On Parameter Tuning in Search Based Software Engineering

Andrea Arcuri and Gordon Fraser
Simula Research Laboratory
Saarland University

Java – Stack/src/test/Stack.java – Eclipse SDK – /Users/fraser/Documents/Workspace2

Package E | Hierarchy

- Stack
  - src
    - test
      - Stack.java
  - JRE System Library [J2SE-1.5]

Stack.java

```
package test;

public class Stack {

    private int maxStack;
    private int emptyStack;
    private int top;
    private char[] items;

    public Stack(int size) {
        maxStack= size;
        emptyStack = -1;
        top = emptyStack;
        items = new char[maxStack];
    }

    public void push(char c) {
        items[++top] = c;
    }

    public char pop() {
        return items[top--];
    }

    public boolean full()  {
        return top + 1 == maxStack;
    }

    public boolean empty()  {
        return top == emptyStack;
    }
}
```

Outline

- test
- Stack
  - maxStack : int
  - emptyStack : int
  - top : int
  - items : char[]
  - Stack(int)
  - push(char) : void
  - pop() : char
  - full() : boolean
  - empty() : boolean

Java – Stack/src/test/Stack.java – Eclipse SDK – /Users/fraser/Documents/Workspace2

Package E | Hierarchy

- Stack
  - src
    - test
      - Stack.java
  - JRE System Library [J2SE-1.5]

Stack.java

```
package test;

public class Stack {

    private int maxStack;
    private int emptyStack;
    private int top;
    private char[] items;

    public Stack(int size) {
        maxStack= size;
        emptyStack = -1;
        top = emptyStack;
        items = new char[maxStack];
    }

    public void push(char c) {
        items[++top] = c;
    }

    public char pop() {
        return items[top--];
    }

    public boolean full()  {
        return top + 1 == maxStack;
    }

    public boolean empty()  {
        return top == emptyStack;
    }
}
```

Outline

- test
- Stack
  - maxStack : int
  - emptyStack : int
  - top : int
  - items : char[]
  - Stack(int)
  - push(char) : void
  - pop() : char
  - full() : boolean
  - empty() : boolean

Java – Stack/src/test/Stack.java – Eclipse SDK – /Users/fraser/Documents/Workspace2

Package E | Hierarchy

- Stack
  - src
    - test
      - Stack.java
  - JRE System Library [J2SE-1.5]

Stack.java

```
package test;

public class Stack {

    private int maxStack;
    private int emptyStack;
    private int top;
    private char[] items;

    public Stack(int size) {
        maxStack= size;
        emptyStack = -1;
        top = emptyStack;
        items = new char[maxStack];
    }

    public void push(char c) {
        items[++top] = c;
    }

    public char pop() {
        return items[top--];
    }

    public boolean full()  {
        return top + 1 == maxStack;
    }

    public boolean empty()  {
        return top == emptyStack;
    }
}
```

Search algorithm

Outline

- test
- Stack
  - maxStack : int
  - emptyStack : int
  - top : int
  - items : char[]
  - Stack(int)
  - push(char) : void
  - pop() : char
  - full() : boolean
  - empty() : boolean

Java – Stack/src/test/Stack.java – Eclipse SDK – /Users/fraser/Documents/Workspace2

Package E | Hierarchy

- Stack
  - src
    - test
      - Stack.java

Population size

Search algorithm

Stack.java

```
package test;

public class Stack {

    private int maxStack;

    p
        maxStack= size;
        emptyStack = -1;
        top = emptyStack;
        items = new char[maxStack];
    }

    public void push(char c) {
        items[++top] = c;
    }

    public char pop() {
        return items[top--];
    }

    public boolean full()  {
        return top + 1 == maxStack;
    }

    public boolean empty()  {
        return top == emptyStack;
    }
}
```

Outline

- test
- Stack
  - maxStack : int
  - emptyStack : int
  - top : int
  - items : char[]
  - Stack(int)
  - push(char) : void
  - pop() : char
  - full() : boolean
  - empty() : boolean

Java – Stack/src/test/Stack.java – Eclipse SDK – /Users/fraser/Documents/Workspace2

Package E | Hierarchy

- Stack
  - src
    - test
      - Stack.java

Stack.java

```
package test;

public class Stack {

    private int maxStack;

    p
        maxStack= size;
        emptyStack = -1;
        top = emptyStack;
        items = new char[maxStack];
    }

    public void push(char c) {
        items[++top] = c;
    }

    public char pop() {
        return items[top--];
    }

    public boolean full()  {
        return top + 1 == maxStack;
    }

    public boolean empty()  {
        return top == emptyStack;
    }
}
```

Outline

- test
- Stack
  - maxStack : int
  - emptyStack : int
  - top : int
  - items : char[]
  - Stack(int)
  - push(char) : void
  - pop() : char
  - full() : boolean
  - empty() : boolean

Population size

Search algorithm

Search budget


Mutation rate
Budget
Selection algorithm
Elitism
Search budget
Search algorithm
Population size
Crossover rate
Scaling window
Bloat control
Bloat control
Crossover technique
Tournament size
Generation gap
Parent replacement
Kin compensation
Local search
Rank bias
Seeding


Mutation rate
Object reuse
Budget
Selection algorithm
Elitism
...
Test suite size
Search budget
Population size
Search algorithm
Test length
Test timeout
Crossover rate
Scaling window
Bloat control
...
Bloat control
Crossover technique
Tournament size
Generation gap
Null probability
Parent replacement
Number range
Local search
...
Rank bias
Seeding

Parameter Tuning

Parameter Tuning

Researcher

Performs empirical studies

Compares techniques

Compares tools

Parameter Tuning

Researcher

Performs empirical studies

Compares techniques

Compares tools

Tool Developer

Wants highest effectiveness on all possible problems

Does not know the problems the tool will be applied to

Parameter Tuning

Researcher

Performs empirical studies

Compares techniques

Compares tools

Tool Developer

Wants highest effectiveness on all possible problems

Does not know the problems the tool will be applied to

User

May not know about SBSE

Wants best performance on his problem

Uses predefined parameters

May only wish to set search duration

How large is the potential impact of a wrong choice of parameter settings?

How large is the potential impact of a wrong choice of parameter settings?

How does a “default” setting perform?

How large is the potential impact of a wrong choice of parameter settings?

How does a "default" setting perform?

If we tune on a set of problems, how will its performance be on other new problems?

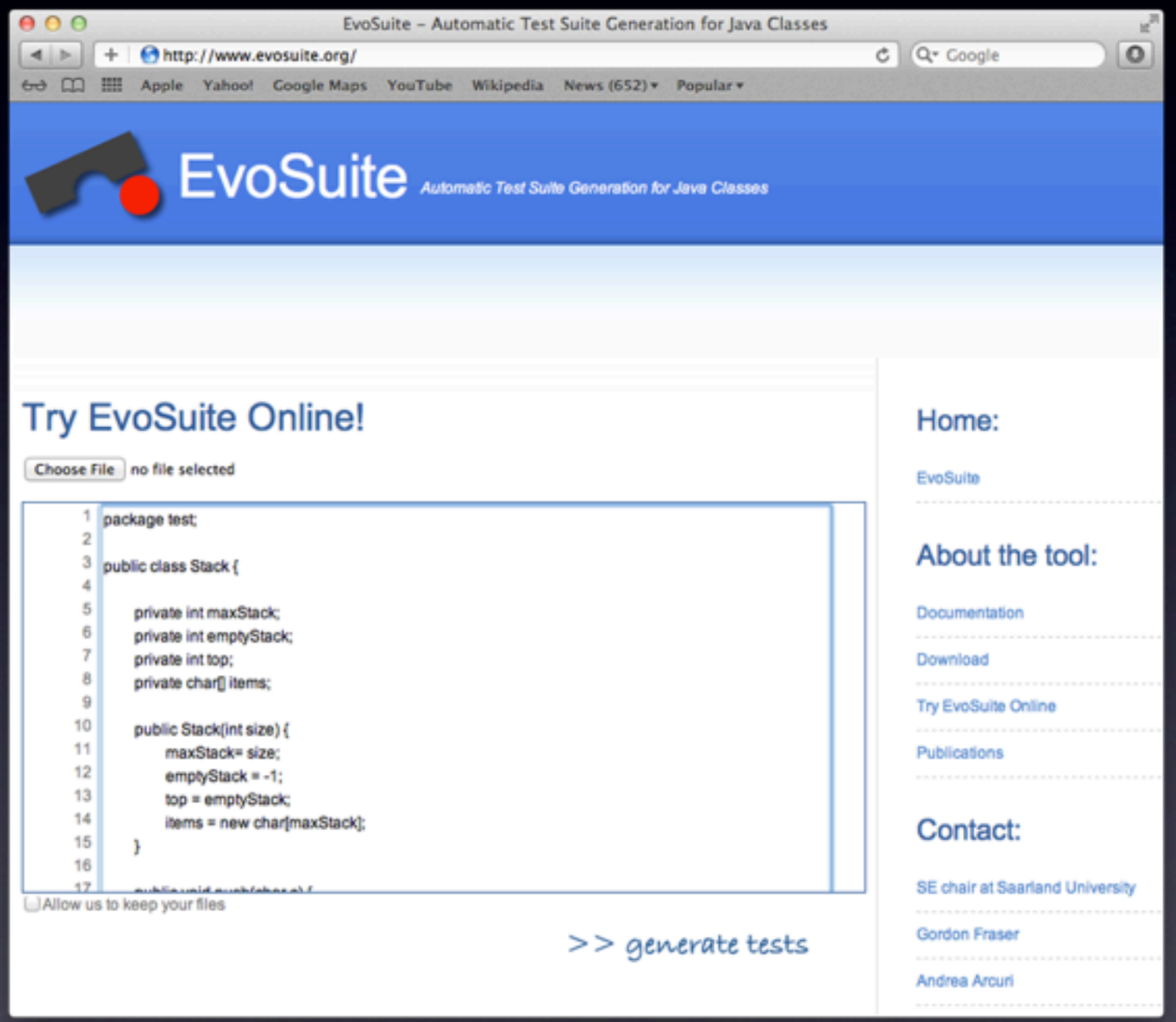
How large is the potential impact of a wrong choice of parameter settings?

How does a "default" setting perform?

If we tune on a set of problems, how will its performance be on other new problems?

What are the effects of the search budget on parameter tuning?

