ELSEVIER

Information and Software Technology 42 (2000) 17–25

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.nl/locate/infsof

An optimal representative set selection method

J.G. Lee, C.G. Chung*

Department of Computer Science and Information Engineering, National Chiao Tung University, Hsin-Chu, Taiwan

Received 28 December 1998; received in revised form 13 March 1999; accepted 26 May 1999

Abstract

The optimal representative set selection problem is defined thus: given a set of test requirements and a test suite that satisfies all test requirements, find a subset of the test suite containing a minimum number of test cases that still satisfies all test requirements. Existing methods for solving the representative set selection problem do not guarantee that obtained representative sets are optimal (i.e. minimal). The enhanced zero–one optimal path set selection method [C.G. Chung, J.G. Lee, An enhanced zero–one optimal path set selection method, Journal of Systems and Software, 39(2) (1997) 145–164] solves the so-called optimal path set selection problem, and can be adapted to solve the optimal representative set selection problem by considering paths as test cases and components to be covered (e.g. branches) as test requirements.

Keywords: Testing; Optimal representative set; Test suite reduction

1. Introduction

In testing (hardware and software testing), testing objectives must be defined first. A testing objective can be considered a set of testing requirements (hereafter, referred to as requirements), and different testing objectives have different sets of requirements. Once a set of requirements is determined, test cases are designed to satisfy the requirements. A set of test cases that can collectively satisfy all requirements is called a test suite [2]. A typical method of constructing a test suite, which has been employed by most automated test case generators [3–7], is to construct a test case for each requirement. A test case designed specifically for a requirement may also satisfy other requirements. As a result, the constructed test suite may contain redundancy because some of its proper subsets may still satisfy all requirements.

Since the costs of executing test cases and managing test suites may often be quite significant, a test suite subset that can still satisfy all requirements is desirable. Such a subset is known as a representative set [8]. Assuming that the cost of executing and managing each test case is the same, a representative set with a minimum number of test cases is desirable and is called an optimal representative set. The optimal representative set selection problem is defined as given a test suite and a set of requirements, find an optimal representative set from the test suite. As mentioned in [2,8], the optimal representative set selection problem is NP-complete [9], and as mentioned in [2] it is equivalent to solving the set-covering problem [10].

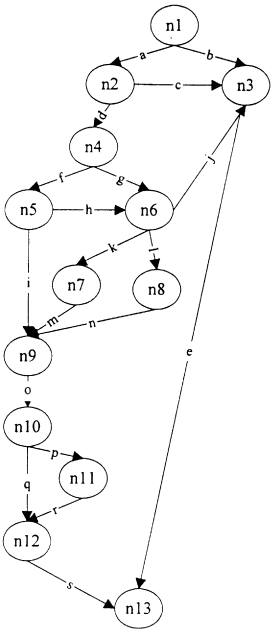
The greedy heuristic method [11] has conventionally been used to solve the set-covering problem; therefore, it can be used to obtain a representative set. The greedy heuristic method repeatedly selects a test case that satisfies the maximum number of unsatisfied requirements at a time until all requirements have been satisfied by a set of selected test cases. Two other studies [2,8] also provide heuristic methods. However, these heuristic methods do not always obtain optimal representative sets. That is, the optimal representative set selection problem has not yet been solved.

The optimal path set selection problem for structural program testing is defined as: given a complete path set P (P is defined as a set containing all paths) and a required coverage criterion, select the subset of P with the minimum number of paths needed to satisfy the required coverage criterion. A coverage criterion can be considered a set of components to be covered (e.g. statements or branches), and different coverage criteria have different sets of components to be covered. The set of components to be covered in the optimal path set selection problem can be considered the set of requirements to be satisfied in the optimal representative set selection problem. The paths and the complete path set in the optimal path set selection problem can be considered the test cases and the test suite in the optimal representative set

* Corresponding author. Tel.: + 886-03-5712121-54768; fax: + 886-03-5724176.

E-mail addresses: jglee@csie.nctu.edu.tw (J.G. Lee), cgchung@csie.nctu.edu.tw (C.G. Chung)

0950-5849/00/$ - see front matter
PII: S0950-5849(99)00052-X



(a) Program digraph

P={p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11},p_{12},p_{13},p_{14}}

p_1=be
p_2=ace
p_3=adgje
p_4=adfhje
p_5=adfioqs
p_6=adgkmoqs
p_7=adfhkmoqs
p_8=adglnoqs
p_9=adfhlnoqs
p_{10}=adfioprs
p_{11}=adgkmoprs
p_{12}=adfhkmoprs
p_{13}=adglnoprs
p_{14}=adfhlnoprs

(b) Complete path set

Fig. 1. A program digraph and its complete path set.

selection problem, respectively. The coverage relationship among paths and components in the optimal path set selection problem can be considered the satisfaction relationship among test cases and requirements in the optimal representative set selection problem. Therefore, the optimal path set selection problem in structural program testing is equivalent to optimal representative set selection problem, and because of this equivalence, optimal path set selection methods can be adapted to solve the optimal representative set selection problem.

