MOPT-Optimized Mutation Scheduling for fuzzers

USENIX ’19

Chenyang Lyu , Shouling Ji, Chao Zhang, Yuwei Li, Wei-Han Lee, Yu Song, and Raheem Beyah

mhkang 2022-07-27

Outline

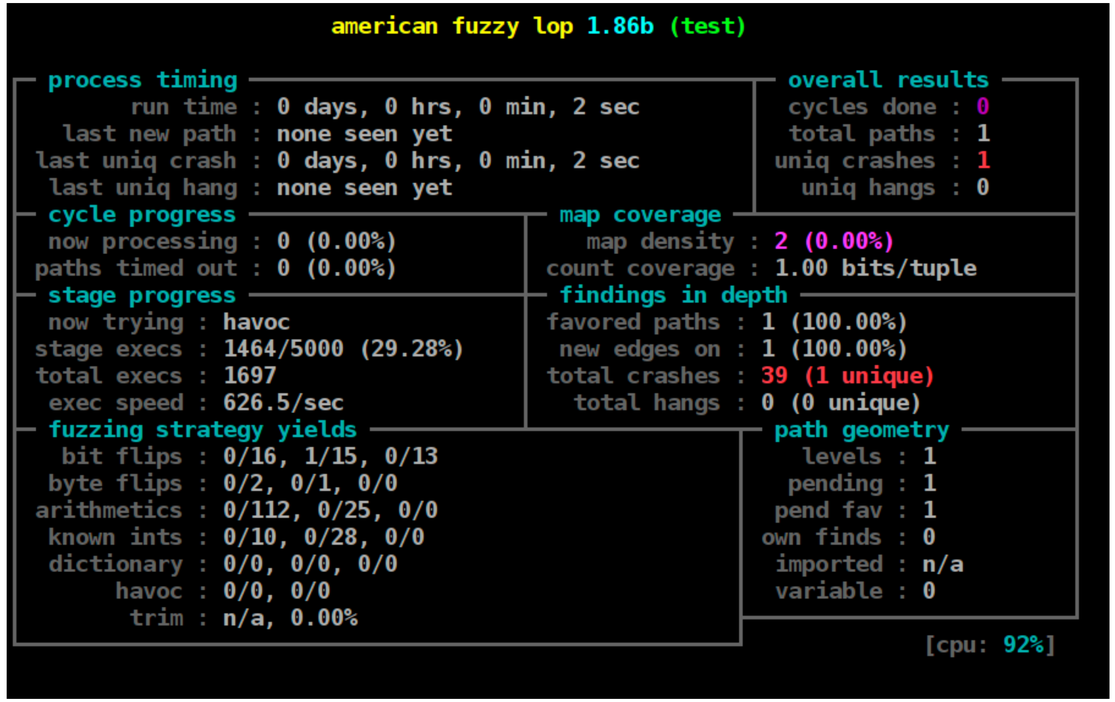
- Introduction
- Particle Swarm Optimization
- MOPT framework
- Evaluation
- Conclusion

What is fuzzing?

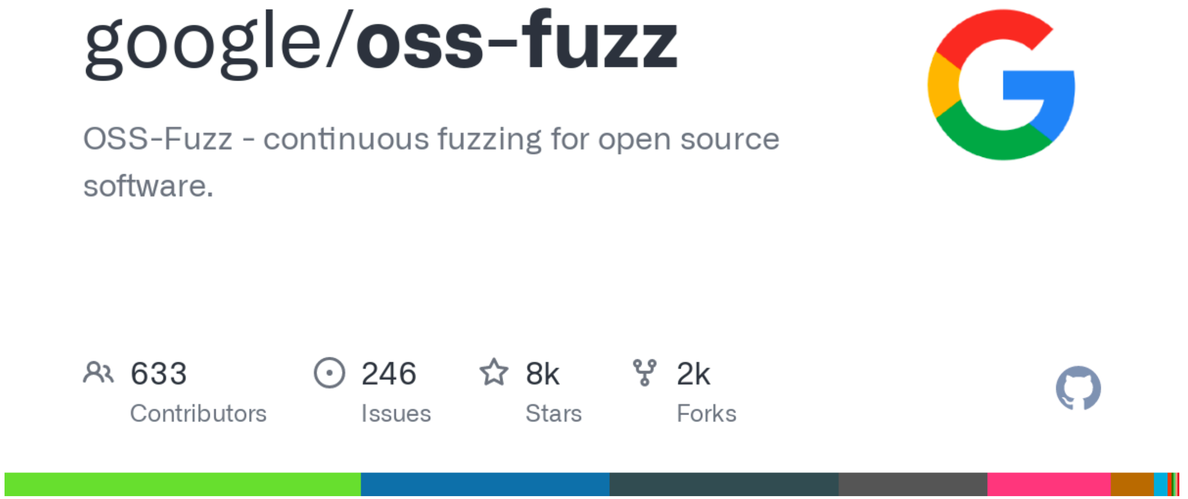
- A software testing technique for exploring vulnerabilities in software

As of June 2021, OSS-Fuzz has found over 30,000 bugs in 500 open source projects.

