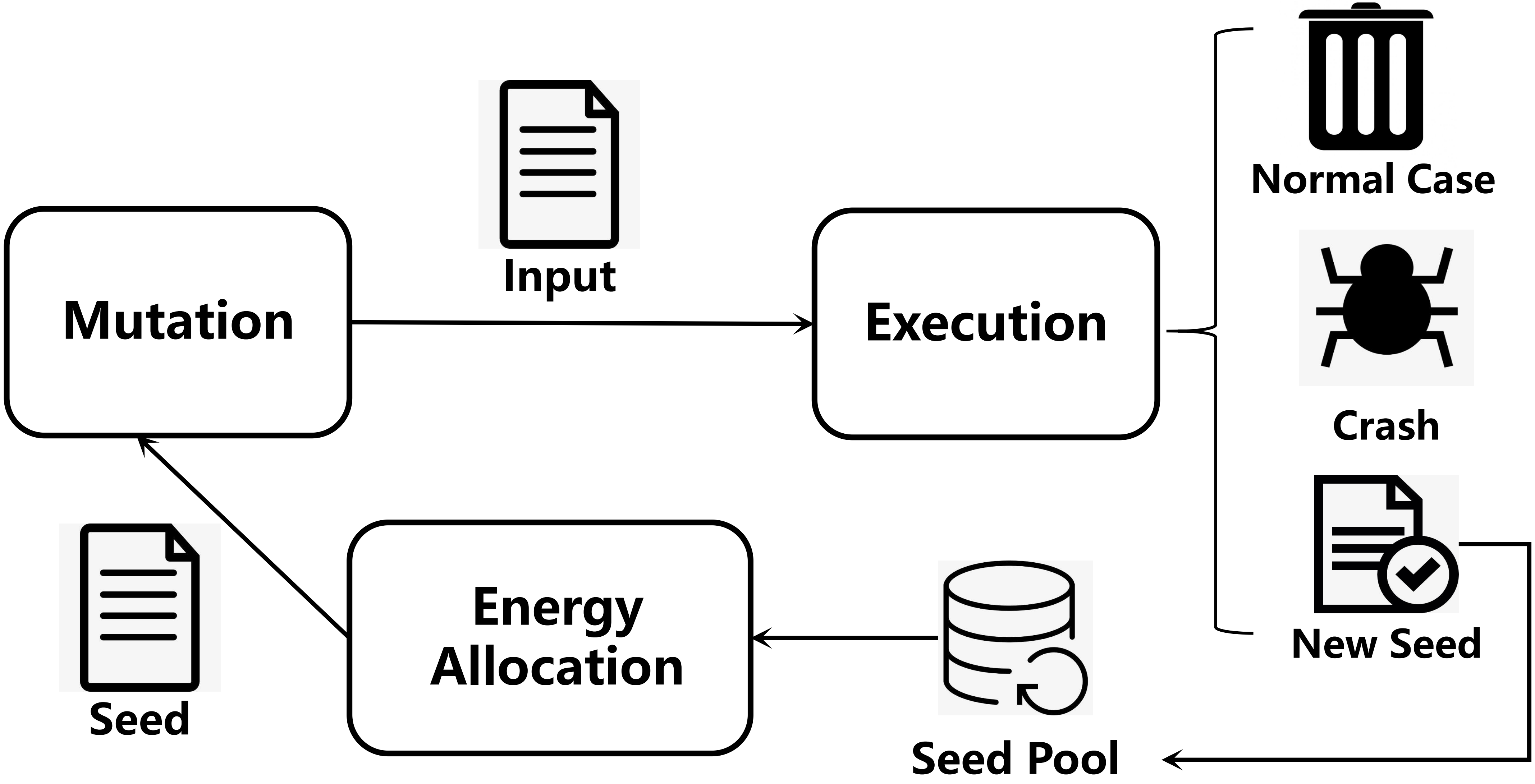
SLIME: Program-Sensitive Energy Allocation for Fuzzing

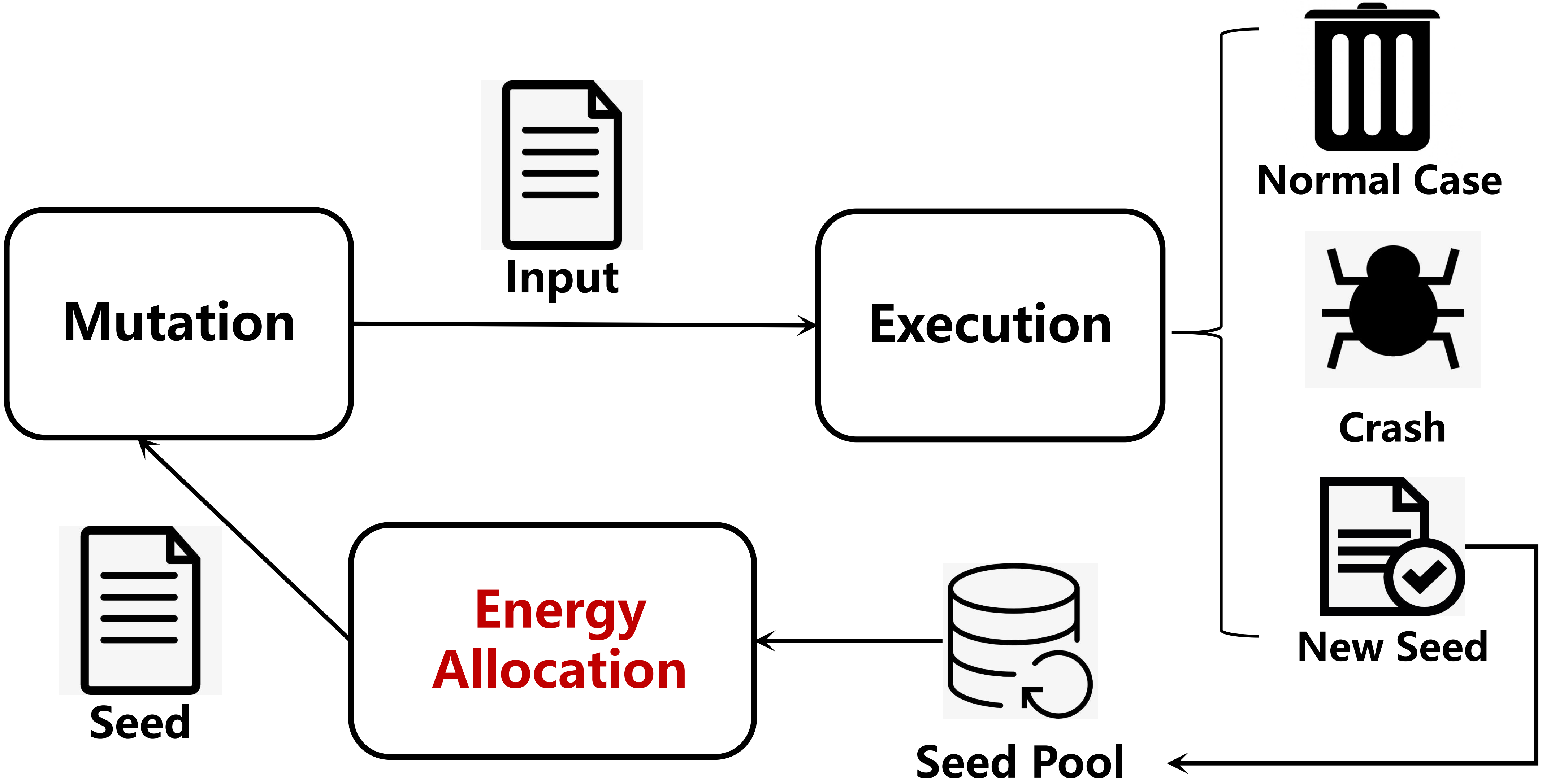
Chenyang Lyu Hong Liang Shouling Ji Xuhong Zhang Binbin Zhao
Meng Han Yun Li Zhe Wang Wenhai Wang Raheem Beyah

Fuzzing



Input
Mutation
Execution
Normal Case
Crash
New Seed
Seed
Energy Allocation
Seed Pool

Fuzzing



Mutation
Input
Execution
Normal Case
Crash
New Seed
Seed
Energy
Allocation
Seed Pool

Energy Allocation

- **Key properties to estimate the potential of seeds**
  - **AFL: seeds which have faster execution speed, trigger more edges and are discovered in the later time**
- **Corresponding algorithms based on key properties**
  - **EcoFuzz: adversarial multi-armed bandit model to allocate more energy to the seeds, which have higher transition probabilities with fewer execution number**

Energy Allocation

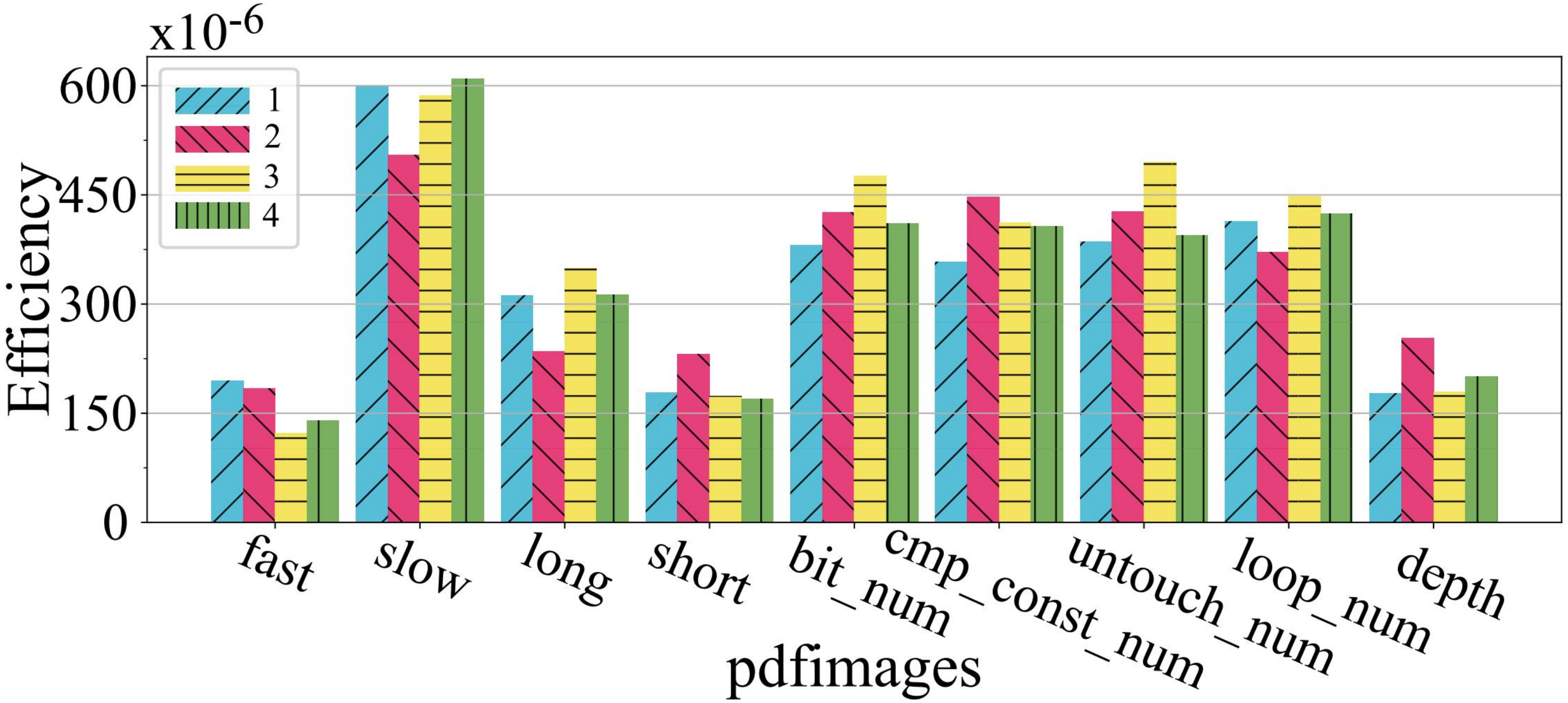
- **Key properties to estimate the potential of seeds**
  - **AFL: seeds which have faster execution speed, trigger more edges and are discovered in the later time**

