Parallel Many-Objective Search for Unit Tests

Verena Bader
University of Passau
Germany

José Campos*
University of Washington
USA

Gordon Fraser
University of Passau
Germany

12th IEEE International Conference on Software Testing (ICST)
Xi'an, China

UNIVERSITY of WASHINGTON

```
// Given
BankAccount bankAccount0 = new BankAccount();
bankAccount0.deposit(10);
```

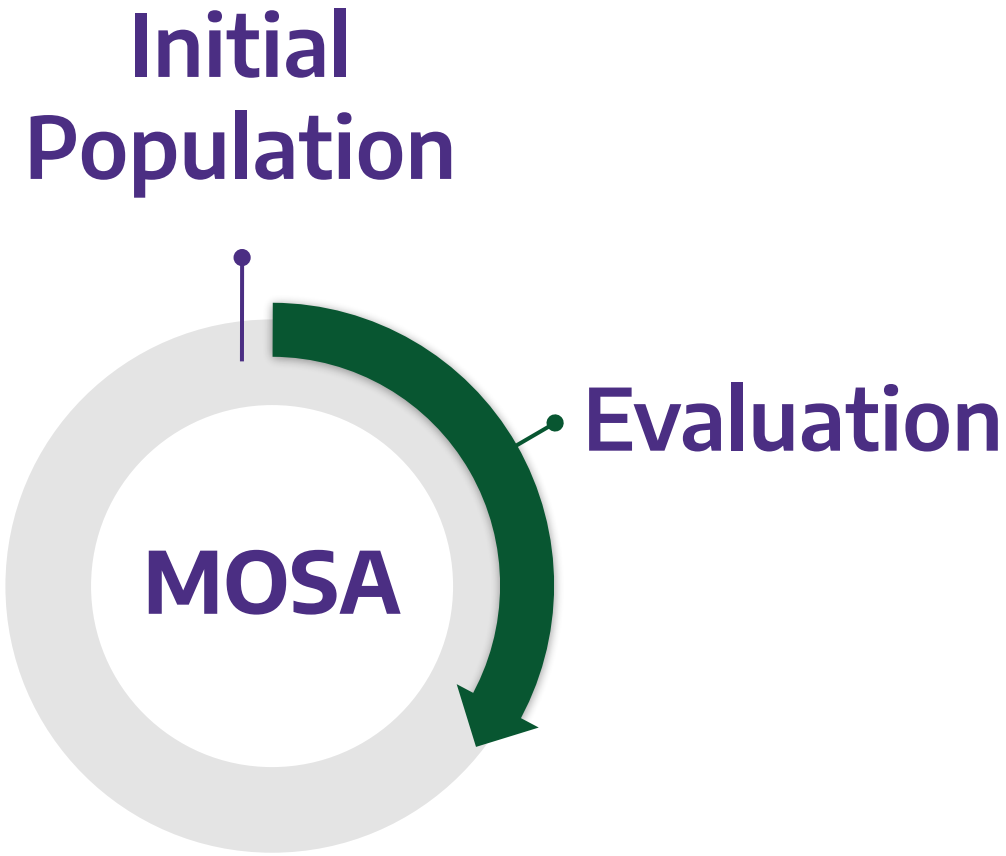
```
// Given
BankAccount bankAccount0 = new BankAccount();
bankAccount0.deposit(10);

// When
boolean boolean0 = bankAccount0.withdraw(25);
```

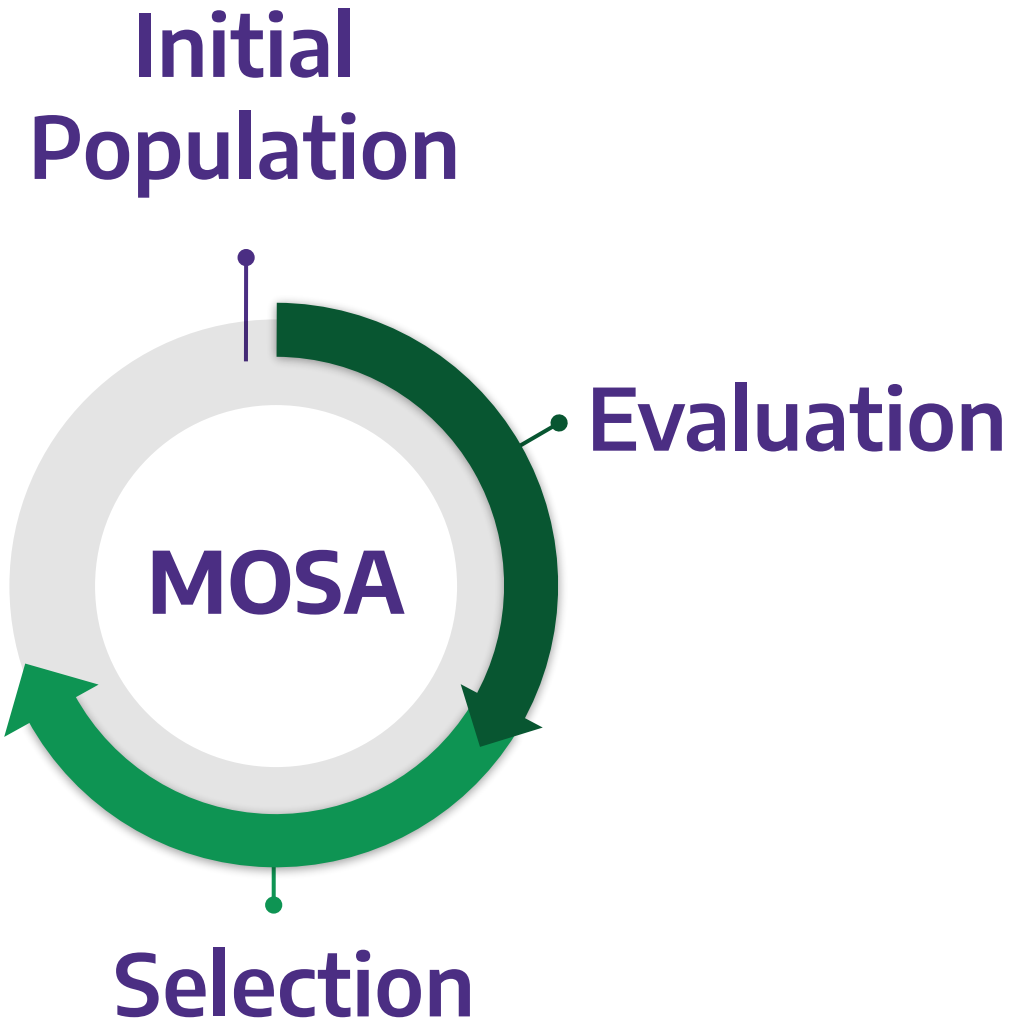
```
// Given
BankAccount bankAccount0 = new BankAccount();
bankAccount0.deposit(10);

// When
boolean boolean0 = bankAccount0.withdraw(25);

// Then
assertThat(boolean0, false);
```

Meta-heuristic search algorithms for unit test generation

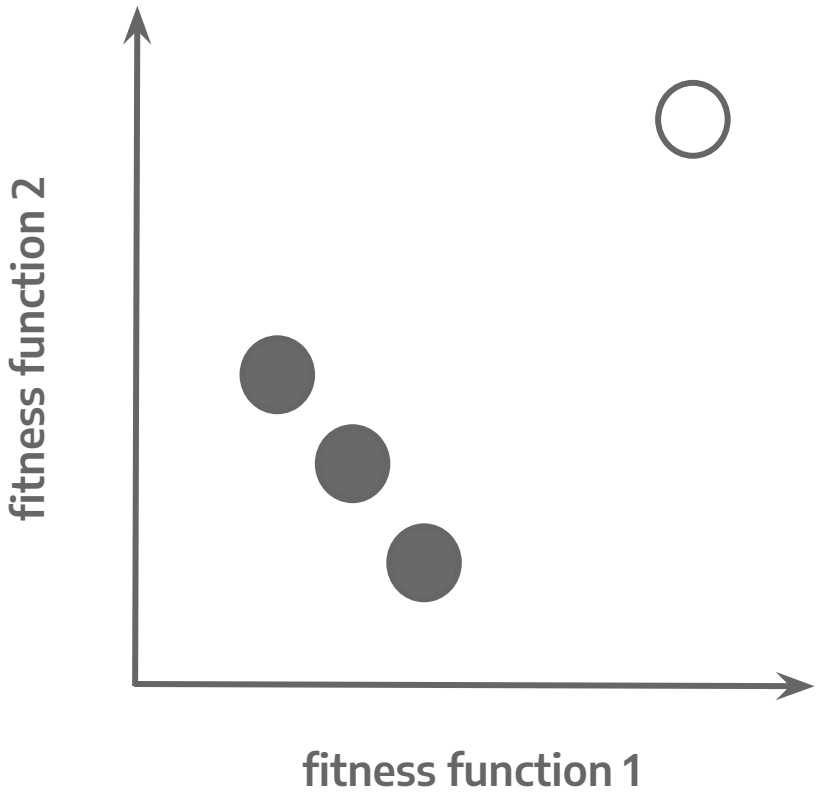
Initial Population

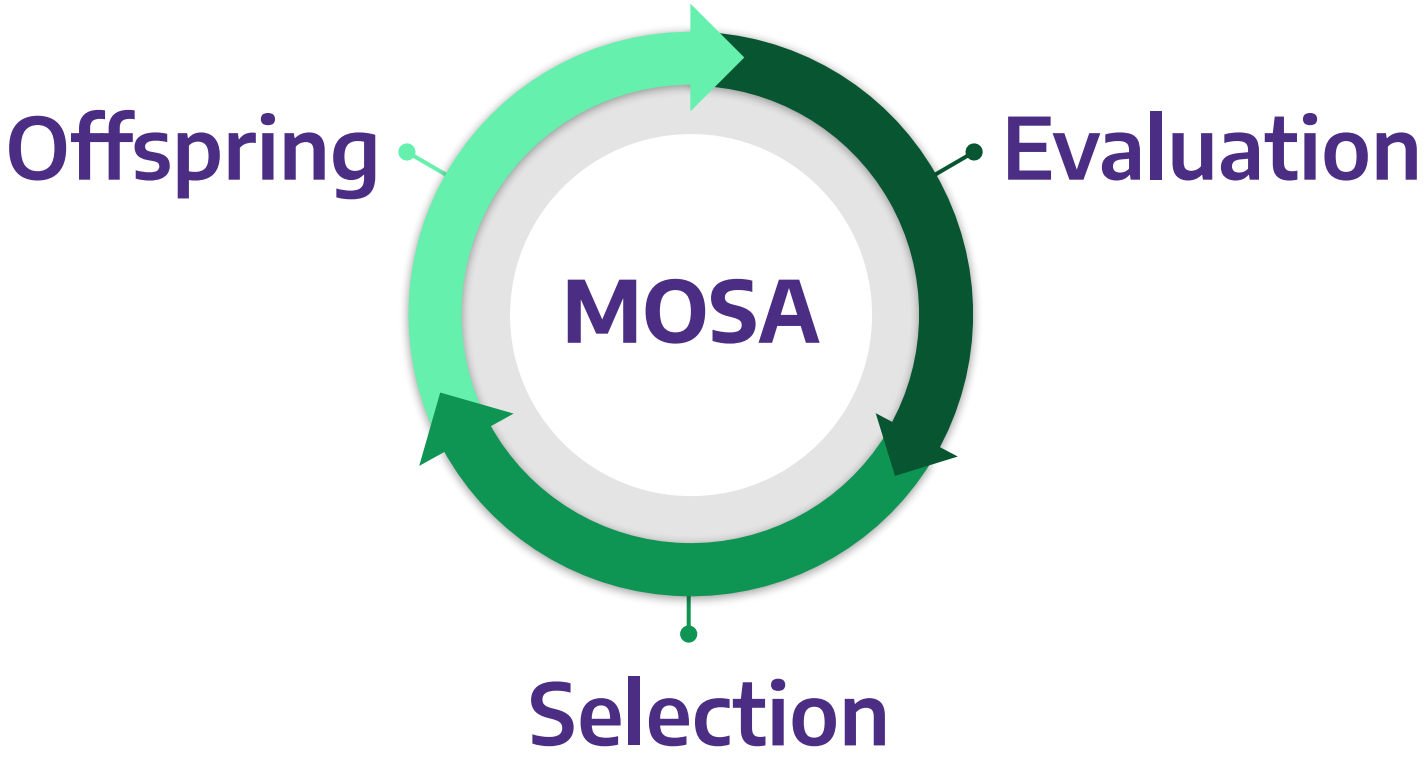
```
@Test
public void test0() {
    int var0 = 10;
    YearMonthDay var1 = new YearMonthDay(var0);
    DateTime var2 = new TimeOfDay();
    DateTime var3 = var1.toDateTime(var2);
    DateTime var4 = var3.minus(var0);
    DateTime var5 = var4.plusSeconds (var0);
}

@Test
public void test1() {
    DateTime var0 = new DateTime("11-09-2017");
    DateTime var1 = new DateTime("25-12-2017");
    int var2 = DateTime.sub(var0, var1);
}
```