There are two methods for solving the optimal path set selection problem, the zero–one optimal path set selection method [12] and the minimum flow method [13]. The former is more powerful than the latter because it can be applied to a large variety of constraints, cost functions, and coverage criteria, but the latter can only be applied to all-statements and all-branches coverage criteria [14,15]. The major drawback of the zero–one optimal path set selection method is that the computation is lengthy, and large programs may take ten or more hours because it is exponentially proportional to the number of candidate paths (i.e. the paths in the complete path set) and proportional to the number of components to be covered. Five reduction rules have been proposed [1] to overcome this drawback. These five rules enhance the zero–one optimal path set selection method and make it applicable to large programs. This paper introduces adaptation of the enhanced zero–one optimal path set selection method to solve the optimal representative set selection problem.

The rest of this paper is organized as follows. Section 2 introduces the enhanced zero–one optimal path set selection method. Section 3 shows adaptation of the enhanced zero–one optimal path set selection method to solve the optimal representative set selection problem, and how to apply this method in testing and maintenance stages. Section 4 provides an example illustrating the use of this method in testing and maintenance stages. Section 5 gives the conclusion.

2. Enhanced zero–one optimal path set selection method [1]

The enhanced zero–one optimal path set selection method consists of zero–one optimal path set selection method and five reduction rules. Section 2.1 introduces the zero–one optimal path set selection method. Section 2.2 shows how to apply the five reduction rules to reduce the long computation of zero–one optimal path set selection method, and lists the steps in the enhanced zero–one optimal path set selection method.

2.1. Zero–one optimal path set selection method [12]

In structural program testing, the structure of the program under test is mapped to a program digraph $G = (N, B)$, where N and B represent the node and branch sets, respectively. A node is a code segment executed sequentially while a branch directs transfer of control flow. Without loss of generality, it may be assumed that only one source node and one terminal node exist in the digraph. A path, starting at the source node and ending at the terminal node, is a sequence of nodes connected by branches. Using network methodologies such as node reduction [16], matrix self-multiplication [17], or linearly independent circuits

path \ branch	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
p_1=be	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p_2=ace	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p_3=adgje	1	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0
p_4=adfhje	1	0	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0
p_5=adfioqs	1	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	1	0	1
p_6=adgkmoqs	1	0	0	1	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1
p_7=adfhkmoqs	1	0	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0	1
p_8=adglnoqs	1	0	0	1	0	0	1	0	0	0	0	1	0	1	1	0	1	0	1
p_9=adfhlnoqs	1	0	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	0	1
p_{10}=adfioprs	1	0	0	1	0	1	0	0	1	0	0	0	0	0	1	1	0	1	1
p_{11}=adgkmoprs	1	0	0	1	0	0	1	0	0	0	1	0	1	0	1	1	0	1	1
p_{12}=adfhkmoprs	1	0	0	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1	1
p_{13}=adglnoprs	1	0	0	1	0	0	1	0	0	0	0	1	0	1	1	1	0	1	1
p_{14}=adfhlnoprs	1	0	0	1	0	1	0	1	0	0	0	1	0	1	1	1	0	1	1

Fig. 2. Branch-path coverage frequency matrix.

[18], the complete path set P of program digraph G can then be constructed. For example, consider the program digraph shown in Fig. 1(a), the corresponding complete path set P is shown in Fig. 1(b). A program with loops may include an extremely large number of paths. To this problem, the tester can limit loop iterations to some constant number.

Given a complete path set P and a required coverage criterion, a corresponding component-path coverage frequency matrix F can be generated showing the coverage frequency relationship among the paths and the components. The rows in F represent the paths in P, the columns in F represent the components to be covered, $F[i,j]$ represents the coverage frequency of the ith path over the jth component. For example, consider the complete path set P shown in Fig. 1(b). The coverage frequency matrix for the all-branches coverage criterion is shown in Fig. 2.

To illustrate the concept of this method, the all-branches coverage criterion is used as the required coverage criterion in all following discussions. Thus, the optimal path set selection problem can be defined as follows. In the complete path set P, which paths must be selected to guarantee that each branch in B is covered at least once and the number of selected paths is minimal? This is a decision problem because a decision as to whether to select each path must be made. In this example, since there are 14 paths in P, we can define 14 decision variables $x_i, i \in \{1,2,\ldots,14\}, x_i$ corresponds to path $p_i, x_i = 1$ if p_i is selected; 0, otherwise.