- **Corresponding algorithms based on key properties**
  - **EcoFuzz: adversarial multi-armed bandit model to allocate more energy to the seeds, which have higher transition probabilities with fewer execution number**

Use **fixed metrics** for different programs

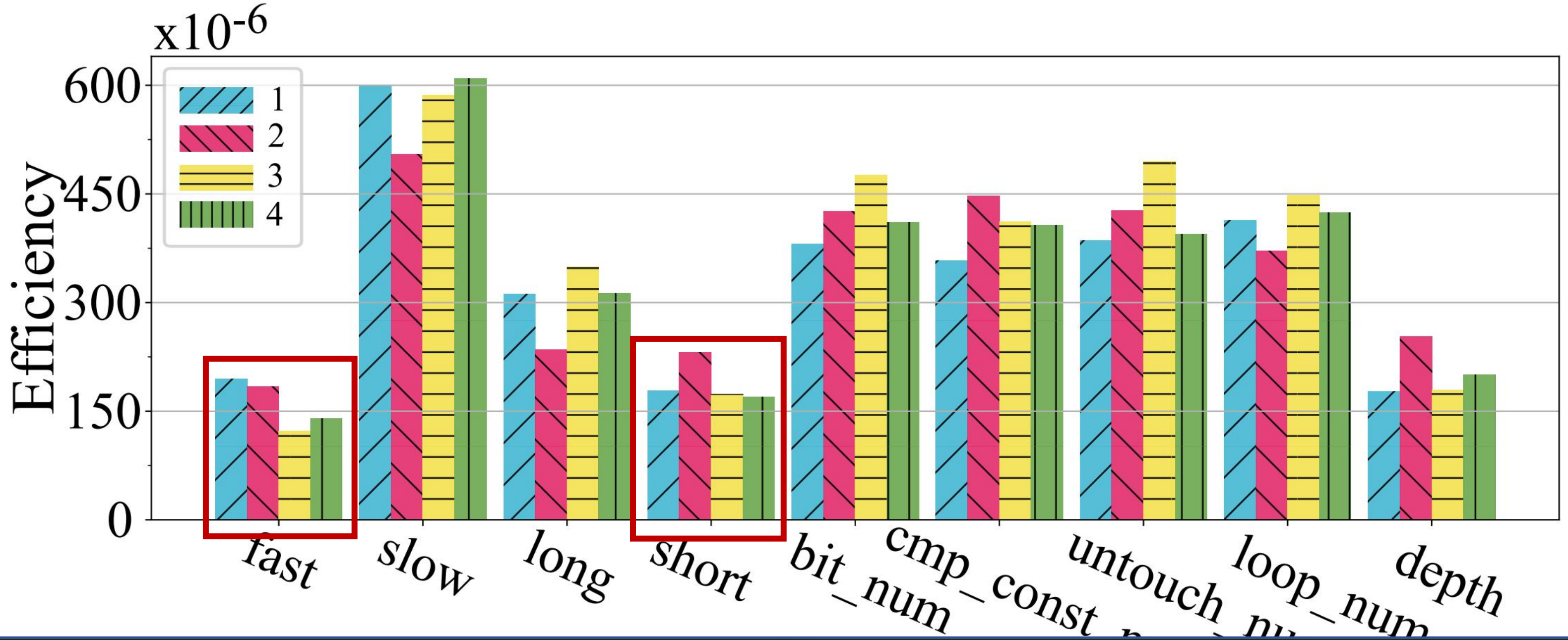
Motivation

➤ **Seeds with the same key properties perform well on every program?**



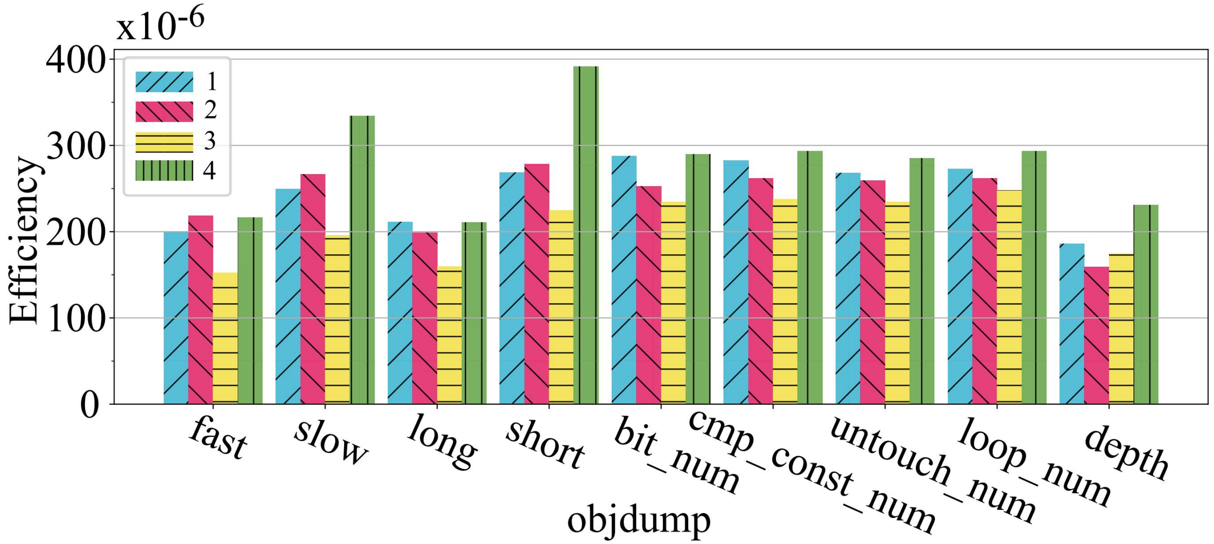
Motivation

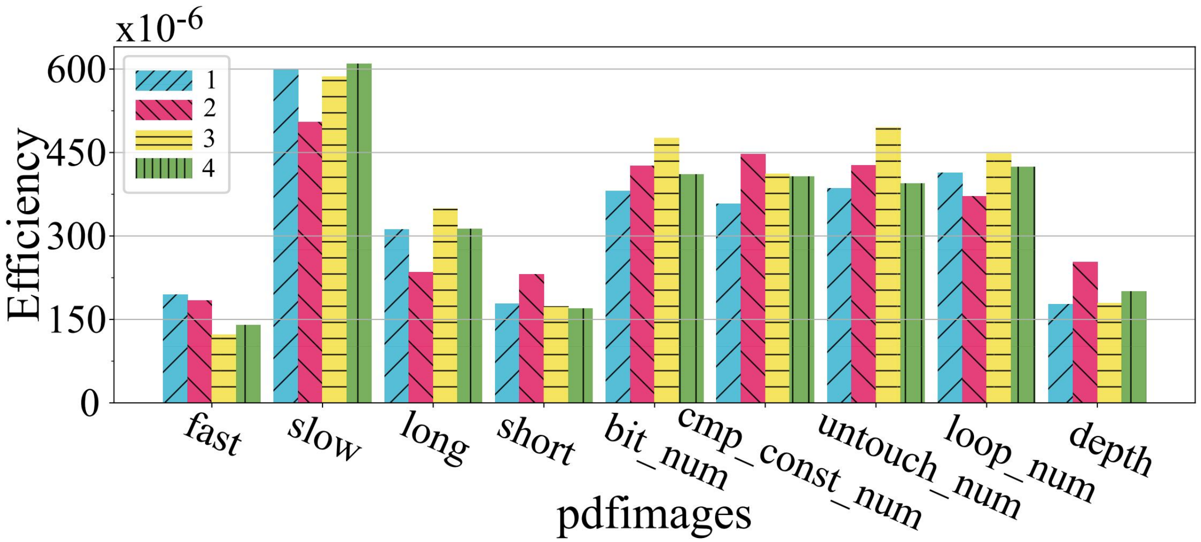
➢ **Seeds with the same key properties perform well on every program?**

