

Figra: Evaluating a larger search space for Cardumen in Automatic Program Repair

Alcides Fonseca, Máximo Oliveira

Smaller search space

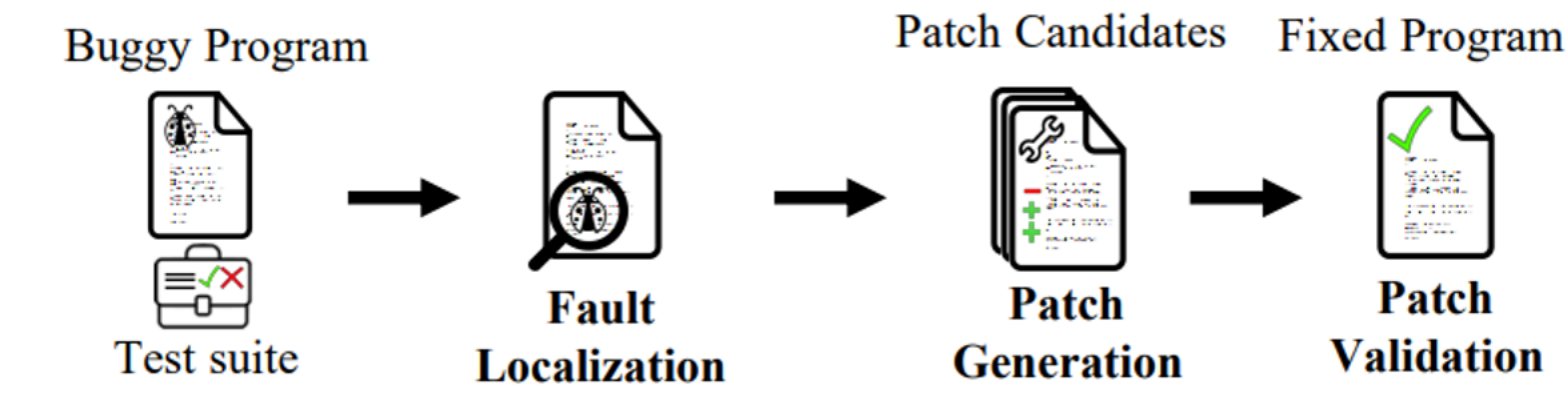
Larger search space



- **Faster exploration**
- **Fewer classes of bugs**

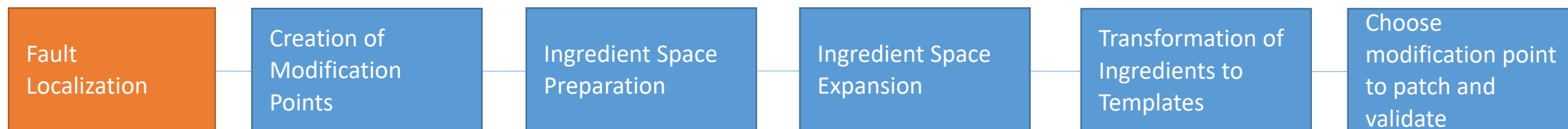
- **Longer or partial exploration**
- **More classes of bugs**

Q: Is a larger search space better?



- More candidate elements (from Imperfect Fault Localization)
- Finer ingredient granularity (e.g., expressions over statements)
- More ingredients (from other scopes, such as other files)
- More fix templates (with more diverse patches)

Figra: our approach



```
1 List<Integer> myMethod(List<Integer> arr) {  
2     int size = arr.size();  
3     int minimum = getNumberFromSomeMethod();  
4     if (size == 0) { suspiciousness = 0.95  
5         return arr;  
6     } else {  
7         int middle = arr.size() / 2;  
8         List<Integer> left = new ArrayList<Integer>(100);  
9         left.addAll(arr.subList(0,middle)); suspiciousness = 0.60  
10    ...  
}
```

