Попарне тестування та Testing Based on Combinatorial Design

Білецький Сергій, Ларін Владислав

План

- Попарне тестування
- Використання ортогональних масивів
- Використання алгоритму добору пар
- Переваги та недоліки попарного тестування
- Інструментальні методи підтримки
- Combinatorial Testing

Попарне тестування

- **Попарне тестування** або **попарний аналіз** (англ. **Pairwise testing**) — це сучасна й ефективна методика тестування, заснована на тому припущенні, що більшість дефектів виникає при взаємодії не більше двох факторів.
- Призначено для перекриття всіх поєднань пар змінних без урахування всіх можливих комбінацій інших зміних
- Дозволяє суттєво зменшити кількість тестів

Попарне тестування

- Для тестування системи з великим числом параметрів, як приклад, перевірка роботи системи під різними операційними системами або робота сайту в різних браузерах.
- Хто знає, яке поєднання параметрів призведе до збою?
- 10 параметрів з 10 значеннями необхідно 10 мільярдів тестів, метод всіх пар використовує всього 177 тестів.

Застосування

- Визначитись з функціональністю, яку треба перевірити
- Дослідити обраний сценарій і виявити його і їх значення.
- Затосувати алгоритм, який створює оптимальне число тестів з повним перебором пар

Ортогональні масиви

- Ортогональний масив (ортогональна таблиця) — це таблиця, яка має наступні властивості:
  - Будь-які два стовпці таблиці містять всі комбінації значень цих стовпців.
  - Якщо яка-небудь пара значень двох стовпців зустрічається кілька разів, то всі можливі парні комбінації значень цих стовпців повинні зустрітися стільки ж разів.
  - У ортогональних масивах необов'язково всі стовпці повинні мати однакову кількість значень. Існують так звані змішані (mixed) ортогональні масиви.

Тестування з ортогональними масивами

- Визначають змінні для вхідних даних в комбінаціях. Наприклад, це можуть бути назви опцій, параметрів налаштувань, допустимих конфігурацій устаткування і т. П.
- Визначають значення, які можуть приймати змінні. Наприклад, конкретні назви пунктів меню, числові значення, назви операційних систем або баз даних і т. П.
- Будують ортогональний масив, який має стовпець для кожної змінної.
- Кожен рядок побудованого масиву інтерпретується як одна комбінація значень змінних для одного тестового випадку.

All-Pairs алгоритм

- All-Pairs Algorithm (алгоритм всіх пар) — це комбінаторна методика, яка була спеціально створена для попарного тестування. В її основі лежить вибір можливих комбінацій значень всіх змінних, в яких містяться всі можливі значення для кожної пари змінних. Виходячи з визначення при цьому буде отримано менше число комбінацій, ніж при використанні ортогональних масивів.

Тестування з використанням All-Pairs

- Для тестування з використанням All-Pairs алгоритму виконують наступні кроки:
  - Аналогічно, як для ортогональних масивів, визначають таблицю всіх змінних і їх значень.
  - Залишають в таблиці тільки всі можливі унікальні комбінації пар значень змінних.

Тестування поєднанням по N

- Концепція по парного тестування може бути розширена до тестування по N змінних
- Збільшує якість перекриття коду
- Не підходить для варіанту, коли користувач робить вибір параметрів в довільному випадку, але при цьому важливий сам порядок

Покрытие тестами сочетаниями по N		
N	Экономная	Профессиональная
1	7	7
2	31	36
3	110	179
4	318	749
5	814	2812

Приклад

Параметр 1	Параметр 2	Параметр 3
Значение 1.1	Значение 2.1	Значение 3.1
Значение 1.2	Значение 2.2	Значение 3.2

Приклад

#	Параметр 1	Параметр 2	Параметр 3
1	Значение 1.1	Значение 2.1	Значение 3.1
2	Значение 1.1	Значение 2.2	Значение 3.1
3	Значение 1.2	Значение 2.1	Значение 3.1
4	Значение 1.2	Значение 2.2	Значение 3.1
5	Значение 1.1	Значение 2.1	Значение 3.1
6	Значение 1.1	Значение 2.1	Значение 3.2
7	Значение 1.2	Значение 2.1	Значение 3.1
8	Значение 1.2	Значение 2.1	Значение 3.2
9	Значение 1.1	Значение 2.1	Значение 3.1
10	Значение 1.1	Значение 2.1	Значение 3.2
11	Значение 1.1	Значение 2.2	Значение 3.1
12	Значение 1.1	Значение 2.2	Значение 3.2

#	Параметр 1	Параметр 2	Параметр 3
1	Значение 1.1	Значение 2.1	Значение 3.1
2	Значение 1.1	Значение 2.2	Значение 3.1
3	Значение 1.2	Значение 2.1	Значение 3.1
4	Значение 1.2	Значение 2.2	Значение 3.1
5	Значение 1.1	Значение 2.1	Значение 3.2
6	Значение 1.2	Значение 2.1	Значение 3.2
7	Значение 1.1	Значение 2.2	Значение 3.2