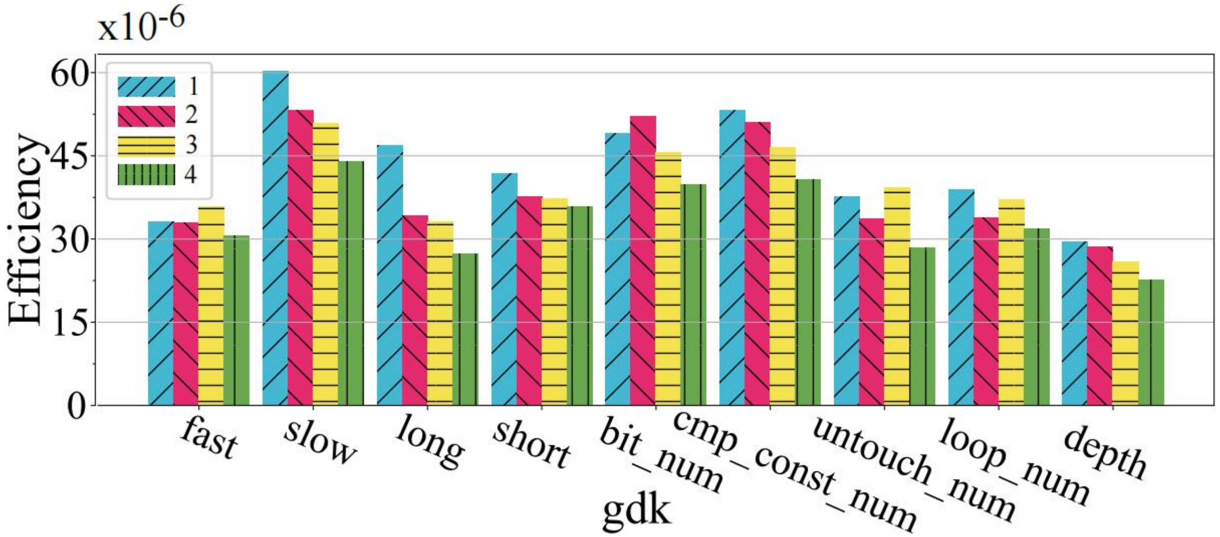


Key properties do not always work

Motivation

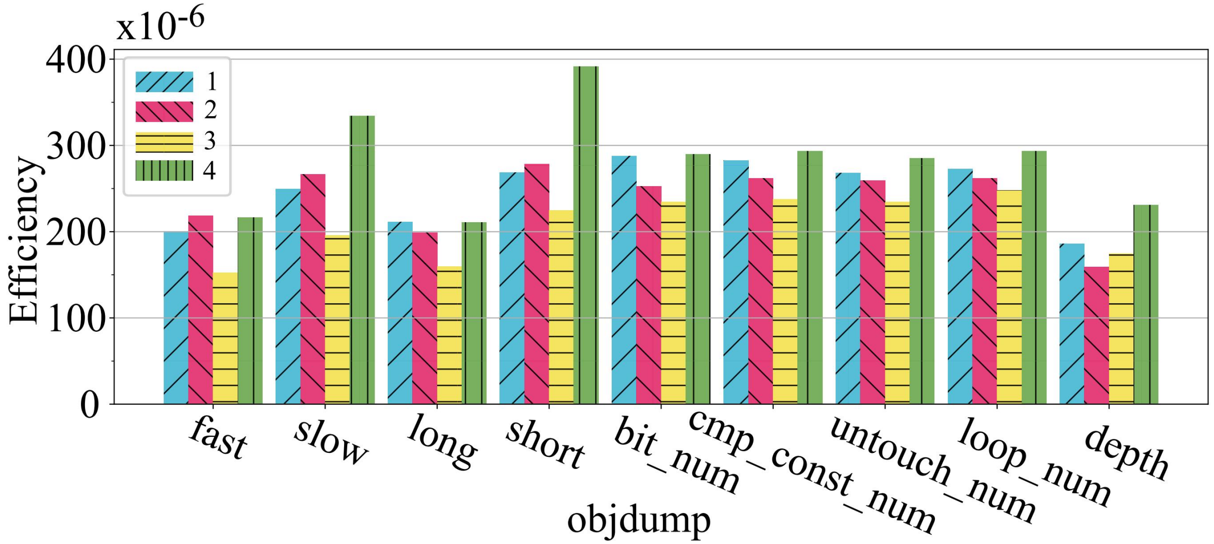


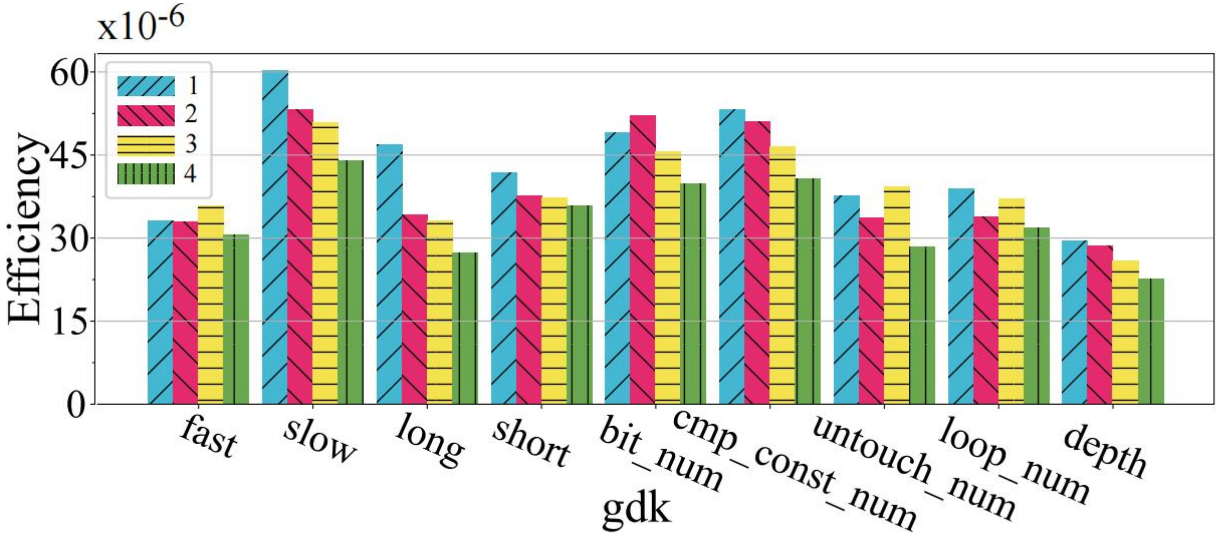




- **Seeds with the same property have different efficiency on different programs.**

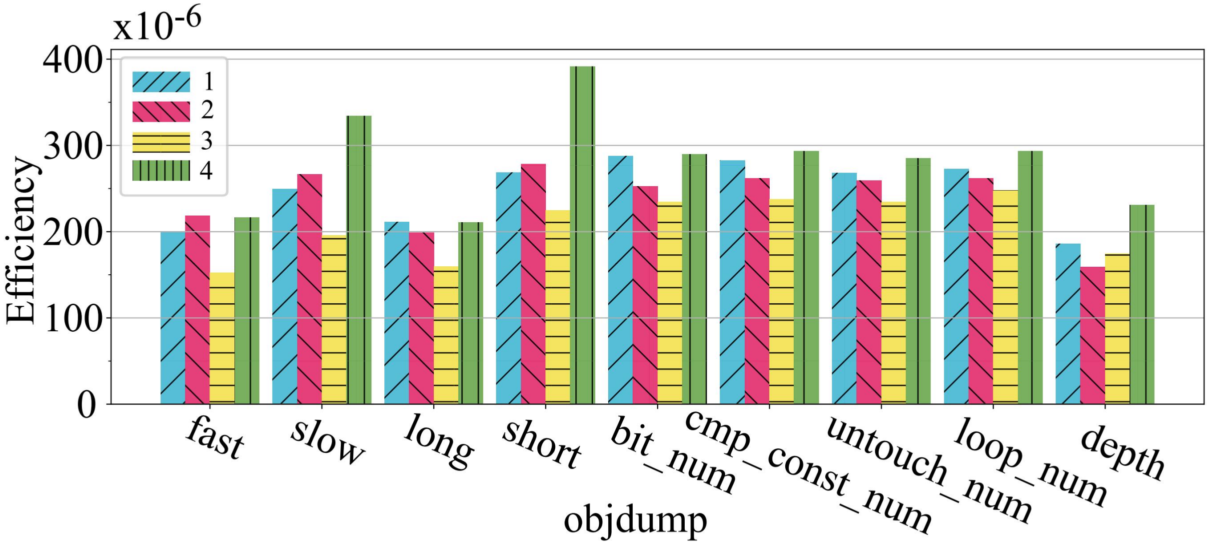
Motivation

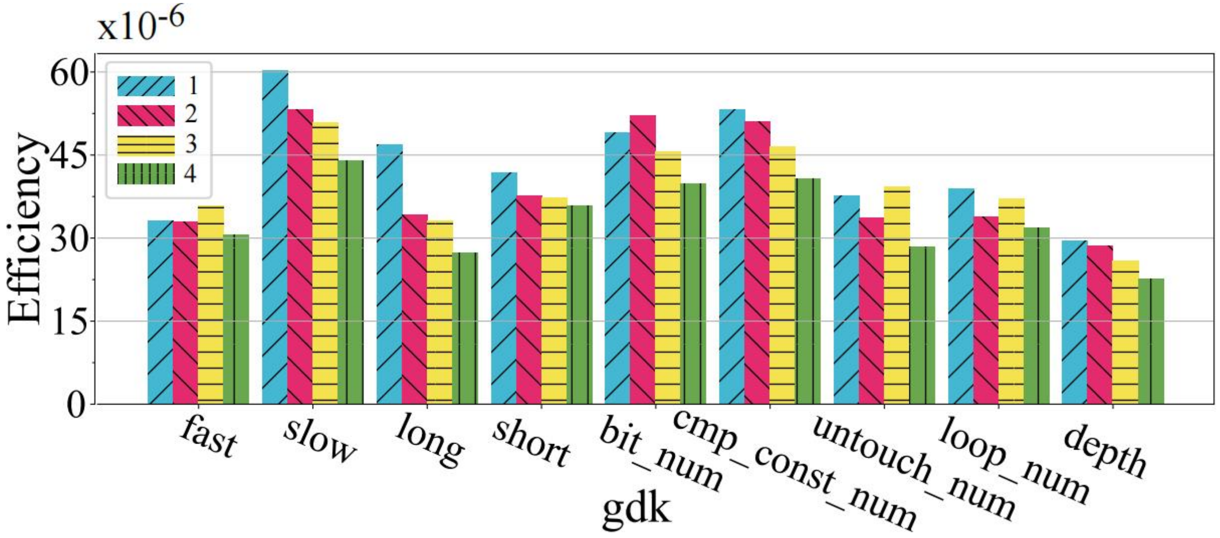




- **Seeds with the same property have similar efficiency in four repeated trials on a program.**

Motivation





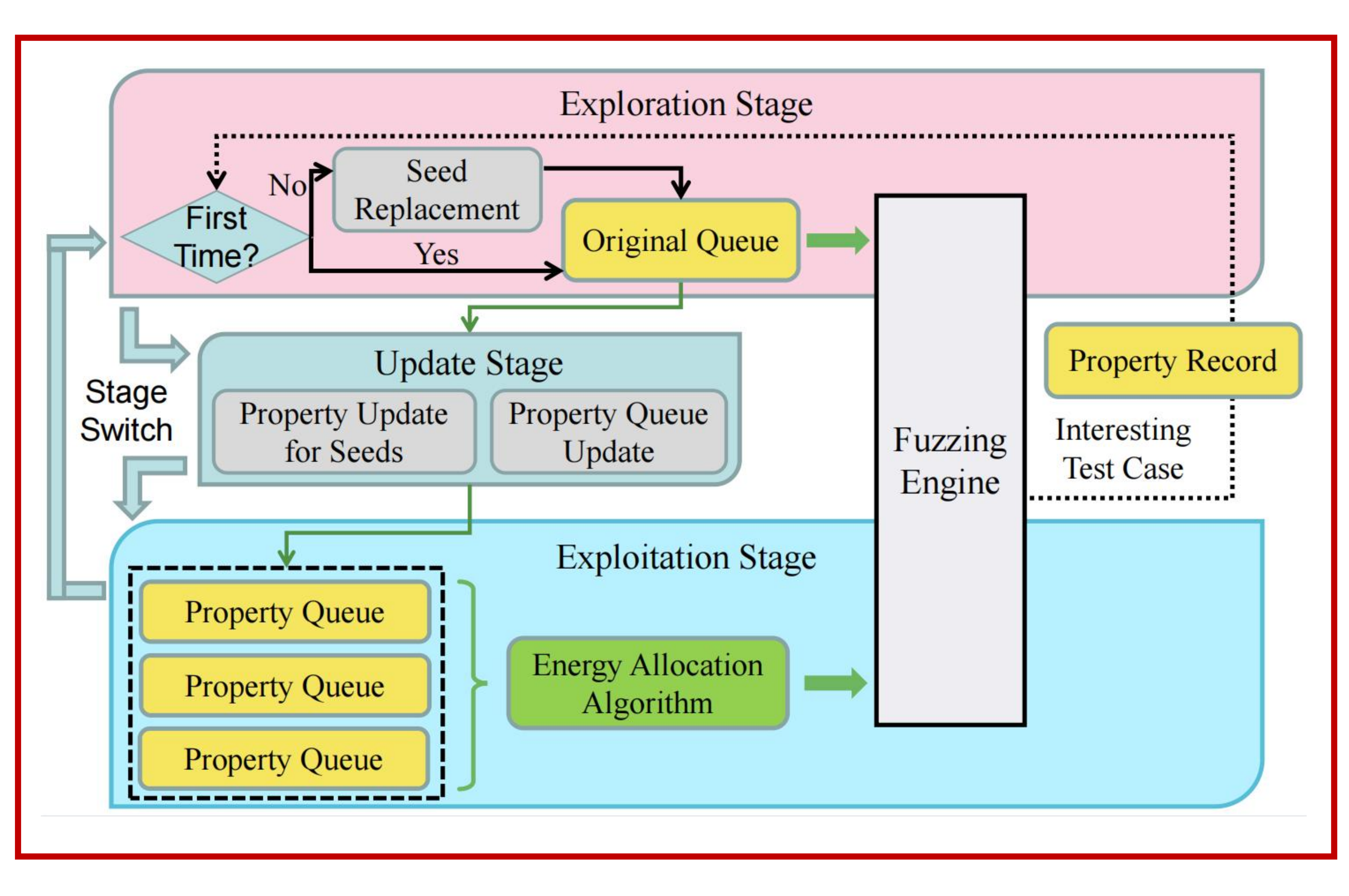
- **Seeds with the same property have similar efficiency in four repeated trials on a program.**
- **Seeds with different properties have different efficiency in four repeated trials on a program.**

Motivation

- Seeds with the same property have different efficiency on different programs.
- Seeds with the same property have similar efficiency in four repeated trials on a program.
- Seeds with different properties have different efficiency in four repeated trials on a program.

