},p_{11},p_{12})$	40072	12
$TP_{1\text{-}12\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_6()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{10}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,**$t_{12}()$**,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_9()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}12\text{-}RA}$	$t_0()$,$t_{30}()$,$t_{22}(p_1,p_2,p_3,p_4)$,$t_{17}()$,$t_8(p_5,p_6,p_7,p_8)$,$t_8(p_9,p_{10},p_{11},p_{12})$,**$t_{12}()$**,$t_{12}()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_7()$	30180	16
$TP_{1\text{-}13\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_3()$,$t_8(p_5,p_6,p_7,p_8)$,**$t_{13}()$**,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{24}()$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{24}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}13\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{27}()$,$t_{17}()$,**$t_{13}()$**,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{26}()$,$t_{19}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{13}()$	30228	12
$TP_{1\text{-}14\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_5()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{12}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,**$t_{14}()$**,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{19}()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	126	20
$TP_{1\text{-}14\text{-}RA}$	$t_0()$,$t_{30}()$,$t_{22}(p_1,p_2,p_3,p_4)$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{31}()$,$t_5()$,$t_8(p_9,p_{10},p_{11},p_{12})$,**$t_{14}()$**,$t_{31}()$,$t_1(p_{13},p_{14},p_{15},p_{16})$	40180	16
$TP_{1\text{-}15\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_5()$,$t_8(p_5,p_6,p_7,p_8)$,$t_9()$,$t_8(p_9,p_{10},p_{11},p_{12})$,**$t_{15}()$**,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{24}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	78	20
$TP_{1\text{-}15\text{-}RA}$	$t_0()$,$t_5()$,$t_8(p_1,p_2,p_3,p_4)$,$t_{11}()$,**$t_{15}()$**,$t_{23}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{24}()$,$t_{24}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$	40102	12
$TP_{1\text{-}16\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_6()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{10}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,**$t_{16}()$**,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{21}()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	126	20
$TP_{1\text{-}16\text{-}RA}$	$t_0()$,$t_6()$,$t_{10}()$,$t_{10}()$,$t_8(p_1,p_2,p_3,p_4)$,$t_{14}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{18}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,**$t_{16}()$**	30108	12
$TP_{1\text{-}17\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_3()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{13}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,**$t_{17}()$**,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_{12}()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	126	20
$TP_{1\text{-}17\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{29}()$,$t_{22}(p_5,p_6,p_7,p_8)$,**$t_{17}()$**,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_9()$,$t_{12}()$,$t_9()$,$t_7()$	30222	12
$TP_{1\text{-}18\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_3()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{13}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,**$t_{18}()$**,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_{13}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	126	20
$TP_{1\text{-}18\text{-}RA}$	$t_0()$,$t_{27}()$,$t_{22}(p_1,p_2,p_3,p_4)$,$t_{24}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{20}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{10}()$,$t_{15}()$,**$t_{18}()$**	30204	12
$TP_{1\text{-}19\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_3()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{14}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,**$t_{19}()$**,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_9()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	126	20
$TP_{1\text{-}19\text{-}RA}$	$t_0()$,$t_2()$,$t_{28}()$,$t_{22}(p_1,p_2,p_3,p_4)$,$t_{18}()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{16}()$,**$t_{19}()$**,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{15}()$	30186	12
$TP_{1\text{-}20\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_4()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{16}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,**$t_{20}()$**,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_9()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	126	20
$TP_{1\text{-}20\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_1(p_5,p_6,p_7,p_8)$,$t_{28}()$,$t_{26}()$,**$t_{20}()$**,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{11}()$,$t_7()$,$t_1(p_{13},p_{14},p_{15},p_{16})$	40198	16
$TP_{1\text{-}21\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_5()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{13}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,**$t_{21}()$**,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_{16}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	126	20
$TP_{1\text{-}21\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{28}()$,**$t_{21}()$**,$t_{16}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{18}()$,$t_{15}()$,$t_{17}()$,$t_8(p_9,p_{10},p_{11},p_{12})$	40276	12
$TP_{1\text{-}22\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{28}()$,**$t_{22}(p_5,p_6,p_7,p_8)$**,$t_{26}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{24}()$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{24}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}22\text{-}RA}$	$t_0()$,$t_{28}()$,$t_{17}()$,$t_{15}()$,$t_{19}()$,$t_8(p_1,p_2,p_3,p_4)$,$t_9()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{13}()$,**$t_{22}(p_9,p_{10},p_{11},p_{12})$**	40036	12
$TP_{1\text{-}23\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{30}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{24}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{24}()$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,**$t_{23}()$**,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	78	20
$TP_{1\text{-}23\text{-}RA}$	$t_0()$,$t_3()$,$t_{16}()$,$t_{22}(p_1,p_2,p_3,p_4)$,**$t_{23}()$**,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{20}()$,$t_{10}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_9()$	30132	12
$TP_{1\text{-}24\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_5()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{14}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,**$t_{24}()$**,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{26}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}24\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{29}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{23}()$,$t_{25}()$,**$t_{24}()$**,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{18}()$	30162	16
$TP_{1\text{-}25\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{28}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{24}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,**$t_{25}()$**,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{26}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	78	20
$TP_{1\text{-}25\text{-}RA}$	$t_0()$,$t_{29}()$,$t_{22}(p_1,p_2,p_3,p_4)$,**$t_{25}()$**,$t_{17}()$,$t_{14}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{31}()$,$t_{30}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$	40204	12
$TP_{1\text{-}26\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{30}()$,$t_{22}(p_5,p_6,p_7,p_8)$,**$t_{26}()$**,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{24}()$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{26}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}26\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{27}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{19}()$,$t_{12}()$,$t_{12}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_{16}()$,**$t_{26}()$**	30168	12
$TP_{1\text{-}27\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_{27}()$**,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{26}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{26}()$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{26}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	78	20
$TP_{1\text{-}27\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_{27}()$**,$t_{31}()$,$t_1(p_5,p_6,p_7,p_8)$,$t_{30}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{17}()$,$t_{12}()$,$t_{16}()$	30222	12
$TP_{1\text{-}28\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_{28}()$**,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{24}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{26}()$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{26}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	72	20
$TP_{1\text{-}28\text{-}RA}$	$t_0()$,$t_4()$,$t_{13}()$,$t_{22}(p_1,p_2,p_3,p_4)$,$t_{17}()$,**$t_8(p_5,p_6,p_7,p_8)$**,$t_7()$,$t_{28}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{26}()$	30180	12
$TP_{1\text{-}29\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_{29}()$**,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{26}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{26}()$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$,$t_{26}()$,$t_{22}(p_{17},p_{18},p_{19},p_{20})$	78	20
$TP_{1\text{-}29\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_{29}()$**,$t_{18}()$,$t_9()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{15}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{25}()$,$t_{22}(p_{13},p_{14},p_{15},p_{16})$	20162	16
$TP_{1\text{-}30\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,**$t_{30}()$**,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{21}()$,$t_8(p_9,p_{10},p_{11},p_{12})$,$t_9()$,$t_8(p_{13},p_{14},p_{15},p_{16})$,$t_9()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	126	20
$TP_{1\text{-}30\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_1(p_5,p_6,p_7,p_8)$,**$t_{30}()$**,$t_{22}(p_9,p_{10},p_{11},p_{12})$,$t_{26}()$,$t_{18}()$,$t_{14}()$,$t_{20}()$,$t_8(p_{13},p_{14},p_{15},p_{16})$	40198	16
$TP_{1\text{-}31\text{-}EA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_3()$,$t_8(p_5,p_6,p_7,p_8)$,$t_{13}()$,$t_{22}(p_9,p_{10},p_{11},p_{12})$,**$t_{31}()$**,$t_1(p_{13},p_{14},p_{15},p_{16})$,$t_3()$,$t_8(p_{17},p_{18},p_{19},p_{20})$	126	20
$TP_{1\text{-}31\text{-}RA}$	$t_0()$,$t_1(p_1,p_2,p_3,p_4)$,$t_{29}()$,$t_{22}(p_5,p_6,p_7,p_8)$,$t_{26}()$,$t_{26}()$,$t_{26}()$,**$t_{31}()$**,$t_1(p_9,p_{10},p_{11},p_{12})$,$t_{30}()$	30168	12

Table A-2. Two sets of subject paths for the class II transport protocol EFSM

Path ID	Subject paths	Fitness	Params.