Приклад

	Параметр 1	Параметр 2	Параметр 3
1	Значение 1.1	Значение 2.1	Значение 3.1
2	Значение 1.1	Значение 2.2	Значение 3.2
3	Значение 1.2	Значение 2.1	Значение 3.2
4	Значение 1.2	Значение 2.2	Значение 3.1

Переваги та недоліки попарного тестування

- Переваги
  - Генерація тестів в універсальних засобах
  - Значне скорочення кількості необхідних тестів для повного перекриття
  - Можливість легко розширювати якість тестування

- Недоліки
  - У випадку коли важливий порядок застосовуваних тестів
  - Ефективний на пізніх стадіях розробки, або повинен бути доповнений функціональними тестами

Інструментальні методи підтримки

- Allpairs
- CATS (Constrained Array Test System)
- OATS (Orthogonal Array Test System)
- AETG
- IPO (PairTest)

Allpairs консольна програма

- Готове рішення allpairs.exe
- Запуск з виводом в консоль

```
C:\allpairs>allpairs.exe printing.txt
```

- Запуск з виводом в файл

```
C:\allpairs>allpairs.exe printing.txt > result.txt
```

- Створення вхідних даних у програмі Excel а далі експорту txt з розділенням використовуючи табуляцію

pps	pr	cp	pid	cont	sc	do	uf	ul	ar	bp	ppt	rpo	ip	ifc	ic	iht	ido	pdof
1	all	1	1	alltext	letter	y	y	y	y	y	y	y	y	y	y	y	y	y
2	current	2c	2	alldiagram	legal	n	n	n	n	n	n	n	n	n	n	n	n	n
4	range	2nc	3	mixed	executive													
8		5c	33		a4-210													
16		5nc			a4-small													

TEST CASES

case	pps	pr	cp	pid	cont	sc	do	uf	ul	ar	bp	ppt	rpo
1	1	all	1	1	alltext	letter	y	y	y	y	y	y	y
2	1	current	2c	2	alldiagrams		legal	n	n	n	n	n	n
3	2	range	1	3	mixed	legal	y	n	y	n	y	n	y
4	2	all	2c	33	alldiagrams		letter	n	y	n	y	n	y
5	4	current	2nc	1	mixed	executive		n	y	y	n	n	y
6	4	range	5c	2	alltext	a4-210	y	n	n	y	y	n	n
7	8	all	2nc	3	alldiagrams		a4-210	y	y	n	n	n	n
8	8	current	5c	33	alltext	executive		n	n	y	y	y	y
9	16	range	5nc	1	alldiagrams		a4-small		n	n	y	y	n
10	16	all	5nc	2	mixed	a4-small		y	y	n	n	y	y
11	1	range	2nc	3	alltext	letter	n	n	~n	n	y	y	n
12	2	current	5c	33	mixed	legal	y	y	~n	y	n	n	y
13	4	all	1	3	alldiagrams		executive		n	n	n	y	n
14	8	range	2c	1	mixed	a4-210	y	y	y	~n	y	y	n
15	16	all	2c	33	alltext	legal	~n	~n	~y	n	n	y	y
16	1	range	5c	2	alldiagrams		executive		y	~y	y	n	y