EvoSuite

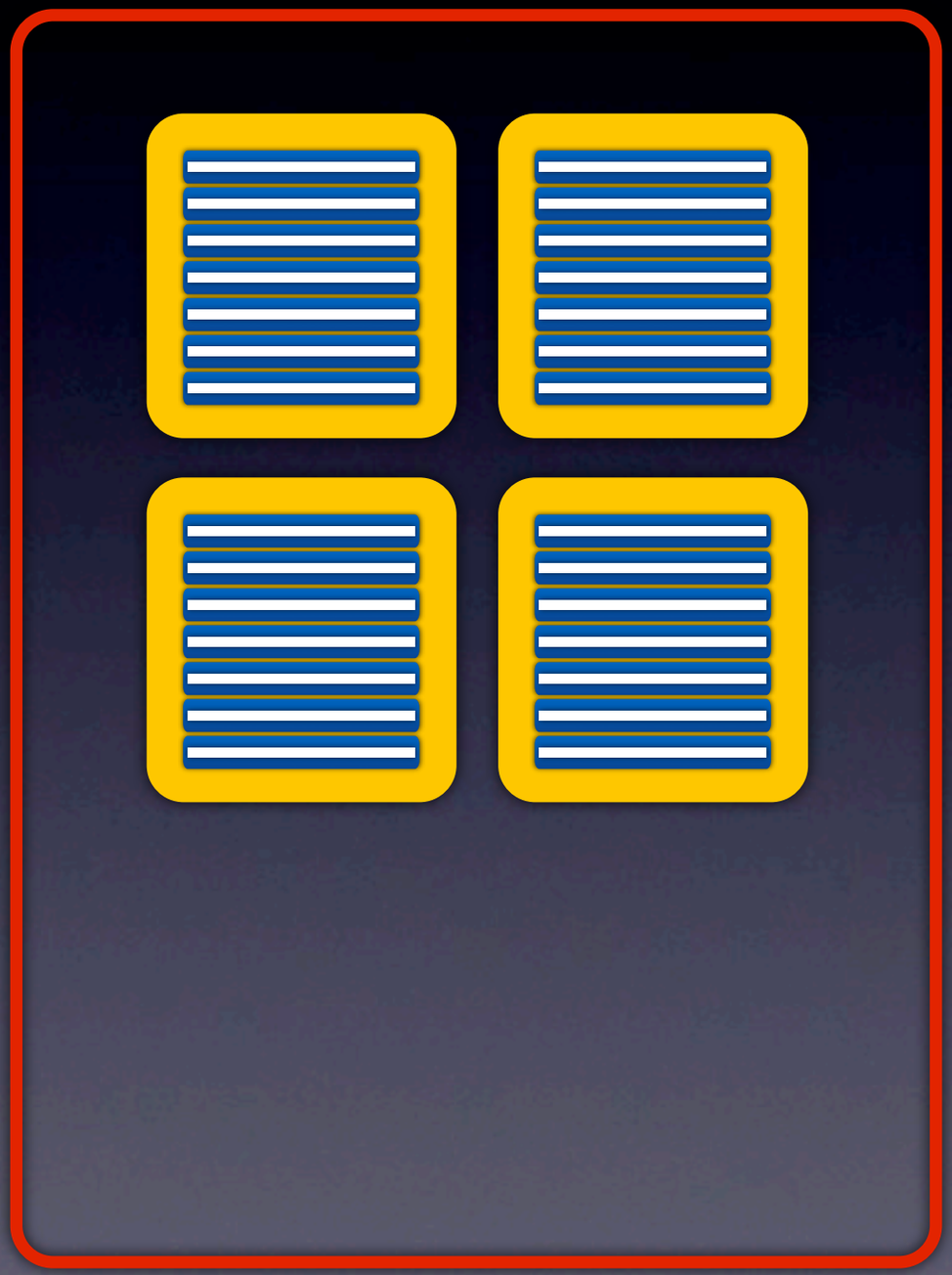

EvoSuite – Automatic Test Suite Generation for Java Classes
http://www.evosuite.org/
Google
Apple
Yahoo!
Google Maps
YouTube
Wikipedia
News (652)
Popular
EvoSuite
Automatic Test Suite Generation for Java Classes
Try EvoSuite Online!
Choose File
no file selected
1 package test;
2
3 public class Stack {
4
5 private int maxStack;
6 private int emptyStack;
7 private int top;
8 private char[] items;
9
10 public Stack(int size) {
11 maxStack= size;
12 emptyStack = -1;
13 top = emptyStack;
14 items = new char[maxStack];
15 }
16
Allow us to keep your files
>> generate tests
Home:
EvoSuite
About the tool:
Documentation
Download
Try EvoSuite Online
Publications
Contact:
SE chair at Saarland University
Gordon Fraser
Andrea Arcuri

Test Suite Generation

```
int var0 = 10
YearMonthDay var1 = new YearMonthDay(var0);
TimeOfDay var2 = new TimeOfDay();
DateTime var3 = var1.toDateTime(var2);
DateTime var4 = var3.minus(var0);
DateTime var5 = var4.plusSeconds(var0);
int var6 = var5.getMillis();
```

Test Suite Generation

Test Suite Generation



Test Suite Generation

Crossover

Crossover

Mutation

Mutation

Mutation

Fitness Example

```
public class Foo {

  void foo(int x, int y) {
    if(x > 0)
      // do something
    if(x == y)
      // do something
  }
}
```

Fitness Example

```
public class Foo {

  void foo(int x, int y) {
    if(x > 0)
      // do something
    if(x == y)
      // do something
  }
}
```

foo(10, 0)

Fitness Example

```
public class Foo {

  void foo(int x, int y) {
    if(x > 0)
      // do something
    if(x == y)
      // do something
  }
}
```

foo(10, 0)

Branch	Distance

Fitness Example

```
public class Foo {

  void foo(int x, int y) {
    if(x > 0)
      // do something
    if(x == y)
      // do something
  }
}
```

foo(10, 0)

Branch	Distance
x > 0	0

Fitness Example

```
public class Foo {

  void foo(int x, int y) {
    if(x > 0)
      // do something
    if(x == y)
      // do something
  }
}
```

foo(10, 0)

Branch	Distance
x > 0	0
x <= 0	10

Fitness Example

```
public class Foo {

  void foo(int x, int y) {
    if(x > 0)
      // do something
    if(x == y)
      // do something
  }
}
```

foo(10, 0)

Branch	Distance
x > 0	0
x <= 0	10
x == y	10

Fitness Example

```
public class Foo {

  void foo(int x, int y) {
    if(x > 0)
      // do something
    if(x == y)
      // do something
  }
}
```

foo(10, 0)

Branch	Distance
x > 0	0
x <= 0	10
x == y	10
x != y	0

Fitness Example

```
public class Foo {

  void foo(int x, int y) {
    if(x > 0)
      // do something
    if(x == y)
      // do something
  }
}
```

foo(10, 0)

foo(0, 0)

Branch	Distance
x > 0	0
x <= 0	10
x == y	10
x != y	0

Fitness Example

```
public class Foo {

  void foo(int x, int y) {
    if(x > 0)
      // do something
    if(x == y)
      // do something
  }
}
```

foo(10, 0)

foo(0, 0)

Branch	Distance
x > 0	0
x <= 0	0
x == y	10
x != y	0

Fitness Example

```
public class Foo {

  void foo(int x, int y) {
    if(x > 0)
      // do something
    if(x == y)
      // do something
  }
}
```

foo(10, 0)

foo(0, 0)

Branch	Distance
x > 0	0
x <= 0	0
x == y	0
x != y	0

Java - Test/evosuite-tests/test/TestStack.java - Eclipse - /Users/fraser/ESECFSE

Package Explorer

- Test
  - src
    - test
      - Stack.java
  - JRE System Library [JavaSE-1.6]
  - Referenced Libraries
  - evosuite-tests
    - test
      - GeneratedTestSuite.java
      - TestStack.java
  - evosuite-files

*TestStack.java

```
public class TestStack extends TestCase {
   //Test case number: 0
   public void test0()  {
      Stack var0 = new Stack();
      assertEquals(var0.getSize(), 0);
   }

   //Test case number: 1
   public void test1()  {
      Stack var0 = new Stack();
      var0.push(0);
      var0.push(0);
      var0.push(0);
      assertEquals(var0.getSize(), 3);
   }

   //Test case number: 2
   public void test2()  {
      Stack var0 = new Stack();
      try {
        var0.pop();
      } catch(RuntimeException e) {
        // Stack is empty
      }
   }

   //Test case number: 3
   public void test3()  {
      Stack var0 = new Stack();
      var0.push(0);
      int var2 = var0.pop();
      assertEquals(var2, 0);
   }
}
```

Writable | Smart Insert | 10 : 42

Parameters

Parameters

- Population size: {4, 10, 50, 100, 200}

Parameters

- Population size: {4, 10, 50, 100, 200}
- Crossover rate: {0, .2, .5, .8, 1}

Parameters

- Population size: {4, 10, 50, 100, 200}
- Crossover rate: {0, .2, .5, .8, 1}
- Elitism rate: {0, 1, 10%, 50%} or steady state

Parameters

- Population size: {4, 10, 50, 100, 200}
- Crossover rate: {0, .2, .5, .8, 1}
- Elitism rate: {0, 1, 10%, 50%} or steady state
- Selection:

Parameters

- Population size: {4, 10, 50, 100, 200}
- Crossover rate: {0, .2, .5, .8, 1}
- Elitism rate: {0, 1, 10%, 50%} or steady state
- Selection:
  - Roulette wheel

Parameters

- Population size: {4, 10, 50, 100, 200}
- Crossover rate: {0, .2, .5, .8, 1}
- Elitism rate: {0, 1, 10%, 50%} or steady state
- Selection:
  - Roulette wheel
  - Tournament with size either 2 or 7

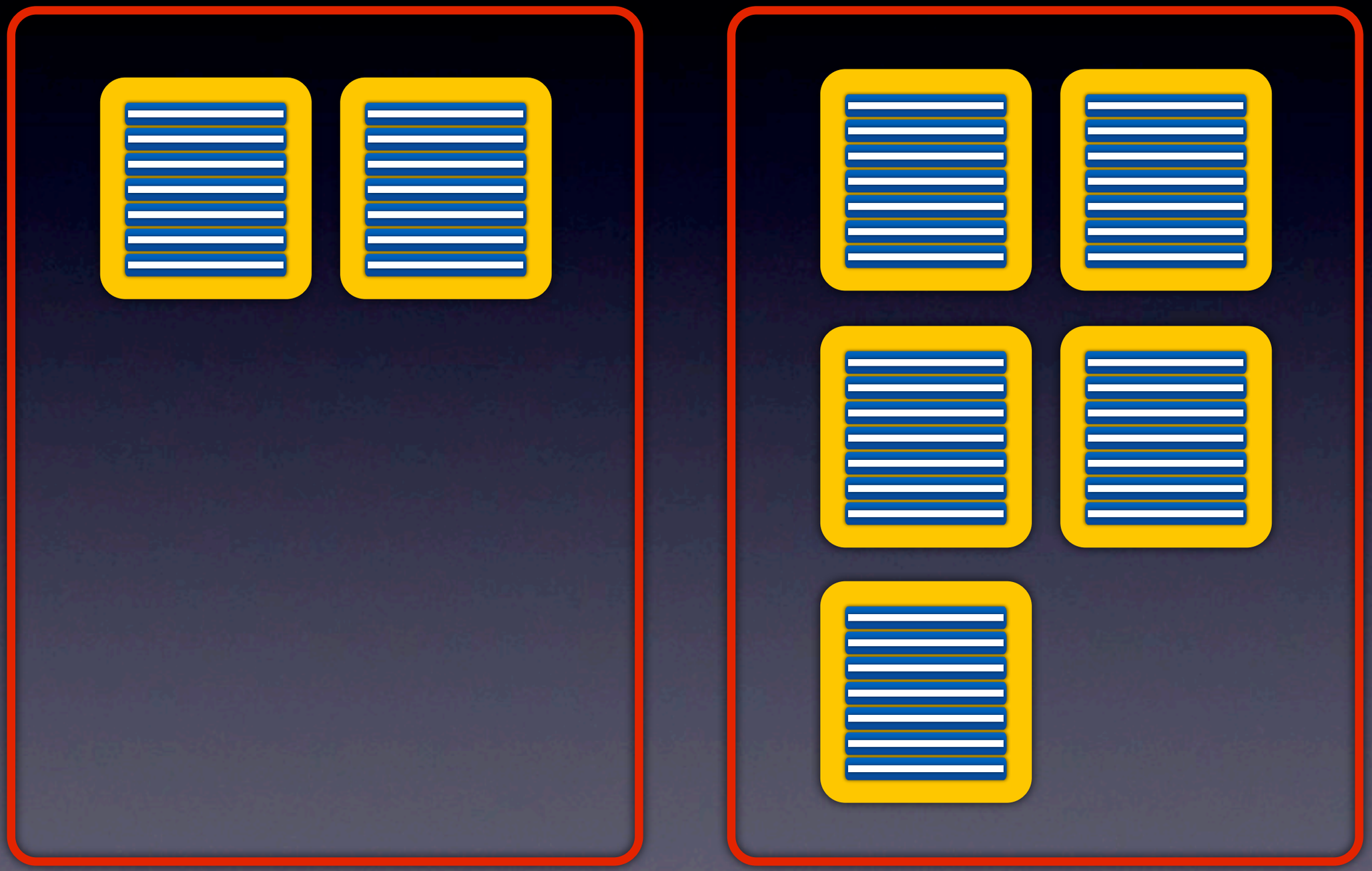
Parameters

- Population size: {4, 10, 50, 100, 200}
- Crossover rate: {0, .2, .5, .8, 1}
- Elitism rate: {0, 1, 10%, 50%} or steady state
- Selection:
  - Roulette wheel
  - Tournament with size either 2 or 7
  - Rank selection with bias either 1.2 or 1.7

