AETG算法设计与测试

刘轩铭 软件工程1801 3180106071

任务描述

编写算法，实现 AETG 组合算法，要求：

- 禁止使用任何工具、方法库以及开源 Covering Array 代码，独立实现对组合算法的代码实现，确保掌握组合测试的 AETG 算法
- 要求提交相关的数据结果，格式为 excel 或 txt
- 对两个平台相关测试进行Cover Array的生成，用 CA（N；K，T，V）描述两个题目的组合问题
- 程序对于题目 1 打印 2-way（pair-wise）的 Covering Array
- 程序对于题目 2 打印 3-way（triple-wise）的 Covering Array
- 打印的 Covering Array 每行行尾需要显示覆盖的 T-way 的个数
- 最后的结果文件可以以附件的形式发送至 zyhxydb@163.com 邮箱, 邮件中需标注（名字，学号，email），主题标注为组合测试作业

算法描述

Assume that we have a system with k test parameters and that the ith parameter has l_i different values. Assume that we have already selected r test cases. We select the $r + 1$ 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 a parameter f and a value l for f such that that parameter value appears in the greatest number of uncovered pairs.
2) Let $f_1 = f$. Then choose a random order for the remaining parameters. Then, we have an order for all k parameters $f_1, \ldots f_k$.
3) Assume that values have been selected for parameters $f_1, \ldots, f_j$. For $1 \le i \le j$, let the selected value for f_i be called v_i. Then, choose a value v_{j+1} for f_{j+1} as follows. For each possible value v for f_j, find the number of new pairs in the set of pairs $\{f_{j+1} = v$ and $f_i = v_i$ for $1 \le i \le j\}$. Then, let v_{j+1} be one of the values that appeared in the greatest number of new pairs.

   Note that, in this step, each parameter value is considered only once for inclusion in a candidate test case. Also, that when choosing a value for parameter f_{j+1}, the possible values are compared with only the j values already chosen for parameters $f_1, \ldots, f_j$.

对于论文中提到的AETG算法，做如下适合编程实现的复述和翻译：

- 该问题是为了解决一个系统的 $CoverArray$（以下简称 CA） 问题。为了对一个系统进行测试，我们的测试样例需要对系统的参数组合进行覆盖。$CA(N;K,T,V)$ 代表该问题的解，也就是覆盖集，它包含 N 个测试组，每个测试组有该系统所需要的 K 个参数；而该覆盖集的强度用 T 来描述，它代表覆盖集包含了参数的所有 T 元组合； V 则是系统参数的取值范围。
- 假设该系统有 K 个参数，每个参数 k_i 有 l_i 个可能的取值。假设一开始有未覆盖对（以下简称 UCP）集$ucps$和覆盖对（以下简称 CP）集 cps。首先计算对于当前系统的那么当我们生成了 r 个测试组后，对于下一个测试组 $r+1$，用以下的方法进行生成：
- 建立一个候选组集，其中应该包括 M 个候选的测试组，每一个候选测试组都用如下的方法生成：

- 一开始候选组为空，为了确定第一个参数和它的取值，遍历每一个没有被选取的参数的所有可能取值，选择在未覆盖对集中出现频率最高的那一个参数和它的取值，记为 (k_i, v_i) 。将其确定并纳入候选测试组中。
- 对于之后的参数进行确定。当还有参数没有确定时，假设已经确定了 $t-1$ 个参数，接下来随机从未确定的参数中选取一个记为 k_t，此时分两种情况：
  - 当 $t < T$ ，此时候选测试组的参数数量还达不到指定的强度。遍历该参数的所有可能取值，将每个取值和已经确定的所有参数进行组合，观察该组合中 $ucps$ 出现的频率。选择频率最高的那一个值作为真值
  - 当 $t > T$ ，此时候选测试组的参数数量达到了指定的强度。遍历该参数的所有可能取值，对于每一个可能取值，需要从 t 个已经确定的参数中选择 $T-1$ 个，和当前取值进行组。由于有 C_t^{T-1} 个选取方案，将所有值加起来得到总数。参数的真实取值应该让这个总数最大。

