

Detecting Performance-Relevant Changes in Configurable Software Projects



Sebastian
Böhm



Florian
Sattler



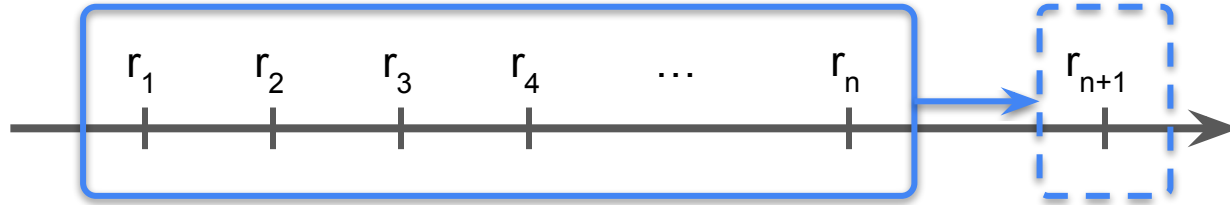
Sven
Apel



UNIVERSITÄT
DES
SAARLANDES



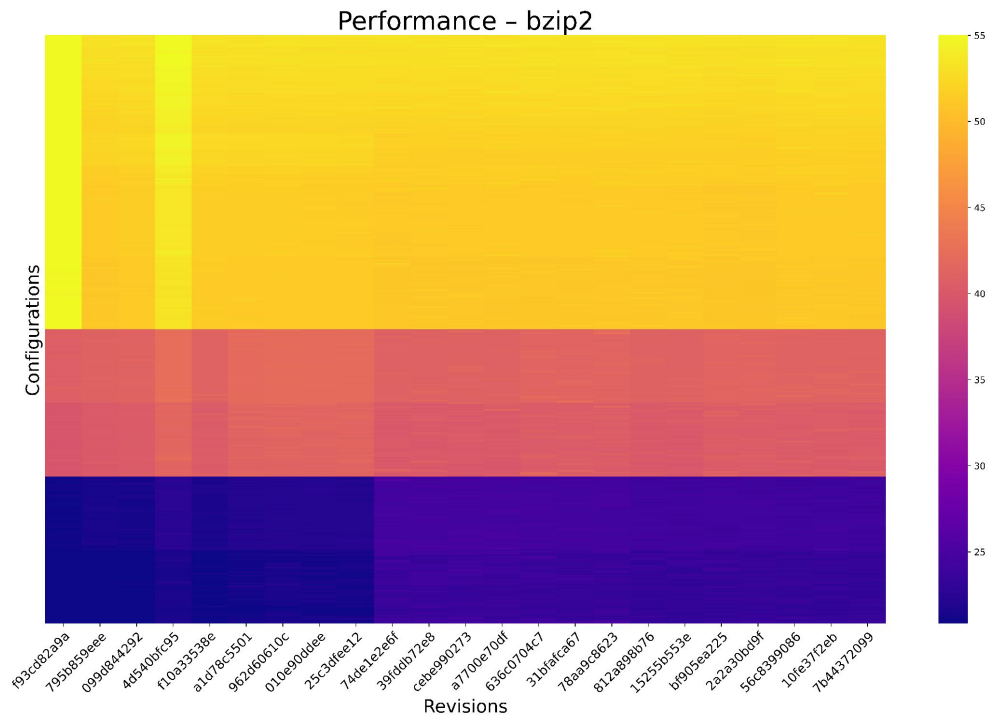
Performance Regression Analysis



Challenge: Configurability

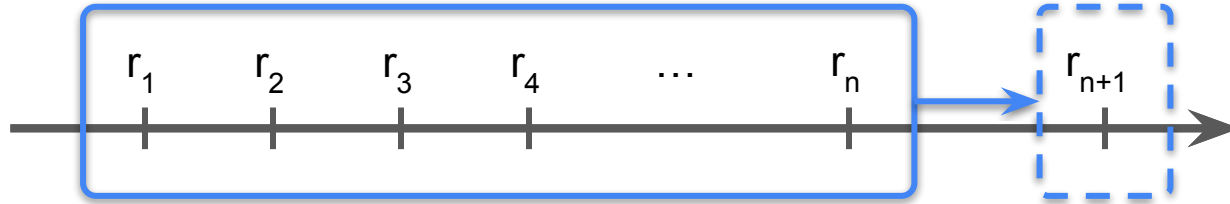
bzip2

- ❖ 23 revisions
- ❖ 576 configurations
- ❖ ca. 155h analysis time
(ca. 7h per revision)
- ❖ only one workload



Performance Regression Analysis of CSS

Goal: Reduce costs of regression analysis as much as possible.