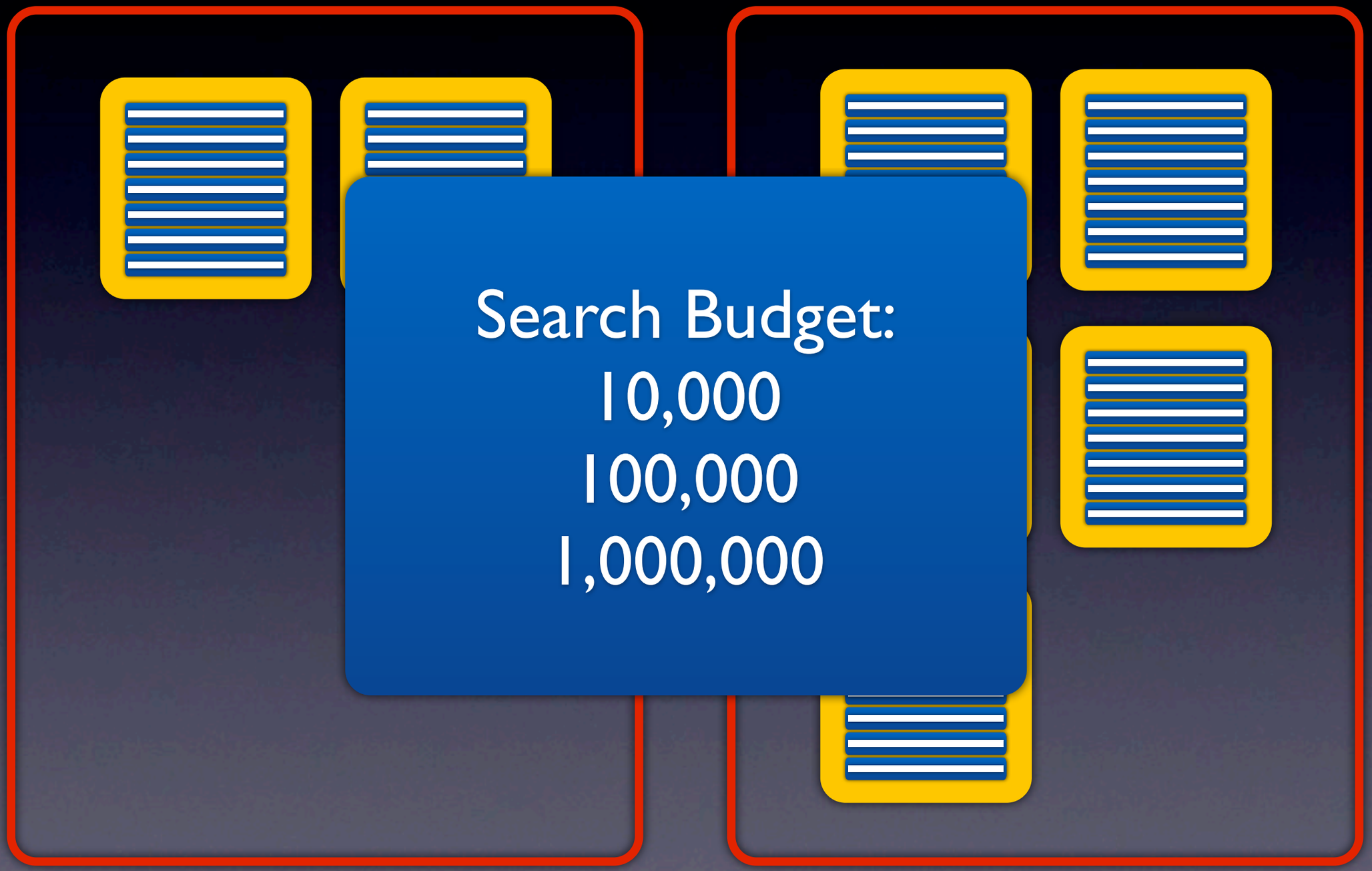
Parameters

- Population size: {4, 10, 50, 100, 200}
- Crossover rate: {0, .2, .5, .8, 1}
- Elitism rate: {0, 1, 10%, 50%} or steady state
- Selection:
  - Roulette wheel
  - Tournament with size either 2 or 7
  - Rank selection with bias either 1.2 or 1.7
- Parent replacement check (activated or not)

Parameters



Parameters



Case Studies

- Apache Commons Math
- Joda Time
- Java Collections
- Industrial Case Study
- Java Zip Utilities
- JGraphT
- String Case Study
- JDom
- Google Collections
- Numerical Case Study
- Commons CLI
- Apache Commons Codec
- NanoXML

Case Studies

- Apache Commons Math
- Joda Time
- Java Collect
- Industrial C
- Java Zip Uti
- JGraphT
- String Case Study
- JDom
- Google Collections
- cal Case Study
- ns CLI
- Commons
- NanoXML

> 80% Coverage
< 100% Coverage
“reasonably large”

$$20 \times 5^4$$

$$20 \times 5^4 \times 2$$

$$20 \times 5^4 \times 2 \times 3$$

$$20 \times 5^4 \times 2 \times 3 \times 15$$

$$20 \times 5^4 \times 2 \times 3 \times 15 =$$

1,250,000

RQ I

How large is the potential impact of a wrong choice of parameter settings?

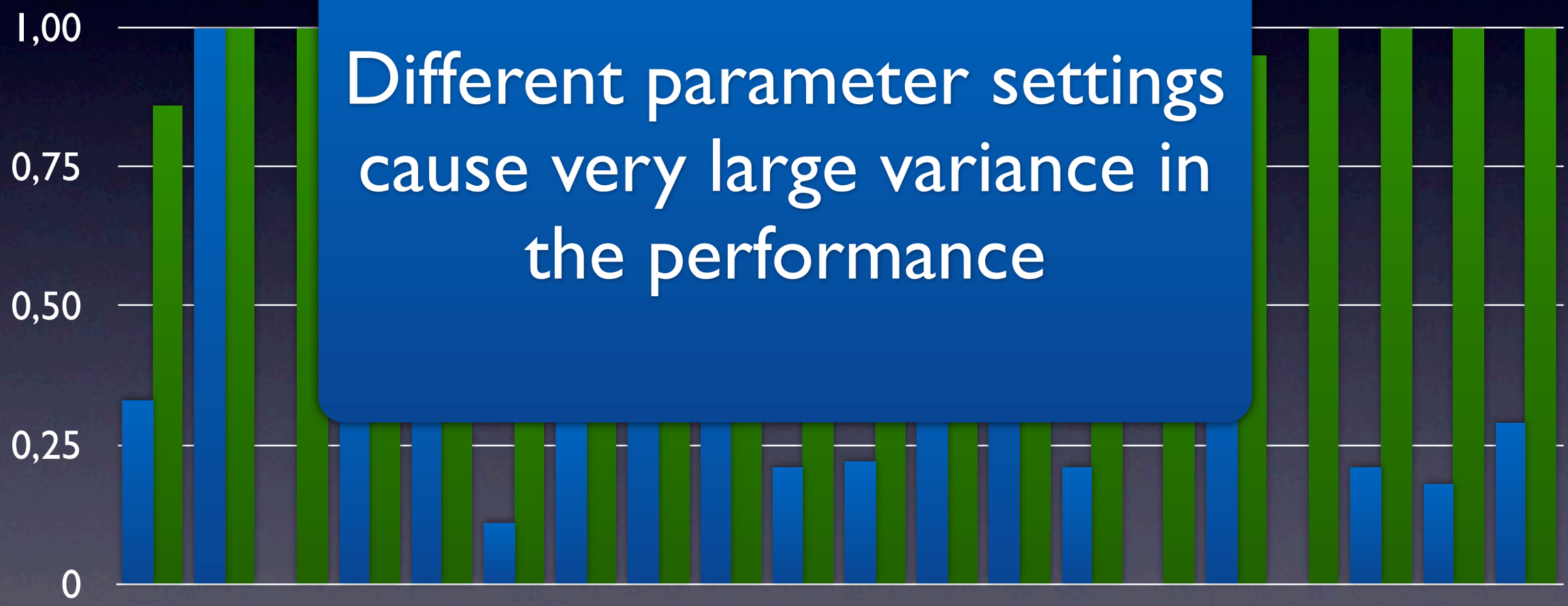
Branch Coverage

Branch Coverage

Branch Coverage

Branch Coverage

Branch Coverage



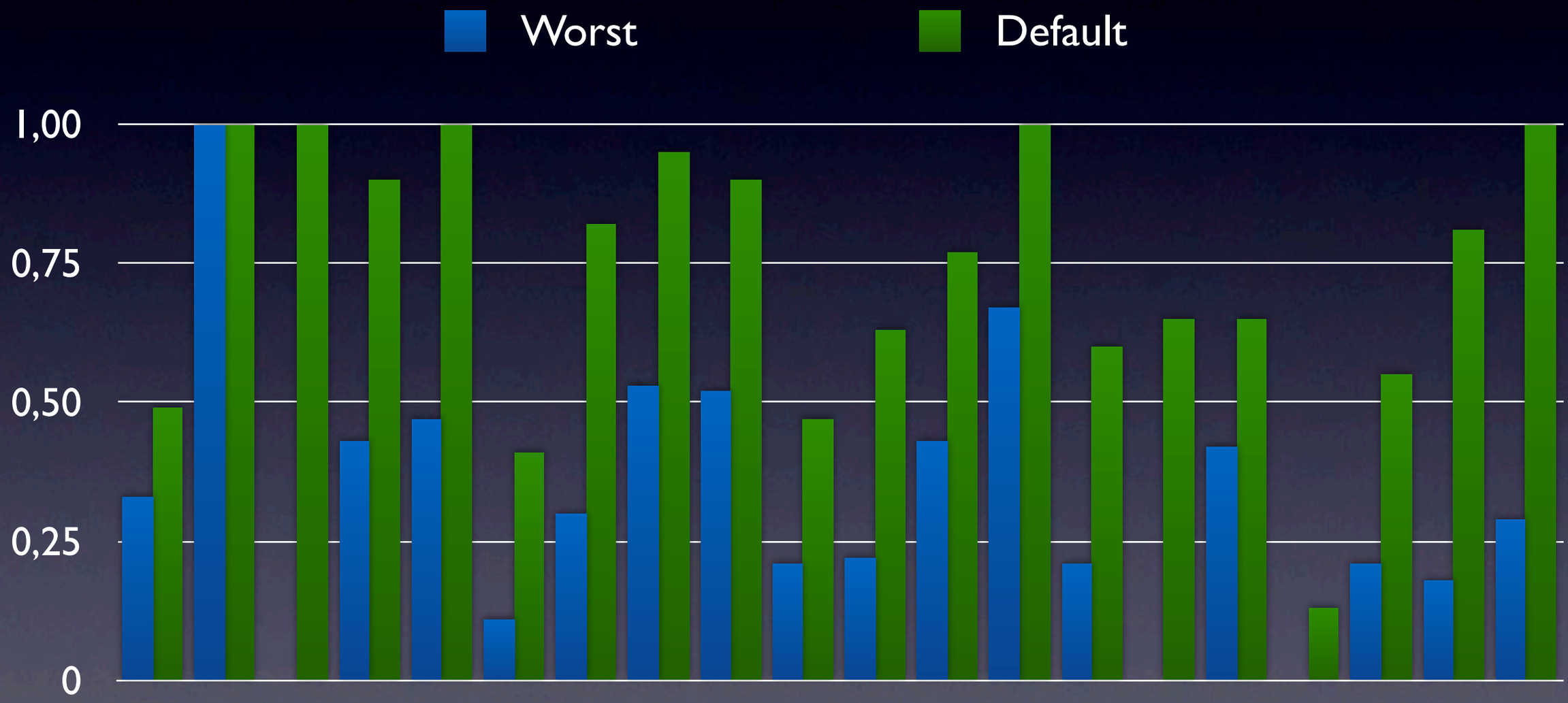
RQ2

How does a “default” setting compare to the best and worst achievable performance?