We first consider the requirement that each branch must be covered at least once. Take branch a as an example. Since this branch appears in p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, and p_{14}, one or more of these paths must be selected. This can be represented by the following constraint inequation:

$$x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} \geq 1, \text{ or}$$

$$1 \times x_2 + 1 \times x_3 + 1 \times x_4 + 1 \times x_5 + 1 \times x_6 + 1 \times x_7 + 1 \times x_8 + 1 \times x_9 + 1 \times x_{10} + 1 \times x_{11} + 1 \times x_{12} + 1 \times x_{13} + 1 \times x_{14} \geq 1$$

Following the same argument, the complete set of constraint inequations for all-branches coverage criterion is set up as follows (the constraint inequations are listed according to the order of branch a to branch s):

$$1 \times x_2 + 1 \times x_3 + 1 \times x_4 + 1 \times x_5 + 1 \times x_6 + 1 \times x_7 + 1 \times x_8 + 1 \times x_9 + 1 \times x_{10} + 1 \times x_{11} + 1 \times x_{12} + 1 \times x_{13} + 1 \times x_{14} \geq 1$$

$$1 \times x_1 \geq 1$$

$$1 \times x_2 \geq 1$$

$$1 \times x_3 + 1 \times x_4 + 1 \times x_5 + 1 \times x_6 + 1 \times x_7 + 1 \times x_8 + 1 \times x_9 + 1 \times x_{10} + 1 \times x_{11} + 1 \times x_{12} + 1 \times x_{13} + 1 \times x_{14} \geq 1$$

$$1 \times x_1 + 1 \times x_2 + 1 \times x_3 + 1 \times x_4 \geq 1$$

$$1 \times x_4 + 1 \times x_5 + 1 \times x_7 + 1 \times x_9 + 1 \times x_{10} + 1 \times x_{12} + 1 \times x_{14} \geq 1$$

$$1 \times x_3 + 1 \times x_6 + 1 \times x_8 + 1 \times x_{11} + 1 \times x_{13} \geq 1$$

$$1 \times x_4 + 1 \times x_7 + 1 \times x_9 + 1 \times x_{12} + 1 \times x_{14} \geq 1$$

$$1 \times x_5 + 1 \times x_{10} \geq 1$$

$$1 \times x_3 + 1 \times x_4 \geq 1$$

$$1 \times x_6 + 1 \times x_7 + 1 \times x_{11} + 1 \times x_{12} \geq 1$$

$$1 \times x_8 + 1 \times x_9 + +1 \times x_{13} + 1 \times x_{14} \geq 1$$

$$1 \times x_6 + 1 \times x_7 + 1 \times x_{11} + 1 \times x_{12} \geq 1$$

$$1 \times x_8 + 1 \times x_9 + 1 \times x_{13} + 1 \times x_{14} \geq 1$$

$$1 \times x_5 + 1 \times x_6 + 1 \times x_7 + 1 \times x_8 + 1 \times x_9 + 1 \times x_{10} + 1 \times x_{11} + 1 \times x_{12} + 1 \times x_{13} + 1 \times x_{14} \geq 1$$

$$1 \times x_{10} + 1 \times x_{11} + 1 \times x_{12} + 1 \times x_{13} + 1 \times x_{14} \geq 1$$

$$1 \times x_5 + 1 \times x_6 + 1 \times x_7 + 1 \times x_8 + 1 \times x_9 \geq 1$$

$$1 \times x_{10} + 1 \times x_{11} + 1 \times x_{12} + 1 \times x_{13} + 1 \times x_{14} \geq 1$$

$$1 \times x_5 + 1 \times x_6 + 1 \times x_7 + 1 \times x_8 + 1 \times x_9 + 1 \times x_{10} + 1 \times x_{11} + 1 \times x_{12} + 1 \times x_{13} + 1 \times x_{14} \geq 1$$

The above inequalities can be summarized as $\sum_{i=1}^{14} f_{ij} x_i \geq 1, \ \forall j = 1, 2, \ldots, 19$ where f_{ij} is the element of the branch-path coverage frequency matrix at the ith row and the jth column. It can also be represented in the following matrix form:

$$\begin{bmatrix} 0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&1&1&0&1&0&0&1&0&0&0&0&0&0&0&0&0 \\ 1&0&0&1&1&1&0&1&0&1&0&0&0&0&0&0&0&0&0 \\ 1&0&0&1&0&1&0&0&1&0&0&0&0&0&1&0&1&0&1 \\ 1&0&0&1&0&0&1&0&0&0&1&0&1&0&1&0&1&0&1 \\ 1&0&0&1&0&1&0&1&0&0&1&0&1&0&1&0&1&0&1 \\ 1&0&0&1&0&0&1&0&0&0&0&1&0&1&1&0&1&0&1 \\ 1&0&0&1&0&1&0&1&0&0&0&1&0&1&1&0&1&0&1 \\ 1&0&0&1&0&1&0&0&1&0&0&1&0&1&1&1&0&1&1 \\ 1&0&0&1&0&0&1&0&0&0&1&0&1&0&1&1&0&1&1 \\ 1&0&0&1&0&1&0&1&0&0&1&0&1&0&1&1&0&1&1 \\ 1&0&0&1&0&0&1&0&0&0&0&1&0&1&1&1&0&1&1 \\ 1&0&0&1&0&1&0&1&0&0&0&1&0&1&1&1&0&1&1 \end{bmatrix}^{\mathrm{T}} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Note that the first matrix on the left-hand side is the branch-path coverage frequency matrix shown in Fig. 2.