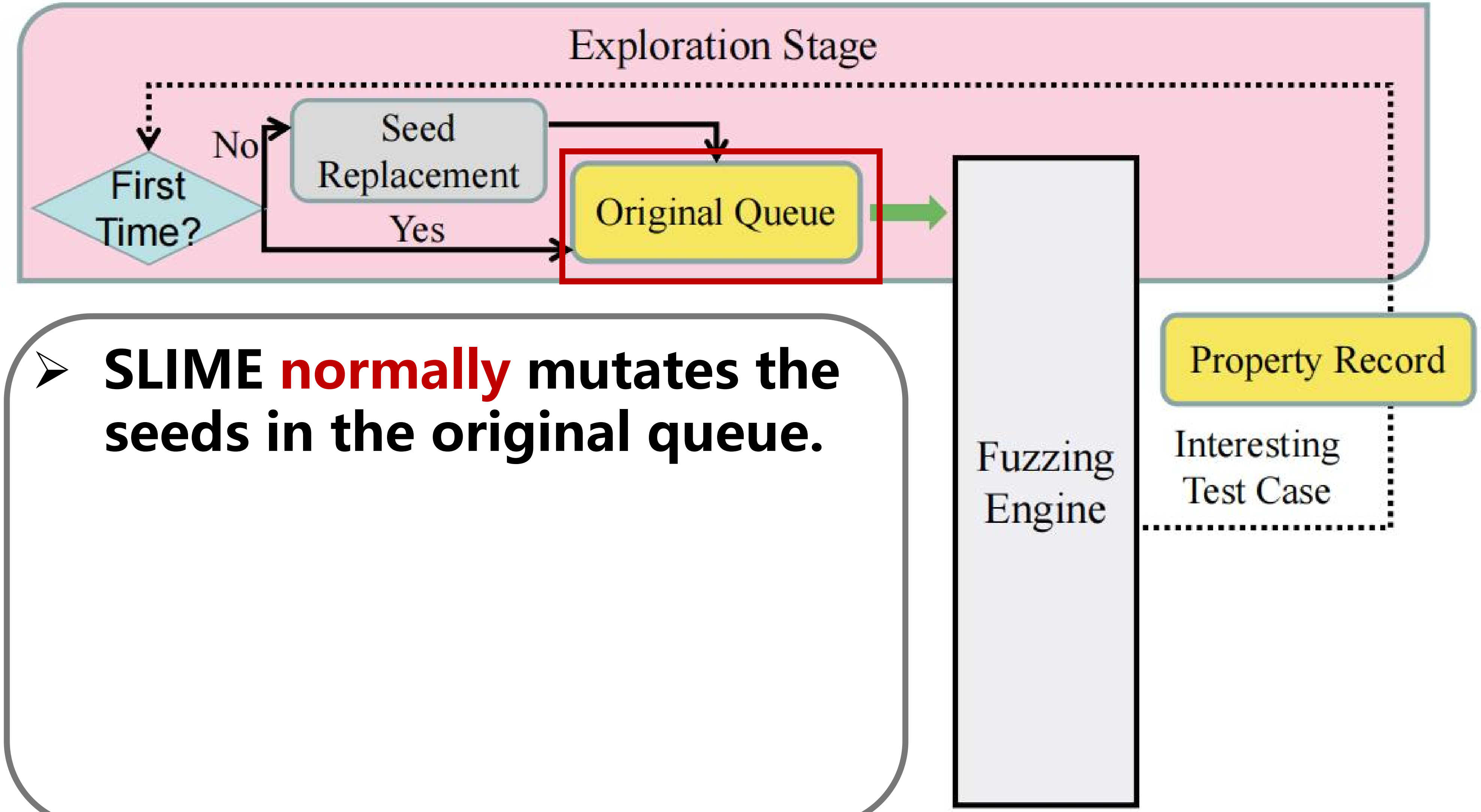
Program-Sensitive Energy Allocation

Frame of SLIME



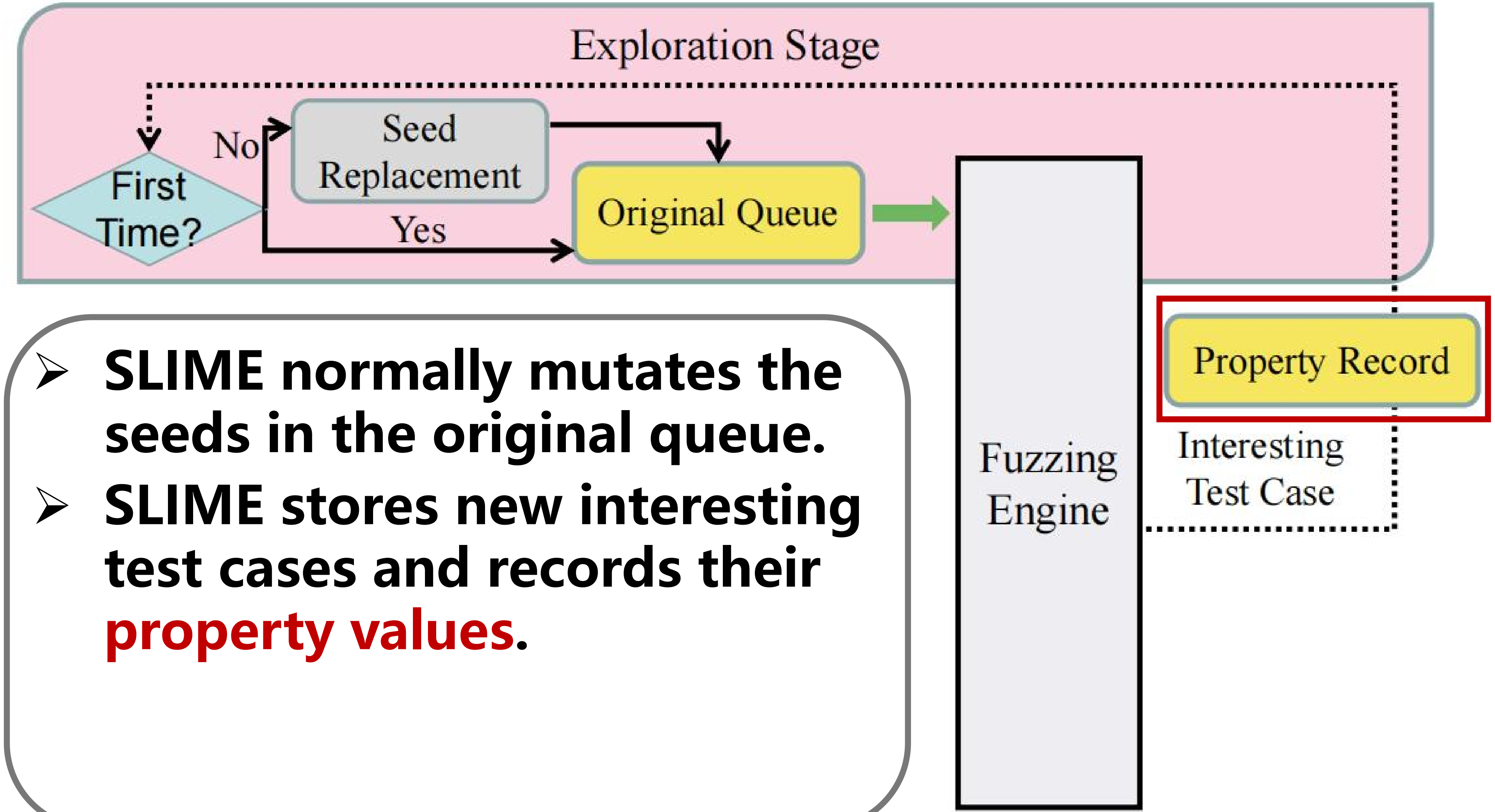
- 17 Kinds of Properties
- Seed Replacement
- Property-Adaptive Energy Allocation Algorithm

Stages of SLIME



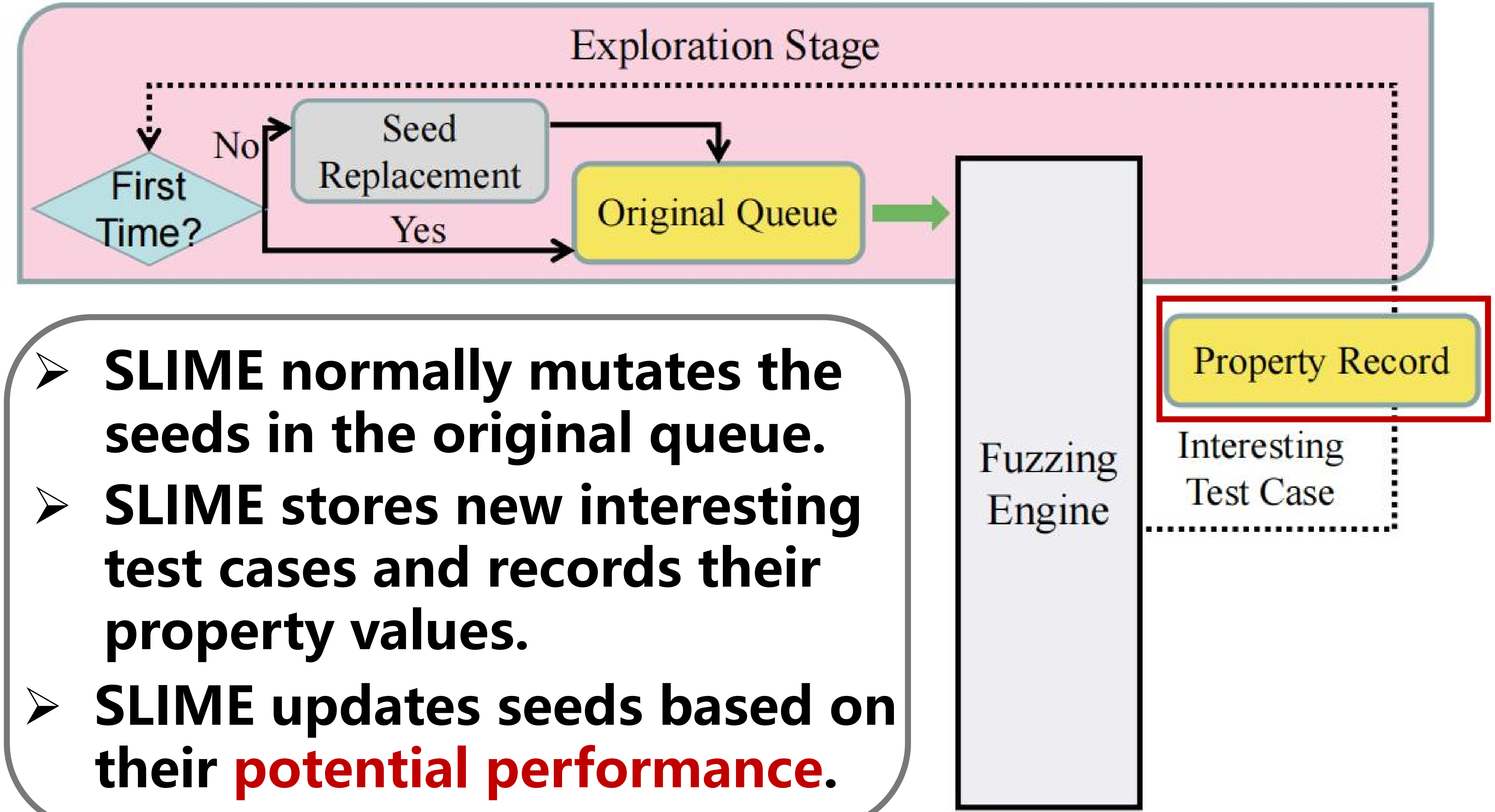
- **SLIME normally mutates the seeds in the original queue.**

Stages of SLIME



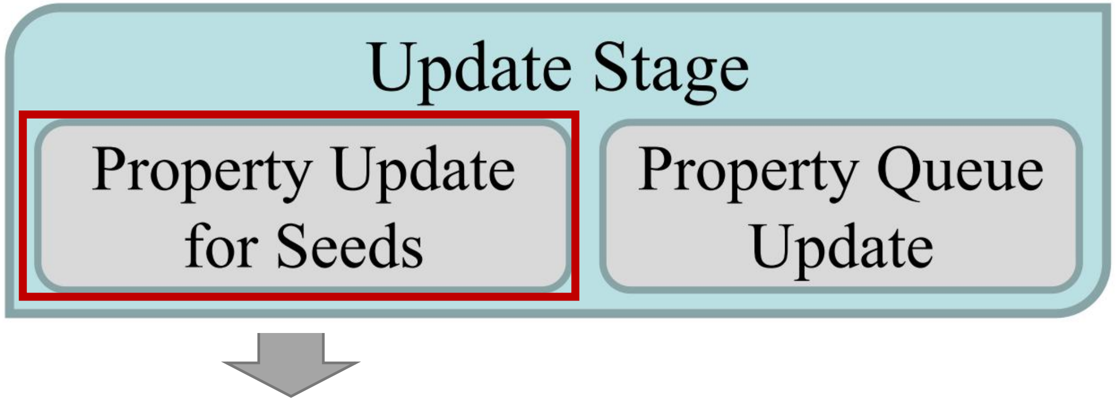
- SLIME normally mutates the seeds in the original queue.
- SLIME stores new interesting test cases and records their property values.

