Optimal String Edit Distance Based Test Suite Reduction for SDL Specifications

Gábor Kovács[1], Gábor Árpád Németh[1], Mahadevan Subramaniam[2], and Zoltán Pap[3]

[1] Department of Telecommunications and Media Informatics – ETIK, Budapest University of Technology and Economics, Magyar tudósok körútja 2, H-1117, Budapest, Hungary
{kovacsg,gabor.nemeth}@tmit.bme.hu
[2] Computer Science Department, University of Nebraska at Omaha Omaha, NE 68182, USA
msubramaniam@mail.unomaha.edu
[3] Ericsson Telecomm. Hungary, H-1117 Budapest, Irinyi J. u. 4-20, Hungary
zoltan.pap@ericsson.com

Abstract. We propose a test selection method that provides efficient test sets for systems based on SDL specifications. Our approach builds on previous results of Voung et al. and Feijs et al. on string edit distance based coverage metrics. The method reduces a set of test cases represented in the MSC (Message Sequence Chart) notation, while maintaining the highest possible distance between all pairs of traces defined by the given test set. The algorithm is tunable by a parameter representing the threshold distance for test redundancy. We show that the algorithm runs in polynomial time of the size of the input test set and that it is independent of the size of the system. We implemented and incorporated the algorithm into our SDL-based test selection framework, and evaluated against existing symbol coverage and fault coverage based test selection approaches by conducting experiments on the well-known INRES and Conference Protocol. Results indicate that the string edit distance based method yields similar results in terms of reduction-capability and coverage as the other approaches, but with significantly less complexity.

Keywords: SDL based test selection, string edit distance, MSC test cases.

1 Introduction

From the design perspective, system development facilitated by formal modelling has several benefits, mainly due to the higher level of abstraction it provides to system architects. A further and often overlooked advantage of modeling techniques comes from the testing phase of the development life cycle: A correct model of the system under development may serve as a basis for automating test development, which is a crucial and increasingly expensive part of the software development process.

R. Reed, A. Bilgic, and R. Gotzhein (Eds.): SDL 2009, LNCS 5719, pp. 82–97, 2009.

Research in the field of testing has established various test generation methods [1,2,3] and tools like Autolink [4], TorX [5], TGV [6] or Phact [7] for different formalisms including extended finite state machine (EFSM) based modelling languages such as SDL (Specification and Description Langugage). These methods typically have a sound mathematical background, but are often criticized for resulting in excessive number of test cases under industrial-size application. A major challenge of automatic test generation is, therefore, the detection and reduction of redundancies among the large number of test cases derived from a system model. This type of problem is addressed by test selection methods [2,8,9,10,11]: their objective is to minimize the cardinality of the target test set without sacrificing its quality. However as shown in [8] the selection problem is NP-hard, so approximative solutions must be defined [10,11].

In this paper we present a method for automatic test selection building on a promising idea first published by Voung et al. [9] and later improved by Feijs et al. [12]. Vuong's original approach defines string edit distance measures and addresses the normalization of traces represented as strings to approximate differences among patterns of system behavior; traces are considered to be similar if their distance is smaller than a given parameter. The latter paper generalizes that original idea by introducing a cycling and a reduction heuristics and gives formulae to precisely calculate the distance between traces containing finite number of traversals of loops around a state in the specification.

We utilize the string edit distance-based test coverage metric to reduce MSC (Message Sequence Chart) test sets. A two-step selection algorithm is proposed to find the minimum cardinality subset with the highest possible diversity: First the minimum cardinality for a given approximation threshold is determined by reducing the distance based selection problem to an assignment problem in bipartite graphs. Then a test set with the highest overall internal edit distance is selected from all subsets with that computed cardinality. We show that the algorithms run in polynomial time of the size of the input test set and that it is independent of the size of the system. Furthermore, the paper proposes an iterative test generation method that can be used in incremental development of compact test sets.

The algorithm has been implemented and customized to accept the SDL specification of the system under test and a test set defined in MSC. The solution has been incorporated into our SDL-based test selection framework [2], and has been evaluated against existing symbol coverage and fault coverage based test selection approaches by conducting experiments on the well-known INRES [13] and Conference Protocol [7]. Results indicate that the string edit distance based method does not show significant differences in terms of reduction-capability and coverage compared to the other approaches, but requires significantly less computation time.

The rest of the paper is organized as follows. A brief overview of our assumptions and notations is given in Sect. 2. Section 3 describes the procedure of distance maximization among the test cases. In Section 4 we introduce how the

proposed method can be used to reduce the size of the resulting test set during an automatic test generation process. We demonstrate our method through an example. Section 5 compares the string edit distance based test selection method with two other appoaches: a fault based and a coverage based solution using the sample SDL systems INRES and Conference Protocol. Section 6 presents our conclusions.

2 Preliminaries

SDL [14] is a well-accepted world standard supported by the ITU (International Telecommunication Union). SDL is widely used in the telecommunications industry for the description of telecommunication protocols, but it may be used in other fields where high reliability is required. Typically complex, event-driven, real-time and communicating systems can be effectively described in SDL.

SDL is built on the extended finite state machine (EFSM) model that is an extension of the basic FSM formalism by adding support for the use of variables. An EFSM is a 5-tuple: $EFSM = (S, I, O, V, H)$, where S is the finite set of states, I is the finite set of input symbols, O is the finite set of output symbols, V is the finite set of variables, and H is the finite set of transitions. Each transition $h \in H$ is a 6-tuple: $h = (s_j, i, P(V), A(V), o, s_k)$, where $s_j \in S$ is the start state of the transition, $i \in I$ is an input, P is a set of predicates on the variables, A is a set of actions on the variables, $o \in O$ is an output and $s' \in S$ is the next state.

