**Computer Science and Application** 计算机科学与应用, 2019, 9(7), 1245-1254
Published Online July 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.97140

Hans汉斯

# An Improved Technique for Generating Test Cases with Depth-First Search Algorithms

**Tingting Wang[1], Yong Hu[2], Chenguang Wang[1], Lei Hou[1], Yun Liu[1], Wei Meng[1]**

[1]Beijing Jinghang Computing and Communication Institute, Beijing
[2]Beijing Special Mechanical and Electrical Engineering Institute, Beijing
Email: 765176342@qq.com

Received: Jun. 27th, 2019; accepted: Jul. 8th, 2019; published: Jul. 15th, 2019

## Abstract

**Software test is an effective method to ensure the quality of software; the test case generation is the key of software testing, in order to guarantee the quality of the system software, make up for the inadequacy of the method to generate test cases, improve the accuracy and integrity of the generated cases. Taking some software of aerospace as the typical representative and application object, the test case generation technology at home and abroad is studied to explore the method of business control flow, data flow and the tree structure view. Combined with the depth first search algorithm and the actual application requirements, the traditional depth-first search algorithm is optimized to improve, make its rapid search effective test cases, and cover with test requirements.**

## Keywords

**Test Case Generation, Business Logic, Improved Depth-First Search Algorithm, Software Testing**

# 深度优先搜索算法生成测试用例的改进技术

**王婷婷[1]，胡　勇[2]，王晨光[1]，侯　磊[1]，刘　芸[1]，孟　伟[1]**

[1]北京京航计算通讯研究所，北京
[2]北京特种机电研究所，北京
Email: 765176342@qq.com

收稿日期：2019年6月27日；录用日期：2019年7月8日；发布日期：2019年7月15日

## 摘　要

**为保证系统软件的质量，弥补目前测试用例生成方法的不足，提高用例生成的准确性和完整性，本论文**

文章引用: 王婷婷, 胡勇, 王晨光, 侯磊, 刘芸, 孟伟. 深度优先搜索算法生成测试用例的改进技术[J]. 计算机科学与应用, 2019, 9(7): 1245-1254. DOI: 10.12677/csa.2019.97140

**以某软件为典型代表和应用对象对国内外测试用例生成技术进行研究，并针对业务控制流、数据流和树形结构图的构造方法进行探索，结合深度优先搜索算法和实际应用要求，对传统的深度优先搜索算法进行优化改进，在每个分支节点增加加权参数，避免节点存在多条路径时丢失其他节点到该节点的路径覆盖数据，以便可以遍历到所有数据分支，使其快速搜索有效测试用例，覆盖测试需求。**

## 关键词

**测试用例生成，业务逻辑，改进深度优先搜索算法，软件测试**



## 1. 引言

目前系统软件高端复杂，据统计设计测试用例的花费占到所有软件测试开销的40%左右，长期以来，大部分测试用例的生成仍通过手工完成，对参与测试的人员经验和专业水平都有着较高的要求，且在实际测试用例生成中也带有很高的盲目性。由于现代装备技术指标要求不断提升，软件的规模和复杂性正在以前所未有的速度增长，对其软件质量的要求也是随之高升，这就需要测试人员掌握一定的测试技术，设计有效的测试用例，尽可能的覆盖到所有项目指标，发现尽可能多的隐含缺陷。目前很多国内外的学者也纷纷踏入该领域，深入研究软件质量问题和测试用例生成方法。为了提高测试用例生成的准确性和完整性，减少测试时间成本和人力成本，论文中以某软件为典型代表和应用对象对国内外测试用例生成技术进行研究，并针对业务控制流、数据流和树形结构图的构造方法进行探索，结合深度优先搜索算法和实际应用要求，对传统的深度优先搜索算法进行优化改进并应用其中，使其快速搜索有效测试用例，覆盖测试需求，进一步保障软件质量。