Stages of SLIME



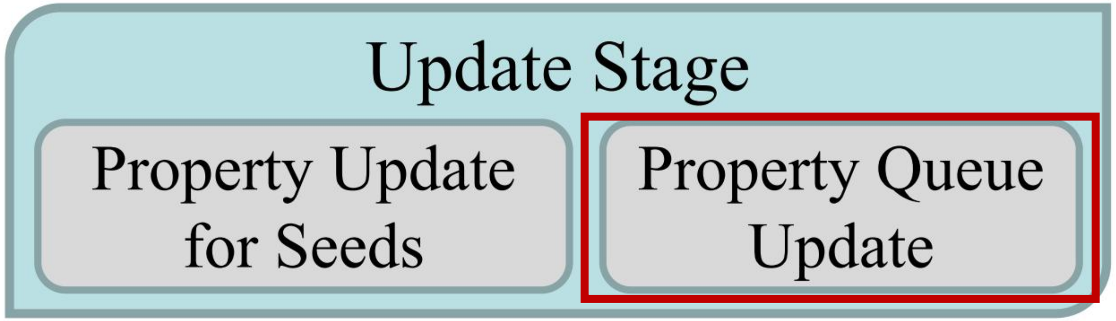
- **SLIME normally mutates the seeds in the original queue.**
- **SLIME stores new interesting test cases and records their property values.**
- **SLIME updates seeds based on their potential performance.**

Stages of SLIME



- **SLIME updates the property values of each seed file.**

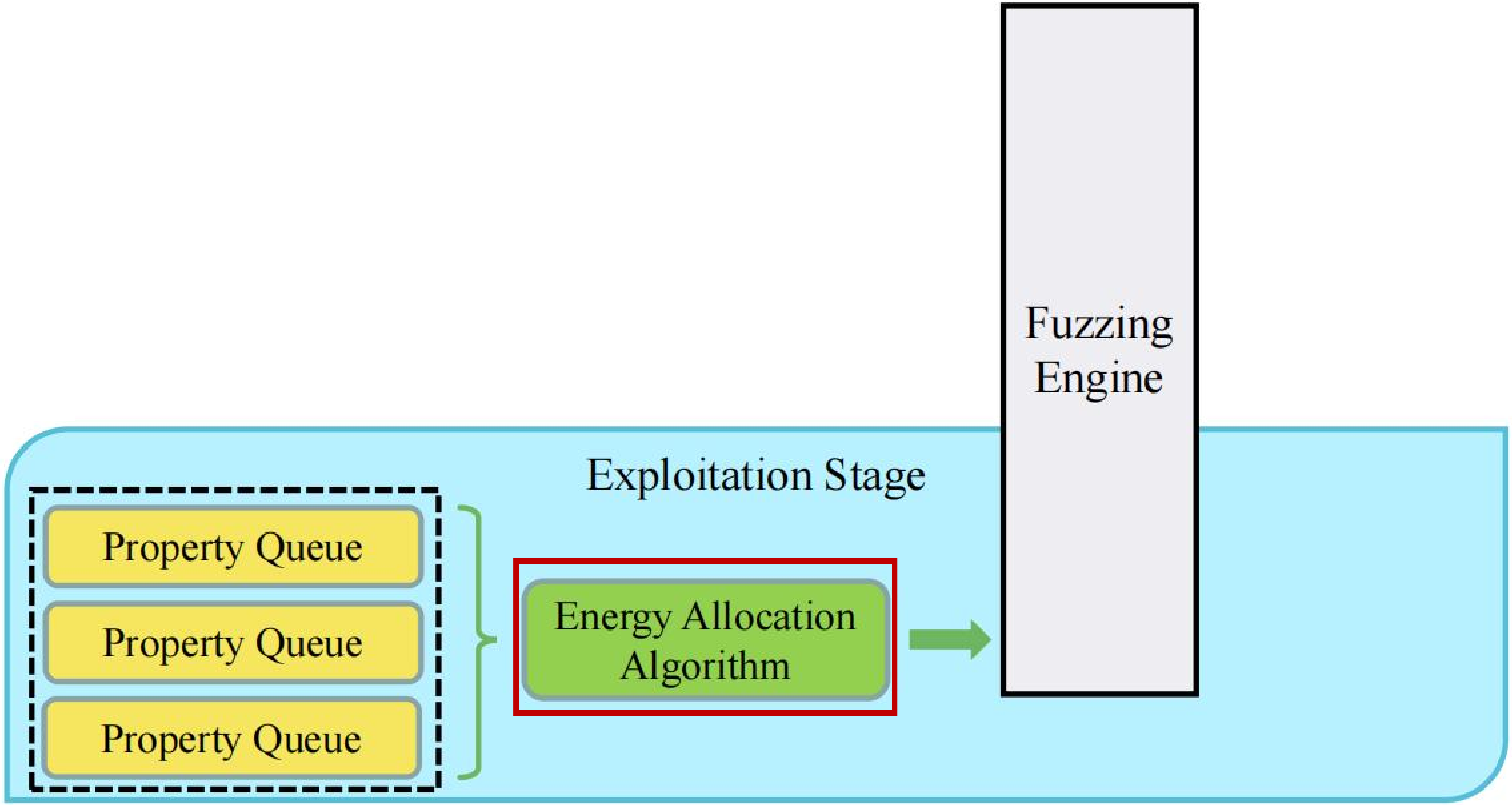
Stages of SLIME



- SLIME updates the property values of each seed file.

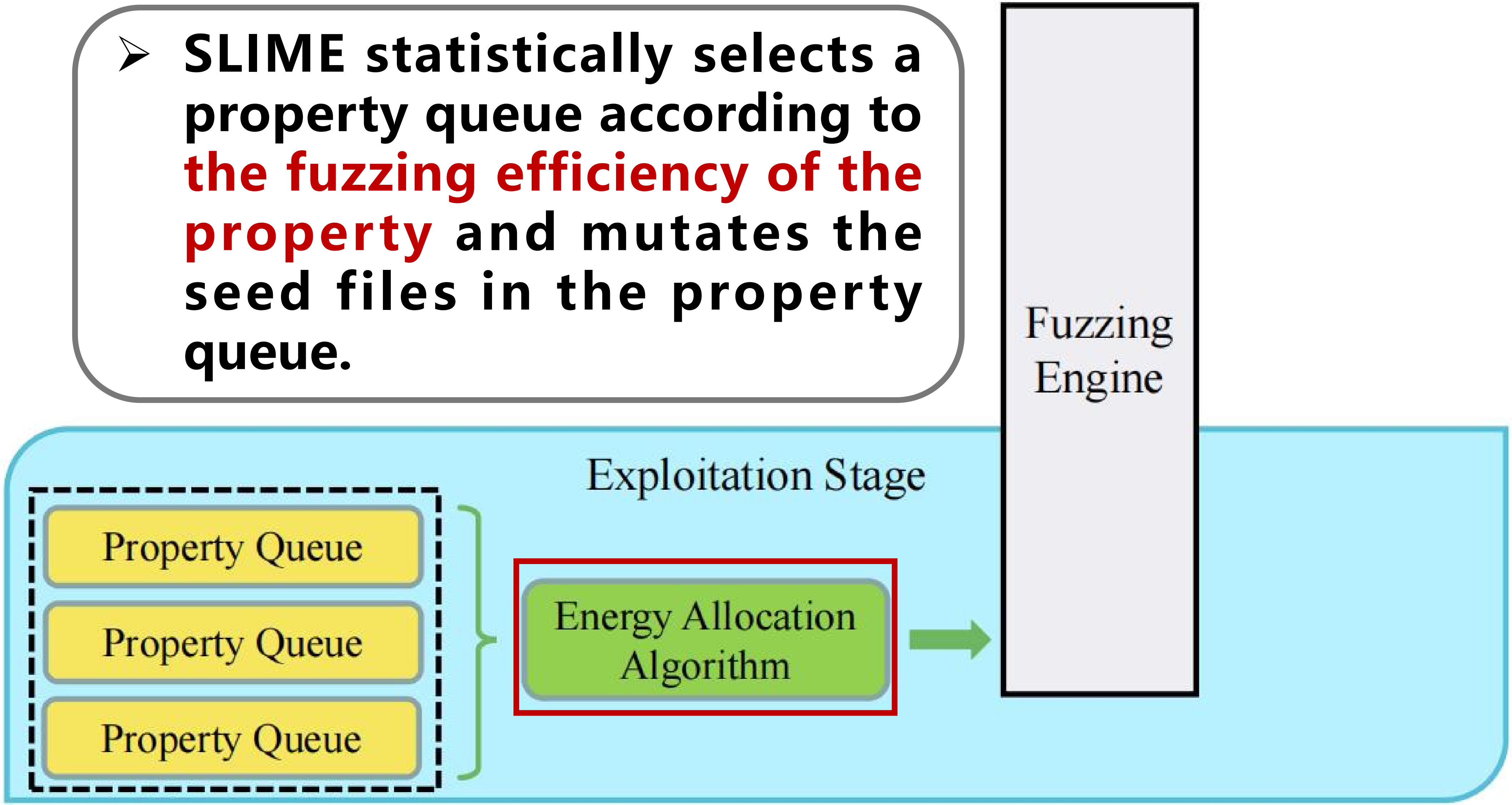
- SLIME empties all the seed files in all the property queues and reconstructs all the property queues.

Stages of SLIME

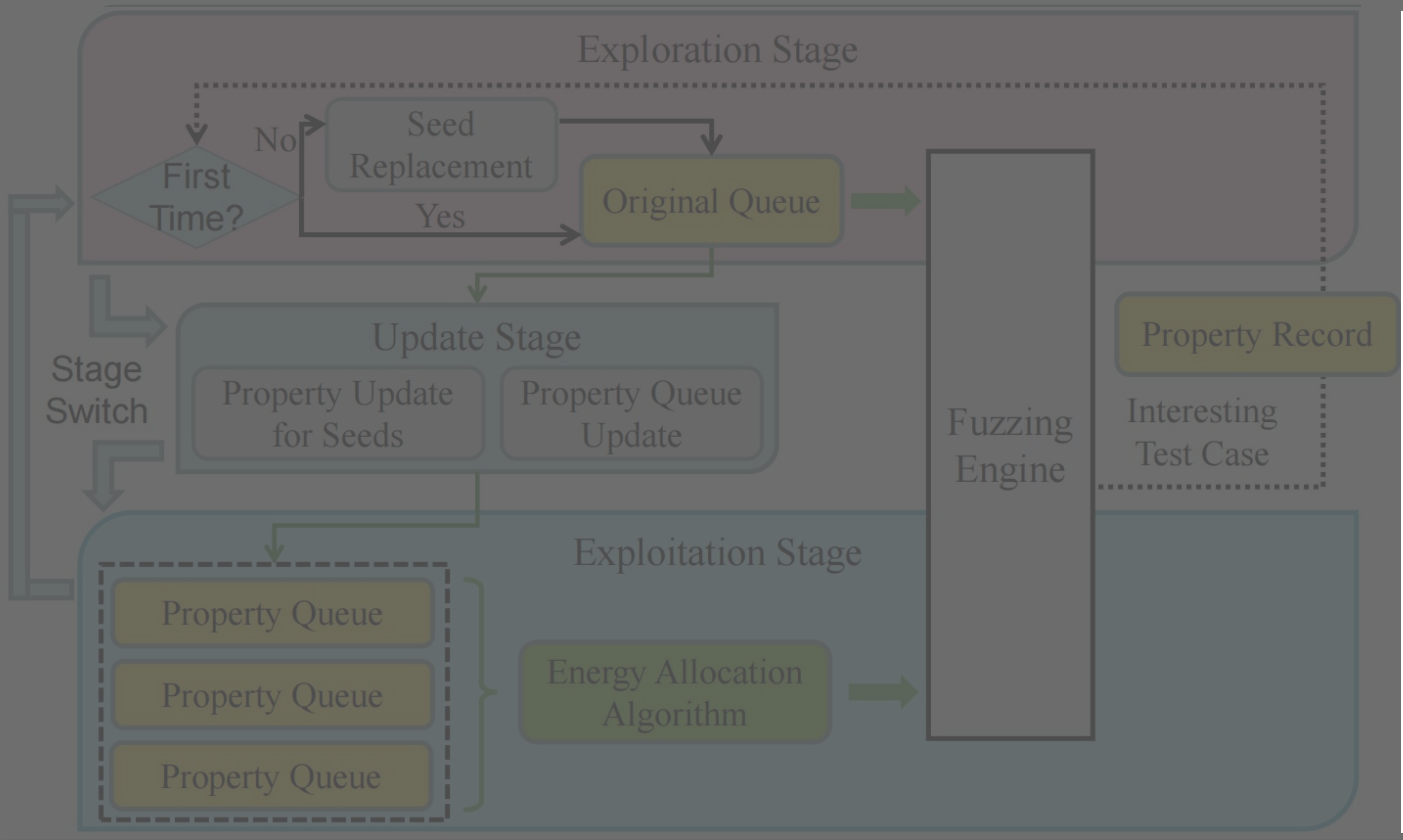


Stages of SLIME

- SLIME statistically selects a property queue according to **the fuzzing efficiency of the property** and mutates the seed files in the property queue.