2.1 Test Cases and Their String Representation

Existing test generation tools like Autolink [4] provide excellent means to derive large number of MSC traces from SDL specifications. In practice the number of such executable traces is infinite and therefore exhaustive testing based on all traces is generally impossible. The purpose of test selection (and test generation in its essence) is to identify a subset of trace set sufficient to establish a required level of confidence in the correctness of the system.

Throughout this paper we consider SDL specifications with a reliable reset signal as a test start point. A test set consists of a set of MSCs, where each test case is a finite trace after a reset input. The MSC traces in the examples are generated by means of random-walk, but any other test derivation tool or algorithm may be considered as well. Note that in this paper we use the terms test sets and trace sets interchangeably.

Traces derived from an SDL specification can be represented as strings on an arbitrary alphabet C. A mapping $M : \{I \cup O\}^+ \rightarrow C$ defines a set of pairs of an SDL signal sequence and a character of alphabet C: $\langle (x_{i1} x_{i2} \dots x_{im}), c_i \rangle$, where $x_{ik} \in I \cup O, k = 1 \dots m, k \in \mathbb{N}, c_i \in C$. Note that according to this definition several successive input or output signals may be mapped to a character of that given alphabet. Such mapping M is the marked trace notation defined in [12].

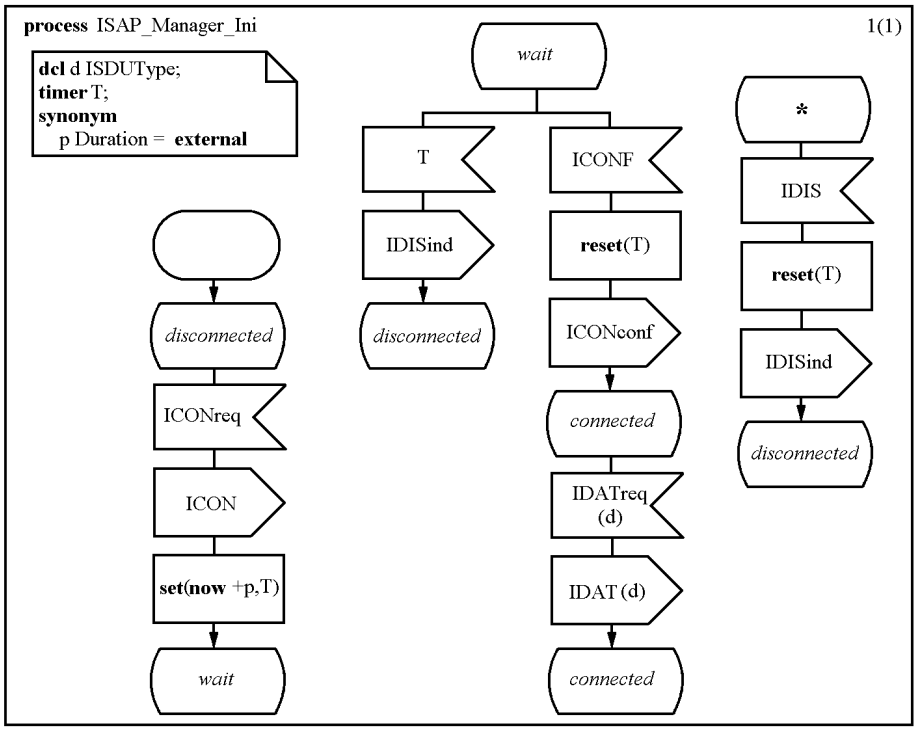


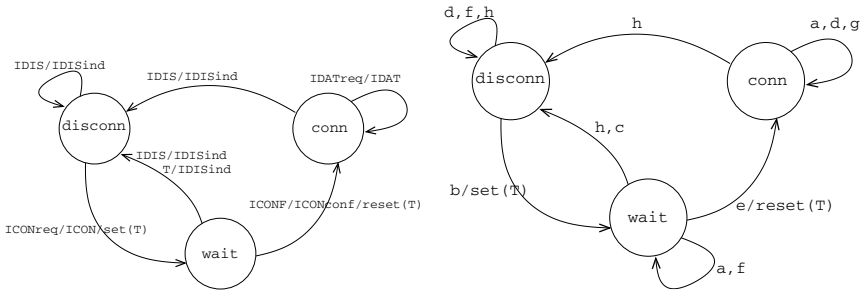
Fig. 1. INRES ISAP Manager Ini process (see [13])

Let us use the `ISAP Manager Ini` process of the INRES system [13] as example, its SDL representation can be seen in Fig. 1. The process has three states $S = \{$`disconnected`, `wait`, `connected`$\}$, the initial state is `disconnected`. The machine is not completely specified, input signals such as the `IDATreq` in state `disconnected` are not shown and considered to be implicitly consumed, that is, they do not change the state of the machine and do not produce output. The input and output signal sets are the following: $I = \{$`ICONreq`, `T`, `ICONF`, `IDATreq`, `IDIS`$\}$ and $O = \{$`ICON`, `IDISind`, `ICONconf`, `IDAT`$\}$. The timeout `T` is considered to be an input signal and according to the specification it is only possible in state `WAIT`.

Let us assume a simple signal sequence to character mapping M defined by Table 1, which maps each possible transition of the process `ISAP Manager Ini` to a character. The table indicates implicit events as well: dash represents that no output signal is produced. The state transition diagram of the unfolded FSM and the graph with mappings applied are presented in Fig. 2(b). Note that the cycling heuristics of [12] is not used in the mapping for the sake of simplicity. EFSM aspects can be taken into account when defining the mapping for parameterized signals: a parameterized signal name can be mapped to a set of different characters depending on the actual parameter values.