- ❖ Reduction in the time dimension
 - Only analyze revision if change could affect performance at all
- ❖ Reduction in the space dimension
 - Only consider affected portion of the configuration space for analysis

Approach

Step 1 (time dimension):

- ❖ Does a given *change* interact with *performance-relevant code*?
- ❖ Performance-relevant interaction
 - Data-flow based change-impact analysis

```
1  int getIterations(int quality, bool strong) {  
2  }  
3  int iterations = quality * 2;  
4  if (strong) { iterations *= iterations; }  
5  return iterations;  
6  }  
7  void compress(vector<int> data, int iterations) {  
8  }  
9  for (int j = 0; j < iterations; ++j) {  
10 // complex computation on data  
11 }  
12 }  
13 void doStuff(vector<int> data, int quality,  
14             bool strong, /* StrongCompression */  
15             bool verbose /* Verbosity */) {  
16     int iterations = getIterations(quality, strong);  
17     if (verbose) { cout << "Iterations: " << iterations; }  
18     compress(data, iterations);  
19 }
```

Annotations:

- A callout box at the top right contains:
 - int iterations = quality;
 - + int iterations = quality * 2;
- A red double-headed arrow and a red question mark are positioned to the right of the `compress` function, indicating a question about its interaction with the `getIterations` function.

Approach

Step 2 (space dimension):

- ❖ What *features* participate in performance-relevant interactions?
 - Find features on the data-flow paths of a performance-relevant interaction
- ❖ Use information to restrict configuration-space
 - e.g., guide sampling approaches to focus on specific configurations

```
1  int getIterations(int quality, bool strong) {  
2  int iterations = quality * 2;  
3  if (strong) { iterations *= iterations; }  
4  return iterations;  
5  }  
6  
7  void compress(vector<int> data, int iterations) {  
8  for (int j = 0; j < iterations; ++j) {  
9  // complex computation on data  
10 }  
11 }  
12  
13 void doStuff(vector<int> data, int quality,  
14             bool strong, /* StrongCompression */  
15             bool verbose /* Verbosity */) {  
16 int iterations = getIterations(quality, strong);  
17 if (verbose) { cout << "Iterations: " << iterations; }  
18 compress(data, iterations);  
19 }
```

- int iterations = quality;
+ int iterations = quality * 2;