## 2. 测试用例生成方法研究背景

随着计算机技术的不断发展，计算机软件占据着重要的地位，其保证软件高质量也成为重中之重，软件测试是保证软件质量的唯一有效的方法，而测试用例的生成又是软件测试的关键，测试人员需要设计、执行、分析大量的测试用例，测试用例设计的好坏也将直接影响软件测试的有效性。软件测试主要是查看其是否满足用户的需要，常常使用功能分解测试方法对其进行验证，完全不考虑程序内部结构，被测程序被当作打不开的黑盒，测试者输入数据驱动程序的执行，在只知道程序输入与输出之间关系的情况下，推断测试结果的正确性[1]。有相关软件大部分均是基于用户界面操作，也常常只考虑软件的界面信息，而忽略了软件的具体需求和逻辑实现流程，这就导致产生的测试用例随意性较大，使得覆盖率降低。

软件测试就是描述一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程。好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案，测试并不仅仅是为了找出错误，通过分析错误产生的原因和错误的发生趋势，可以帮助项目管理者发现当前软件开发过程中的缺陷，以便及时改进。

软件缺陷是导致软件失效的最主要因素。在各类软件中，一个安全关键软件的缺陷可能会造成数千万的经济损失，甚至可能造成无法挽回的政治和社会后果。因此，提高软件系统的可靠性与安全性，最大程度的降低软件缺陷，一直是软件研制过程中最重要的工作之一。由于各行各业对软件技术指标要求不断提升，软件的规模和复杂性正在以前所未有的速度增长，而软件行业的统计结果表明，软件中的缺

陷数量与规模和复杂性呈非线性增长的关系，如果不能有效的控制缺陷，软件质量无疑将成为设备质量的最严重的制约瓶颈，这一形势在当前新设备密集、多型号并举的情况下尤其严峻。这就需要测试人员掌握一定的测试技术设计有效的测试用例，尽可能的覆盖到所有项目需求，发现尽可能多的隐含缺陷，进一步保障软件质量。

## 3. 测试用例生成技术研究

### 3.1. 测试用例生成技术的缺陷

随着信息技术的高速发展，系统软件高端复杂，使得目前需要站在更高角度上对软件进行安全性和可用性测试，因而测试用例的生成也就成为软件测试的关键任务和难点。据统计设计测试用例的花费占到所有软件测试开销的 40%左右，长期以来，大部分测试用例的生成仍通过手工完成[2]，对参与测试的人员经验和专业水平都有着较高的要求。因此，实际情况中的测试用例生成带有很高的盲目性，结果测试用例不仅数量多，而且测试效果不佳，使得测试成本偏高[2]。设备系统软件大部分功能的实现是通过复杂的人机交互完成的，目前的测试用例生成方法也常常只考虑软件的界面信息，而不考虑软件的具体需求和逻辑实现流程，这就导致产生的测试用例随意性较大从而导致生成的测试用例不能很好地覆盖需求。

### 3.2. 国内外对测试用例生成技术的研究

在 20 世纪 80 年代以来，软件测试用例的智能化生成问题逐渐成为国内外学者的研究热点。Jeff Offutt 等人在软件测试领域著有多篇论文，在文献中他们提出一种从 UML 状态机生成测试用例的技术[2] [3]。Marlong 等人描述的基于规格说明的测试最初也以 UML 状态机图作为基础来自动化地生成测试驱动和测试脚本以及测试组件，并实现了一个原型系统，基于设计和规格说明的面向对象测试(Design And Specification-Based Object-Oriented Testing, DAS-BOOT) [2]。

数据流测试技术是一种基于代码的白盒测试技术，它能提供充分的代码覆盖，已经被广泛应用到面向对象软件测试中。Yogesh Singh 等人提出的通过契约理念加强数据流分析，该技术通过规约得到类流(Class Flow Graph)，然后应用传统的数据流技术选择测试用例[4]。契约理念使我们可以只测试那些可行的方法或者方法序列，即只有当方法的前置条件得到满足后才执行相应的方法序列，由于程序中的许多条件都转换为前置条件的一部分，这样就大大减少被测数据流信息[5]。Mary Jean Harrold 等人提出的对类进行的数据流测试技术，通过生成控制流图(Control Flow Graph)辅助数据流计算[4]，使用计算的数据流关系指导测试用例的选择，该技术的主要优势就是能对整个类进行数据流测试。