  对于这 M 个候选的测试组，选择其中覆盖了 UCP 数量最多的那个，作为真实的候选组加入到 CA 中去。

  之后更新$ucps$和cps。
- 重复上述过程，直到 $ucps$ 为空并且 cps 中元素的数量和应有覆盖对的数量相同

编程实现

伪代码描述

```
def AETG(params, wise_num)

    """
    :param params:      LIST    list contains the number of possible value of each params
    :param wise_num:    INT     wise number, which represents the intensity of CA.
    :return:            MATRIX  represents CA, each position is a index of value as to a specified parameter.
    """

    params_num := len(params)    # K

    covered_pairs := set()   # contains all pairs(including higher dimension)

    uncovered_pairs := set()

    get all covered pairs and add them into uncovered_pairs

    M = 50 # M is the number of candidates. STEP 1

    CA = []  # the matrix of Covering Array, which is the return value

    while len(covered_pairs) != all_pairs_num: # STEP 6

        candidates = []

        for j in range(M):  # STEP 5

            candidate = []

            # STEP 2
            find (f1, v1), where v1 has the highest frequency in uncovered pairs.

            candidate[f1] = v1

            # STEP 3, 4
            while there is still parameter not selected:

                randomly choose the next parameter, which is not selected yet.

                if has_selected_params_num < wise_num:

                     for vi in range(params[f]):

                         find the frequency combination (candidate, vi) has in ucps

                     v := vi where vi has the highest frequency

                else:

                    for vi in range(params[f]):

                        for t-1 dimension array in candidate:

                            calculate covered ucp number for combination (vi, t-1 dimension array)

                    v = vi where vi has the highest frequency

                # update
                candidate[f] = v

            candidates.append(candidate)

        find the optimal candidate, which covers the most number of ucp

        CA.append(candidates[optimal_candidate_index])

    return CA
```

Python实现

由于Python自身有着集合的数据类型可以使用，且对于算法实现较为简单，故本次使用Python进行编程实现。

源代码可以参见工程文件中的 main.py。

其中，调用了 numpy 库的列表元素求积功能， random 库的随机函数功能， time 库的时间戳功能，并且使用 openpyxl 库进行excel文件的读写。

为了优化结果，对程序循环了10遍，然后选取了结果中最优的那个CA作为结果。

程序可以直接运行，实现了不限制强度的 AETG算法，并且根据实际两个网站的数据情况，对两个网站的测试集进行了生成。程序的运行结果是两个excel文件，包含了生成的测试集。