$TP_{2\text{-}0\text{-}EA}$	$\mathbf{t_0(p_0)}$,$t_4()$,$t_0(p_1)$,$t_4()$,$t_1(p_2,p_3)$,$t_6()$,$t_{18}()$,$t_1(p_4,p_5)$,$t_6()$,$t_{18}()$	0	6
$TP_{2\text{-}0\text{-}RA}$	$\mathbf{t_0(p_0)}$,$t_2(p_1,p_2)$,$t_{12}(p_3,p_4)$,$t_{14}(p_5,p_6)$,$t_{17}()$,$t_{18}()$,$t_1(p_7,p_8)$,$t_6()$,$t_{18}()$,$t_0(p_9)$	7	10
$TP_{2\text{-}1\text{-}EA}$	$\mathbf{t_1}(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_1(p_2,p_3)$,$t_6()$,$t_{18}()$,$t_1(p_4,p_5)$,$t_6()$,$t_{18}()$,$t_1(p_6,p_7)$	0	8
$TP_{2\text{-}1\text{-}RA}$	$\mathbf{t_1(p_0,p_1)}$,$t_5(p_2)$,$t_{14}(p_3,p_4)$,$t_8(p_5)$,$t_{16}()$,$t_{20}()$,$t_0(p_6)$,$t_3(p_7,p_8)$,$t_{20}()$,$t_1(p_9,p_{10})$	10070	11
$TP_{2\text{-}2\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_1(p_2,p_3)$,$t_6()$,$t_{18}()$,$t_0(p_4)$,$\mathbf{t_2(p_5,p_6)}$,$t_{17}()$,$t_{18}()$,	4	7
$TP_{2\text{-}2\text{-}RA}$	$t_0(p_0)$,$\mathbf{t_2(p_1,p_2)}$,$t_{11}(p_3,p_4)$,$t_8(p_5)$,$t_{16}()$,$t_{20}()$,$t_0(p_6)$,$t_4()$,$t_1(p_7,p_8)$,$t_5(p_9)$	10068	10
$TP_{2\text{-}3\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_0(p_2)$,$\mathbf{t_3(p_3,p_4)}$,$t_{19}()$,$t_0(p_5)$,$t_4()$,$t_1(p_6,p_7)$,$t_6()$	6	8
$TP_{2\text{-}3\text{-}RA}$	$t_0(p_0)$,$\mathbf{t_3(p_1,p_2)}$,$t_{19}()$,$t_1(p_3,p_4)$,$t_5(p_5)$,$t_{13}(p_6,p_7)$,$t_7()$,$t_9(p_8)$,$t_{10}(p_9)$,$t_{16}()$	82	10
$TP_{2\text{-}4\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_1(p_2,p_3)$,$t_6()$,$t_{18}()$,$t_0(p_4)$,$t_4()$,$t_0(p_5)$,$\mathbf{t_4()}$	0	6
$TP_{2\text{-}4\text{-}RA}$	$t_0(p_0)$,$\mathbf{t_4()}$,$t_0(p_1)$,$t_3(p_2,p_3)$,$t_{19}()$,$t_0(p_4)$,$t_2(p_5,p_6)$,$t_8(p_7)$,$t_{14}(p_8,p_9)$,$t_{16}()$	10070	10
$TP_{2\text{-}5\text{-}EA}$	$t_1(p_0,p_1)$,$\mathbf{t_5(p_2)}$,$t_{16}()$,$t_{19}()$,$t_1(p_3,p_4)$,$t_6()$,$t_{18}()$,$t_1(p_5,p_6)$,$t_6()$,$t_{18}()$	4	7
$TP_{2\text{-}5\text{-}RA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_1(p_2,p_3)$,$\mathbf{t_5(p_4)}$,$t_{12}(p_5,p_6)$,$t_{17}()$,$t_{18}()$,$t_1(p_7,p_8)$,$t_6()$	34	9
$TP_{2\text{-}6\text{-}EA}$	$t_0(p_0)$,$t_4()$,$t_1(p_1,p_2)$,$\mathbf{t_6()}$,$t_{18}()$,$t_0(p_3)$,$t_4()$,$t_0(p_4)$,$t_4()$,$t_0(p_5)$	0	6
$TP_{2\text{-}6\text{-}RA}$	$t_1(p_0,p_1)$,$\mathbf{t_6()}$,$t_{18}()$,$t_1(p_2,p_3)$,$t_6()$,$t_{18}()$,$t_0(p_4)$,$t_2(p_5,p_6)$,$t_{16}()$,$t_{20}()$	4	7
$TP_{2\text{-}7\text{-}EA}$	$t_0(p_0)$,$t_4()$,$t_1(p_1,p_2)$,$t_6()$,$t_{18}()$,$t_1(p_3,p_4)$,$t_5(p_5)$,$\mathbf{t_7()}$,$t_{17}()$,$t_{18}()$	28	6
$TP_{2\text{-}7\text{-}RA}$	$t_0(p_0)$,$t_2(p_1,p_2)$,$\mathbf{t_7()}$,$t_7()$,$t_{14}(p_3,p_4)$,$t_{12}(p_5,p_6)$,$t_{14}(p_7,p_8)$,$t_{14}(p_9,p_{10})$,$t_{15}()$,$t_{11}(p_{11},p_{12})$	910	13
$TP_{2\text{-}8\text{-}EA}$	$t_1(p_0,p_1)$,$t_5(p_2)$,$t_{10}(p_3)$,$\mathbf{t_8(p_4)}$,$t_{16}()$,$t_{19}()$,$t_1(p_5,p_6)$,$t_6()$,$t_{18}()$,$t_1(p_7,p_8)$	40	9
$TP_{2\text{-}8\text{-}RA}$	$t_0(p_0)$,$t_3(p_1,p_2)$,$t_{20}()$,$t_1(p_3,p_4)$,$t_6()$,$t_{18}()$,$t_1(p_5,p_6)$,$t_5(p_7)$,$\mathbf{t_9(p_8)}$,$t_8(p_9)$	10040	10
$TP_{2\text{-}9\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_0(p_2)$,$t_4()$,$t_0(p_3)$,$t_4()$,$t_0(p_4)$,$t_2(p_5,p_6)$,$\mathbf{t_9(p_7)}$	10	8
$TP_{2\text{-}9\text{-}RA}$	$t_0(p_0)$,$t_2(p_1,p_2)$,$t_7()$,$t_{15}()$,$t_7()$,$t_8(p_3)$,$\mathbf{t_9(p_4)}$,$t_{11}(p_5,p_6)$,$t_{17}()$,$t_{18}()$	10496	7
$TP_{2\text{-}10\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_0(p_2)$,$t_2(p_3,p_4)$,$\mathbf{t_{10}(p_5)}$,$t_{17}()$,$t_{18}()$,$t_1(p_6,p_7)$,$t_6()$	4	8
$TP_{2\text{-}10\text{-}RA}$	$t_1(p_0,p_1)$,$t_5(p_2)$,$t_8(p_3)$,$t_{15}()$,$\mathbf{t_{10}(p_4)}$,$t_{10}(p_5)$,$t_7()$,$t_{13}(p_6,p_7)$,$t_{12}(p_8,p_9)$,$t_{15}()$	10358	10
$TP_{2\text{-}11\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_0(p_2)$,$t_2(p_3,p_4)$,$t_{10}(p_5)$,$\mathbf{t_{11}(p_6,p_7)}$,$t_{17}()$,$t_{18}()$,$t_1(p_8,p_9)$	40	10
$TP_{2\text{-}11\text{-}RA}$	$t_1(p_0,p_1)$,$t_5(p_2)$,$t_{15}()$,$t_{15}()$,$t_{12}(p_3,p_4)$,$\mathbf{t_{11}(p_5,p_6)}$,$t_{13}(p_7,p_8)$,$t_{17}()$,$t_{18}()$,$t_1(p_9,p_{10})$	160	11
$TP_{2\text{-}12\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_0(p_2)$,$t_2(p_3,p_4)$,$\mathbf{t_{12}(p_5,p_6)}$,$t_{16}()$,$t_{19}()$,$t_1(p_7,p_8)$,$t_6()$	34	9
$TP_{2\text{-}12\text{-}RA}$	$t_0(p_0)$,$t_2(p_1,p_2)$,$t_{10}(p_3)$,$t_7()$,$t_8(p_4)$,$t_8(p_5)$,$t_{11}(p_6,p_7)$,$\mathbf{t_{12}(p_8,p_9)}$,$t_{11}(p_{10},p_{11})$,$t_{12}(p_{12},p_{13})$	772	14
$TP_{2\text{-}13\text{-}EA}$	$t_0(p_0)$,$t_4()$,$t_1(p_1,p_2)$,$t_6()$,$t_{18}()$,$t_1(p_3,p_4)$,$t_5(p_5)$,$\mathbf{t_{13}(p_6,p_7)}$,$t_{17}()$,$t_{18}()$	46	8