We next consider the requirement that the number of selected paths must be minimal. Since the value of the objective function $z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14}$ is the number of paths in the selected path set, z is the minimization target. Combining the objective function and constraint, the optimal path set selection problem is formulated as the following zero–one integer programming problem.

$$\min \text{ (minimize) } z = \sum_{i=1}^{14} x_i,$$

s.t. (subject to)

$$\begin{bmatrix} 0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&1&1&0&1&0&0&1&0&0&0&0&0&0&0&0&0 \\ 1&0&0&1&1&1&0&1&0&1&0&0&0&0&0&0&0&0&0 \\ 1&0&0&1&0&1&0&0&1&0&0&0&0&0&1&0&1&0&1 \\ 1&0&0&1&0&0&1&0&0&0&1&0&1&0&1&0&1&0&1 \\ 1&0&0&1&0&1&0&1&0&0&1&0&1&0&1&0&1&0&1 \\ 1&0&0&1&0&0&1&0&0&0&0&1&0&1&1&0&1&0&1 \\ 1&0&0&1&0&1&0&1&0&0&0&1&0&1&1&0&1&0&1 \\ 1&0&0&1&0&1&0&0&1&0&0&1&0&1&1&1&0&1&1 \\ 1&0&0&1&0&0&1&0&0&0&1&0&1&0&1&1&0&1&1 \\ 1&0&0&1&0&1&0&1&0&0&1&0&1&0&1&1&0&1&1 \\ 1&0&0&1&0&0&1&0&0&0&0&1&0&1&1&1&0&1&1 \\ 1&0&0&1&0&1&0&1&0&0&0&1&0&1&1&1&0&1&1 \end{bmatrix}^{\mathrm{T}} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

$x_i 0$ or $1, \ i \in \{1, 2, \ldots, 14\}$.

Consider the general form of program digraph $G = (N, B)$

and its complete path set $P = \{p_1, p_2, \ldots, p_m\}$. Define a decision variable array $X(m \times 1) = [(x_i)], i \in \{1, 2, \ldots, m\}$, where $x_i = 1$ if p_i is selected; 0, otherwise. Let $|B|$ denote the total number of branches in program digraph G and $F(m \times |B|)$ the branch-path coverage frequency matrix. The optimal path set selection problem is formulated as the following zero–one integer programming problem:

$$\min z = \sum_{i=1}^{m} x_i$$

$$\text{s.t. } F^{\mathrm{T}}X \geq I,\ x_i = 0 \text{ or } 1,\ i \in \{1, 2, \ldots, m\},\ I(|B| \times 1) = [1\ 1 \ldots 1]^{\mathrm{T}}.$$

From the above discussion we know that the zero–one integer programming method can be applied to all-branches coverage criterion. As shown in [12,14,15], this method can be applied to any structural program testing coverage criterion and an optimal solution is guaranteed.

This method can be extended [12,14,15] to handle the following two cases: (1) different paths have different test costs; and (2) only critical components need to be covered by selected paths. For the first case, we define a cost array $C = [(c_i)]$ where c_i represents the test cost of path p_i. Then, change the objective function to $z = CX = \sum_{i=1}^{m} c_i x_i$. For the second case, define a coverage requirement array $R = [(r_i)]$ where $r_i = 1$ if the ith component must be covered; 0, otherwise. Thus, the general optimal path set selection problem can be modeled as:

$$\min z = CX = \sum_{i=1}^{m} c_i x_i$$

$$\text{s.t. } F^{\mathrm{T}}X \geq R,\ x_i = 0 \text{ or } 1,\ i \in \{1, 2, \ldots, m\}.$$

The problem definitely has a solution because the program under test is assumed to be well-formed and $X = [11\ldots1]^{\mathrm{T}}$ is a solution in the worst case. Many effective algorithms are available to find a solution of the problem [19,20]. Among them, the Balas' zero–one additive algorithm [19] is considered to be the fastest. The complexity of the zero–one integer programming method is proportional to $(2^{|\mathrm{Paths}|} \times |\mathrm{Components}|)$ where $|\mathrm{Paths}|$ represents the number of candidate paths and $|\mathrm{Components}|$ represents the number of components to be covered. The computation time is exponentially proportional to $|\mathrm{Paths}|$ which is the major drawback of this method.

2.2. Reducing zero–one optimal path set selection method computation

The computation time required by the zero–one optimal path set selection method can be reduced if the size of the coverage frequency matrix (i.e. the number of candidate paths and the number of components to be covered) is reduced. We first consider every path's test cost to be the same and then different paths having different test costs.

