浙江大学

本科实验报告

课程名称：软件质量保证与测试

姓　　名：王俊怡

学　　院：计算机科学与技术学院

专　　业：软件工程

学　　号：3210106016

指导教师：赵晓琼

2023 年 12 月 17 日

AETG Algorithm Implementation Experiment Report

1 Introduction

1.1 Combinational Testing

Combinatorial testing, is a software testing technique used to systematically test different combinations of input parameters or configuration settings in order to identify defects or vulnerabilities in a software system.

In software development, there are often many possible combinations of input values or configuration settings that can affect the behavior of a system. Testing all possible combinations individually can be time-consuming and impractical, especially for complex software with numerous parameters. Combinational testing offers a more efficient way to cover a wide range of scenarios instead of testing each one separately.

1.2 AETG

The AETG (Automated Efficient Test Generation) algorithm is a method to generate test cases in combination testing that cover all possible combinations of input parameters. This algorithm is a form of combinatorial testing designed to efficiently test complex systems with many interacting components or settings.

The AETG algorithm reduces the number of test cases needed to achieve significant coverage, thereby saving time and resources in the testing process. [1]

2 Algorithm Description

Assume that we have a system with k test parameters and that the i th parameter has l_i different values. We select each test case by first generating M different candidate test cases and then choosing one that covers the most new pairs. Each candidate test case is selected by the following greedy algorithm:

1. Choose the first parameter f and a value l for f such that it appears in the greatest number of uncovered pairs.
2. Let $f_1 = f$. Then randomly generate a order for the remaining parameters. Then, we have an order for all k parameters $f_1, ..., f_k$.
3. For each parameter f_i in the order, choose a value l_i for f_i. The value l_i is chosen by the following method.
   - Assume that we have already selected $i - 1$ values for parameters $f_1, ..., f_{i-1}$ and we will now select the value for f_i.
   - if $i \leq t$, for each value of f_i, we combine it with the $l_1, ..., l_{i-1}$. Now we have $l_1, ..., l_i$. Then calculate the number of new pairs that will be covered by the new case. Then we choose the value that covers the most new pairs.
   - if $i > t$, for each value of f_i, we select $t - 1$ parameters from $f_1, ..., f_{i-1}$ and traverse all possible combinations. Then we will follow the same steps as before, choose the value that covers the most new pairs.
4. After we have selected all k values, we have a candidate test case. Then we choose the candidate test case that covers the most new pairs.
5. Repeat step 1-4 until the uncovered-pairs-set is empty.

3 Algorithm Implementation

3.1 Data Structure

3.1.1 Class `Data`

Class Data has 4 variables: `name`, `options`, `parameters`, `data`. `name` is the name of the data. `options` is the title of each parameters. `parameters` is an array stores the number of the values of each parameters, which will be used in AETG algorithm. `data` is a 2D array that stores the data.

For example, if we have parameters like this: `[5, 7, 2, 3, 6]`, it represents that we have 5 parameters and the first parameter has 5 values, the second parameter has 7 values, and so on. Then we can use this array in AETG to generate the test case.

3.1.2 Test Case

The test case in AETG algorithm is an array. For example, if the test case is `[-1, 0, 2, -1, 1]`, it means that the first parameter is not selected, the second parameter is the first value, the third parameter is the third value, and so on.

3.1.3 Uncoveres Pairs

At the beginning of the algorithm, all uncovered pairs (ucps) will be generated. The uncovered pairs is an array that stores the pairs that are not covered. For each ucp, we find all combinations of **t** parameters and all combinations of the values for each parameters. We represent it as `[[0, 0, -1, -1, -1], [0, 1, 1, -1, -1], ...]` where t=2.

3.2 Algorithm Implementation

3.2.1 Pseudo Code

```
uncovered_pairs = generate_uncovered_pairs(parameters, t)
test_case = []
while uncovered_pairs is not empty:
    # generate_candidate_test_cases
    candidate_test_cases = []
    for m: 1 to M
        test_case = []
        test_case[0] = choose_best_first_value()
        for i: 1 to t
            for each value in parameters[i]:
                tmp_test_case[i] = value
                pairs_count = count_pairs(tmp_test_case, uncovered_pairs)
            best_value = choose_best_value()
            test_case[i] = best_value
            candidate_test_cases.append(test_case)
        for i: t+1 to k
            for each combination of t-1 parameters:
                for each value in parameters[i]:
                    test_case[i] = value
                    candidate_test_cases.append(test_case)
    test_case = choose_best_candidate(candidate_test_cases, uncovered_pairs)
    uncovered_pairs = update_uncovered_pairs(test_case, uncovered_pairs)
```

3.2.2 Details

1. Random selection

When selecting the first parameter and the best value for each parameters, we may encounter situations with multiple optimal solutions. In this case, I **ramdomly** choose one of them. If we always choose the first optimal solution, the algorithm may get stuck in a local optimal solution.