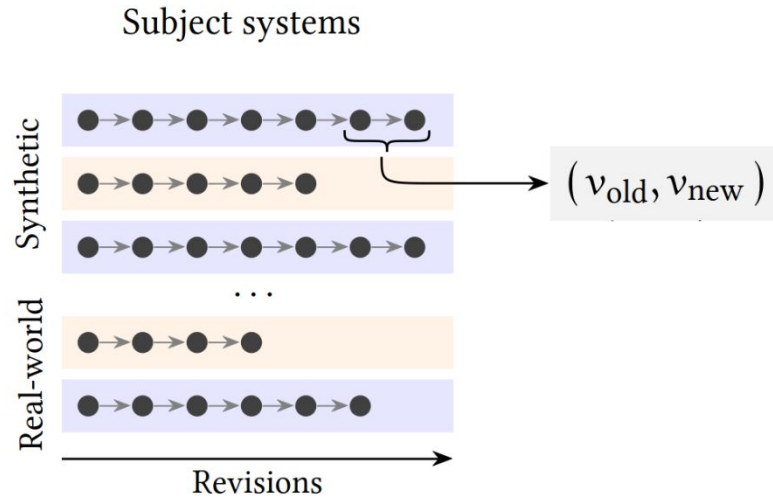
Research Questions

***RQ₁**: Do performance-relevant interactions indicate performance-relevant changes?*

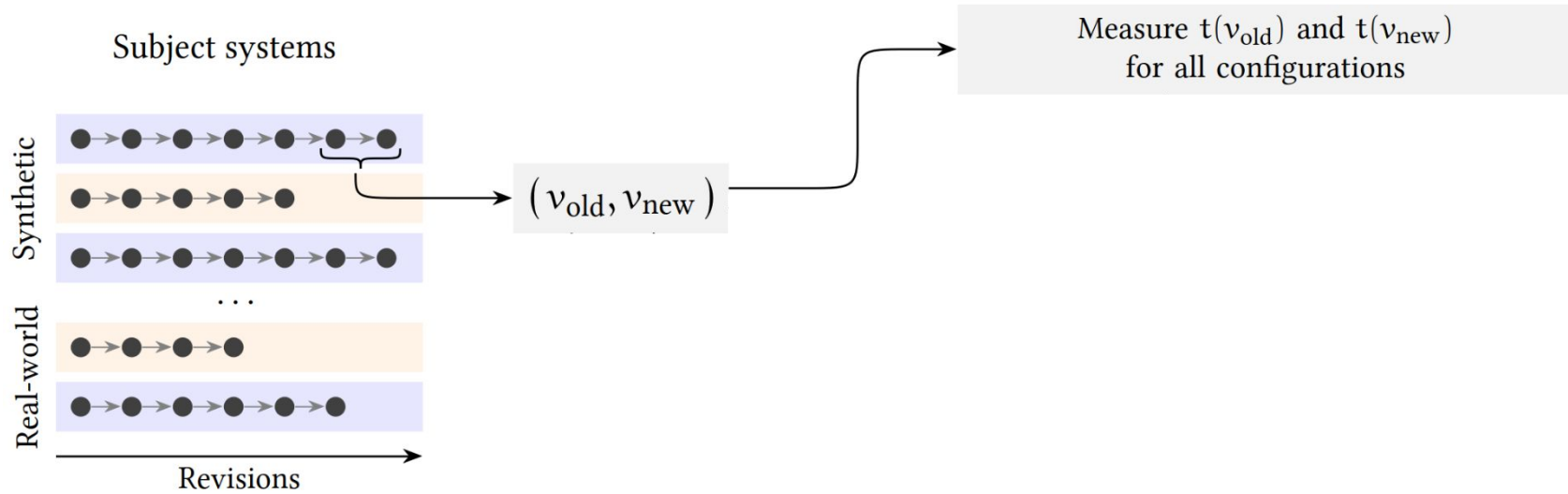
***RQ₂**: Does our approach correctly identify relevant features?*