Table 1. Signal sequence to character mapping for ISAP Manager Ini

ICONreq/−	a
ICONreq/ICON	b
T/IDISind	c
ICONF/−	d
ICONF/ICONconf	e
IDATreq/−	f
IDATreq/IDAT	g
IDIS/IDISind	h



(a) INRES `ISAP Manager Ini` process (b) INRES `ISAP Manager Ini` process after event to character mapping

Fig. 2. The state transition diagram of the FSM of the ISAP Manager Ini process from the sample INRES protocol in original form and after event to character mapping

Figure 2(a) shows the state transition graph of the `ISAP Manager Ini` when that is unfolded into a finite state machine, and Fig. 2(b) shows the graph after applying the event to string mapping defined above.

2.2 Trace Distance

The edit distance between strings σ_1 and σ_2 is the minimum number of edit operations (character insert, character delete and character overwrite) needed to transform σ_1 to σ_2 [15]. Different edit operations may have distinct costs assigned, for the sake of simplicity we consider unit edit operator costs in the current paper.

Let Σ be a finite set of strings over the alphabet C, and let $d : \Sigma \times \Sigma \to \mathbb{R}$ be a distance metric such that $d(\sigma_1, \sigma_1) = 0$, $d(\sigma_1, \sigma_2) = d(\sigma_2, \sigma_1)$ and $d(\sigma_1, \sigma_3) \leq d(\sigma_1, \sigma_2) + d(\sigma_2, \sigma_3)$, for all $\sigma_1, \sigma_2, \sigma_3 \in S$. If the lengths of σ_1 and σ_2 are l_1 and l_2 respectively, the time and space complexity of computing the distance is $O(l_1 l_2)$. The distance metric computation above is according to one defined in [15], but other approaches can be considered as well.

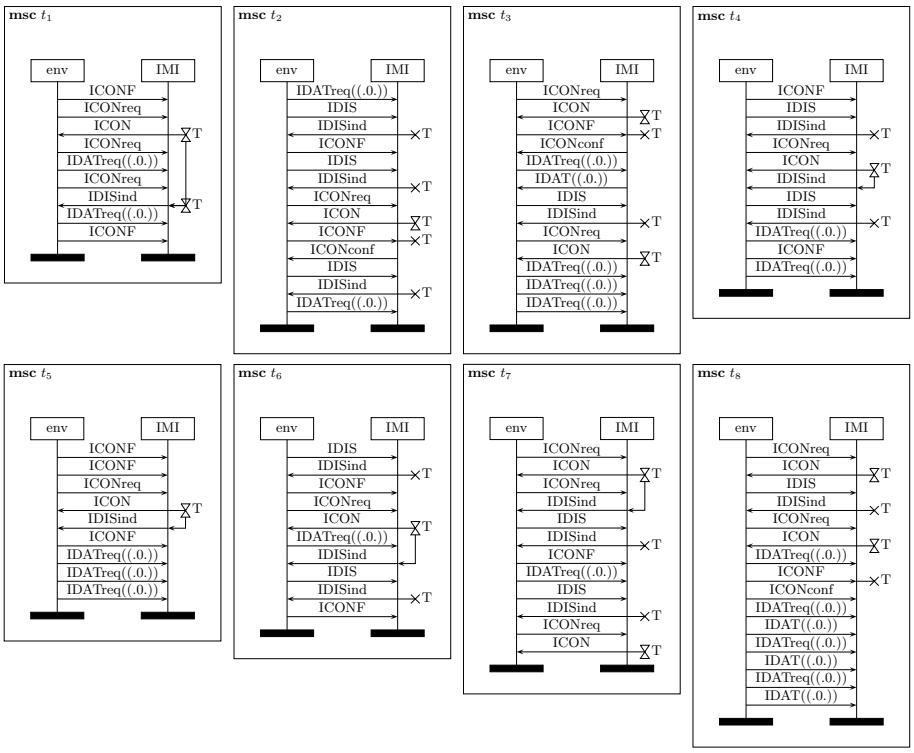


Fig. 3. The T test set of eight MSCs generated with random walk. (IMI abbreviates `ISAP Manager Ini` from the INRES system)

Let $\mathbf{D} = [d_{ij}]$ be a distance matrix of the set of MSC traces T. Let $d_{ij} = d(M(t_i),\ M(t_j))$, where $1 \leq i, j \leq |T|, i, j \in \mathbb{N}$. Note that $\mathbf{D}$ is symmetric because of the symmetric nature of distance and the size of the matrix is $|T| \times |T|$. Let $\mathbf{D}_U$ and $\mathbf{D}_L$ denote the strictly upper and lower triangular matrices of $\mathbf{D}$ respectively.

The metric space involves the normalization of edit distance values as it is done in [9] and [12]. For the sake of simplicity – and without any loss of generality – in the example of this paper we dispense with the normalization by using traces of a given length and consider that the distances are non-negative integer numbers not greater than that length.

Example. Let the test set T be composed of the MSCs of Fig. 3 derived with random walk for the `ISAP Manager Ini` process. According to the M mapping of Table 1 $M(T) = \{dbafacfd,\ fhdhbehf,\ beghbfff,\ dhbchfdf,\ ddbcdfff,\ hdbafchd,\ bachdfhb,\ bhbfeggg\}$ are the string representations of T. Note that reset inputs are omitted in the figure for the sake of clarity, and all test sequences have reset input as prefix thus start from initial state `disconnected`. These eight strings together traverse 11 of the 13 transitions of the machine in Fig. 2(b).

The distance matrix of this trace set $M(T)$ with unit cost edit operations is:

$$\mathbf{D} = \begin{bmatrix} 0&8&7&6&6&3&6&7 \\ 8&0&5&5&7&6&6&7 \\ 7&5&0&6&5&8&5&6 \\ 6&5&6&0&3&6&6&6 \\ 6&7&5&3&0&6&6&7 \\ 3&6&8&6&6&0&6&7 \\ 6&6&5&6&6&6&0&7 \\ 7&7&6&6&7&7&7&0 \end{bmatrix}$$

3 String Edit Distance Based Test Suite Reduction

This section proposes a method for selecting a test set with the highest possible diversity with regard to the distance based coverage metric. The inputs of the method are the **D** distance matrix of the traces, and an ε approximation parameter. The method consists of two stages: First the minimum cardinality of the target test set is calculated assuming the ε threshold, then the test set with the maximum internal distance is selected.

