UNIVERSITÄT OSNABRÜCK

Towards Using Code Coverage Metrics for Performance Comparison on the Implementation Level

Mathias Menninghaus and Elke Pulvermüller

Motivation

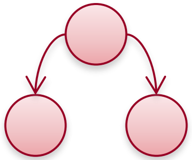
- Compare complex data structures and algorithms on the implementation level
- Either no benchmark or test set available or
- Consider problems with benchmarks:
  - Implementation may be biased to perform in a certain benchmark
  - Benchmark may not uncover best and worst cases of an implementation

Simple Example

```
public int max(int a, int b)
```

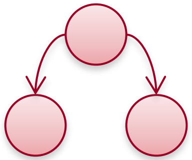
A

```
int c = a - b;
return c < 0 ? b : a;
```



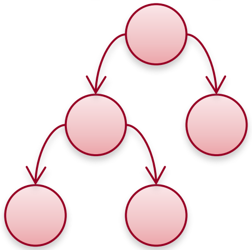
B

```
int c = b - a;
return c < 0 ? a : b;
```



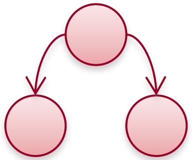
C

```
if (a == b) {
    return a;
} else {
    return a < b ? b : a;
}
```



D

```
return a < b ? b : a;
```



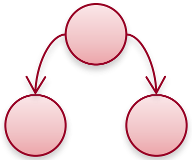
Simple Example

Generate test cases with maximized (basic block) coverage

A

```
int c = a - b;
return c < 0 ? b : a;
```

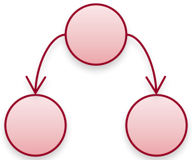
max(0, 0)
max(0, 1)



B

```
int c = b - a;
return c < 0 ? a : b;
```

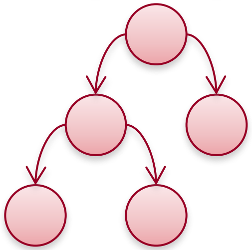
max(0, 0)
max(1, 0)



C

```
if (a == b) {
   return a;
} else {
   return a < b ? b : a;
}
```

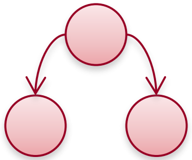
max(0, 0)
max(0, 1)
max(1, 0)



D

```
return a < b ? b : a;
```

max(0, 0)
max(0, 1)



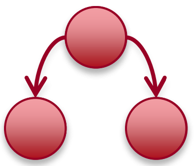
Simple Example

Perform each test set on each implementation (e.g. variant A)

A

```
int c = a - b;
return c < 0 ? b : a;
```

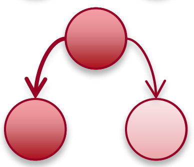
max(0, 0)
max(0, 1)



B

```
int c = b - a;
return c < 0 ? a : b;
```

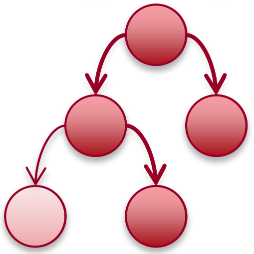
max(0, 0)
max(1, 0)



C

```
if (a == b) {
   return a;
} else {
   return a < b ? b : a;
}
```

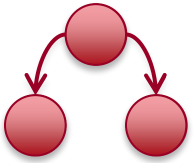
max(0, 0)
max(0, 1)
max(1, 0)



D

```
return a < b ? b : a;
```

max(0, 0)
max(0, 1)



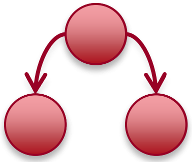
Simple Example

Compute performance for each test set on each implementation (A)

A

```
int c = a - b;
return c < 0 ? b : a;
```

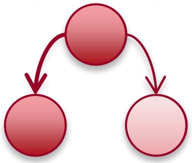
Test set	Performance
max(0, 0)	5
max(0, 1)	5



B

```
int c = b - a;
return c < 0 ? a : b;
```

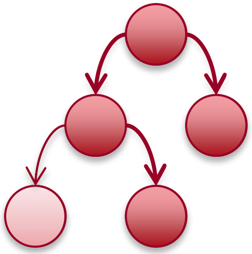
Test set	Performance
max(0, 0)	5
max(1, 0)	5



C

```
if (a == b) {
   return a;
} else {
   return a < b ? b : a;
}
```

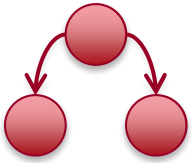
Test set	Performance
max(0, 0)	
max(0, 1)	3
max(1, 0)	5



D

```
return a < b ? b : a;
```

Test set	Performance
max(0, 0)	3
max(0, 1)	3



Method

- Combined coverage
  - favor implementations which cover the test sets of the other implementations
  - punish implementations which only cover one test set
- Weighted Performance
  - weight performance by the coverage
  - treat every test case equally

$$c_i = \prod_{j=1}^{n} (cov_{ij})^{\frac{k}{n}}$$

$$p_i = \left(\sum_{j=1}^{n} \frac{p_{ij}}{(cov_{ij})^k} \right) / n$$

Simple Example - Results

Implementation	Combined coverage	Weighted performance
A	0.81	6.56
B	0.66	8.12
C	0.71	5.77
D	0.81	3.93

- C has a good performance but covers less of the other implementations
- D has the best performance and the best coverage

Conclusion

- If no suitable benchmark or test set exists
- Instead of guessing the correct evaluation setup
- Create workloads based on the implementation of each competitor
- Automatically find test inputs with best and worst case performance
- Treat every test case in the same way

Future Work

- Framework tested for simple algorithms (min/max, sorting)
- Expand on complex data structures (high-dimensional spatio-temporal indices)
- Test case generation by evolutionary algorithms
- Compare to existing benchmarks