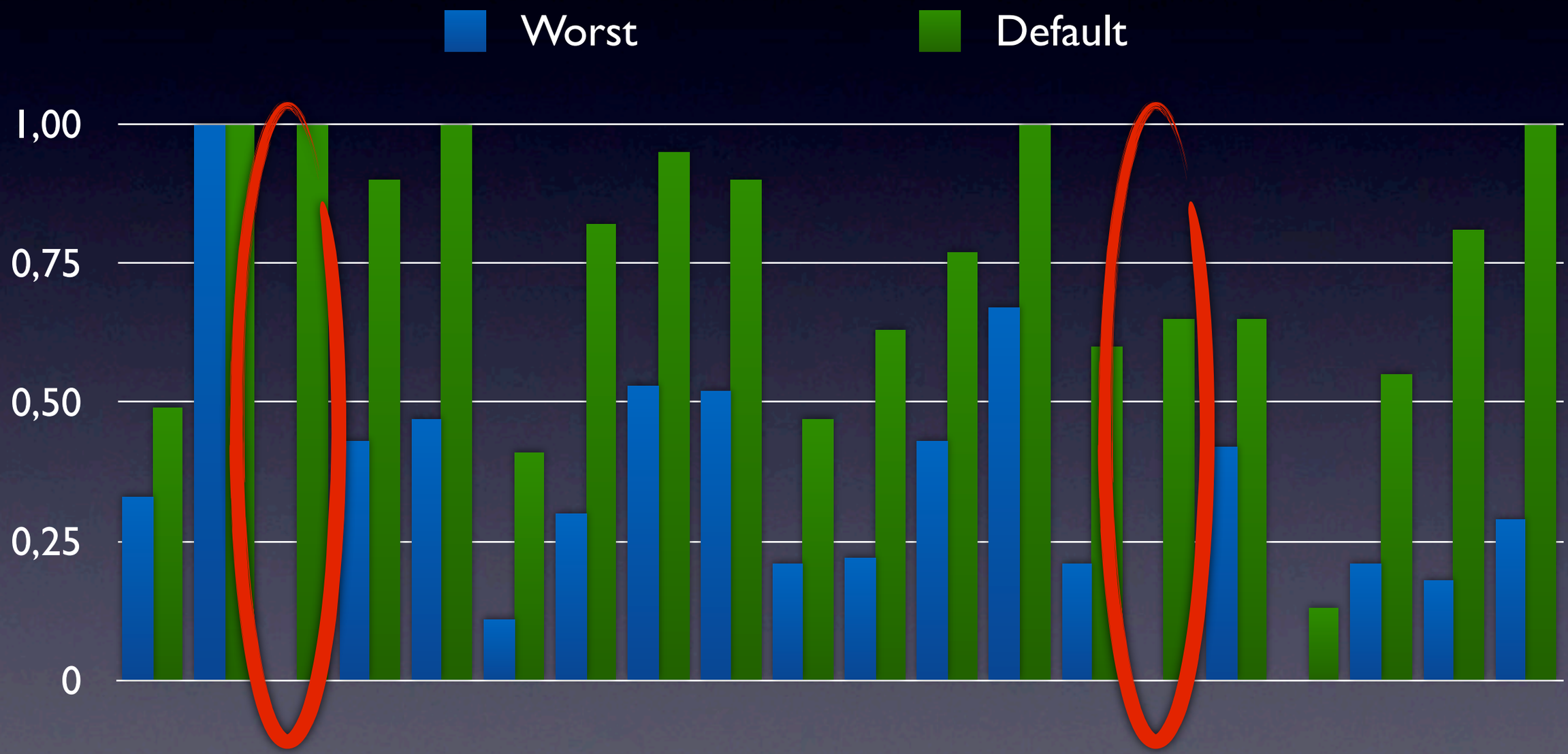
Default Values

- Population size: 100
- Crossover rate: 0.8
- Rank selection with 1.7 bias
- 10% elitism rate
- No parent replacement check