$TP_{2\text{-}13\text{-}RA}$	$t_0(p_0)$,$t_2(p_1,p_2)$,$\mathbf{t_{13}(p_3,p_4)}$,$t_7()$,$t_7()$,$t_7()$,$t_{12}(p_5,p_6)$,$t_{11}(p_7,p_8)$,$t_{12}(p_9,p_{10})$,$t_{11}(p_{11},p_{12})$	834	13
$TP_{2\text{-}14\text{-}EA}$	$t_0(p_0)$,$t_4()$,$t_0(p_1)$,$t_4()$,$t_1(p_2,p_3)$,$t_6()$,$t_{18}()$,$t_1(p_4,p_5)$,$t_5(p_6)$,$\mathbf{t_{14}(p_7,p_8)}$	40	9
$TP_{2\text{-}14\text{-}RA}$	$t_0(p_0)$,$t_2(p_1,p_2)$,$t_{15}()$,$t_7()$,$\mathbf{t_{14}(p_3,p_4)}$,$t_{12}(p_5,p_6)$,$t_{17}()$,$t_{18}()$,$t_0(p_7)$,$t_4()$	306	8
$TP_{2\text{-}15\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_0(p_2)$,$t_2(p_3,p_4)$,$\mathbf{t_{15}()}$,$t_{16}()$,$t_{19}()$,$t_1(p_5,p_6)$,$t_6()$	28	7
$TP_{2\text{-}15\text{-}RA}$	$t_0(p_0)$,$t_3(p_1,p_2)$,$t_{19}()$,$t_1(p_3,p_4)$,$t_6()$,$t_{18}()$,$t_1(p_5,p_6)$,$t_5(p_7)$,$t_8(p_8)$,$\mathbf{t_{15}()}$	10058	9
$TP_{2\text{-}16\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_1(p_2,p_3)$,$t_6()$,$t_{18}()$,$t_1(p_4,p_5)$,$t_5(p_6)$,$\mathbf{t_{16}()}$,$t_{19}()$	4	7
$TP_{2\text{-}16\text{-}RA}$	$t_1(p_0,p_1)$,$t_5(p_2)$,$\mathbf{t_{16}()}$,$t_{20}()$,$t_0(p_3)$,$t_2(p_4,p_5)$,$t_{10}(p_6)$,$t_9(p_7)$,$t_7()$,$t_{17}()$	68	8
$TP_{2\text{-}17\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_1(p_2,p_3)$,$t_6()$,$t_{18}()$,$t_1(p_4,p_5)$,$t_5(p_6)$,$\mathbf{t_{17}()}$,$t_{18}()$	4	7
$TP_{2\text{-}17\text{-}RA}$	$t_0(p_0)$,$t_2(p_1,p_2)$,$t_8(p_3)$,$t_{14}(p_4,p_5)$,$t_{15}()$,$t_{16}()$,$t_{20}()$,$t_1(p_6,p_7)$,$t_5(p_8)$,$\mathbf{t_{17}()}$	10092	9
$TP_{2\text{-}18\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$\mathbf{t_{18}()}$,$t_1(p_2,p_3)$,$t_6()$,$t_{18}()$,$t_1(p_4,p_5)$,$t_6()$,$t_{18}()$,$t_1(p_6,p_7)$	0	8
$TP_{2\text{-}18\text{-}RA}$	$t_0(p_0)$,$t_4()$,$t_0(p_1)$,$t_4()$,$t_0(p_2)$,$t_4()$,$t_1(p_3,p_4)$,$t_5(p_5)$,$t_{17}()$,$\mathbf{t_{18}()}$	4	6
$TP_{2\text{-}19\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_0(p_2)$,$t_2(p_3,p_4)$,$t_{16}()$,$\mathbf{t_{19}()}$,$t_0(p_5)$,$t_4()$,$t_0(p_6)$	4	7
$TP_{2\text{-}19\text{-}RA}$	$t_1(p_0,p_1)$,$t_5(p_2)$,$t_{17}()$,$t_{18}()$,$t_0(p_3)$,$t_3(p_4,p_5)$,$\mathbf{t_{19}()}$,$t_0(p_6)$,$t_4()$,$t_0(p_7)$	10	8
$TP_{2\text{-}20\text{-}EA}$	$t_1(p_0,p_1)$,$t_6()$,$t_{18}()$,$t_0(p_2)$,$t_2(p_3,p_4)$,$t_{16}()$,$\mathbf{t_{20}()}$,$t_0(p_5)$,$t_4()$,$t_1(p_6,p_7)$	4	8
$TP_{2\text{-}20\text{-}RA}$	$t_1(p_0,p_1)$,$t_5(p_2)$,$t_{17}()$,$t_{18}()$,$t_0(p_3)$,$t_3(p_4,p_5)$,$\mathbf{t_{20}()}$,$t_0(p_6)$,$t_3(p_7,p_8)$,$t_{19}()$	16	9

Table A-3. Two sets of subject paths for the Lift system EFSM

Path ID	Subject paths	Fitness	Params.
$TP_{3\text{-}0\text{-}EA}$	**$t_0()$**,$t_1(p_1)$,$t_2(p_2,p_3)$,$t_1(p_4)$,$t_2(p_5,p_6)$,$t_1(p_7)$,$t_2(p_8,p_9)$,$t_1(p_{10})$,$t_2(p_{11},p_{12})$,$t_1(p_{13})$	72	13
$TP_{3\text{-}0\text{-}RA}$	**$t_0()$**,$t_1(p_1)$,$t_{19}(p_2,p_3,p_4)$,$t_{21}(p_5)$,$t_8(p_6,p_7)$,$t_4(p_8,p_9,p_{10})$,$t_{19}(p_{11},p_{12},p_{13})$,$t_{21}(p_{14})$,$t_7(p_{15})$,$t_{22}(p_{16},p_{17},p_{18})$	40214	18
$TP_{3\text{-}1\text{-}EA}$	$t_0()$,**$t_1(p_1)$**,$t_2(p_2,p_3)$,$t_1(p_4)$,$t_2(p_5,p_6)$,$t_1(p_7)$,$t_2(p_8,p_9)$,$t_1(p_{10})$,$t_2(p_{11},p_{12})$,$t_1(p_{13})$	72	13
$TP_{3\text{-}1\text{-}RA}$	$t_0()$,$t_{17}(p_1,p_2,p_3)$,$t_{13}(p_4)$,$t_{13}(p_5)$,$t_{14}(p_6,p_7)$,$t_{11}(p_8,p_9,p_{10})$,$t_6(p_{11},p_{12},p_{13})$,$t_1(p_{14})$,**$t_1(p_{15})$**,$t_5(p_{16},p_{17},p_{18})$	30242	18
$TP_{3\text{-}2\text{-}EA}$	$t_0()$,$t_1(p_1)$,**$t_2(p_2,p_3)$**,$t_1(p_4)$,$t_2(p_5,p_6)$,$t_1(p_7)$,$t_2(p_8,p_9)$,$t_1(p_{10})$,$t_2(p_{11},p_{12})$,$t_1(p_{13})$	72	13
$TP_{3\text{-}2\text{-}RA}$	$t_0()$,$t_2(p_1,p_2)$,$t_{16}(p_3,p_4,p_5)$,$t_{13}(p_6)$,$t_{23}(p_7,p_8,p_9)$,$t_{20}(p_{10})$,**$t_2(p_{11},p_{12})$**,$t_{19}(p_{13},p_{14},p_{15})$,$t_{24}(p_{16})$,$t_{11}(p_{17},p_{18},p_{19})$	40214	19
$TP_{3\text{-}3\text{-}EA}$	$t_0()$,$t_1(p_1)$,$t_2(p_2,p_3)$,**$t_3(p_4,p_5,p_6)$**,$t_7(p_7)$,$t_8(p_8,p_9)$,$t_7(p_{10})$,$t_8(p_{11},p_{12})$,$t_7(p_{13})$,$t_8(p_{14},p_{15})$	110	15
$TP_{3\text{-}3\text{-}RA}$	$t_0()$,**$t_3(p_1,p_2,p_3)$**,$t_7(p_4)$,$t_8(p_5,p_6)$,$t_{22}(p_7,p_8,p_9)$,$t_{20}(p_{10})$,$t_2(p_{11},p_{12})$,$t_1(p_{13})$,$t_1(p_{14})$,$t_{17}(p_{15},p_{16},p_{17})$	40184	17
$TP_{3\text{-}4\text{-}EA}$	$t_0()$,$t_5(p_1,p_2,p_3)$,$t_7(p_4)$,$t_8(p_5,p_6)$,$t_7(p_7)$,$t_8(p_8,p_9)$,$t_7(p_{10})$,$t_8(p_{11},p_{12})$,**$t_4(p_{13},p_{14},p_{15})$**,$t_1(p_{16})$	130	16
$TP_{3\text{-}4\text{-}RA}$	$t_0()$,$t_5(p_1,p_2,p_3)$,**$t_4(p_4,p_5,p_6)$**,$t_2(p_7,p_8)$,$t_{19}(p_9,p_{10},p_{11})$,$t_{24}(p_{12})$,$t_{14}(p_{13},p_{14})$,$t_{11}(p_{15},p_{16},p_{17})$,$t_7(p_{18})$,$t_8(p_{19},p_{20})$	30204	20