To reduce the number of candidate paths and the number of components that must be covered, we have the following observations: (1) if a component does not have to be covered, it can be ignored during path selection; (2) if a component must be covered and is covered by only one path, the path that covers the component must be selected; (3) if every path covering a component, say, c_i, also covers another component, say, c_j, the requirement that c_i and c_j must be covered at least once can be reduced to c_i must be covered at least once; and (4) if a path, say, p_i, covers all the components covered by another path, say, p_j, and some additional components, then p_j can be ignored during path selection due to the existence of p_i. Based on these observations, five reduction rules to reduce both the number of candidate paths and the number of components to be covered were proposed [1]. The following notation is defined to illustrate the five reduction rules formally:

Symbol	Representation
$F(m \times n) = [(f_{ij})]$	Coverage frequency matrix
$R(n \times 1) = [(r_i)]$	Coverage requirement array, $r_i = 0$ or 1
Row_i	ith row matrix of F
Col_j	jth column matrix of F
$\lvert\mathrm{Row}i\rvert = \sum_{k=1}^{n} f_{ik}$	Summation of F's ith row non-zero entries
$\lvert\mathrm{Col}j\rvert = \sum_{k=1}^{m} f_{kj}$	Summation of F's jth column non-zero entries

The five reduction rules are described below.

Rule 1. Surely Satisfied Constraint: If a component does not have to be covered at least once, its corresponding constraint is surely satisfied and can thus be ignored. This rule can be expressed formally as follows:

If $r_i = 0,\ i \in \{1, 2, \ldots, n\}$, then (a) delete Col_i;

(b) delete the ith row of R (i.e. r_i).

Rule 2. Essential Path: A path is essential if and only if it alone covers one or more components. After an essential path, say, p_k, has been selected, the components covered by p_k can be ignored during subsequent computation because they have been covered by the selected path p_k. This rule can be expressed formally as follows:

If $|\mathrm{Col}_j| = f_{kj}$ and $f_{kj} \geq 1,\ k \in \{1, 2, \ldots, m\}$,

$j \in \{1, 2, \ldots, n\}$, then (a) set $x_k = 1$;

(b) delete Col_i and corresponding r_i, $\forall f_{ki} \geq 1$,

$i \in \{1, 2, \ldots, n\}$; and (c) delete Row_k.

Rule 3. Dominant Component/Dominant Column: Component c_i dominates component c_j if and only if every path covering c_i also covers c_j. In this situation, the requirement that c_j must be covered at least once can be ignored because c_i must be covered at least once. This rule can be

	Component 1	Component 2	Component 3	Test cost
p_1	1	1	1	100
p_2	1	1	0	10
p_3	1	0	1	10
p_4	0	1	1	10

Fig. 3. Coverage frequency matrix and path test costs.

expressed formally as follows:

If $|\mathrm{Col}_i| > 0,\ |\mathrm{Col}_j| > 0,$ and $f_{kj} \geq f_{ki}$,

$\forall k \in \{1, 2, \ldots, m\},$ then delete $\mathrm{Col}_j,$ delete r_j.

Rule 4. Dominant Path/Dominant Row: Path p_i dominates path p_j if and only if p_i covers all the components covered by p_j. In this situation, p_j can be ignored due to the existence of p_i. This rule can be expressed formally as follows:

If $f_{ik} \geq f_{jk},\ \forall k \in \{1, 2, \ldots, n\},$ then (a) set $x_j = 0$;

(b) delete Row_j.

Rule 5. Zero Path/Zero Row: Path p_i is a zero path if and only if it does not cover any component. This situation happens after Rule 2 has been applied and the essential path is dominant over another path. In this case the zero path can be directly deleted without affecting the problem solution. This rule can be expressed formally as follows:

If $f_{ij} = 0,\ \forall j \in \{1, 2, \ldots, n\},$ then (a) delete Row_i;

(b) set $x_i = 0$.

Among the five reduction rules, rule one should be applied first because it simplifies computation by removing unnecessary constraint inequations. Then, the other four reduction rules can be applied repeatedly to reduce the size of the coverage frequency matrix until none are applicable. A formal algorithm demonstrating application of the five reduction rules can be found in [1].

If different paths have different test costs, reduction rule 4 can not be directly applied. For example, consider the coverage frequency matrix and each path's test cost shown in Fig. 3. If reduction rule 4 is applied, p_1 will be selected and the total test cost will be 100. However, the optimal path set should be $\{p_2, p_3\}$, $\{p_2, p_4\}$ or $\{p_3, p_4\}$ because the total test cost would then be 20. Therefore, if different paths have different test costs reduction rule 4 should be modified as follows:

Rule 4′ . Dominant Path/Dominant Row: Path p_i dominates path p_j if and only if p_i covers all the components covered by p_j and the test cost of p_i is less than or equal to that of p_j. In this situation, p_j can be ignored due to the existence of p_i. Let *Cost*(*x*) be a function returning the cost of path *x*, this rule can be expressed formally as follows:

If $f_{ik} \geq f_{jk},\ \forall k \in \{1, 2, \ldots, n\}$ and $\mathrm{Cost}(p_i)$

$\leq \mathrm{Cost}(p_j),$ then (a) set $x_j = 0,$ (b) delete Row_j.

The five reduction rules can be combined with the zero–one optimal path set selection method to obtain an enhanced zero–one optimal path set selection method. The steps in the enhanced zero–one optimal path set selection method are listed as follows:

Step 1: Generate a component-path coverage frequency matrix *F*, define a coverage requirement array *R*, and define a cost array *C*.

Step 2: Apply the five reduction rules to reduce the size of *F*. If the reduced component-path coverage frequency matrix is empty (i.e. all columns have been deleted), stop; otherwise, go to step 3.