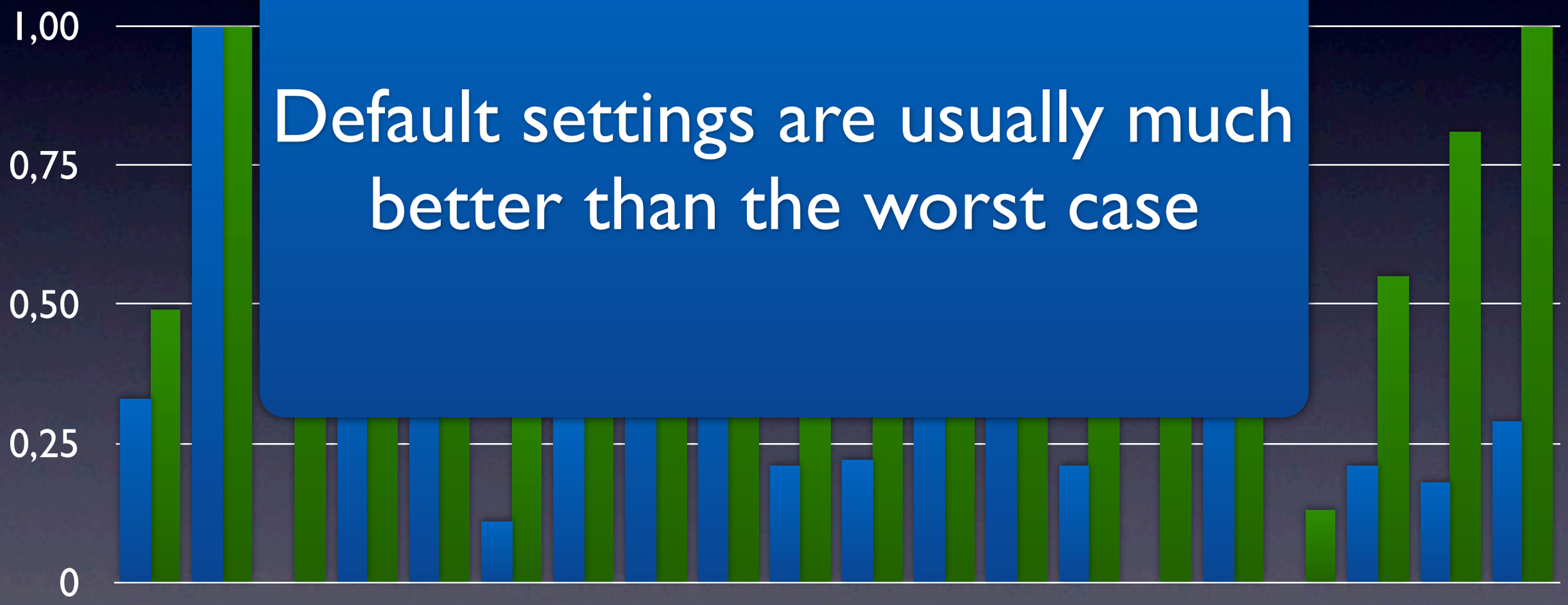
Branch Coverage



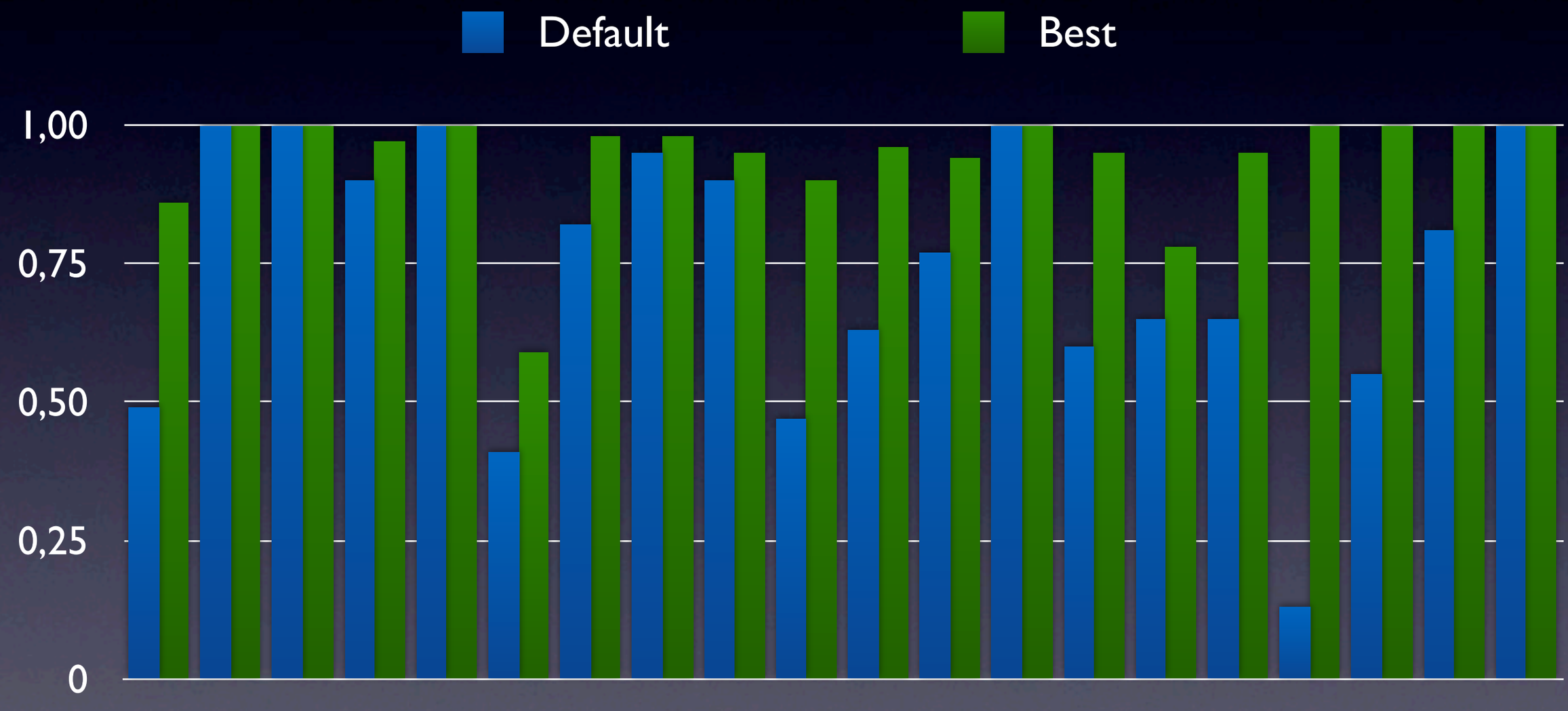
Branch Coverage



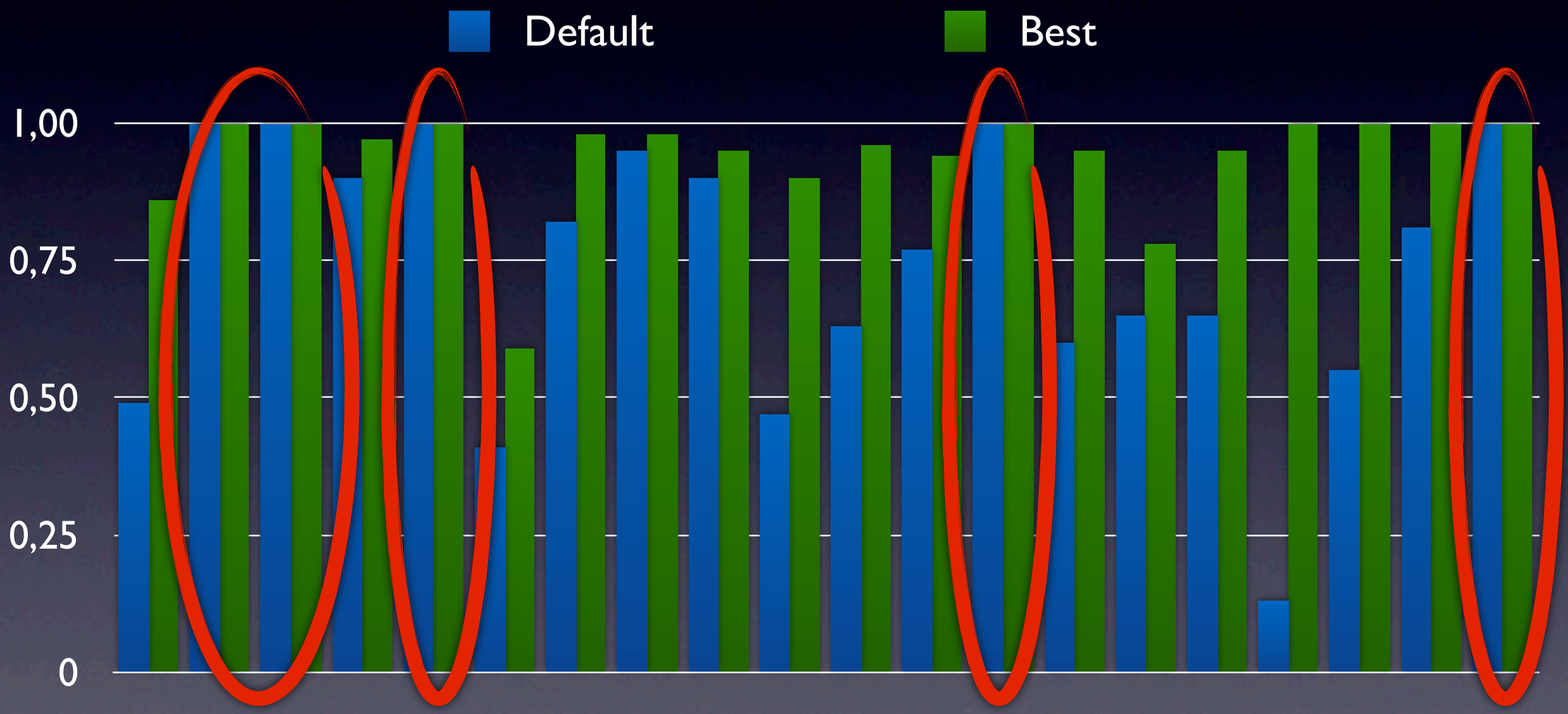
Branch Coverage



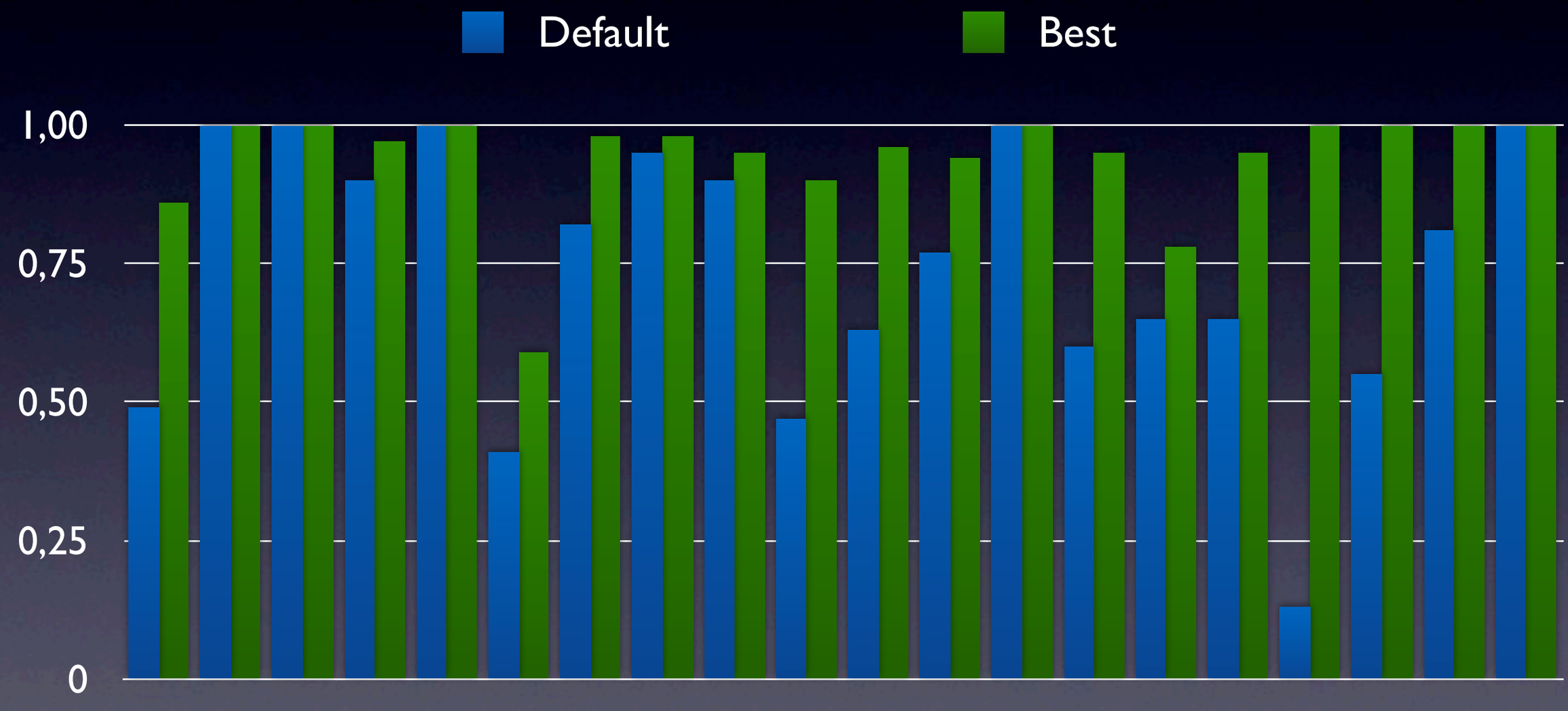
Branch Coverage



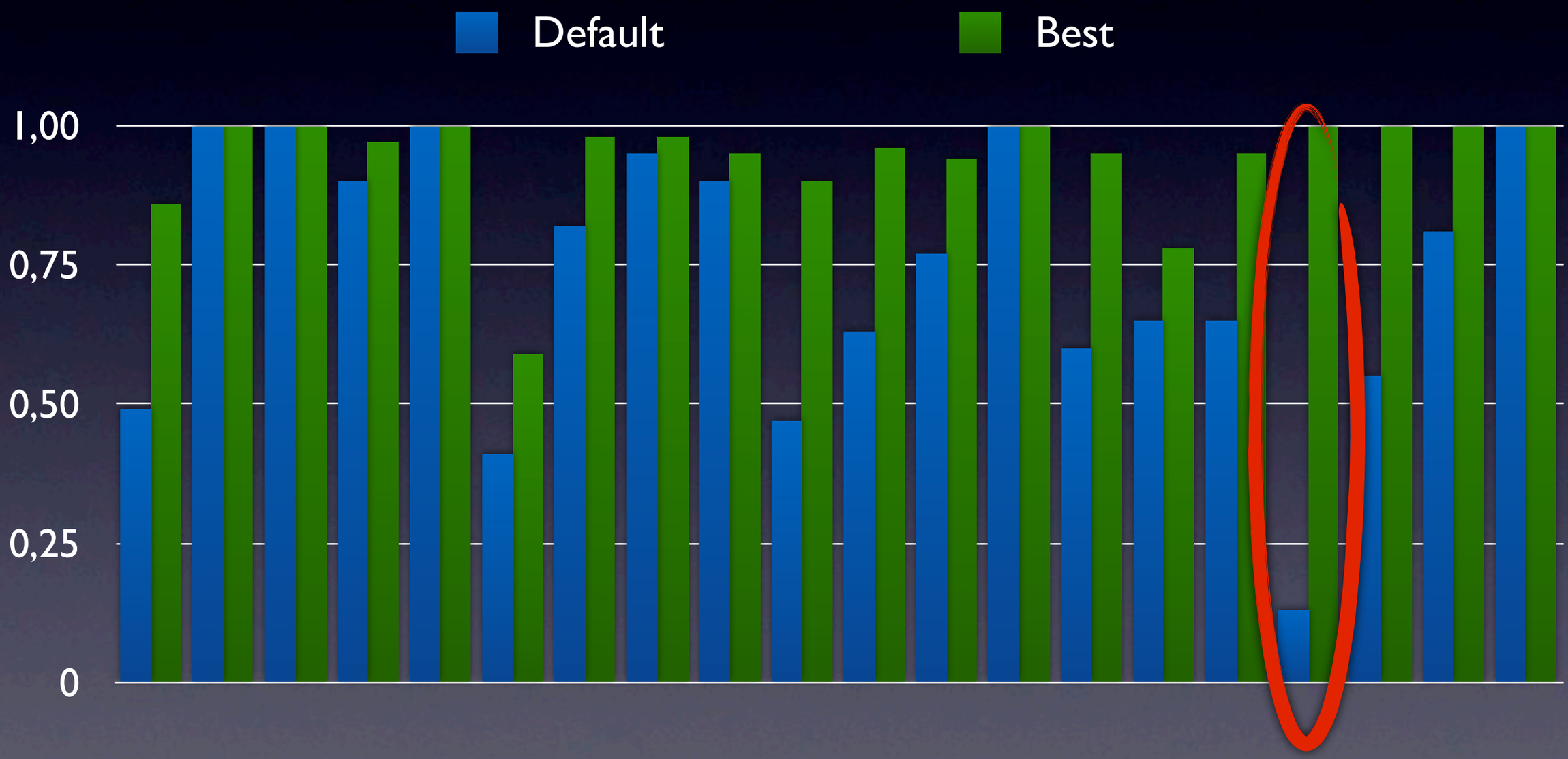
Branch Coverage



