Nanjing University

Neural Program Repair:

Systems, Challenges and Solutions

Wenkang Zhong, Chuanyi Li, Jidong Ge, Bin Luo

State Key Laboratory for Novel Software Technology

Nanjing University

What is NPR (Neural Program Repair)?

- APR (Automated Program Repair) aims to fix bugs automatically.
- NPR is an emerging direction of APR that apply neural models.
- Generally, NPR frames APR as a bug-to-patch translation.

```
static Map<Object, Object> getRegistry()
{
        return REGISTRY.get() != null ?
        REGISTRY.get() :,Collections.<Objec
        t, Object>emptyMap();
}
```

→ NPR System →

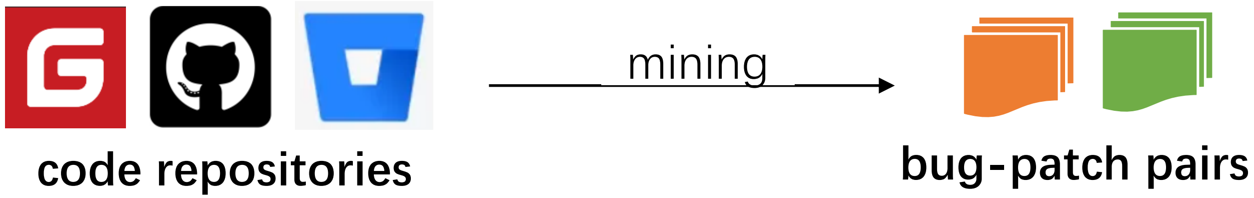
```
static Map<Object, Object> getRegistry()
{
        return REGISTRY.get();
}
```

end-to-end

Why focusing on NPR?

Recently, more and more researchers are paying attention to NPR.

- Advantages of NPR techniques
  - Remarkable performance
  - Accessible resources for training



However, understanding NPR systems is not easy.

  - Requires expertise in both APR and Deep Learning field

What we provide in this paper

- An in-detail review of previous NPR systems
- To make NPR systems more understandable,
  - decompose NPR systems into a 4-phase pipeline.
- To mine potential improvements,
  - analyze design choices on each phase.
  - identify three challenges, discuss the current solutions.

NPR Systems – Included Studies

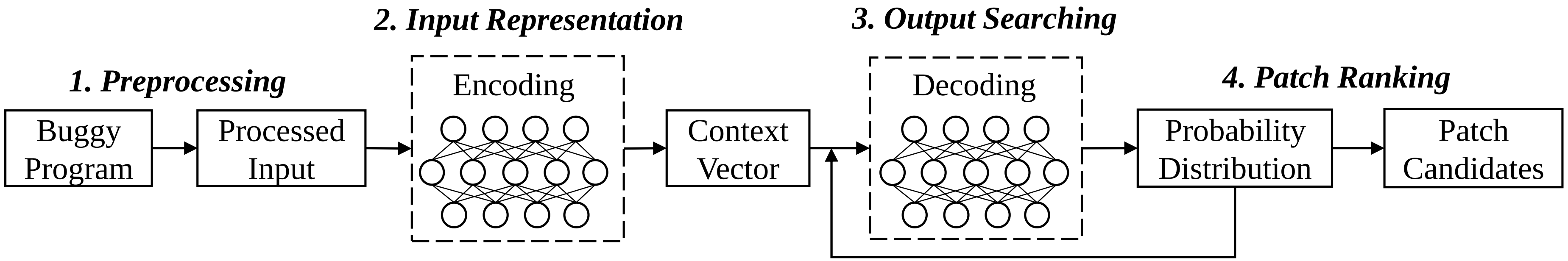
Time	System	Publication Channel	Evaluated Language
2020	ICSE	DLFix	Java
2021	ICSE	CURE	Java
2022	ICSE	RewardRepair	Java
2021	PMLR	TFix	JavaScript
2020	ICML	DrRepair	C, C++
2019	TOSEM	Tufano	Java
2019	TOSEM	CODIT	Java
2019	TSE	SequenceR	Java
2020	ICLR	Hoppity	JavaScript
2019	ICLR	Vasic	C#,python
2020	ASE	PatchEdits	Java
2020	ISSTA	CoCoNut	Java, C, Python
2021	MSR	CodeBERT-ft	Java
2021	ACL(Findings)	Grammar-Transformer	Java
2017	AAAI	DeepFix	C
2021	FSE	Recoder	Java