国内学者金虎将遗传算法与等价类思想，根据适应度值动态调整测试用例数量，确保了测试的准确性，避免了无效测试用例的生成[4]。黄丽芬将传统遗传算法加以改进，提出一种遗传-蚁群算法的软件测试数据生成算法[6]，充分发挥了遗传算法的全局搜索和蚁群算法的局部搜索优势，提高了测试数据的生成能力。相关文献提出利用符号执行方法去改善测试用例的路径覆盖率以及挖掘软件漏洞，利用符号执行生成测试用例，在不需要用户隐私信息的情况下，从开发者的角度重现用户在使用时发生的软件错误。还有学者利用软件系统模型来驱动产生测试用例的形式化方法，根据选择模型的不同，具体的实现方法不同，目前典型的软件系统模型有：有限状态机(FSM)、UML 和马尔可夫链等模型[7]。

论文中提出了一种基于业务逻辑的测试用例生成方法，以某软件为典型代表和应用对象，根据业务逻辑分析业务处理中数据的状态转变，得到业务控制流，在控制流的辅助下根据业务数据的流向和变化得到相应的数据流，再将数据流转换为对应的树形结构图。分析深度优先搜索算法实现的原理并结合实际需求对其进行算法改进，且应用于树形结构图中，生成覆盖所有流向及约束条件的用例集，通过该方法可以找到黑盒测试覆盖不到的错误，使生成的测试用例很好的覆盖测试需求。

## 4. 测试用例生成技术改进研究

基于业务逻辑生成测试用例，可根据业务逻辑总结数据流分支，其中数据流技术作为一种白盒测试技术，能提供充分的代码覆盖，对应的树形结构图可以清晰地描述系统整体结构和业务逻辑关系，通过深度优先搜索算法可遍历到每个分支，保证生成的测试数据可充分覆盖测试需求。论文中提出了一种测试用例生成的系统模型，对深度优先搜索算法具体的算法思想进行介绍，并根据具体的业务流程设计测试用例。再通过系统中某实例软件，验证系统模型的可行性。

### 4.1. 控制流构造方法技术分析

控制流是控制程序逻辑执行的先后顺序，控制流实际上是数据流融入控制层之后形成的逻辑处理和程序跳转的结果，它所控制的对象是数据，数据在逻辑处理过程中的形式和状态的变化，控制层的核心职责是处理业务逻辑，只有对业务逻辑的处理是我们在控制层所关心的核心内容，而除此之外的代码，则应该通过合理的设计，转化为一个标准而规范的事件处理流程。

控制流由对应的控制流图(Control Flow Graph, CFG)也叫控制流程图来体现，这是一个过程或程序的抽象表现，代表了一个程序执行过程中会遍历到的所有路径。它用图的形式表示一个过程内所有基本块执行的可能流向，也能反映一个过程的实时执行过程。其中控制流生成方法如下所示：

1) 划分事件处理流程步骤；

2) 定义事件处理节点对象；

3) 组织事件处理节点对象的执行次序。

下面以某设备软件为例，完成实际业务控制流图的设计，模块功能描述为：设备管理软件向用户提供数据采集转发功能、获取各类设备状态信息、获取时间和定位信息并同步设备等。其对应控制流图如图 1 所示：