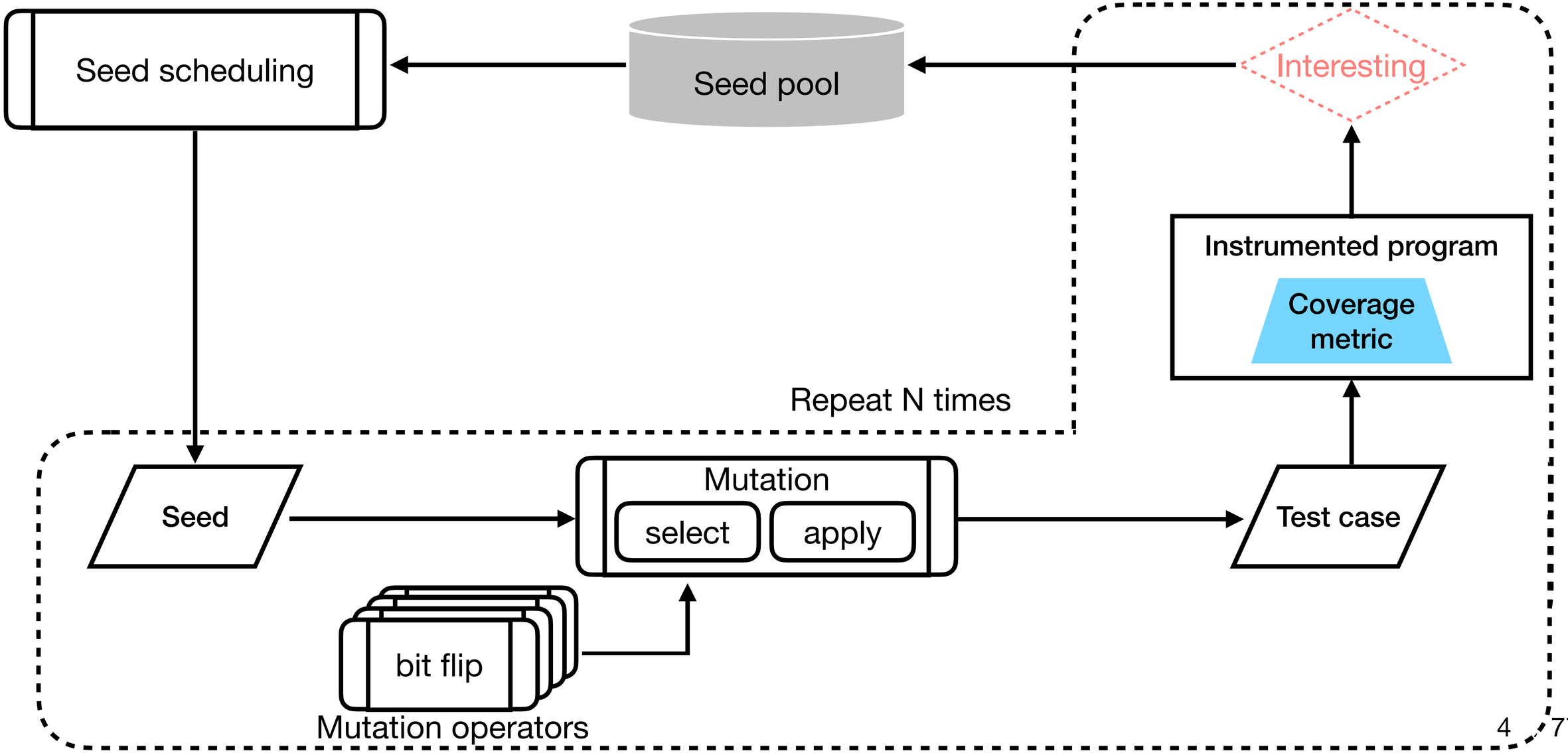
AFL



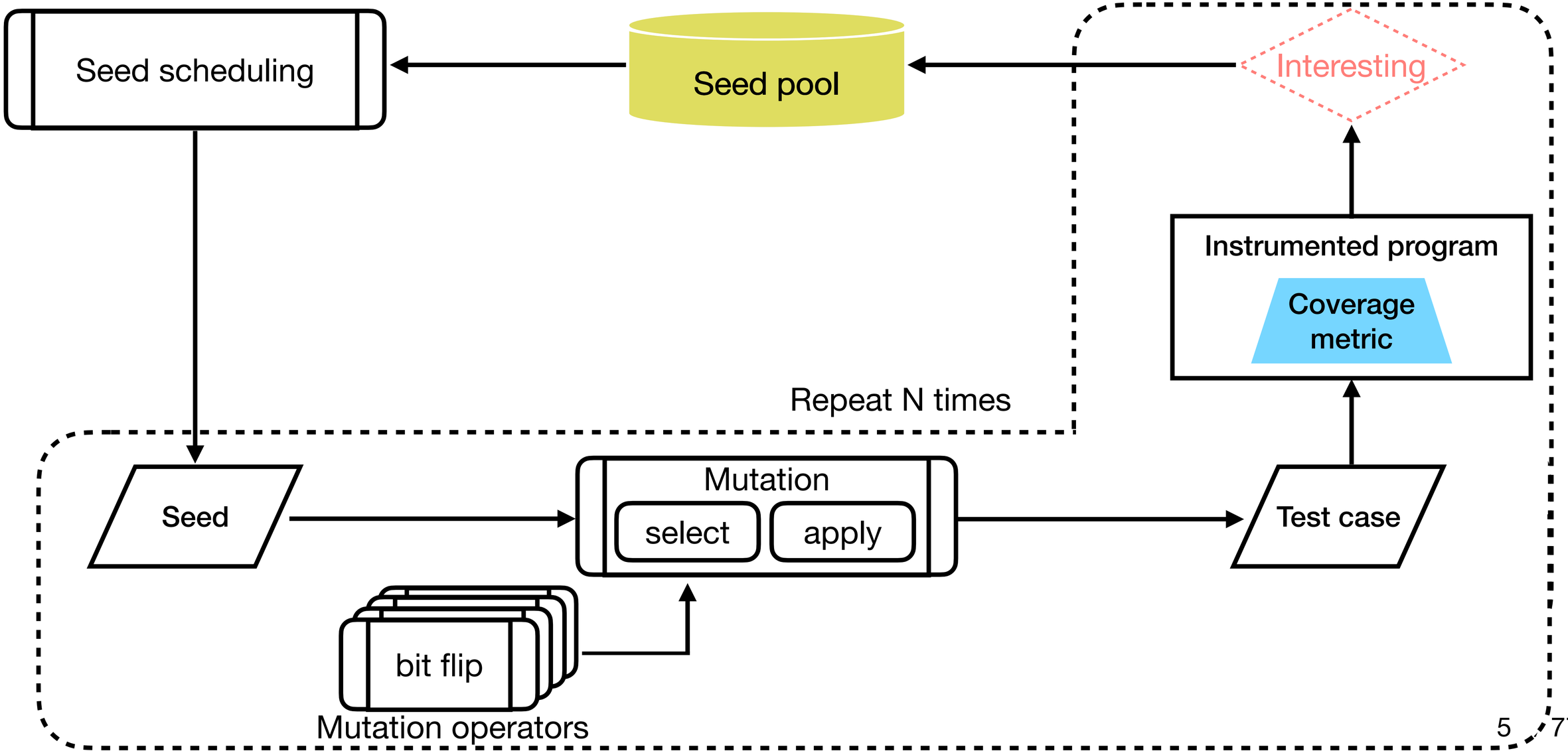
OSS-Fuzz



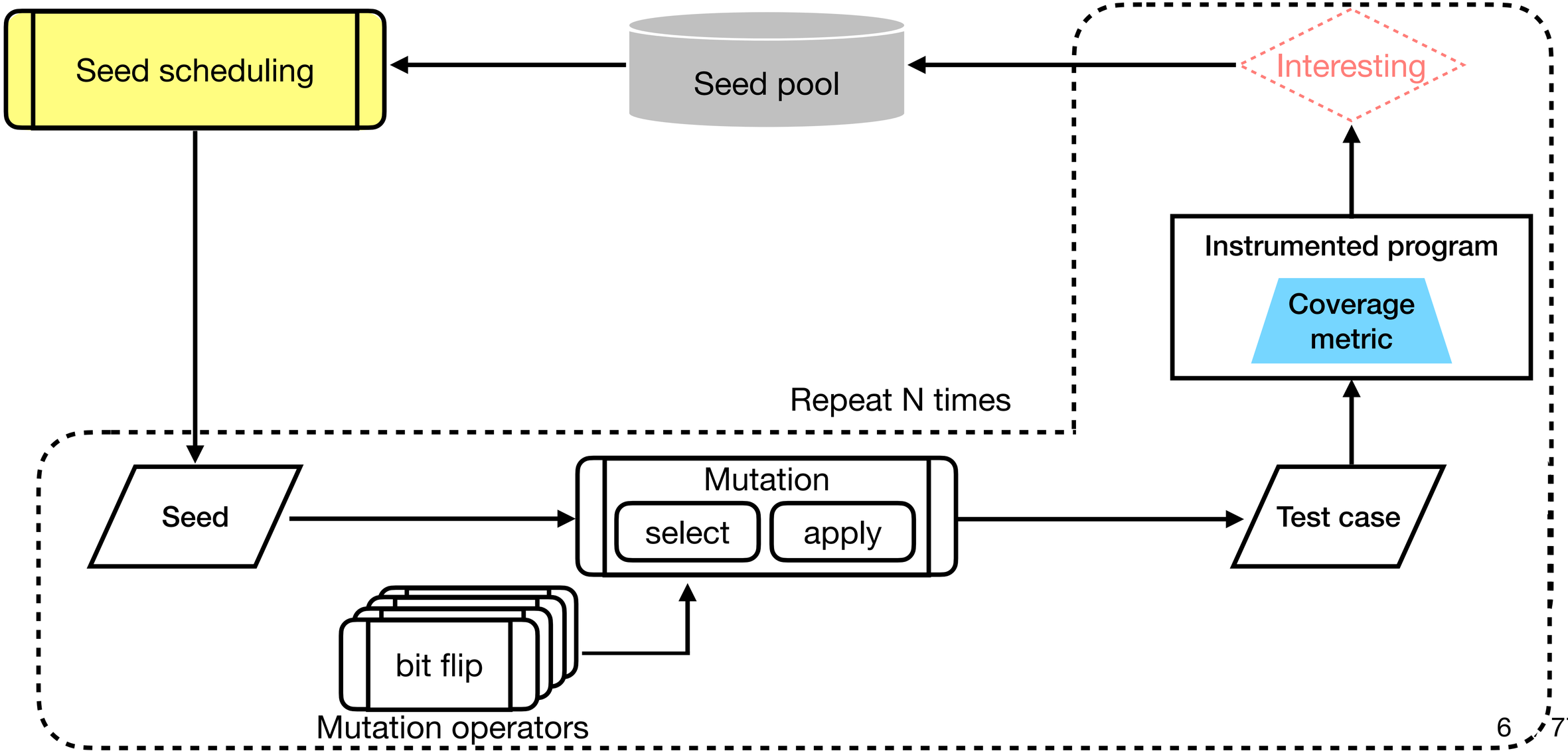
Mutation-based fuzzing



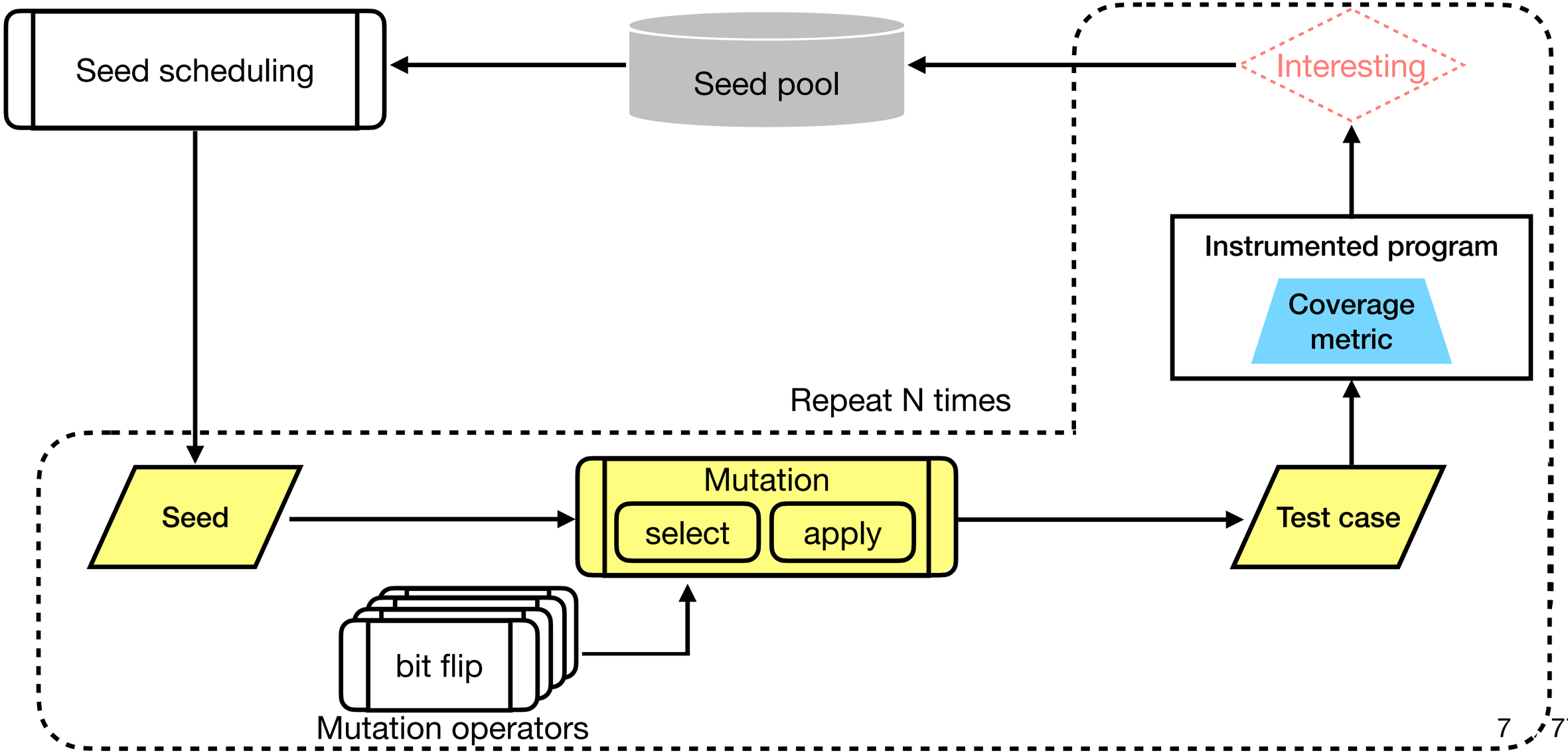
Mutation-based fuzzing



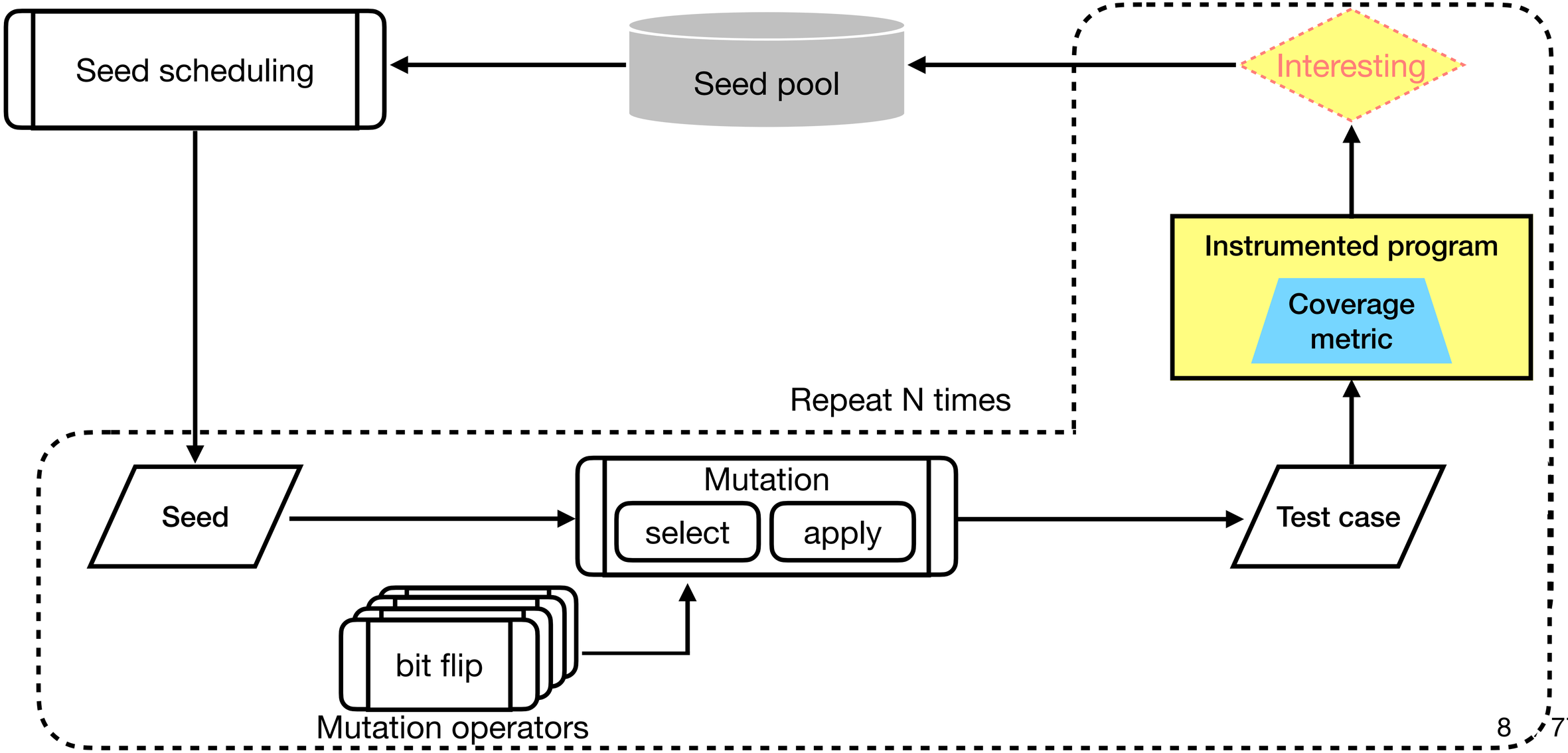
Mutation-based fuzzing



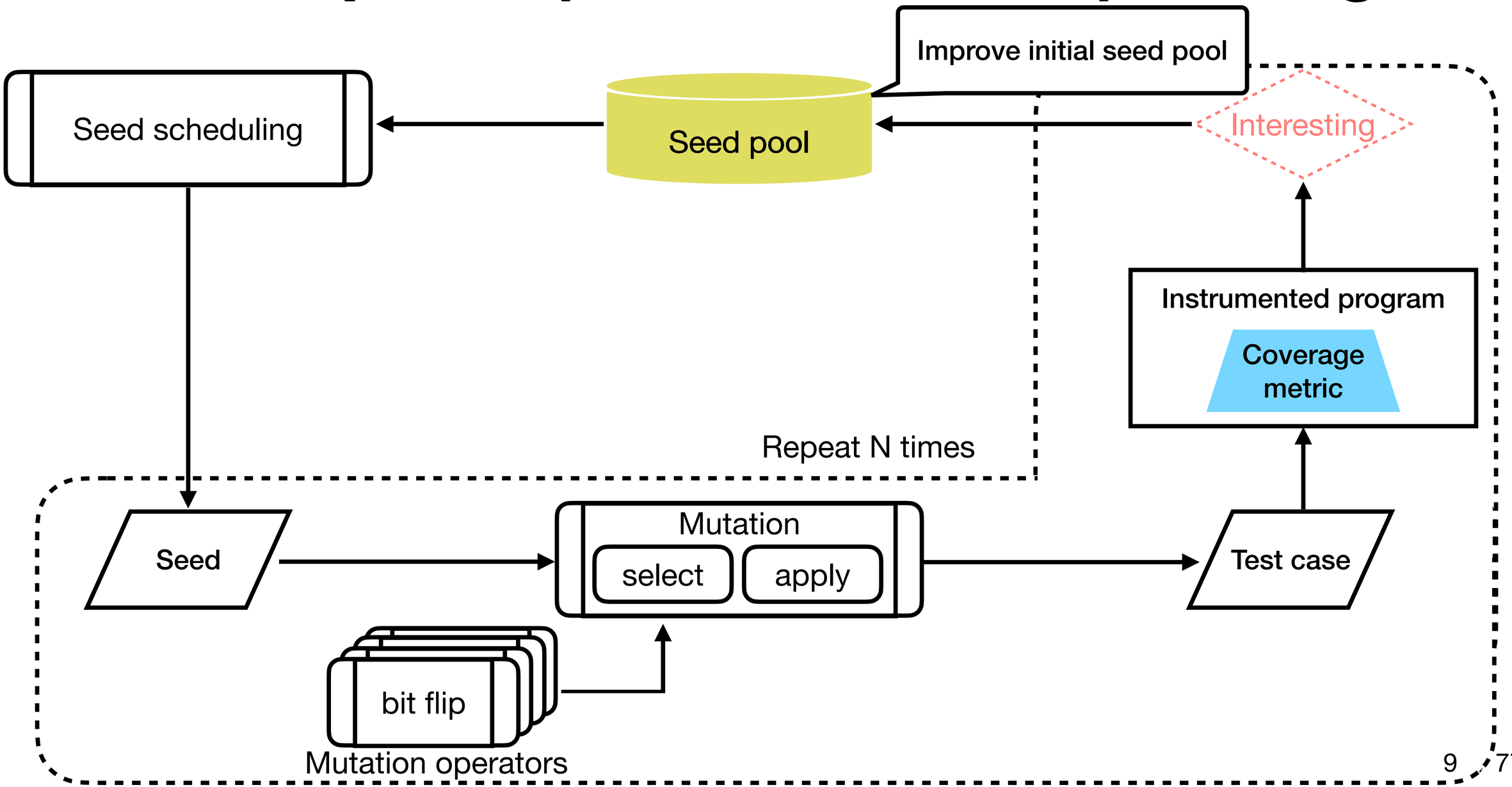