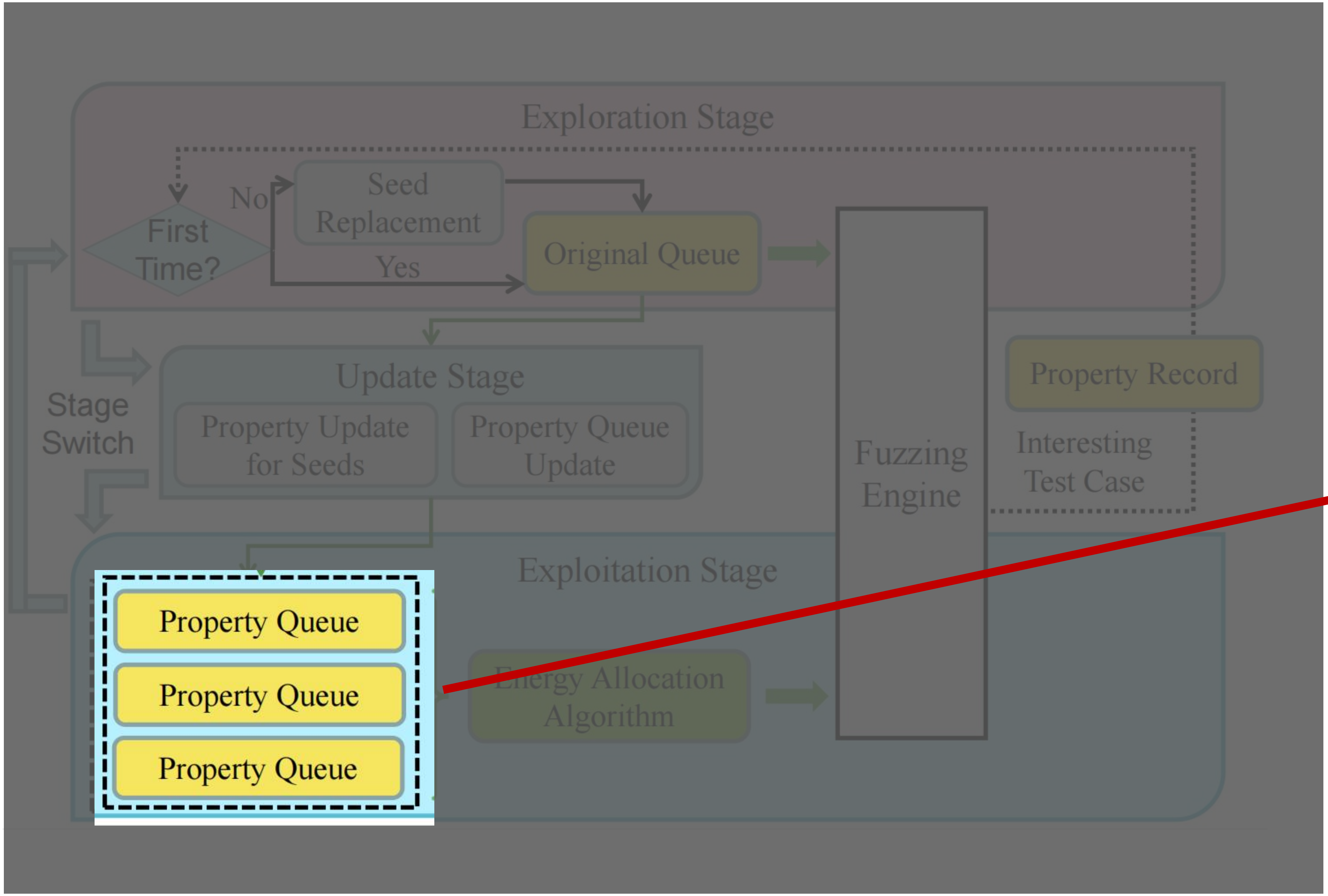


Frame of SLIME



- **17 Kinds of Properties**
- **Seed Replacement**
- **Property-Adaptive Energy Allocation Algorithm**

Frame of SLIME



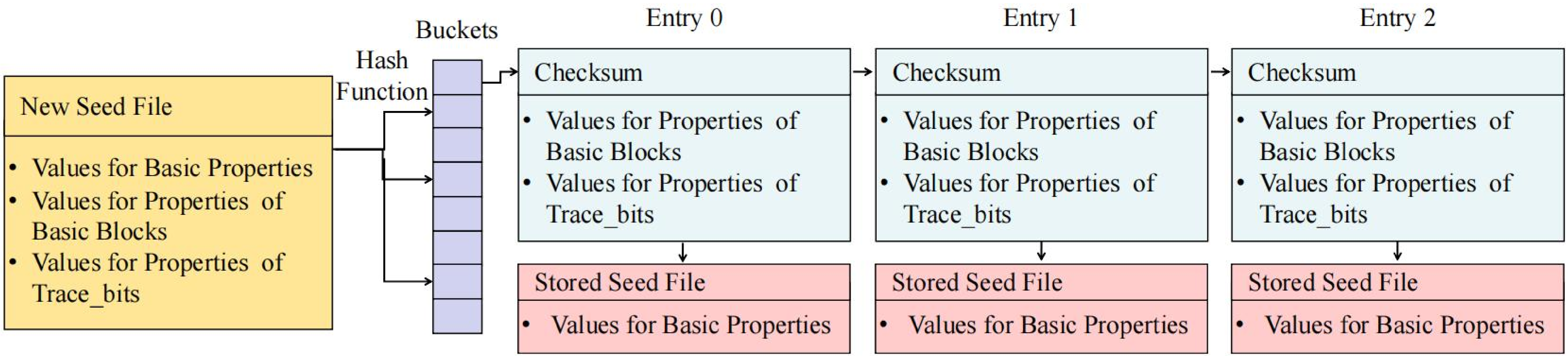
Define 17 kinds of properties from 3 perspectives

Properties and Queue Structure of SLIME

- **Define 17 kinds of properties from 3 perspectives**

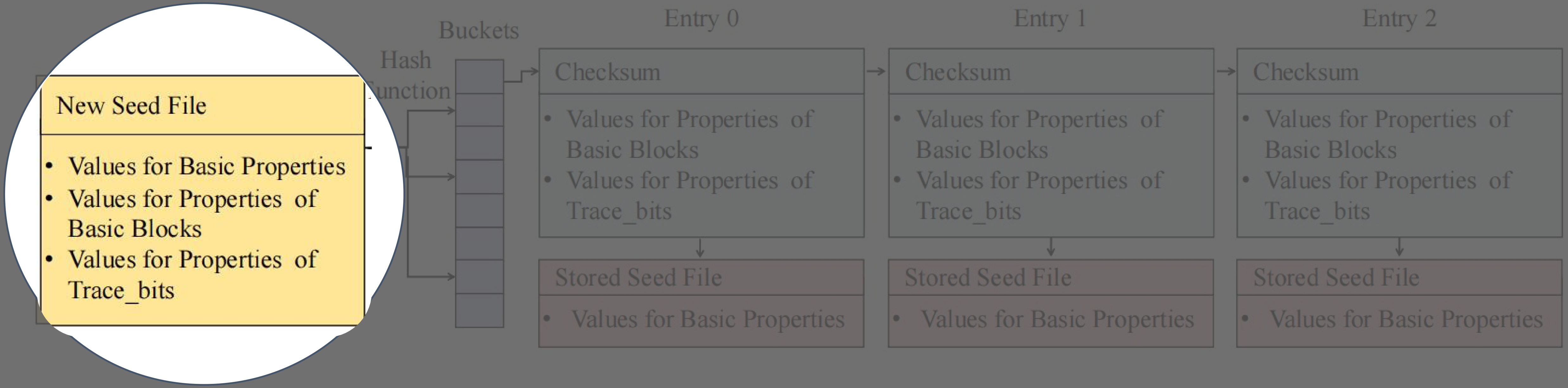
Basic properties related to seed files	Properties of the triggered basic block	Properties of triggered trace_bits
• fast • slow • long • short • depth • interesting • edge_change_eff • rare_file	• cmp_const_num • untouch_num • mem_num • func_num • global_num • global_assign_num • crash_num	• bit_num • loop_num

Properties and Queue Structure of SLIME



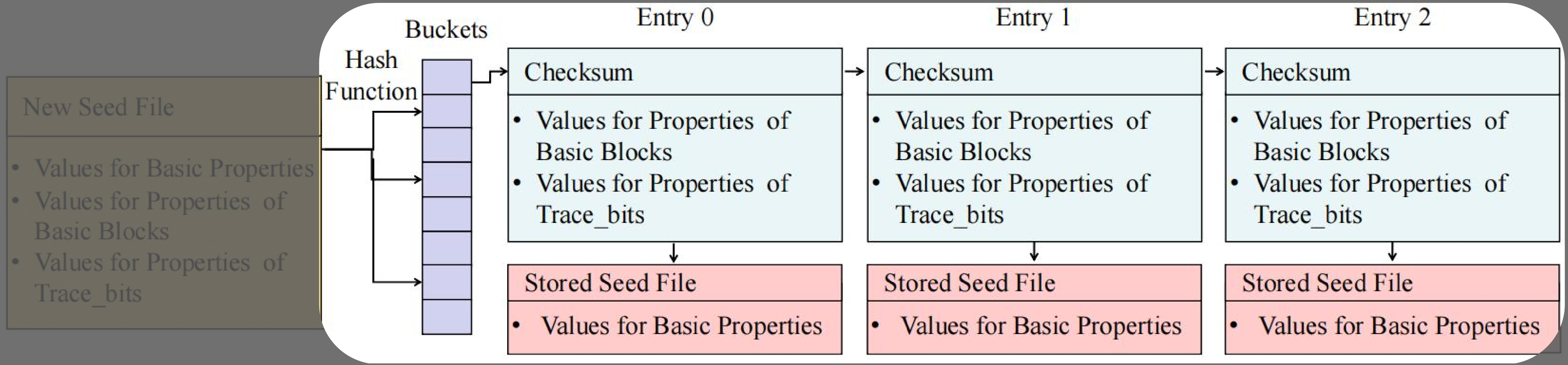
New structure instead of a linked list

Properties and Queue Structure of SLIME



New seed

Properties and Queue Structure of SLIME



Hash table

Seed Replacement

- SLIME finds the top 3 most efficient properties according to their frequency on the high-efficency seeds.

  High-efficiency seeds: The seeds with the interesting property, which generate the most interesting test cases in the original queue.