Initial
Population
Evaluation
MOSA


Initial
Population
Evaluation
MOSA
Selection

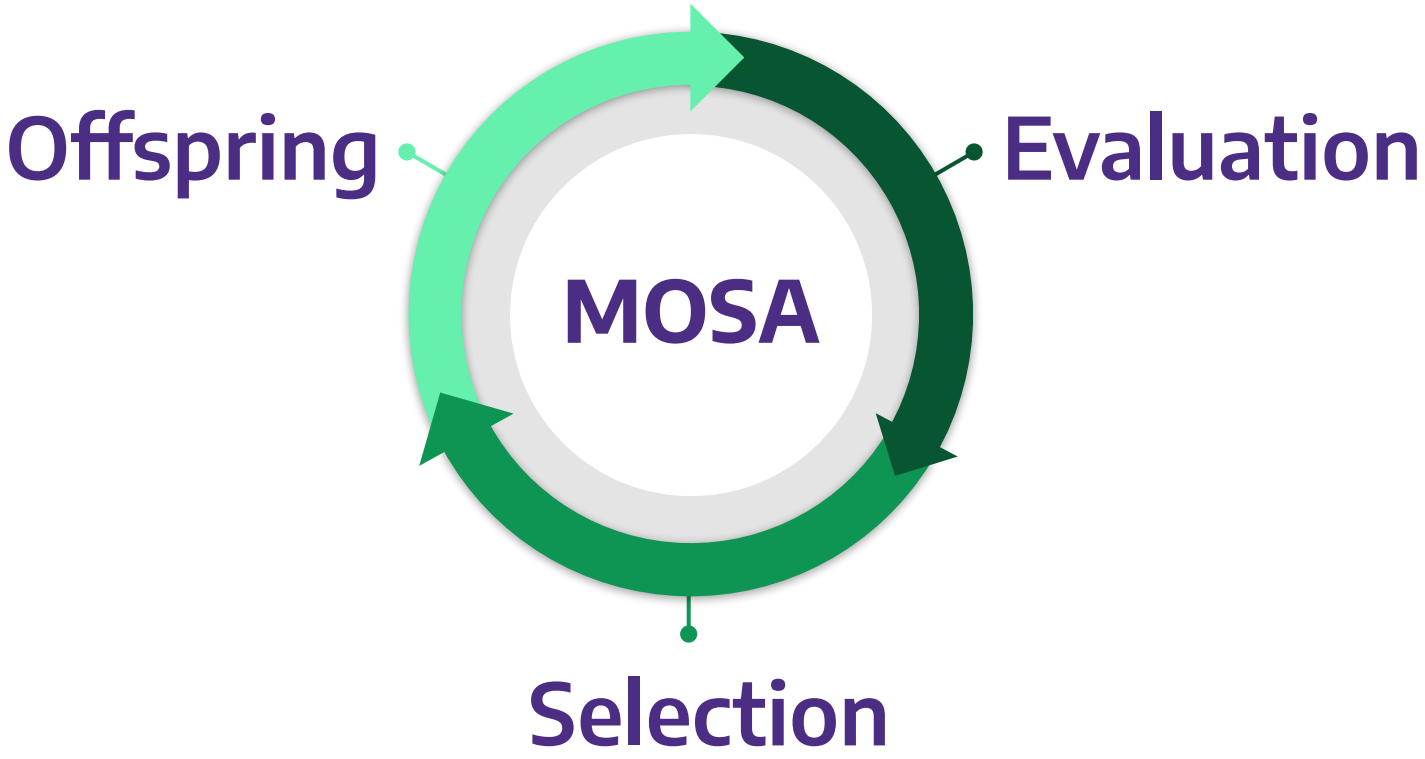
	branch b1	branch b2	Rank
test 0	0.35	0.10	Rank 0
test 1	0.15	0.35	Rank 0
test 2	0.80	0.90	Rank 2
test 3	0.20	0.25	Rank 1


fitness function 2
fitness function 1

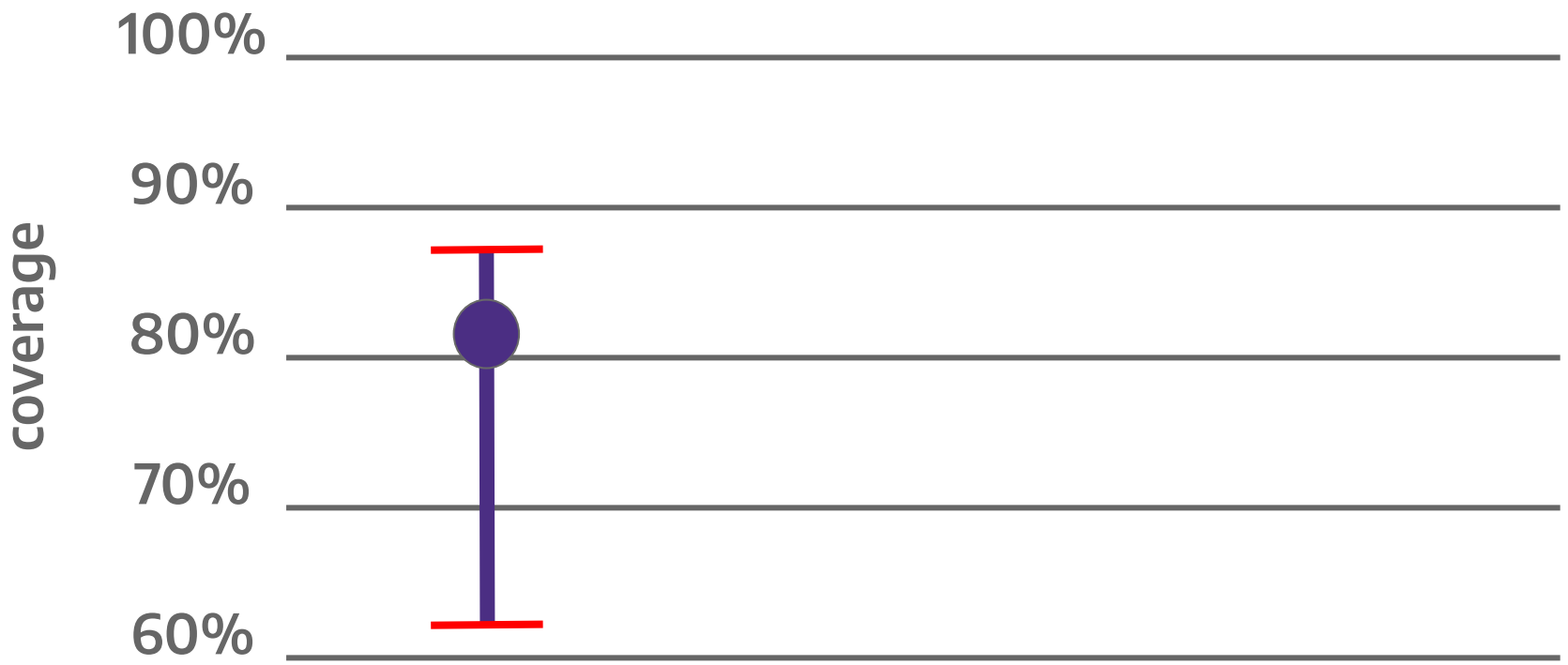

Offspring
Evaluation
MOSA
Selection

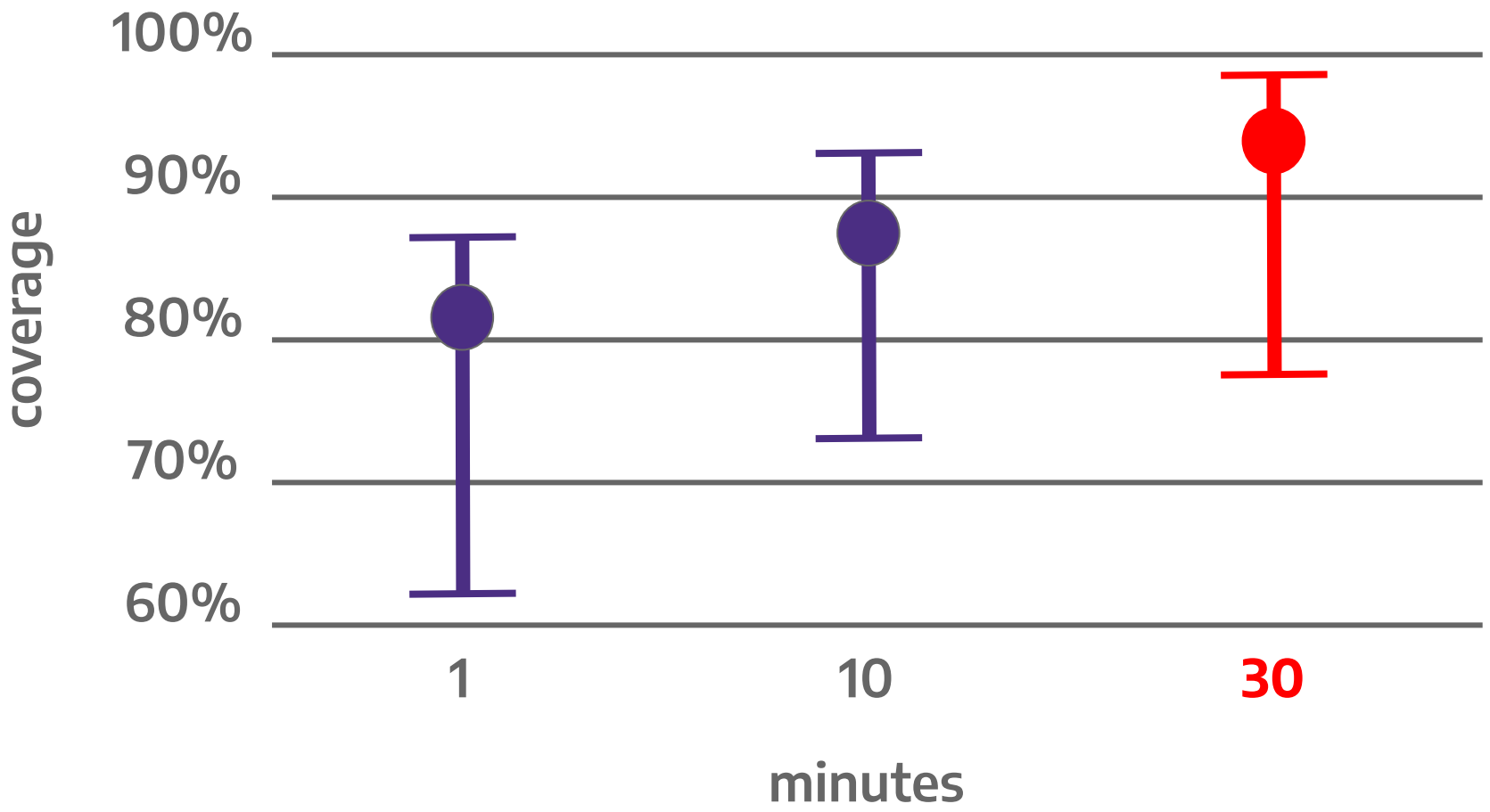
```
@Test
public void test2() {
    int var0 = 10;
    YearMonthDay var1 = new YearMonthDay(var0);
    DateTime var2 = new TimeOfDay();
-   DateTime var3 = var1.toDateTime(var2);
-   DateTime var4 = var3.minus(var0);
-   DateTime var5 = var4.plusSeconds (var0);
+   DateTime var3 = new DateTime("11-09-2017");
+   DateTime var4 = new DateTime("25-12-2017");
+   int var5 = DateTime.sub(var3, var4);
}
```

```
@Test
public void test0() {
    int var0 = 10;
    YearMonthDay var1 = new YearMonthDay(var0);
    DateTime var2 = new TimeOfDay();
    DateTime var3 = var1.toDateTime(var2);
-   DateTime var4 = var3.minus(var0);
-   DateTime var5 = var4.plusSeconds (var0);
+   DateTime var4 = new DateTime("11-09-2017");
+   DateTime var5 = new DateTime("25-12-2017");
}

+ @Test
+ public void test3() {
+   DateTime var0 = new TimeOfDay();
+   DateTime var1 = new TimeOfDay();
+   int var2 = DateTime.add(var0, var1);
+ }
```