Fault
Localization

Creation of
Modification
Points

Ingredient Space
Preparation

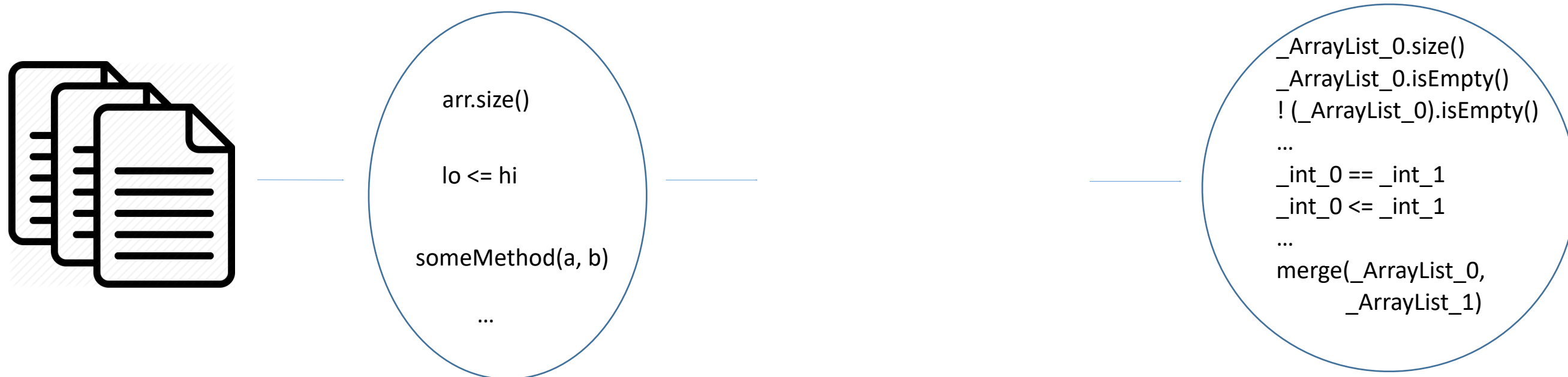
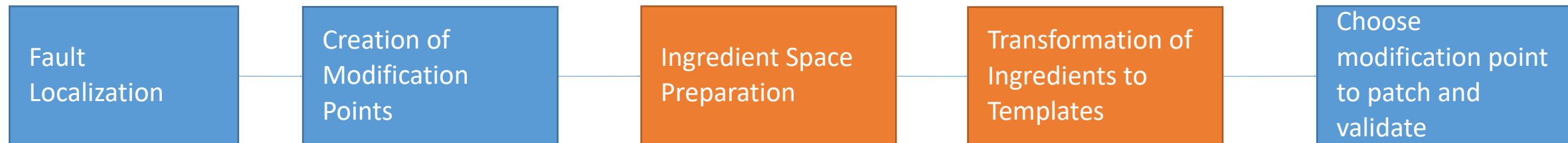
Ingredient Space