Mutation-based fuzzing



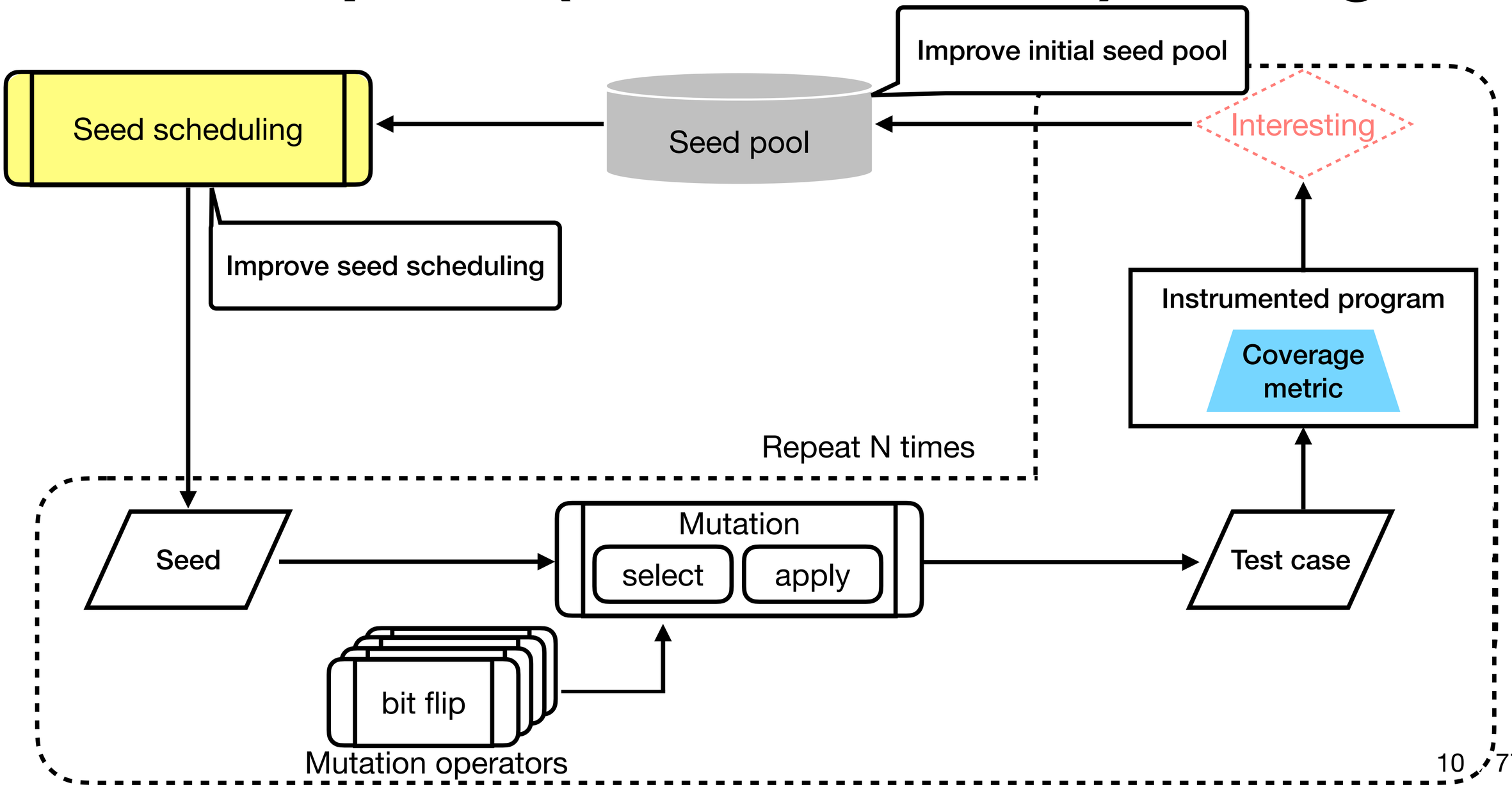
Mutation-based fuzzing



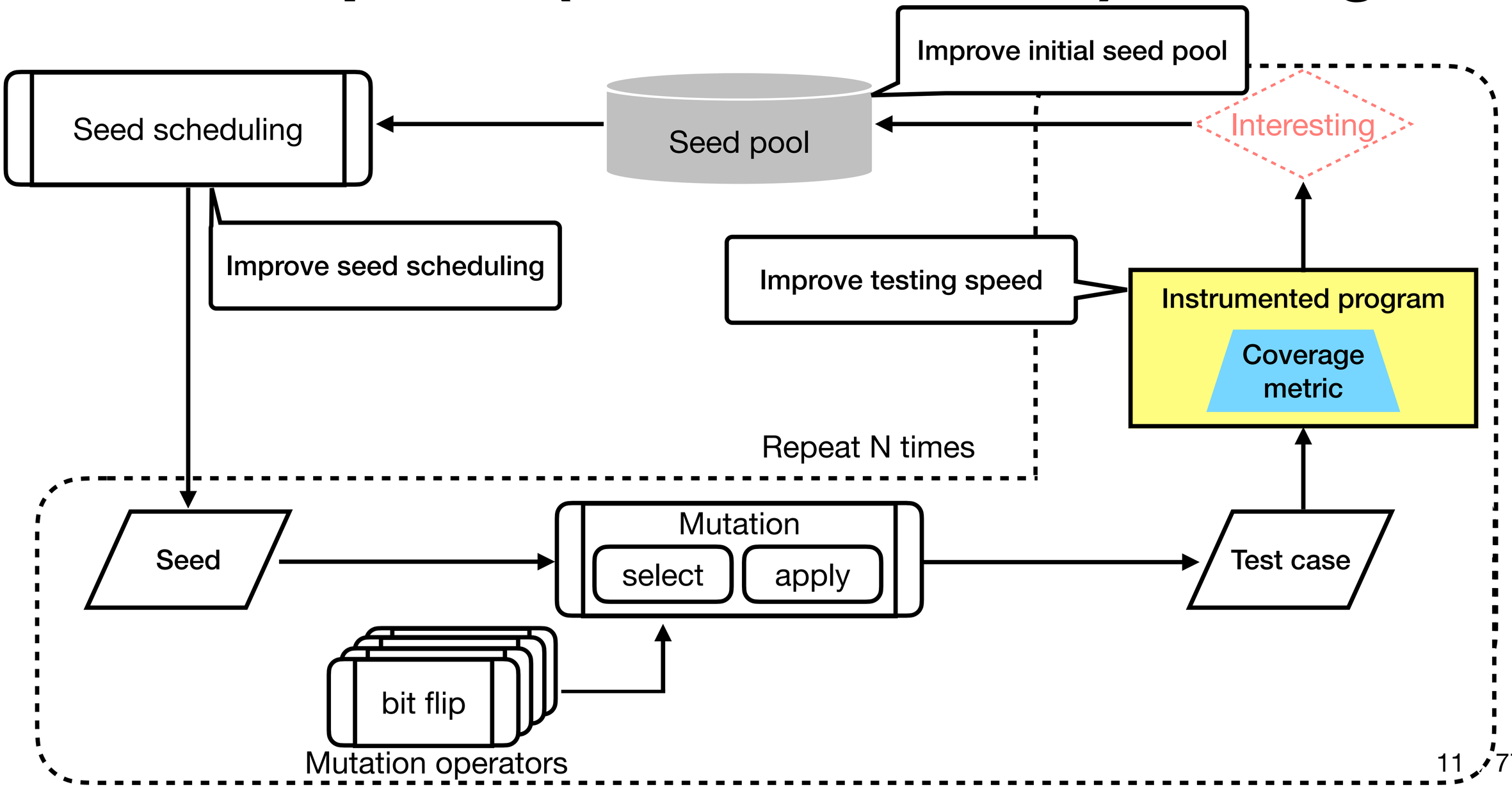
How to improve (mutation-based) fuzzing?



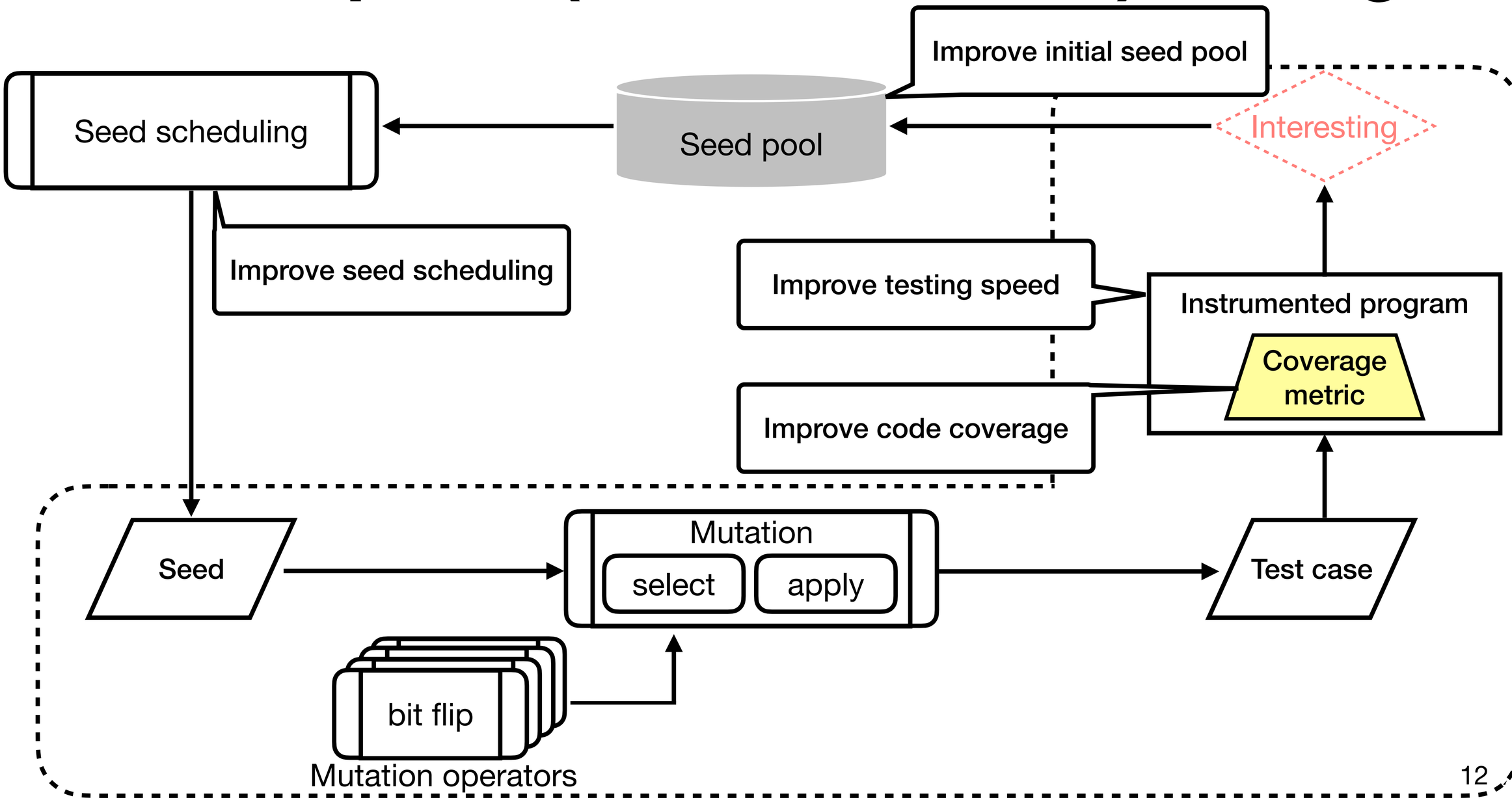
How to improve (mutation-based) fuzzing?



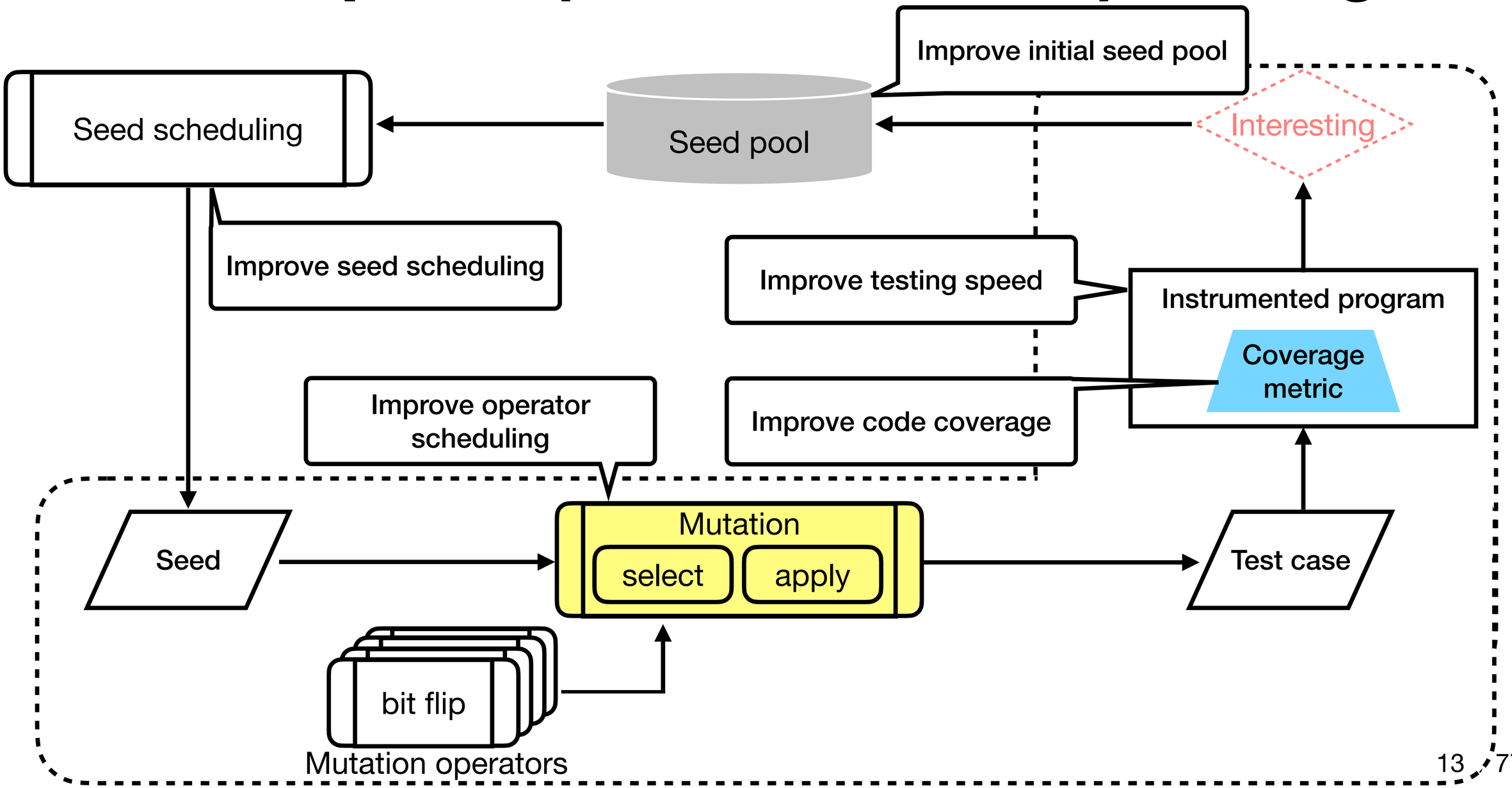
How to improve (mutation-based) fuzzing?