Step 3: Translate the reduced component-path coverage frequency matrix into constraint inequations (each column corresponding to a constraint inequation) and define the objective function that excludes decision variables with known values (i.e. 0 or 1).

Step 4: Solve the zero–one integer programming problem formulated in step 3 using an available software package (for example LINDO [21]).

The output of this method is the union of the essential paths selected by reduction rule 2 in step 2 and the paths selected by the software package used in step 4. Examples illustrating how to apply the enhanced zero–one optimal path set selection method can be found in [1].

3. Adapting the enhanced zero–one optimal path set selection method to solve the optimal representative set selection problem

To an optimal representative set selection problem, assume that the given test suite T contains m test cases, requirement set R contains n requirements, and each test case in T satisfies one or more requirements in R. An m by n satisfaction matrix S can be generated to represent the satisfaction relationship among the test cases in T and the requirements in R. The rows of S represent the test cases in T, the columns of S represent the requirements in R; $S[i, j] = 1$ if test case $t_i \in T$ satisfies requirement $r_j \in R$; 0, otherwise. Since each test case in T is either selected or not selected, we define a decision variable array, $X(\mathrm{m} \times 1) = [(x_i)]$, $i \in \{1, 2, \ldots, m\}$, where $x_i = 1$ if t_i is selected, 0, otherwise. Since the cost of each test case may be different, we define a cost array, $C = [(c_i)]$, where c_i represents the cost of test case $t_i \in T$. The optimal representative set selection problem is formulated as the following zero–one integer programming problem:

$$\min z = CX = \sum_{i=1}^{m} c_i x_i$$

$$\text{s.t. } S^{\mathrm{T}}X \geq I,\ \ x_i = 0 \text{ or } 1,\ \ i \in \{1, 2, \ldots, m\},\ \ I(n \times 1) = [1\ 1 \ldots 1]^{\mathrm{T}}.$$

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}	r_{17}	r_{18}	r_{19}
t_1	1	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0
t_2	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0	0	0	0	0
t_3	1	1	1	1	0	1	0	0	0	0	1	1	1	1	1	0	1	1	0
t_4	1	1	1	1	0	1	0	0	0	0	0	1	0	0	1	1	1	1	1
t_5	1	1	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
t_6	1	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1
t_7	1	1	1	1	1	0	1	0	1	1	0	0	0	0	1	0	0	0	0
t_8	1	1	1	1	0	1	0	0	0	0	1	1	1	1	0	1	0	0	1
t_9	1	1	1	1	1	1	0	1	0	1	0	1	0	0	1	0	1	1	0
t_{10}	1	1	1	1	1	1	1	0	1	1	1	0	1	1	0	0	0	0	0
t_{11}	1	1	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
t_{12}	1	1	1	1	1	1	1	0	1	1	0	1	0	0	1	0	1	1	0

Fig. 4. Satisfaction matrix

Thus, we can apply the enhanced zero–one optimal path set selection method to find an optimal representative set.

The optimal representative set selection method can be used in both the testing and maintenance stages. In the testing stage, a set of requirements R is defined first and then a test suite T is generated to satisfy all the requirements in R. The satisfaction relationship among the test cases in T and the requirements in R is analyzed and represented in a satisfaction matrix. If different paths have different costs, we define a cost array, $C = [(c_i)]$, where c_i represents the cost of path p_i. The optimal representative set selection method can then be applied to obtain an optimal representative set.

In the maintenance stage, requirements may be added or deleted due to program modifications. Existing test cases may not satisfy all new requirements after new requirements are added, and new test cases may have to be added. Some existing test cases may not satisfy any requirement after some existing requirements are deleted, so these test cases will have to be deleted. The satisfaction matrix S must then be updated to represent the satisfaction relationship among the test cases in the new test suite T and the requirements in the new requirement set R.

Programs should be re-tested after modification. If the modification only relates to some requirements in R, we can define a coverage requirement array, $R = [(r_i)]$, where $r_i = 1$ if the ith requirement is modification related, 0, otherwise [22,23]. We may then apply the optimal representative set selection method to find a minimal representative set to rerun.

4. Example

In this section, an example is used to illustrate application of the optimal representative set selection method to testing and maintenance stages. A system (either hardware or software) has been designed to satisfy 19 requirements, and 12 test cases have been designed to test for satisfaction of these requirements. The 19 requirements can be considered functional testing requirements that must be satisfied for software integration or functional testing requirements for hardware such as CPUs, hard disks, RAMs, floppy disks, etc. The corresponding satisfaction matrix is shown in Fig. 4.

