浙江大學

AETG算法设计与测试

题　　目：	AETG算法设计与测试
授课教师：	师江帆
姓　　名：	王伟杰
学　　号：	3210106034
日　　期：	2023-12-16

1 任务介绍

要求实现AETG算法，并应用于两个网站的组合测试（CIT）建模。具体要求如下：

1. **输入**：两个网站的CIT建模。
   - 京东官网（Web）：京东官网链接
   - 携程定机票（Web）：携程机票链接
2. **输出**：生成一个覆盖Pair-Wise的Covering Array。
3. **额外要求**：
   - 生成的Covering Array必须是通用的，证明3-wise也有效。
   - 提交材料需包括Readme文件、代码和报告。

2 算法设计

AETG算法是一个基于组合设计的测试方法。它使用组合设计生成测试集，以覆盖系统测试参数的成对、三元或n元组合。这些参数决定了系统的测试场景，例如系统配置参数、用户输入和其他外部事件。AETG算法的特点是，其生成的测试集大小与测试参数数量的增长呈对数关系，这使得测试者能够定义具有数十个参数的测试模型。

AETG系统利用新的组合算法生成测试集，覆盖所有有效的n元参数组合。在多种应用中，例如单元测试、系统测试和互操作性测试中，AETG系统已被用来生成高级测试计划和详细的测试案例。在一些应用中，它大大降低了测试计划开发的成本。

这种方法的动机是，许多系统中的问题故障通常是由少数测试参数的相互作用引起的。理想情况下，测试计划应该覆盖这些相互作用。第二个动机是，为固定的n值覆盖所有n元参数组合所需的测试数量与参数数量的增长呈对数关系。这使测试者能够定义具有数十个参数的测试模型，同时只需要很少的测试案例。

1. 算法概述

- **参数定义**：系统有 K 个参数，每个参数 k_i有 l_i个可能的值。
- **覆盖数组**（CA）：由 N 个测试用例组成，每个测试用例是 K 参数的一个特定组合。CA 应满足，任意 N*T 的子数组包含所有 T 参数的所有组合。

- **贪心策略**：通过贪心选择生成测试用例，以尽快覆盖所有参数组合。

2. 算法步骤

- **初始化**：设置参数数 K，每个参数的值的数量 l_i，以及测试的强度 T（通常是 2，意味着要覆盖所有参数对的组合）。
- **生成候选测试用例**：随机选择参数和其值，循环直至为所有参数分配值，形成一个测试用例。重复此过程 C 次生成 C 个候选测试用例。
- **选择最优测试用例**：从这些候选中选择一个覆盖最多未覆盖参数对的测试用例，将其加入 CA。
- **重复过程**：重复上述过程直到所有参数对都被覆盖。
- **优化 CA**：重复整个算法 R 次，从中选择具有最小测试用例数的 CA。

3. 具体实现

在这段代码中，`AETG` 函数实现了上述算法。主要包括以下部分：

- **参数准备**：定义了参数数、覆盖对、未覆盖对等。
- **候选测试用例生成**：通过随机选择和贪心算法生成候选测试用例。
- **最优测试用例选择**：基于覆盖最多未覆盖对的原则选择测试用例。
- **覆盖对更新**：每次选择测试用例后，更新已覆盖和未覆盖的对的集合。
- **重复和优化**：重复整个过程 R 次，从中选择最优的 CA。

3 伪代码

1. 定义 `test_case_jd()` 函数来生成京东网站的测试用例：
    - 设置测试用例的参数，如品牌、屏幕比例等。
    - 调用 `AETG` 函数生成覆盖数组（CA）。
    - 将结果写入 Excel 文件。
2. 定义 `test_case_xc()` 函数来生成携程网站的测试用例：
    - 设置测试用例的参数，如出发地、目的地等。
    - 同样调用 `AETG` 函数生成覆盖数组。
    - 将结果写入另一个 Excel 文件。
3. 实现 `AETG` 函数，它是 AETG 算法的核心：
    - 定义参数、覆盖对、未覆盖对、候选数组等。
    - 循环生成测试用例，直到所有的组合都被覆盖。
    - 返回最优的覆盖数组和每个测试用例覆盖的对数。