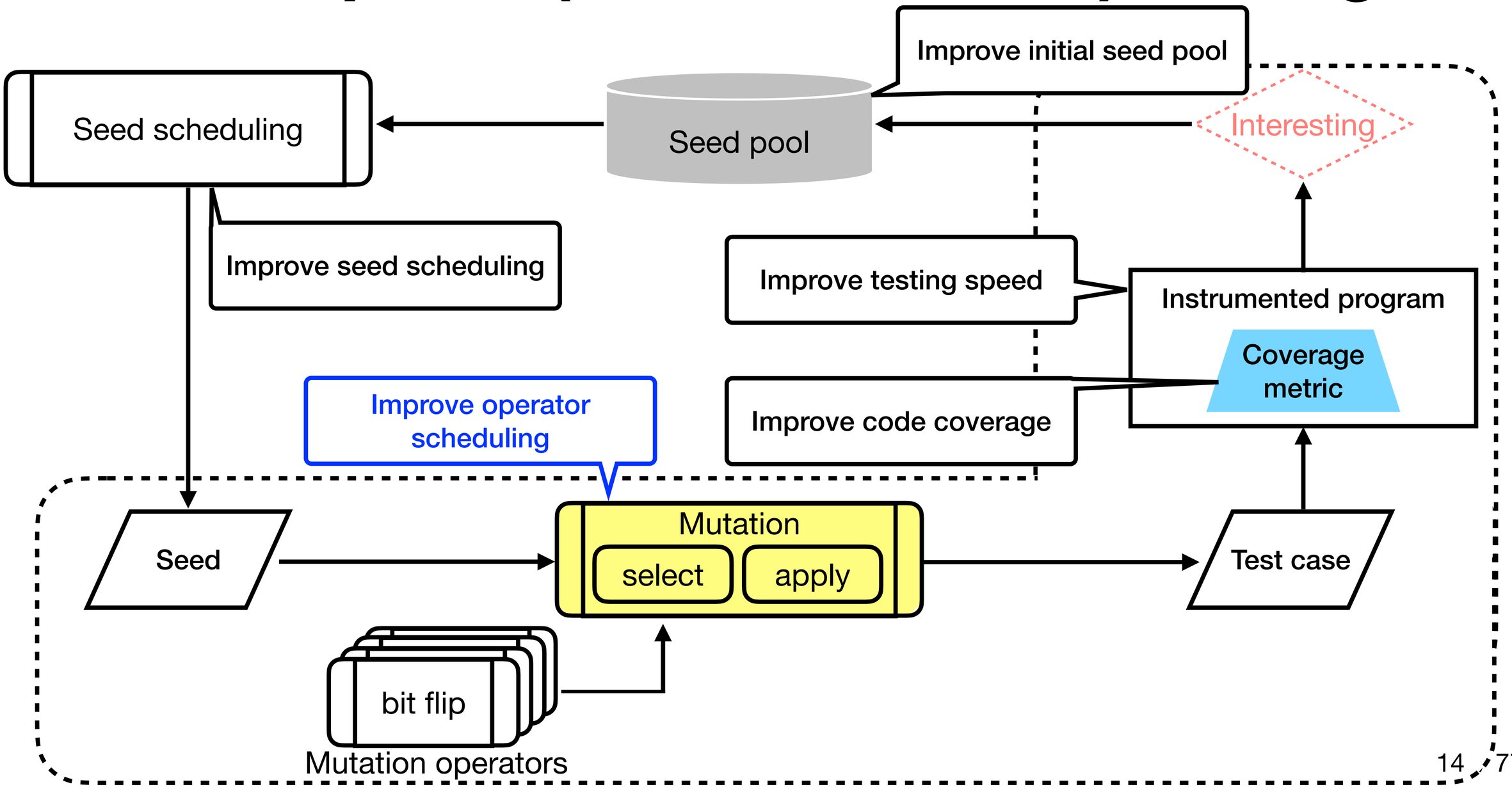
How to improve (mutation-based) fuzzing?



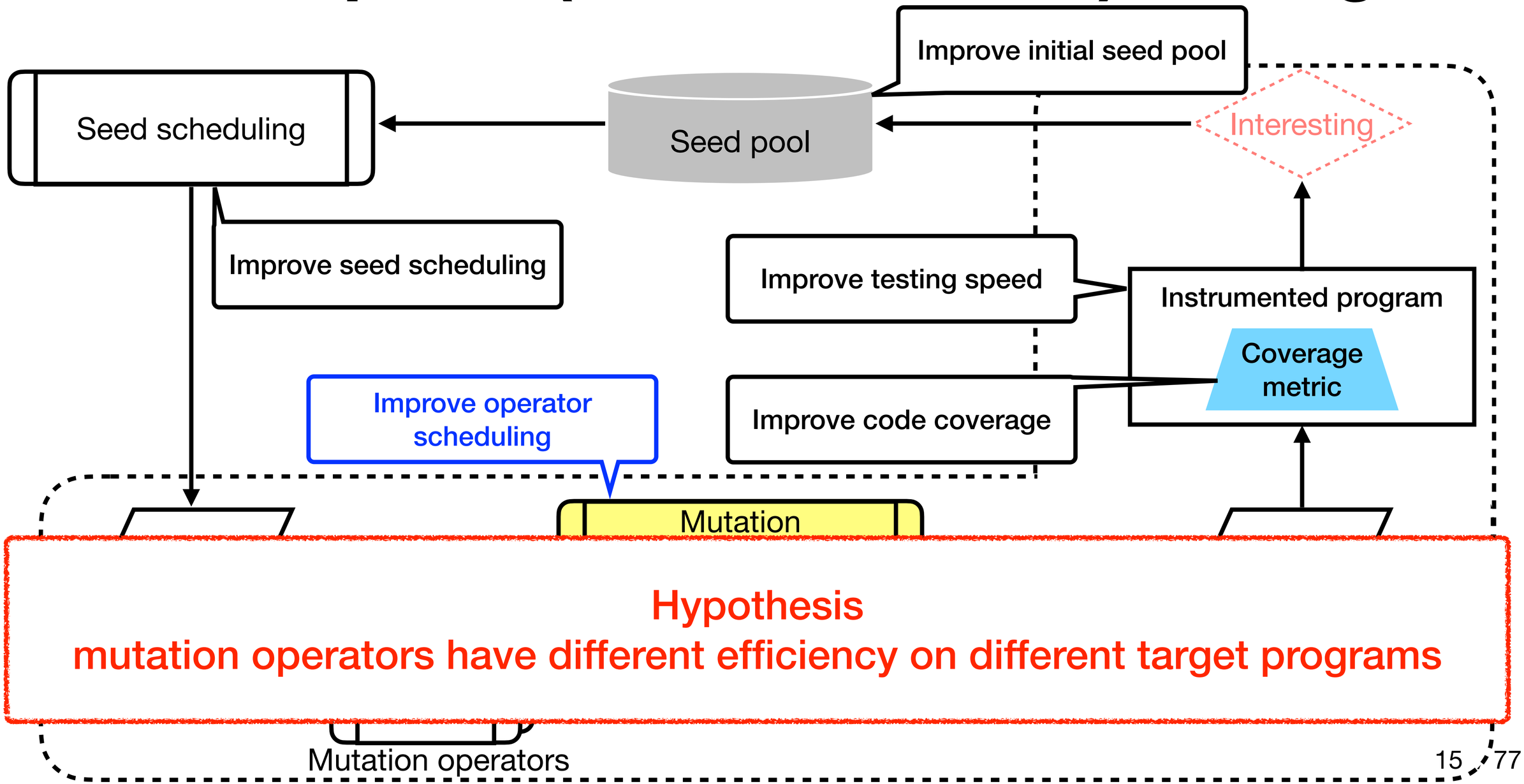
How to improve (mutation-based) fuzzing?



How to improve (mutation-based) fuzzing?



How to improve (mutation-based) fuzzing?



Mutation efficiency of AFL

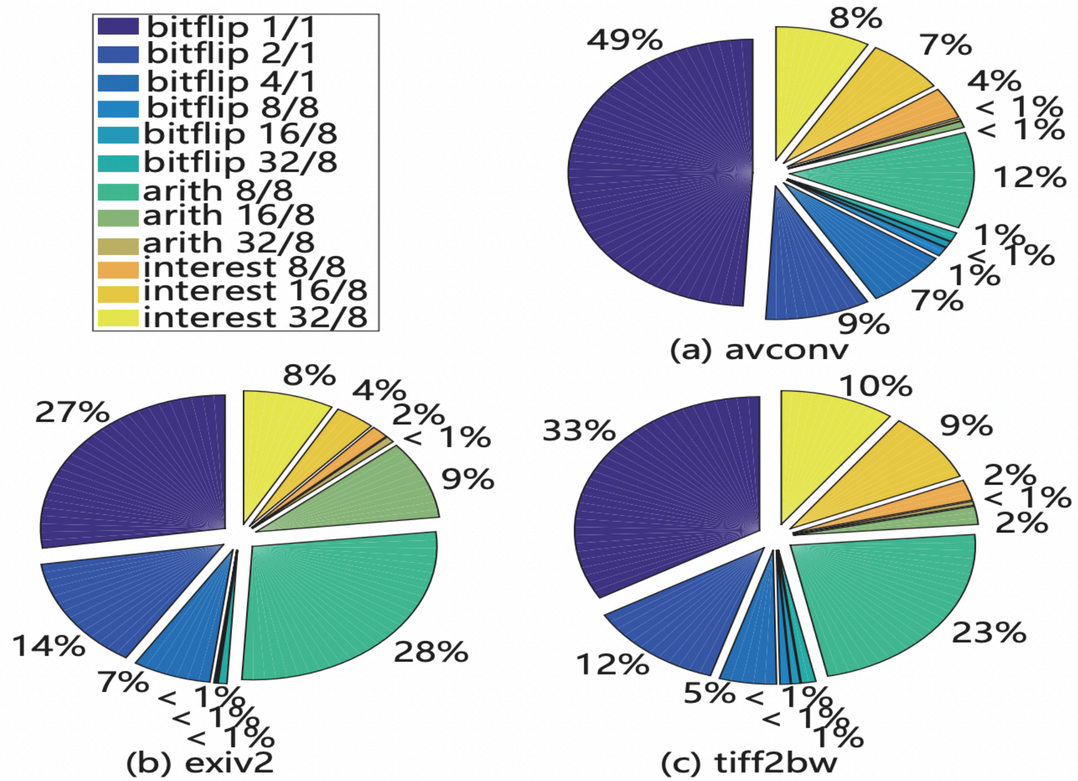


Figure 3: Percentages of interesting test cases produced by different operators in the deterministic stage of AFL.

Mutation efficiency of AFL

operators have different efficiency

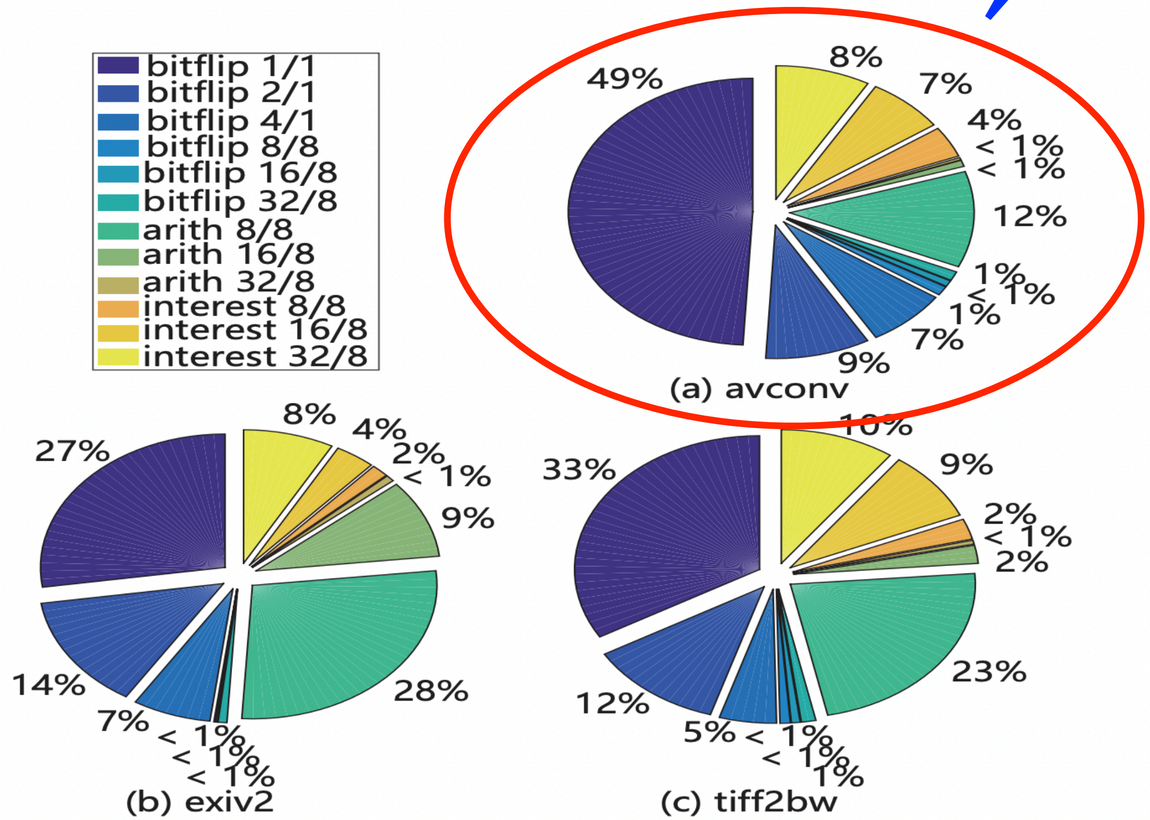


Figure 3: Percentages of interesting test cases produced by different operators in the deterministic stage of AFL.

Mutation efficiency of AFL

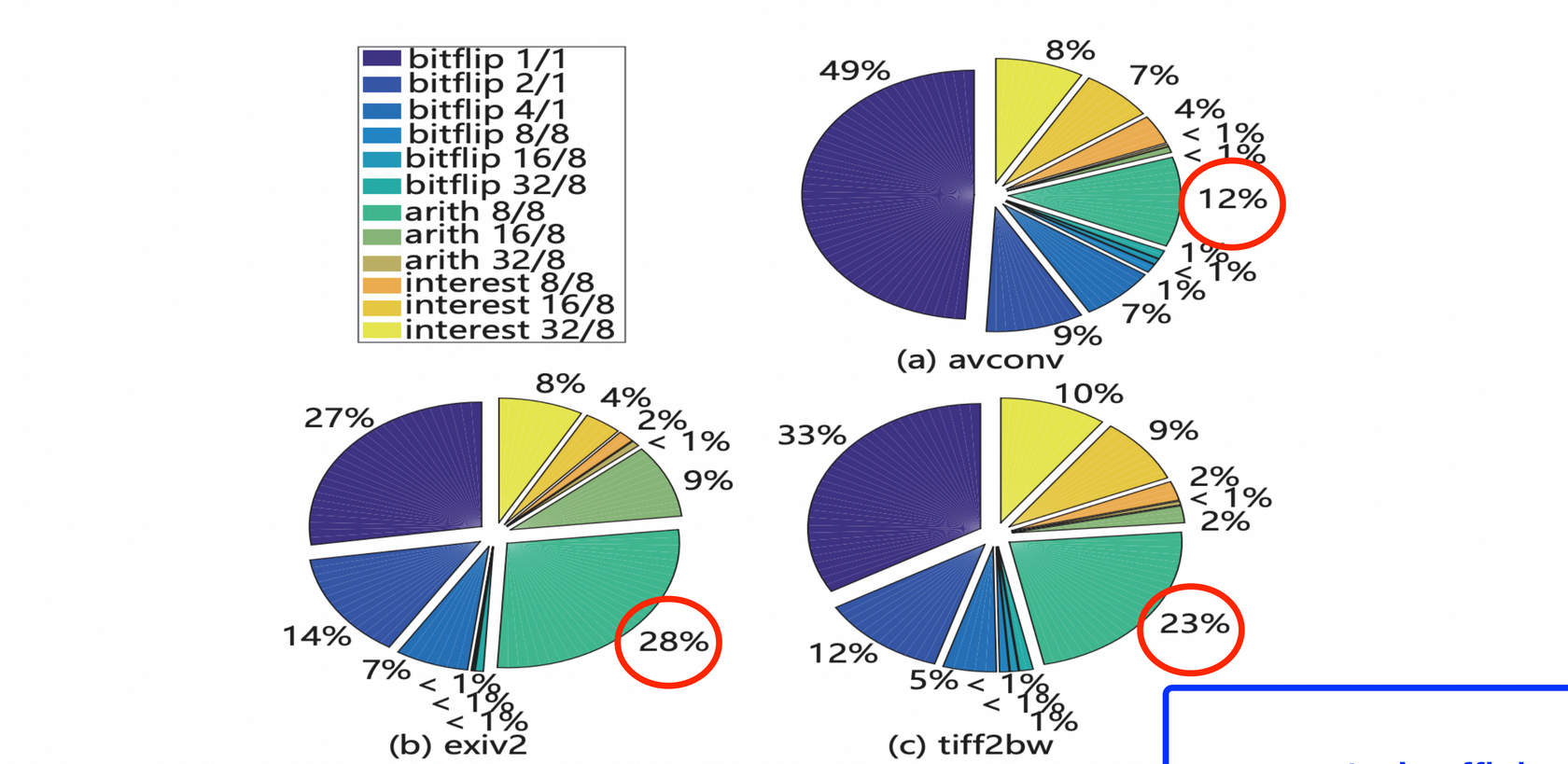


Figure 3: Percentages of interesting test case
different operators in the deterministic stage

an operator's efficiency varies with the target program

Mutation operators and scheduling

- Mutation operators characterize where and how to mutate the seed
  - For example, bit flip operators, in AFL, invert one or several consecutive bits

