

## 基于变迁的完全路径覆盖测试

刘继华<sup>1\*</sup>, 陈策<sup>2</sup>

(1. 吕梁学院 计算机科学与技术系, 山西 吕梁 033000; 2. 北京特种机电研究所 陆军装备软件测评中心, 北京 100012)

(\*通信作者电子邮箱 157598066@qq.com)

**摘要:**为解决基于状态节点搜索的完全路径覆盖所产生的测试用例数过多和难以实现连续测试的问题,提出了一种基于变迁的完全路径覆盖测试准则,并设计和实现了一种深度优先搜索与宽度优先搜索相结合的基于变迁完全路径覆盖测试用例自动生成算法。实验结果表明,基于变迁的完全路径覆盖准则比基于状态的完全路径覆盖准则更为严格,相应的算法可以产生更优的测试用例集,能更方便地完成软件的连续动态测试。

**关键词:**软件测试;状态图测试;完全路径覆盖;测试用例

**中图分类号:** TP311.5 **文献标志码:** A

### Complete path coverage testing based on change

LIU Ji-hua<sup>1\*</sup>, CHEN Ce<sup>2</sup>

(1. Computer Science and Technology Department, Luliang University, Luliang Shanxi 033000, China;

2. Beijing Institute of Special Machine and Electron, Army Equipment Software Testing Center, Beijing 100012, China)

**Abstract:** In order to solve the problem that the test cases generated from the complete path coverage based on the search on state node are too many and it is difficult to achieve continuous test, it is necessary to put forward a test criteria for complete path coverage based on change, design and implement an automatic generation algorithm (DWFS algorithm) of complete path coverage based on change, combining the depth first search and width first search. The experimental results show that the standards based on the change of the complete path coverage are more stringent than those based on the complete state coverage, and the corresponding DWFS algorithm can produce better test suite and can more easily accomplish software continuous dynamic test.

**Key words:** software testing; state chart test; complete path coverage; test case

## 0 引言

软件测试是软件质量保证的主要手段,按照是否根据代码进行测试,软件测试可分为黑盒测试和白盒测试;按照是否需要运行软件进行测试,软件测试又可分为静态测试和动态测试<sup>[1-2]</sup>。黑盒动态测试的主要依据是软件规格说明,状态图是软件规格说明的一种半形式化建模方法,基于状态图的测试是软件黑盒动态测试的研究方向之一,已有很多学者提出若干状态图覆盖准则<sup>[2-3]</sup>,文献[4]中也给出了满足完整转换路径覆盖的测试用例深度优先搜索算法,但这些成果中主要是以状态的覆盖来定义路径,但状态无法包含变迁所需要的事件和监护条件,以状态为基础的完全路径搜索算法所生成的测试用例数仍较多,无法满足软件实际测试的需要。本文通过研究状态图及其测试覆盖准则,对基于状态的完全路径定义进行了修改和完善,定义了一种基于变迁的完全路径覆盖准则,以状态变迁方阵来对状态图进行形式化描述,设计和实现了可以以状态变迁方阵为输入,以基于变迁的完全路径集合为输出的深度优先搜索与宽度优先搜索相结合的测试用例自动生成算法模型(Combined Arithmetic of Depth First Search and Width First Search, DWFS)。实验结果表明,本文所提出的基于变迁的完全路径覆盖准则以及相应的 DWFS 算法可以生成比基于状态的完全路径覆盖准则及其搜索算法更优的测试用例集,能更好地覆盖状态、事件以及监护条件,也

更适合实际测试时需要连续对软件进行测试的客观情况。

## 1 状态图及其测试覆盖准则

### 1.1 状态图的定义

状态图刻画了一个系统的所有可能状态及引起状态转移的事件,它可以对系统的动态行为进行建模。状态图可以完整地表示出系统的状态,一系列状态转换构成了对系统的一个完整的应用,利用状态图进行测试用例设计可以模拟出软件的实际使用。