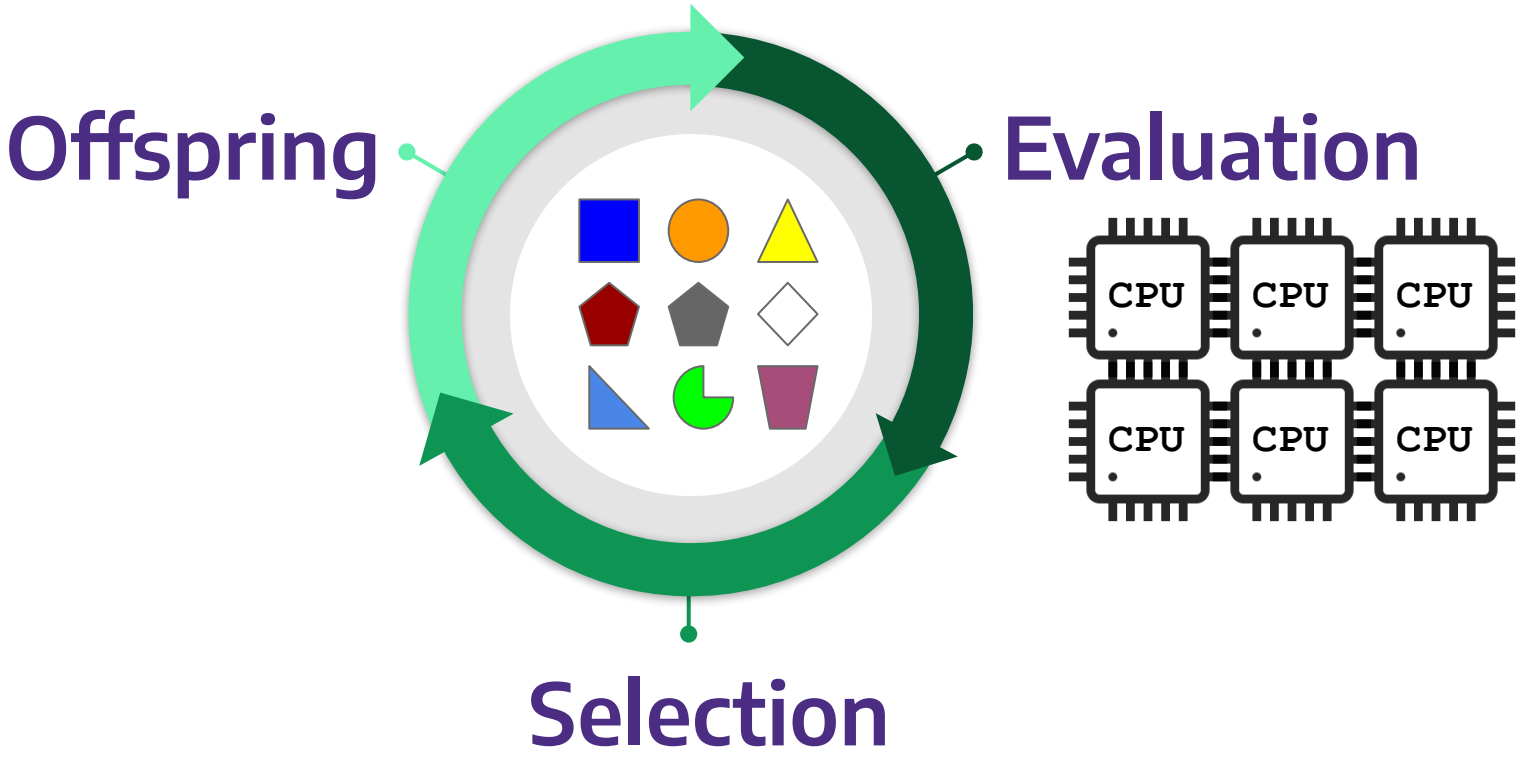

Offspring
Evaluation
MOSA
Selection

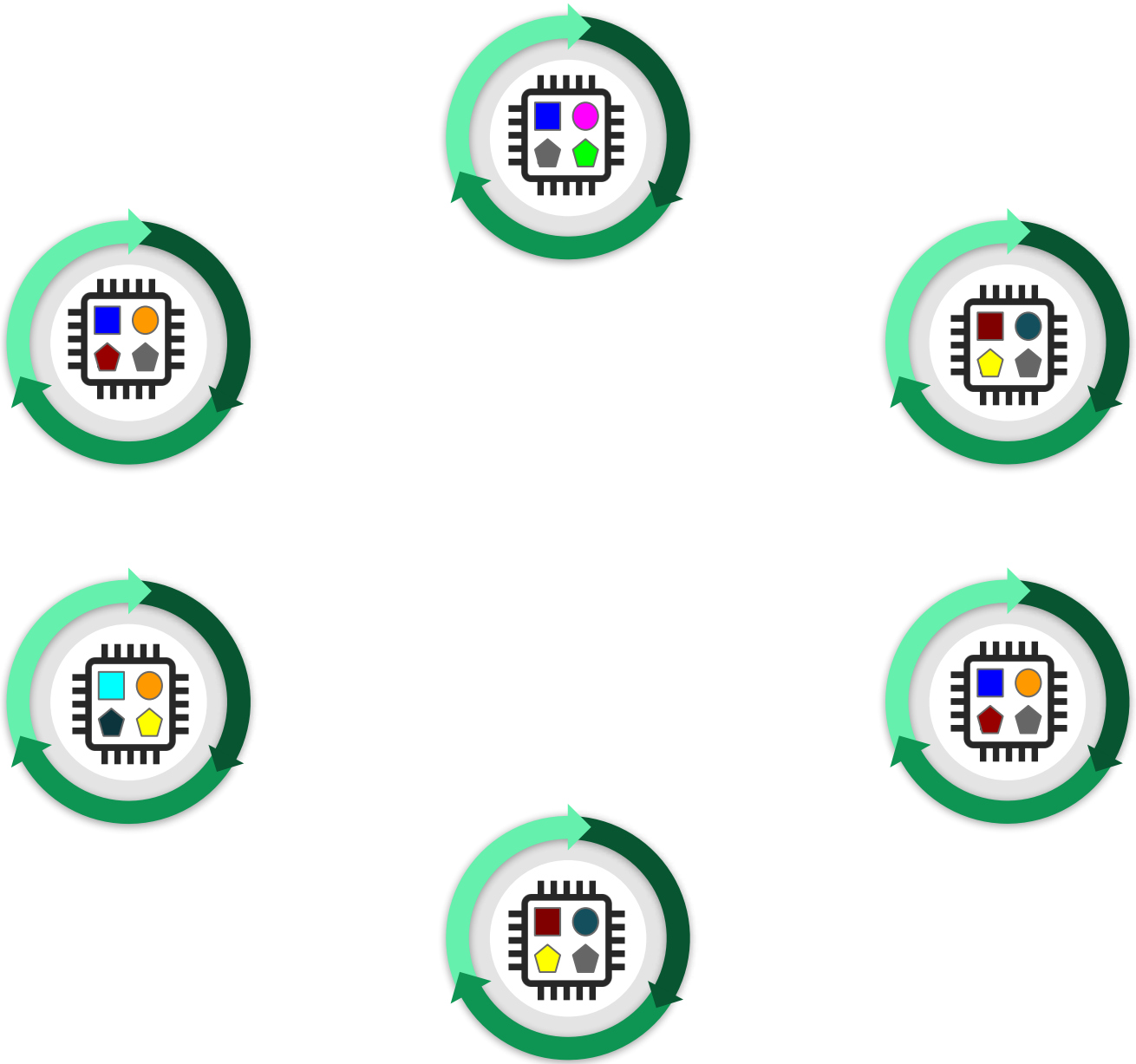

100%
90%
80%
70%
60%
coverage

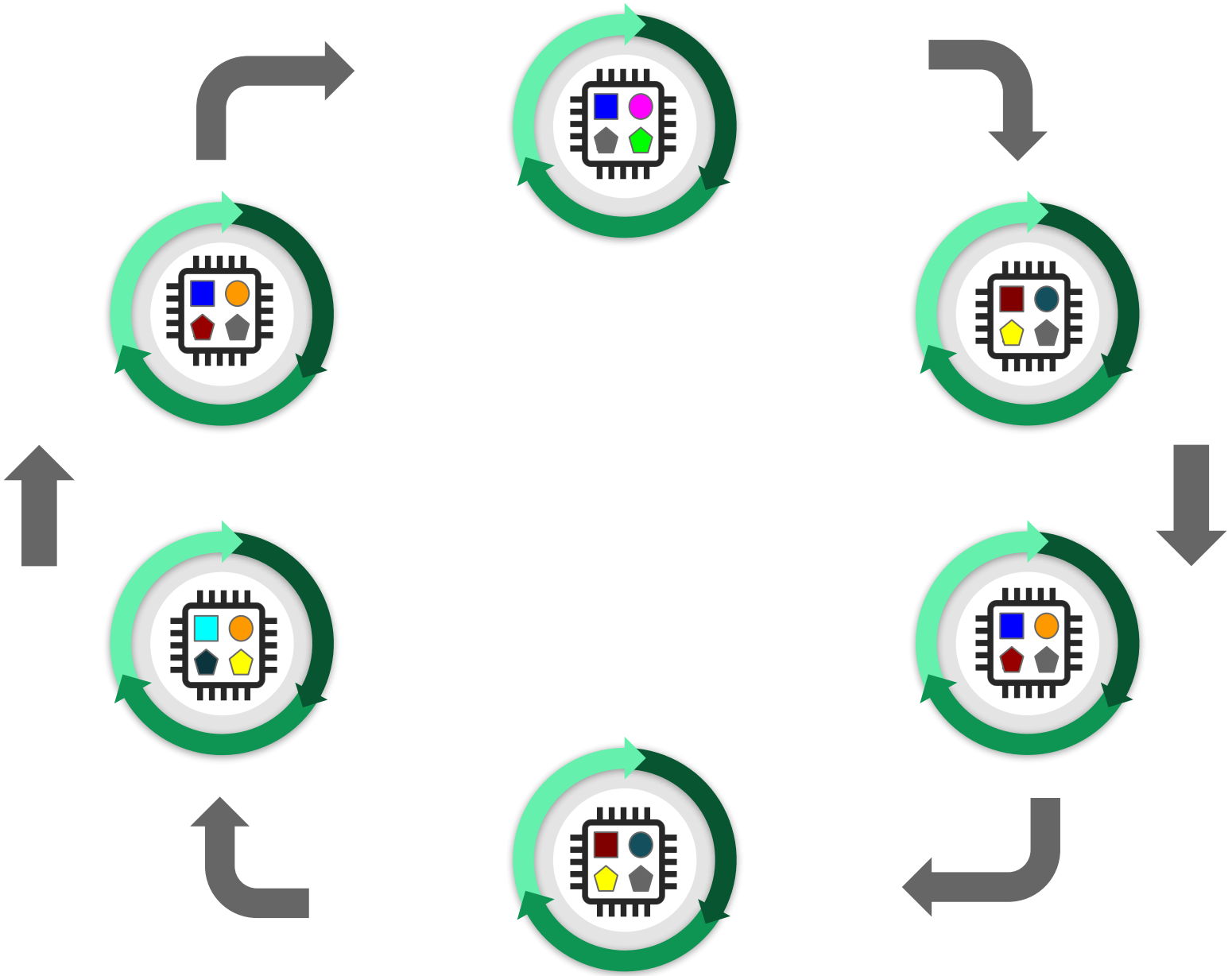

100%
90%
80%
70%
60%
coverage


100%
90%
80%
70%
60%
coverage
1
10
30
minutes

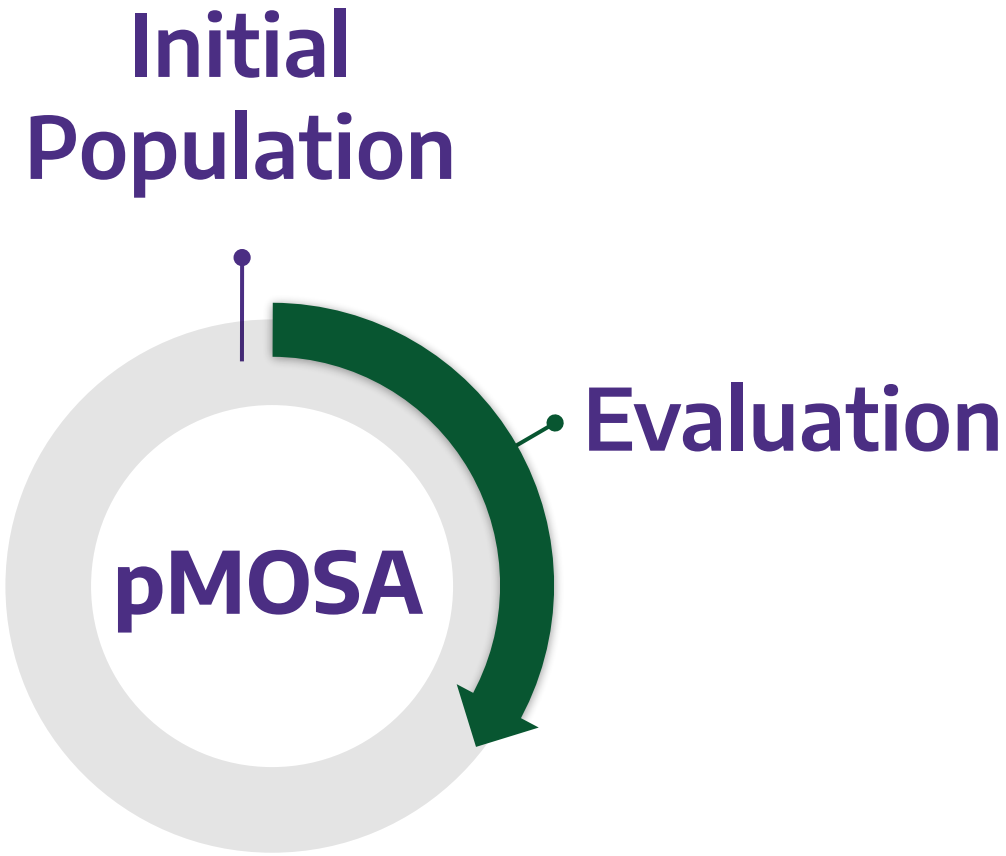