16 systems in total

Compile Error: 2
Common Error： 14

Java: 11
C: 3
JavaScript: 2
Python: 2
C#: 1
C++: 1

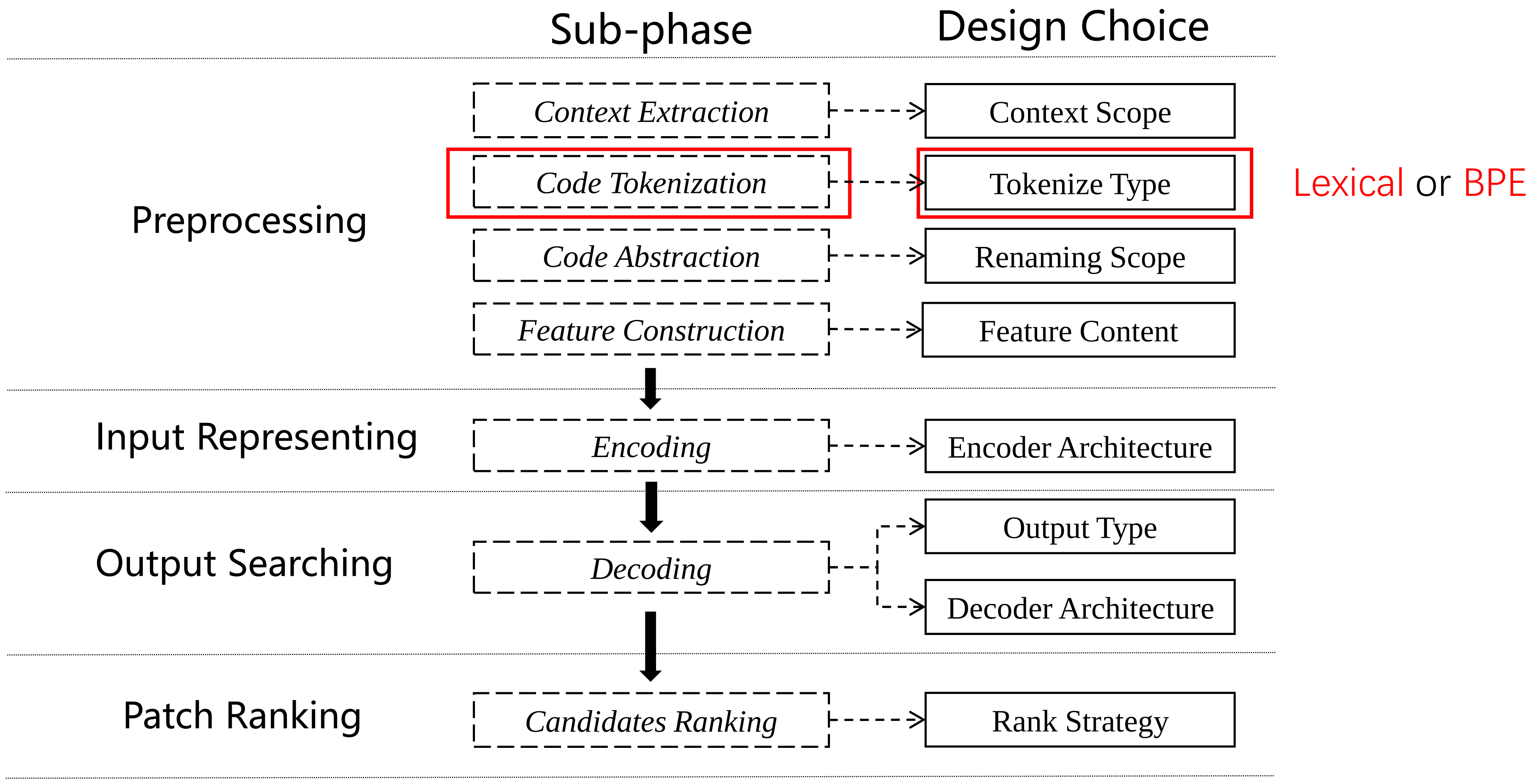
NPR Systems – Overall Procedure



Generally, NPR approaches can be decomposed into 4 phases:

- Preprocessing
  - transform original programs into forms that are acceptable by neural models
- Input Representation
  - encode processed input into vectors
- Output Searching
  - estimate the probability of patches
- Patch Ranking
  - reduce the size of candidates

NPR Systems – Design Space



NPR Systems – Summary of Design Choices

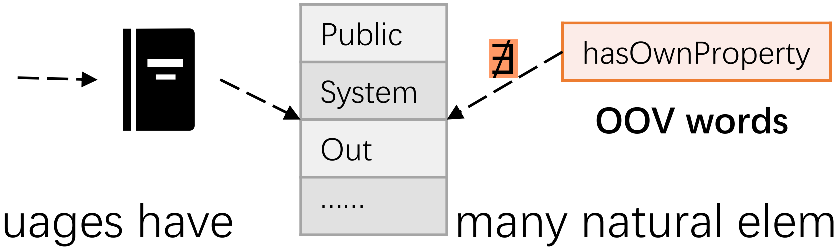
System	Context	Abstraction	Tokenization	Input	Encoder	Decoder	Output	Rank Strategy
CoCoNut	Method	Literal	Lexical+Camel	Code	FConv-context	FConv	Code	Beam Search
CODIT	Node Ancestor	\	Lexical	Code	BiLSTM	BiLSTM+copy	Code	Beam Search
				CFG Rule	BiLSTM	BiLSTM+copy	CFG Rule	
Cure	Method	Literal	Camel+BPE	Code	PT-GPT+Fconv-context	PT-GPT+Fconv	Code	Code-aware
CodeBERT	Node Ancestor	\	BPE	Code	CodeBERT	Transformer Dec.	Code	Beam Search
DeepFix	Method	\	Lexical	Code	GRU	GRU	Code	Beam Search
DLFix	Method	Literal	Lexical	AST	Tree-LSTM	Tree-LSTM	Node	DL-based
DrRepair	Method	\	Lexical	Code, NL	LSTM	LSTM+copy	Code	Beam Search
Hoppity	Method	\	Lexical	Graph	GNN	Edit Operator	Node Edit	Beam Search
PatchEdits	Line	\	BPE	Code	Transformer Enc.	Transformer Dec. +copy	Code Edit	Beam Search
Recoder	Method	Identifier	Lexical	Code, AST	Hybrid Reader	Modified TreeGen	Node Edit	Beam Search
RewardRepair	Class	\	BPE	Code	PT-T5	PT-T5	Code	Beam Search
Tufano	Method	Identifier,Literal	Lexical	Code	BiLSTM	BiLSTM	Code	Beam Search
SequenceR	Class	\	Lexical	Code	BiLSTM	BiLSTM+copy	Code	Beam Search
TFix	Neighbor Lines	\	BPE	Code, NL	PT-T5	PT-T5	Code	Beam Search
Tang	Method	Identifier	Lexical	Code	Transformer Enc.	Grammar Decoder	CFG Rule	Beam Search
		String		CFG Rule	Transformer Enc.			
Vasic	Method	\	Lexical	Code	LSTM+copy	Linear	Positon+Code	Beam Search

NPR Systems – Challenges

What are motivations of various design choices?

Limit use of code-related information → Compared with natural languages, programming languages have richer information, such as the **AST, Data Flow Graph, Control Flow Graph**……

The OOV problem → NPR systems use a pre-defined vocab



Programming Languages have many natural elements that can be named by programmers such identifiers and literals

Large search space → **Suppose**: a 5000-word vocab, a 15-word output
Search space: 5000^{15}