**定义 1**<sup>[5]</sup> 状态图形式化为一个六元组  $SC: (S, T, E, G, A)$ , 其中:

$S$  是系统状态的集合,  $S_0 \in S$  为系统的初始状态,  $S_e \subseteq S$  表示系统的终止状态, 系统的初始状态只有一个, 而终止状态可以有多个;

$T \subseteq S \times S$ , 表示状态变迁的集合, 一个变迁  $T_i \in T$  连接着两个状态  $S_s$  和  $S_t$ ,  $S_s$  为转换的源状态,  $S_t$  为转换的目标状态, 可分别记为  $T_i: S_s$  和  $T_i: S_t$ ;

$E$  为事件的集合, 事件分为触发器事件和一般事件, 触发器事件是能引起状态转换的事件, 一般事件是引起动作执行但不发生系统状态转换的事件;

$G$  为监护条件的集合, 监护条件是一个布尔表达式, 当触发器事件发生时, 监护条件被赋值, 为 true 时可导致转换的发生, 为 false 时则不会引起转换的发生;

收稿日期: 2012-05-04; 修回日期: 2012-06-27。 基金项目: 吕梁学院校内自然科学基金资助项目(ZRXN201216)。

作者简介: 刘继华(1975-), 女, 山西稷山人, 讲师, 硕士, 主要研究方向: 软件工程; 陈策(1975-), 男, 山西万荣人, 副研究员, 博士, 主要研究方向: 软件工程。

$A$  是动作的集合,动作与事件相关联,系统每进入一个状态时会引发进入动作,离开一个状态时会引发退出动作,处于某个状态时会一直执行某个动作,状态发生转换时也会引发转换动作。

**定义2** 状态图  $SC$  的一条路径是一个有向的由状态变迁构成的序列,记为  $Path: \langle Si_1, Tj_1, Si_2, Tj_2, Si_k, Tj_k, Si_{k+1}, \dots, Tj_x, Sj_y \rangle$ , 其中,  $Si_k \in S$ , 而  $Tj_k \in T, i_k \in \{1, 2, \dots, n\} (1 \leq k \leq y), j_l \in \{1, 2, \dots, m\} (1 \leq l \leq y), x$  和  $y$  均为正整数,  $k$  与  $k+1$  仅表示在路径中  $Si_{k+1}$  是  $Si_k$  后继位置,并不限定  $i_{k+1}$  与  $i_k$  之间的关系,二者可以相同,也可以不同。

**定义3** 状态图  $SC$  的一条完全路径是一个由状态变迁构成的路径,其第一个位置是状态图的起始位置  $S_0$ , 最后一个位置是:

- 1) 没有后续变迁的状态;
- 2) 循环返回状态,即在状态变迁过程中,又回到曾经经过的状态;
- 3) 显式指出的终止状态。

## 1.2 状态图的测试覆盖准则

基于状态图的测试,可以定义5种级别的测试覆盖准则:状态覆盖准则  $S_{cv}$ 、变迁覆盖准则  $T_{cv}$ 、变迁对覆盖准则  $TT_{cv}$ 、变迁监护条件覆盖准则  $G_{cv}$ 、完全路径覆盖准则  $P_{cv}^{[3]}$ 。

**定义4** 状态覆盖准则  $S_{cv}$ : 令  $TS$  为测试用例集,对于  $\forall s \in S, \exists ts \in TS$ , 使得测试用例  $ts$  运行时能够覆盖状态  $s$ 。

状态图中的每一个状态都是可达的,因而每一个状态被访问一次是很容易做到的。 $S_{cv}$  是状态图测试覆盖准则中最容易被满足、也是要求最低的覆盖准则,所需的测试用例数往往也是最少的。