Parallel Genetic Algorithm

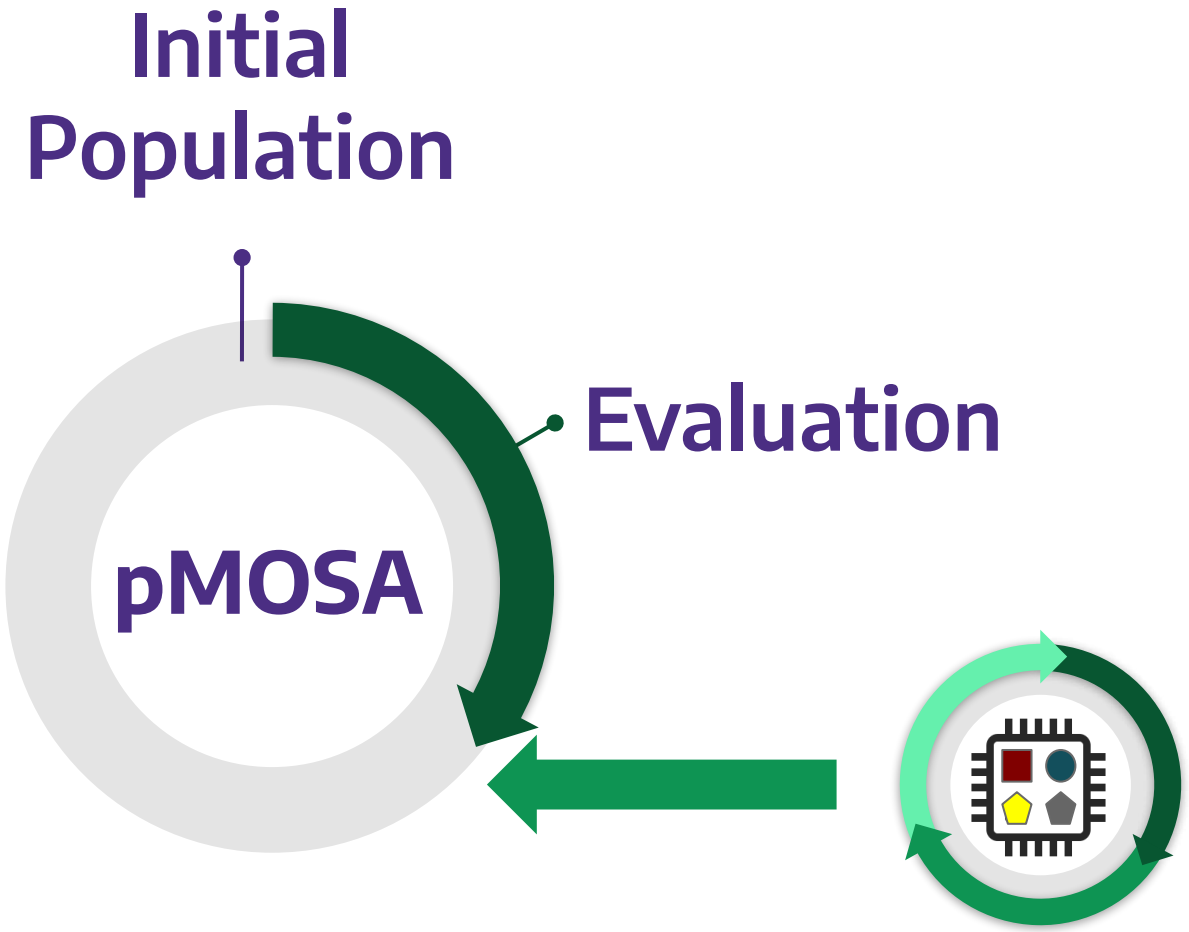

Offspring
Evaluation
CPU
CPU
CPU
CPU
CPU
CPU
Selection

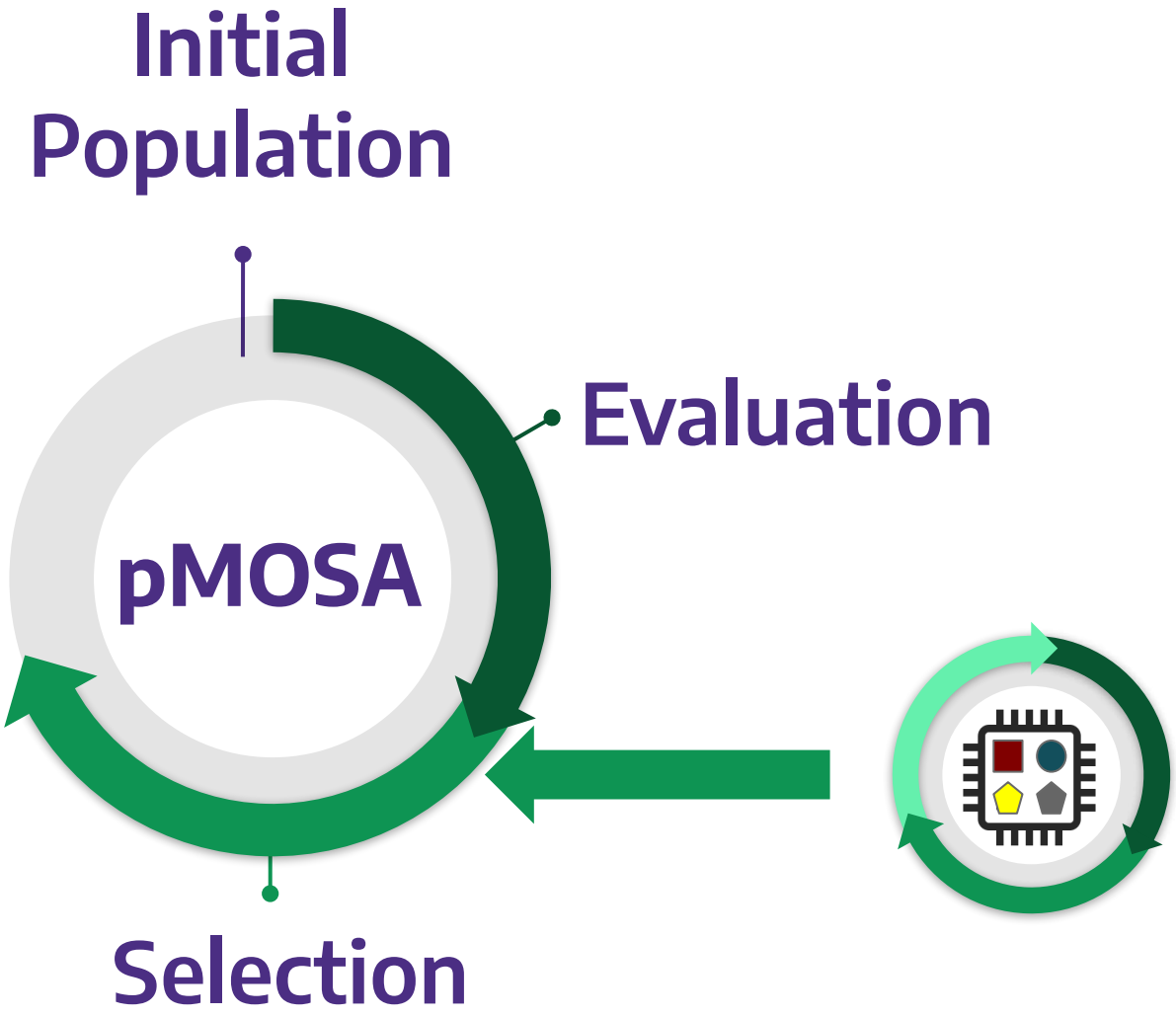


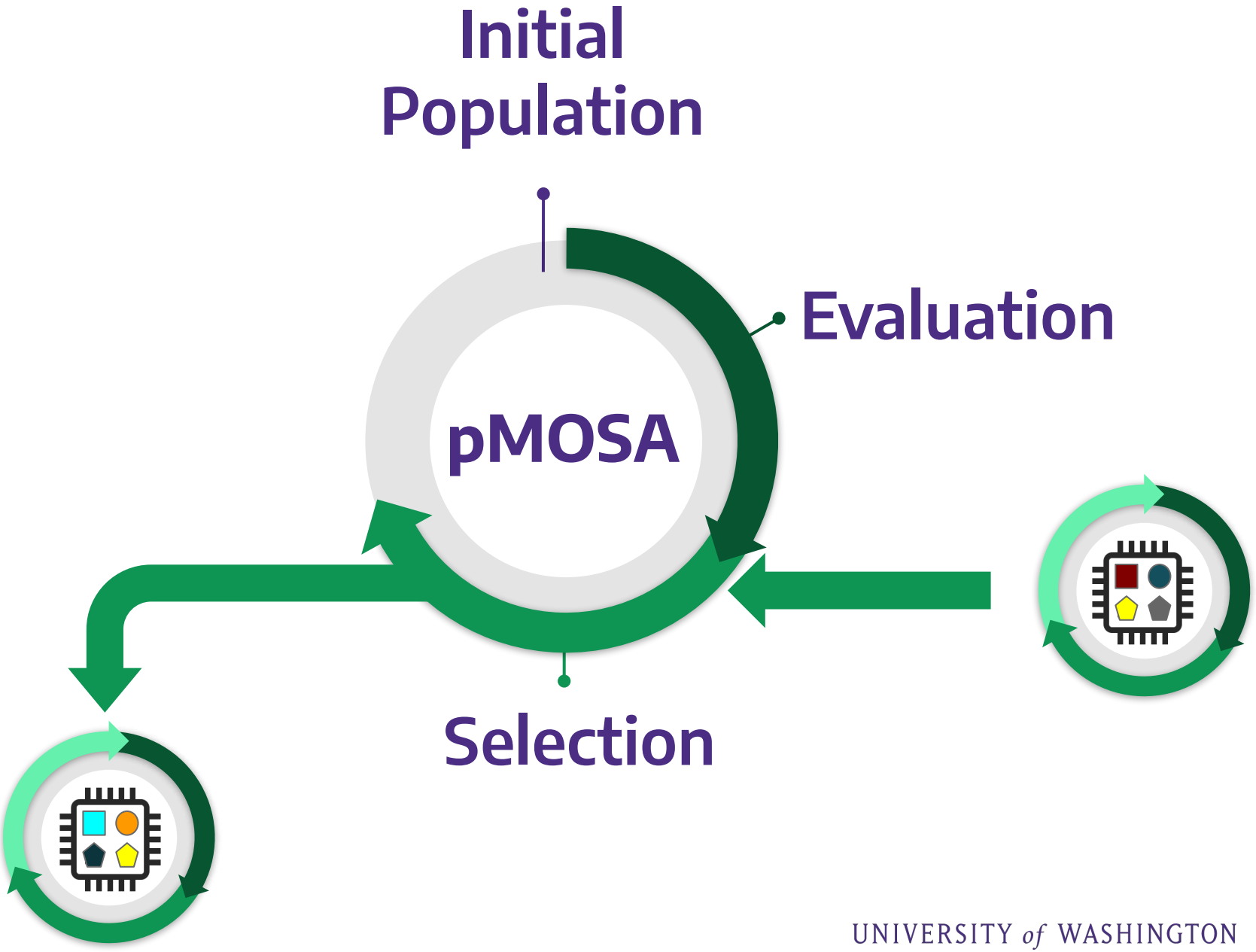


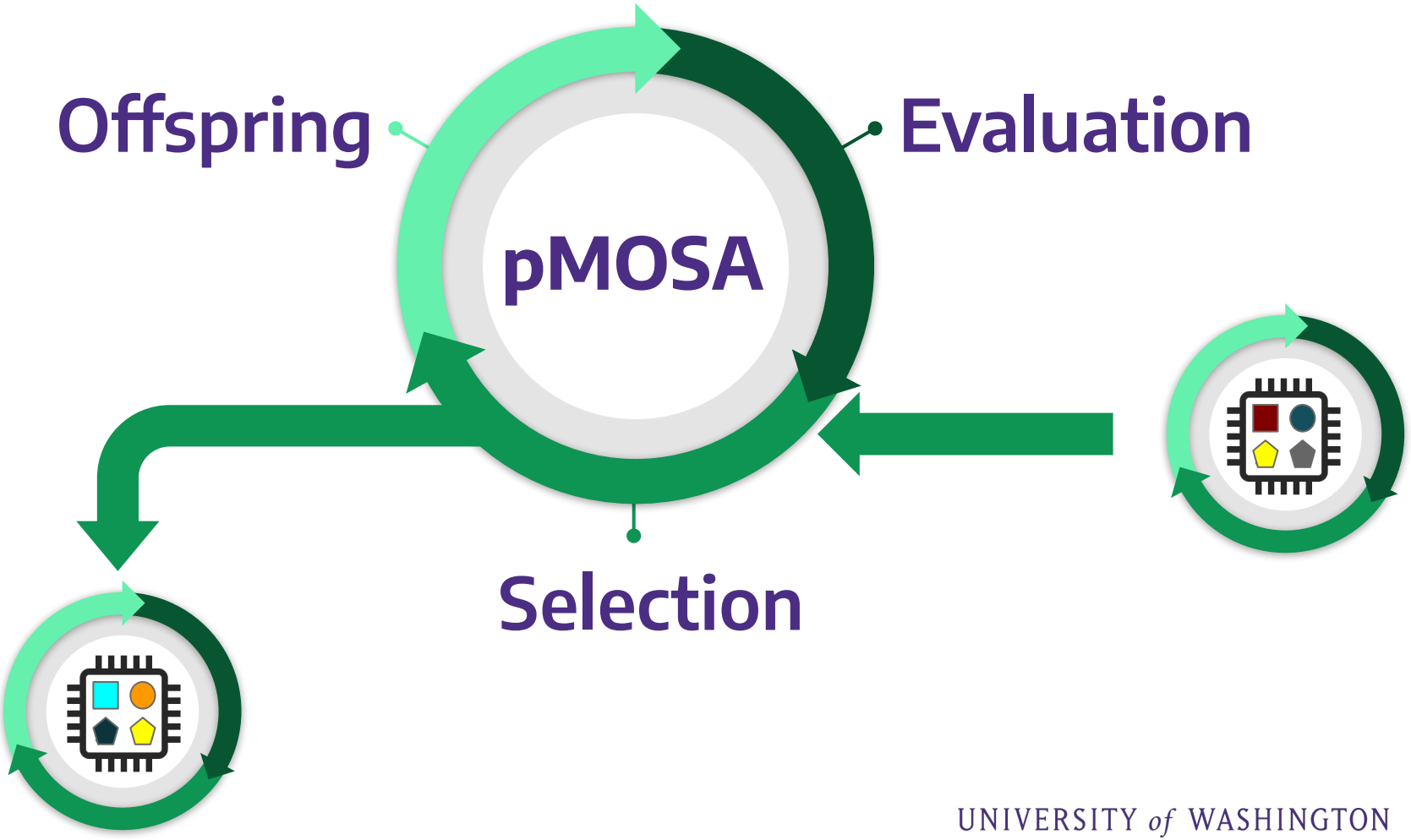
Parallel Many-Objective Sorting Algorithm