2. Dynamic M

   In the AETG algorithm, we use the greedy algorithm to generate test cases multiple times and finally select the one with the most coverage from these test cases. At the beginning, the algorithm can easliy find a test case that covers a lot of pairs. However, as the number of uncovered pairs decreases, the algorithm will be more difficult to find a test case that covers a lot of pairs. To solve this problem, I designed a dynamic method to adjust the number of M.

```
# User input the min and max value of M from the command line.
rate = 0
max_cover_count = int(math.factorial(len(parameters)) / math.factorial(t) / math.factorial(len(parameters) - t))
while len(uncovered_pairs) > 0:
    # aetg code...
    m = int(m_min + rate * (m_max - m_min))
    for m_cnt in range(m):
        # generate candidate test cases...
    rate = (max_cover_count - best_test_case[1])/(max_cover_count - 10)
    if rate > 1:
        rate = 1
```

4 Results

The complete results can be seen in file 'result/output_website_t-wise.csv'.

4.1 JingDong

4.1.1 CIT Model

```
# 品牌
brand = ['ThinkPad', 'DELL', '华为', 'Lenovo', 'Apple', 'hp', 'ASUS', 'MI', 'HONOR']
# 能效等级
energy_efficiency = ['一级', '二级', '三级']
# SSD
ssd = ['3TB', '128GB', '256GB+1TB', '512GB+2TB', '3TB*2']
# 厚度
thickness = ['15.0mm 及以下', '15.1-18.0mm', '18.1-20.0mm', '20.0mm 以上']
# 机身材质
body_material = ['金属', '金属+复合材质', '复合材质', '含碳纤维']
# 屏幕尺寸
screen_size = ['13.0 英寸以下', '13.0-13.9 英寸', '14.0-14.9 英寸', '15.0-15.9 英寸', '16.0-16.9 英寸']
# 刷新率
refresh_rate = ['144Hz', '60Hz', '120Hz', '90Hz', '165Hz']
```

4.1.2 Partial Test Cases

1. 2-wise

	column 1	column 2	column 3	column 4	column 5	column 6	column 7	column 8
1	品牌	能效等级	SSD	厚度	机身材质	屏幕尺寸	刷新率	覆盖对数
2	ThinkPad	三级	3TB	20.0mm以上	复合材质	15.0-15.9英寸	120Hz	21
3	华为	二级	256GB+1TB	18.1-20.0mm	含碳纤维	15.0-15.9英寸	60Hz	21
4	ASUS	一级	3TB	15.1-18.0mm	金属	13.0英寸以下	165Hz	21
5	HONOR	二级	3TB	15.0mm及以下	金属+复合材质	14.0-14.9英寸	90Hz	21
6	MI	三级	3TB*2	18.1-20.0mm	金属	14.0-14.9英寸	144Hz	21
7	hp	一级	128GB	18.1-20.0mm	金属+复合材质	13.0-13.9英寸	165Hz	20
8	hp	三级	256GB+1TB	15.1-18.0mm	含碳纤维	16.0-16.9英寸	90Hz	20
9	ThinkPad	二级	512GB+2TB	15.1-18.0mm	复合材质	13.0-13.9英寸	60Hz	19
10	Lenovo	一级	3TB*2	15.0mm及以下	复合材质	16.0-16.9英寸	60Hz	20
11	DELL	三级	512GB+2TB	15.0mm及以下	含碳纤维	13.0英寸以下	144Hz	19
12	Apple	二级	128GB	20.0mm以上	金属	13.0英寸以下	90Hz	19
13	Apple	一级	3TB*2	20.0mm以上	含碳纤维	14.0-14.9英寸	120Hz	17
14	Lenovo	三级	256GB+1TB	15.1-18.0mm	金属+复合材质	15.0-15.9英寸	144Hz	15
15	MI	一级	256GB+1TB	15.0mm及以下	复合材质	15.0-15.9英寸	90Hz	14
16	HONOR	三级	256GB+1TB	20.0mm以上	金属	13.0-13.9英寸	165Hz	15
17	DELL	二级	3TB	18.1-20.0mm	复合材质	16.0-16.9英寸	165Hz	14
18	华为	三级	512GB+2TB	20.0mm以上	金属+复合材质	13.0英寸以下	120Hz	13
19	ASUS	三级	128GB	20.0mm以上	复合材质	16.0-16.9英寸	60Hz	13
20	ASUS	二级	256GB+1TB	15.0mm及以下	金属	13.0-13.9英寸	120Hz	12
21	ThinkPad	二级	3TB*2	18.1-20.0mm	金属+复合材质	16.0-16.9英寸	144Hz	10
22	Apple	一级	512GB+2TB	18.1-20.0mm	复合材质	13.0英寸以下	90Hz	9
23	DELL	一级	3TB*2	15.1-18.0mm	金属	15.0-15.9英寸	120Hz	10
24	HONOR	三级	128GB	15.1-18.0mm	复合材质	14.0-14.9英寸	120Hz	9
25	Lenovo	一级	3TB	15.0mm及以下	含碳纤维	13.0-13.9英寸	165Hz	9
26	华为	一级	128GB	20.0mm以上	复合材质	13.0英寸以下	144Hz	8
27	MI	三级	512GB+2TB	20.0mm以上	金属	16.0-16.9英寸	165Hz	8
28	hp	二级	256GB+1TB	20.0mm以上	金属	13.0英寸以下	60Hz	8
29	Apple	三级	3TB	15.0mm及以下	金属+复合材质	14.0-14.9英寸	60Hz	8
30	ThinkPad	一级	256GB+1TB	15.1-18.0mm	金属	14.0-14.9英寸	165Hz	7
31	Lenovo	二级	128GB	15.0mm及以下	金属	15.0-15.9英寸	165Hz	6
32	华为	三级	3TB*2	15.0mm及以下	金属	13.0-13.9英寸	90Hz	8
33	HONOR	一级	512GB+2TB	18.1-20.0mm	含碳纤维	15.0-15.9英寸	144Hz	7
34	MI	二级	128GB	15.1-18.0mm	金属+复合材质	13.0英寸以下	120Hz	6
35	hp	三级	3TB*2	15.0mm及以下	复合材质	14.0-14.9英寸	165Hz	5