4. 实现一些辅助函数：

   - `calc_covered_pairs_num`：计算覆盖对的数量。
   - `get_optimal_CA`：从候选中选出最优的覆盖数组。
   - `get_uncovered_pairs_in_subset`：获取子集中的未覆盖对。
   - `is_part_of_uncovered_pair`：检查一个组合是否是未覆盖对的一部分。
   - `calc_covered_ucp_num_for_next_value`：计算下一个值覆盖的未覆盖对数。
   - `calc_new_case_generated`：计算新生成的测试用例覆盖了多少新的对。
   - `update_covered_pairs`：更新已覆盖和未覆盖的对。

5. 在主函数中，调用 `test_case_jd()` 和 `test_case_xc()` 来生成测试用例。

4 结果

覆盖集的强度设置为2，问题描述为$CA(N;7,2,(15,5,4,13,5,17,9))$，一共有255个用例需要覆盖，京东生成的数据如下：

	品牌	屏幕比例	能效等级	固态硬盘	材质	内存容量	屏幕刷新	覆盖数量
case_1	all	all	all	all	all	all	all	21
case_2	thinkpad	16:10	1	3TB	all	4GB	60Hz	21
case_3	dell	16:10	2	128GB	金属	6GB	all	21
case_4	dell	16:9	3	all	金属+复合	4GB	165Hz	21
case_5	huawei	4:3	1	128GB	金属+复合	all	360Hz	21
case_6	all	3:2	2	3TB	复合材质	48GB	165Hz	21
case_7	dell	all	1	256GB+1TI	含碳纤维	48GB	60Hz-120I	21
case_8	hp	3:2	1	all	金属	18GB	90Hz	21
case_9	thinkpad	all	3	128GB	复合材质	18GB	120Hz	21
case_10	thinkpad	4:3	2	all	含碳纤维	192GB	144Hz	21
case_11	lenvo	16:10	all	256GB+1TI	金属+复合	18GB	240Hz	21
case_12	huawei	16:9	all	3TB	金属	192GB	60Hz-120I	21
case_13	apple	4:3	all	512GB+2TI	复合材质	4GB	90Hz	21
case_14	hp	16:9	2	3TB2	all	8GB	360Hz	21
case_15	asus	3:2	all	64GB	含碳纤维	6GB	60Hz	21
case_16	all	4:3	3	240GB	金属	8GB	60Hz	21
case_17	lenvo	16:9	1	4TB*2	复合材质	12GB	all	21
case_18	huawei	3:2	3	256GB	all	12GB	144Hz	21
case_19	apple	all	2	4TB*3	金属+复合	12GB	60Hz	21
case_20	hp	16:10	3	512GB	含碳纤维	16GB	all	20
case_21	mi	all	all	512GB*2	金属	16GB	165Hz	19
case_22	honor	4:3	1	64GB	all	20GB	165Hz	19
case_23	dere	16:10	2	64GB	复合材质	all	60Hz-120I	19
case_24	acer	all	3	3TB	金属+复合	6GB	90Hz	18
case_25	mechrevo	16:9	3	512GB+2TI	含碳纤维	all	240Hz	19
case_26	microsof	16:9	1	240GB	all	6GB	120Hz	18
case_27	haier	4:3	3	4TB*2	all	24GB	60Hz-120I	18
case_28	thinkpad	3:2	all	3TB2	金属+复合	32GB	all	18
case_29	hp	all	all	240GB	金属+复合	36GB	144Hz	18
case_30	honor	all	all	4TB*2	金属	40GB	360Hz	18
case_31	acer	16:10	all	256GB	金属	64GB	120Hz	18
case_32	asus	16:9	2	256GB+1TI	all	96GB	90Hz	18
case_33	dere	all	1	512GB	all	128GB	240Hz	18