Initial
Population
Evaluation
pMOSA


Initial
Population
Evaluation
pMOSA


Initial
Population
Evaluation
pMOSA
Selection


Initial
Population
Evaluation
pMOSA
Selection

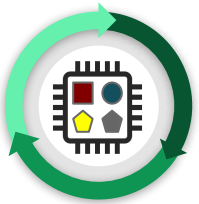

Offspring
Evaluation
pMOSA
Selection

Empirical Evaluation

Tuning

34 Java classes

216 configurations



pMOSA

60 seconds

30 repetitions

Parameter tuning

	Tuned values
# Clients	2, 4, 8
Migration frequency	1, 5, 10, 25
Migration rate	1, 5, 10, 15, 20, 25
Migration selection function	random best rank

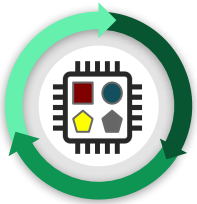
Parameter tuning

	Tuned values	Best values		
# Clients	2, 4, 8	2	4	8
Migration frequency	1, 5, 10, 25	25	5	10
Migration rate	1, 5, 10, 15, 20, 25	20	1	1
Migration selection function	random best rank	random	rank	random

Tuning

34 Java classes

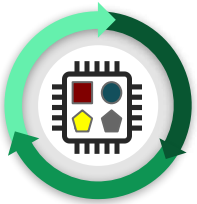
216 configurations



pMOSA

Sequential vs Parallel

312 Java classes



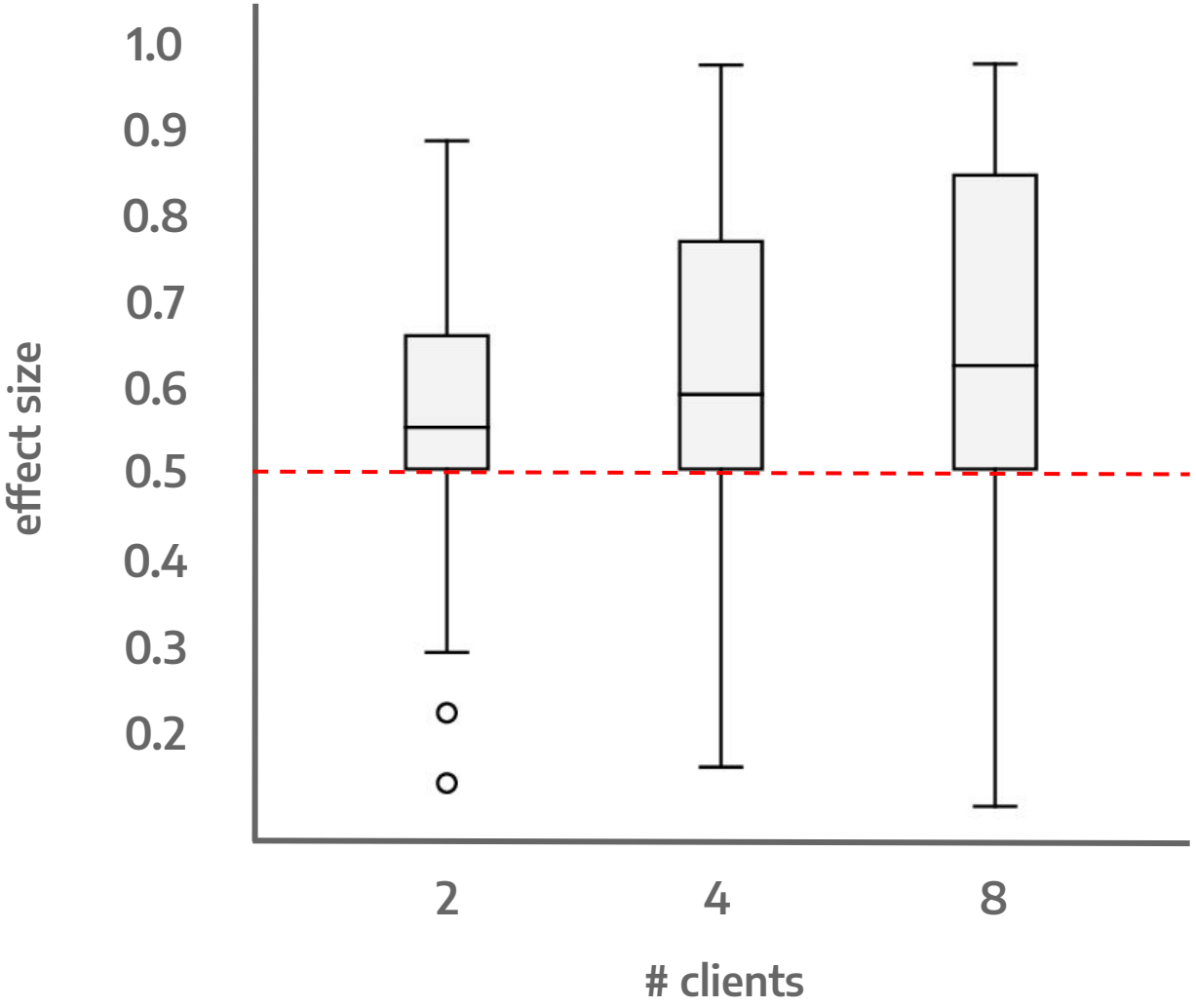
2 algorithms
MOSA, pMOSA

60 seconds

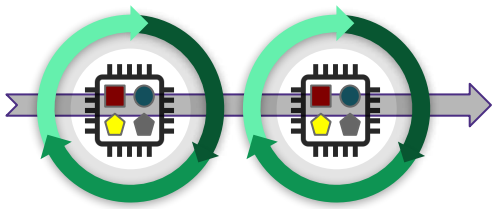
30 repetitions

RQ1 - Does parallel MOSA improve over sequential MOSA?

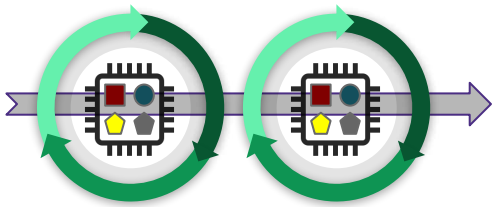
Improvements over sequential



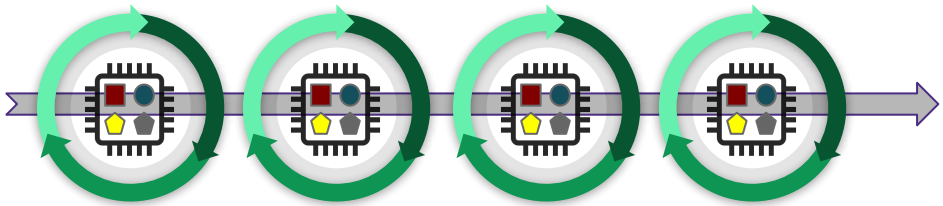
Improvements over sequential

2  Significantly higher coverage on **80** classes

Improvements over sequential



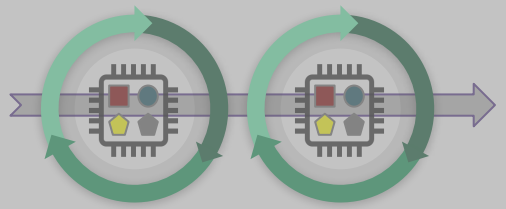
Significantly higher coverage on **80** classes



Significantly higher coverage on **133** classes

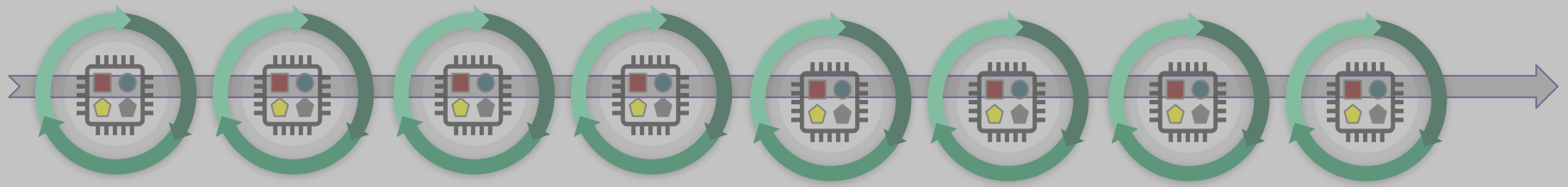
Improvements over sequential

Improvements over sequential

2 

Significantly higher coverage on **80** classes

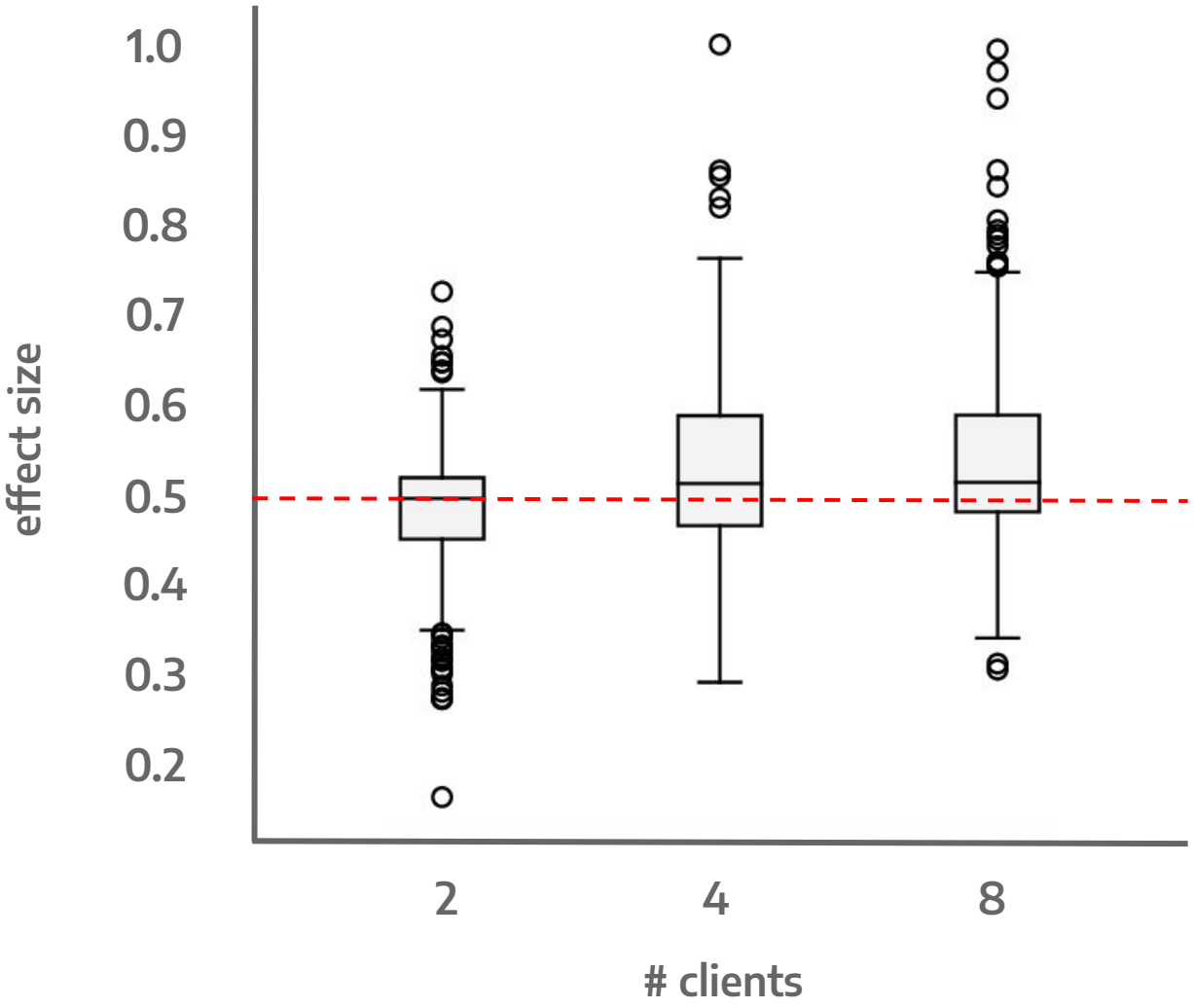
Coverage increases from 79% to 84% using parallelisation with 8 clients