During testing, we want to find a minimal representative set that satisfies the 19 requirements. Assuming the execution costs of all 12 test cases are the same, the objective is then to minimize $z = x_1 + x_2 + x_3 + \cdots + x_{12}$. The five reduction rules are applied to reduce the satisfaction matrix size. Since requirement r_1 dominates the requirements in $\{r_2, r_3, r_4\}$, r_5 dominates $\{r_1, r_{10}\}$, r_7 dominates $\{r_5, r_9\}$, r_{11} dominates $\{r_6, r_{13}, r_{14}\}$, r_{16} dominates $\{r_{12}, r_{19}\}$, and r_{17} dominates $\{r_{15}, r_{18}\}$, reduction rule 3 can be applied. As a result, the columns corresponding to the requirements in $\{r_2, r_3, r_4, r_1, r_{10}, r_5, r_9, r_6, r_{13}, r_{14}, r_{12}, r_{19}, r_{15}, r_{18}\}$ are deleted and the reduced satisfaction matrix is as shown in Fig. 5. Since test case t_1 dominates the test case in $\{t_7\}$, t_2 dominates $\{t_5, t_{11}\}$, t_4 dominates $\{t_6\}$, and t_{10} dominates $\{t_1\}$, reduction rule 4 can be applied and the reduced satisfaction matrix is then as shown in Fig. 6. Since no more reduction rules can be applied, the problem is reduced to:

$$\min z = x_2 + x_3 + x_4 + x_8 + x_9 + x_{10} + x_{12}$$

$$\text{s.t. } x_{10} + x_{12} \geq 1$$

$$x_2 + x_9 \geq 1$$

$$x_2 + x_3 + x_8 + x_{10} \geq 1$$

$$x_4 + x_8 \geq 1$$

$$x_3 + x_4 + x_9 + x_{12} \geq 1.$$

The LINDO software package is used to solve the reduced problem, and yields $\{t_2, t_8, t_{12}\}$, which means the obtained optimal representative set is $\{t_2, t_8, t_{12}\}$.

The satisfaction matrix shown in Fig. 4 may also be considered an updated satisfaction matrix for the maintenance stage. Assume that modification relates only to the requirements in $\{r_6, r_7, r_8, r_9, r_{10}, r_{11}, r_{12}\}$. Thus, a coverage requirement array is defined as $R = [0000011111110000000]^T$. The five reduction rules are applied to reduce the satisfaction matrix size. Since the requirements in $\{r_1, r_2, r_3, r_4, r_5, r_{13}, r_{14}, r_{15}, r_{16}, r_{17}, r_{18},$

	r_7	r_8	r_{11}	r_{16}	r_{17}
t_1	1	0	0	0	0
t_2	0	1	1	0	0
t_3	0	0	1	0	1
t_4	0	0	0	1	1
t_5	0	1	0	0	0
t_6	0	0	0	1	0
t_7	1	0	0	0	0
t_8	0	0	1	1	0
t_9	0	1	0	0	1
t_{10}	1	0	1	0	0
t_{11}	0	1	0	0	0
t_{12}	1	0	0	0	1

Fig. 5. Reduced satisfaction matrix (after applying reduction rule 3).

	r_7	r_8	r_{11}	r_{16}	r_{17}
t_2	0	1	1	0	0
t_3	0	0	1	0	1
t_4	0	0	0	1	1
t_8	0	0	1	1	0
t_9	0	1	0	0	1
t_{10}	1	0	1	0	0
t_{12}	1	0	0	0	1

Fig. 6. Reduced satisfaction matrix (after applying reduction rule 4).

	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}
t_1	0	1	0	1	1	0	0
t_2	1	0	1	0	1	1	0
t_3	1	0	0	0	0	1	1
t_4	1	0	0	0	0	0	1
t_5	0	0	1	0	1	0	0
t_6	1	0	0	0	0	0	1
t_7	0	1	0	1	1	0	0
t_8	1	0	0	0	0	1	1
t_9	1	0	1	0	1	0	1
t_{10}	1	1	0	1	1	1	0
t_{11}	0	0	1	0	1	0	0
t_{12}	1	1	0	1	1	0	1

Fig. 7. Reduced satisfaction matrix (after applying reduction rule 1).

	r_7	r_8	r_{11}	r_{12}
t_1	1	0	0	0
t_2	0	1	1	0
t_3	0	0	1	1
t_4	0	0	0	1
t_5	0	1	0	0
t_6	0	0	0	1
t_7	1	0	0	0
t_8	0	0	1	1
t_9	0	1	0	1
t_{10}	1	0	1	0
t_{11}	0	1	0	0
t_{12}	1	0	0	1

Fig. 8. Reduced satisfaction matrix (after applying reduction rule 3).

	r_7	r_8	r_{11}	r_{12}
t_2	0	1	1	0
t_3	0	0	1	1
t_9	0	1	0	1
t_{10}	1	0	1	0
t_{12}	1	0	0	1

Fig. 9. Reduced satisfaction matrix (after applying reduction rule 4).

