LASIGE driven by excellence

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

Alcides Fonseca, Máximo Oliveira

FCT Fundação para a Ciência e a Tecnologia

Funded by FCT projects:
UIDB/00408/2020, UIDP/00408/2020
LISBOA-01-0247-FEDER-045915 and
EXPL/CCI-COM/1306/2021

Lisb@20 20 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Fix Search Space Trade-off

LASIGE driven by excellence

Smaller search space | **Larger search space**

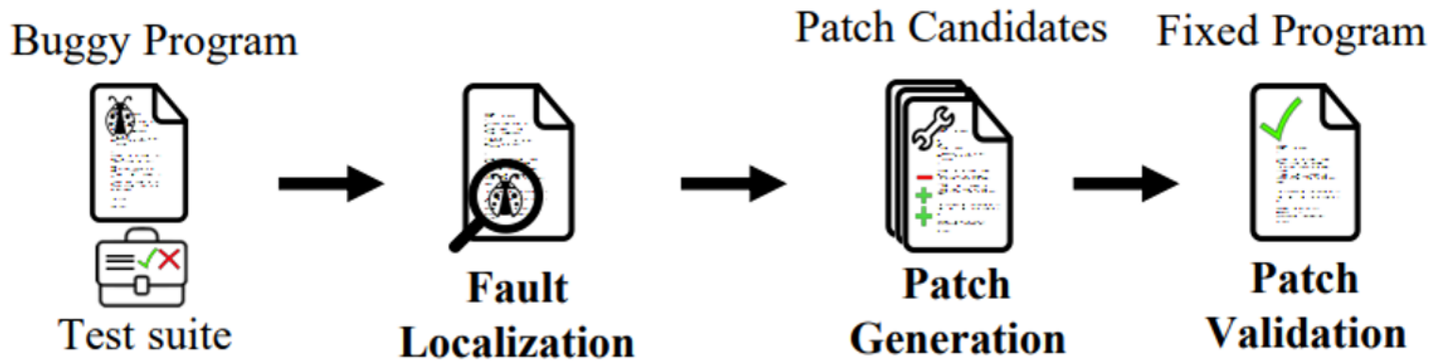
- **Faster exploration**
- **Fewer classes of bugs**

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

Q: Is a larger search space better?

Ciências ULisboa FCT Fundação para a Ciência e a Tecnologia

How to increase Search Space



- 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

FIGRA APPROACH

LASIGE driven by excellence

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       ...
```


Ciências ULisboa FCT Fundação para a Ciência e a Tecnologia

FIGRA APPROACH

LASIGE driven by excellence

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         ...
```


Ciências ULisboa FCT Fundação para a Ciência e a Tecnologia

CARDUMEN APPROACH

LASIGE driven by excellence

Fault Localization → Creation of Modification Points → Ingredient Space Preparation → Transformation of Ingredients to Templates → Choose modification point to patch and validate

arr.size()

lo <= hi

someMethod(a, b)

...

_ArrayList_0.size()
_ArrayList_0.isEmpty()
! (_ArrayList_0).isEmpty()
...
_int_0 == _int_1
_int_0 <= _int_1
...
merge(_ArrayList_0,
_ArrayList_1)

Program → Collect Ingredients → Ingredient Space → Convert to templates → Template Pool

Ciências ULisboa FCT Fundação para a Ciência e a Tecnologia

FIGRA APPROACH

LASIGE driven by excellence

Fault Localization → Creation of Modification Points → Ingredient Space Preparation → Ingredient Space Expansion → Transformation of Ingredients to Templates → Choose modification point to patch and validate

arr.size()

lo <= hi

someMethod(a, b)

...

arr.size()
arr.isEmpty()
!arr.isEmpty()
...
lo == hi
lo <= hi
...
someMethod(a, b)

_ArrayList_0.size()
_ArrayList_0.isEmpty()
! (_ArrayList_0).isEmpty()
...
_int_0 == _int_1
_int_0 <= _int_1
...
merge(_ArrayList_0, _ArrayList_1)

Program → (Collect Ingredients) → Ingredient Space → (Expand Ingredients when possible) → Expanded Ingredient Space → (Convert to templates) → Template Pool

Ciências ULisboa FCT Fundação para a Ciência e a Tecnologia

FIGRA EXTENDED TEMPLATES

LASIGE driven by excellence

- Operator replacement
  - Inspired by PAR and jMutRepair
  - (e.g., foo < bar => foo == bar, foo <= 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))

Ciências ULisboa FCT Fundação para a Ciência e a Tecnologia

Evaluation

Can Figra find plausible patches?