For our discussions we assume the notion of ε-approximation as defined by Feijs et. al. in [12]: T' is an ε-cover of T, where

$$T' \subseteq T, \varepsilon \geq 0 \iff \forall t \in T : \exists t' \in T' : d(M(t), M(t')) \leq \varepsilon.$$

This implies that for each T and ε there exists at least one minimal cardinality T' ε-cover of T, for which $\forall t'_i, t'_j \in T' : d(M(t'_i), M(t'_j)) > \varepsilon, t'_i \neq t'_j$. The test set T is divided into two disjoint subsets: a T'' subset of test cases that can and a $T'_0 \subseteq T'$ subset of test cases that can not be ε-covered by other test cases. The test cases from T'_0 must be included in T', and from T'' the minimum number of cases must be selected: $T' = T'_0 \cup \mathsf{reduce}(T'')$. Thus the maximum reduction of T'' yields the most compact T'.

Algorithm 1 finds the size of the most compact test suite T' that can be achieved with the string edit distance based selection in polynomial time for a given distance matrix. The inputs of the algorithm are the distance matrix of T and a ε threshold parameter. For the computation two matrices A and A_μ of boolean values and a bipartite graph $G' = (N'_R \cup N'_C, E')$ are used. The algorithm first determines which test cases of T ε-cover each other, and if they do, it is marked in **A** with 1 value (lines 4-7). Then, the size of the T'_0 set is determined in lines 8-10. In lines 11-15, if exactly the same coverage is found for two test cases, then one of them is eliminated. Lines 16-22 construct a bipartite graph G' based on matrix **A** such that the set of rows of **A** is mapped to the node set N'_R and the set of columns of **A** is mapped to the node set N'_C, and if a_{ij} is not 0, then there is an edge from n'_i to n'_j, where $n'_i \in N'_R, n'_j \in N'_C$. The Hopcroft-Karp algorithm [16] is used for finding a maximum cardinality assignment μ in G' (line 23). The k minimal size returned in line 28 is the size

Algorithm 1. Reducing the string edit distance based test case selection problem to a matching problem in bipartite graphs

input : $\mathbf{D}$ distance matrix of T test set; ε threshold
output: k, the maximum number of redundant cases for the given ε
1 data($\mathbf{A} = [a_{ij}], a_{ij} \in \{0,1\}; \mathbf{A}_\mu = [a_{\mu ij}], a_{\mu ij} \in \{0,1\}$;
2 $G' = (N', E')$ *graph, where* $N' = N'_R \cup N'_C$, $\forall n_i, n_j \in N'_R : (n_i, n_j) \notin E'$, $\forall n_i, n_j \in N'_C : (n_i, n_j) \notin E'$)
/* Initialization */
3 $k := 0, N'_R := \emptyset, N'_C := \emptyset, E' := \emptyset, \mathbf{A} := 0, \mathbf{A}_\mu := 0$;
/* Computing the **A** matrix */
4 **foreach** i, j, $1 \le i, j \le |T|$ **do**
5 **if** $d_{ij} < \varepsilon$ **then** $a_{ij} = 1$;
6 **else** $a_{ij} = 0$;
7 **endfch**
/* Counting the elements of T'_0 */
8 **foreach** i **do**
9 **if** $\sum_j a_{ij} = 0$ **then** $k := k + 1$;
10 **endfch**
/* Finding test cases with the same coverage */
11 **foreach** k, l, $1 \le k \le |T| - 1, k < l \le |T|$ **do**
if $\sum_j (a_{kj} \mathsf{xor} a_{lj}) = 0$ **then**
12
13 **foreach** j, $1 \le j \le |T|$ **do** $a_{lj} := 0$, $a_{jl} := 0$;
14 **endif**
15 **endfch**
/* Constructing the G' graph */
16 **forall the** $t_i \in T$ **do**
17 $N'_R := N'_R \cup t_i$;
18 $N'_C := N'_C \cup t_i$;
19 **endfall**
20 **foreach** i, j, $1 \le i, j \le |T|$ **do**
21 **if** $a_{ij} > 0$ **then** $E' := E' \cup \{(n'_i, n'_j)\}, n'_i \in N'_R, n'_j \in N'_C$;
22 **endfch**
23 Let E'_μ be the matching selected by the Hopcroft-Karp algorithm;
/* Marking the pairs of the maximum matching in $\mathbf{A}_\mu$ */
24 **foreach** i,j, $1 \le i, j \le |T|$ **do**
25 **if** $(n'_i, n'_j) \in E'_\mu$ **then** $\mathbf{A}_{\mu\,\mathbf{ij}} = \mathbf{1}$;
26 **else** $\mathbf{A}_{\mu\,\mathbf{ij}} = \mathbf{0}$;
27 **endfch**
28 **return** $k := k + \mathsf{rank}(A^L_\mu)$

of T'_0 plus the rank of the upper or lower triangual matrix of $\mathbf{A}_\mu$ constructed based on the maximum matching in lines 24-27.

The complexity of the construction of $\mathbf{A}$ (lines 4-7) and G' (lines 16-22) are $O(|T|^2)$. The worst-case complexity of the Hopcroft-Karp algorithm [16] applied to the graph $G' = (N', E')$ in line 23 is $O(\sqrt{|N'|}|E'|)$. Since $|N'| \le 2|T|$ and

$$\mathbf{A}_{\varepsilon=5} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 \\ 0 & 0 & 1 & \boxed{1} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & \boxed{1} & 0 \\ 0 & \boxed{1} & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

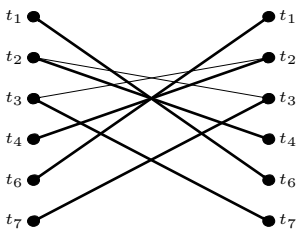


Fig. 4. The **A** matrix and the bipartite graph G' constructed from it

$|E'| \leq |T|^2$ according to lines 16-22, its complexity is $O(|T|^{5/2})$. Hence the worst case complexity of this algorithm is determined by the search for same rows in **A** in lines 11-15, which is $O(T^3)$.