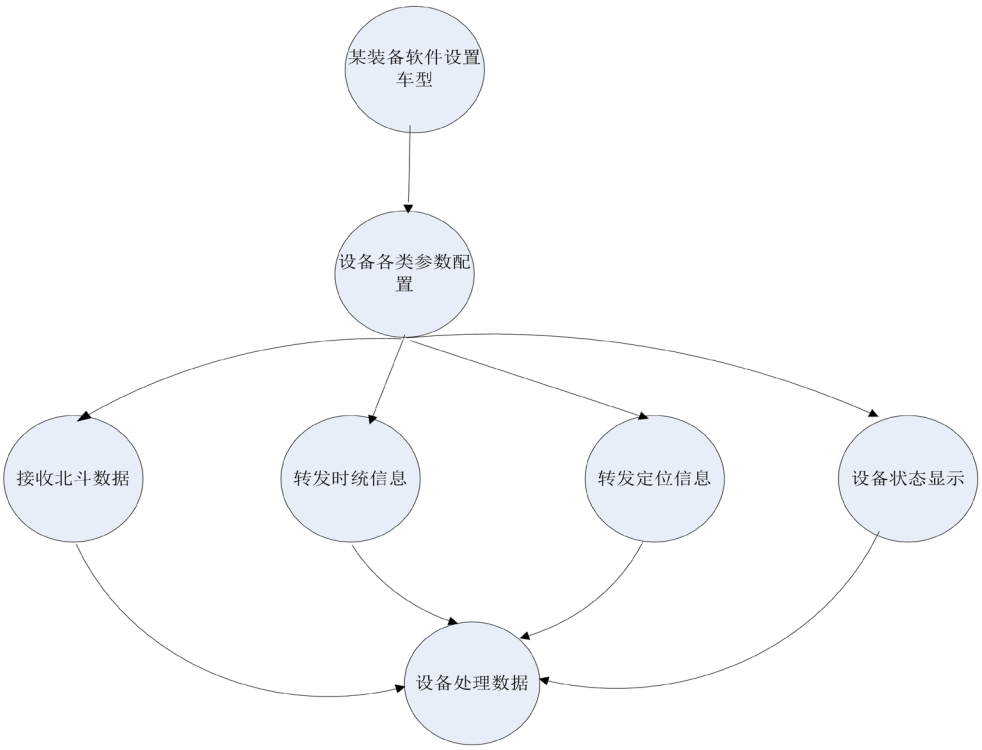


**Figure 1.** Control flow chart of a software function module
**图 1.** 某软件某功能模块控制流程图

### 4.2. 数据流的构造方法技术分析

数据流是一组有序、有起点和终点的字节的数据序列，是一串连续不断的数据的集合，包括输入流和输出流。数据流由对应的数据流图(Data Flow Diagram，简称 DFD)体现，它从数据传递和加工角度，以图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程，是结构化系统分析方法的主要表达工具及用于表示软件模型的一种图示方法[4]。它是一种便于用户理解和分析系统数据流程的图形工具，他摆脱了系统和具体内容，精确的在逻辑上描述系统的功能、输入、输出和数据存储等，是系统逻辑模型的重要组成部分。

数据流图从数据传递和加工的角度，把一个系统看成一个整体功能，明确信息的输入与输出,以图形的方式刻画数据流从输入到输出的移动变换过程。其中数据流图包含四个元素：外部实体、处理过程、数据流、数据存储，其模型如图 2 所示。

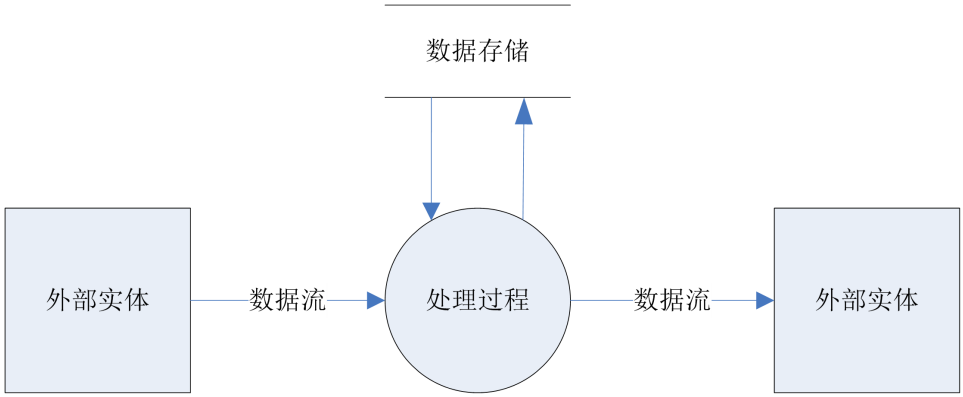


**Figure 2.** Data flow diagram model
**图 2.** 数据流图模型

结合系统业务逻辑，由上到下，先分析顶层数据流图，再将整体划分为多个独立且互通的模块，再由系统控制流图作为辅助，分别描述顶层互通模块和各个子模块相关数据流图，其生成方法如下所示：