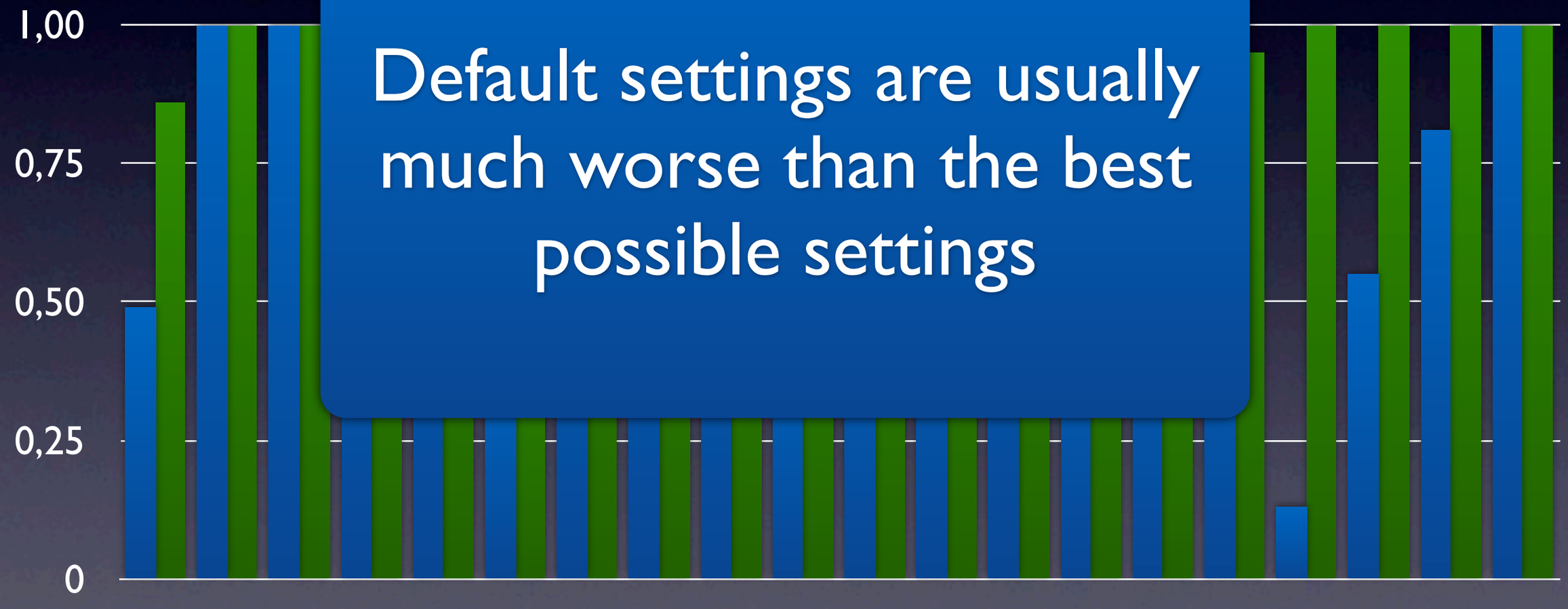
Branch Coverage



Branch Coverage



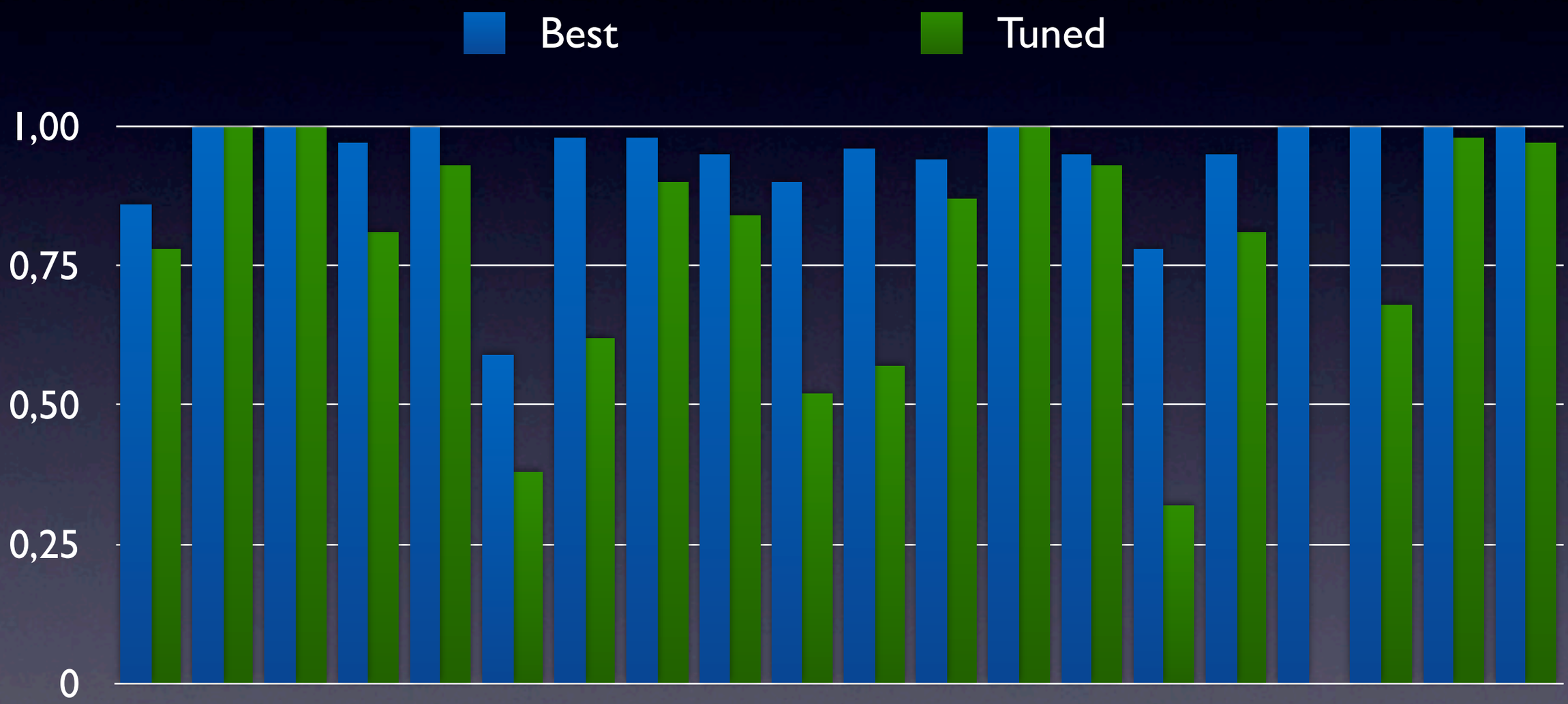
Branch Coverage



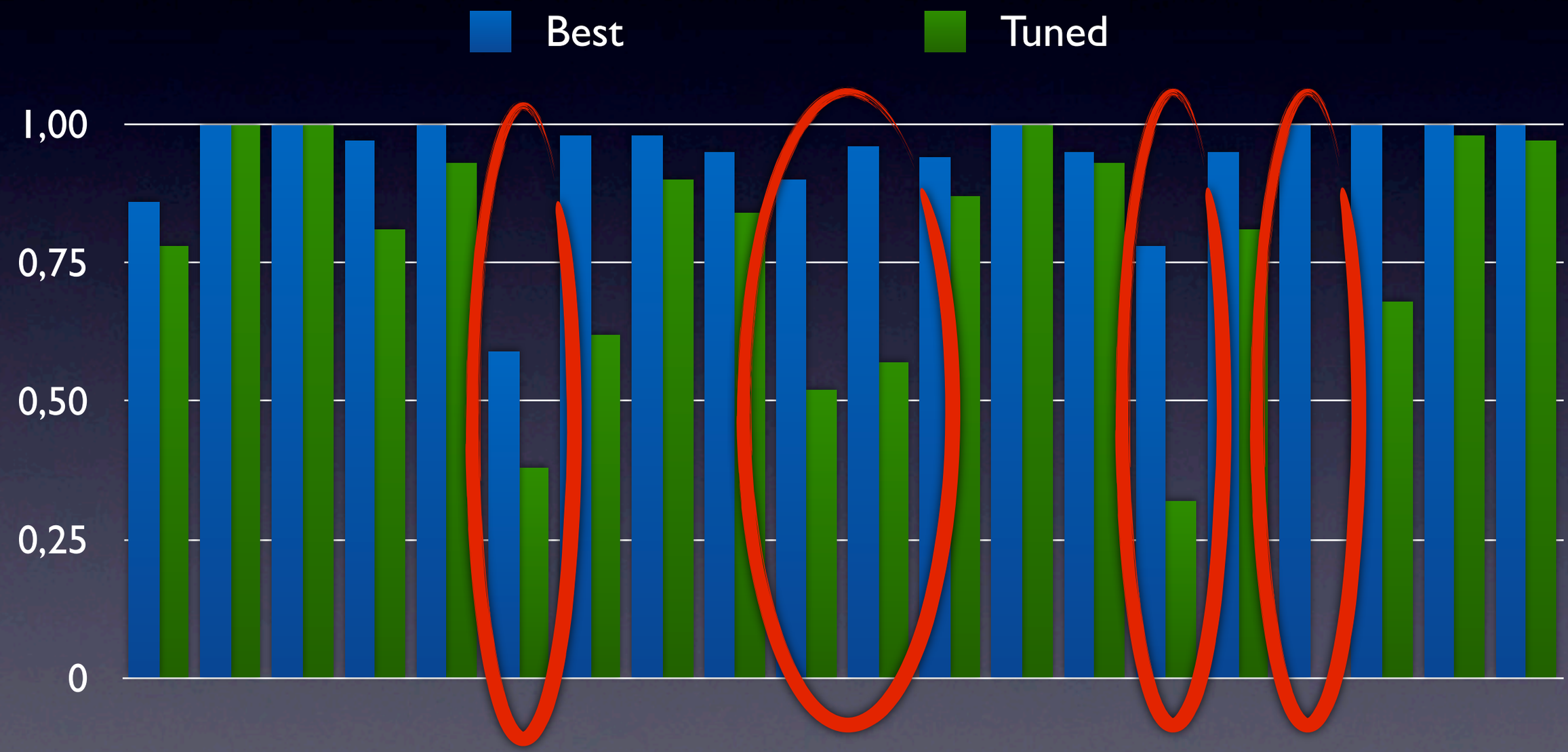
RQ3

If we tune a search algorithm based on a set of classes, how will its performance be on other new classes?