Allpairs python додаток

- Працює з N-парами

```
import metacomm.combinatorics.all_pairs2
all_pairs = metacomm.combinatorics.all_pairs2.all_pairs2
"""
Demo of the basic functionality - just getting pairwise/n-wise combinations
"""
parameters = [ [ "Brand X", "Brand Y" ]
             , [ "98", "NT", "2000", "XP"]
             , [ "Internal", "Modem" ]
             , [ "Salaried", "Hourly", "Part-Time", "Contr." ]
             , [ 6, 10, 15, 30, 60 ]
             ]
triplewise = all_pairs( parameters, n=5)
print "TRIPLEWISE:"
for i, v in enumerate(triplewise):
    print "%i:\t%s" % (i, str(v))
```

```
C:\Users\Lenovo\Desktop\AllPairs-2.0.1\examples>python example1.2.py
TRIPLEWISE:
0:      ['Brand X', '98', 'Internal', 'Salaried', 6]
1:      ['Brand Y', 'NT', 'Modem', 'Hourly', 6]
2:      ['Brand Y', '2000', 'Modem', 'Part-Time', 6]
3:      ['Brand X', 'XP', 'Internal', 'Contr.', 6]
4:      ['Brand X', 'XP', 'Internal', 'Contr.', 10]
5:      ['Brand Y', '2000', 'Modem', 'Part-Time', 10]
6:      ['Brand Y', 'NT', 'Modem', 'Hourly', 10]
7:      ['Brand Y', '98', 'Modem', 'Salaried', 6]
8:      ['Brand X', '98', 'Internal', 'Salaried', 10]
9:      ['Brand X', 'XP', 'Internal', 'Contr.', 15]
10:     ['Brand X', 'NT', 'Modem', 'Hourly', 6]
11:     ['Brand Y', 'NT', 'Modem', 'Hourly', 15]
12:     ['Brand Y', '2000', 'Internal', 'Part-Time', 6]
13:     ['Brand Y', 'NT', 'Modem', 'Salaried', 6]
14:     ['Brand Y', '98', 'Modem', 'Salaried', 10]
15:     ['Brand Y', '98', 'Modem', 'Salaried', 15]
16:     ['Brand Y', '2000', 'Modem', 'Part-Time', 15]
17:     ['Brand Y', 'XP', 'Internal', 'Part-Time', 6]
18:     ['Brand X', '98', 'Internal', 'Salaried', 15]
19:     ['Brand X', '98', 'Internal', 'Salaried', 30]
20:     ['Brand Y', 'NT', 'Modem', 'Hourly', 30]
21:     ['Brand X', 'XP', 'Internal', 'Contr.', 30]
22:     ['Brand X', '2000', 'Modem', 'Part-Time', 6]
23:     ['Brand Y', '2000', 'Modem', 'Part-Time', 30]
24:     ['Brand Y', '98', 'Modem', 'Salaried', 30]
25:     ['Brand Y', '98', 'Modem', 'Salaried', 60]
26:     ['Brand Y', '2000', 'Internal', 'Part-Time', 30]
27:     ['Brand Y', '2000', 'Internal', 'Part-Time', 15]
28:     ['Brand Y', '2000', 'Internal', 'Part-Time', 10]
29:     ['Brand Y', '2000', 'Internal', 'Part-Time', 60]
30:     ['Brand Y', 'NT', 'Modem', 'Hourly', 60]
31:     ['Brand Y', 'XP', 'Internal', 'Contr.', 6]
32:     ['Brand X', 'XP', 'Internal', 'Contr.', 60]
33:     ['Brand Y', '2000', 'Modem', 'Part-Time', 60]
```

```
C:\Users\Lenovo\Desktop\AllPairs-2.0.1\examples>python example1.2.py
TRIPLEWISE:
0:      ['Brand X', '98', 'Internal', 'Salaried', 6]
1:      ['Brand Y', 'NT', 'Modem', 'Hourly', 6]
2:      ['Brand Y', '2000', 'Internal', 'Part-Time', 10]
3:      ['Brand X', 'XP', 'Modem', 'Contr.', 10]
4:      ['Brand X', '2000', 'Modem', 'Part-Time', 15]
5:      ['Brand Y', 'XP', 'Internal', 'Hourly', 15]
6:      ['Brand Y', '98', 'Modem', 'Salaried', 30]
7:      ['Brand X', 'NT', 'Internal', 'Contr.', 30]
8:      ['Brand X', '98', 'Internal', 'Hourly', 60]
9:      ['Brand Y', '2000', 'Modem', 'Contr.', 60]
10:     ['Brand Y', 'NT', 'Modem', 'Salaried', 60]
11:     ['Brand Y', 'XP', 'Modem', 'Part-Time', 60]
12:     ['Brand Y', '2000', 'Modem', 'Hourly', 30]
13:     ['Brand Y', '98', 'Modem', 'Contr.', 15]
14:     ['Brand Y', 'XP', 'Modem', 'Salaried', 15]
15:     ['Brand Y', 'NT', 'Modem', 'Part-Time', 15]
16:     ['Brand Y', 'XP', 'Modem', 'Part-Time', 30]
17:     ['Brand Y', '98', 'Modem', 'Part-Time', 6]
18:     ['Brand Y', '2000', 'Modem', 'Salaried', 6]
19:     ['Brand Y', '98', 'Modem', 'Salaried', 10]
20:     ['Brand Y', 'XP', 'Modem', 'Contr.', 6]
21:     ['Brand Y', 'NT', 'Modem', 'Hourly', 10]
```

Combinatorial Testing

Тестування випадкового набору комбінацій вхідних параметрів, замість повного перебору.

Проблеми:

Які гарантії це надає?

Покриття коду?

В наслідок стратегії чорного ящику Code Coverage від випадкового набору тестів невідомий, тому ми не можемо використати його як критерій якості

Рівень достовірності

Натомість припустимо щоб наше ПЗ було вільно від помилок на **X%** з рівнем достовірності **Y%**

Рівень достовірності в **Y%** говорить про те, що якщо ми будемо постійно проводити експерименти (тобто в нашому прикладі - робити випадкову вибірку певного розміру з безлічі вхідних даних і запускати на ній тести), то в **Y%** випадків результати цих експериментів будуть знаходитися в межах допустимого (тобто серед них буде не більше **1% (при X = 99%)** завершилися неуспішно).

Приклад

99% рівень достовірності для 99% рівня якості	
Кількість унікальних комбінацій параметрів	Кількість необхідних тестів (перевірок)
100	99
1 000	943
10 000	6 247
100 000	14 627
1 000 000	16 369
10 000 000	16 613
100 000 000	16 638
Неизвестно	16 641

Залежність кількості тестів від рівня якості (всього можл. тестів 1 млн)

99% рівень достовірності для різних рівнів якості	
Бажаний рівень якості	Кількість необхідних тестів
90%	166
95%	9 513
99%	16 369
99,9%	624 639
99,99%	994 027
99,999%	999 940

Умови застосування Combinatorial Testing

- Обрання досить помірного рівня якості (99% - 99,9%)
- Чесна генерація випадкової вибірки (період генератору більший за кількість тестів)