1) 确定系统软件的应用实体；

2) 分析实体间业务交互数据；

3) 解析实体输入输出，确定各个实体的出度和入度；

4) 根据数据处理过程确定数据流向，每个处理至少有一个输入数据流和一个输出数据流，反映此处理数据的来源与加工的结果；

5) 根据存储数据和分支流向，确定相关约束条件。

面以某软件某功能模块为例，根据上节业务控制流图完成数据流图的设计，如图 3 所示。

### 4.3. 树形结构图的构造方法研究

树形结构图是 n 个节点的集合，且可清晰明了的表明节点间的层次关系和数据分支，根据数据流向可递归遍历所有节点，树形结构图生成方法如下所示：

1) 确定节点个数并根据先后顺序确定层次关系；

2) 确定每个节点的出度和入度数；

3) 根据数据流确定节点分支；

下面以某软件为例，根据上节业务数据流图完成树形结构图的设计，如图 4 所示：

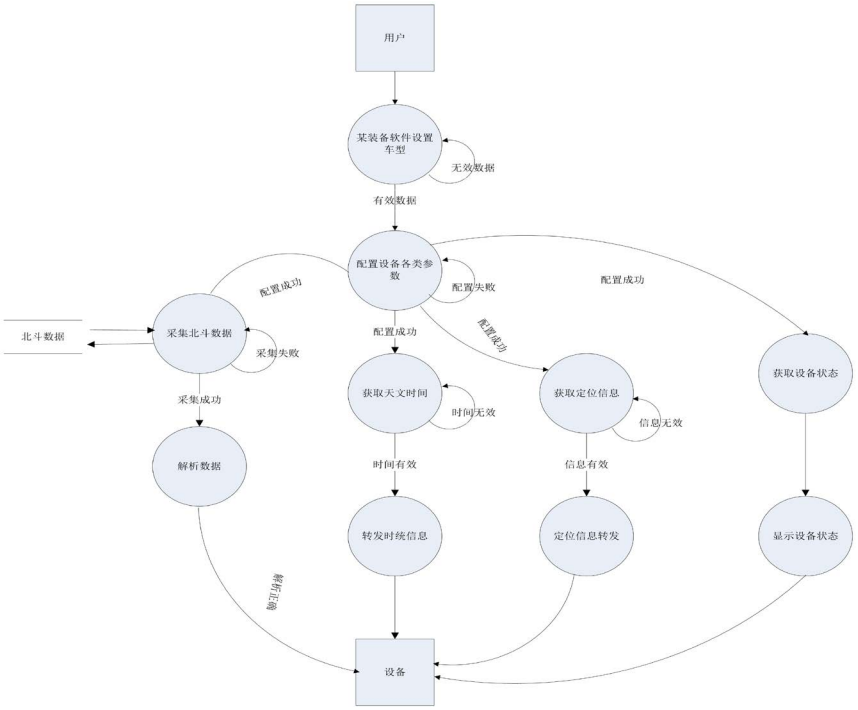


**Figure 3.** Data flow diagram of a software function module
**图 3.** 某软件某功能模块数据流图

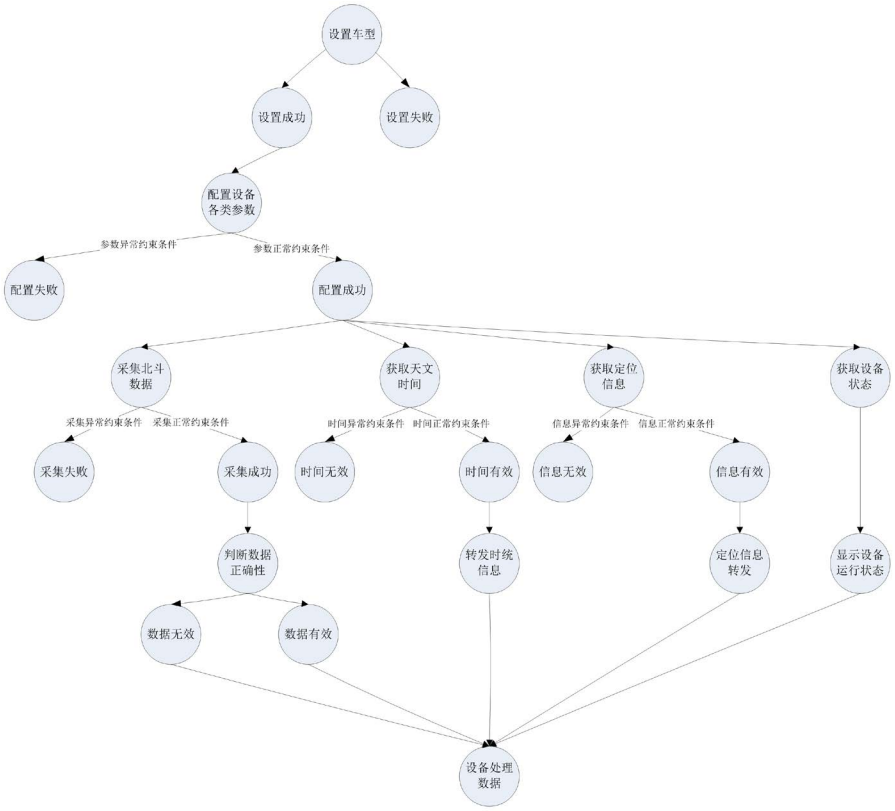


**Figure 4.** Tree structure chart of a software function module
**图 4.** 某软件某功能模块树形结构图

### 4.4. 改进的深度优先搜索算法研究

深度优先搜索(DFS)属于图算法的一种，其过程简要来说是对每一个可能的分支路径深入到不能再深入为止，而且每个节点只能访问一次，但传统的深度优先搜索算法在覆盖数据分支时存在漏洞，若某个节点入度数大于 1 时就会丢失相应的路径用例，不能完全覆盖测试需求。