Figure 1: 2-wise-jingdong

2. 3-wise

	column 1	column 2	column 3	column 4	column 5	column 6	column 7	column 8
1	品牌	能效等级	SSD	厚度	机身材质	屏幕尺寸	刷新率	覆盖对数
2	ThinkPad	一级	3TB	20.0mm以上	复合材质	15.0-15.9英寸	90Hz	35
3	ThinkPad	三级	128GB	20.0mm以上	金属	14.0-14.9英寸	165Hz	35
4	华为	二级	256GB+1TB	15.0mm及以下	金属+复合材质	13.0英寸以下	165Hz	35
5	HONOR	二级	512GB+2TB	20.0mm以上	金属	13.0英寸以下	60Hz	35
6	hp	一级	512GB+2TB	18.1-20.0mm	金属+复合材质	14.0-14.9英寸	60Hz	35
7	Lenovo	三级	3TB*2	15.1-18.0mm	含碳纤维	13.0英寸以下	60Hz	35
8	MI	二级	3TB*2	20.0mm以上	含碳纤维	15.0-15.9英寸	144Hz	35
9	Apple	一级	3TB*2	18.1-20.0mm	含碳纤维	16.0-16.9英寸	90Hz	35
10	DELL	三级	3TB	15.0mm及以下	金属	15.0-15.9英寸	144Hz	35
11	Lenovo	一级	512GB+2TB	15.1-18.0mm	金属	16.0-16.9英寸	120Hz	35
12	hp	三级	256GB+1TB	15.0mm及以下	复合材质	14.0-14.9英寸	90Hz	35
13	ASUS	二级	128GB	18.1-20.0mm	含碳纤维	13.0-13.9英寸	60Hz	35
14	DELL	二级	256GB+1TB	15.1-18.0mm	含碳纤维	14.0-14.9英寸	120Hz	35
15	Apple	三级	512GB+2TB	18.1-20.0mm	复合材质	13.0英寸以下	144Hz	35
16	HONOR	三级	3TB*2	18.1-20.0mm	金属+复合材质	15.0-15.9英寸	120Hz	35
17	MI	一级	3TB	15.0mm及以下	金属+复合材质	16.0-16.9英寸	60Hz	34
18	ASUS	一级	3TB*2	15.0mm及以下	金属	13.0-13.9英寸	165Hz	35
19	ASUS	二级	512GB+2TB	15.1-18.0mm	金属+复合材质	16.0-16.9英寸	90Hz	34
20	Lenovo	二级	256GB+1TB	20.0mm以上	复合材质	13.0-13.9英寸	165Hz	34
21	MI	一级	128GB	15.1-18.0mm	金属+复合材质	13.0-13.9英寸	90Hz	33
22	hp	二级	128GB	18.1-20.0mm	复合材质	15.0-15.9英寸	120Hz	33
23	ThinkPad	三级	512GB+2TB	15.0mm及以下	金属+复合材质	13.0-13.9英寸	60Hz	33
24	ASUS	三级	3TB	20.0mm以上	金属+复合材质	13.0英寸以下	120Hz	34
25	Apple	一级	128GB	15.0mm及以下	含碳纤维	14.0-14.9英寸	144Hz	34
26	ThinkPad	三级	3TB	15.1-18.0mm	复合材质	16.0-16.9英寸	165Hz	33
27	hp	一级	256GB+1TB	20.0mm以上	金属	13.0英寸以下	144Hz	34
28	华为	二级	128GB	18.1-20.0mm	金属	14.0-14.9英寸	90Hz	33
29	HONOR	三级	256GB+1TB	15.1-18.0mm	金属+复合材质	16.0-16.9英寸	144Hz	32
30	HONOR	一级	512GB+2TB	15.0mm及以下	含碳纤维	13.0英寸以下	120Hz	32
31	hp	一级	3TB*2	15.1-18.0mm	复合材质	14.0-14.9英寸	144Hz	31
32	Apple	一级	256GB+1TB	15.1-18.0mm	含碳纤维	15.0-15.9英寸	165Hz	33
33	Apple	二级	3TB	15.0mm及以下	含碳纤维	13.0-13.9英寸	90Hz	32
34	ASUS	三级	256GB+1TB	18.1-20.0mm	金属	16.0-16.9英寸	60Hz	33
35	华为	三级	128GB	15.1-18.0mm	复合材质	15.0-15.9英寸	60Hz	32