8 

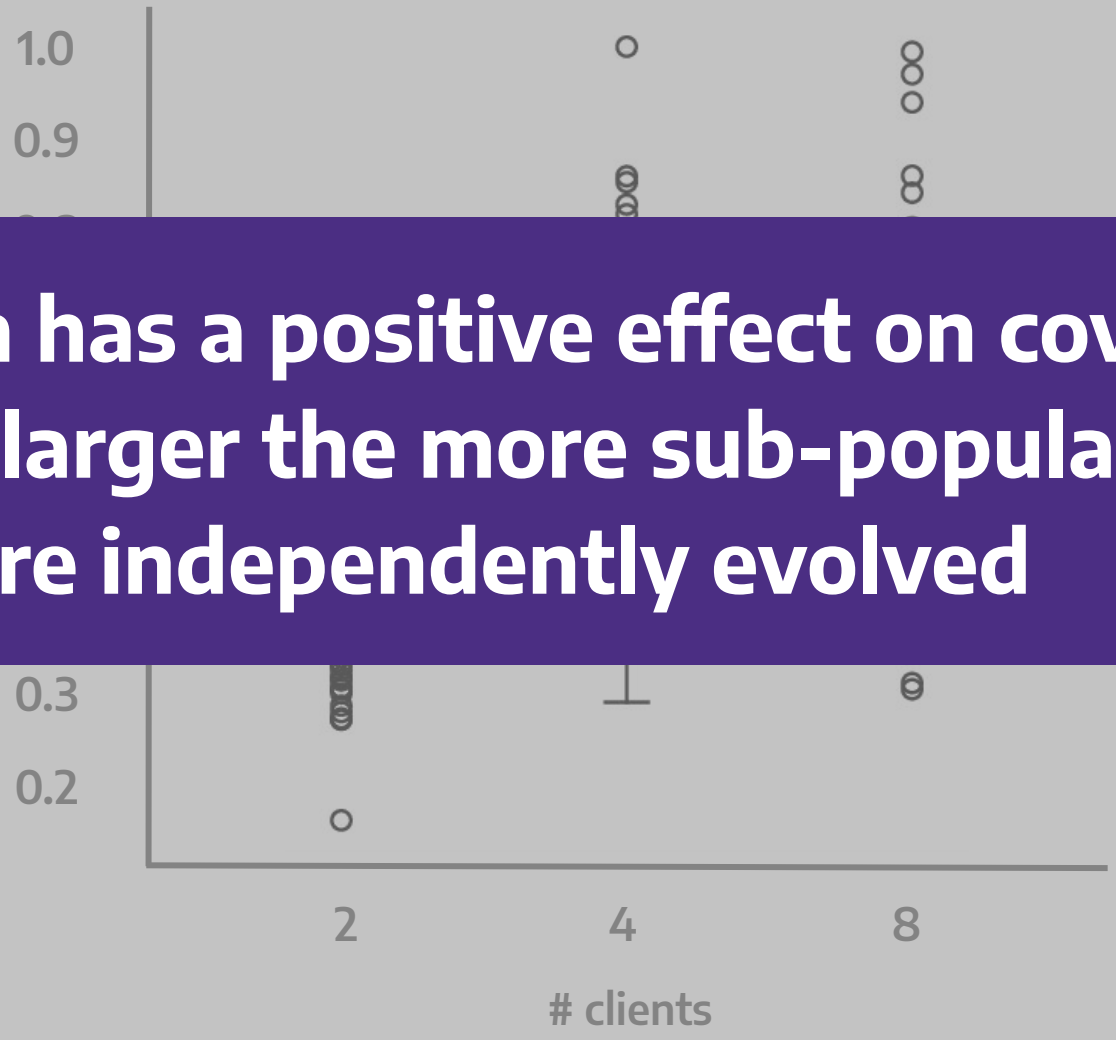
Significantly higher coverage on **142** classes

RQ2 - Does migration contribute to better performance?

Migration vs non-migration



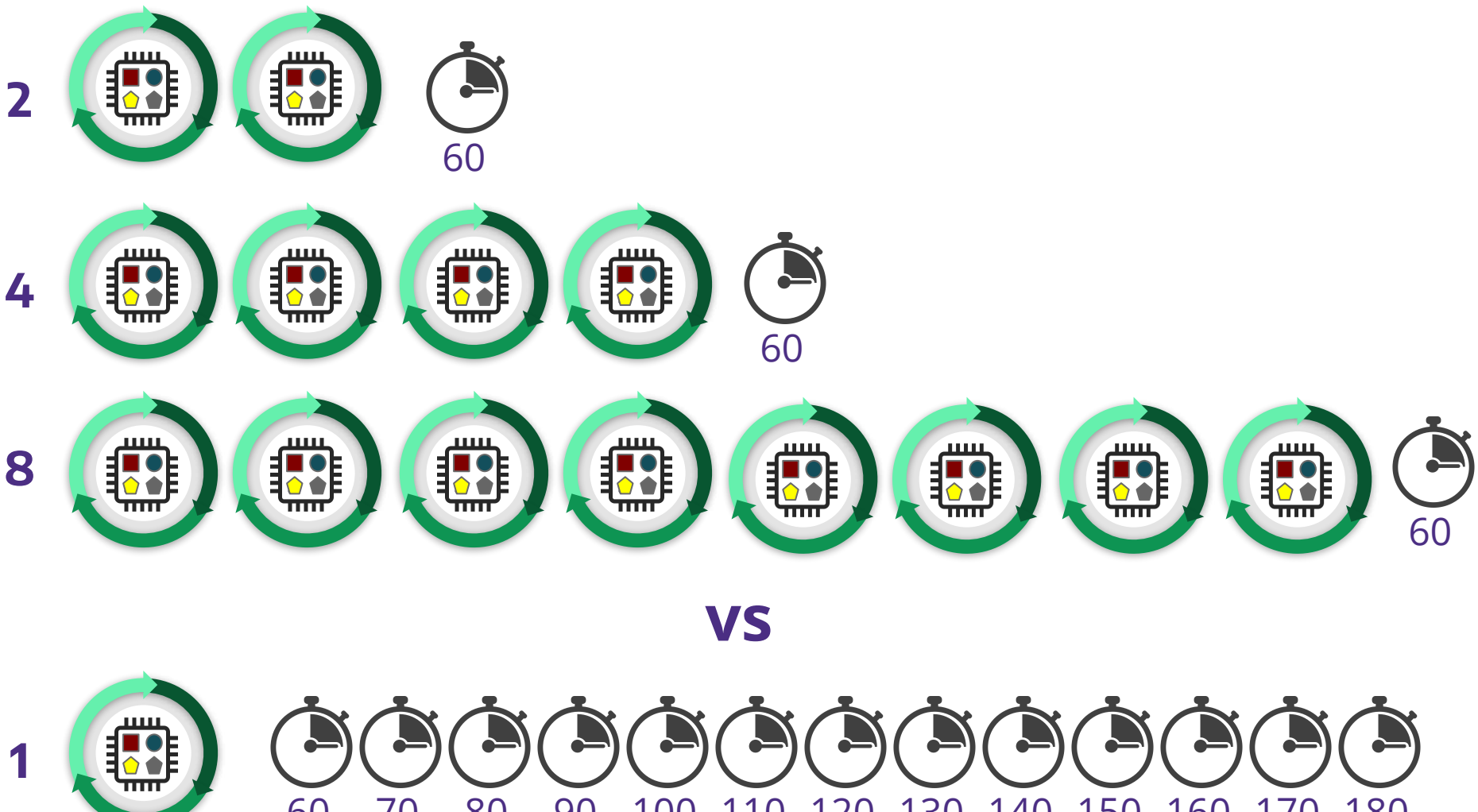
Migration vs non-migration



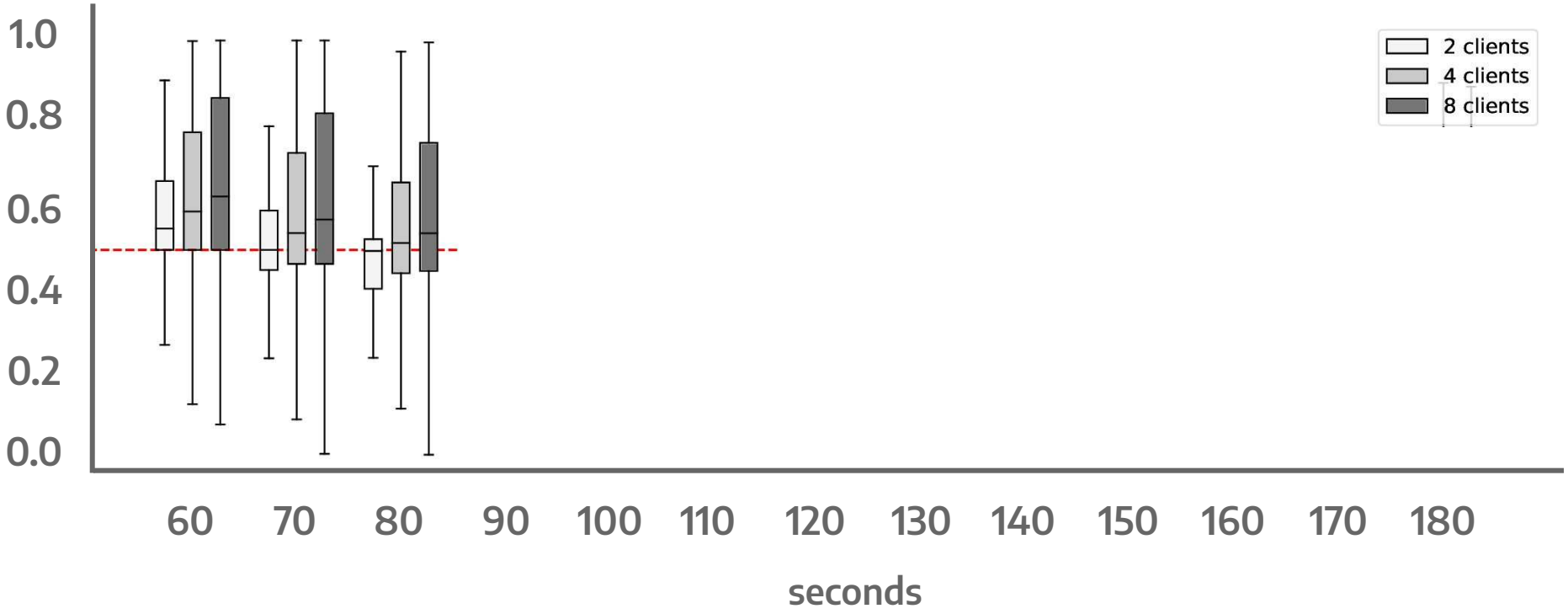
Migration has a positive effect on coverage and it is larger the more sub-populations are independently evolved

RQ3 – Does parallelisation reduce the overall runtime to achieve coverage?