- Suppose the frequency of each property when testing gdk is like the following list.

fast	slow	long	short	depth	loop_num	rare_file	crash_num
6%	**32%**	12%	9%	7%	11%	8%	3%

untouch_num	mem_num	func_num	global_num	global_assign_num	cmp_const_num	bit_num	edge_change_eff
10%	7%	5%	8%	6%	**25%**	**27%**	14%

Seed Replacement

➢ **SLIME calculates the temporary score of a seed file on one of the top 3 most efficient properties by comparing its property value with the best performance.**

Property queue for slow：

$Seed_{best} \rightarrow Seed_{i} \rightarrow \dots \dots \rightarrow Seed_{end}$

Property queue for bit_num：

$Seed_{best} \rightarrow Seed_{i} \rightarrow \dots \dots \rightarrow Seed_{end}$

Property queue for bit_num：

$Seed_{best} \rightarrow Seed_{i} \rightarrow \dots \dots \rightarrow Seed_{end}$

$Score_{slow}$：
exec_time / max(exec_time)

$Score_{bit\_num}$：
bit_num / max(bit_num)

$Score_{cmp\_const\_num}$：
cmp_const_num / max(cmp_const_num)

Seed Replacement

- **SLIME calculates the temporary score of a seed file on one of the top 3 most efficient properties by comparing its property value with the best performance.**

Property queue for slow：

$Seed_{best} \rightarrow Seed_i \rightarrow \dots \dots \rightarrow Seed_{end}$

Property queue for bit_num：

$Seed_{best} \rightarrow Seed_i \rightarrow \dots \dots \rightarrow Seed_{end}$

Property queue for bit_num：

$Seed_{best} \rightarrow Seed_i \rightarrow \dots \dots \rightarrow Seed_{end}$

$Score_{slow}$：
exec_time / max(exec_time)

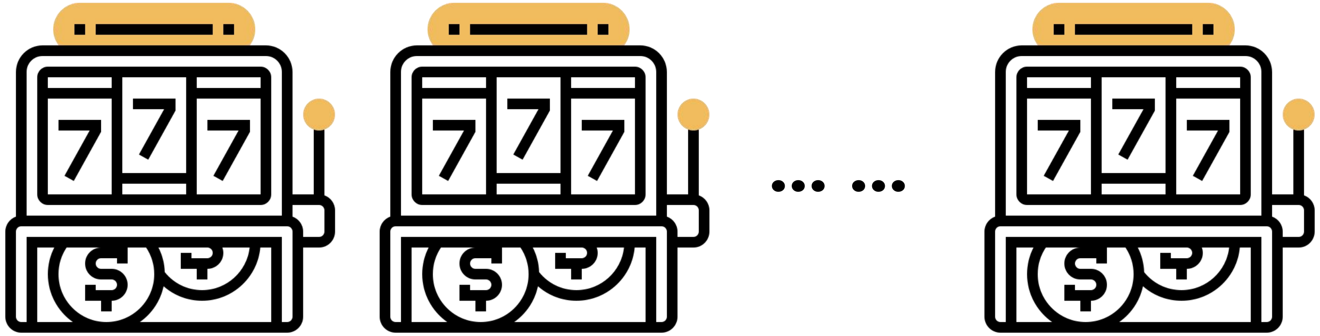
$Score_{bit\_num}$：
bit_num / max(bit_num)

$Score_{cmp\_const\_num}$：
cmp_const_num / max(cmp_const_num)

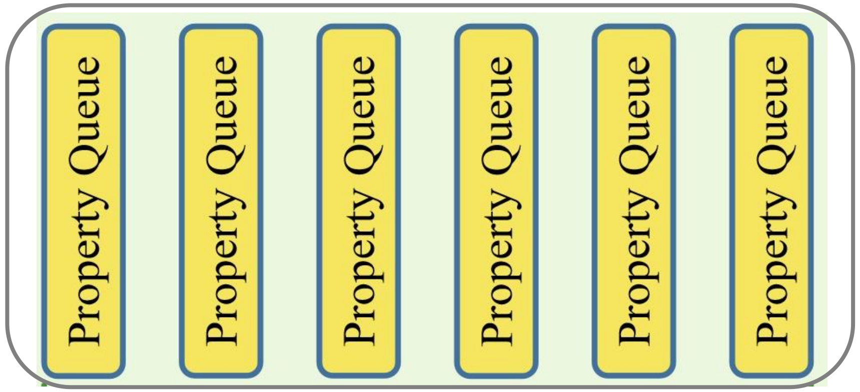
$$potential\_score = Score_{slow} + Score_{bit\_num} + Score_{cmp\_const\_num}$$

Property-Adaptive Energy Allocation Algorithm

- Since SLIME is supposed to select **the property queues containing the efficient seeds** more times and improve the fuzzing performance, the queue selection problem can be regarded as a **multi-armed bandits** problem.



Which arm to pick next?

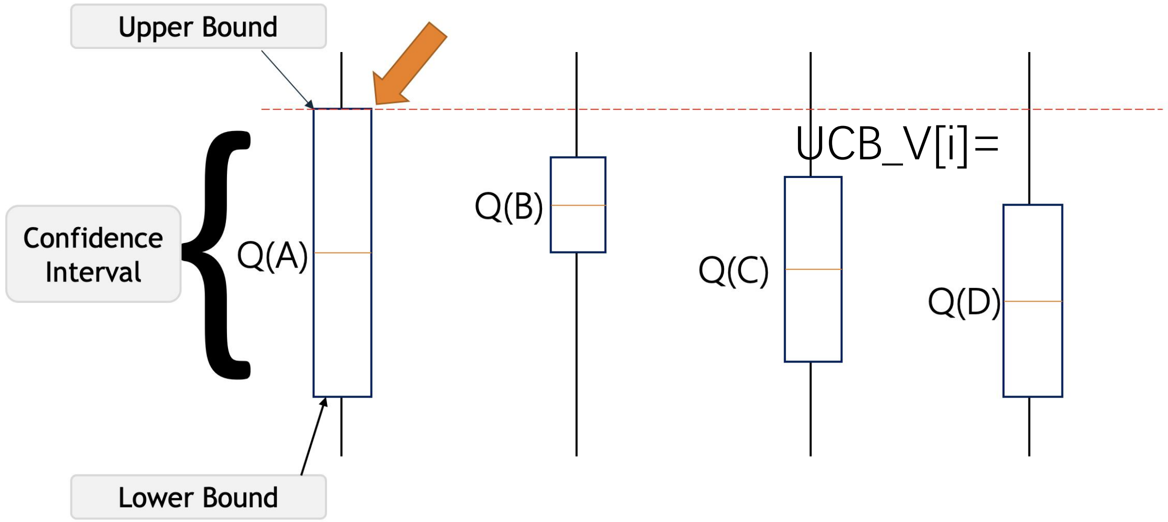


Which queue to select?

Customized UCB-V algorithm

Property-Adaptive Energy Allocation Algorithm

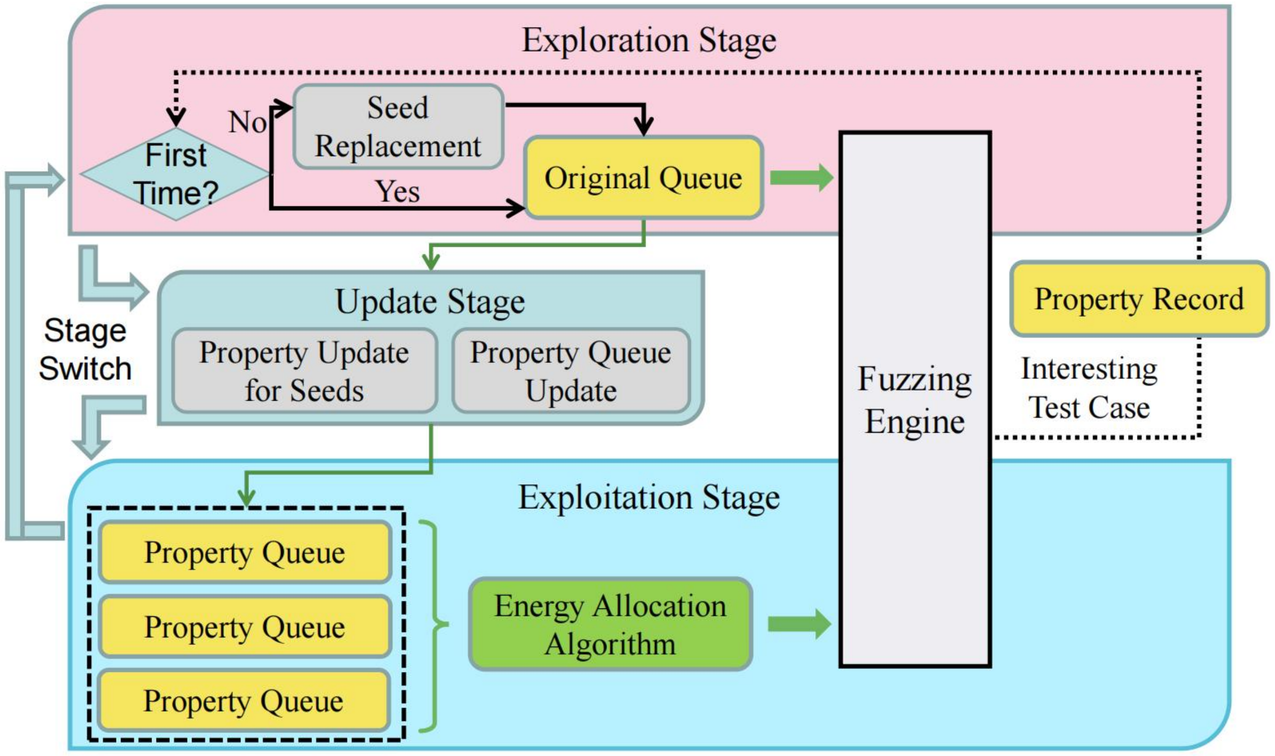
- SLIME estimates the confidence interval for **the number of newly discovered interesting test cases** if selecting a property queue in the Exploitation Stage.

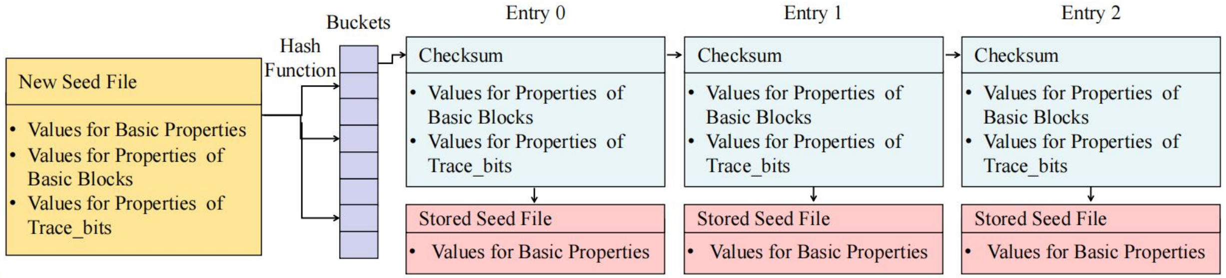


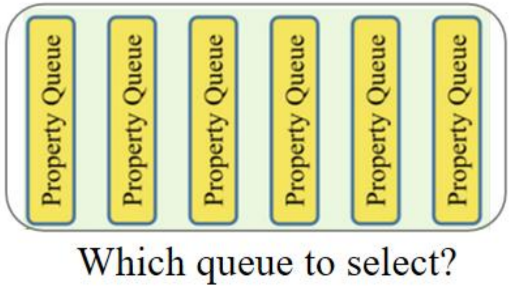
- The customized UCB-V algorithm regards **the upper confidence bound** of the estimated interval as the reward.