**定义5** 变迁覆盖准则  $T_{cv}$ : 对于  $\forall ts \in T, \exists ts \in TS$ , 使得测试用例  $ts$  运行时能够覆盖变迁  $ts$ 。

变迁覆盖准则先使系统到达某一当前状态,如果一个变迁的事件被接受,并且这个变迁的监护条件为真,则这个变迁被激活。对于任意一个变迁  $ts$ , 可以表示为源状态  $S_s$  和目的状态  $S_t$ , 因此满足变迁覆盖准则  $T_{cv}$  一定满足状态覆盖准则  $S_{cv}$ 。

**定义6** 变迁对覆盖准则  $TT_{cv}$ : 对于每一对相邻的变迁,测试用例集  $TS$  必须包含遍历每一对从输入到输出的变迁。

假设一个状态有  $M$  个输入变迁,  $N$  个输出变迁,则需要  $M \times N$  个测试用例,变迁对覆盖准则显然包含了变迁覆盖准则。

**定义7** 变迁监护条件覆盖准则  $G_{cv}$ : 要求状态图中的每个变迁至少被覆盖一次,且与变迁相关的监护条件取真和取假至少覆盖一次。

变迁监护条件覆盖准则  $G_{cv}$  可以检测合法和非法变迁情况,满足监护条件为真时产生变迁为合法变迁,满足监护条件为假时产生变迁则为非法变迁。当一个变迁没有监护条件时,可以认为它的监护条件是一个永真式,此时针对该条变迁的覆盖准则与变迁监护条件覆盖准则是相同的。

**定义8** 完全路径覆盖准则  $P_{cv}$ : 设  $SPS$  是状态图中的完全路径集合,对于  $\forall sp \in SPS$ , 必须  $\exists ts \in TS$ , 使得测试用例  $ts$  运行时能够覆盖  $sp$ 。

完全路径覆盖准则覆盖了状态图中的所有完全路径,而每个完全路径可以代表对系统的一次实际使用,完全路径覆盖准则包含了状态覆盖准则和变迁覆盖,因此在基于状态图的测试中,使用最多的是完全路径覆盖准则。

这5种级别测试充分性准则之间的包含关系为:  $P_{cv} \supseteq T_{cv} \supseteq S_{cv}, G_{cv} \supseteq T_{cv} \supseteq S_{cv}, TT_{cv} \supseteq T_{cv} \supseteq S_{cv}$ , 但  $P_{cv}, G_{cv}, TT_{cv}$  之

间则没有包含关系。

## 1.3 完全路径树生成算法

从状态图中的初始状态  $S_0$  出发,利用广度优先搜索,可以建立一个状态变迁树,从该树的每个叶子节点出发向前追溯到根节点(即初始状态  $S_0$ ),便形成一条状态图的完全路径,将这样的状态变迁树称为状态图的完全路径树。生成完全路径树的基本过程<sup>[4-5]</sup>是:

算法1 完全路径树的广度优先搜索算法。

- 1) 以初始状态  $S_0$  作为树的根节点。
- 2) 从初始节点  $S_0$  开始分析其状态迁移情况,将从  $S_0$  出发的所有状态迁移标记为树的分支,将状态迁移到达的状态标记为初始节点的孩子节点。
- 3) 对孩子节点进行状态迁移分析,若某个孩子节点代表的状态在先前的搜索过程中出现过或者该节点为状态图的终止节点,则把该孩子节点标记为叶子节点;否则按照步骤2)的方式继续对其他孩子节点进行扩展。
- 4) 判断所有的状态和迁移是否都已经在树中出现,若是则结束在状态图中的搜索,否则返回2)继续。

## 2 基于变迁的完全路径覆盖测试

### 2.1 基于变迁的完全路径覆盖准则

在定义3中,一条完全路径的最后一个位置的约束是要求处于该位置的状态要么是终止状态,要么是在完全路径搜索过程中出现过的状态,或者是没有后续变迁的状态。当一个状态没有后续变迁的时候并且该状态不是终止状态,那么软件将始终处于该状态中,这实际上对应于软件的一个死机错误。因此,完全路径的最后一个状态或者是终止状态或者是完全路径搜索过程中出现过的状态<sup>[6]</sup>。