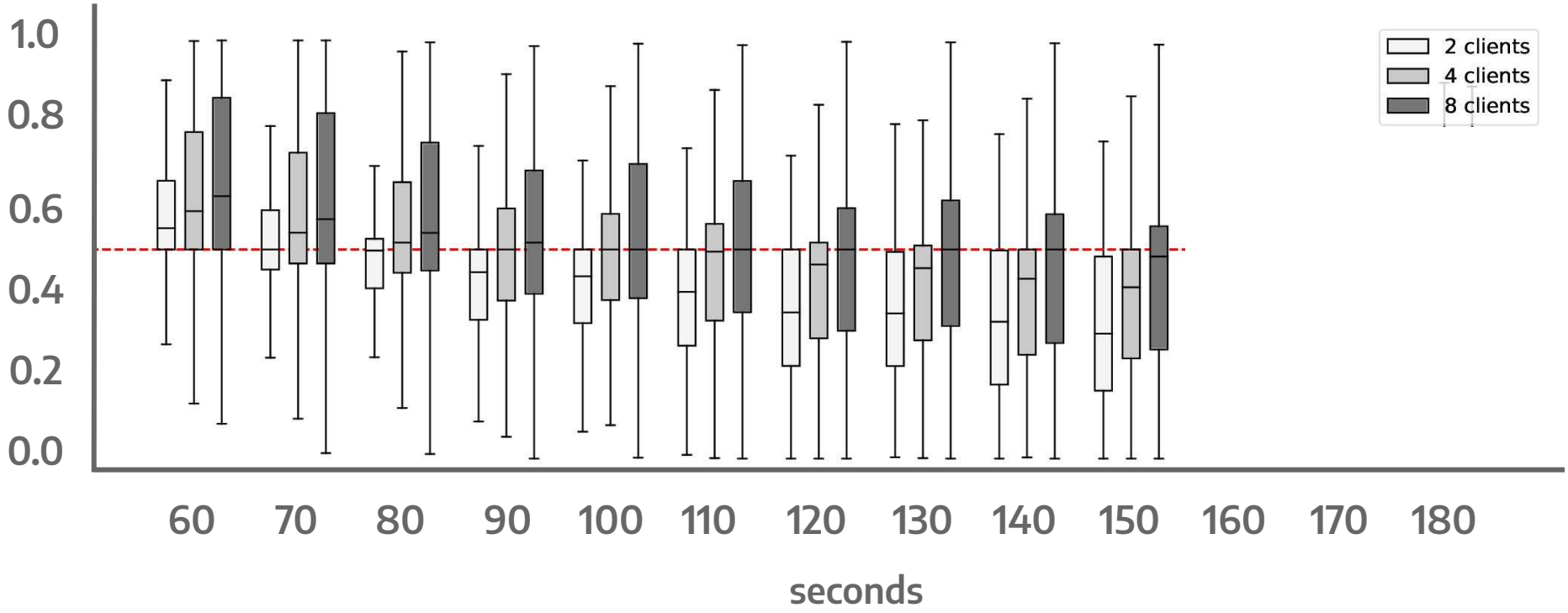
Runtime reduction



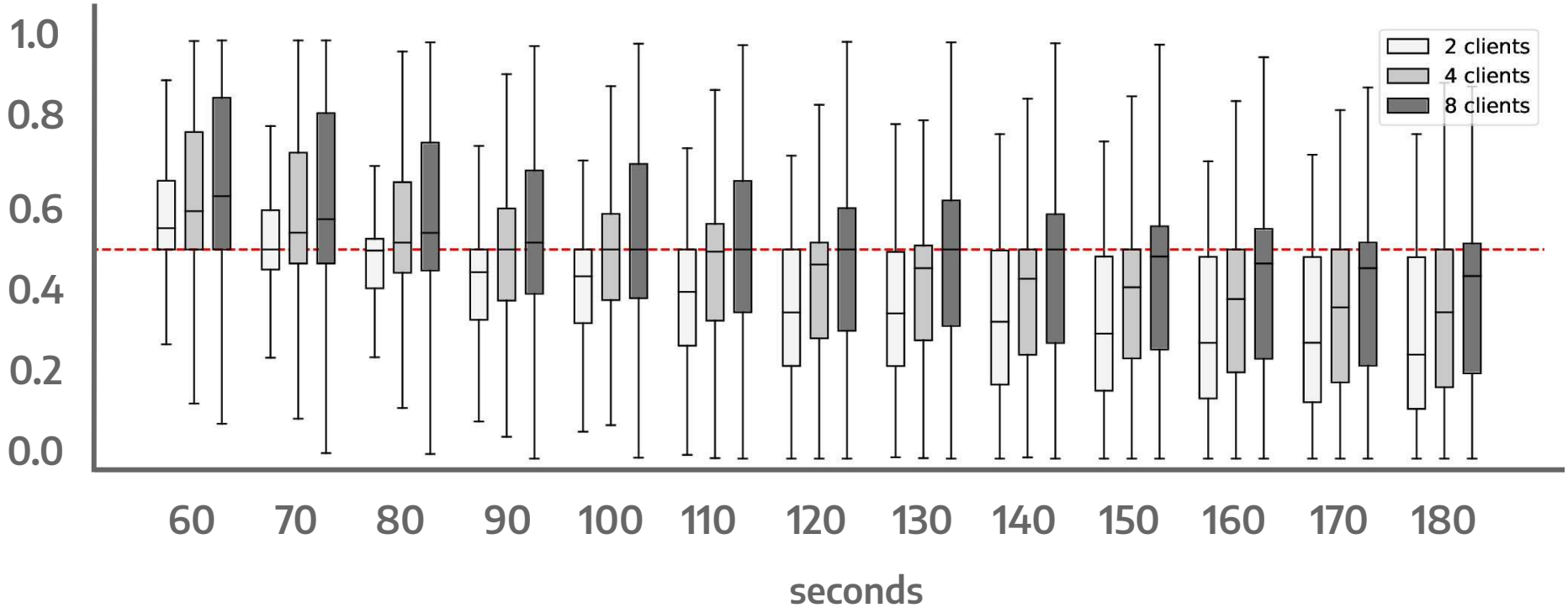
Runtime reduction



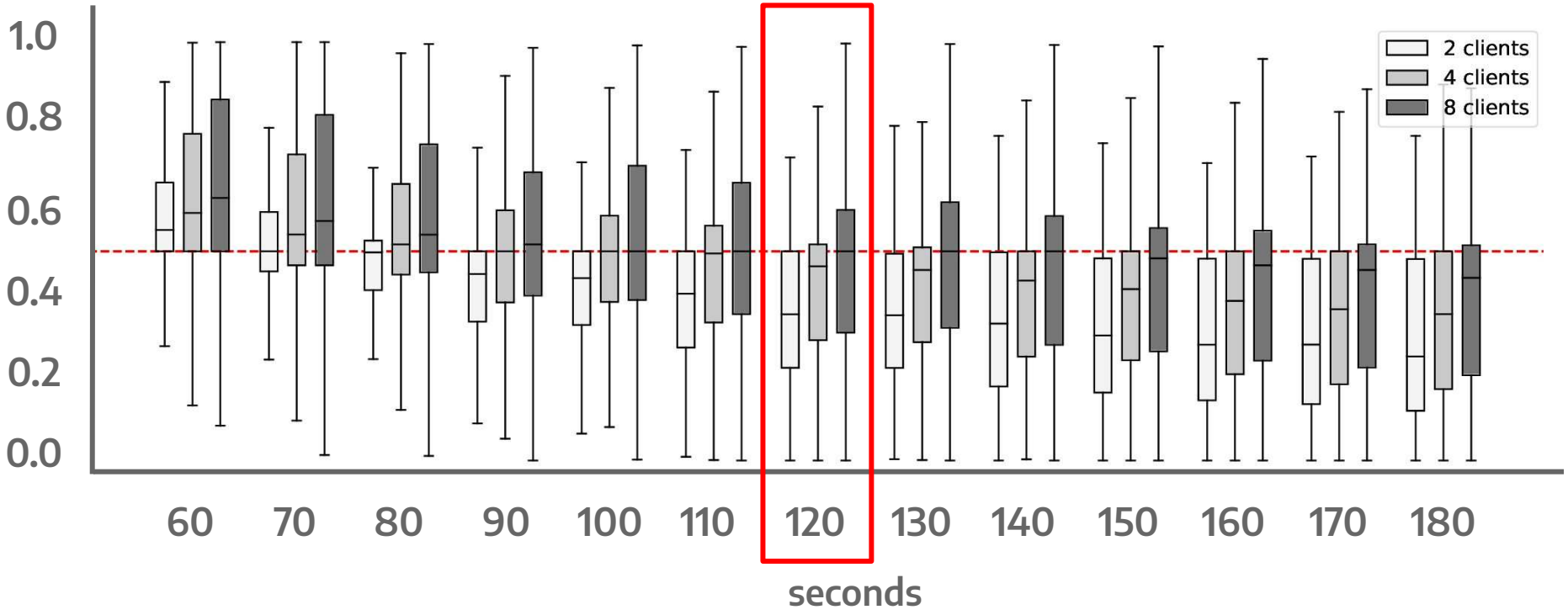
Runtime reduction



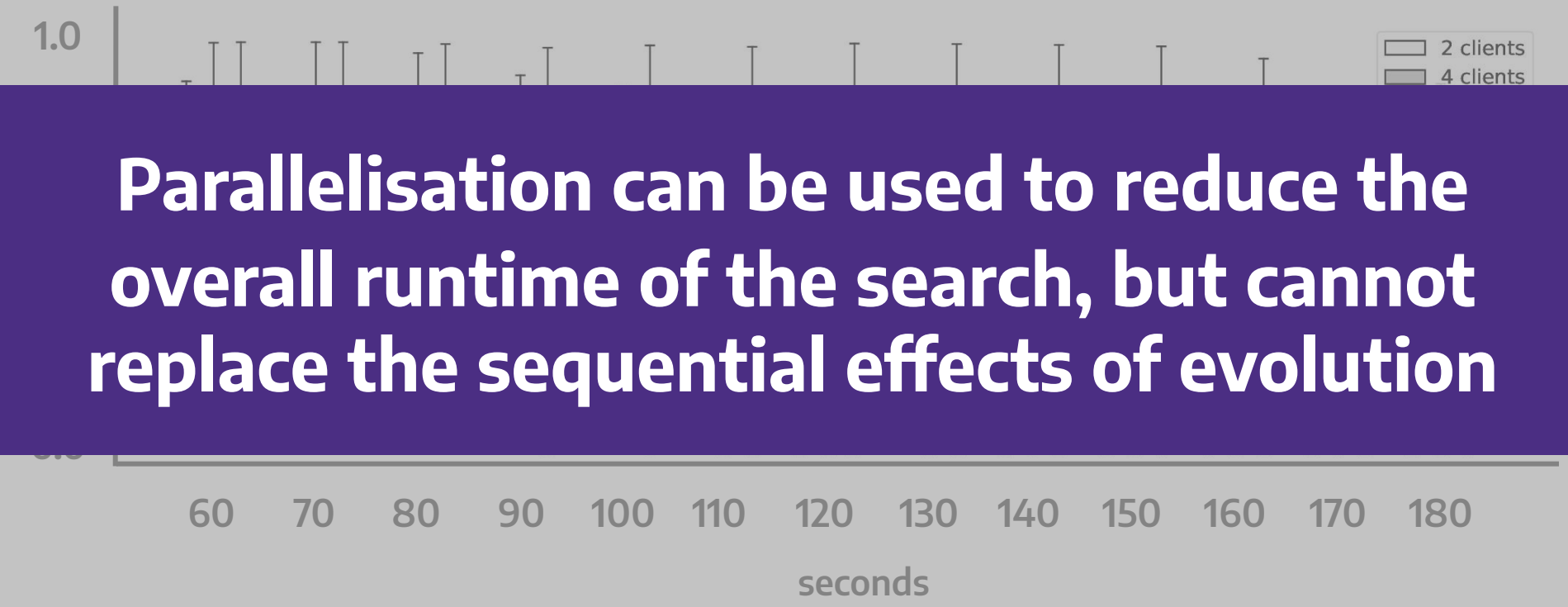
Runtime reduction



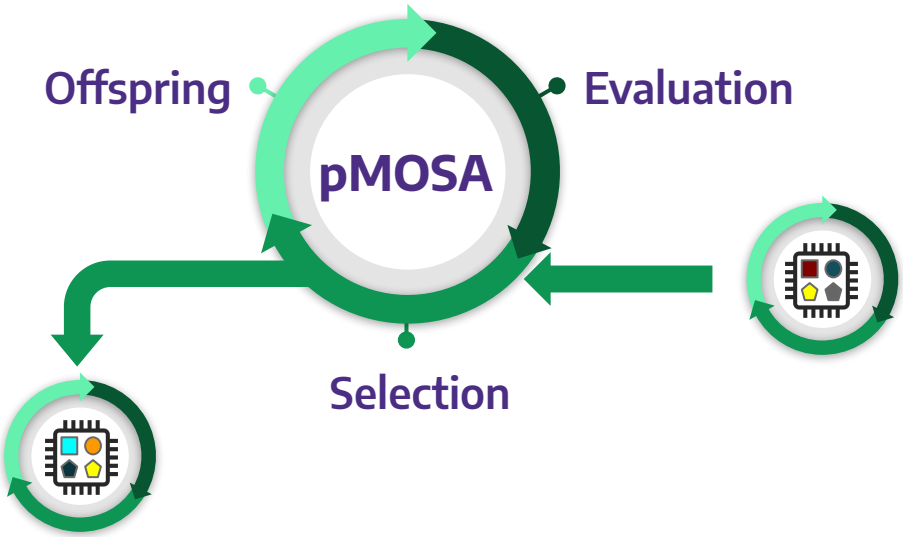
Runtime reduction

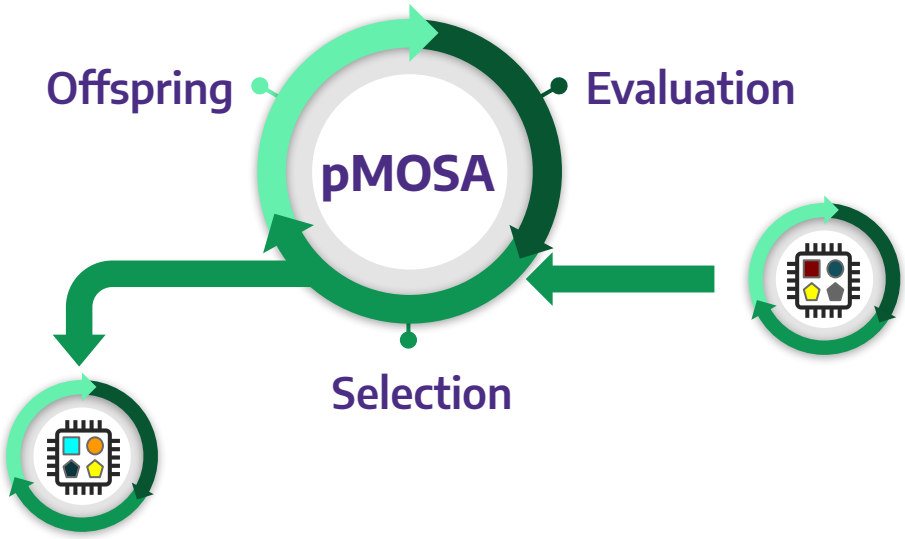


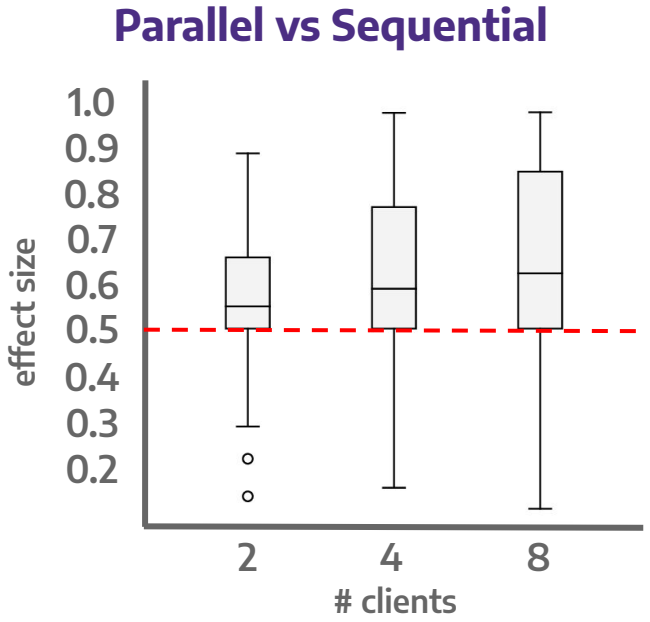
Runtime reduction

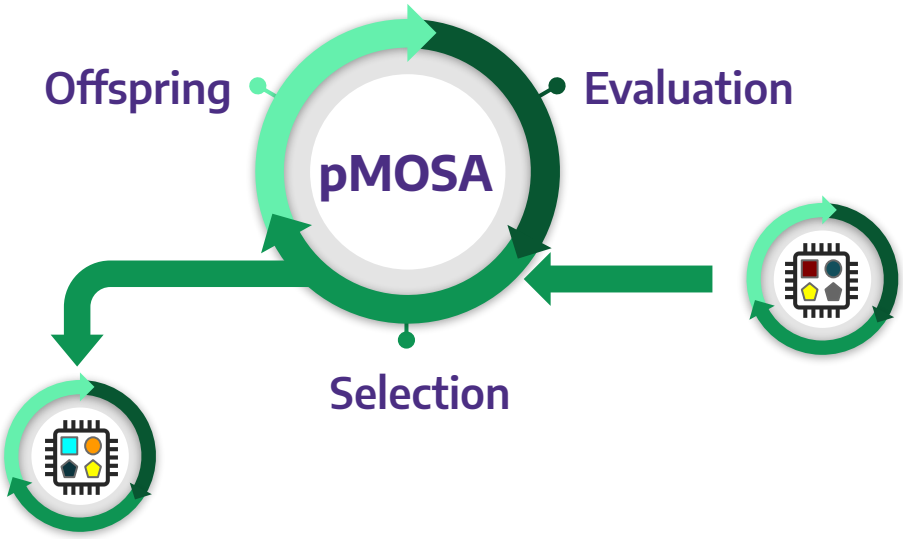


Parallelisation can be used to reduce the overall runtime of the search, but cannot replace the sequential effects of evolution