Expansion

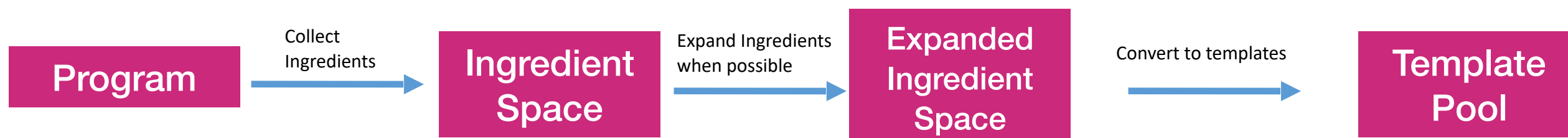
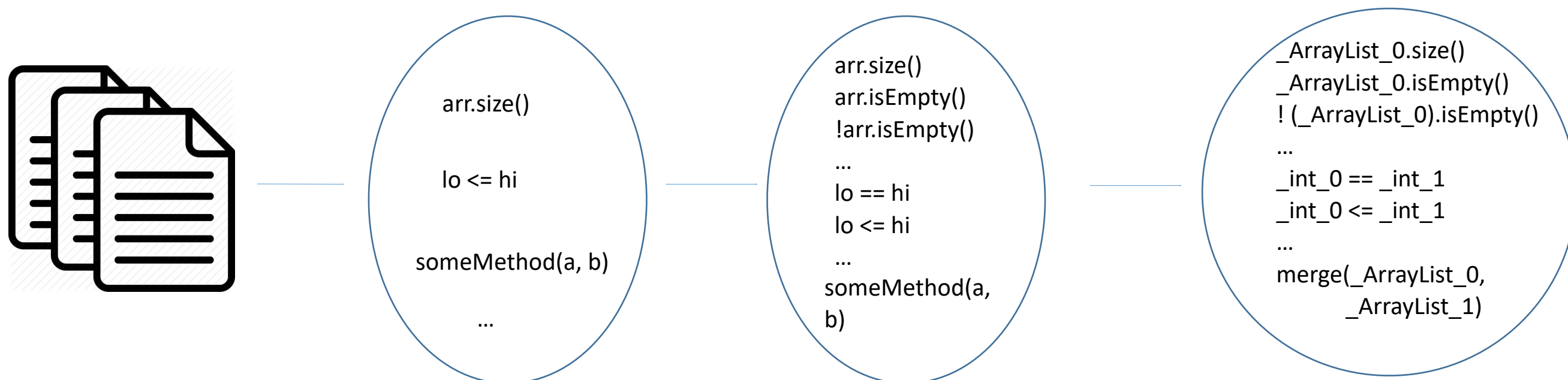
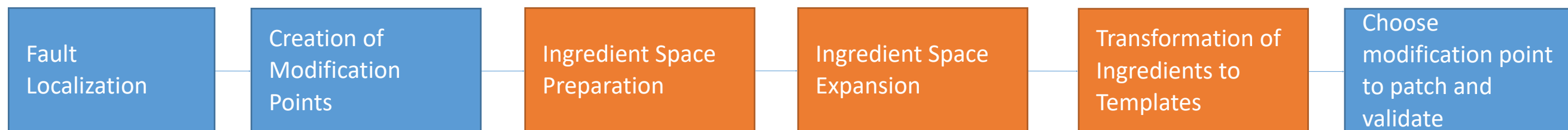
Transformation of
Ingredients to
Templates