经过对完全路径覆盖准则的研究,发现以状态节点为基础的完全路径覆盖强调的是状态的覆盖,并且在对完全路径的搜索过程中,无论是宽度优先搜索还是深度优先搜索,得到的完全路径数都只能是  $|T| - |S| + 2$  个。在实际工程测试时,测试人员总是希望能够连续完成对软件的执行驱动来实现测试,当有事件发生并且监护条件满足时,软件的状态才会发生变化,每个变迁都需要有事件和监护条件相对应,而状态则没有这样的要求<sup>[7]</sup>。因此,以变迁为基础来定义完全路径,能够更好地覆盖状态、事件以及监护条件。

**定义9**<sup>[8]</sup> 状态图  $SC$  的一条基于变迁的完全路径是一个由状态变迁构成的路径,其第一个位置是状态图的起始位置  $S_0$ , 最后一个位置  $Sj_y$  是:

- 1) 显式指出的终止状态,即  $Sj_y = S_e$ ;
- 2) 由  $Sj_y$  作为目的状态的变迁  $T_{jk}$  在本完全路径中或其他已搜索出的完全路径中出现过。

根据上述定义和完全路径覆盖准则  $P_{cv}$  的要求,也可以定义基于变迁的完全路径覆盖准则。

**定义10** 基于变迁的完全路径覆盖准则  $TP_{cv}$ : 设  $TPS$  是状态图中的完全路径集合,对于  $\forall tp \in TPS$ , 必须  $\exists ts \in TS$ , 使得测试用例  $ts$  运行时能够覆盖  $tp$ 。

### 2.2 基于变迁的完全路径 DWFS 算法

要实现状态图的测试用例自动生成,首要的一个问题就是状态图的形式表示问题,定义一个状态变迁方阵:

$$ST_{n \times n} = (st_{ij}); n = |S|, 0 \leq i, j \leq n-1$$

其中:  $st_{ij} = t_k, t_k \in T$ 。用方阵  $ST$  就可以表示出完整的状态图。在  $ST$  中,  $st_{ij}$  表示由状态  $s_i$  变迁到状态  $s_j$ , 其对应的变迁是  $t_k$ ,

当  $st_{ij}$  为 0 时,表示由状态  $s_i$  无法变迁到状态  $s_j$ ,  $ST$  中不为 0 的元素个数为  $|T|$ 。

利用  $ST_{n \times n}$  作为输入,设计了满足基于变迁完全路径覆盖的以深度优先搜索与宽度优先搜索相结合的测试用例自动生成算法 DWFS,其基本思想是从  $ST_{n \times n}$  的第 0 行开始搜索第一个不为 0 的变迁  $st_{ij}$ ,然后转到第  $j$  行,查找不等于 0 的  $st_{jk}$ ,当变迁  $st_{jk}$  的目标状态  $s_k$  为终止状态  $s_e$  或者变迁  $st_{jk}$  已在查找到的路径中出现过,则形成一条基于变迁的完全路径;然后返回第 0 行重新开始,查找下一条基于变迁的完全路径,在返回第 0 行重新开始时,如果  $st_{ij}$  已出现,则不再使用该  $st_{ij}$ ;当所有的变迁均已在路径中出现后,便形成了所有的测试用例。

算法 2 基于变迁的完全路径覆盖 DWFS 算法。

输入:状态变迁方阵  $ST_{n \times n}$ 。

输出:基于变迁的完全路径集合  $TPS$ 。

1) 初始化,令  $i = 0, j = 0, TPS = \emptyset$ , 完全路径  $tp = \emptyset$ , 变迁集  $T$  初始化为  $ST_{n \times n}$  中所有不为 0 的  $st_{ij}$ 。