***RQ₃**: How much analysis effort can be potentially saved with the information provided by our approach?*

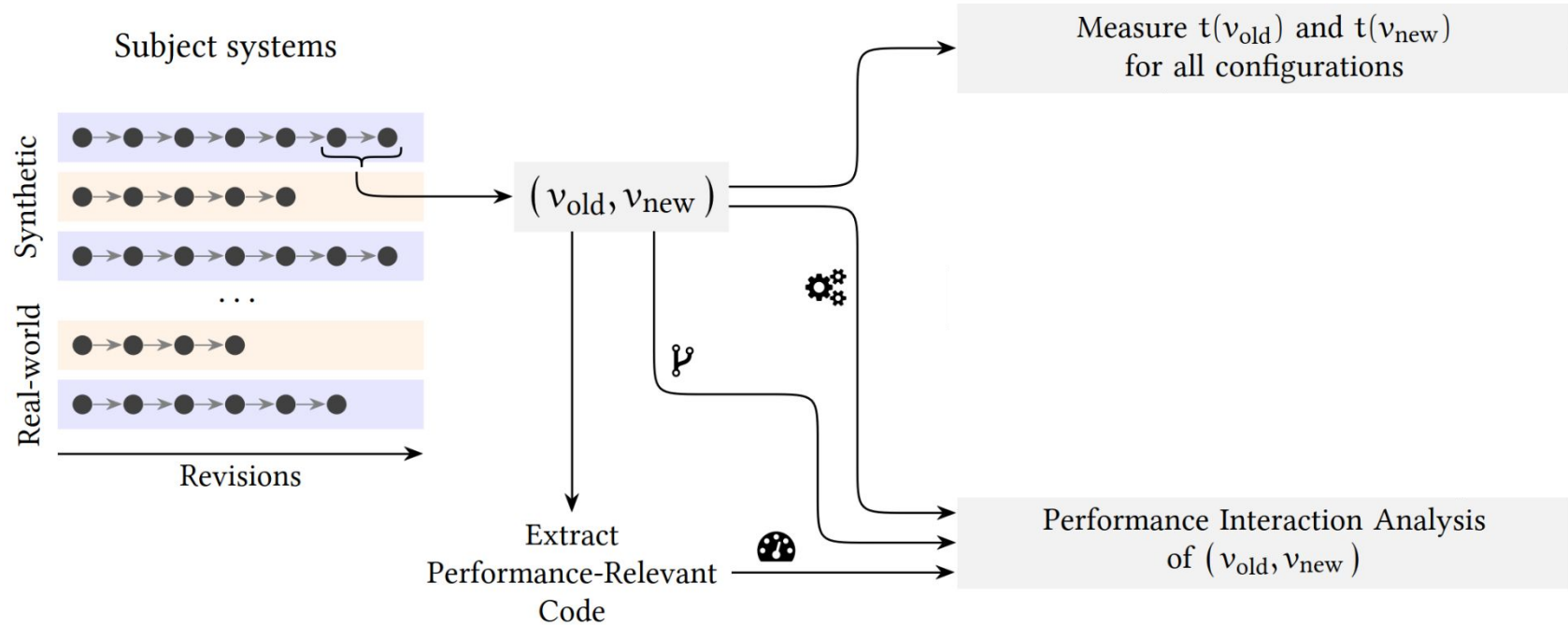
Experiment Setup