Mutation operators and scheduling

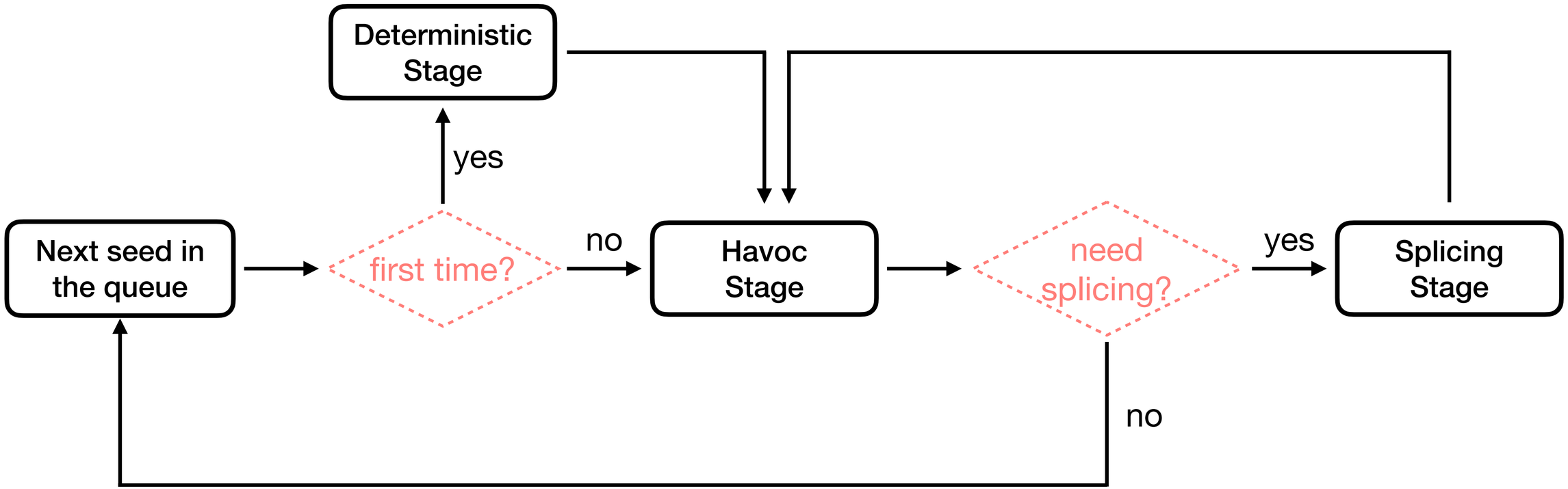
- Mutation operators characterize where and how to mutate the seed
  - For example, bit flip operators, in AFL, invert one or several consecutive bits

Mutation operators and scheduling

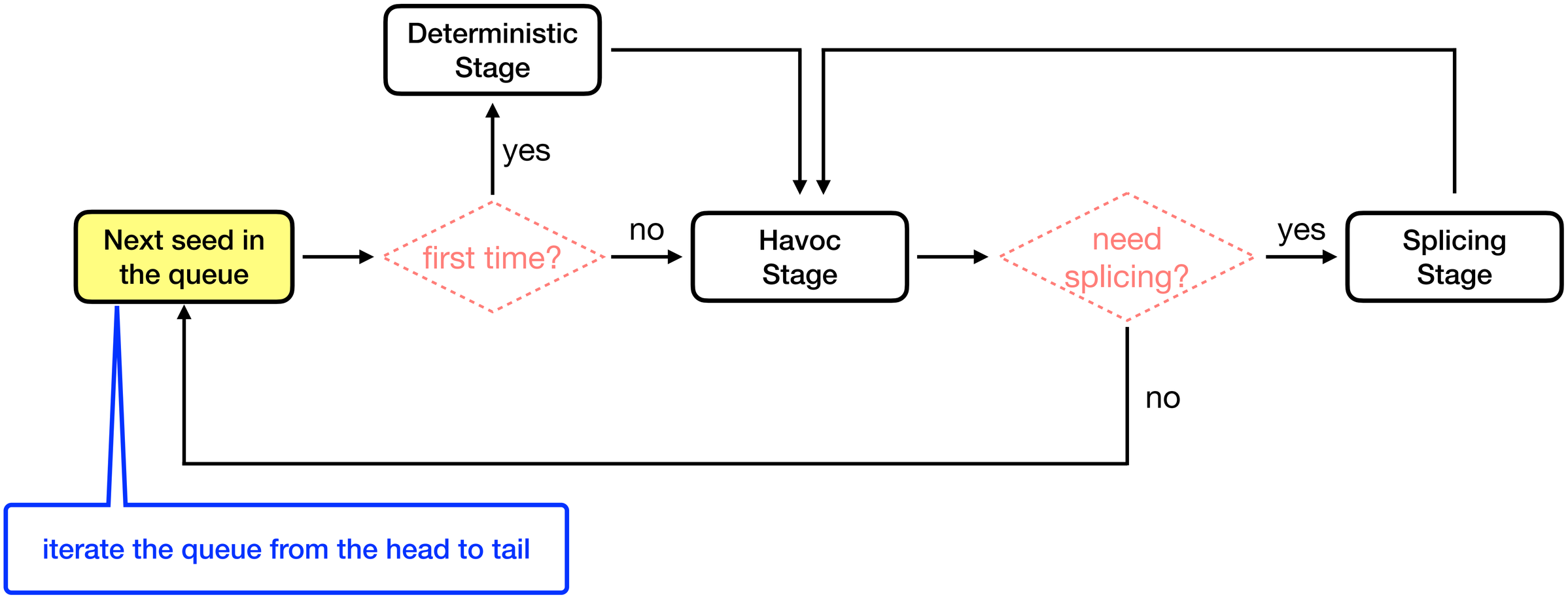
- Mutation operators characterize where and how to mutate the seed
  - For example, bit flip operators, in AFL, invert one or several consecutive bits