2) 判断  $st_{ij}$  是否为 0,若  $st_{ij} = 0, j = j + 1$ , 返回 2) 继续判断;否则,转向 3)。

3) 将  $s_i \cdot t_k \cdot s_j$  加入到完全路径  $tp$  中,即  $tp \leftarrow tp + s_i \cdot t_k \cdot s_j$ , 将  $t_k$  从变迁集  $T$  中删除,即  $T \leftarrow T - t_k$ , 并且令  $st_{ij} = t_k = 0$ 。

4) 判断  $s_j = s_e \parallel t_k \in TPS$ , 若是,则完全路径  $tp$  已形成,将其加入到完全路径集合  $TPS$  中,即  $TPS \leftarrow TPS + tp$ , 继续 5); 否则,  $i = j, j = 0$ , 返回 2)。

5) 判断  $T = \emptyset$ , 若是,则结束输出  $TPS$ ; 否则,令  $i = 0, j = 0, tp = \emptyset$ , 返回 2), 搜索下一条基于变迁的完全路径。

### 3 实验结果及分析

#### 3.1 模型实例<sup>[9]</sup>

图 1 是某系统截获目标时的状态图。在该图中,  $S1$  为状态图的起点(初始态),  $S5$  为状态图的终点(终态)。

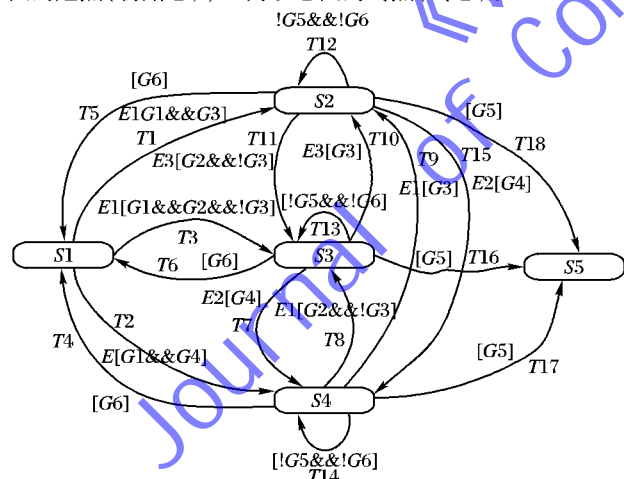


图 1 某系统截获目标状态图

状态集合  $S$  为  $\{S1, S2, S3, S4, S5\}$ ,  $S1$  为目标搜索状态,  $S2$  为红外截获状态,  $S3$  为电视截获状态,  $S4$  为雷达截获状态,  $S5$  为已截获目标状态。

事件集合  $E$  为  $\{E1, E2\}$ ,  $E1$  为光电截获按钮触发器事件,  $E2$  为雷达截获触发器事件。

监护条件集合  $G$  为  $\{G1, G2, G3, G4, G5, G6\}$ ,  $G1$  为是否选定了目标,  $G2$  为电视是否为开机状态,  $G3$  为红外是否为开机状态,  $G4$  为雷达是否为开机状态,  $G5$  为目标是否在截获范围内,  $G6$  为截获超时。