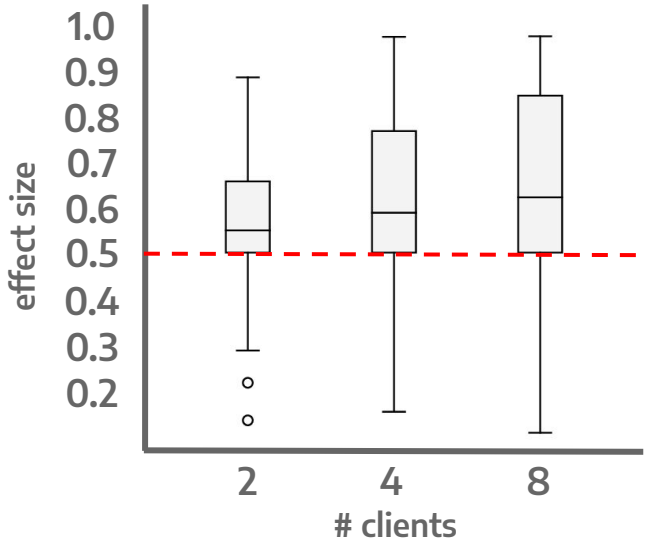

Offspring
Evaluation
pMOSA
Selection


Offspring
Evaluation
pMOSA
Selection

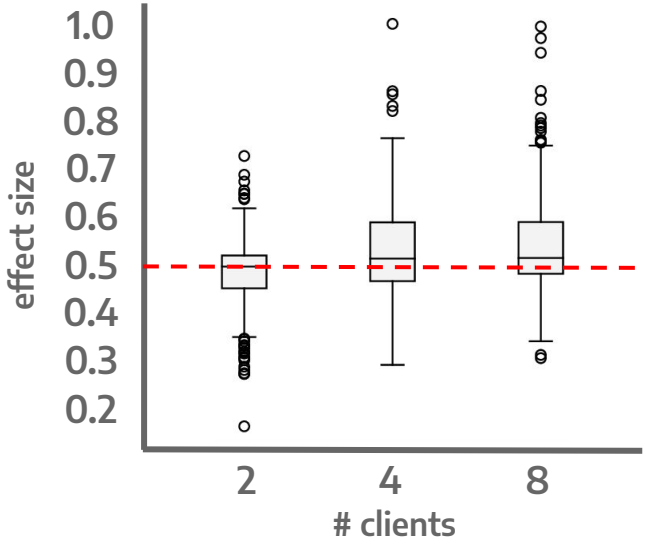

Parallel vs Sequential
effect size
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
2
4
8
clients

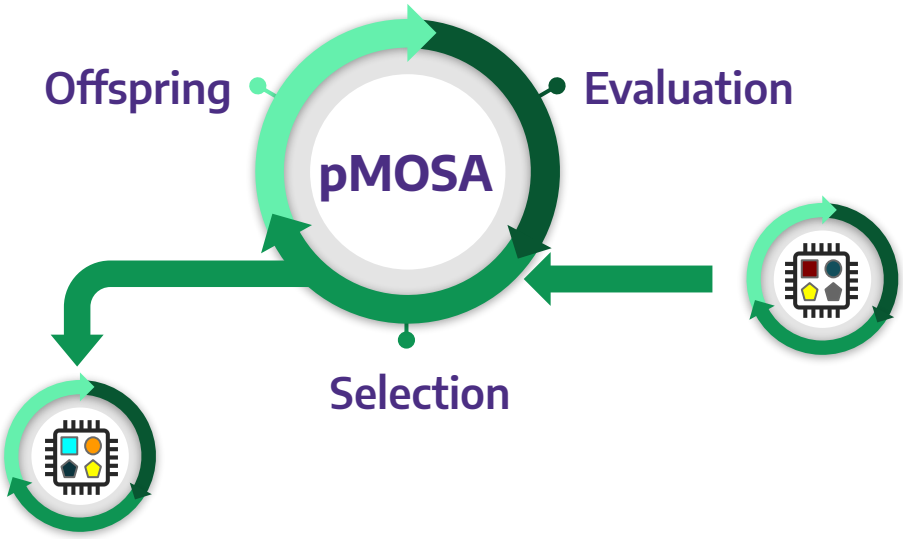


Parallel vs Sequential

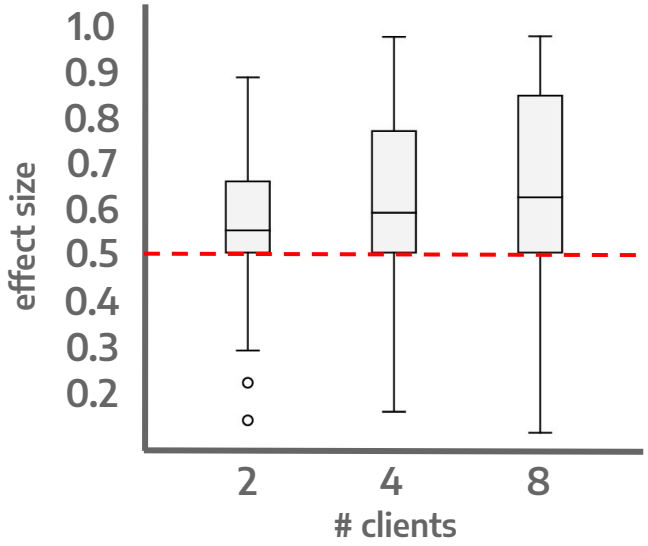


Migration vs non-migration

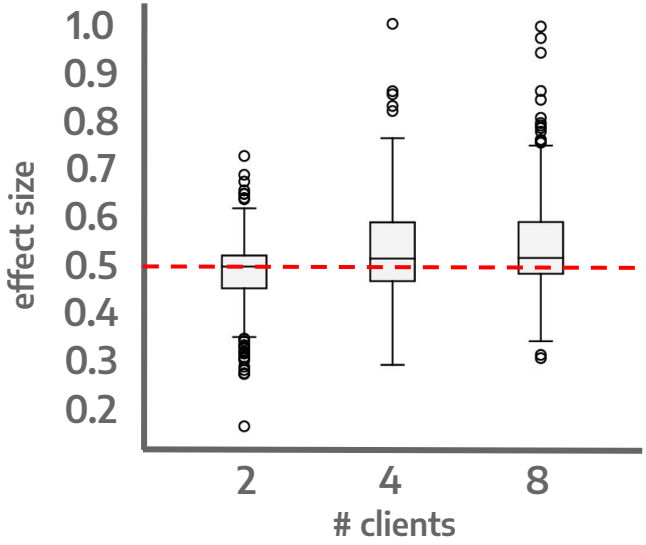




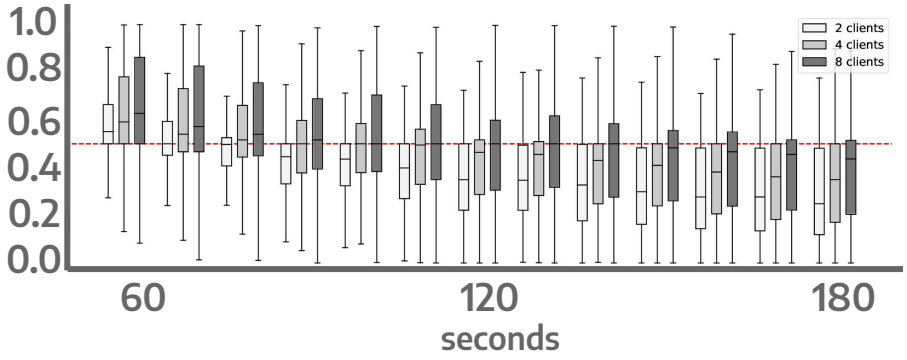
Parallel vs Sequential

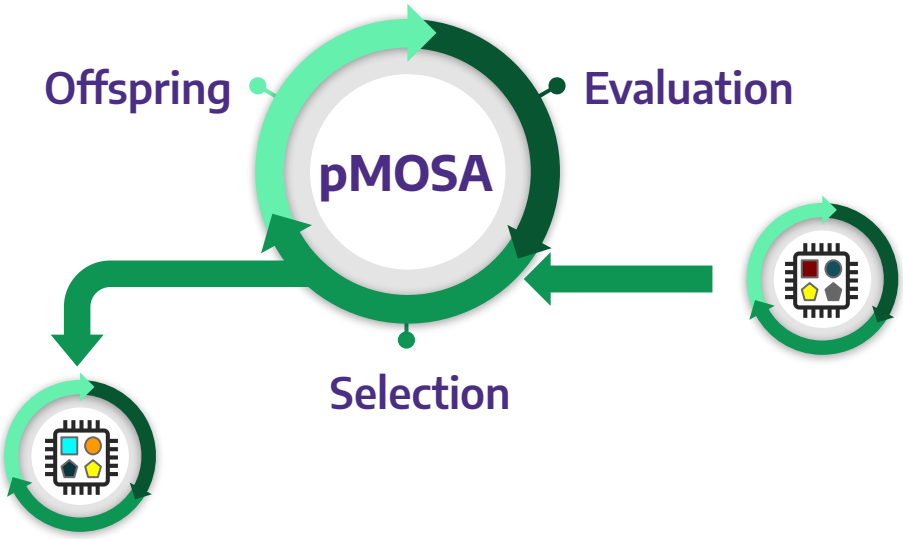


Migration vs non-migration

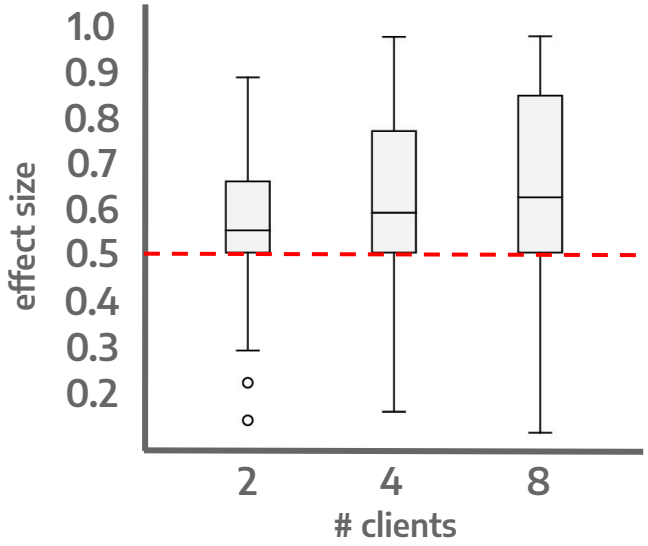


Runtime reduction

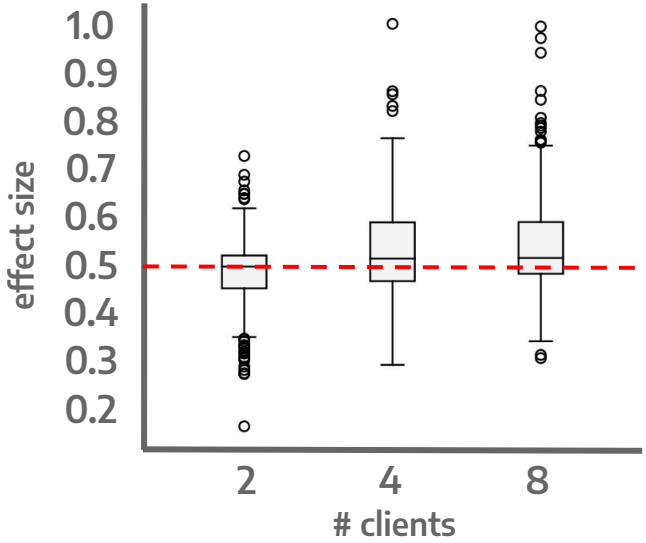




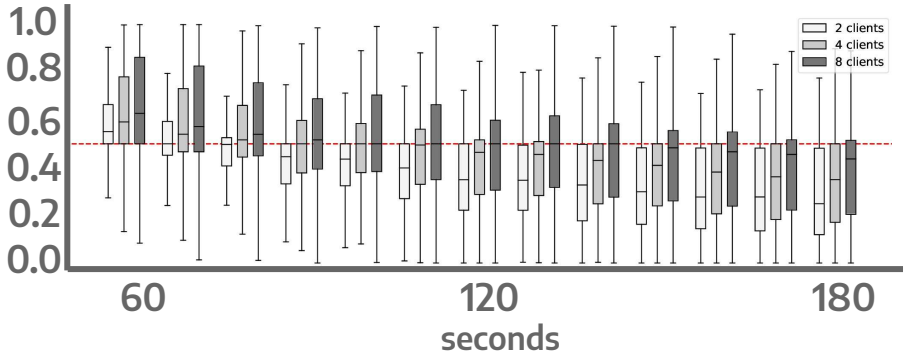
Parallel vs Sequential



Migration vs non-migration



Runtime reduction



http://www.evosuite.org/