Example. The ε-coverage of matrix **D** of the example at the end of Sect. 2.2 with $\varepsilon = 5$ (note that we dispense with normalization) and the bipartite graph G' constructed from **A** are in Fig. 4. In matrix **A** on the left the eighth is the only row that contains only 0 values, therefore $T'_0 = \{t_8\}$. The second and the fifth row are identical, so we may remove either t_2 or t_5 from the further processing as a redundant case. The bipartite graph G' and the matching problem constructed from T'' without the redundant t_5 can be seen on the right. Edges show that a trace ε-covers an other test case. Bold lines select a maximum cardinality matching: the selected elements of $\mathbf{A}_\mu$ boxed in the matrix **A** are $(1,6)$,$(2,4)$,$(3,7)$,$(4,2)$,$(6,1)$,$(7,3)$. The rank of the upper or lower triangular matrices of $\mathbf{A}_\mu$ is 3, therefore beside t_5, three additional test cases can be removed: $T''_U = \{t_1, t_2, t_3, t_5\}$ or $T''_L = \{t_4, t_5, t_6, t_7\}$. The resulting T' candidates are $T'_1 = \{t_1, t_2, t_3, t_8\}$, if t_5 was considered to be redundant, or $T'_2 = \{t_1, t_3, t_5, t_8\}$, if t_2 was considered to be redundant and $T'_3 = \{t_4, t_6, t_7, t_8\}$. In general more maximum cardinality assignments may exist, but all with the same cardinality. □

According to [9] and [12] two patterns of behavior are approximated to be less similar if the distance between their string representation is greater. The redundancy among the test cases in a test set is the least, if the sum of all pairwise distances is maximal. Hence, if more than one minimal cardinality T' solutions exist, the one with the maximum internal distance should be preferred.

Selecting the T' test set with minimal cardinality from the test set T, such that T' is an ε-cover of T and the test cases differ from each other as much as possible can be calculated in a polynomial time of $|T|$. This means that the sum of distances between all pairs of the test cases of T' is maximal, therefore the optimization problem is:

$$\max \sum_{t_i \in T', t_j \in T'} d(M(t_i), M(t_j)), \tag{1}$$

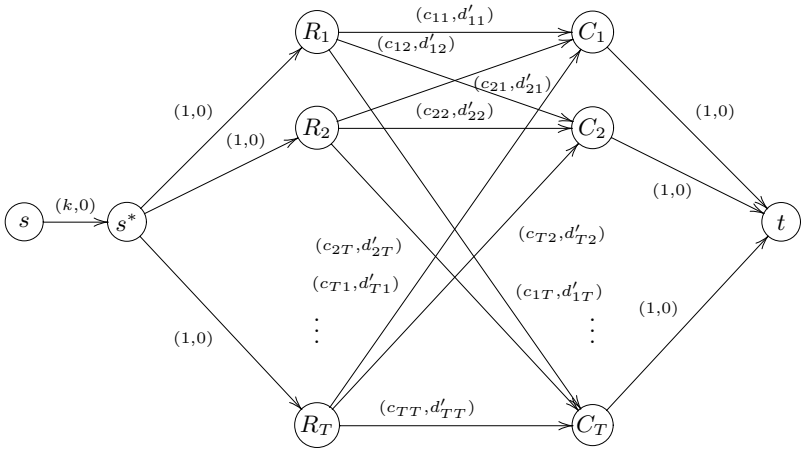


Fig. 5. The flow problem equivalent to the maximum distance k-cardinality matching

where $\forall i,j : d(M(t_i), M(t_j)) > \varepsilon$. A T' with the minimum k cardinality can be determined with Algorithm 1. There may be many T's with this k cardinality, and there is at least one with the maximum distance sum.

This problem is equivalent to the following minimum cost maximum flow problem. Let $\mathbf{C} = [c_{ij}]$, where $c_{ij} \in \{0,1\}$ a capacity matrix, and let the distance matrix $\mathbf{D} = [d_{ij}]$ of T be a cost matrix. Let $G' = (N', E')$ be a bipartite graph of distance matrix $\mathbf{D}$ as constructed in Algorithm 1. Let $G = (N, E)$ be directed weighted graph extending G' such that $N = \{s, s^*\} \cup N' \cup \{t\}$ and $E = \{(s, s^*), (s^*, n'_i)\} \cup E' \cup \{(n'_j, t)\}$, for all $n'_i \in N'_R, n'_j \in N'_C$ and let all edges between nodes N'_R and N'_C be directed from n'_R to n'_C. Each edge is assigned with a capacity value and a cost defined as follows. Let the capacity of all edges $e \in E$ be $c = 1$, except for (s, s^*) that has a capacity of k. Hence, the maximum flow capacity is determined by the $\{(s, s^*)\}$ cut and it equals to k. Let the cost of edges (s, s^*),(s^*, n'_i) and (n'_j, t) be 0, and let the cost of edges (n'_i, n'_j), where $n'_i \in N'_R, n'_j \in N'_C$, be $d'_{ij} = \max_{i,j}(d_{ij}) - d_{ij}$ for all i and j. See Fig. 5.