现结合实际应用对深度优先搜索算法进行改进，对每个节点增加加权参数 i，这样就避免节点存在多条路径时而丢失其他节点到该节点的路径覆盖数据，以便可以遍历到所有数据分支，覆盖所有业务需求。则改进的深度优先搜索算法(PDFS)递归调用包含以下操作：

1) 访问搜索到的未被访问的邻接点且记录其入度数；

2) 将此顶点标记为已访问节点且入度数减 1；

3) 搜索该顶点的未被访问的邻接点，若该邻接点存在，则从此邻接点开始进行同样的访问和搜索。

PDFS 详细的遍历策略如下描述：

从一个顶点 v 出发，首先将 v 标记为已遍历的顶点且记录其入度数为 0，然后选择一个邻接于 v 的尚未遍历的顶点 u 且记录其入度数，如果 u 不存在，本次搜索终止；如果 u 存在，那么从 u 又开始一次 PDFS 遍历且入度数减 1。如此循环直到不存在未被遍历的顶点且所有节点入度数均为 0。以图 5 为例描述 PDFC 搜索过程。

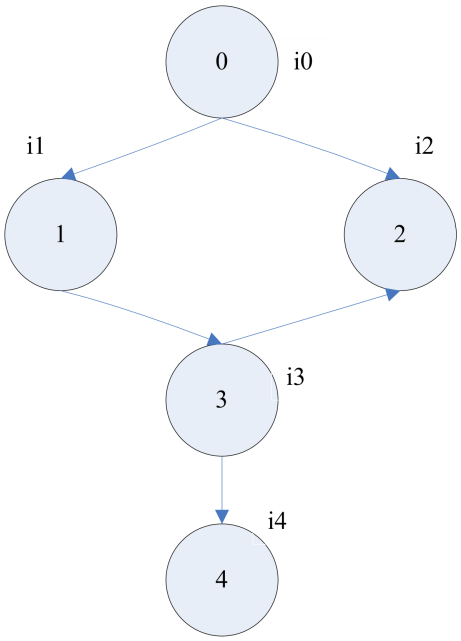


**Figure 5.** PDFC example diagram
**图 5.** PDFC 示例图

1) 从顶点 0 开始，将 0 标记为已遍历且记录入度数 i0 为 0，然后选择未被遍历的邻接 0 的顶点 1，记录 i1 为 1；

2) 标记顶点 1，执行 i1 减 1 操作，i1 为 0，然后选择 3 且记录 i3 为 1，标记 3 并执行 i3 减 1 操作，i3 为 0，然后选择顶点 3 邻接的顶点 2，记录 i2 为 2；