$TP_{3\text{-}5\text{-}EA}$	$t_0()$,**$t_5(p_1,p_2,p_3)$**,$t_7(p_4)$,$t_8(p_5,p_6)$,$t_7(p_7)$,$t_8(p_8,p_9)$,$t_7(p_{10})$,$t_8(p_{11},p_{12})$,$t_7(p_{13})$,$t_8(p_{14},p_{15})$	92	15
$TP_{3\text{-}5\text{-}RA}$	$t_0()$,$t_1(p_1)$,**$t_5(p_2,p_3,p_4)$**,$t_8(p_5,p_6)$,$t_9(p_7,p_8,p_9)$,$t_{14}(p_{10},p_{11})$,$t_{14}(p_{12},p_{13})$,$t_{15}(p_{14},p_{15},p_{16})$,$t_1(p_{17})$,$t_2(p_{18},p_{19})$	30168	19
$TP_{3\text{-}6\text{-}EA}$	$t_0()$,$t_5(p_1,p_2,p_3)$,**$t_6(p_4,p_5,p_6)$**,$t_1(p_7)$,$t_2(p_8,p_9)$,$t_1(p_{10})$,$t_2(p_{11},p_{12})$,$t_1(p_{13})$,$t_2(p_{14},p_{15})$,$t_1(p_{16})$	112	16
$TP_{3\text{-}6\text{-}RA}$	$t_0()$,$t_{19}(p_1,p_2,p_3)$,$t_{24}(p_4)$,$t_{14}(p_5,p_6)$,$t_{13}(p_7)$,$t_{14}(p_8,p_9)$,$t_{11}(p_{10},p_{11},p_{12})$,$t_7(p_{13})$,**$t_6(p_{14},p_{15},p_{16})$**,$t_2(p_{17},p_{18})$	30196	18
$TP_{3\text{-}7\text{-}EA}$	$t_0()$,$t_5(p_1,p_2,p_3)$,**$t_7(p_4)$**,$t_8(p_5,p_6)$,$t_7(p_7)$,$t_8(p_8,p_9)$,$t_7(p_{10})$,$t_8(p_{11},p_{12})$,$t_7(p_{13})$,$t_8(p_{14},p_{15})$	92	15
$TP_{3\text{-}7\text{-}RA}$	$t_0()$,$t_5(p_1,p_2,p_3)$,$t_{22}(p_4,p_5,p_6)$,$t_{24}(p_7)$,$t_{11}(p_8,p_9,p_{10})$,$t_{12}(p_{11},p_{12},p_{13})$,$t_{13}(p_{14})$,$t_{14}(p_{15},p_{16})$,$t_{11}(p_{17},p_{18},p_{19})$,**$t_7(p_{20})$**	30206	20
$TP_{3\text{-}8\text{-}EA}$	$t_0()$,$t_5(p_1,p_2,p_3)$,$t_7(p_4)$,$t_8(p_5,p_6)$,$t_7(p_7)$,$t_8(p_8,p_9)$,$t_7(p_{10})$,**$t_8(p_{11},p_{12})$**,$t_7(p_{13})$,$t_8(p_{14},p_{15})$	92	15
$TP_{3\text{-}8\text{-}RA}$	$t_0()$,$t_2(p_1,p_2)$,$t_2(p_3,p_4)$,$t_3(p_5,p_6,p_7)$,$t_{22}(p_8,p_9,p_{10})$,$t_{21}(p_{11})$,**$t_8(p_{12},p_{13})$**,$t_{22}(p_{14},p_{15},p_{16})$,$t_{20}(p_{17})$,$t_{19}(p_{18},p_{19},p_{20})$	30232	20
$TP_{3\text{-}9\text{-}EA}$	$t_0()$,$t_1(p_1)$,$t_2(p_2,p_3)$,$t_{19}(p_4,p_5,p_6)$,$t_{21}(p_7)$,**$t_9(p_8,p_9,p_{10})$**,$t_{13}(p_{11})$,$t_{14}(p_{12},p_{13})$,$t_{13}(p_{14})$,$t_{14}(p_{15},p_{16})$	152	16
$TP_{3\text{-}9\text{-}RA}$	$t_0()$,$t_2(p_1,p_2)$,$t_{19}(p_3,p_4,p_5)$,$t_{21}(p_6)$,**$t_9(p_7,p_8,p_9)$**,$t_{23}(p_{10},p_{11},p_{12})$,$t_{20}(p_{13})$,$t_{16}(p_{14},p_{15},p_{16})$,$t_{11}(p_{17},p_{18},p_{19})$,$t_7(p_{20})$	20252	20
$TP_{3\text{-}10\text{-}EA}$	$t_0()$,$t_1(p_1)$,$t_2(p_2,p_3)$,$t_{17}(p_4,p_5,p_6)$,$t_{13}(p_7)$,$t_{14}(p_8,p_9)$,**$t_{10}(p_{10},p_{11},p_{12})$**,$t_7(p_{13})$,$t_8(p_{14},p_{15})$,$t_7(p_{16})$	130	16
$TP_{3\text{-}10\text{-}RA}$	$t_0()$,$t_{19}(p_1,p_2,p_3)$,$t_{24}(p_4)$,$t_{14}(p_5,p_6)$,$t_{14}(p_7,p_8)$,$t_{14}(p_9,p_{10})$,**$t_{10}(p_{11},p_{12},p_{13})$**,$t_4(p_{14},p_{15},p_{16})$,$t_{19}(p_{17},p_{18},p_{19})$,$t_{20}(p_{20})$	40214	20
$TP_{3\text{-}11\text{-}EA}$	$t_0()$,$t_{17}(p_1,p_2,p_3)$,**$t_{11}(p_4,p_5,p_6)$**,$t_7(p_7)$,$t_8(p_8,p_9)$,$t_7(p_{10})$,$t_8(p_{11},p_{12})$,$t_7(p_{13})$,$t_8(p_{14},p_{15})$,$t_7(p_{16})$	112	16
$TP_{3\text{-}11\text{-}RA}$	$t_0()$,$t_{17}(p_1,p_2,p_3)$,$t_{13}(p_4)$,$t_{13}(p_5)$,$t_{14}(p_6,p_7)$,$t_{23}(p_8,p_9,p_{10})$,$t_{21}(p_{11})$,$t_8(p_{12},p_{13})$,$t_9(p_{14},p_{15},p_{16})$,**$t_{11}(p_{17},p_{18},p_{19})$**	30192	19
$TP_{3\text{-}12\text{-}EA}$	$t_0()$,$t_{17}(p_1,p_2,p_3)$,$t_{11}(p_4,p_5,p_6)$,**$t_{12}(p_7,p_8,p_9)$**,$t_{13}(p_{10})$,$t_{14}(p_{11},p_{12})$,$t_{13}(p_{13})$,$t_{14}(p_{14},p_{15})$,$t_{13}(p_{16})$,$t_{14}(p_{17},p_{18})$	132	18
$TP_{3\text{-}12\text{-}RA}$	$t_0()$,$t_{17}(p_1,p_2,p_3)$,$t_{14}(p_4,p_5)$,$t_{14}(p_6,p_7)$,$t_{15}(p_8,p_9,p_{10})$,$t_1(p_{11})$,$t_{19}(p_{12},p_{13},p_{14})$,$t_{21}(p_{15})$,$t_8(p_{16},p_{17})$,**$t_{12}(p_{18},p_{19},p_{20})$**	40222	20
$TP_{3\text{-}13\text{-}EA}$	$t_0()$,$t_{17}(p_1,p_2,p_3)$,**$t_{13}(p_4)$**,$t_{14}(p_5,p_6)$,$t_{13}(p_7)$,$t_{14}(p_8,p_9)$,$t_{13}(p_{10})$,$t_{14}(p_{11},p_{12})$,$t_{13}(p_{13})$,$t_{14}(p_{14},p_{15})$	92	15
$TP_{3\text{-}13\text{-}RA}$	$t_0()$,$t_1(p_1)$,$t_{16}(p_2,p_3,p_4)$,**$t_{13}(p_5)$**,$t_{14}(p_6,p_7)$,$t_{14}(p_8,p_9)$,$t_{15}(p_{10},p_{11},p_{12})$,$t_3(p_{13},p_{14},p_{15})$,$t_{22}(p_{16},p_{17},p_{18})$,$t_{21}(p_{19})$	40210	19
$TP_{3\text{-}14\text{-}EA}$	$t_0()$,$t_{17}(p_1,p_2,p_3)$,$t_{13}(p_4)$,**$t_{14}(p_5,p_6)$**,$t_{13}(p_7)$,$t_{14}(p_8,p_9)$,$t_{13}(p_{10})$,$t_{14}(p_{11},p_{12})$,$t_{13}(p_{13})$,$t_{14}(p_{14},p_{15})$	92	15
$TP_{3\text{-}14\text{-}RA}$	$t_0()$,$t_2(p_1,p_2)$,$t_1(p_3)$,$t_2(p_4,p_5)$,$t_{19}(p_6,p_7,p_8)$,$t_{24}(p_9)$,**$t_{14}(p_{10},p_{11})$**,$t_{10}(p_{12},p_{13},p_{14})$,$t_4(p_{15},p_{16},p_{17})$,$t_5(p_{18},p_{19},p_{20})$	30210	20
$TP_{3\text{-}15\text{-}EA}$	$t_0()$,$t_1(p_1)$,$t_2(p_2,p_3)$,$t_1(p_4)$,$t_2(p_5,p_6)$,$t_{19}(p_7,p_8,p_9)$,$t_{24}(p_{10})$,**$t_{15}(p_{11},p_{12},p_{13})$**,$t_1(p_{14})$,$t_2(p_{15},p_{16})$	152	16
$TP_{3\text{-}15\text{-}RA}$	$t_0()$,$t_1(p_1)$,$t_{17}(p_2,p_3,p_4)$,$t_{14}(p_5,p_6)$,$t_{23}(p_7,p_8,p_9)$,$t_{21}(p_{10})$,$t_7(p_{11})$,$t_{22}(p_{12},p_{13},p_{14})$,$t_{24}(p_{15})$,**$t_{15}(p_{16},p_{17},p_{18})$**	40214	18
$TP_{3\text{-}16\text{-}EA}$	$t_0()$,$t_1(p_1)$,$t_2(p_2,p_3)$,**$t_{16}(p_4,p_5,p_6)$**,$t_{13}(p_7)$,$t_{14}(p_8,p_9)$,$t_{13}(p_{10})$,$t_{14}(p_{11},p_{12})$,$t_{13}(p_{13})$,$t_{14}(p_{14},p_{15})$	110	15
$TP_{3\text{-}16\text{-}RA}$	$t_0()$,$t_1(p_1)$,$t_1(p_2)$,$t_2(p_3,p_4)$,**$t_{16}(p_5,p_6,p_7)$**,$t_{10}(p_8,p_9,p_{10})$,$t_{22}(p_{11},p_{12},p_{13})$,$t_{20}(p_{14})$,$t_{19}(p_{15},p_{16},p_{17})$,$t_{21}(p_{18})$	10232	18
$TP_{3\text{-}17\text{-}EA}$	$t_0()$,**$t_{17}(p_1,p_2,p_3)$**,$t_{13}(p_4)$,$t_{14}(p_5,p_6)$,$t_{13}(p_7)$,$t_{14}(p_8,p_9)$,$t_{13}(p_{10})$,$t_{14}(p_{11},p_{12})$,$t_{13}(p_{13})$,$t_{14}(p_{14},p_{15})$	92	15
$TP_{3\text{-}17\text{-}RA}$	$t_0()$,**$t_{17}(p_1,p_2,p_3)$**,$t_{13}(p_4)$,$t_{10}(p_5,p_6,p_7)$,$t_7(p_8)$,$t_8(p_9,p_{10})$,$t_6(p_{11},p_{12},p_{13})$,$t_3(p_{14},p_{15},p_{16})$,$t_7(p_{17})$,$t_8(p_{18},p_{19})$	40182	19
$TP_{3\text{-}18\text{-}EA}$	$t_0()$,$t_{17}(p_1,p_2,p_3)$,**$t_{18}(p_4,p_5,p_6)$**,$t_1(p_7)$,$t_2(p_8,p_9)$,$t_1(p_{10})$,$t_2(p_{11},p_{12})$,$t_1(p_{13})$,$t_2(p_{14},p_{15})$,$t_1(p_{16})$	112	16
$TP_{3\text{-}18\text{-}RA}$	$t_0()$,$t_{19}(p_1,p_2,p_3)$,$t_{24}(p_4)$,$t_{13}(p_5)$,**$t_{18}(p_6,p_7,p_8)$**,$t_2(p_9,p_{10})$,$t_{16}(p_{11},p_{12},p_{13})$,$t_{23}(p_{14},p_{15},p_{16})$,$t_{24}(p_{17})$,$t_{13}(p_{18})$	30196	18
$TP_{3\text{-}19\text{-}EA}$	$t_0()$,$t_1(p_1)$,$t_2(p_2,p_3)$,$t_1(p_4)$,$t_2(p_5,p_6)$,**$t_{19}(p_7,p_8,p_9)$**,$t_{24}(p_{10})$,$t_{13}(p_{11})$,$t_{14}(p_{12},p_{13})$,$t_{13}(p_{14})$	114	14
$TP_{3\text{-}19\text{-}RA}$	$t_0()$,$t_2(p_1,p_2)$,$t_2(p_3,p_4)$,**$t_{19}(p_5,p_6,p_7)$**,$t_{21}(p_8)$,$t_7(p_9)$,$t_8(p_{10},p_{11})$,$t_9(p_{12},p_{13},p_{14})$,$t_{10}(p_{15},p_{16},p_{17})$,$t_{12}(p_{18},p_{19},p_{20})$	30210	20
$TP_{3\text{-}20\text{-}EA}$	$t_0()$,$t_1(p_1)$,$t_2(p_2,p_3)$,$t_1(p_4)$,$t_2(p_5,p_6)$,$t_1(p_7)$,$t_2(p_8,p_9)$,$t_{19}(p_{10},p_{11},p_{12})$,**$t_{20}(p_{13})$**,$t_1(p_{14})$	114	14
$TP_{3\text{-}20\text{-}RA}$	$t_0()$,$t_2(p_1,p_2)$,$t_1(p_3)$,$t_5(p_4,p_5,p_6)$,$t_8(p_7,p_8)$,$t_{22}(p_9,p_{10},p_{11})$,**$t_{20}(p_{12})$**,$t_3(p_{13},p_{14},p_{15})$,$t_4(p_{16},p_{17},p_{18})$,$t_1(p_{19})$	20240	19
$TP_{3\text{-}21\text{-}EA}$	$t_0()$,$t_1(p_1)$,$t_2(p_2,p_3)$,$t_1(p_4)$,$t_2(p_5,p_6)$,$t_{19}(p_7,p_8,p_9)$,**$t_{21}(p_{10})$**,$t_7(p_{11})$,$t_8(p_{12},p_{13})$,$t_7(p_{14})$	114	14
$TP_{3\text{-}21\text{-}RA}$	$t_0()$,$t_2(p_1,p_2)$,$t_{19}(p_3,p_4,p_5)$,**$t_{21}(p_6)$**,$t_6(p_7,p_8,p_9)$,$t_2(p_{10},p_{11})$,$t_{19}(p_{12},p_{13},p_{14})$,$t_{21}(p_{15})$,$t_4(p_{16},p_{17},p_{18})$,$t_2(p_{19},p_{20})$	40214	20
$TP_{3\text{-}22\text{-}EA}$	$t_0()$,$t_1(p_1)$,$t_2(p_2,p_3)$,$t_1(p_4)$,$t_2(p_5,p_6)$,$t_{19}(p_7,p_8,p_9)$,$t_{21}(p_{10})$,**$t_{22}(p_{11},p_{12},p_{13})$**,$t_{24}(p_{14})$,$t_{13}(p_{15})$	156	15
$TP_{3\text{-}22\text{-}RA}$	$t_0()$,$t_{19}(p_1,p_2,p_3)$,$t_{24}(p_4)$,$t_{13}(p_5)$,$t_{14}(p_6,p_7)$,$t_{11}(p_8,p_9,p_{10})$,**$t_{22}(p_{11},p_{12},p_{13})$**,$t_{21}(p_{14})$,$t_{22}(p_{15},p_{16},p_{17})$,$t_{20}(p_{18})$	30194	18
$TP_{3\text{-}23\text{-}EA}$	$t_0()$,$t_{17}(p_1,p_2,p_3)$,$t_{13}(p_4)$,$t_{14}(p_5,p_6)$,$t_{13}(p_7)$,$t_{14}(p_8,p_9)$,**$t_{23}(p_{10},p_{11},p_{12})$**,$t_{24}(p_{13})$,$t_{13}(p_{14})$,$t_{14}(p_{15},p_{16})$	134	16
$TP_{3\text{-}23\text{-}RA}$	$t_0()$,$t_1(p_1)$,$t_2(p_2,p_3)$,$t_{19}(p_4,p_5,p_6)$,$t_{24}(p_7)$,**$t_{23}(p_8,p_9,p_{10})$**,$t_{20}(p_{11})$,$t_{19}(p_{12},p_{13},p_{14})$,$t_{24}(p_{15})$,$t_{13}(p_{16})$	198	16
$TP_{3\text{-}24\text{-}EA}$	$t_0()$,$t_1(p_1)$,$t_2(p_2,p_3)$,$t_1(p_4)$,$t_2(p_5,p_6)$,$t_{19}(p_7,p_8,p_9)$,**$t_{24}(p_{10})$**,$t_{13}(p_{11})$,$t_{14}(p_{12},p_{13})$,$t_{13}(p_{14})$	114	14
$TP_{3\text{-}24\text{-}RA}$	$t_0()$,$t_2(p_1,p_2)$,$t_{19}(p_3,p_4,p_5)$,**$t_{24}(p_6)$**,$t_{10}(p_7,p_8,p_9)$,$t_{22}(p_{10},p_{11},p_{12})$,$t_{20}(p_{13})$,$t_{19}(p_{14},p_{15},p_{16})$,$t_{20}(p_{17})$,$t_2(p_{18},p_{19})$	20236	19

Table A-4. Two sets of subject paths for the ATM EFSM

Path ID	Subject paths	Fitness	Params.
$TP_{4\text{-}1\text{-}EA}$	**$t_1()$**,$t_4(p_0)$,$t_6(p_1)$,$t_{25}()$,$t_{26}()$,$t_7()$,$t_9()$,$t_8()$,$t_{10}()$,$t_{25}()$	36	2
$TP_{4\text{-}1\text{-}RA}$	**$t_1()$**,$t_4(p_0)$,$t_5(p_1)$,$t_{23}(p_2)$,$t_1()$,$t_4(p_3)$,$t_6(p_4)$,$t_8()$,$t_{17}(p_5)$,$t_{22}(p_6)$	120	7
$TP_{4\text{-}2\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_{25}()$,$t_{26}()$,$t_7()$,$t_9()$,$t_{23}(p_2)$,$t_1()$,**$t_2(p_3)$**	54	4
$TP_{4\text{-}2\text{-}RA}$	$t_1()$,$t_3(p_0)$,$t_1()$,$t_4(p_1)$,$t_{24}(p_2)$,$t_1()$,**$t_2(p_3)$**,$t_3(p_4)$,$t_1()$,$t_4(p_5)$	10238	6
$TP_{4\text{-}3\text{-}EA}$	$t_1()$,$t_2(p_0)$,**$t_3(p_1)$**,$t_1()$,$t_4(p_2)$,$t_5(p_3)$,$t_7()$,$t_9()$,$t_8()$,$t_{10}()$	208	4
$TP_{4\text{-}3\text{-}RA}$	$t_1()$,$t_2(p_0)$,**$t_3(p_1)$**,$t_1()$,$t_2(p_2)$,$t_3(p_3)$,$t_1()$,$t_4(p_4)$,$t_{24}(p_5)$,$t_1()$	380	6
$TP_{4\text{-}4\text{-}EA}$	$t_1()$,**$t_4(p_0)$**,$t_6(p_1)$,$t_8()$,$t_{10}()$,$t_{25}()$,$t_{26}()$,$t_{25}()$,$t_{26}()$,$t_7()$,	36	2
$TP_{4\text{-}4\text{-}RA}$	$t_1()$,**$t_4(p_0)$**,$t_6(p_1)$,$t_8()$,$t_{20}(p_2)$,$t_{22}(p_3)$,$t_{20}(p_4)$,$t_{22}(p_5)$,$t_{18}(p_6)$,$t_{21}(p_7)$	104	8
$TP_{4\text{-}5\text{-}EA}$	$t_1()$,$t_4(p_0)$,**$t_5(p_1)$**,$t_7()$,$t_9()$,$t_{25}()$,$t_{26}()$,$t_{25}()$,$t_{26}()$,$t_{25}()$	36	2
$TP_{4\text{-}5\text{-}RA}$	$t_1()$,$t_2(p_0)$,$t_3(p_1)$,$t_1()$,$t_4(p_2)$,**$t_5(p_3)$**,$t_{25}()$,$t_{28}(p_4,p_5,p_6)$,$t_{30}(p_7)$,$t_{28}(p_8,p_9,p_{10})$	414	11
$TP_{4\text{-}6\text{-}EA}$	$t_1()$,$t_4(p_0)$,**$t_6(p_1)$**,$t_{25}()$,$t_{26}()$,$t_{25}()$,$t_{26}()$,$t_{25}()$,$t_{26}()$,$t_{25}()$,	36	2
$TP_{4\text{-}6\text{-}RA}$	$t_1()$,$t_2(p_0)$,$t_3(p_1)$,$t_1()$,$t_3(p_2)$,$t_1()$,$t_4(p_3)$,**$t_6(p_4)$**,$t_7()$,$t_{13}(p_5)$	10222	6
$TP_{4\text{-}7\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_{25}()$,$t_{26}()$,**$t_7()$**,$t_9()$,$t_8()$,$t_{10}()$,$t_8()$	36	2
$TP_{4\text{-}7\text{-}RA}$	$t_1()$,$t_3(p_0)$,$t_1()$,$t_3(p_1)$,$t_1()$,$t_4(p_2)$,$t_5(p_3)$,**$t_7()$**,$t_{14}(p_4)$,$t_{15}(p_5)$	20088	6
$TP_{4\text{-}8\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,**$t_8()$**,$t_{10}()$,$t_7()$,$t_9()$,$t_{25}()$,$t_{26}()$,$t_7()$	36	2
$TP_{4\text{-}8\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_8()$,$t_{10}()$,$t_7()$,$t_{11}(p_2)$,$t_{15}(p_3)$,$t_9()$,**$t_8()$**	60	4
$TP_{4\text{-}9\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_{25}()$,$t_{26}()$,$t_{25}()$,$t_{26}()$,$t_7()$,**$t_9()$**,$t_{25}()$	36	2
$TP_{4\text{-}9\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_7()$,$t_{12}(p_2)$,$t_{15}(p_3)$,**$t_9()$**,$t_8()$,$t_{10}()$,$t_{25}()$	60	4
$TP_{4\text{-}10\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_8()$,**$t_{10}()$**,$t_{25}()$,$t_{26}()$,$t_{25}()$,$t_{26}()$,$t_{25}()$	36	2
$TP_{4\text{-}10\text{-}RA}$	$t_1()$,$t_2(p_0)$,$t_3(p_1)$,$t_1()$,$t_4(p_2)$,$t_6(p_3)$,$t_8()$,$t_{20}(p_4)$,$t_{22}(p_5)$,**$t_{10}()$**	232	6
$TP_{4\text{-}11\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_{25}()$,$t_{26}()$,$t_7()$,**$t_{11}(p_2)$**,$t_{16}(p_3)$,$t_9()$,$t_{25}()$	60	4
$TP_{4\text{-}11\text{-}RA}$	$t_1()$,$t_3(p_0)$,$t_1()$,$t_3(p_1)$,$t_1()$,$t_4(p_2)$,$t_6(p_3)$,$t_7()$,**$t_{11}(p_4)$**,$t_{15}(p_5)$	20072	6
$TP_{4\text{-}12\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_7()$,**$t_{12}(p_2)$**,$t_{15}(p_3)$,$t_9()$,$t_8()$,$t_{10}()$,$t_8()$	60	4
$TP_{4\text{-}12\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_7()$,$t_{13}(p_2)$,$t_{16}(p_3)$,$t_{11}(p_4)$,$t_{16}(p_5)$,**$t_{12}(p_6)$**,$t_{15}(p_7)$	104	8
$TP_{4\text{-}13\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_7()$,$t_9()$,$t_7()$,**$t_{13}(p_2)$**,$t_{16}(p_3)$,$t_9()$,$t_8()$	56	4
$TP_{4\text{-}13\text{-}RA}$	$t_1()$,$t_2(p_0)$,$t_4(p_1)$,$t_5(p_2)$,$t_7()$,$t_{14}(p_3)$,$t_{15}(p_4)$,**$t_{13}(p_5)$**,$t_{15}(p_6)$,$t_{14}(p_7)$	270	8
$TP_{4\text{-}14\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_7()$,**$t_{14}(p_2)$**,$t_{16}(p_3)$,$t_9()$,$t_8()$,$t_{10}()$,$t_{25}()$	76	4
$TP_{4\text{-}14\text{-}RA}$	$t_1()$,$t_2(p_0)$,$t_4(p_1)$,$t_5(p_2)$,$t_{25}()$,$t_{26}()$,$t_7()$,**$t_{14}(p_3)$**,$t_{16}(p_4)$,$t_{12}(p_5)$	242	6
$TP_{4\text{-}15\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_7()$,$t_9()$,$t_{25}()$,$t_{26}()$,$t_7()$,$t_{13}(p_2)$,**$t_{15}(p_3)$**	56	4
$TP_{4\text{-}15\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_7()$,$t_9()$,$t_7()$,$t_{14}(p_2)$,$t_{15}(p_3)$,$t_{12}(p_4)$,**$t_{15}(p_5)$**	100	6
$TP_{4\text{-}16\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_7()$,$t_9()$,$t_7()$,$t_{13}(p_2)$,**$t_{16}(p_3)$**,$t_9()$,$t_{25}()$	56	4
$TP_{4\text{-}16\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_7()$,$t_{11}(p_2)$,**$t_{16}(p_3)$**,$t_{14}(p_4)$,$t_{15}(p_5)$,$t_{12}(p_6)$,$t_{15}(p_7)$	124	8