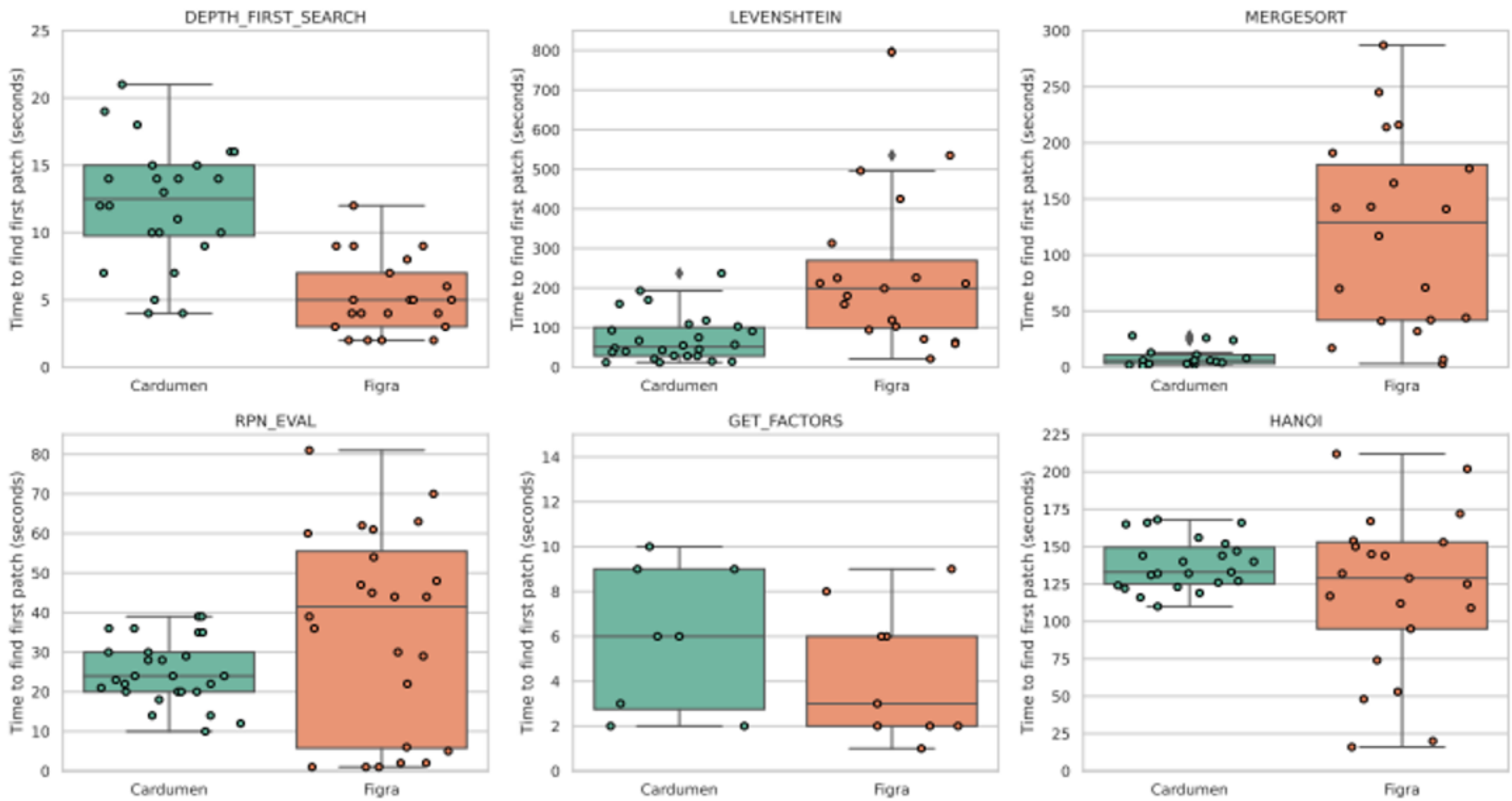
LASIGE driven by excellence

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.