3) 顶点 2 标记后 i2 为 1，没有与它邻接的未标记的点，所以返回 3 选择另一个邻接 3 并且未被标记的顶点 4，记录 i4 为 1；

4) 标记 4 且 i4 为 0，顶点 4 没有更多的符合条件的点，因此搜索终止，返回到 3，3 没有更多的点，搜索终止返回到 1，最后返回到 0，顶点 0 仍有入度数不为 0 的点，选择邻接点 2，执行 i2 减 1 操作，i2 为 0，没有与它邻接的未标记的点，再次返回到 0，搜索终止。

### 4.5. 根据改进的深度优先搜索算法分析测试用例

由上图 PDFS 模型图还原为传统的深度优先搜索算法，即去除各个分支的加权值，根据未改进的算法进行遍历，得到相应的测试用例集，记为 U{0→1, 1→3, 3→2, 3→4}。再根据上图 PDFS 模型图和遍历策略得到相应的测试用例集，记为 U'{0→1, 1→3, 3→2, 3→4, 0→2}，通过比对可发现，由传统深度优先搜索算法得到的用例集会比优化算法得到的用例集要少，即{0→2}，这就会造成测试用例的丢失，功能项覆盖不全面，通过优化改进的深度优先搜索算法可避免该现象。

下面以某装备软件的某功能模块为例，得到相应的测试用例集。将功能模块树形图转化为简易的加权树形结构图(简化后的部分功能项)，如图 6 所示。

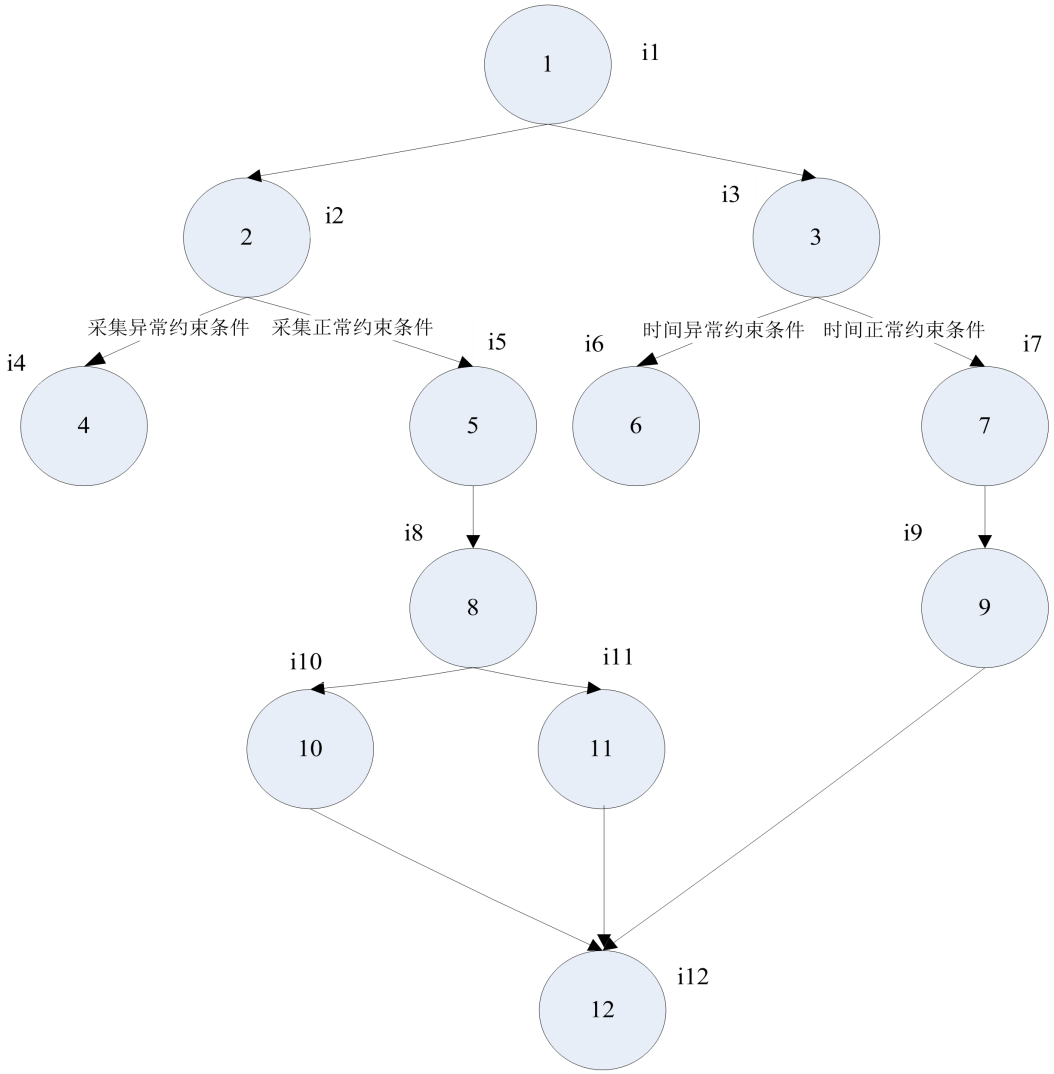


**Figure 6.** A simple weighted tree structure diagram of a software function module
**图 6.** 某软件某功能模块简易加权树形结构图

根据传统的 DFS 得到所有的用例集如下所示：

{1→2：采集北斗数据；

2→4：采集数据异常(含多种采集异常情况，可重复执行)；

2→5：采集数据正常；

5→8：验证数据正确性；

8→10：采集数据无效(构造错误数据，可重复执行)；

10→12：设备处理采集的无效数据；

8→11：采集数据有效；

1→3：获取天文时间；

3→6：时间信息异常(含多种信息异常情况，可重复执行)；

3→7：时间信息正常；

7→9：转发时统信息。

}

根据优化改进的 PDFS 算法得到所有的用例集如下所示：

由树形结构图可知 i1 值为 0，i2~i11 值均为 1，i12 值为 3；

用例集 U 为

{1→2：采集北斗数据，i2=0；

2→4：采集数据异常(含多种采集异常情况，可重复执行)，i4=0；

2→5：采集数据正常，i5=0；

5→8：验证数据正确性，i8=0；