共有255个测试数据，实现全覆盖

携程问题描述为$CA(N;7,2,(8,8,6,6,14,2,4))$，一共有112个用例需要覆盖，测试数据如下：

	出发地	目的地	出发日期	返回日期	乘客数量	直飞	舱型	覆盖数量
case_1	beijing	beijing	2021-05-(	2021-05-(	1	true	经济/超经	21
case_2	shanghai	shanghai	2021-05-(	2021-05-(	2	false	公务/头等	21
case_3	shanghai	guangzhou	2021-05-(	2021-05-(	3	true	公务舱	21
case_4	guangzhou	shenzhen	2021-05-(	2021-05-(	4	true	头等舱	21
case_5	guangzhou	beijing	2021-05-(	2021-05-(	5	false	公务舱	21
case_6	shenzhen	chengdu	2021-05-(	2021-05-(	5	true	公务/头等	21
case_7	shenzhen	guangzhou	2021-05-(	2021-05-(	1	false	头等舱	21
case_8	chengdu	shanghai	2021-05-(	2021-05-(	6	true	经济/超经	20
case_9	chongqing	chengdu	2021-05-(	2021-05-(	3	false	经济/超经	20
case_10	hangzhou	chongqing	2021-05-(	2021-05-(	7	true	公务舱	19
case_11	nanjing	hangzhou	2021-05-(	2021-05-(	5	true	经济/超经	19
case_12	beijing	nanjing	2021-05-(	2021-05-(	4	false	公务/头等	20
case_13	hangzhou	hangzhou	2021-05-(	2021-05-(	6	false	头等舱	20
case_14	chengdu	chongqing	2021-05-(	2021-05-(	5	false	头等舱	18
case_15	chongqing	nanjing	2021-05-(	2021-05-(	8	true	公务舱	18
case_16	nanjing	shenzhen	2021-05-(	2021-05-(	7	false	公务/头等	18
case_17	chengdu	beijing	2021-05-(	2021-05-(	9	true	公务/头等	16
case_18	shanghai	shenzhen	2021-05-(	2021-05-(	10	true	经济/超经	16
case_19	shenzhen	chongqing	2021-05-(	2021-05-(	11	true	经济/超经	15
case_20	nanjing	nanjing	2021-05-(	2021-05-(	12	true	头等舱	16
case_21	beijing	shanghai	2021-05-(	2021-05-(	13	true	公务舱	15
case_22	chongqing	beijing	2021-05-(	2021-05-(	2	true	头等舱	15
case_23	guangzhou	guangzhou	2021-05-(	2021-05-(	2	true	经济/超经	13
case_24	hangzhou	nanjing	2021-05-(	2021-05-(	9	false	经济/超经	14
case_25	beijing	chengdu	2021-05-(	2021-05-(	9	true	头等舱	14
case_26	guangzhou	hangzhou	2021-05-(	2021-05-(	8	true	公务/头等	14
case_27	chongqing	chongqing	2021-05-(	2021-05-(	10	false	公务/头等	14
case_28	guangzhou	chengdu	2021-05-(	2021-05-(	11	false	公务舱	14
case_29	shenzhen	hangzhou	2021-05-(	2021-05-(	12	false	公务舱	14
case_30	shenzhen	beijing	2021-05-(	2021-05-(	13	false	经济/超经	13
case_31	shanghai	chongqing	2021-05-(	2021-05-(	5	true	头等舱	12
case_32	chengdu	shenzhen	2021-05-(	2021-05-(	1	true	公务舱	12
case_33	hangzhou	guangzhou	2021-05-(	2021-05-(	11	true	公务/头等	12
case_34	nanjing	shanghai	2021-05-(	2021-05-(	4	true	公务舱	12

共112个测试数据，实现全覆盖