变迁集合为  $T = \{T_i \mid i = 1, 2, \dots, 18\}$ 。

#### 3.2 基于状态的完全路径树生成

利用算法 1 所确定的完全路径数生成步骤,图 1 所示的状态图所对应的完全路径树如图 2 所示,在图中,叶子节点上标识的数字指出了完全路径被找到的顺序。

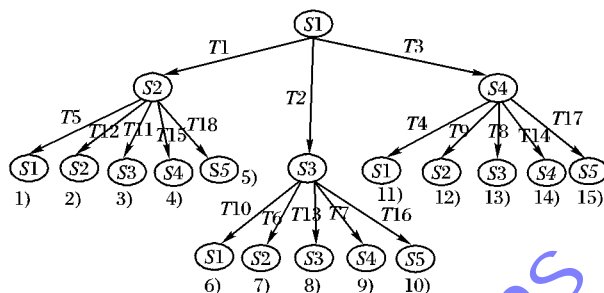


图 2 截获目标状态完全路径树

从完全路径树的叶子节点向前回溯,可得到满足完全路径覆盖的完全路径集为:1)  $S1 \rightarrow T1 \rightarrow S2 \rightarrow T5 \rightarrow S1$ ; 2)  $S1 \rightarrow T1 \rightarrow S2 \rightarrow T12 \rightarrow S2$ ; 3)  $S1 \rightarrow T1 \rightarrow S2 \rightarrow T11 \rightarrow S3$ ; 4)  $S1 \rightarrow T1 \rightarrow S2 \rightarrow T15 \rightarrow S4$ ; 5)  $S1 \rightarrow T1 \rightarrow S2 \rightarrow T18 \rightarrow S5$ ; 6)  $S1 \rightarrow T2 \rightarrow S3 \rightarrow T6 \rightarrow S1$ ; 7)  $S1 \rightarrow T2 \rightarrow S3 \rightarrow T10 \rightarrow S2$ ; 8)  $S1 \rightarrow T2 \rightarrow S3 \rightarrow T13 \rightarrow S3$ ; 9)  $S1 \rightarrow T2 \rightarrow S3 \rightarrow T7 \rightarrow S4$ ; 10)  $S1 \rightarrow T2 \rightarrow S3 \rightarrow T16 \rightarrow S5$ ; 11)  $S1 \rightarrow T3 \rightarrow S4 \rightarrow T4 \rightarrow S1$ ; 12)  $S1 \rightarrow T3 \rightarrow S4 \rightarrow T9 \rightarrow S2$ ; 13)  $S1 \rightarrow T3 \rightarrow S4 \rightarrow T8 \rightarrow S3$ ; 14)  $S1 \rightarrow T3 \rightarrow S4 \rightarrow T14 \rightarrow S4$ ; 15)  $S1 \rightarrow T3 \rightarrow S4 \rightarrow T17 \rightarrow S5$ 。

得到完全路径集后,将每个完全路径中的变迁用触发器事件和监护表达式替换,便得到测试用例所需的测试数据取值<sup>[10]</sup>。

#### 3.3 基于变迁的完全路径生成

图 1 状态图所对应的变迁转换矩阵  $ST_{5 \times 5}$  见表 1 所示。

表 1 状态转换矩阵  $ST$

源状态	目标状态				
	S1	S2	S3	S4	S5
S1	—	T1	T2	T3	—
S2	T5	T12	T11	T15	T18
S3	T6	T10	T13	T7	T16
S4	T4	T9	T8	T14	T17
S5	—	—	—	—	—

利用本文所提出的 DWFS 算法,可以得到的基于变迁的完全路径集为:

$TP1 = \langle S1 \ T1 \ S2 \ T5 \ S1 \ T2 \ S3 \ T6 \ S1 \ T3 \ S4 \ T4 \ S1 \rangle$

$TP2 = \langle S1 \ T1 \ S2 \ T12 \ S2 \ T4 \ S3 \ T10 \ S2 \ T15 \ S4 \ T9 \ S2 \ T18 \ S5 \rangle$

$TP3 = \langle S1 \ T1 \ S2 \ T12 \ S2 \ T4 \ S3 \ T13 \ S3 \ T7 \ S4 \ T8 \ S3 \ T16 \ S5 \rangle$

$TP4 = \langle S1 \ T1 \ S2 \ T12 \ S2 \ T4 \ S3 \ T13 \ S3 \ T7 \ S4 \ T14 \ S4 \ T17 \ S5 \rangle$

该路径集的路径总数为 4,从每个基于变迁的完全路径可以得到一个测试用例,测试用例共 4 个。而如果用基于状态的完全路径覆盖,则无论是采用何种搜索算法,得到的完全路径集的路径数均为 15;在基于状态的完全路径覆盖中,出现的状态数为 45,而在基于变迁的完全路径覆盖中,出现的状态数则为 29;二者出现的变迁数则是相同的<sup>[11]</sup>。对比情况见表 2。

表 2  $P_{cv}$  与  $TP_{cv}$  的对比分析

覆盖准则	用例数	状态数	变迁数
$P_{cv}$	15	45	18
$TP_{cv}$	4	29	18

(下转第 3081 页)



变换算法。

图 3 为四叉树深度为 8, 栅格数为  $256 \times 256$ , 随机生成 200 个生长源的加权 Voronoi 图。

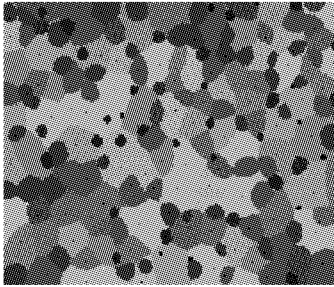


图 3 加权 Voronoi 图

#### 4 结语

本文提出基于四叉树结构的加权 Voronoi 图膨胀生成算法, 通过对栅格空间进行层次划分, 以时间消耗值取代加权距离计算生长源的占领区域。首先, 四叉树结构将空间面片进行剖分, 较好地反映了生长源的空间分布及疏密程度。其次, 算法效率较高, 时间复杂度低, 可以实现生长源的快速膨胀, 对生长源的数量与空间分布的敏感度较低。采用基于栅格空间的方法, 可扩展用于平面内任意生长源的加权 Voronoi 图构建, 以及空间混合实体(点、线、面混合)加权 Voronoi 图的构建, 具有较强的实用性和扩展性。

加权 Voronoi 图的生长精度受到栅格大小的限制, 如何有效平衡加权 Voronoi 图生长精度与四叉树节点数是今后研究的重点。顾及到四叉树节点之间的邻域关系, 进而提高最近生长源的检索效率, 以及考虑生长源权重的面片剖分是后继有待研究的问题。

#### 参考文献:

- [1] 陈军. Voronoi 动态空间数据模型[M]. 北京: 测绘出版社, 2002.
- [2] GONG YONGXI, LI GUICAI, TIAN YUAN. A vector-based algorithm to generate and update multiplicatively weighted Voronoi diagrams

for points, polylines, and polygons [J]. Computers & Geosciences, 2011, 42(9): 118 - 125.

- [3] GONG PINLIANG. Generating and updating multiplicatively weighed Voronoi diagrams for point, line and polygon features in GIS [J]. Computer & Geosciences, 2008, 34(4): 411 - 421.
- [4] 张有会. 加权 Voronoi 图画法的研究[J]. 计算机科学, 2001, 28(6): 126 - 130.
- [5] LETSCHER D. Vector weighted Voronoi diagram and delaunay triangulation [J]. Canadian Conference on Computation Geometry, 2007, 8(20/21/22): 165 - 169.
- [6] 张继忠, 胡艳, 马永强. 基于 Delaunay 三角网剖分生成 Voronoi 图算法[J]. 计算机应用, 2010, 30(1): 75 - 77.
- [7] 王海军, 邓羽, 张文婷. 利用元胞自动机和遗传算法的 Voronoi 图生成[J]. 武汉大学学报: 信息科学版, 2010, 35(7): 778 - 881.
- [8] REITSMA R, TRUBIN S, MORTENSEN E. Weight - proportional space partitioning using adaptive voronoi diagrams [J]. Geoinformatica, 2007, 11(3): 383 - 405.
- [9] RICCA F, SCOZZARI A, SIMEONE B. Weighted Voronoi region algorithms for political districting [J]. Mathematical and Computer Modeling, 2008, 48(9/10): 1468 - 1477.
- [10] CALVÃO L C, NOVAES A G N, de CURSI S J E, et al. A multiplicative-weighted Voronoi diagram approach to logistics districting [J]. Computers & Operations Research, 2006, 33(1): 93 - 114.
- [11] WANG CAOAN, TSIN Y H. Finding constrained and weighted Voronoi diagram in the plane [J]. Computational Geometry, 1998, 10(2): 89 - 104.
- [12] AURENHAMMER F, EDELSBRUNNER H. An optimal algorithm for constructing the weighted Voronoi diagram in the plane [J]. Pattern Recognition, 1984, 17(2): 251 - 257.
- [13] 李佳田, 陈军, 赵仁亮, 等. 基于线性四叉树结构的 Voronoi 图反向膨胀生成方法[J]. 测绘学报, 2008, 37(2): 236 - 242.
- [14] 谢顺平, 王结臣, 冯学智, 等. 基于节点逼近提取的平面点集 Voronoi 图构建算法[J]. 测绘学报, 2007, 36(4): 436 - 442.
- [15] 赵慧, 宋星. 基于线性四叉树的快速邻域查算法[J]. 计算机工程与设计, 2007, 28(18): 4333 - 4335.