The minimum cost maximum flow can be found by solving the following optimization with linear programming, and can be expressed as

$$\max \textstyle\sum_{i=1}^{|T|} \sum_{j=1}^{|T|} c_{ij} d_{ij}$$

$$\text{where} \quad \sum_{j=1}^{|T|} c_{ij} \leq 1, i = 1, ..., |T|, \qquad \sum_{i=1}^{|T|} c_{ij} \leq 1, j = 1, ..., |T|,$$
$$\sum_{i=1}^{|T|} \sum_{j=1}^{|T|} c_{ij} = k, \text{ and} \qquad c_{ij} \in \{0,1\}.$$

This problem has been defined as the k-cardinality assignment problem by Dell'Amico and Martello and has been shown to be a P-space problem in [17].

Example. It has been shown that $k = 4$ for the $\varepsilon = 5$ case and therefore the "best" solution contains $|T| - k = 8 - 4 = 4$ test cases. In the example three solutions have been found. The sum of distances between pairs of T'_1 is 40, for T'_2 and T'_3 it is 38, so T'_1 is the best reduced suite.

4 Application to Selective Automatic Test Generation

This section proposes a selective extension to automatic test generation methods. When a new test case is derived from an SDL specification, before adding that to the test set as usual, the method proposed in Sect. 3 is used to find redundancies immediately. Note that this method may lead to suboptimal solution compared to the single stage optimization, but the resulting test set is already reduced and makes test selection unnecessary.

We generalize the distance function to evaluate the effect of merging a new trace into an already existing trace set. Let $d : (t', T) \rightarrow \mathbb{R}$ compute the distance between the trace t' and the trace set T such that $d(t', T) = \min_i d(t', t_i)$, where $t_i \in T$.

Algorithm 2. Selective automatic test generation

input : ε; K iteration limit; Sp specification
output: T set

```
1 T[0] := ∅;
2 k := 1;
3 repeat
4     t' := derive(Sp);
5     if d(t', T) < ε then T[k] := reduce(T[k − 1] ∪ t');
6     else T[k] := T[k − 1] ∪ t' k := k + 1;
7 until k > K ;
8 return T[k]
```

Algorithm 2 is a selective method for automatic test generation. The inputs of the method are a ε selection threshold, a K iteration limit, which is an upper bound for the number of generated test cases. The output is a reduced test set. The algorithm can co-work with any test generation algorithm [4,5,6,7]. In each iteration cycle a new test case t' is derived. That test case is added to the test set immediately if its distance from every element of the test set is greater than the given threshold. Otherwise the union of the newly generated test case and the old test set is optimized with the method of Sect. 3 to maintain the density of the test set.

Example. In the example let the iteration limit be $K = 8$, and let MSC traces from Fig. 3 be generated iteration by iteration. In general the iteration limit, the length and the number of sequences are independent, but setting all these configuration parameters to 8 simplifies this example.

Two cases are investigated, the $\varepsilon = 5$ and the $\varepsilon = 6$ case. The string edit operations insert, delete and replace are assumed to have unit cost. The iterative test generation procedure is presented only for the first case. The distance matrix for the test cases of this example can be found in Sect. 2.2, and the coverage matrix can be seen Fig. 4.

Let us first investigate the $\varepsilon = 5$ case. This $\mathbf{A}_{\varepsilon=5}$ matrix implies that the resulting set consists of four strings.

Step 1. The first string is $t' = dbafacfd$, so $T_{\varepsilon=5}[1] = \{dbafacfd\}$. Since this is the only element of T in step 1, $\mathbf{D}_{\varepsilon=5}[1] = 0$ and $\mathbf{A}_{\varepsilon=5}[1] = 0$. This sequence traverses 6 of the 13 transitions of the FSM in Fig. 2(b).

Step 2. In this step the string $t' = fhdhbehf$ is added to $T_{\varepsilon=5}[1]$, thus $T_{\varepsilon=5}[2] = \{dbafacfd, fhdhbehf\}$ and

$$\mathbf{D}_{\varepsilon=5}[2] = \begin{bmatrix} 0 & 8 \\ 8 & 0 \end{bmatrix} \quad \mathbf{A}_{\varepsilon=5}[2] = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The second sequence traverses also 6 transitions and two sequences together traverse 9 different transitions.

Step 3. The new string added to T is $t' = beghbfff$. This new string traverses 5 different transitions and provides one new, not yet traversed transition for $T_{\varepsilon=5}[2]$. However $d(t_2[2], t') = 5$, that is, these two sequences ε-cover each other, therefore one of them is dropped despite of providing a new transition.

$$\mathbf{D}_{\varepsilon=5}[3] = \begin{bmatrix} 0 & 8 & 7 \\ 8 & 0 & 5 \\ 7 & 5 & 0 \end{bmatrix} \quad \mathbf{A}_{\varepsilon=5}[3] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & \boxed{1} & 0 \end{bmatrix}$$

As the $d(t_1[2], t_2[2]) = 8 > d(t_1[2], t') = 7$, $t_1[2]$ and $t_2[2]$ are kept.

Step 4. The situation is the same as in step 3. The new string $t' = dhbchfdf$ traverses 5 transitions from which none is new for $T_{\varepsilon=5}[3]$.

$$\mathbf{D}_{\varepsilon=5}[4] = \begin{bmatrix} 0 & 8 & 6 \\ 8 & 0 & 5 \\ 6 & 5 & 0 \end{bmatrix} \quad \mathbf{A}_{\varepsilon=5}[4] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & \boxed{1} & 0 \end{bmatrix}$$

As the $d(t_1[3], t_2[3]) = 8 > d(t_1[3], t') = 6$, $t_1[3]$ and $t_2[3]$ are kept.

Step 5. In this step $t' = ddbcdfff$ is added to the sequence set, that contains $T_{\varepsilon=5}[5] = \{dbafacfd, fhdhbehf, ddbcdfff\}$. This sequence traverses only 4 different transitions which are already traversed by $T_{\varepsilon=5}[4] = T_{\varepsilon=5}[2]$, thus a redundant sequence is added to the set.