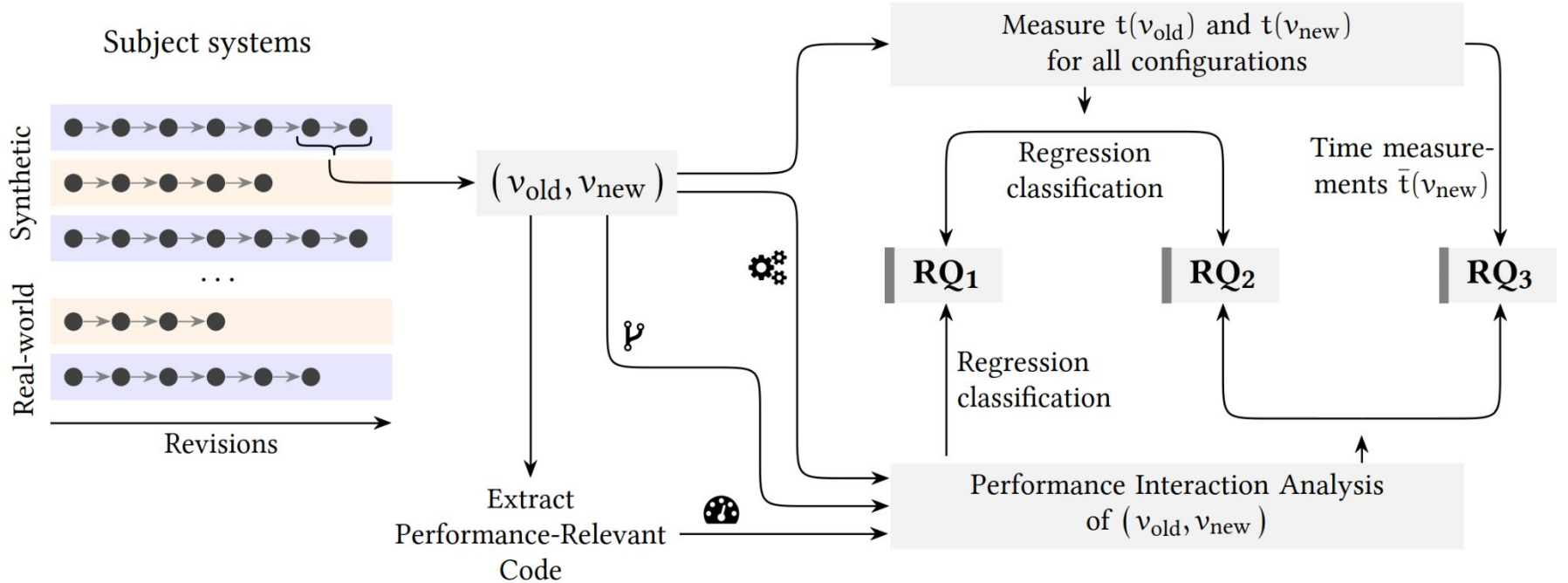
Experiment Setup



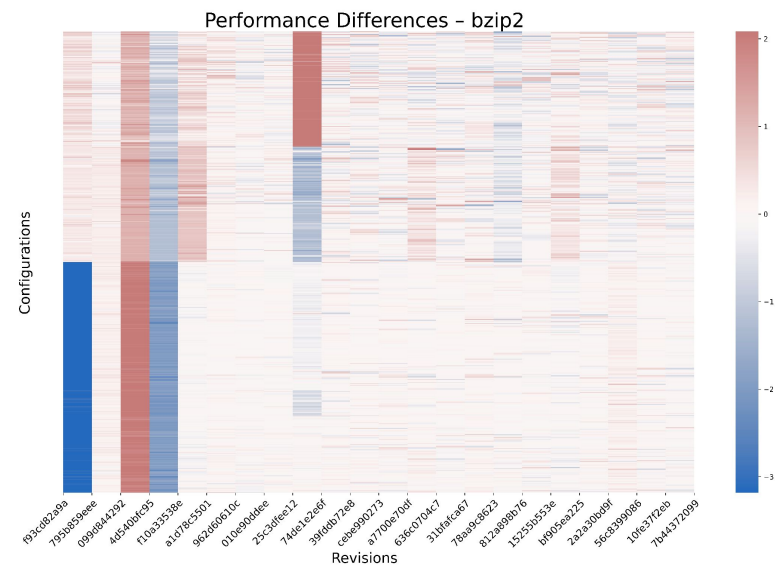
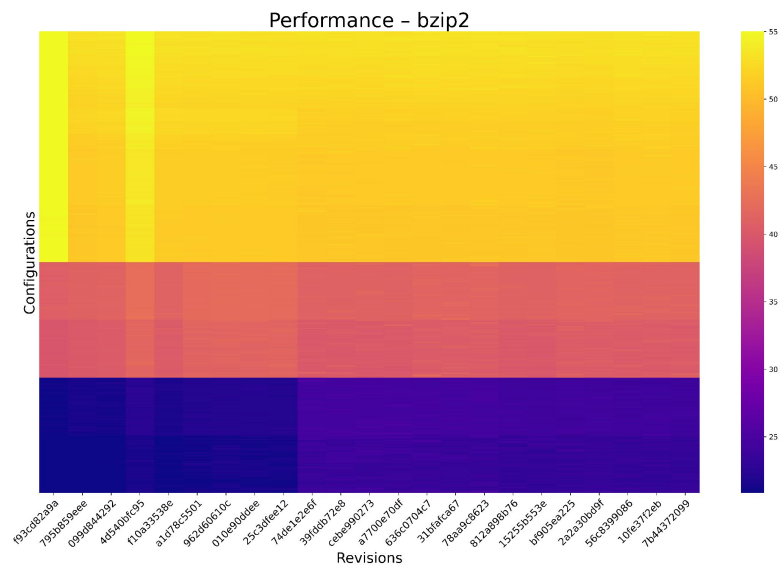
Experiment Setup