$TP_{4\text{-}17\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_8()$,$t_{17}(p_2)$,$t_{21}(p_3)$,$t_{10}()$,$t_{25}()$,$t_{26}()$,**$t_7()$**	72	4
$TP_{4\text{-}17\text{-}RA}$	$t_1()$,$t_3(p_0)$,$t_1()$,$t_4(p_1)$,$t_6(p_2)$,$t_8()$,**$t_{17}(p_3)$**,$t_{22}(p_4)$,$t_{19}(p_5)$,$t_{22}(p_6)$	10102	7
$TP_{4\text{-}18\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_{25}()$,$t_{26}()$,$t_7()$,$t_9()$,$t_8()$,**$t_{18}(p_2)$**,$t_{22}(p_3)$	56	4
$TP_{4\text{-}18\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_8()$,$t_{20}(p_2)$,$t_{22}(p_3)$,**$t_{18}(p_4)$**,$t_{22}(p_5)$,$t_{10}()$,$t_{25}()$	80	6
$TP_{4\text{-}19\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_8()$,$t_{10}()$,$t_8()$,**$t_{19}(p_2)$**,$t_{21}(p_3)$,$t_{10}()$,$t_8()$	60	4
$TP_{4\text{-}19\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_{23}(p_2)$,$t_1()$,$t_4(p_3)$,$t_6(p_4)$,$t_8()$,**$t_{19}(p_5)$**,$t_{21}(p_6)$	108	7
$TP_{4\text{-}20\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_8()$,$t_{10}()$,$t_8()$,**$t_{20}(p_2)$**,$t_{22}(p_3)$,$t_{10}()$,$t_8()$	60	4
$TP_{4\text{-}20\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_8()$,$t_{18}(p_2)$,$t_{21}(p_3)$,$t_{18}(p_4)$,$t_{21}(p_5)$,**$t_{20}(p_6)$**,$t_{22}(p_7)$	100	8
$TP_{4\text{-}21\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_8()$,$t_{18}(p_2)$,**$t_{21}(p_3)$**,$t_{10}()$,$t_{25}()$,$t_{26}()$,$t_8()$	56	4
$TP_{4\text{-}21\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_{25}()$,$t_{26}()$,$t_8()$,$t_{20}(p_2)$,$t_{21}(p_3)$,$t_{19}(p_4)$,**$t_{21}(p_5)$**	84	6
$TP_{4\text{-}22\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_8()$,$t_{18}(p_2)$,**$t_{22}(p_3)$**,$t_{10}()$,$t_{25}()$,$t_{26}()$,$t_7()$	56	4
$TP_{4\text{-}22\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_8()$,$t_{20}(p_2)$,$t_{22}(p_3)$,$t_{20}(p_4)$,$t_{21}(p_5)$,$t_{19}(p_6)$,**$t_{22}(p_7)$**	108	8
$TP_{4\text{-}23\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_7()$,$t_9()$,$t_{25}()$,$t_{26}()$,$t_{25}()$,$t_{26}()$,**$t_{23}(p_2)$**	48	3
$TP_{4\text{-}23\text{-}RA}$	$t_1()$,$t_3(p_0)$,$t_1()$,$t_3(p_1)$,$t_1()$,$t_4(p_2)$,$t_6(p_3)$,**$t_{23}(p_4)$**,$t_1()$,$t_2(p_5)$	20066	6
$TP_{4\text{-}24\text{-}EA}$	$t_1()$,$t_4(p_0)$,**$t_{24}(p_1)$**,$t_1()$,$t_4(p_2)$,$t_5(p_3)$,$t_7()$,$t_9()$,$t_8()$,$t_{10}()$	72	4
$TP_{4\text{-}24\text{-}RA}$	$t_1()$,$t_3(p_0)$,$t_1()$,$t_2(p_1)$,$t_4(p_2)$,**$t_{24}(p_3)$**,$t_1()$,$t_3(p_4)$,$t_1()$,$t_2(p_5)$	20208	6
$TP_{4\text{-}25\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_7()$,$t_9()$,**$t_{25}()$**,$t_{26}()$,$t_{25}()$,$t_{26}()$,$t_8()$	36	2
$TP_{4\text{-}25\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,**$t_{25}()$**,$t_{27}(p_2,p_3,p_4)$,$t_{29}(p_5)$,$t_{26}()$,$t_{23}(p_6)$,$t_1()$,$t_2(p_7)$	116	8
$TP_{4\text{-}26\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_7()$,$t_9()$,$t_{25}()$,**$t_{26}()$**,$t_{25}()$,$t_{26}()$,$t_8()$	36	2
$TP_{4\text{-}26\text{-}RA}$	$t_1()$,$t_2(p_0)$,$t_4(p_1)$,$t_5(p_2)$,$t_7()$,$t_{14}(p_3)$,$t_{16}(p_4)$,$t_9()$,$t_{25}()$,**$t_{26}()$**	230	5
$TP_{4\text{-}27\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_{25}()$,**$t_{27}(p_2,p_3,p_4)$**,$t_{30}(p_5)$,$t_{26}()$,$t_{25}()$,$t_{26}()$,$t_8()$	98	6
$TP_{4\text{-}27\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_{25}()$,**$t_{27}(p_2,p_3,p_4)$**,$t_{29}(p_5)$,$t_{28}(p_6,p_7,p_8)$,$t_{30}(p_9)$,$t_{28}(p_{10},p_{11},p_{12})$,$t_{29}(p_{13})$	264	14
$TP_{4\text{-}28\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_7()$,$t_9()$,$t_{25}()$,$t_{26}()$,$t_{25}()$,**$t_{28}(p_2,p_3,p_4)$**,$t_{30}(p_5)$	98	6
$TP_{4\text{-}28\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_{25}()$,$t_{27}(p_2,p_3,p_4)$,$t_{29}(p_5)$,**$t_{28}(p_6,p_7,p_8)$**,$t_{29}(p_9)$,$t_{27}(p_{10},p_{11},p_{12})$,$t_{30}(p_{13})$	196	14
$TP_{4\text{-}29\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_{25}()$,$t_{26}()$,$t_{25}()$,$t_{26}()$,$t_{25}()$,$t_{27}(p_2,p_3,p_4)$,**$t_{29}(p_5)$**	98	6
$TP_{4\text{-}29\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_{25}()$,$t_{28}(p_2,p_3,p_4)$,$t_{29}(p_5)$,$t_{27}(p_6,p_7,p_8)$,**$t_{29}(p_9)$,$t_{26}()$**,$t_{25}()$	146	10
$TP_{4\text{-}30\text{-}EA}$	$t_1()$,$t_4(p_0)$,$t_5(p_1)$,$t_{25}()$,$t_{27}(p_2,p_3,p_4)$,**$t_{30}(p_5)$**,$t_{26}()$,$t_8()$,$t_{10}()$,$t_{25}()$	98	6
$TP_{4\text{-}30\text{-}RA}$	$t_1()$,$t_4(p_0)$,$t_6(p_1)$,$t_{25}()$,$t_{28}(p_2,p_3,p_4)$,$t_{30}(p_5)$,$t_{28}(p_6,p_7,p_8)$,**$t_{30}(p_9)$**,$t_{26}()$,$t_8()$	254	10

Table A-5. Two sets of subject paths for the Inres initiator EFSM

Path ID	Subject paths	Fitness	Params.