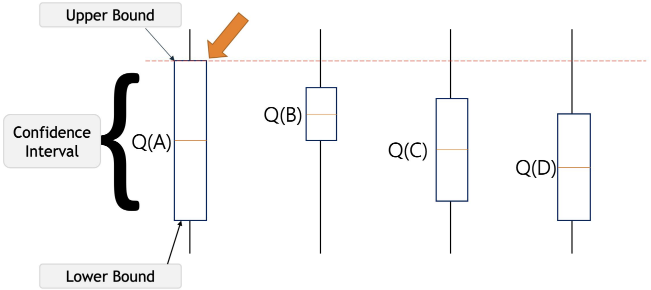
$$UCB\_V[i] = \frac{R[i]}{N[i]} + \sqrt{\frac{2 \times Variance[i] \times log(N\_total)}{N[i]}} + \frac{3 \times log(N\_total)}{N[i]}$$

Design of SLIME









$$UCB\_V[i] = \frac{R[i]}{N[i]} + \sqrt{\frac{2 \times Variance[i] \times \log(N\_total)}{N[i]}} + \frac{3 \times \log(N\_total)}{N[i]}$$

Program-Sensitive Energy Allocation

Evaluation

Experiment Settings

- **Compared fuzzers**

  AFL, MOpt, AFL++, AFL++HIER, EcoFuzz, TortoiseFuzz

- **Target programs**

Program	Version	Input format	Test instruction
cflow	1.6	txt	@@
ffmpeg	4.0.1	mp4	-y -i @@ -c:v mpeg4 -c:a copy -f mp4 /dev/null
gdk	gdk-pixbuf 2.31.1	jpg	@@ /dev/null
imginfo	jasper 2.0.12	jpg	-f @@
jhead	3.00	jpg	@@
mp3gain	1.5.2-r2	mp3	@@
objdump	binutils 2.28	binary	-S @@
pdfimages	xpdf 4.00	pdf	@@ /dev/null
tiffsplit	libtiff 3.9.7	tiff	@@

Each evaluation lasts for 120 hours and is repeated 20 times.

Length Selection Analysis

The unique vulnerability discovery of SLIME with the different property queue lengths

Programs	Length ratio of property queues to the original queue			
	1/10	4/10	8/10	10/10
ffmpeg	1	**2**	1	1
jhead	4	**5**	4	4
objdump	15	**16**	15	15
tiffsplit	12	**13**	11	11
total	32	**36**	31	31

Each trial lasts 96 hours and is repeated 4 times to reduce randomness.

SLIME performs the **best with a length ratio of 4/10**

Vulnerability Discovery

	AFL	MOpt	AFL++	AFL++HIER	EcoFuzz	TortoiseFuzz	SLIME
cflow	0	2	2	1	2	1	**4**
ffmpeg	0	2	2	0	0	0	**3**
gdk	23	31	26	23	26	20	**32**
imginfo	0	0	0	0	0	0	**1**
jhead	5	6	5	0	5	5	**10**
mp3gain	8	17	16	18	16	5	**23**
objdump	5	30	28	5	14	5	**39**
pdfimages	1	75	49	44	48	0	**87**
tiffsplit	9	24	15	0	12	12	**32**
total	51	187	143	91	123	48	**231**

SLIME finds the **most** total unique vulnerabilities

Vulnerability Discovery

The number and types of new unique vulnerabilities[1] which are only found by SLIME and are missed by other fuzzers

	SEGV on unknown address, READ memory access	heap-buffer-overflow	stack-overflow	memory leaks	allocation-size-too-big	total
ffmpeg	1	0	0	0	0	1
gdk	0	3	0	0	0	3
jhead	0	4	0	0	0	4
objdump	7	1	0	0	0	8
pdfimages	1	10	4	0	0	15
tiffsplit	0	3	0	5	2	10
total	9	21	4	5	2	$\boxed{41}$

New unique vulnerabilities: vulnerabilities that 1) cannot be found by other fuzzers and 2)are not published on the CVE website

SLIME finds **more** new unique vulnerabilities missed by others

Vulnerability Discovery

The properties and values of each original seed of SLIME that triggers a new unique vulnerability on objdump after mutation. A value in bold font means that the original seed has the corresponding property.

seed_id	long (file size)	global_num	global_assign_num	func_num
No. 1	**32,391.00**	**660.00**	**108.00**	47.00
No. 2	10,432.00	583.00	**101.00**	70.00
No. 3	13,488.00	564.00	91.00	**78.00**
No. 4	**32,452.00**	**720.00**	**121.00**	47.00
mean among all the seeds	54,116.90	508.08	84.18	64.78
median among all the seeds	13,952.00	577.50	94.00	57.50

SLIME mutates the **important** seeds more times

Vulnerability Discovery

The published CVE IDs found by each fuzzer

	CVE ID	AFL	MOpt	AFL++	AFL++HIER	EcoFuzz	TortoiseFuzz	SLIME
cflow	CVE-2020-23856							●
	CVE-2019-16166							●
	CVE-2019-16165		●	●	●	●	●	●
imginfo	CVE-2017-6851							●
jhead	CVE-2020-6624	●	●	●		●	●	●
	CVE-2019-1010302	●	●	●				●
	CVE-2019-19035					●	●	
mp3gain	CVE-2017-14410			●	●			●
	CVE-2017-14409		●	●	●	●		●
	CVE-2017-14407	●	●	●	●	●	●	●
objdump	CVE-2021-3487		●	●		●		●
	CVE-2019-17450							●
	CVE-2019-9072		●	●				●
	CVE-2018-1000876		●	●				●
	CVE-2018-7568		●	●		●		●
	CVE-2017-16831		●	●		●		●
	CVE-2017-16826		●	●		●		●
	CVE-2017-15024		●	●				●
	CVE-2017-14938		●	●				●
	CVE-2017-8396	●	●	●	●	●	●	●
pdfimages	CVE-2020-24999		●					●
	CVE-2019-13291		●			●		●
	CVE-2019-13281		●	●		●		●
	CVE-2019-10022							●
	CVE-2018-18458		●					
	CVE-2018-18455		●	●	●	●		
	CVE-2018-16368	●	●	●	●	●		●
	CVE-2018-7453	●	●	●	●	●		●
	total	6	21	19	8	15	5	**25**

SLIME achieves the **best** performance on CVE discovery

Coverage Discovery

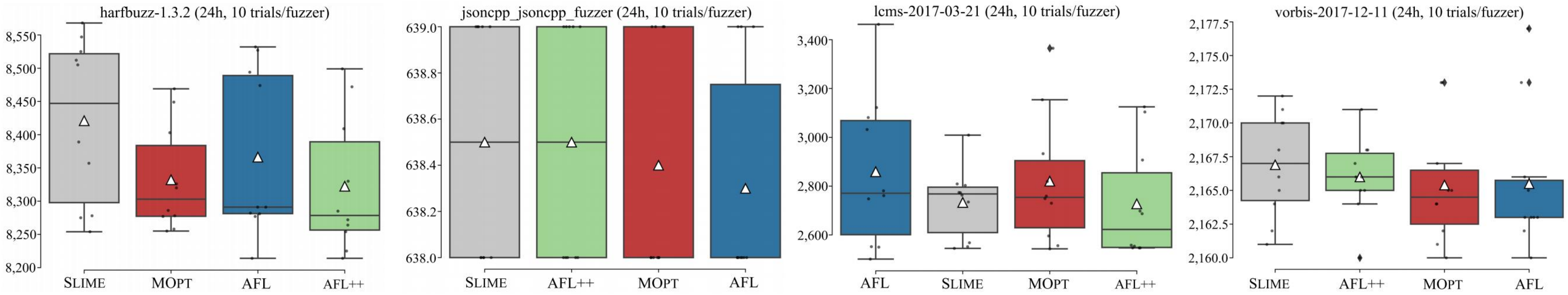
The number of average edge coverage in 20 trials found by each fuzzer

	AFL	MOpt	AFL++	AFL++HIER	EcoFuzz	TortoiseFuzz	SLIME
cflow	1,760.50	1,784.05	1,785.65	1,804.55	1,784.30	1,762.25	**1,819.75**
ffmpeg	18,046.25	31,343.45	**43,838.75**	33,317.40	21,293.55	17,002.20	32,825.75
gdk	1,458.40	1,985.00	1,577.95	1,505.40	1,236.30	1,335.75	**2,038.85**
imginfo	2,192.15	3,258.90	2,743.35	1,179.55	2,815.85	1,716.05	**3,500.25**
jhead	283.00	283.00	283.00	283.00	283.00	283.00	283.00
mp3gain	1,203.40	1,297.45	1,282.85	1,278.05	1,281.25	1,191.80	**1,300.40**
objdump	5,568.45	7,687.20	**8,071.95**	5,785.40	7,009.00	5,576.85	7,954.95
pdfimages	10,275.55	11,497.80	11,245.80	10,909.60	10,626.35	10,239.35	**11,932.70**
tiffsplit	3,036.35	3,291.75	3,290.40	2,289.60	2,964.20	2,988.90	**3,308.50**

SLIME performs the **best** on most programs

Coverage Discovery

The boxplot of region coverage found in 10 trials on FuzzBench



Each evaluation lasts 24 hours and is repeated 10 times to reduce the randomness. '△' and '—' represent the mean and median. The fuzzer with the highest median coverage is on the left. Y-axis: the region coverage found in each trial.

SLIME performs the **best** on a standardized benchmark

Energy Allocation Algorithm Analysis

The number of unique vulnerabilities found by MOpt, AFL++, SLIME_rand[1], and SLIME

		MOpt	AFL++	SLIME_rand	SLIME
# of total unique vulnerabilities in 20 trials	gdk	31	26	30	**32**
	objdump	30	28	30	**39**
	tiffsplit	24	15	23	**32**
	Total	85	69	83	**103**
Average # of unique vulnerabilities in each trial	gdk	17.15	14.00	18.45	**20.03**
	objdump	7.65	5.90	10.10	**12.10**
	tiffsplit	8.40	4.00	7.70	**15.75**
	Total	33.20	23.90	36.25	**47.88**

SLIME_rand: selects each property queue randomly in the Exploitation Stage.
Each evaluation lasts 120 hours and is repeated 20 times.

The property queue construction cannot significantly improve the vulnerability discovery performance

Energy Allocation Algorithm Analysis

The number of unique vulnerabilities found by MOPT, AFL++, SLIME_rand, and SLIME

		MOPT	AFL++	SLIME_rand	SLIME
# of total unique vulnerabilities in 20 trials	gdk	31	26	30	**32**
	objdump	30	28	30	**39**
	tiffsplit	24	15	23	**32**
	Total	85	69	83	**103**
Average # of unique vulnerabilities in each trial	gdk	17.15	14.00	18.45	**20.03**
	objdump	7.65	5.90	10.10	**12.10**
	tiffsplit	8.40	4.00	7.70	**15.75**
	Total	33.20	23.90	36.25	**47.88**

Our customized UCB-V algorithm can improve performance

Seed Replacement Analysis

The average edge coverage increment of SLIME_no[1] and SLIME when using an extensive data set, which has found the most edge coverage, as the initial seed set.

Programs	Original edge results	SLIME_no	SLIME	Increase [a]
ffmpeg	32,825.75	+3,813.35	+4,531.15	+18.82%
imginfo	3,500.25	+207.15	+328.75	+58.70%
pdfimages	11,932.70	+36.60	+50.30	+37.43%
tiffsplit	3,308.50	+27.30	+29.60	+8.42%

[a]Increase is calculated by the results of SLIME divided by SLIME_no's.

SLIME_no: SLIME without the Seed Replacement. Each evaluation lasts 48 hours and is repeated 20 times.

Seed Replacement contributes to the coverage performance

Discussion and Limitation

➢ Energy allocation between different stages

- SLIME mainly focuses on adaptively assigning mutation energy in the Exploitation Stage. How to make better use of different energy allocation strategies in the two stages and seek an energy allocation balance is an interesting future work.

➢ Further utilization of the estimated quality

- SLIME quantifies the estimated quality for each seed, which is calculated by its property values on the top 3 efficient properties. How to optimize the usage of the estimated quality could be a promising topic.

THANKS

Zhejiang University, NESA Lab

https://nesa.zju.edu.cn

SLIME: https://github.com/diewufeihong/SLIME