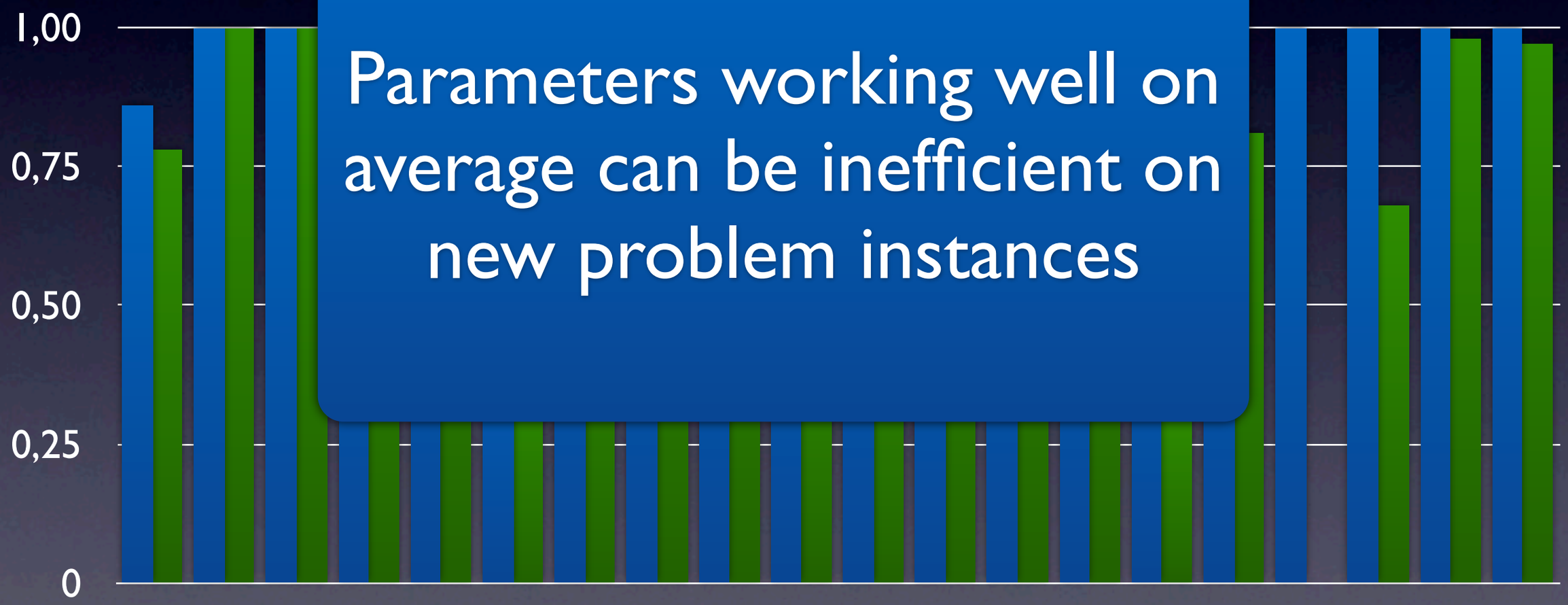
Branch Coverage



Branch Coverage

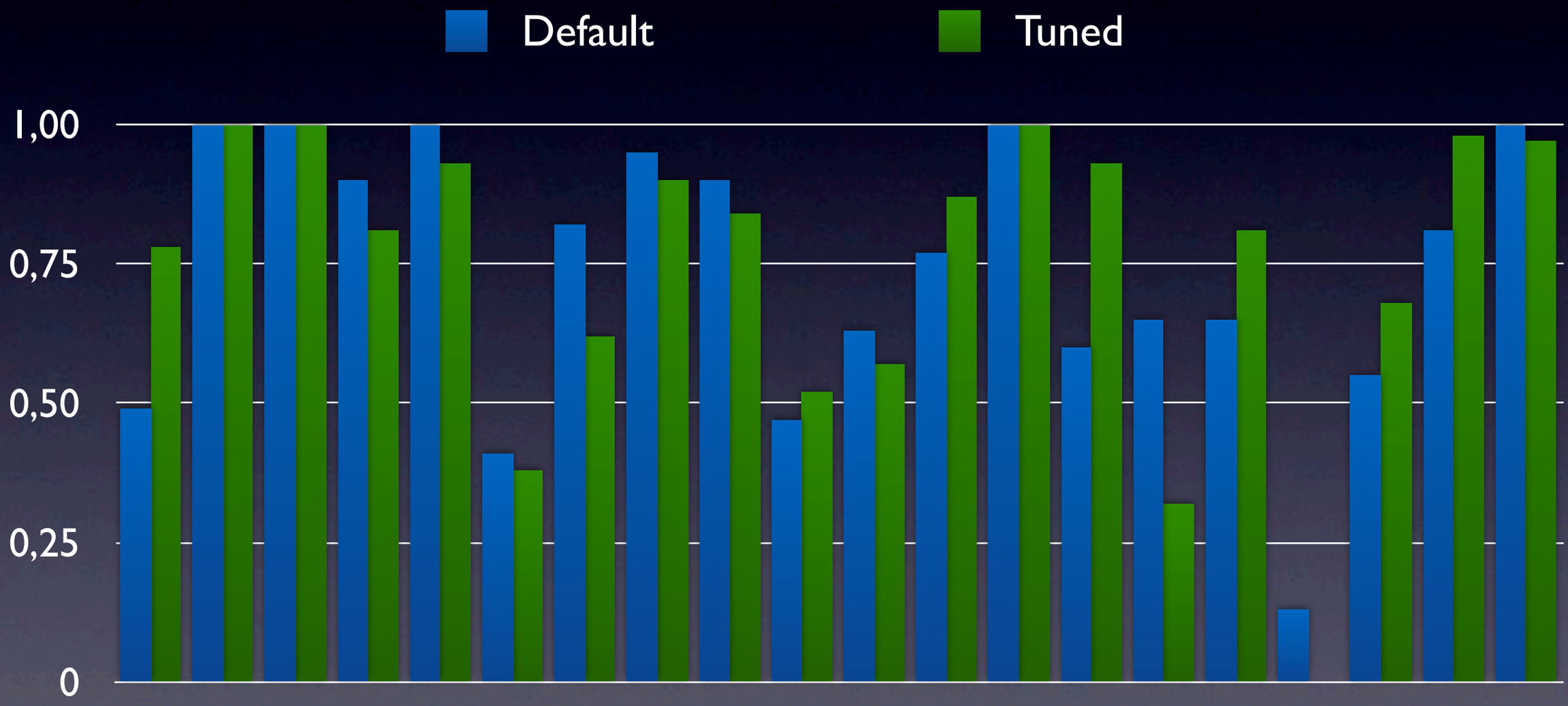


Branch Coverage



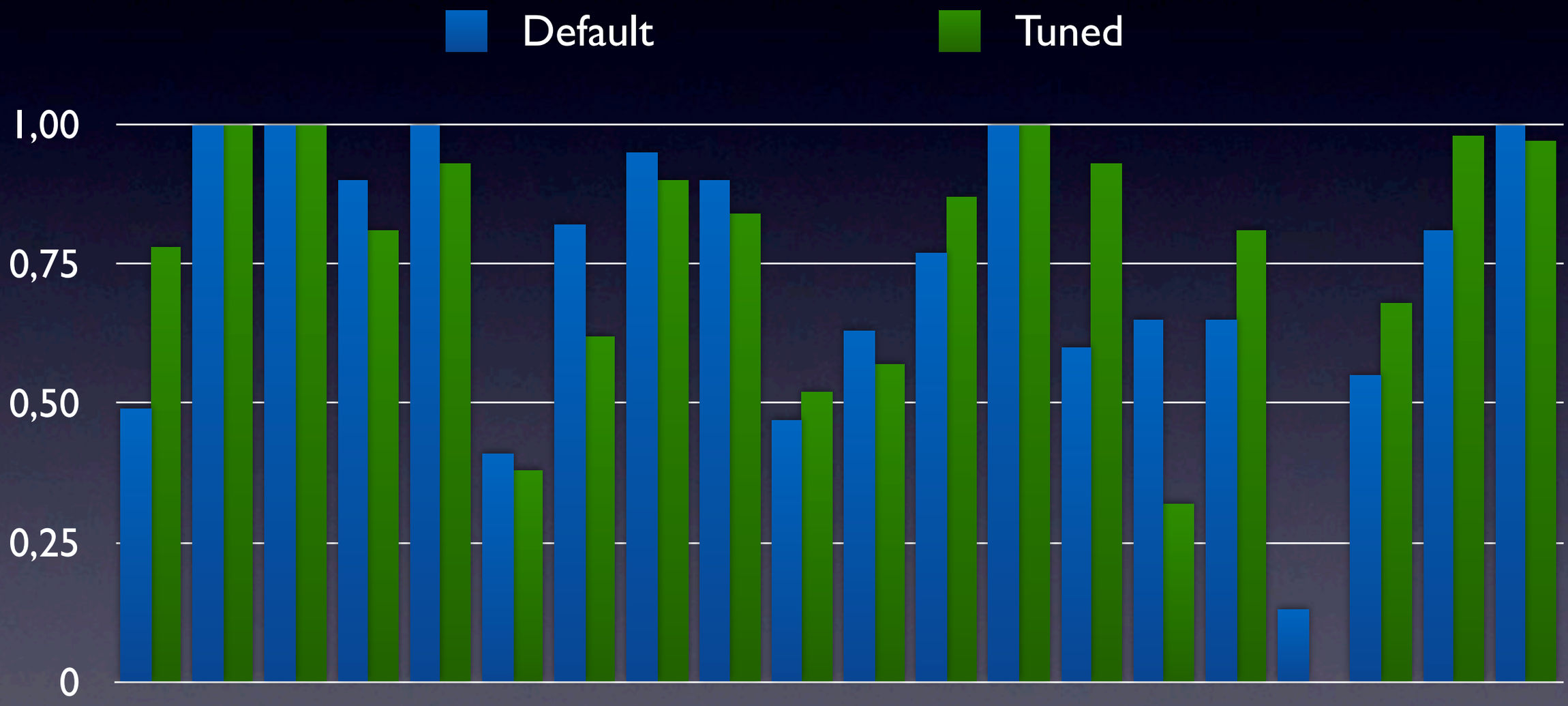
Parameters working well on average can be inefficient on new problem instances

Branch Coverage

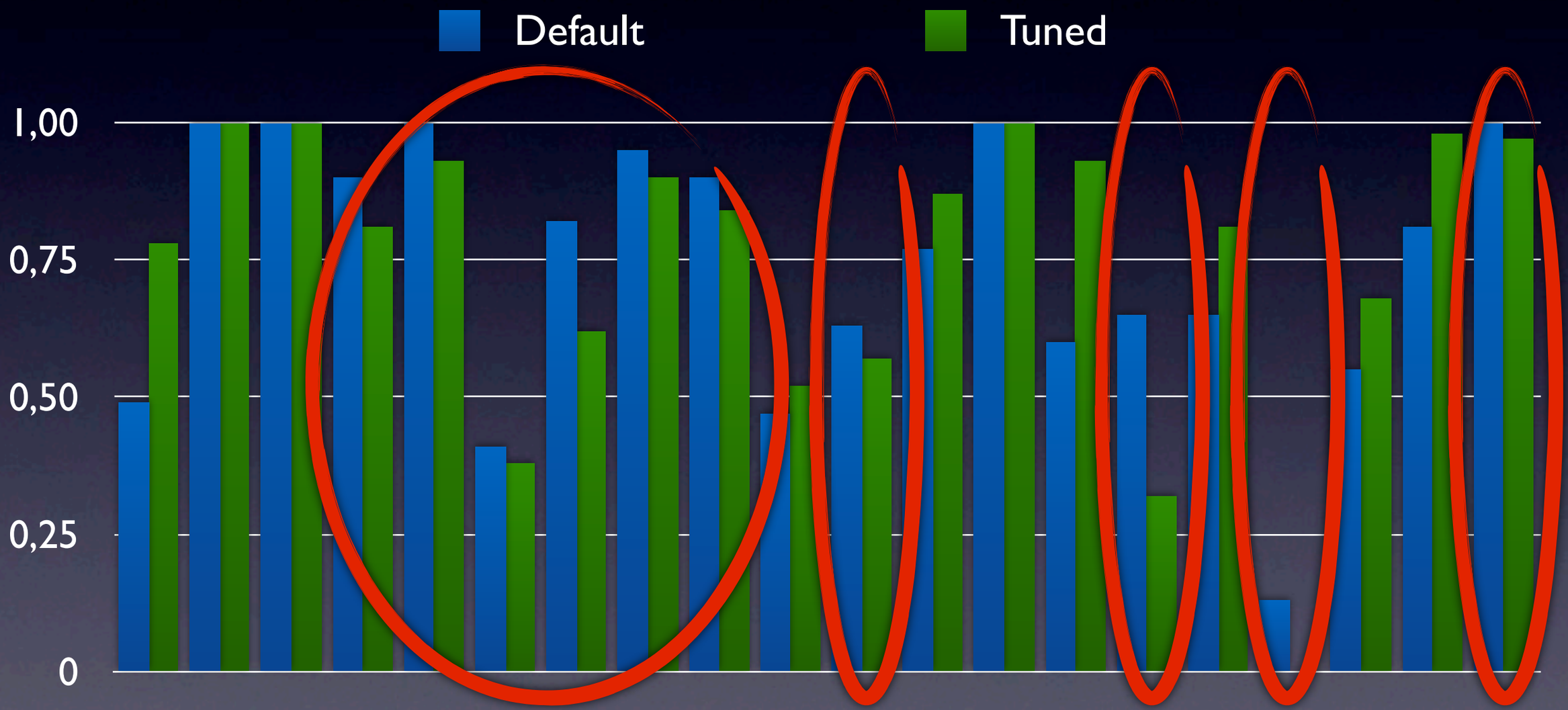


Branch Coverage

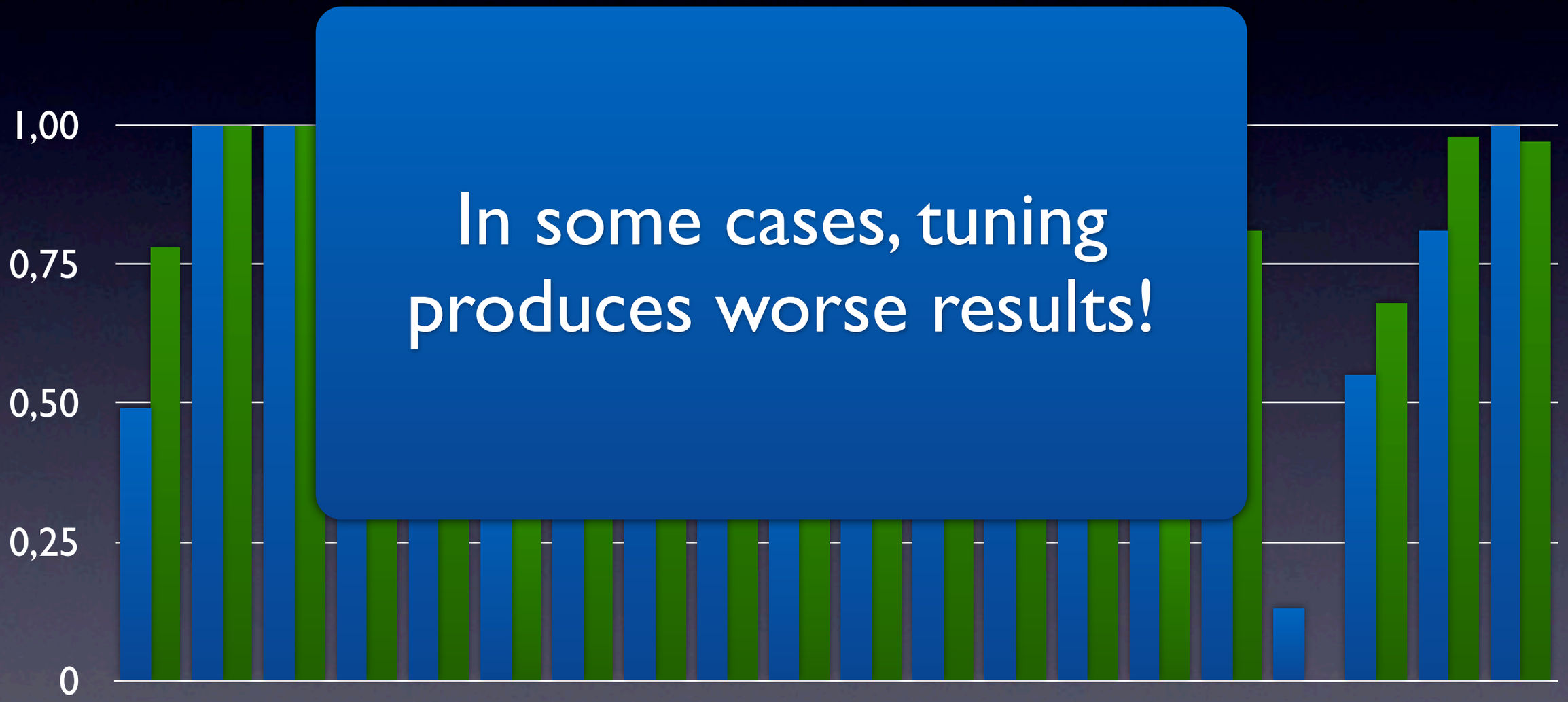
Branch Coverage



Branch Coverage



Branch Coverage



Tuning

Tuning

- Use very large problem set

Tuning

- Use very large problem set
- Evaluate all possible parameter combinations

Tuning

- Use very large problem set
- Evaluate all possible parameter combinations

Tuning

- Use very large problem set
- Evaluate all possible parameter combinations

Tuning

- Use very large problem set
- Evaluate all possible parameter combinations

Response Surface Methodology

Tuning

k-fold cross validation

- Use very large problem set
- Evaluate all possible parameter combinations

Response Surface Methodology

RQ4

What are the effects of the search budget on parameter tuning?