- And, the mutation scheduling scheme selects mutation operators

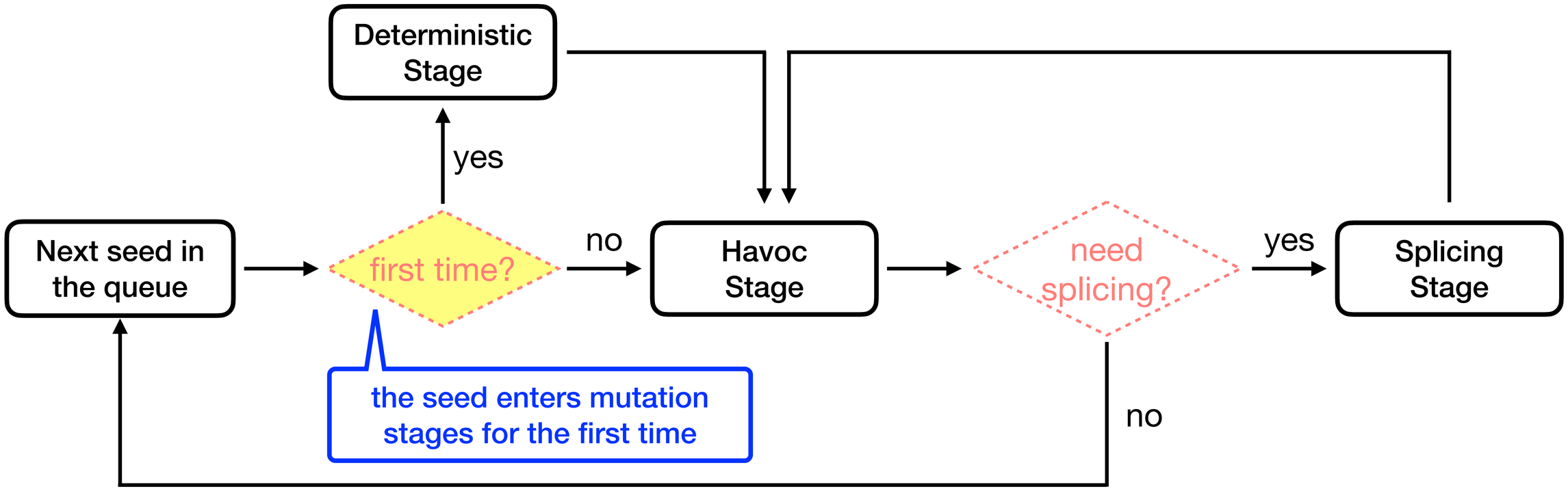
Mutation scheduling of AFL



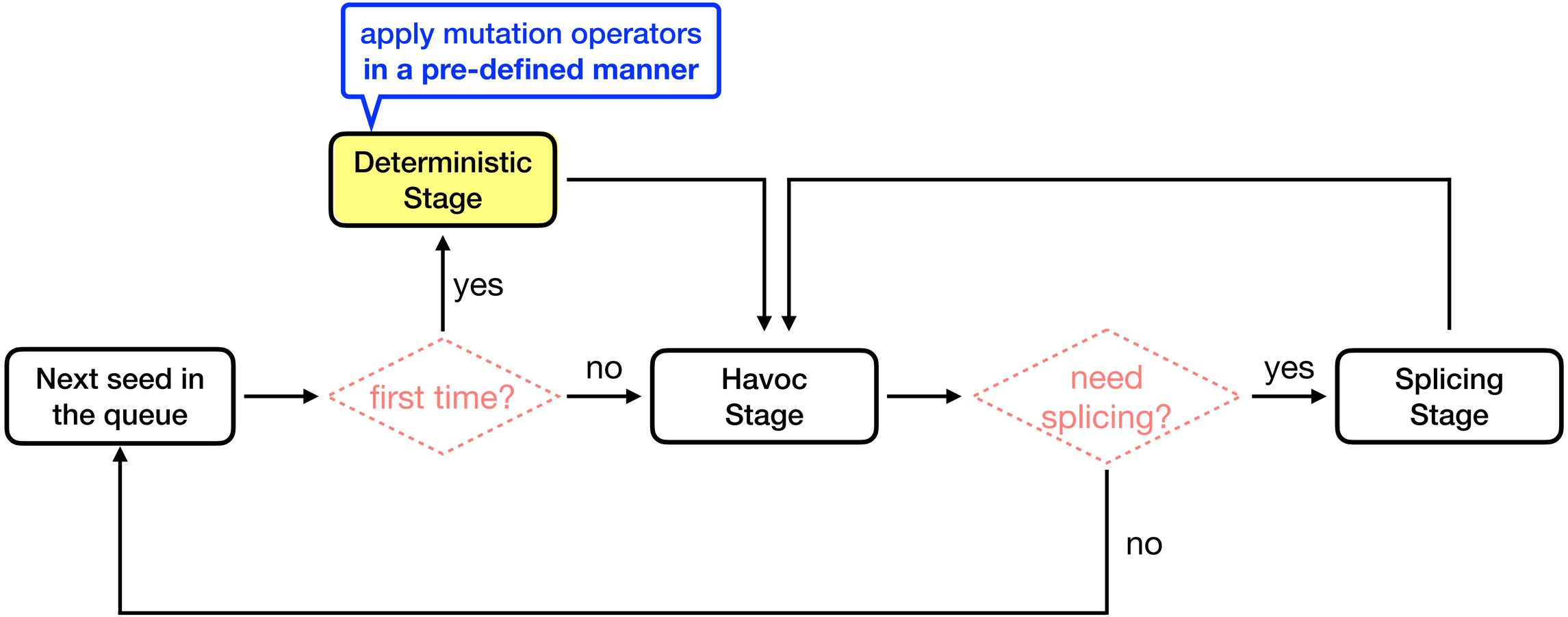
Mutation scheduling of AFL



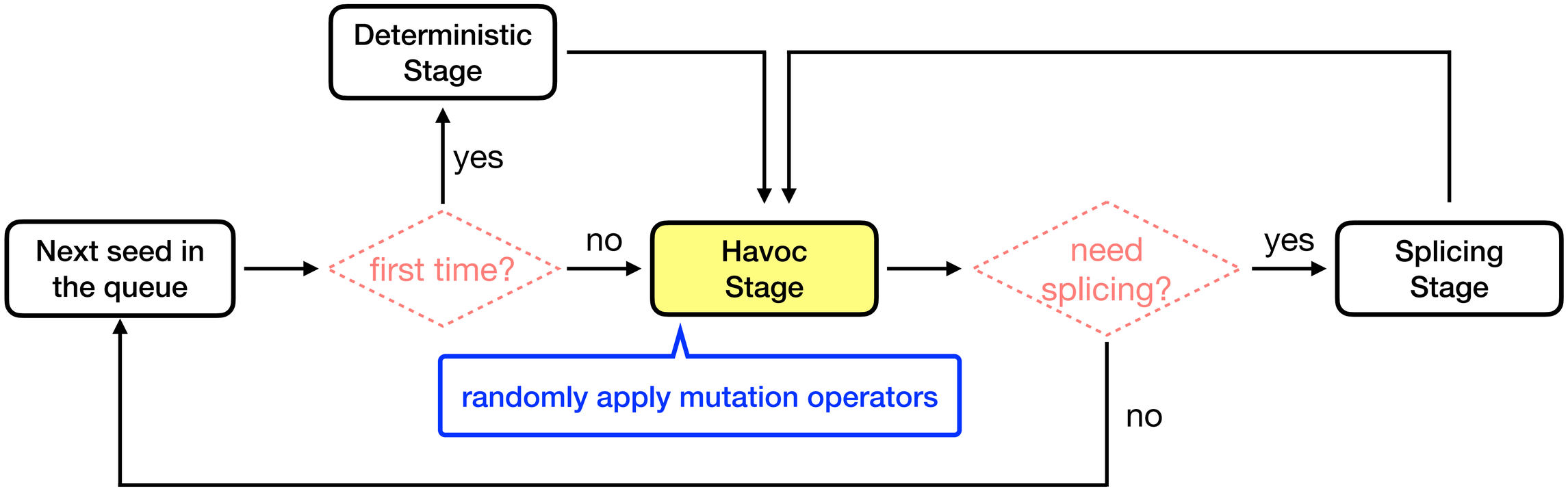
Mutation scheduling of AFL



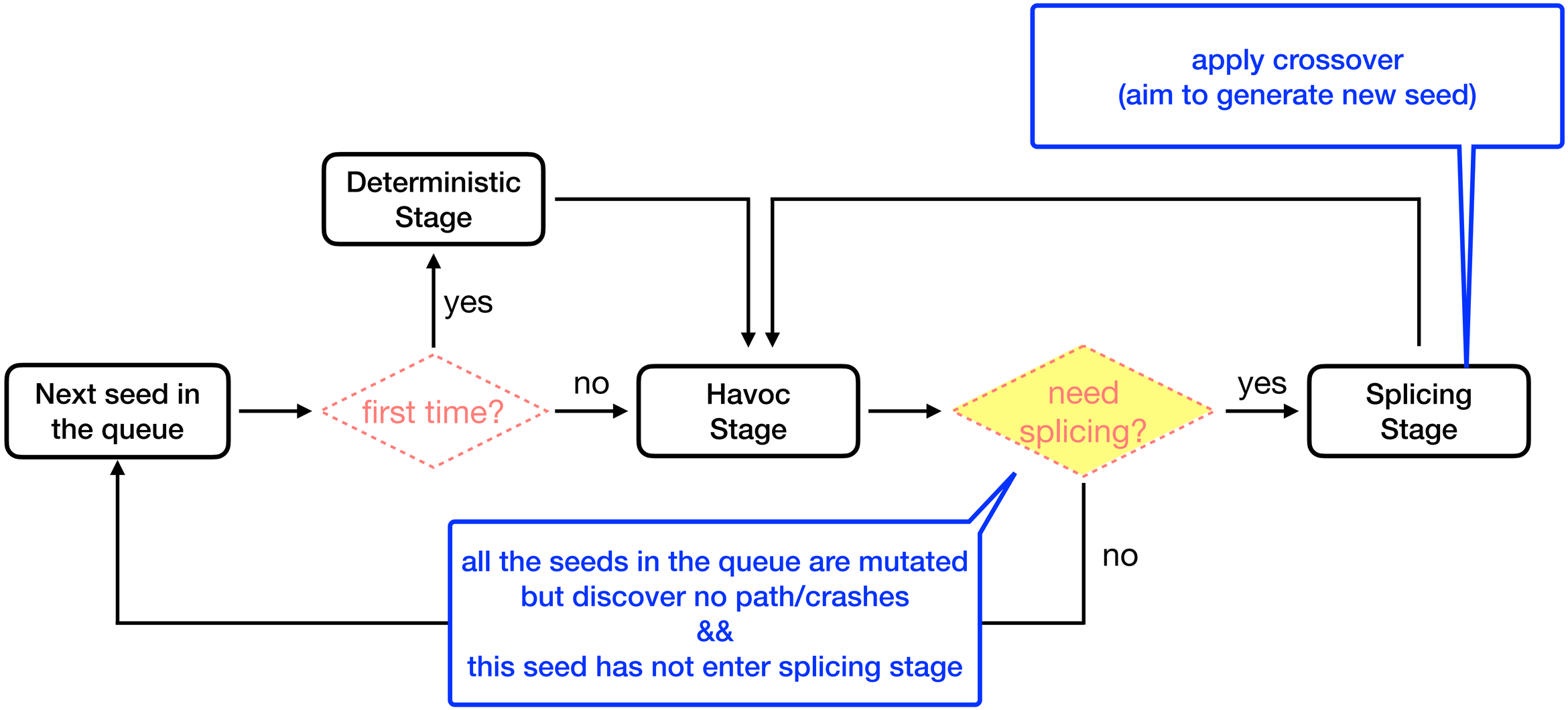
Mutation scheduling of AFL



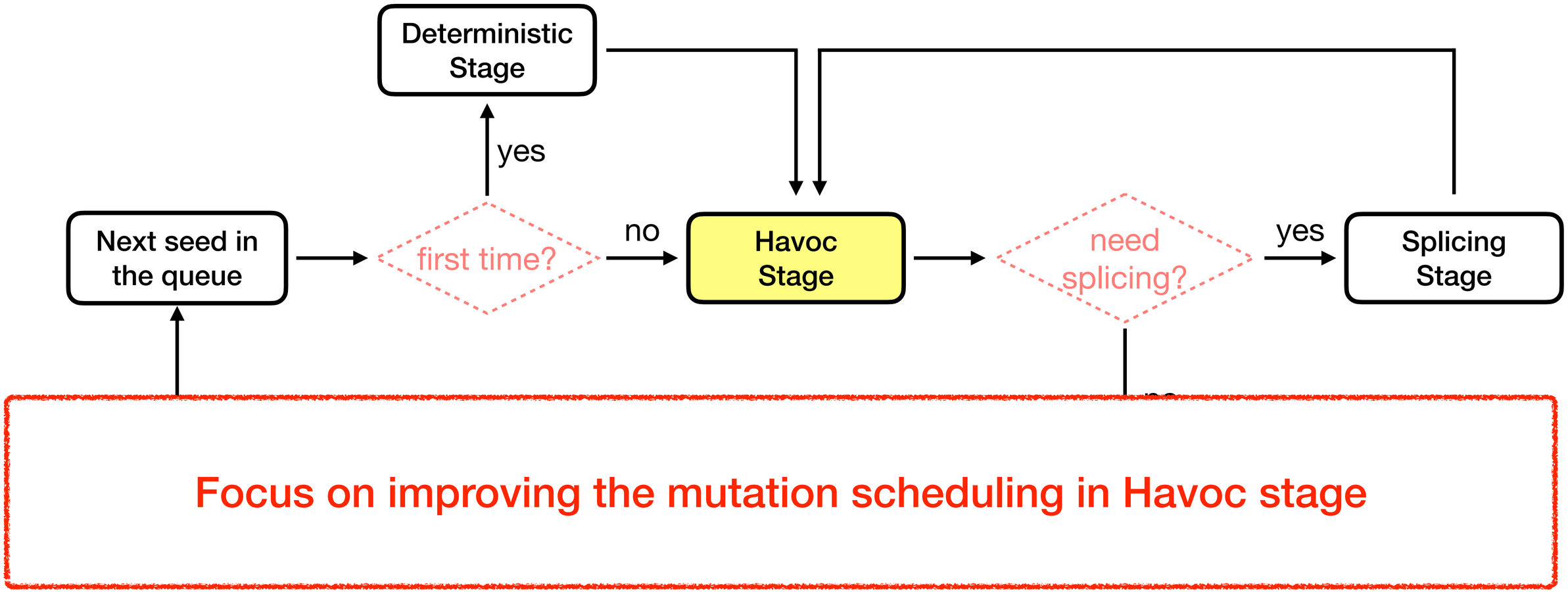
Mutation scheduling of AFL



Mutation scheduling of AFL



Mutation scheduling of AFL



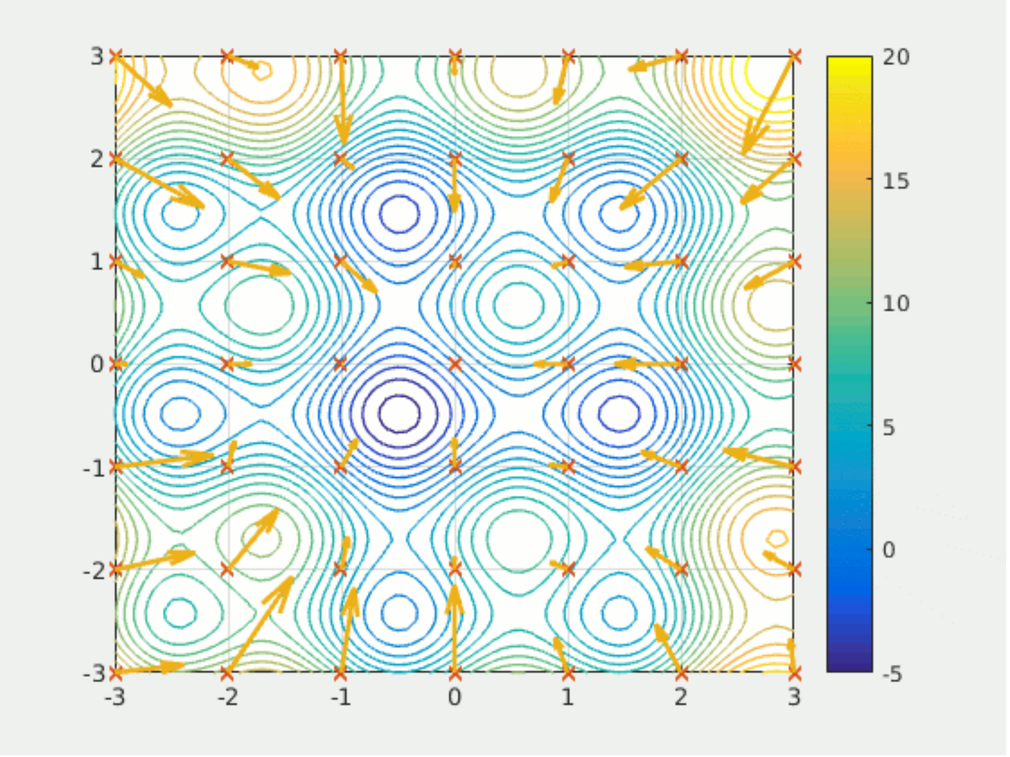
Optimizing mutation scheduling scheme

- Optimize mutation scheduling scheme by **finding an optimal probability distribution of mutation operators** for the target program
  - the mutation scheduler chooses next operators by following the distribution

- This paper proposes MOPT which optimizes the probability distribution of mutation operators
  - the optimization process is inspired by Particle Swarm Optimization (PSO) algorithm

Particle Swarm Optimization (PSO)

- PSO algorithm
  - employs multiple particles to search the solution space iteratively
  - each particle p in the swarm, it moves towards its local best position and global best position in each iteration



Particle Swarm Optimization (PSO)

- Solution Representation: a particle with **position** x_i^t and **velocity** v_i^t

each **particle is described by its position and velocity**
(Note: t = current time step)

Particle Swarm Optimization (PSO)

- Solution Representation: a particle with **position** x_i^t and **velocity** v_i^t

- Initialization: $x_i^0, v_i^0 \in [lb, ub]$

Initial position and velocity is randomly sampled from uniform distribution bounded with ***lb* (lower bound) and *ub* (upper bound)**
Note: *lb* and *ub* is a hyper-parameters, i.e., a user need to manually set these values

Particle Swarm Optimization (PSO)

- Solution Representation: a particle with **position** x_i^t and **velocity** v_i^t
- Initialization: $x_i^0, v_i^0 \in [lb, ub]$
- Solution update
  - position update: $x_i^{(k+1)} = x_i^k + v_i^{(k+1)}$

In each time step, the position of each particle is **updated by adding the distance that the particle moves in a unit time** ($v_i^{(k+1)} = 1 \times v_i^{(k+1)}$) **with the "updated" velocity**

Particle Swarm Optimization (PSO)

- Solution Representation: a particle with **position** x_i^t and **velocity** v_i^t
- Initialization: $x_i^0, v_i^0 \in [lb, ub]$
- In each time step, a velocity of each particle is **updated using its local best position p_i and the global best position g**
  - velocity update:

$$v_i^{(k+1)} = v_i^k + r_1(p_i - x_i^k) + r_2(g - x_i^k)$$
$$where \quad r_1 \sim U(0, \phi_1), \quad r_2 \sim U(0, \phi_2)$$

Velocity update in PSO

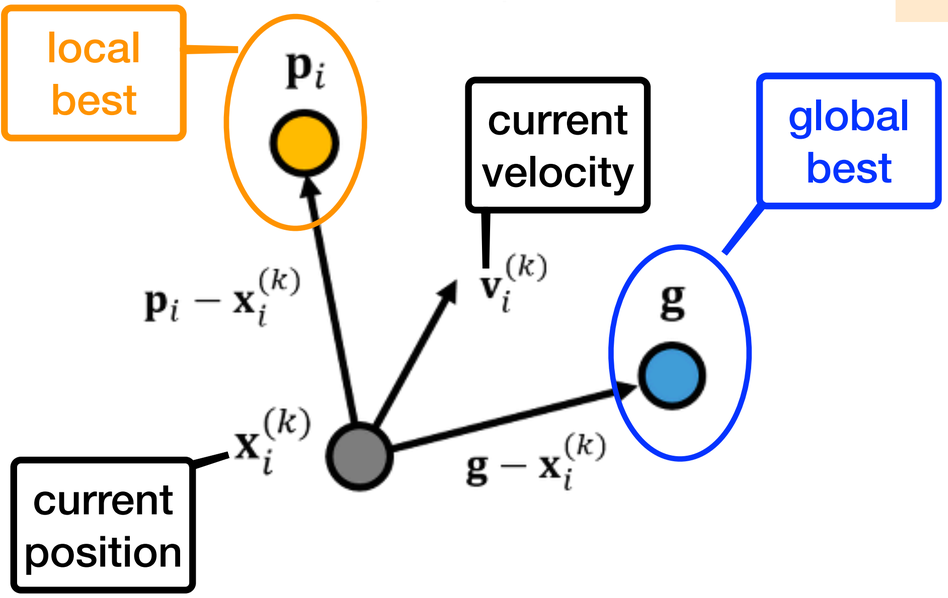
Velocity update:

$$v_i^{(k+1)} = v_i^k + r_1(p_i - x_i^k) + r_2(g - x_i^k)$$

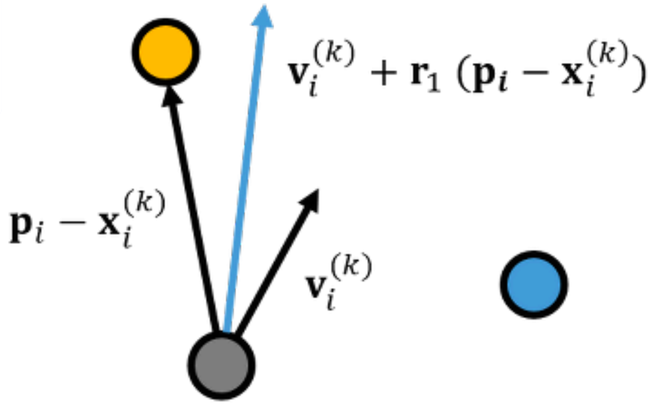
$$where \quad r_1 \sim U(0, \phi_1), \quad r_2 \sim U(0, \phi_2)$$

weights are randomly drawn from uniform distributions

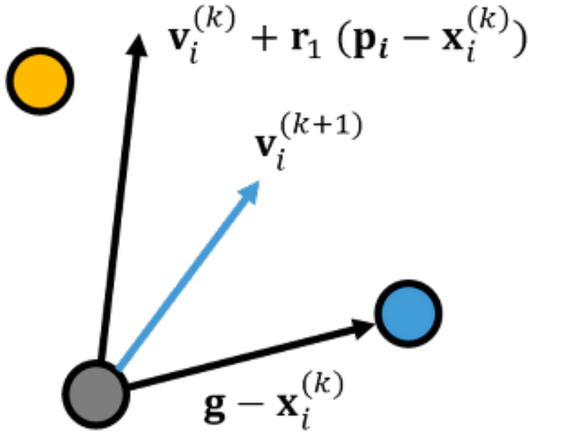
a. Velocity components



b. Applying individual influence



c. Applying swarm influence



$$\mathbf{v}_i^{(k+1)} = \mathbf{v}_i^{(k)} + \mathbf{r}_1(\mathbf{p}_i - \mathbf{x}_i^{(k)}) + \mathbf{r}_2(\mathbf{g} - \mathbf{x}_i^{(k)})$$