程序测试结果

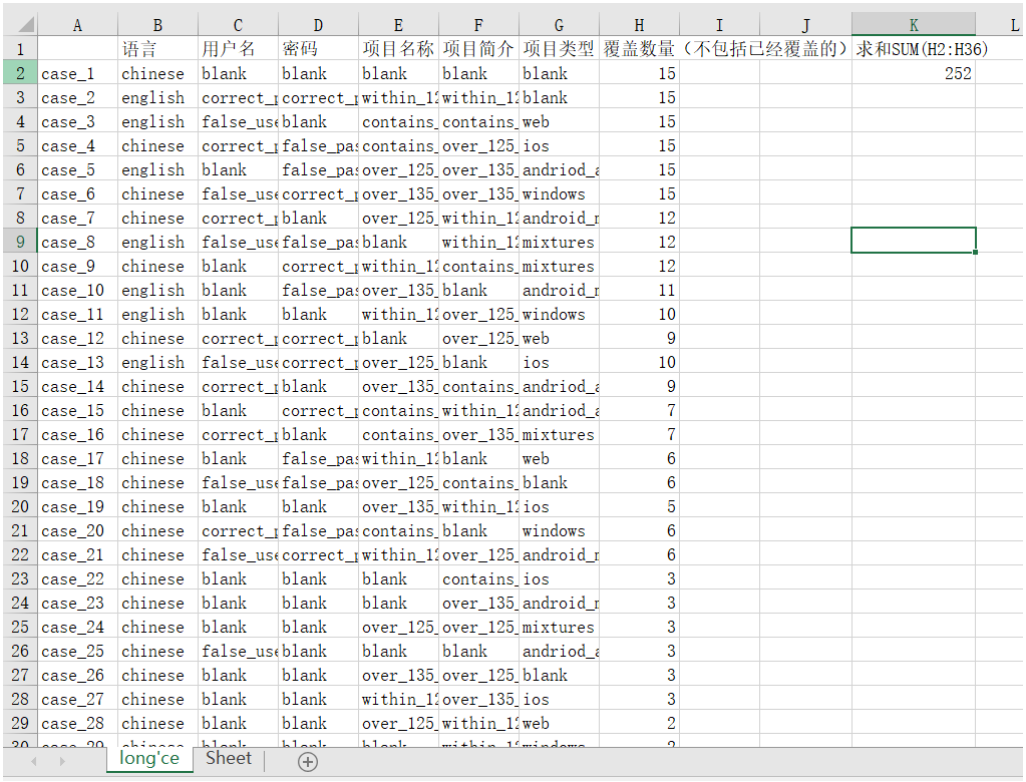
龙测平台登录并创建项目用例生成

对平台的数据进行调研，设置了如下的参数和其可能取值：

```
language = ['chinese', 'english']
user_name = ['blank', 'correct_password', 'false_username']
password = ['blank', 'correct_password', 'false_password']
proj_name = ['blank', 'within_125_bits', 'contains_special_char', 'over_125_bits',
             'over_135_bits_and_contains_special_char']
proj_description = ['blank', 'within_125_bits', 'contains_special_char', 'over_125_bits',
                    'over_135_bits_and_contains_special_char']
proj_type = ['blank', 'web', 'ios', 'andriod_apps', 'windows', 'android_miniprograms', 'mixtures']
```

由于覆盖集的强度为2，那么该问题可以被描述为 $CA(N;6,2,(2,3,3,5,5,7))$。经过计算，该问题一共有252个pair需要覆盖。

将相关数据带入算法并运行，得到了 longce.xlsx 文件，该文件可以在result中找到并查看。部分结果如下：



	A	B	C	D	E	F	G	H	I	J	K	L
1		语言	用户名	密码	项目名称	项目简介	项目类型	覆盖数量（不包括已经覆盖的）			求和SUM(H2:H36)	
2	case_1	chinese	blank	blank	blank	blank	blank	15			252	
3	case_2	english	correct_	correct_	within_1	within_1	blank	15				
4	case_3	english	false_us	blank	contains_	contains_	web	15				
5	case_4	chinese	correct_	false_pas	contains_	over_125_	ios	15				
6	case_5	english	blank	false_pas	over_125_	over_135_	andriod_	15				
7	case_6	chinese	false_us	correct_	over_135_	over_135_	windows	15				
8	case_7	chinese	correct_	blank	over_125_	within_1	android_	12				
9	case_8	english	false_us	false_pas	blank	within_1	mixtures	12				
10	case_9	chinese	blank	correct_	within_1	contains_	mixtures	12				
11	case_10	english	blank	false_pas	over_135_	blank	android_	11				
12	case_11	english	blank	blank	within_1	over_125_	windows	10				
13	case_12	chinese	correct_	correct_	blank	over_125_	web	9				
14	case_13	english	false_us	correct_	over_125_	blank	ios	10				
15	case_14	chinese	correct_	blank	over_135_	contains_	andriod_	9				
16	case_15	chinese	blank	correct_	contains_	within_1	andriod_	7				
17	case_16	chinese	correct_	blank	contains_	over_135_	mixtures	7				
18	case_17	chinese	blank	false_pas	within_1	blank	web	6				
19	case_18	chinese	false_us	false_pas	over_125_	contains_	blank	6				
20	case_19	chinese	blank	blank	over_135_	within_1	ios	5				
21	case_20	chinese	correct_	false_pas	contains_	blank	windows	6				
22	case_21	chinese	false_us	correct_	within_1	over_125_	android_	6				
23	case_22	chinese	blank	blank	blank	contains_	ios	3				
24	case_23	chinese	blank	blank	blank	over_135_	android_	3				
25	case_24	chinese	blank	blank	over_125_	over_125_	mixtures	3				
26	case_25	chinese	false_us	blank	blank	blank	andriod_	3				
27	case_26	chinese	blank	blank	over_135_	over_125_	blank	3				
28	case_27	chinese	blank	blank	within_1	over_135_	ios	3				
29	case_28	chinese	blank	blank	over_125_	within_1	web	2				

long'ce Sheet

可以看到，一共生成了35个测试组，并标记出了每个组覆盖的pair数量（不包括之前已经覆盖的）。将其求和，发现等于252，说明我们的 CA 已经完成了对于测试用例的覆盖。