$r_{19}\}$ need not be satisfied, reduction rule 1 can be applied and the reduced satisfaction matrix is then as shown in Fig. 7. Since requirement r_7 dominates the requirements in $\{r_9, r_{10}\}$ and r_{11} dominates $\{r_6\}$, reduction rule 3 can be applied and the reduced satisfaction matrix is then as shown in Fig. 8. Since test case t_1 dominates the test case in $\{t_7\}$, t_2 dominates $\{t_5, t_{11}\}$, t_3 dominates $\{t_4, t_6, t_8\}$, and t_{10} dominates $\{t_1\}$, reduction rule 4 can be applied and the reduced satisfaction matrix is then as shown in Fig. 9. Since no more reduction rules can be applied, the problem is reduced to:

$$\min z = x_2 + x_3 + x_9 + x_{10} + x_{12}$$

$$\text{s.t. } x_{10} + x_{12} \geq 1$$

$$x_2 + x_9 \geq 1$$

$$x_2 + x_3 + x_{10} \geq 1$$

$$x_3 + x_9 + x_{12} \geq 1.$$

The LINDO software package may then be used to obtain $\{t_2, t_{12}\}$, which indicates that only the test cases $\{t_2, t_{12}\}$ need to be rerun to verify the correctness of the modification.

5. Conclusion

The optimal representative set selection problem and the optimal path set selection problem both involve finding a minimum subset from a given set that satisfies given requirements, so they can be classified as set-covering problems. This means the enhanced zero–one optimal path set selection method, used in structural program testing, can be adapted to solve the optimal representative set selection problem. The proposed method can be applied in testing and maintenance stages. The proposed method can be applied even when different test cases have different test costs. A program based on the proposed method that can be executed on IBM compatible PC has been implemented. The optimal representative set can be obtained automatically within a reasonable time after the satisfaction matrix and objective function have been inputted.

References

[1] C.G. Chung, J.G. Lee, An enhanced zero–one optimal path set selection method, Journal of Systems and Software 39 (2) (1997) 145–164.

[2] T.Y. Chen, M.F. Lau, A new heuristic for test suite reduction, Information and Software Technology 40 (5-6) (1998) 347–354.

[3] R.A. DeMillo, A.J. Offutt, Constraint-based automatic test data generation, IEEE Transactions on Software Engineering 17 (9) (1991) 900–910.

[4] R. Gupta, M.L. Soffa, Employing static information in the generation of test cases, Software Testing, Verification and Reliability 3 (1993) 29–48.

[5] B. Korel, Automated software test data generation, IEEE Transactions on Software Engineering 16 (8) (1990) 870–879.

[6] C.V. Ramamoorthy, S.B.F. Ho, W.T. Chen, On the automated generation of program test data, IEEE Transactions on Software Engineering 2 (4) (1976) 293–300.

[7] W.T. Tsai, D. Volovik, T.F. Keefe, Automated test case generation for programs specified by relational algebra queries, IEEE Transactions on Software Engineering 16 (3) (1990) 316–324.

[8] M.J. Harrold, R. Gupta, M.L. Soffa, A methodology for controlling the size of a test suite, ACM Transactions on Software Engineering and Methodology 2 (3) (1993) 270–285.

[9] M.R. Garey, D.S. Johnson, in: V. Klee (Ed.), Computer and intractability, a guide to the theory of NP-completeness, Freeman, New York, 1979.

[10] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to algorithms, MIT Press, Cambridge, MA, 1990.

[11] D.S. Johnson, Approximation algorithms for combinatorial problems, Journal of Computer and System Sciences 9 (3) (1974) 256–278.

[12] H.S. Wang, S.R. Hsu, J.C. Lin, A generalized optimal path selection model for structural testing, Journal of Systems and Software, July (1989) 55–63.

[13] S.C. Ntafos, S.L. Hakimi, On path cover problems in digraphs and applications to program testing, IEEE Transactions on Software Engineering SE-5 (5) (1979) 520–529.

[14] J.C. Lin, C.G. Chung, S.C. Lo, On path selection model for structural program testing, Proceedings of the Fifth IASTED International Symposium, 1989, pp. 69–72.

[15] J.C. Lin, C.G. Chung, Zero–one integer programming model in path selection problem of structural testing, IEEE Proceeding COMPSAC 89, November (1989) 618–627.

[16] B. Beizer, Software testing techniques, Data System Analysts Inc., Pennsanken, New Jersey, 1984.

[17] N.F. Schneidewind, Application of program graphs and software development and testing, IEEE Transactions on Software Engineering August (1979) 192–198.

[18] T.J. McCabe, Complexity measure, IEEE Transactions on Software Engineering SE-2 (4) (1976) 308–320.

[19] M.M. Syslo, N. Deo, J.S. Kowalik, Zero–one integer programming, Prentice-Hall, Englewood Cliffs, NJ, 1983, pp. 100–116.

[20] F.S. Hillier, G.J. Lieberman, Introduction to operations research, 5, McGraw-Hill, New York, 1990.

[21] L. Schrage, User's manual for linear, integer, and quadratic programming with lindo release 5.0, The Scientific Press, South San Francisco, CA, 1991.

[22] K.F. Fischer, A test case selection method for the validation of software maintenance modifications, IEEE Proceeding COMPSAC 77, November (1977) 421–426.

[23] K.F. Fischer, F. Raji, A. Chruscicki, A methodology for re-testing modified software, Proceeding of National Telecommunication Conference, CS Press, Los Alamitos, CA, 1981, pp. B6.3.1–6.