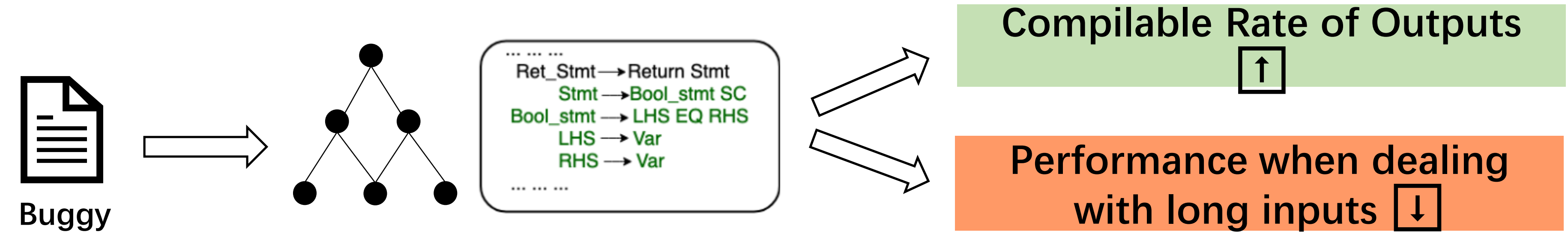
NPR Systems – Solutions

Challenges	Influenced phases	Current solutions
Limit use of code-related information	***Preprocessing***	add the context of the buggy program as inputs
		extract additional features
	Input Representing	use structural encoders (tree-based or graph-based)
The OOV problem	***Preprocessing***	renaming identifiers and literals
		use BPE or Camel-aware tokenization
	Output Searching	copy mechanism
Large search space	***Patch Ranking***	code-aware filter or DL-based filter

NPR Systems – Discussion of current solutions

Limit use of code-related information

Finding 1: The introduction of grammar rules is helpful for generating compilable patches.
Example: CODIT , Recoder, Tang
Limitation: Existing methods of introducing grammar rules are to model the input and output as CFG rules, not a human-like way.
Future direction: Let the model learn how to follow the syntax rules when outputting code tokens.

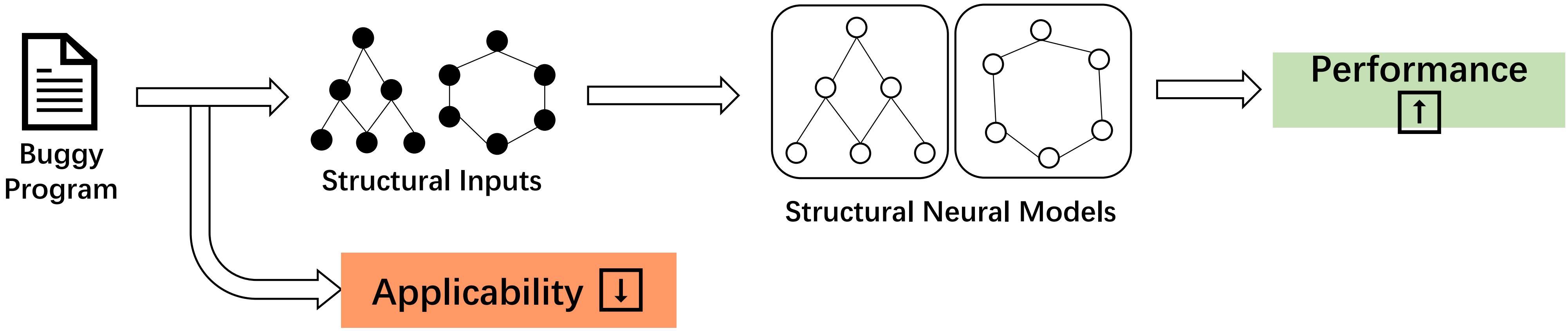


A short code text may produce a long sequence of CFG rules

NPR Systems – Discussion of current solutions

Limit use of code-related information

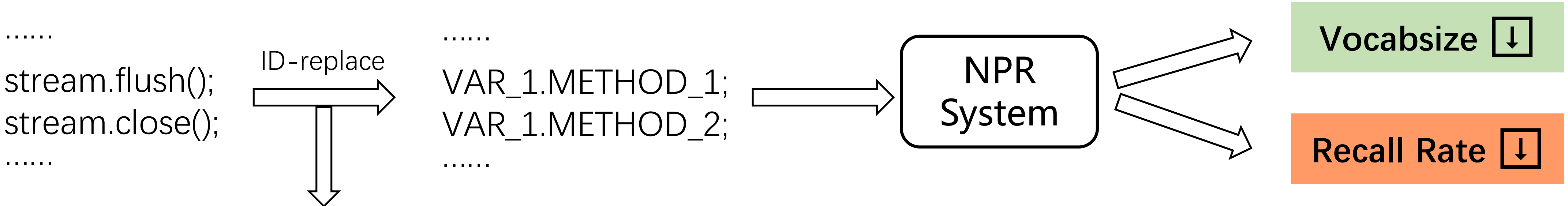
Finding 2: Structural models can be more precise at encoding structural inputs such as the AST.
Example: DLFix, Recoder, Hoppity
Limitation: Using structural models may decrease the applicability
Future direction: Investigating the performance differences between structured input and sequential input.