所以该问题可以被描述为 $CA(35;6,2,(2,3,3,5,5,7))$。

京东网站平板电视搜索用例生成

对平台的数据进行调研，设置了如下的参数和其可能取值：

```
brand = ['blank', 'xiaomi', 'chuangwei', 'hisense', 'tcl', 'sony', 'changhong', 'konkia', 'haier', 'samsung',
         'huawei', 'philips', 'coocaa', 'sharp', 'panasonic', 'vidaa', 'lg', 'maxhub', 'vih']
size = ['blank', '78+', '70-75', '65', '58-60', '55', '48-50', '39-45', '32-']
efficiency = ['blank', '1', '2', '3', 'none']
resolution = ['blank', '7680*4320', '3840*2160', '1920*1080', '1366*768', '1024*600']
tv_type = ['blank', 'OLED', 'full', 'AI', 'big-screen', 'edu', 'smart', 'ultrathin', '4K', 'quantum', 'curve', 'laser',
           'commercial', 'game']
brand_type = ['blank', 'joint', 'domestic', 'internet', 'commercial']
dist = ['blank', '3.5+', '3-3.5', '2.5-3', '2-2.5', '2-']
user_better_select = ['blank', 'jd_goods', 'new_goods']
channel = ['blank', 'offline', 'online']
```

由于覆盖集的强度为3，那么该问题可以被描述为 $CA(2448; 9, 3, (19, 9, 5, 6, 14, 5, 6, 3, 3))$。经过计算，该问题一共有33626个pair需要覆盖。

将相关数据带入算法并运行，得到了 jingdong.xlsx 文件，该文件可以在result中找到并查看。部分结果如下：

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		品牌	屏幕尺寸	能效等级	分辨率	电视类型	品牌类型	观看距离	用户优选	产品渠道	覆盖数量（不包括已经覆盖的）			各覆盖数量之和	
2	case_1	blank	blank	blank	blank	blank	blank	blank	blank	blank	84			33626	
3	case_2	xiaomi	78+	1	7680*432(	blank	joint	blank	jd_goods	offline	84				
4	case_3	chuangwe:	70-75	1	blank	OLED	blank	3.5+	new_goods	offline	84				
5	case_4	hisense	78+	2	blank	full	joint	3.5+	blank	online	84				
6	case_5	blank	78+	3	3840*216(	OLED	domestic	3.5+	jd_goods	blank	84				
7	case_6	xiaomi	65	3	blank	blank	domestic	3-3.5	new_goods	online	84				
8	case_7	tcl	58-60	none	blank	AI	internet	3-3.5	blank	offline	84				
9	case_8	tcl	70-75	blank	1920*108(	blank	internet	3.5+	jd_goods	online	84				
10	case_9	xiaomi	78+	2	1920*108(	AI	blank	2.5-3	new_goods	blank	84				
11	case_10	xiaomi	65	1	1366*768	big-scre(	internet	3.5+	blank	blank	84				
12	case_11	sony	blank	2	blank	big-scre(	domestic	2.5-3	jd_goods	offline	84				
13	case_12	blank	70-75	none	7680*432(	full	joint	3-3.5	new_goods	blank	84				
14	case_13	blank	65	2	7680*432(	edu	blank	2-2.5	jd_goods	online	84				
15	case_14	chuangwe:	70-75	2	3840*216(	smart	commercia	2-2.5	blank	blank	84				
16	case_15	hisense	blank	blank	7680*432(	edu	internet	2-	new_goods	offline	84				
17	case_16	blank	58-60	1	1920*108(	smart	domestic	2.5-3	blank	online	84				
18	case_17	chuangwe:	58-60	none	1366*768	OLED	commercia	blank	jd_goods	online	84				
19	case_18	hisense	55	3	1024*600	big-scre(	commercia	blank	new_goods	blank	84				
20	case_19	xiaomi	55	blank	1024*600	OLED	domestic	2-2.5	blank	offline	84				
21	case_20	tcl	48-50	1	1024*600	smart	blank	3-3.5	jd_goods	blank	84				
22	case_21	tcl	blank	1	3840*216(	AI	joint	2-2.5	new_goods	online	84				
23	case_22	changhon	48-50	2	1366*768	ultrathir	blank	2-	blank	offline	84				
24	case_23	changhon	55	blank	1366*768	edu	joint	2.5-3	jd_goods	blank	84				
25	case_24	konkia	55	none	3840*216(	blank	commercia	2.5-3	new_goods	offline	83				
26	case_25	chuangwe:	39-45	3	1024*600	4K	internet	2-	blank	online	84				
27	case_26	blank	39-45	2	1366*768	smart	internet	blank	new_goods	offline	82				

可以看到，一共生成了2448个测试组，并标记出了每个组覆盖的pair数量（不包括之前已经覆盖的）。将其求和，发现等于33626，说明我们的 CA 已经完成了对于测试用例的覆盖。

所以该问题可以被描述为 $CA(2448; 9, 3, (19, 9, 5, 6, 14, 5, 6, 3, 3))$.