Branch Coverage

Worst Default Best

Branch Coverage

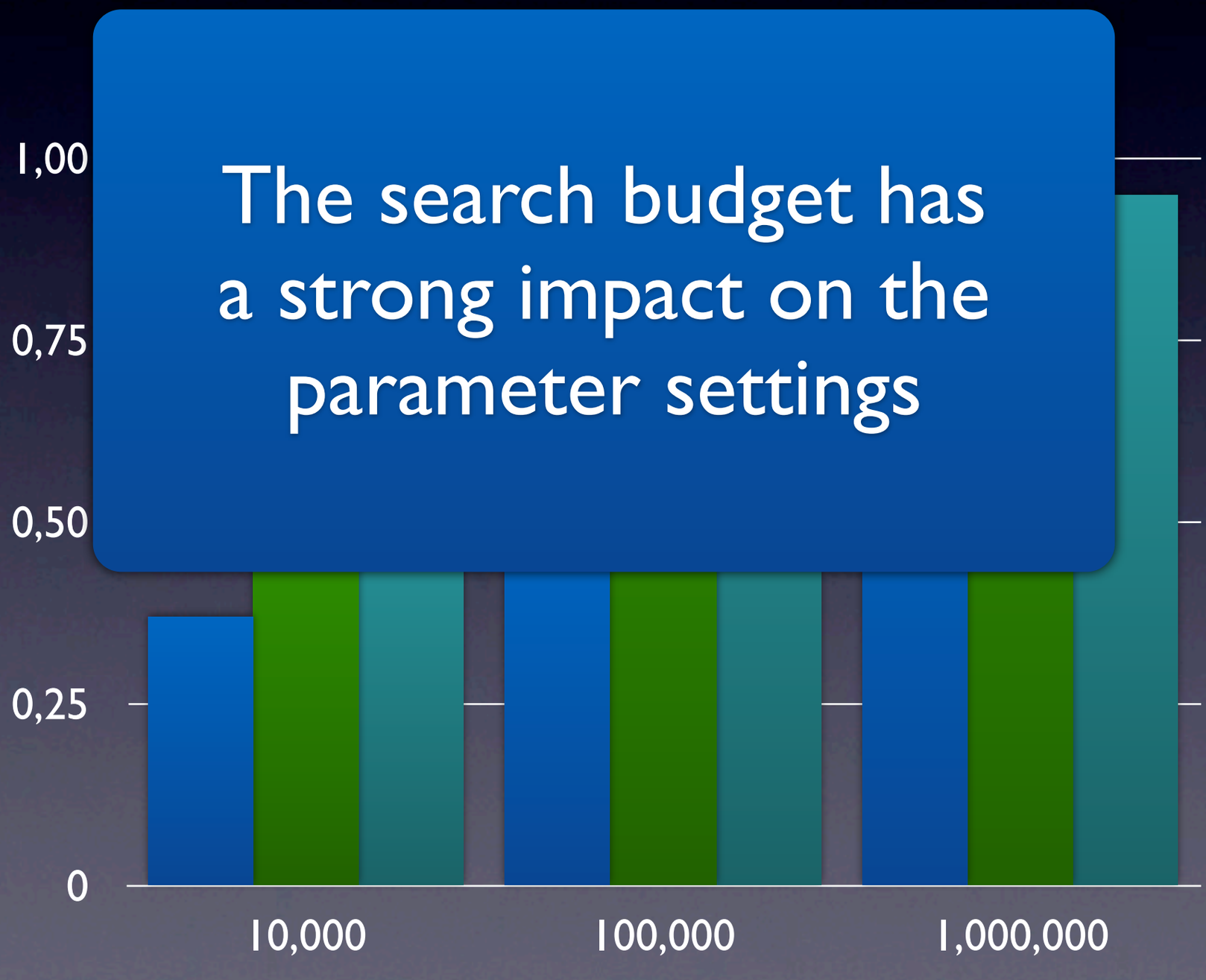
Branch Coverage

Branch Coverage

Branch Coverage

Branch Coverage

Branch Coverage



The search budget has a strong impact on the parameter settings

Branch Coverage

Branch Coverage

User

May not know about SBSE

Wants best performance on his problem

May only wish to set search duration

Tool Developer

Produces a search-based tool

Wants most pleasant experience for practitioner

Wants highest effectiveness on all possible problems

Researcher

Compares tools

Compares techniques

Performs empirical studies

Branch Coverage

User

May not know about SBSE

Wants best performance on his problem

May only wish to set search duration

Tool Developer

Produces a search-based tool

Wants most pleasant experience for practitioner

Wants highest effectiveness on all possible problems

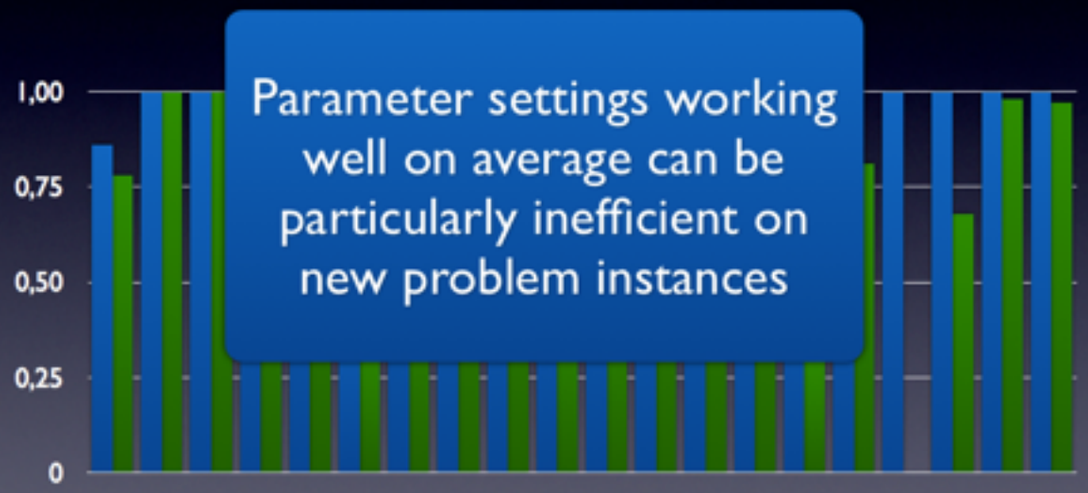
Researcher

Compares tools

Compares techniques

Performs empirical studies

Branch Coverage



Branch Coverage

User

- May not know about SBSE
- Wants best performance on his problem
- May only wish to set search duration

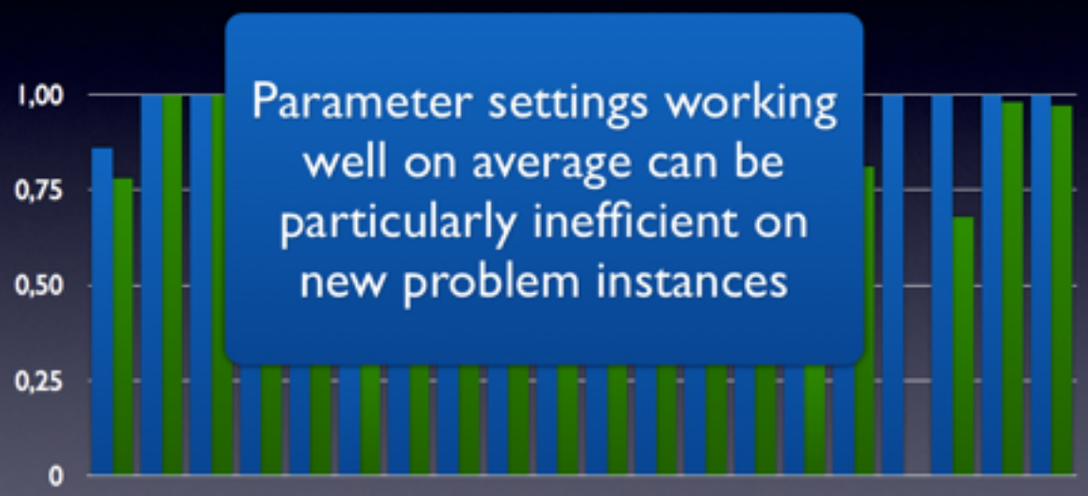
Tool Developer

- Produces a search-based tool
- Wants most pleasant experience for practitioner
- Wants highest effectiveness on all possible problems

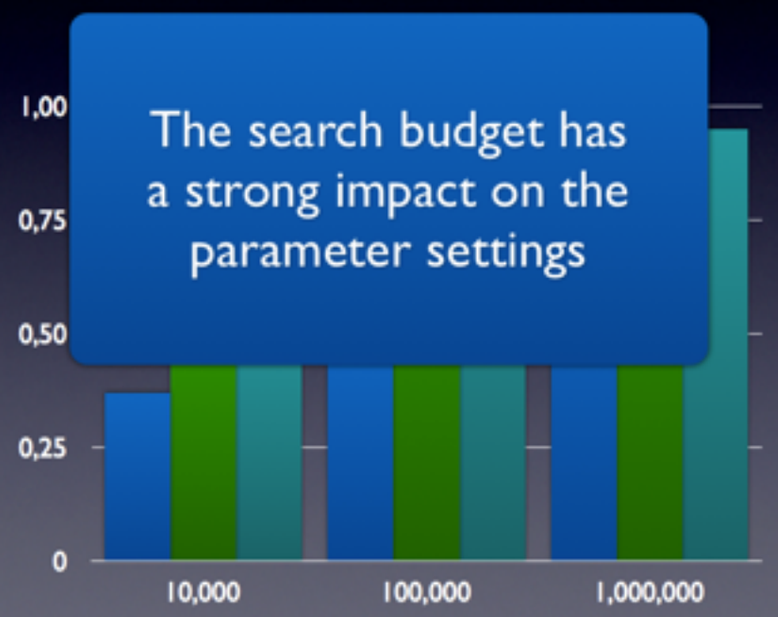
Researcher

- Compares tools
- Compares techniques
- Performs empirical studies

Branch Coverage



Branch Coverage



Branch Coverage

User

May not know about SBSE

Wants best performance on his problem

May only wish to set search duration

Tool Developer

Produces a search-based tool

Wants most pleasant experience for practitioner

Wants highest effectiveness on all possible problems

Researcher

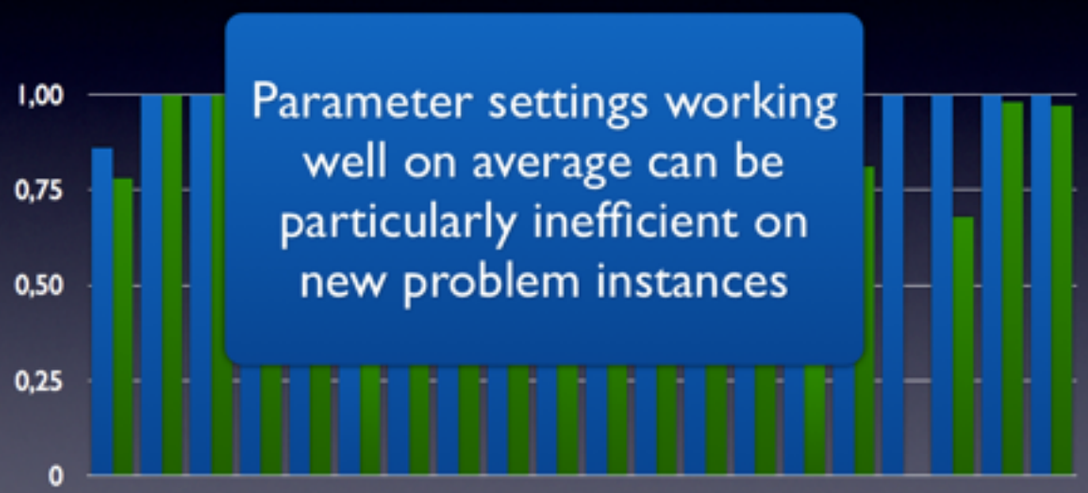
Compares tools

Compares techniques

Performs empirical studies

www.evosuite.org

Branch Coverage



Branch Coverage