$TP_{5\text{-}0\text{-}EA}$	**t_0()**,t_1(),t_2(),t_5(),t_{15}(),t_{12}(),t_1(),t_{13}(),t_{12}(),t_1()	0	0
$TP_{5\text{-}0\text{-}RA}$	**t_0()**,t_{12}(),t_1(),t_3(),t_2(),t_5(),t_{15}(),t_1(),t_3(),t_2()	0	0
$TP_{5\text{-}1\text{-}EA}$	t_0(),**t_1()**,t_2(),t_{14}(),t_{12}(),t_{12}(),t_{12}(),t_{12}(),t_{12}(),t_1()	0	0
$TP_{5\text{-}1\text{-}RA}$	t_0(),**t_1()**,t_4(),t_{12}(),t_{12}(),t_{12}(),t_1(),t_4(),t_{12}(),t_1()	20000	0
$TP_{5\text{-}2\text{-}EA}$	t_0(),t_{12}(),t_1(),t_3(),t_{13}(),t_{12}(),t_1(),**t_2()**,t_{14}(),t_1()	0	0
$TP_{5\text{-}2\text{-}RA}$	t_0(),t_{12}(),t_{12}(),t_{12}(),t_1(),t_{13}(),t_1(),**t_2()**,t_{14}(),t_1()	0	0
$TP_{5\text{-}3\text{-}EA}$	t_0(),t_1(),t_{13}(),t_{12}(),t_1(),t_{13}(),t_{12}(),t_{12}(),t_1(),**t_3()**	0	0
$TP_{5\text{-}3\text{-}RA}$	t_0(),t_{12}(),t_{12}(),t_{12}(),t_1(),t_{13}(),t_1(),t_{13}(),t_1(),**t_3()**	0	0
$TP_{5\text{-}4\text{-}EA}$	t_0(),t_1(),t_2(),t_{14}(),t_1(),t_{13}(),t_1(),t_3(),**t_4()**,t_1()	136	0
$TP_{5\text{-}4\text{-}RA}$	t_0(),t_1(),t_3(),**t_4()**,t_1(),t_{13}(),t_{12}(),t_1(),t_3(),t_2()	136	0
$TP_{5\text{-}5\text{-}EA}$	t_0(),t_{12}(),t_{12}(),t_1(),t_2(),t_{14}(),t_1(),t_2(),**t_5()**,t_{10}()	0	0
$TP_{5\text{-}5\text{-}RA}$	t_0(),t_1(),t_3(),t_{13}(),t_{12}(),t_1(),t_2(),**t_5()**,t_{10}(),t_{15}()	0	0
$TP_{5\text{-}6\text{-}EA}$	t_0(),t_{12}(),t_1(),t_2(),t_5(),t_7(p_0),t_5(),**t_6(p_1)**,t_5(),t_{15}()	48	2
$TP_{5\text{-}6\text{-}RA}$	t_0(),t_1(),t_2(),t_5(),**t_6(p_0)**,t_{14}(),t_{12}(),t_1(),t_2(),t_{14}()	10000	1
$TP_{5\text{-}7\text{-}EA}$	t_0(),t_1(),t_2(),t_5(),**t_7(p_0)**,t_5(),t_{15}(),t_1(),t_{13}(),t_{12}()	24	1
$TP_{5\text{-}7\text{-}RA}$	t_0(),t_1(),t_4(),t_1(),t_4(),t_1(),t_2(),t_5(),**t_7(p_0)**,t_5()	20024	1
$TP_{5\text{-}8\text{-}EA}$	t_0(),t_1(),t_3(),t_2(),t_5(),**t_8(p_0)**,t_{15}(),t_1(),t_{13}(),t_{12}()	6	1
$TP_{5\text{-}8\text{-}RA}$	t_0(),t_1(),t_4(),t_{12}(),t_1(),t_2(),t_5(),**t_8(p_0)**,t_8(p_1),t_{15}()	10160	2
$TP_{5\text{-}9\text{-}EA}$	t_0(),t_1(),t_2(),t_5(),t_{10}(),**t_9(p_0)**,t_{12}(),t_1(),t_{13}(),t_1()	142	1
$TP_{5\text{-}9\text{-}RA}$	t_0(),t_{12}(),t_1(),t_2(),t_5(),**t_9(p_0)**,t_1(),t_{13}(),t_{12}(),t_{12}()	10006	1
$TP_{5\text{-}10\text{-}EA}$	t_0(),t_{12}(),t_1(),t_3(),t_{13}(),t_1(),t_3(),t_2(),t_5(),**t_{10}()**	0	0
$TP_{5\text{-}10\text{-}RA}$	t_0(),t_1(),t_4(),t_1(),t_{13}(),t_{12}(),t_1(),t_2(),t_5(),**t_{10}()**	10000	0
$TP_{5\text{-}11\text{-}EA}$	t_0(),t_{12}(),t_{12}(),t_1(),t_2(),t_5(),t_{10}(),**t_{11}()**,t_{12}(),t_1()	136	0
$TP_{5\text{-}11\text{-}RA}$	t_0(),t_1(),t_2(),t_{14}(),t_1(),t_3(),t_2(),t_5(),**t_{11}()**,t_{12}(),	10000	0
$TP_{5\text{-}12\text{-}EA}$	t_0(),t_{12}(),t_1(),t_2(),t_{14}(),**t_{12}()**,t_1(),t_2(),t_{14}(),t_1()	0	0
$TP_{5\text{-}12\text{-}RA}$	t_0(),t_1(),t_3(),t_3(),t_4(),**t_{12}()**,t_{12}(),t_1(),t_2(),t_{14}()	324	0
$TP_{5\text{-}13\text{-}EA}$	t_0(),t_1(),**t_{13}()**,t_{12}(),t_1(),t_2(),t_{14}(),t_{12}(),t_{12}(),t_{12}()	0	0
$TP_{5\text{-}13\text{-}RA}$	t_0(),t_1(),t_3(),**t_{13}()**,t_1(),t_2(),t_5(),t_{15}(),t_{12}(),t_1()	0	0
$TP_{5\text{-}14\text{-}EA}$	t_0(),t_{12}(),t_1(),t_2(),**t_{14}()**,t_{12}(),t_1(),t_3(),t_2(),t_{14}()	0	0
$TP_{5\text{-}14\text{-}RA}$	t_0(),t_1(),t_4(),t_1(),t_2(),**t_{14}()**,t_{12}(),t_{12}(),t_1(),t_4()	20000	0
$TP_{5\text{-}15\text{-}EA}$	t_0(),t_1(),t_2(),t_{14}(),t_1(),t_2(),t_5(),**t_{15}()**,t_1(),t_3()	0	0
$TP_{5\text{-}15\text{-}RA}$	t_0(),t_1(),t_{13}(),t_1(),t_4(),t_1(),t_2(),t_5(),**t_{15}()**,t_{12}()	10000	0