NPR Systems – Discussion of current solutions

The OOV problem

Finding 1: Abstraction of source programs can efficiently reduce the size of the vocabulary, thus mitigating the OOV problem.
Limitation: Abstraction of codes may decrease the recall rate of the NPR model.
Future Direction: More balanced abstraction methods



Identifier	ID
flush()	METHOD_1
close()	METHOD_2

Identifier-ID map

What if the NPR system outputs a METHOD_3 ?

NPR Systems – Discussion of current solutions

The OOV problem

Finding 2: BPE-based tokenization also works for mitigating the OOV problem.
Limitation: BPE produces long inputs and long outputs, which are not handled well by neural network models.
Future Direction: A combination of word-level tokenization and BPE

stream.flush(); $\xrightarrow{\text{BPE}}$ str ea m . flu sh ( ) ;

Unknown words $\boxed{\downarrow}$

Performance when dealing with long inputs $\boxed{\downarrow}$

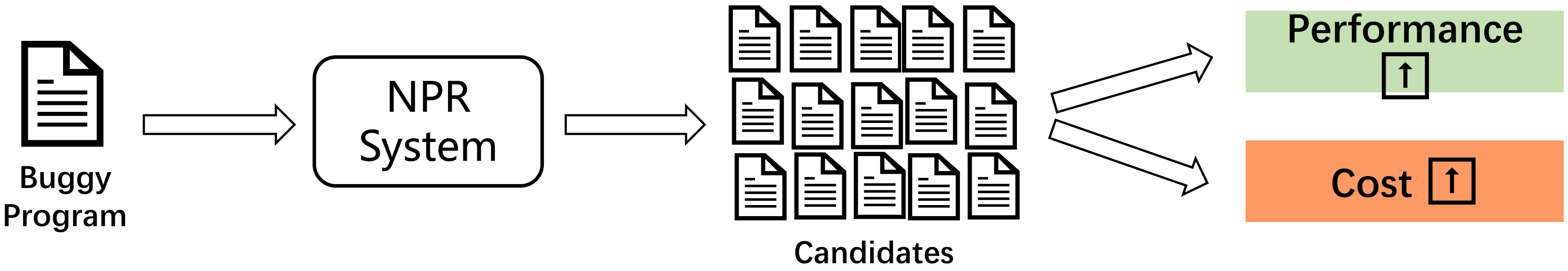
str ea m . flu sh ( ) ; **length: 9**

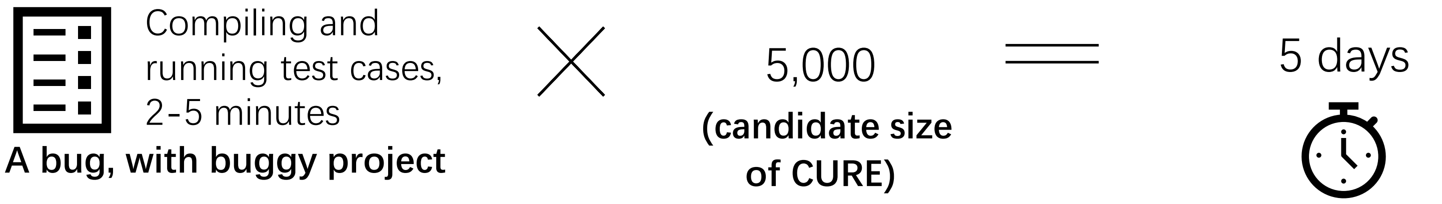
stream . flush ( ) ; **length: 6**

NPR Systems – Discussion of current solutions

Large search space

Finding 1: The number of candidates is not *the more, the better.*
Reason: Since NPR models are a kind of probability-estimation model, a larger candidate set will have a higher probability to contain a correct patch. However, the time and cost price of large candidate sets are usually ignored.
Future Direction: Investigating the performance-cost balance from an empirical perspective





Conclusion

- A decompose of previous NPR systems.
- An exploration of the design space.
- A summary of major challenges.
- Discussions of current solutions and possible improvements.

Future Work

- More rules when generating patches.
- Explicable NPR models.
- Multi-perspective evaluation.

- Thank you!

Q&A