(上接第 3077 页)

#### 4 结语

基于变迁的完全路径覆盖与基于状态的完全路径覆盖相比, 可以生成更少的测试用例, 同时对状态的测试强度则远高于基于状态的完全路径覆盖。以状态转换矩阵作为状态图的形式描述, 利用本文所提出的深度优先搜索为主, 宽度优先搜索为辅的测试用例自动生成算法 DWFS 可以自动生成满足变迁完全路径覆盖测试用例集, 已在多个实际的软件测试中得到实际应用, 具有一定的推广应用价值。后续的工作是将该项成果开发成更为实用的软件测试工具, 实现可视化状态图建模和测试用例数据自动生成, 增强其工程应用效果。

#### 参考文献:

- [1] WILLIAM E L. Software testing and continuous and quality improvement[M]. [S. l.]: Florida CRC Press, 2000.
- [2] 单锦辉, 姜瑛, 孙萍. 软件测试研究进展[J]. 北京大学学报: 自然科学版, 2005, 41(1): 134 - 145.
- [3] 占学德, 缪淮扣. 基于 UML 状态图测试的充分性准则[J]. 计算机科学, 2005, 32(5): 230 - 235.

- [4] 江曼, 王天青, 潘金贵. 基于 UML 状态图的面向对象软件测试用例生成[J]. 计算机科学, 2006, 33(6): 284 - 286.
- [5] CHOW T. Testing software designs modeled by finite-state machines [J]. IEEE Transactions on Software Engineering, 1978, SE-4(3): 178 - 187.
- [6] XIONG YIWEI, LIU SHAOYING. Criteria for generating specification based tests[C]// Fifth IEEE International Conference on Engineering of Complex Computer Systems. Washington, DC: IEEE Computer Society, 1999: 119 - 129.
- [7] 聂鹏, 耿技, 秦志光. 软件测试用例自动生成算法综述[J]. 计算机应用研究, 2012, 29(2): 402 - 405.
- [8] 屈波, 聂长海, 徐宝文. 基于测试用例设计信息的回归测试优先级算法[J]. 计算机学报, 2008, 31(3): 81 - 89.
- [9] 毛澄映, 卢炎生. 构件软件回归测试用例选择策略[J]. 计算机研究与发展, 2006, 43(10): 91 - 98.
- [10] 章晓芳, 徐宝文, 聂长海, 等. 一种基于测试需求约简的测试用例集优化方法[J]. 软件学报, 2007, 18(4): 821 - 831.
- [11] 张娟. 软件测试中测试用例复用的研究[D]. 上海: 上海大学, 2012.
- [12] 崔应霞. 组合测试技术的研究与应用[D]. 合肥: 安徽大学, 2011.