8→10：采集数据无效(构造错误数据，可重复执行)，i10=0；

10→12：设备处理采集的无效数据，i12=3；

8→11：采集数据有效，i11=0；

11→12：设备处理采集的有效数据，i12=2；

1→3：获取天文时间，i3=0；

3→6：时间信息异常(含多种信息异常情况，可重复执行)，i6=0；

3→7：时间信息正常，i7=0；

7→9：转发时统信息，i9=0；

9→12：设备执行时间同步，i12=1。

}

由此可看出优化的 PDFS 比传统的 DFS 设计测试用例更全面，覆盖测试项更充分。

## 5. 总结

论文中对生成测试用例的深度优先搜索算法的优化和改进，可为软件测试人员提供可靠指导和有效建议，缩短软件测试周期，可以提高测试人员对系统测试的充分性，降低测试验证周期，节约时间成本和人力成本，同时也可为开发人员提供参考，使其在软件研发过程中减少设计缺陷，确保软件高质量。论文中目前以某软件为例进行了验证，后续还需扩大范围进一步研究，并完善搜索算法。

## 参考文献

[1] 唐滔, 钟华. 基于会话的 Web 应用测试方法[J]. 中国科技博览, 2011(3): 55-56.

[2] 韩璐. 基于 Web 用户行为的测试用例生成技术研究与实现[D]: [硕士学位论文]. 郑州: 郑州大学, 2016: 1-4.

[3] Offutt, J., Liu, S., Abdurazik, A. and Ammann, P. (2003) Generating Test Data from State Based Specifications. *The Journal of Software Testing*, *Verification and Reliability*, **13**, 25-53. https://doi.org/10.1002/stvr.264

[4] 邓名杰. 基于数据流分析的测试用例自动生成技术[D]: [硕士学位论文]. 大连: 大连海事大学, 2009: 3-6.

[5] 曹文静, 徐胜红. 基于数据流图的测试用例生成技术研究[C]//中国通信学会青年工作委员会. 2009 通信理论与技术新发展－第十四届全国青年通讯学术会议. 北京: 电子工业出版社, 2009: 145-147.

[6] 黄丽芬. 软件测试数据自动生成算法的仿真研究[J]. 计算机仿真, 2012, 29(10): 245-382.

[7] 高莉, 李龙澎. 基于 UML 状态图的测试技术研究[J]. 计算机技术与发展, 2009(5): 68-71.