Experiment Setup



Change of Representation

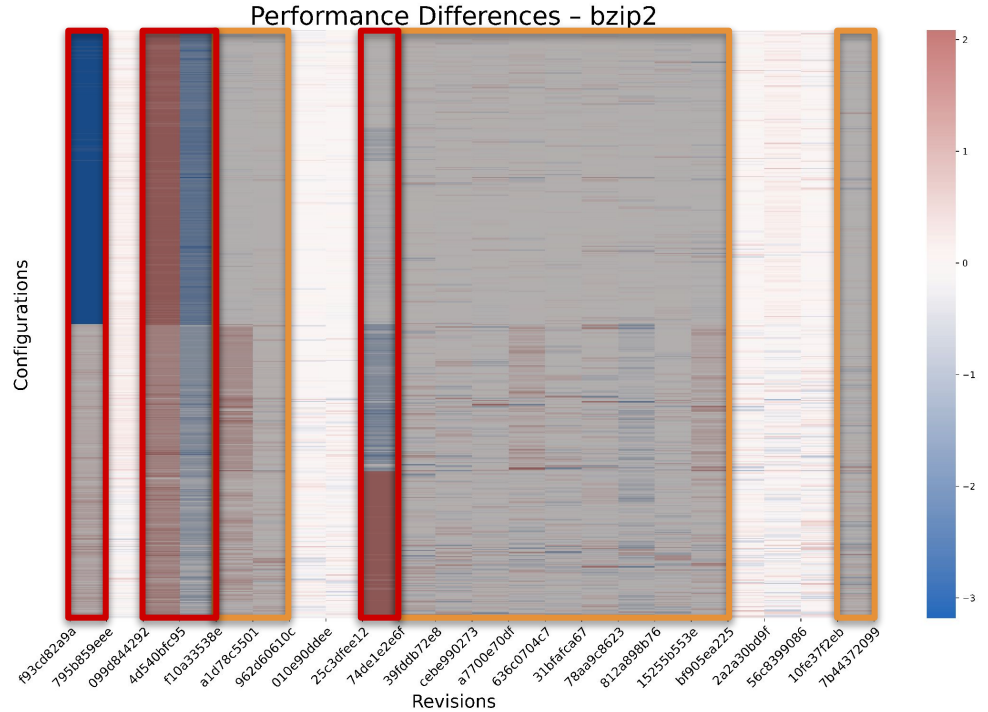


Baseline

bzip2

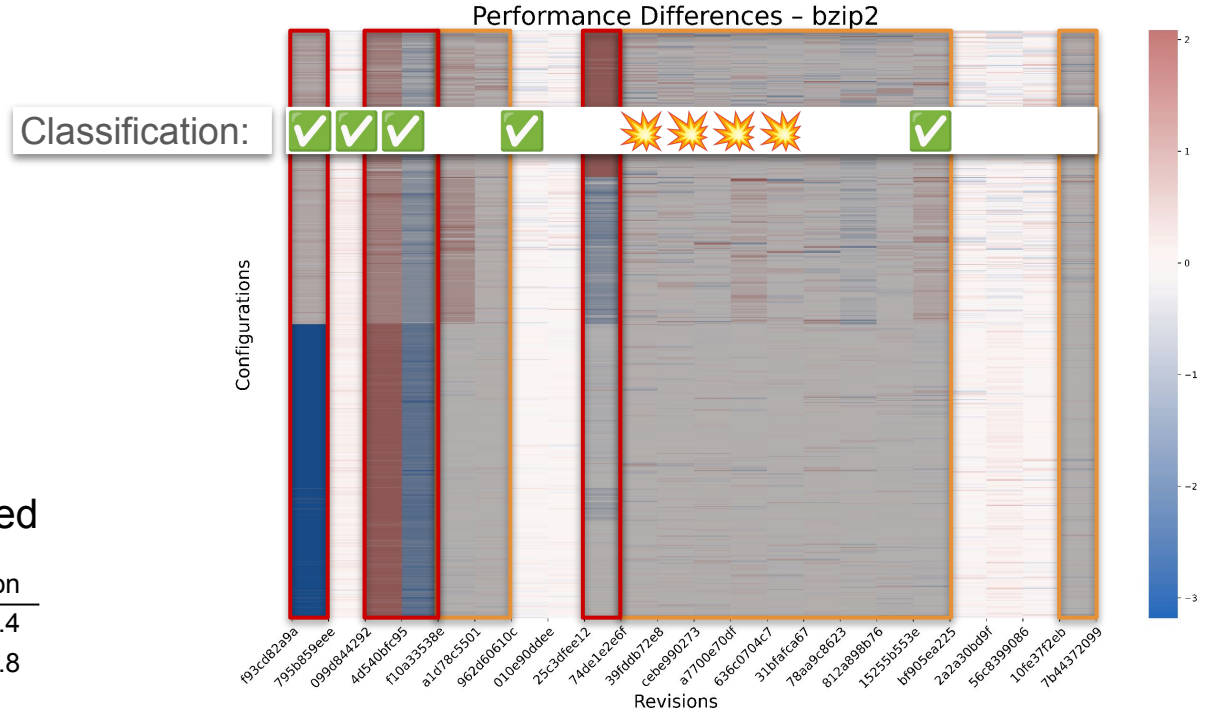
Baseline:

- ❖ regression if >0 configurations with diff >5% of runtime
- ❖ 4 “strong” regressions
- ❖ 12 “weak” regressions (possibly measurement noise)