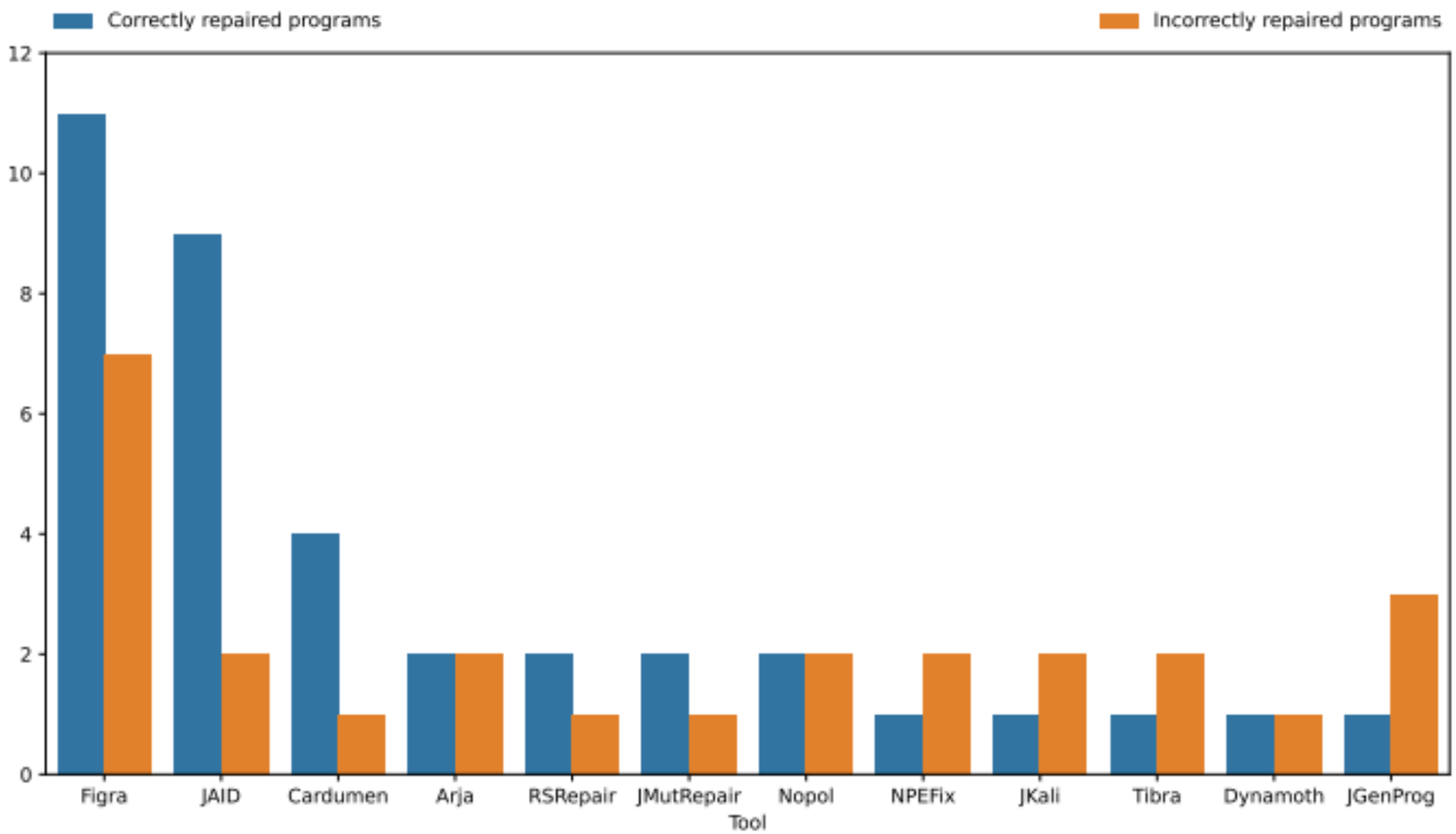
Ciências ULisboa FCT Fundação para a Ciência e a Tecnologia

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?



Figra	JAID	Cardumen	Arja	RSRepair	JMutRepair	Nopol	NPEFix	JKali	Tibra	Dynamoth	JGenProg
61%	81%	80%	50%	66%	66%	50%	33%	33%	33%	50%	25%

Can Figra outperform existing approaches?

Repair tool \ 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.

CONCLUSIONS

- 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.

LASIGE driven by excellence

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

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

FCT Fundação para a Ciência e a Tecnologia

Funded by FCT projects:
UIDB/00408/2020, UIDP/00408/2020
LISBOA-01-0247-FEDER-045915 and
EXPL/CCI-COM/1306/2021

Lisb@20 20 COMPETE 2020 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu