1 Introduction

The purpose of this lab was to become familiar with several black-box testing techniques. The techniques introduced in this lab were Extreme Point Combination (EPC), and weak n x 1 testing. We will be testing two Java programs – Drone and RemoteCar. In EPC testing, we complete a single domain analysis for a given subdomain in order to identify the domain limits for each dimension. Then, we can choose test cases for each domain – max, min, slightly under min, and slightly over max. Another additional test case is added within the valid subdomain. Together, this is $4^n + 1$ cases, where n is number of input variables. In weak nx1 strategy, we try to find linear boundaries of the problem using domain analysis. In domain analysis, n points are selected on each linear boundary, where n is the amount of input variables. One additional point is also chosen outside of each boundary. If the boundary is open, then all points on the boundary receive exterior processing. In this case, choose an additional point within the boundary. If the boundary is closed, then all points on the boundary receive interior processing. In this case, choose an additional point outside the boundary. This strategy produces $b(n+1)+1$ test cases, where b is the number of linear boundaries, and n is the number of input variables.

2 Part One - Drone program

2.1 Q1

For task one, the Drone application was tested using EPC and weak nx1 strategy. Drone is a new GPS-enabled automated pilot system being testing for unmanned reconnaissance aircraft. The system controls the unmanned aircraft from takeoff to landing. The unmanned aircraft has enough fuel to reach a max distance k before needing to return back to base. The aircraft can have a travel a max of 3 waypoints x_1, x_2, x_3. In this experiment, k is 100. Therefore, $x_1 + x_2 + x_3 \leq k$ where x_1, x_2, x_3 must be ≥ 0. The program should output "Success" if it is successful and "Failure" if it is not.

2.2 Q2

Looking at all the failed test cases, we can see that whenever x_2 is negative, the output is "Success" when it should be "Failure". For the EPC testing strategy, we were able to see 2 types of failures, when the output is incorrectly "Success" or "Failure". For the weak nx1 strategy, we were only able to see when "Success!" is incorrectly outputted.

2.3 Q3

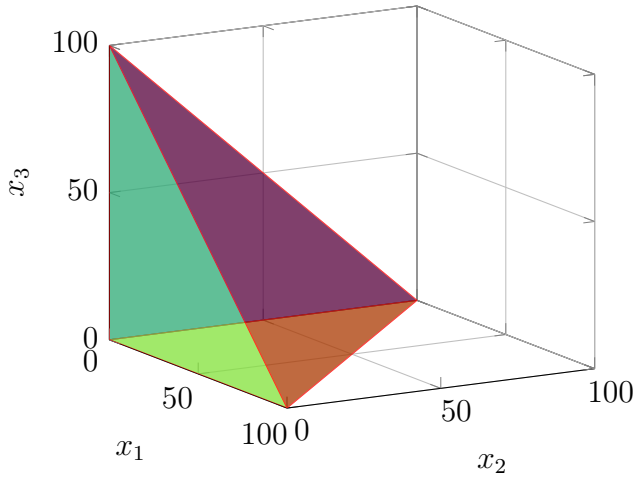

Figure 1: Subdomain of all valid inputs for drone program

2.4 Q4

The EPC approach was able to find 2 different types of failures whereas the weak nx1 approach was only able to find 1 type of failure. The two types of failures seem to have stemmed from the same issue (when $x_2 < 0$), so although the weak nx1 approach was unable to find both types, it was able to still find the primary issue causing both of the problems. It seems like weak nx1 testing is more efficient in finding problems since the weak nx1 approach was able to find it with 17 test cases compared to the 65 test cases that the EPC approach required.

2.5 Q5

EPC Testing - Drone

Test #	Description	x1	x2	x3	Expected	Actual
1	EPC case	-1	0	0	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
2	EPC case	-1	0	100	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
3	EPC case	-1	0	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
4	EPC case	-1	0	101	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
5	EPC case	-1	100	0	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
6	EPC case	-1	100	100	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
7	EPC case	-1	100	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
8	EPC case	-1	100	101	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
9	EPC case	-1	-1	0	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
10	EPC case	-1	-1	100	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
11	EPC case	-1	-1	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
12	EPC case	-1	-1	101	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
13	EPC case	-1	101	0	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
14	EPC case	-1	101	100	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
15	EPC case	-1	101	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
16	EPC case	-1	101	101	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
17	EPC case	0	-1	0	ERROR: Invald argument - negative value	Success!
18	EPC case	0	-1	100	ERROR: Invald argument - negative value	Success!
19	EPC case	0	-1	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
20	EPC case	0	-1	101	ERROR: Invald argument - negative value	Success!
21	EPC case	0	0	0	Success!	Success!
22	EPC case	0	0	100	Success!	Success!
23	EPC case	0	0	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
24	EPC case	0	0	101	Failure!	Failure!
25	EPC case	0	100	0	Success!	Success!
26	EPC case	0	100	100	Failure!	Failure!
27	EPC case	0	100	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
28	EPC case	0	100	101	Failure!	Failure!
29	EPC case	0	101	0	Failure!	Failure!
30	EPC case	0	101	100	Failure!	Failure!
31	EPC case	0	101	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
32	EPC case	0	101	101	Failure!	Failure!
33	EPC case	100	0	101	Failure!	Failure!
34	EPC case	100	100	101	Failure!	Failure!
35	EPC case	100	-1	101	ERROR: Invald argument - negative value	Failure!
36	EPC case	100	101	101	Failure!	Failure!
37	EPC case	100	0	100	Failure!	Failure!
38	EPC case	100	100	100	Failure!	Failure!
39	EPC case	100	-1	100	ERROR: Invald argument - negative value	Failure!
40	EPC case	100	101	100	Failure!	Failure!
41	EPC case	100	0	0	Success!	Success!
42	EPC case	100	100	0	Failure!	Failure!
43	EPC case	100	-1	0	ERROR: Invald argument - negative value	Success!
44	EPC case	100	101	0	Failure!	Failure!
45	EPC case	100	0	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
46	EPC case	100	100	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
47	EPC case	100	-1	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
48	EPC case	100	101	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
49	EPC case	101	0	100	Failure!	Failure!
50	EPC case	101	0	0	Failure!	Failure!
51	EPC case	101	101	101	Failure!	Failure!
52	EPC case	101	0	101	Failure!	Failure!
53	EPC case	101	-1	0	ERROR: Invald argument - negative value	Success!
54	EPC case	101	-1	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
55	EPC case	101	100	0	Failure!	Failure!
56	EPC case	101	100	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
57	EPC case	101	100	101	Failure!	Failure!
58	EPC case	101	-1	100	ERROR: Invald argument - negative value	Failure!
59	EPC case	101	101	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
60	EPC case	101	-1	101	ERROR: Invald argument - negative value	Failure!
61	EPC case	101	101	0	Failure!	Failure!
62	EPC case	101	100	100	Failure!	Failure!
63	EPC case	101	101	100	Failure!	Failure!
64	EPC case	101	0	-1	ERROR: Invald argument - negative value	ERROR: Invald argument - negative value
65	Valid case	10	20	30	Success!	Success!

Weak nx1 Testing - Drone

Test #	Description	x1	x2	x3	Expected	Actual
1	Test case in the boundary	15	15	15	Success!	Success!
4	x1 = 0, x2 + x3 < 100	0	15	35	Success!	Success!
5	x2 = 0, x1 + x3 < 100	5	0	75	Success!	Success!
3	x1 = 0, x2 + x3 < 100	0	10	55	Success!	Success!
6	x2 = 0, x1 + x3 < 100	10	0	55	Success!	Success!
8	x3 = 0, x1 + x2 < 100	5	75	0	Success!	Success!
10	x3 = 0, x1 + x2 < 100	15	35	0	Success!	Success!
7	x2 = 0, x1 + x3 < 100	15	0	35	Success!	Success!
9	x3 = 0, x1 + x2 < 100	10	55	0	Success!	Success!
2	x1 = 0, x2 + x3 < 100	0	5	75	Success!	Success!
12	x1 + x2 + x3 = 100	10	50	40	Success!	Success!
11	x1 + x2 + x3 = 100	33	33	34	Success!	Success!
13	x1 + x2 + x3 = 100	20	30	50	Success!	Success!
14	Just outside of x1 + x2 + x3 boundary	33	33	35	Failure!	Failure!
17	Just outside of x3 = 0 boundary	15	15	-1	ERROR: Invald ar	ERROR: Invalc
16	Just outside of x2 = 0 boundary	15	-1	15	ERROR: Invald ar	Success!
15	Just outside of x1 = 0 boundary	-1	15	15	ERROR: Invald ar	ERROR: Invalc

3 Part 2 - Remote Car Program

3.1 Q1

For task 2, we tested a remote-controlled car piloting system using EPC and weak nx1 testing. The car may not be any more than a distance r from the origin. The program takes Cartesian inputs in the form of (x, y) which specify where the car should move. The point must fall on the circle with radius 1, centered around the origin. If it is, the program will output "Ok"; otherwise, it will output "Out of range".

3.2 Q2

We approximated the circle using four linear equations:

$$y = \begin{cases} x + 1 & -1 \leq x \leq 0 \\ -(1 + x) & -1 \leq x \leq 0 \\ x - 1 & 0 \leq x \leq 1 \\ 1 - x & 0 \leq x \leq 1 \end{cases}$$

3.3 Q3

For the EPC testing we had $4^2 + 1 = 17$ tests. No tests failed during EPC testing and seemed more effective in testing the circle subdomain than weak nx1 testing.

The weak nx1 domain approximation seemed not very effective. It had 4 errors out of the 13 tests. It worked best if test inputs are chose quite far away from the boundaries, since it's possible for there to be a false negative if a input is chosen too close to the boundary. This is possible when test values are chosen just outside the approximated boundary. If we increased the number of lines, the number of test cases would also increase, and the accuracy would increase. Weak nx1 testing requires $b(n+1)+1$ tests, where b is the number of segments and n is the amount of input variables. If we used 12 segments, we would have $12(2+1)+1=37$ test cases.

For this situation, EPC testing is more accurate than weak nx1 testing. Weak nx1 had 4 test case failures out of 13 due to the linear approximation of the subdomain. No actual errors were found in the application. EPC testing had no test failures despite having a similar number of tests as weak nx1 testing. With more segments, weak nx1 testing will be more accurate.

3.4 Q4

EPC Testing - Remote Car

Test #	Description	x	y	Expected	Actual
1	EPC case	-1.1	1.1	Out of range!	Out of range!
2	EPC case	1	1	Out of range!	Out of range!
3	EPC case	-1.1	-1.1	Out of range!	Out of range!
4	EPC case	-1	1	Out of range!	Out of range!
5	EPC case	1.1	-1	Out of range!	Out of range!
6	EPC case	1.1	-1.1	Out of range!	Out of range!
7	EPC case	1	-1	Out of range!	Out of range!
8	EPC case	-1	-1.1	Out of range!	Out of range!
9	EPC case	1.1	1	Out of range!	Out of range!
10	EPC case	-1.1	-1	Out of range!	Out of range!
11	EPC case	-1.1	1	Out of range!	Out of range!
12	EPC case	1.1	1.1	Out of range!	Out of range!
13	EPC case	1	-1.1	Out of range!	Out of range!
14	EPC case	1	1.1	Out of range!	Out of range!
15	EPC case	-1	-1	Out of range!	Out of range!
16	EPC case	-1	1.1	Out of range!	Out of range!
17	Valid case	0.1	0.1	Ok.	Ok.

Weak nx1 Testing - Remote Car

Test #	Description	x	y	Expected	Actual
1	Test case in the boundary	0	0	Ok.	Ok.
5	y = x - 1 boundary	0.9	-0.1	Ok.	Ok.
9	y = -x - 1 boundary	-0.1	-0.9	Ok.	Ok.
2	y = x + 1 boundary	-0.9	0.1	Ok.	Ok.
8	y = -x - 1 boundary	-0.9	-0.1	Ok.	Ok.
3	y = x + 1 boundary	-0.1	0.9	Ok.	Ok.
6	y = -x + 1 boundary	0.1	0.9	Ok.	Ok.
7	y = -x + 1 boundary	0.9	0.1	Ok.	Ok.
4	y = x - 1 boundary	0.1	-0.9	Ok.	Ok.
13	Just outside y = -x - 1 boundary	-0.5	-0.6	Out of range!	Ok.
10	Just outside y = x + 1 boundary	-0.5	0.6	Out of range!	Ok.
11	Just outside y = x - 1 boundary	0.5	-0.6	Out of range!	Ok.
12	Just outside y = -x + 1 boundary	0.5	0.6	Out of range!	Ok.

4 Conclusion

In this lab, we were introduced to two black box testing techniques – Extreme Point Combination (EPC), and weak nx1 testing. We tested two Java programs – Drone and Remote Car.

In part 1, we tested the Drone program. When applying EPC testing, we found 9 failed test cases. When applying weak nx1 testing, we were only able to find 1 failed test case. These errors stemmed from the same one problem and both testing methods were able to find it. The problem we found was when $x_2 < 0$. In part 2 we tested the Remote Car program. When applying EPC testing we found no failed test cases; however when applying weak nx1 testing, we found 4 failed test cases. These test case failures were because of the linear approximation we used when applying weak nx1 testing. It was difficult to know if the program wasn't working properly or the testing was not working as intended.

One testing strategy is not better than the other. It is better to choose the testing strategy appropriate for the program being tested for the best results. The weak nx1 strategy seems to excel when the problem is already linear so no linear approximation is needed. With the linear approximation, weak nx1 testing is a lot less reliable. EPC works really well in situations that weak nx1 strategy is not so good at. Although there are many more test cases, EPC testing seems more accurate. It didn't produce any false positive cases and more failures in the program were found than in weak nx1 testing.