Figure 2: 3-wise-jingdong

4.2 XieCheng

4.2.1 CIT Model

```
# 票型
ticket_type = ['单程', '往返', '多程']
# 出发地
start = ['国内', '国际·中国港澳台热门', '亚洲', '欧洲', '美洲', '非洲', '大洋洲']
# 目的地
end = ['国内', '国际·中国港澳台热门', '亚洲', '欧洲', '美洲', '非洲', '大洋洲']
# 仅看直飞
direct = ['是', '否']
# 舱型
cabin_type = ['经济/超经舱', '公务/头等舱', '公务舱', '头等舱']
# 乘客类型
customer = ['仅成人', '成人与儿童', '成人与婴儿', '成人与儿童与婴儿']
```

5 Conclusion and Discussion

5.1.1 Partial Test Cases

1. 2-wise

	column 1	column 2	column 3	column 4	column 5	column 6	column 7
1	票型	出发地	目的地	仅看直飞	舱型	乘客类型	覆盖对数
2	往返	国际·中国港澳台热门	大洋洲	否	公务/头等舱	仅成人	15
3	单程	亚洲	美洲	是	经济/超经舱	成人与儿童	15
4	多程	欧洲	国内	否	头等舱	成人与儿童	15
5	往返	大洋洲	国际·中国港澳台热门	是	头等舱	成人与婴儿	15
6	多程	美洲	欧洲	是	公务舱	成人与儿童与婴儿	15
7	多程	国内	亚洲	否	经济/超经舱	成人与婴儿	14
8	单程	非洲	国内	是	公务舱	仅成人	13
9	单程	非洲	非洲	否	公务/头等舱	成人与儿童与婴儿	13
10	往返	亚洲	美洲	否	公务舱	成人与婴儿	11
11	单程	欧洲	大洋洲	是	公务舱	成人与婴儿	10
12	单程	美洲	国际·中国港澳台热门	否	头等舱	仅成人	10
13	单程	大洋洲	欧洲	否	经济/超经舱	仅成人	10
14	往返	国内	非洲	是	公务/头等舱	成人与儿童	11
15	多程	国际·中国港澳台热门	亚洲	是	公务舱	成人与儿童	9
16	往返	欧洲	国际·中国港澳台热门	否	经济/超经舱	成人与儿童与婴儿	9
17	多程	亚洲	国际·中国港澳台热门	否	公务/头等舱	仅成人	8
18	多程	国际·中国港澳台热门	美洲	是	头等舱	成人与儿童与婴儿	7
19	往返	非洲	亚洲	否	头等舱	成人与婴儿	6
20	多程	大洋洲	非洲	否	公务舱	成人与儿童	6
21	多程	非洲	大洋洲	是	经济/超经舱	成人与儿童	7
22	往返	美洲	国内	是	经济/超经舱	成人与婴儿	7
23	往返	国际·中国港澳台热门	欧洲	是	公务/头等舱	成人与婴儿	6
24	单程	国内	大洋洲	是	头等舱	成人与儿童与婴儿	6
25	单程	大洋洲	亚洲	否	公务/头等舱	成人与儿童与婴儿	6
26	单程	国际·中国港澳台热门	非洲	否	经济/超经舱	仅成人	5
27	往返	欧洲	美洲	是	公务/头等舱	仅成人	5
28	单程	亚洲	欧洲	否	头等舱	成人与儿童与婴儿	4
29	单程	国内	国际·中国港澳台热门	否	公务舱	成人与儿童与婴儿	3
30	多程	美洲	国内	否	公务/头等舱	成人与儿童与婴儿	3
31	往返	美洲	非洲	是	头等舱	成人与婴儿	3
32	单程	非洲	欧洲	否	公务/头等舱	成人与儿童	2
33	往返	亚洲	亚洲	是	公务/头等舱	仅成人	2
34	往返	美洲	美洲	是	公务/头等舱	成人与儿童	2
35	多程	国内	国内	否	公务舱	仅成人	2