Choose
modification point
to patch and
validate

```
1 List<Integer> myMethod(List<Integer> arr) {  
2     int size = arr.size();  
3     int minimum = getNumberFromSomeMethod();  
4     if (size == 0) { suspiciousness = 0.95  
5         return arr;  
6     } else {  
7         int middle = arr.size() / 2;  
8         List<Integer> left = new ArrayList<Integer>(100);  
9         left.addAll(arr.subList(0, middle)); suspiciousness = 0.60  
10        ...  
}
```

CARDUMEN APPROACH





- **Operator replacement**
 - Inspired by PAR and jMutRepair
 - (e.g., $\text{foo} < \text{bar} \Rightarrow \text{foo} = \text{bar}$, $\text{foo} \leq \text{bar}$, ...)
- **+1 and -1**
 - Off-by-one errors are a common class of bugs (Rice et al, 2017)
- **Method invocations**
 - All methods available in the context can be called in the fix.
 - (e.g., `isEmpty()`, even if not being invoked in the program)
- **!e, negation of boolean values**
 - Inspired by jMutRepair, but applicable outside branching conditions.
- **Permutation of arguments**
 - Another common class of bugs (Rice et al, 2017)
 - (e.g., $f(x, y)$ becomes $f(y, x)$)

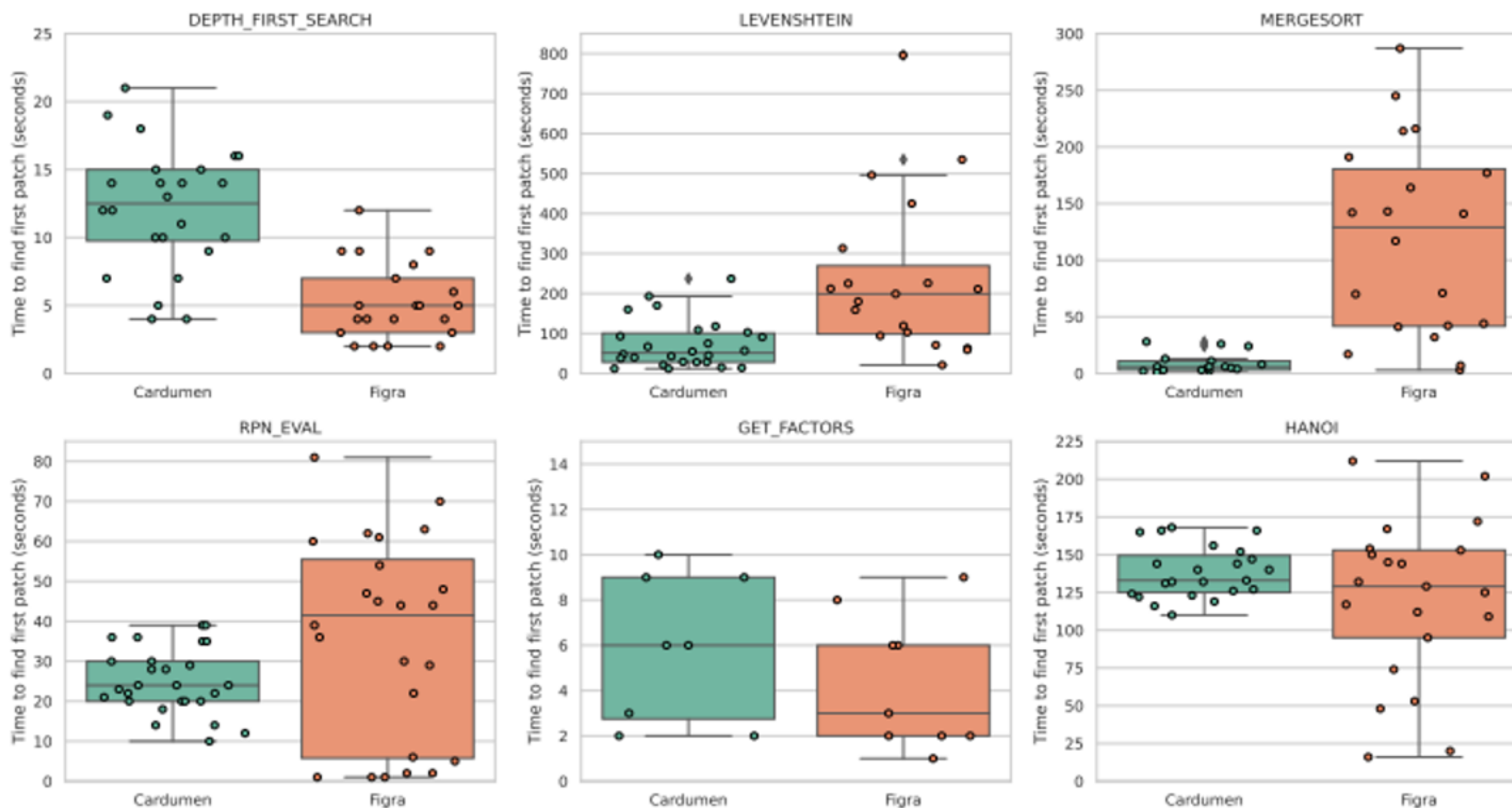
Evaluation

CAN FIGRA FIND PLAUSIBLE PATCHES?

Repair Tool	# Plausible Patches	# Patched Programs
JGenProg	164	4
Arja	113	4
RSRepair	31	3
NPEFix	9	2
Cardumen	5	5
Tibra	4	2
Nopol	4	4
JKali	3	2
JMutRepair	3	3
Dynamoth	2	2
JAID	NA	11
Cardumen (improved version)	16	6
Figra	82	18
Total	436 patches	27 patched programs