Step 6. The sixth string to be added is $t' = hdbafchd$. However $d(t_1[5], t') = 3$, therefore one of them (t_1 or t') is considered to be redundant:

$$\mathbf{D}_{\varepsilon=5}[6] = \begin{bmatrix} 0 & 8 & 6 & 3 \\ 8 & 0 & 7 & 6 \\ 6 & 7 & 0 & 6 \\ 3 & 6 & 6 & 0 \end{bmatrix} \quad \mathbf{A}_{\varepsilon=5}[6] = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \boxed{1} & 0 & 0 & 0 \end{bmatrix}$$

Because $d(t_1[5], t_2[5]) + d(t_1[5], t_3[5]) = 14 > d(t', t_2[5]) + d(t', t_3[5]) = 12$, $t_1[5]$ is kept and t' is dropped.

Step 7. Though the sequence $t' = bachdfhb$ traversing 6 different transitions is completely redundant for $T_{\varepsilon=5}[6]$, it is added to the set, because $\mathbf{A}_{\varepsilon=5}[7] = 0$. This t' also does not increase the number of traversed transitions.

Step 8. The last sequence $t' = bhbfeggg$ is also added to $T_{\varepsilon=5}[8]$ which is now $T_{\varepsilon=5}[8] = \{dbafacfd, fhdhbehf, ddbcdfff, bachdfhb, bhbfeggg\}$. This last sequence traverses two more transitions so $T_{\varepsilon=5}[8]$ traverses the 11 of the 13. This resulting set contains five sequences, one more as it would have been necessary according to $\mathbf{A}_{\varepsilon=5}$. The final distance matrix is:

$$\mathbf{D}_{\varepsilon=5}[8] = \begin{bmatrix} 0 & 8 & 6 & 6 & 7 \\ 8 & 0 & 7 & 6 & 7 \\ 6 & 7 & 0 & 6 & 7 \\ 6 & 6 & 6 & 0 & 7 \\ 7 & 7 & 7 & 7 & 0 \end{bmatrix}$$

When setting now $\varepsilon = 6$ instead of 5, the resulting set is further reduced. According to the $\mathbf{A}_{\varepsilon=6}[8]$ below two more traces can be removed: $t_3[8]$ and $t_4[8]$. (Note that in the $\varepsilon = 5$ case these two cases were named previously as redundant.) The remaining three traces $T_{\varepsilon=6}[8] = \{dbafacfd, fhdhbehf, bhbfeggg\}$ still traverse 11 of the 13 transitions.

$$\mathbf{A}_{\varepsilon=6}[8] = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & \boxed{1} & 0 & 1 & 0 \\ \boxed{1} & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

When using $\varepsilon = 6$ from the beginning of the iteration cycle, the final trace set is $T_{\varepsilon=6} = \{dbafacfd, fhdhbehf, bhbfeggg\}$. Without the iterative cycle the resulting set would be $T_{\varepsilon=6} = \{dbafacfd, fhdhbehf\}$ with 9 transitions traversed.

The transition coverage is the same for both the $\varepsilon = 5$ and the $\varepsilon = 6$ cases. In step 3 one new transition is discovered, but the trace is dropped due to redundancy. Two new transitions are found in step 8 including the one dropped in step 3.

The total number of distance calculations is 20 in the $\varepsilon = 5$ case and 13 in the $\varepsilon = 6$ case. Without the iterative cycle it would be 28, hence matrix $\mathbf{A}$ is provided with less calculations. The gain depends on the relation between trace lengths and the total number of traces generated.

5 Empirical Analysis

This section gives simulation based evaluation of the – single-staged – string edit distance based test case selection method. A process from each of two systems, the `ISAP Manager Ini` process from the INRES [13] and the Conference

Table 2. Properties of the unfolded SDL systems and input parameters for the test selection methods

Name	States	Inputs	Outputs	Non-implicit transitions	Faults injected	Symbols used in mapping
INRES	3	5	4	7	25	8
Conference	5	11	3	19	78	22

Protocol [7] both represented in SDL, and test sets derived manually and automatically are used in this investigation. The result of the method proposed in this section are compared to the results of two different selection strategies based on pairing of test cases and test case requirements [8]. One of these two methods is the fault coverage based test selection method proposed in [2], where the test case requirements are considered to be faults injected systematically into the system according to the given fault model. The other is a transition coverage based method that regards the checking of a transition of an FSM as a test case requirement. Hence the two SDL systems are unfolded into FSMs.

Table 2 characterizes the two unfolded SDL systems. The INRES `ISAP Manager Ini` process' FSM consists of three states, has four inputs, a timer and four outputs. The total number of transitions is 15, the number of different faults that can be injected to that process according to the fault model proposed in [2] is 25. The input and output events are mapped to eight characters. The Conference Protocol limited to three users and at most one conference at a time has five states, eleven inputs and three outputs when unfolded into an FSM. 16 of the 55 transitions are non-implicit, and the same fault model yields 78 mutant systems. 22 characters are used in the mapping to represent the input and output events of this system.

Three test sets are used to evaluate the selection method proposed in this section. One is an automatically generated set of the eight cases that has already been used in the example in Sect. 2.1. The second is an automatically generated one of thirteen cases. These first two sets are executed against the `ISAP Manager Ini` process. A third automatically generated set is run againsts the Conference Protocol.

The results of simulations can be seen in Table 3. In the experiment on the INRES system's `ISAP Manager Ini` process with eight automatically generated test cases, the string edit distance based method and the transition coverage based method provide exactly the same three test cases. These three cases miss four faults that could be detected with the fault coverage method. As automatically generated cases are more likely to check implicit transitions their transition coverage is greater than in case of the manual cases. In the experiment with 13 manually generated cases, the string edit distance based method achieves a bigger reduction (13 to 6 vs. 13 to 8) than the transition coverage based method at the cost of not traversing all transitions. Both provide the same fault coverage, but miss four injected faults that could be found.