Figure 3: 2-wise-xiecheng

2. 3-wise

	column 1	column 2	column 3	column 4	column 5	column 6	column 7
1	票型	出发地	目的地	仅看直飞	舱型	乘客类型	覆盖对数
2	多程	国内	美洲	否	公务舱	成人与婴儿	20
3	往返	欧洲	欧洲	是	公务舱	成人与儿童	20
4	单程	欧洲	非洲	是	头等舱	成人与婴儿	20
5	单程	欧洲	亚洲	否	经济/超经舱	成人与儿童	20
6	往返	亚洲	国内	否	头等舱	成人与婴儿	20
7	往返	美洲	美洲	是	公务/头等舱	仅成人	20
8	多程	国际·中国港澳台热门	国际·中国港澳台热门	否	公务/头等舱	成人与儿童	20
9	多程	国内	国际·中国港澳台热门	是	经济/超经舱	仅成人	20
10	单程	亚洲	大洋洲	是	公务/头等舱	成人与儿童	20
11	往返	大洋洲	大洋洲	否	经济/超经舱	成人与儿童与婴儿	20
12	多程	非洲	国内	是	头等舱	成人与儿童与婴儿	20
13	多程	大洋洲	欧洲	否	头等舱	仅成人	20
14	单程	国际·中国港澳台热门	非洲	否	公务舱	成人与儿童与婴儿	20
15	单程	非洲	亚洲	是	公务舱	仅成人	20
16	往返	国际·中国港澳台热门	国际·中国港澳台热门	是	头等舱	成人与儿童与婴儿	19
17	单程	国内	大洋洲	否	头等舱	仅成人	19
18	多程	大洋洲	大洋洲	是	公务舱	成人与婴儿	19
19	往返	非洲	非洲	否	公务/头等舱	成人与婴儿	19
20	往返	国际·中国港澳台热门	欧洲	否	经济/超经舱	仅成人	18
21	单程	美洲	欧洲	否	公务/头等舱	成人与儿童与婴儿	19
22	单程	国际·中国港澳台热门	国内	是	经济/超经舱	成人与婴儿	19
23	多程	大洋洲	亚洲	是	经济/超经舱	成人与儿童	17
24	多程	欧洲	国内	否	公务舱	仅成人	18
25	多程	欧洲	亚洲	是	公务/头等舱	成人与儿童与婴儿	18
26	往返	国内	国际·中国港澳台热门	否	公务舱	成人与儿童	17
27	多程	亚洲	美洲	否	经济/超经舱	成人与儿童与婴儿	18
28	单程	大洋洲	国际·中国港澳台热门	否	公务/头等舱	成人与婴儿	17
29	往返	美洲	非洲	是	经济/超经舱	成人与儿童	17
30	单程	美洲	美洲	否	头等舱	成人与儿童	17
31	往返	大洋洲	国内	是	公务/头等舱	仅成人	16
32	多程	美洲	欧洲	是	公务/头等舱	成人与婴儿	16
33	单程	国内	美洲	是	公务/头等舱	成人与儿童与婴儿	16
34	单程	美洲	国际·中国港澳台热门	是	公务舱	成人与婴儿	15
35	多程	亚洲	非洲	是	头等舱	仅成人	16

Figure 4: 3-wise-xiecheng

References

[1] D. Cohen, S. Dalal, M. Fredman, and G. Patton, “The AETG system: an approach to testing based on combinatorial design”, *IEEE Transactions on Software Engineering*, vol. 23, no. 7, pp. 437–444, 1997, doi: 10.1109/32.605761.