Customized PSO algorithm in MOPT

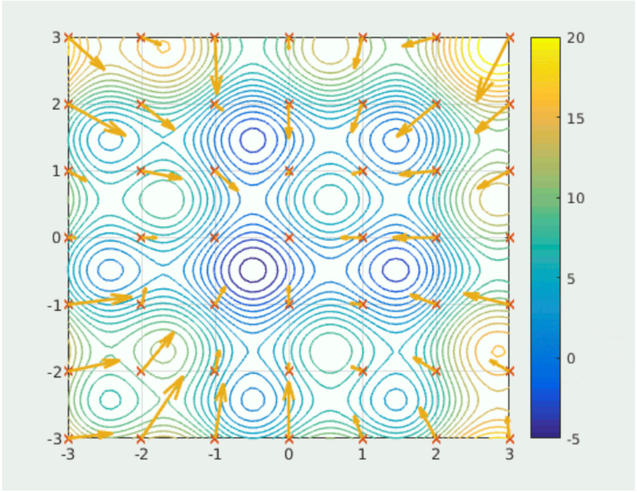
- MOpt adopts PSO to optimize the probability distribution of mutation operators

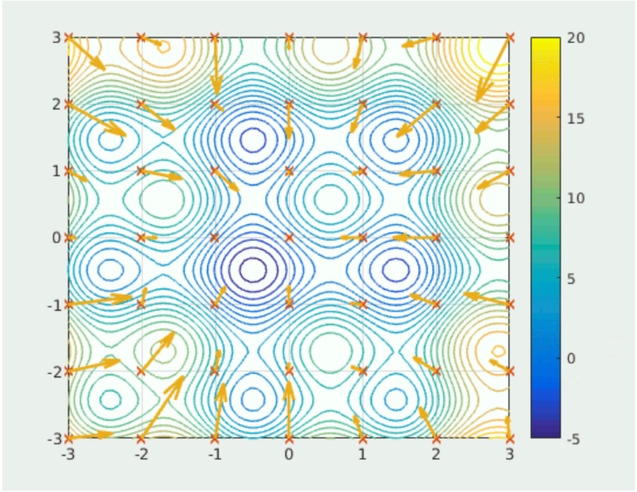
- In MOpt, a particle is a mutation operator and a swarm, a set of particles, represents a solution
  - In PSO, a particle is a position which represents a solution

- The core idea of PSO is to optimize multiple solutions concurrently using the shared information discovered by each solution
  - Therefore, MOpt needs to optimize multiple swarms whereas PSO optimizes a single swarm

Customized PSO algorithm in MOPT

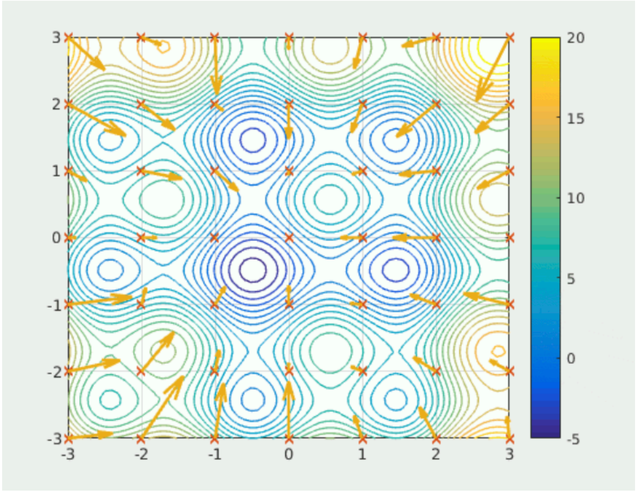
technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n

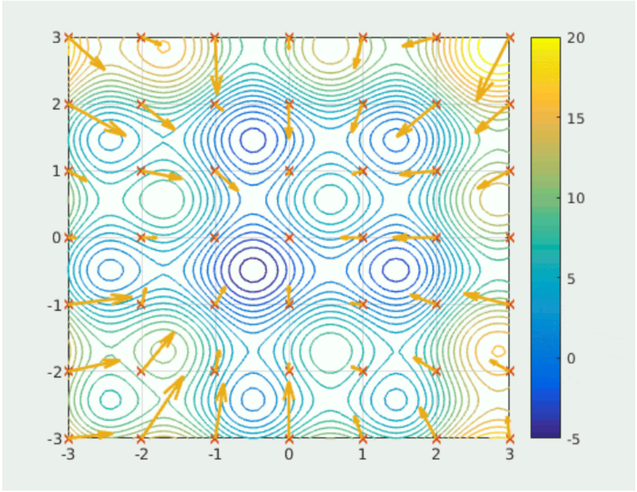




Customized PSO algorithm in MOPT

technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n

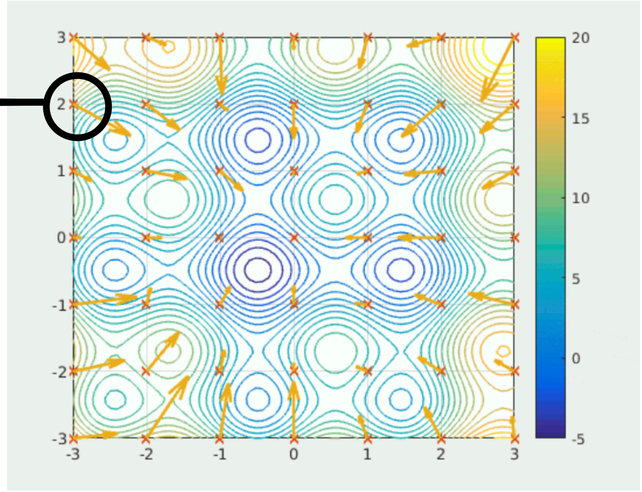


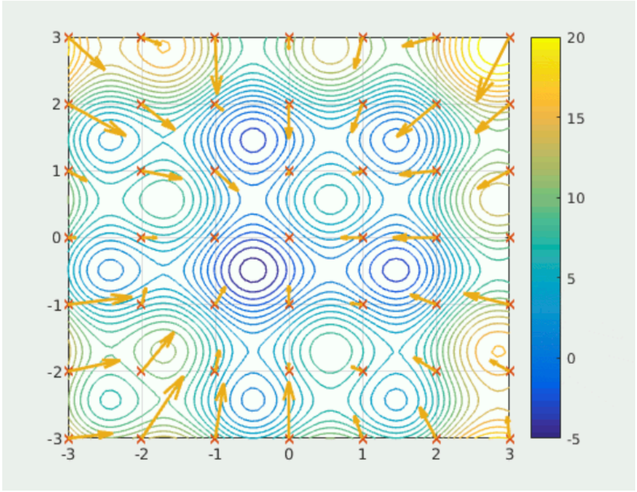


Customized PSO algorithm in MOPT

technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n

position

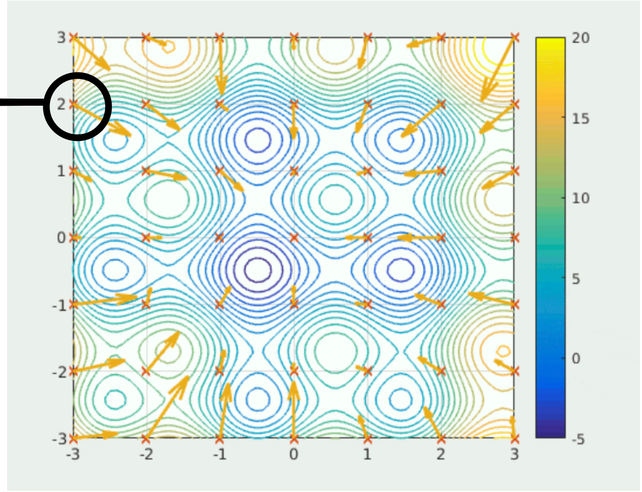


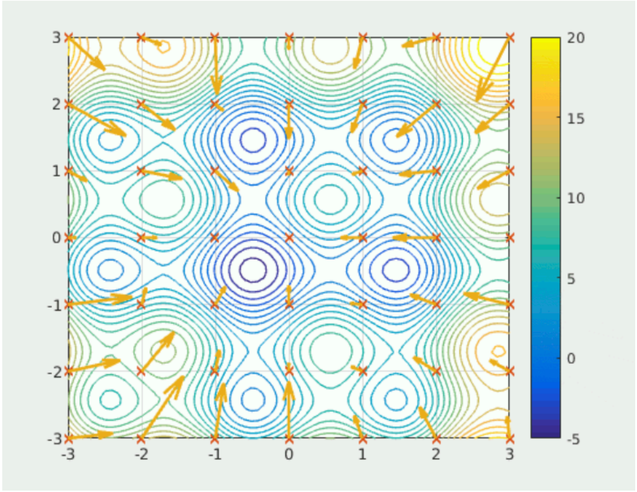


Customized PSO algorithm in MOPT

technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n

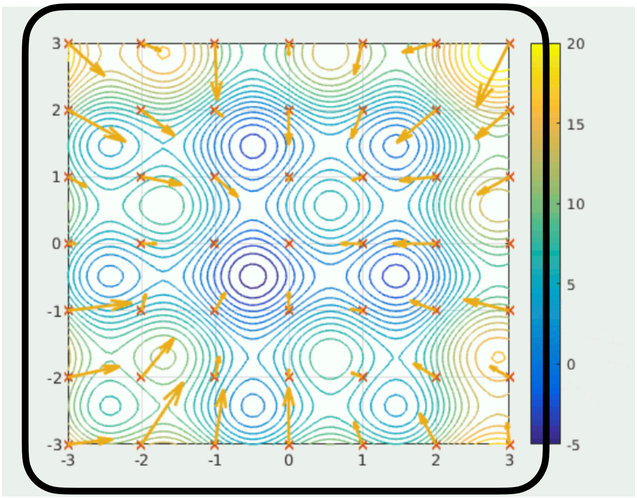
particle = position

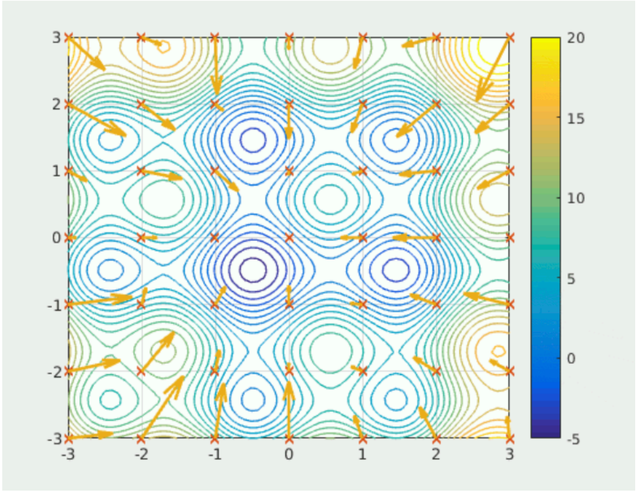




Customized PSO algorithm in MOPT

technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n

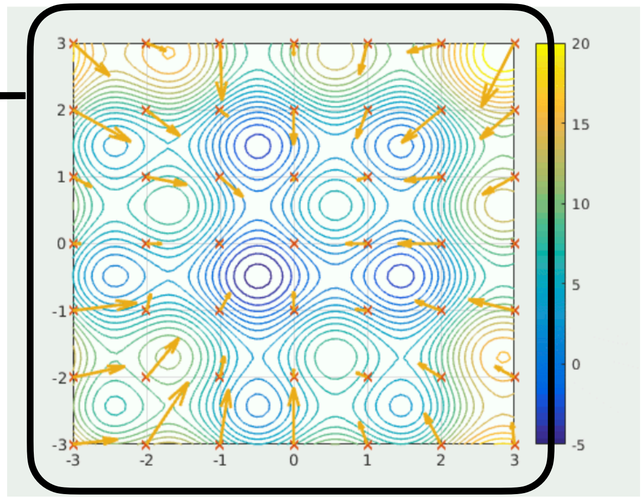


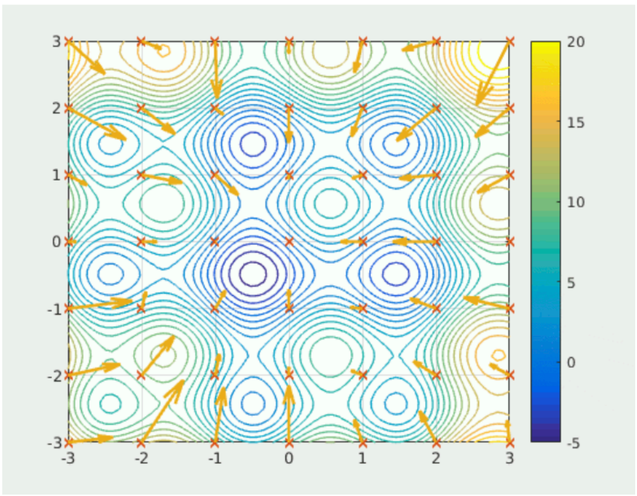


Customized PSO algorithm in MOPT

technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n

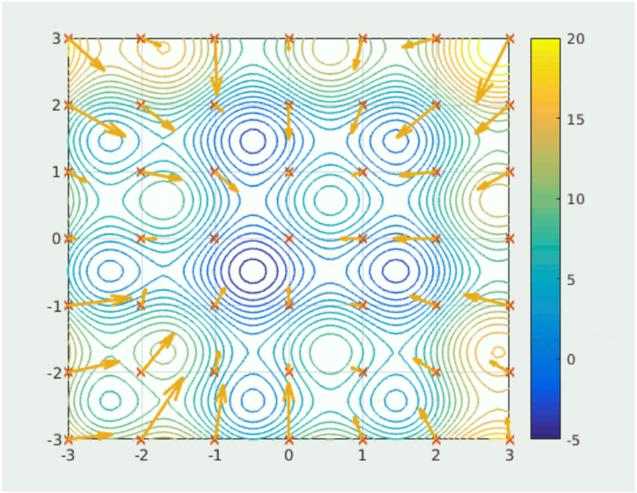
a single swarm

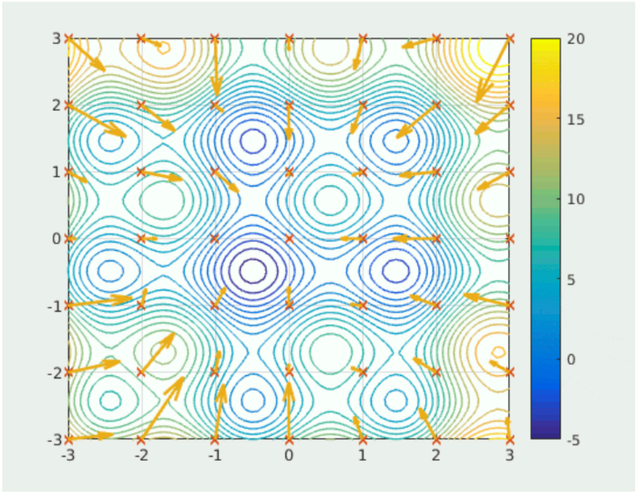




Customized PSO algorithm in MOPT

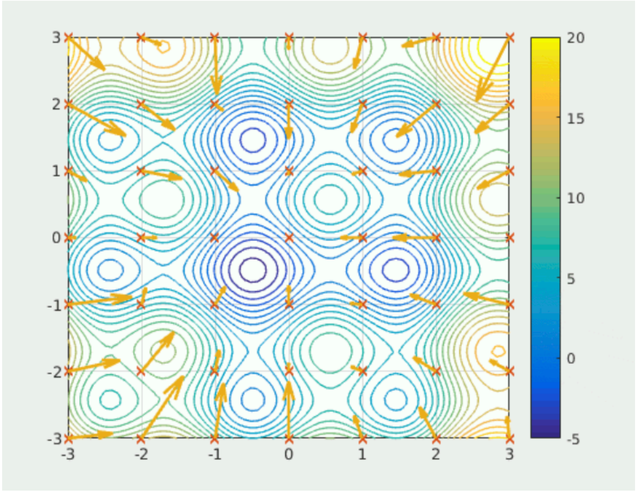
technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n