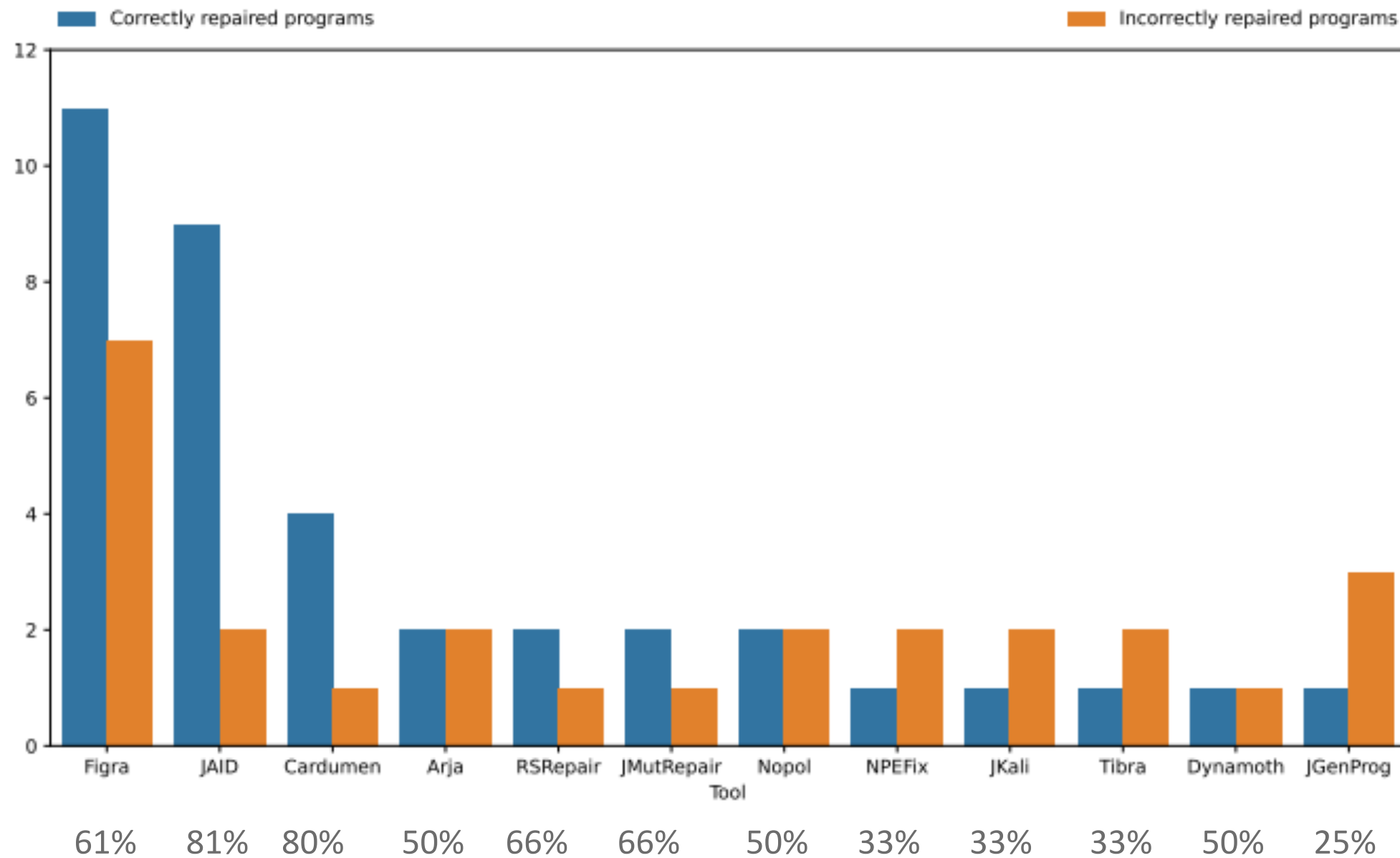
Java state-of-the-art APR tools over the QuixBugs benchmark.

How does having a larger ingredient space impact the time to find the first plausible patch?



Time taken to find the first plausible patch on 30 different seeds, for both Cardumen and Figra.

To what extent are the generated patches correct?



Can Figra outperform existing approaches?

Repair tool \ Benchmark	Benchmark	
	Defects4J (395)	IntroClassJava (297)
ARJA	86 (21%)	23 (7%)
GenProg-A	45 (11%)	18 (6%)
Kali-A	72 (18%)	5 (1%)
RSRepair-A	62 (15%)	22 (7%)
Cardumen	77 (19%)	0 (0%)
jGenProg	49 (12%)	4 (1%)
jKali	29 (7%)	5 (1%)
jMutRepair	23 (5%)	24 (8%)
Nopol	107 (27%)	32 (10%)
DynaMoth	74 (18%)	6 (2%)
JAID	94 (24%)	84 (28%)
NPEFix	9 (2%)	0 (0%)
Figra	17 (4%)	40 (13%)

Number of bugs patched by at least one test-adequate patch.

- Figra expands the search space through ingredient expansion (in comparison with Cardumen, that has a fine granularity)
- Plausible patches are not always found faster nor slower than in Cardumen.
 - More research is needed to understand what features impact performance.
- Figra can fix more bugs than Cardumen in QuixBugs and IntroClassJava, but not Defects4J.
 - More research is needed to understand what kind of programs take advantage of the larger search space.

Figra: Evaluating a larger search space for Cardumen in Automatic Program Repair

Alcides Fonseca, Máximo Oliveira
amfonseca@fc.ul.pt