RQ1: Detecting Regressions

bzip2



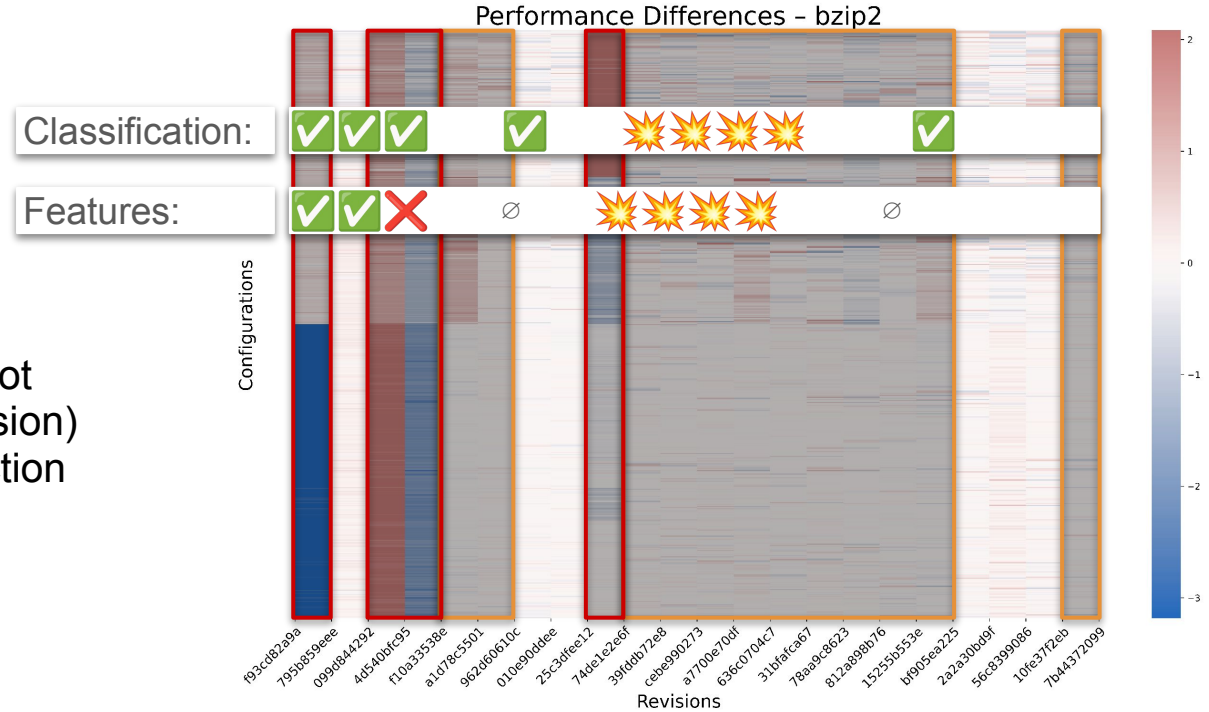
Results for bzip2:

- ❖ 5 revisions classified as potential regressions
- ❖ 4 analysis errors
- ❖ 2 strong regressions missed

	Recall	Precision
only strong regressions	0.50	0.4
all regressions	0.25	0.8

RQ2: Finding Relevant Features

bzip2



Results for bzip2:

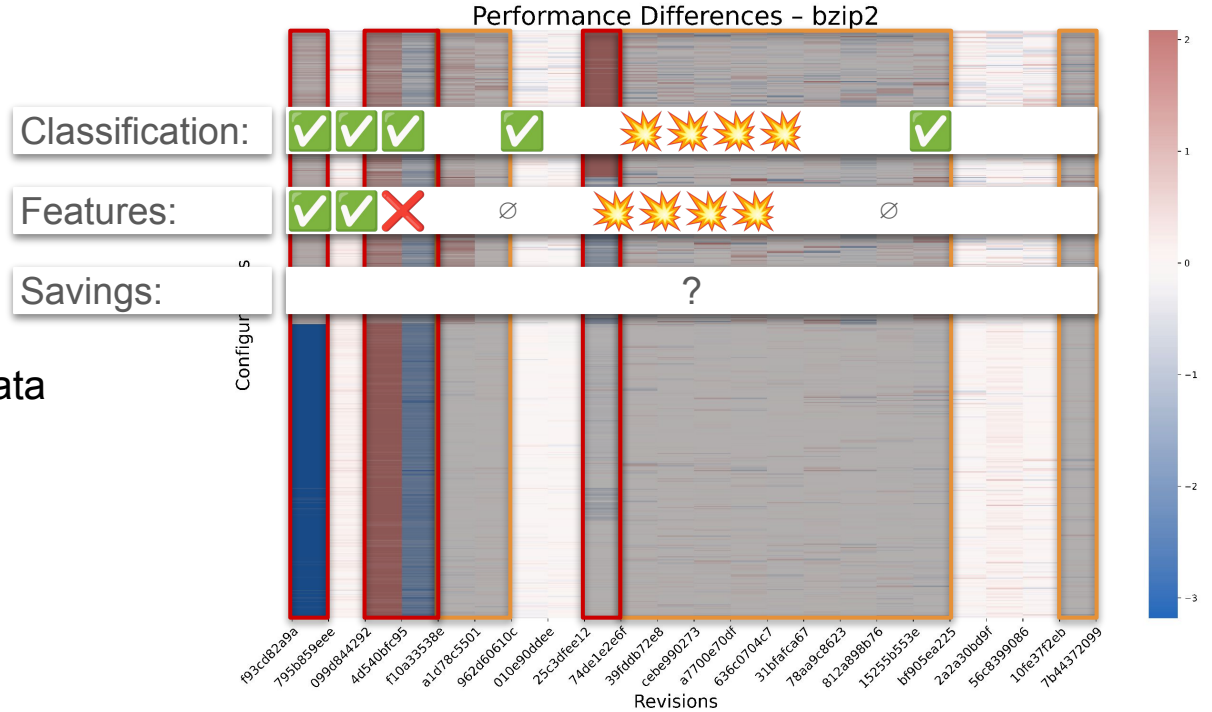
- ❖ Important features often not recognized (i.e., compression)
- ❖ Limitation of feature detection algorithm