Table 3. Results of the selection experiments

Test set	System	Method	Number of selected cases	Number of faults detected	Transitions covered
Automatic (8)	INRES	String	4/8	19/25	11/13
		Transition	3/8	19/25	11/13
		Fault	4/8	23/25	11/13
Manual (13)	INRES	String	6/13	19/25	6/13
		Transition	8/13	19/25	10/13
		Fault	4/13	23/25	8/13
Automatic (40)	Conference	String	14/40	59/78	39/55
		Transition	11/40	59/78	40/55
		Fault	6/40	60/78	25/55

In case of the Conference Protocol all three methods provide nearly the same fault coverage. The biggest reduction is achieved by the fault coverage based method, but that traverses the least transitions.

In general according to the experiments shown above, the smallest reduced test and the highest fault coverage ration is provided by the fault based method. The highest transition coverage can be achieved by the transition coverage based method. The string based method provides results close to the transition coverage based method, but this is automatic and requires much less algorithmic steps to execute than the automatic fault based method.

6 Conclusions

This paper proposes an efficient approach for both single stage and iterative test selection processes. A method is provided to select a subset of a test set in polynomial time by searching for similar patterns of events. The approach builds on previous results of string edit distance based test selection methodology. We propose two selection criteria: one to identify the minimum cardinality of the target test set for a given a ε-cover, and an other to select the test cases that differ from each other as much as possible assuming the string distance based metric. An iterative procedure is given to reduce the computation requirement for long test cases, which is specially suited for maintaining test sets for regression testing. The operation of the algorithm is demonstrated through an example.

The method is compared with fault and coverage based test selection techniques using the sample systems INRES and Conference Protocol. The string edit distance based method provides similar fault detection capability as the transition coverage based selection, and requires less computation than the other two methods.

A generalization of the method is possible for test cases represented as a labeled, rooted, unordered trees. By calculating the distance matrix of such trees the method is applicable for test case selection without fundamental changes.

References

1. Lee, D., Yiannakakis, M.: Principles and methods of testing finite state machines – a survey. Proceedings of the IEEE 43(3), 1090–1123 (1996)
2. Kovács, G., Pap, Z., Le, V.D., Wu-Hen-Chang, A., Csopaki, G.: Applying mutation analysis to SDL specifications. In: Reed, R., Reed, J. (eds.) SDL 2003. LNCS, vol. 2708, pp. 269–284. Springer, Heidelberg (2003)
3. Wong, W.E., Restrepo, A., Qi, Y., Choi, B.: An EFSM-based test generation for validation of SDL specifications. In: AST 2008: Proceedings of the 3rd international workshop on Automation of software test, pp. 25–32. ACM, New York (2008)
4. Schmitt, M., Grabowski, J., Hogrefe, D., Koch, B.: Autolink – a tool for the automatic and semi-automatic test generation. In: Wolisz, A., Schieferdecker, I., Rennoch, A. (eds.) Formale Beschreibungstechniken für verteilte Systeme, Nr. 315, GMD-Studien. GMD-Forschungszentrum Informationstechnik GmbH (1997)
5. Tretmans, G., Brinksma, H.: Torx: Automated model-based testing. In: Hartman, A., Dussa-Ziegler, K. (eds.) First European Conference on Model-Driven Software Engineering, pp. 31–43 (2003)
6. Jard, C., Jeron, T.: TGV: Theory, principles and algorithms. In: 6th World Conference on Integrated Design and Process Technology, IDPT 2002 (2002)
7. Heerink, L., Feenstra, J., Tretmans, J.: Formal test automation: The conference protocol with phact. In: Ural, H., Probert, R.L., von Bochmann, G. (eds.) 13th IFIP International Conference on Testing of Communicating Systems (TestCom 2000), pp. 211–220. Kluwer Academic, Dordrecht (2000)
8. Harrold, M., Gupta, R., Soffa, M.: A methodology for controlling the size of a test suite. Transactions on Software Engineering and Methodology 2(3), 270–285 (1993)
9. Vuong, S.T., Alilovic-Curgus, J.: On test coverage metrics for communication protocols. In: von Bochmann, G., Dssouli, R., Das, A. (eds.) Proceedings of the IFIP TC6/WG6.1 Fourth International Workshop on Protocol Test Systems IV, pp. 31–45. North Holland, Amsterdam (1992)
10. Csöndes, T., Kotnyek, B., Szabó, J.: Application of heuristic methods for conformance test selection. European Journal of Operational Research 142(1), 203–218 (2001)
11. Williams, A., Probert, R.: Formulation of the interaction test coverage problem as an integer program. In: Schieferdecker, I., König, H., Wolisz, A. (eds.) Proceedings of the IFIP 14th international Conference on Testing Communicating Systems XIV. IFIP Conference Proceedings, vol. 210, pp. 283–298. Kluwer, B.V., Deventer (2002)
12. Feijs, L., Goga, N., Mauw, S., Tretmans, J.: Test selection, trace distance and heuristics. In: Schieferdecker, I., König, H., Wolisz, A. (eds.) Proceedings of the IFIP 14th international Conference on Testing Communicating Systems XIV. IFIP Conference Proceedings, vol. 210, pp. 267–282. Kluwer, B.V., Deventer (2002)
13. Ellsberger, J., Hogrefe, D., Sarma, A.: SDL Formal Object-oriented Language for Communicating Systems. Prentice-Hall, Englewood Cliffs (1997)
14. International Telecommunications Union: Recommendation Z.100 (11/07), Specification and Description Language (SDL), `http://www.itu.int/rec/T-REC-Z.100/en`
15. Wagner, R., Fischer, M.: The string-to-string correction problem. Journal of the ACM 21(1), 168–173 (1974)
16. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press and McGraw-Hill (2001)
17. Dell'Amico, M., Martello, S.: The k-cardinality assignment problem. Discrete Applied Mathematics 76(1), 103–121 (1997)