RQ3: Potential Savings

bzip2

Results for bzip2:

- ❖ Not viable with available data



Open Questions




- ❖ RQ1: How should we classify regressions?
 - Threshold (absolute or relative)
 - Statistical test
 - Set of affected features is not empty (see below)
- ❖ RQ2: How to identify relevant features in the baseline?
 - Idea 1: Check if configurations in restricted space show regression
 - Idea 2: Use recursive random search¹ on diffs to find features with high influence

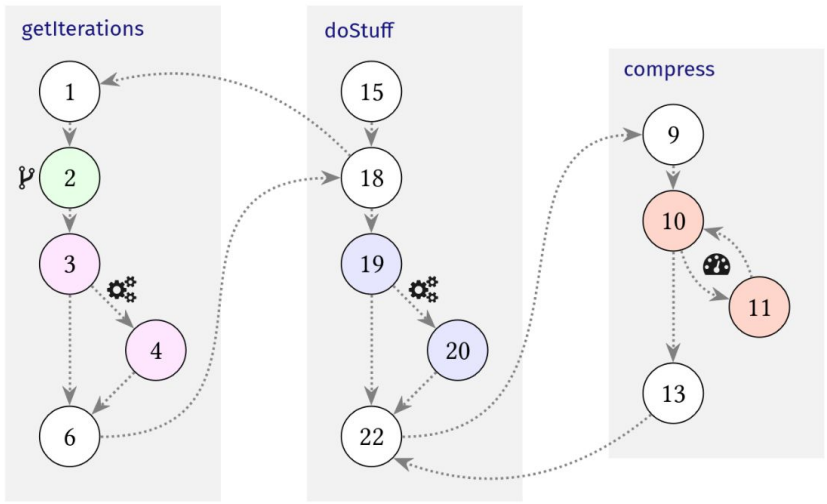
¹ Jeho Oh, Don Batory, and Rubén Heradio. 2023. Finding Near-optimal Configurations in Colossal Spaces with Statistical Guarantees. ACM Trans. Softw. Eng. Methodol. 33, 1 (2024), 7

Example

```
1 int getIterations(int quality, bool strong) {
2     int iterations = quality * 2;
3     if (strong) {
4         iterations *= iterations;
5     }
6     return iterations;
7 }
8
9 void compress(vector<int> data, int iterations) {
10    for (int j = 0; j < iterations; ++j) {
11        // complex computation on data
12    }
13 }
14
15 void doStuff(vector<int> data, int quality,
16             bool strong, /* StrongCompression */
17             bool verbose /* Verbosity */){
18     int iterations = getIterations(quality, strong);
19     if (verbose){
20         cout << "Iterations: " << iterations;
21     }
22     compress(data, iterations);
23 }
```

- int iterations = quality;
+ int iterations = quality * 2;

-  Code change
-  Feature code
-  Performance-relevant code



Example

```
1 int getIterations(int quality, bool strong) {
2   ✂ int iterations = quality * 2;
3   if (strong) {
4     ⚙ iterations *= iterations;
5   }
6   return iterations;
7 }
8
9 void compress(vector<int> data, int iterations) {
10  for (int j = 0; j < iterations; ++j) {
11    // complex computation on data
12  }
13 }
14
15 void doStuff(vector<int> data, int quality,
16             bool strong, /* StrongCompression */
17             bool verbose /* Verbosity */){
18   int iterations = getIterations(quality, strong);
19   if (verbose){
20     ⚙ cout << "Iterations: " << iterations;
21   }
22   compress(data, iterations);
23 }
```

- int iterations = quality;
+ int iterations = quality * 2;

- ✂ Code change
- ⚙ Feature code
- 🏠 Performance-relevant code