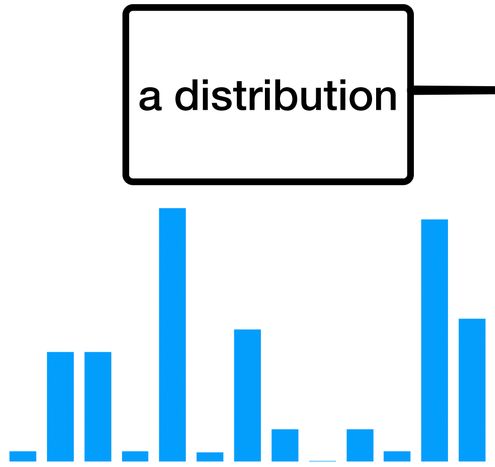


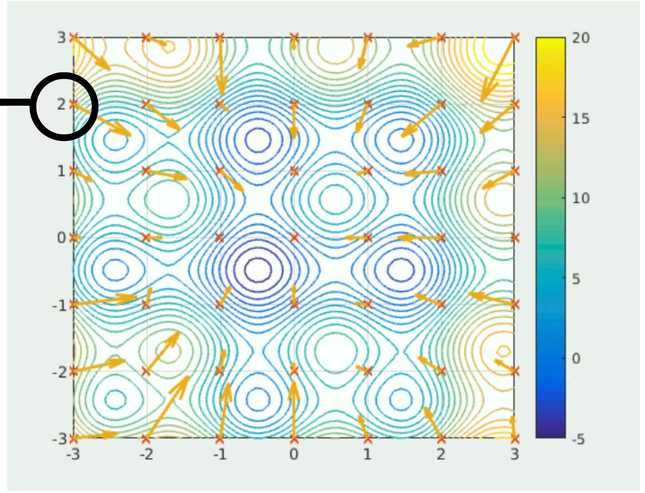


Customized PSO algorithm in MOPT

technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n

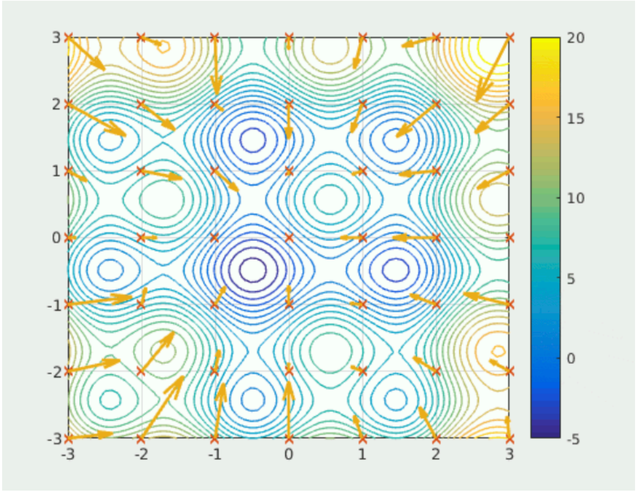


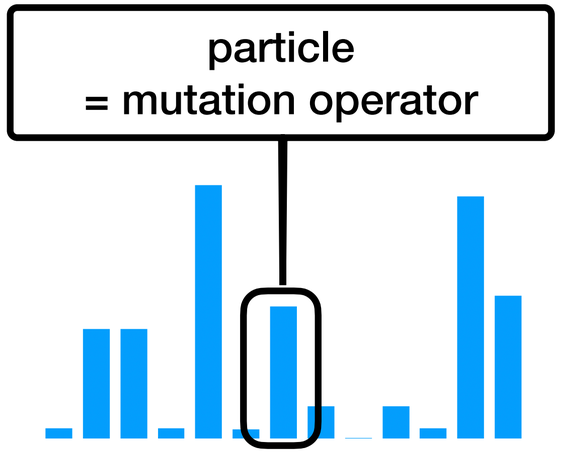


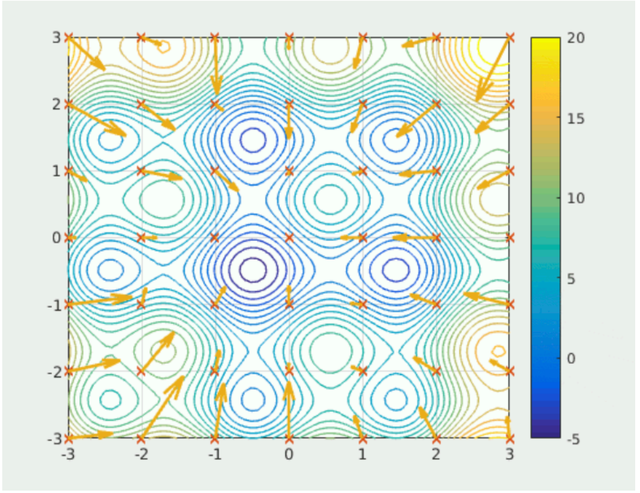


Customized PSO algorithm in MOPT

technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n

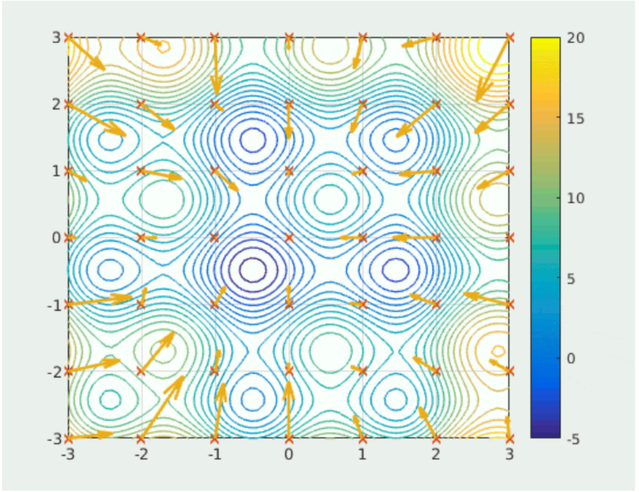


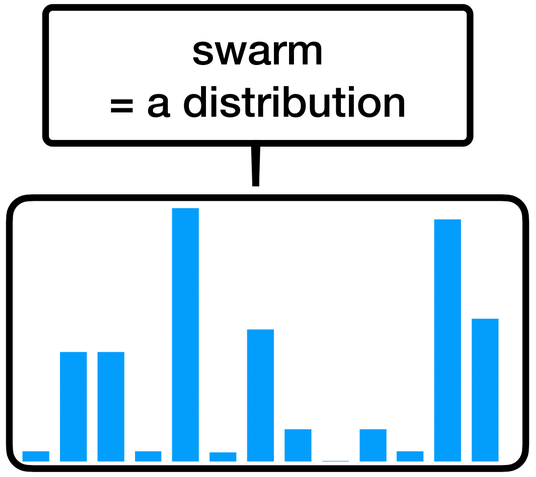


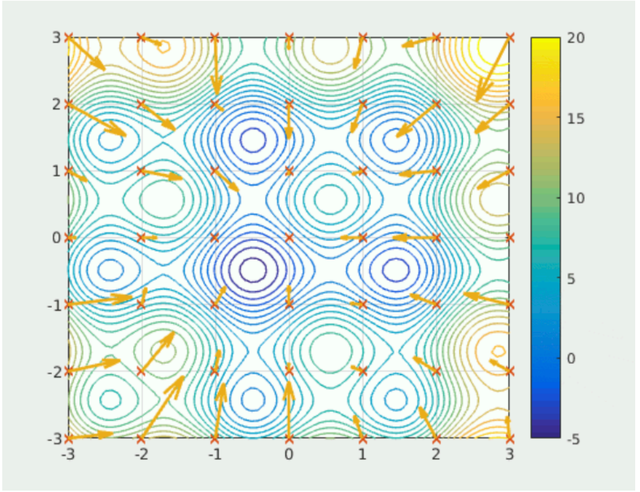


Customized PSO algorithm in MOPT

technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n

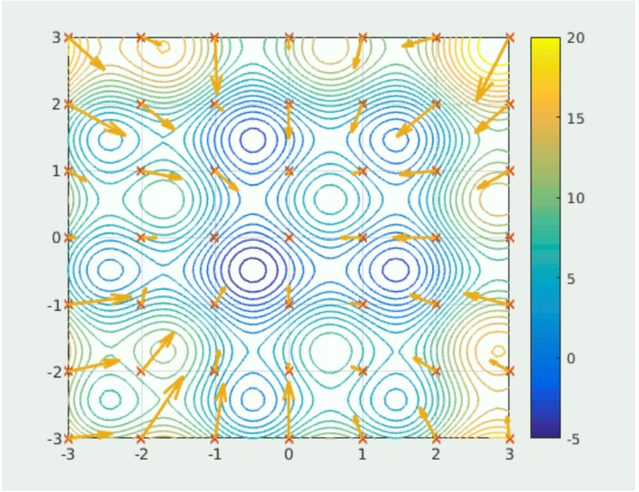


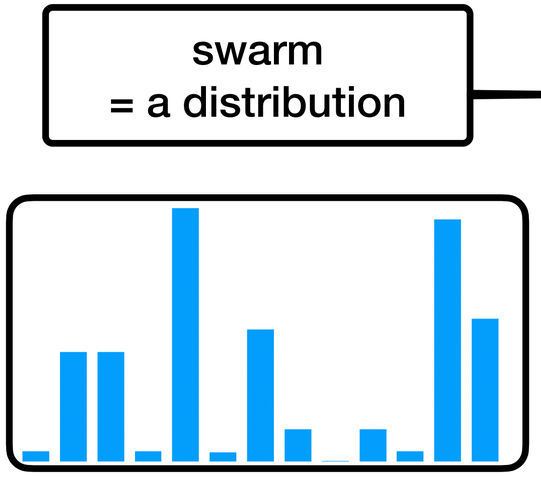


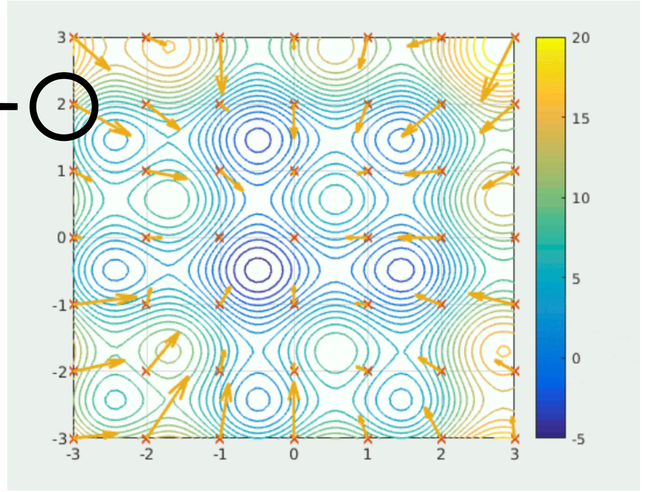


Customized PSO algorithm in MOPT

technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n

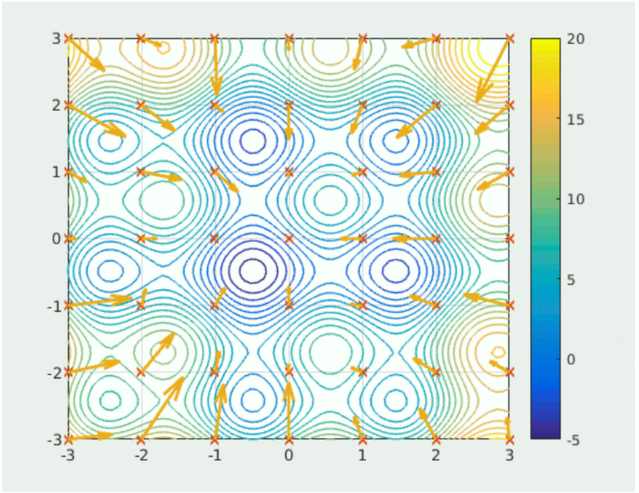




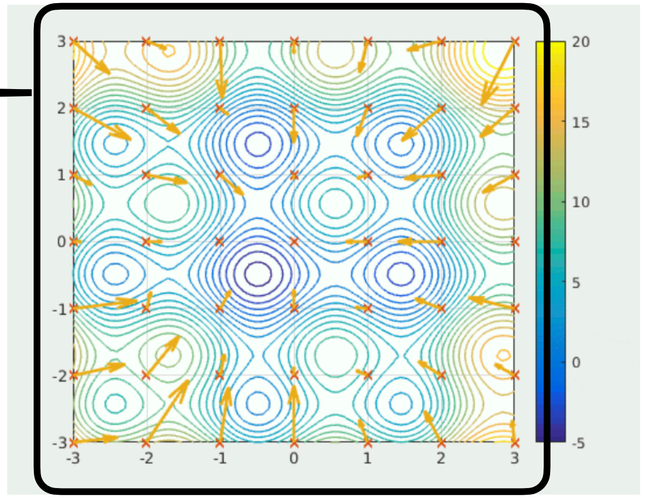


Customized PSO algorithm in MOPT

technique	original PSO	customized PSO in MOpt
objective	find an optimal position	find an optimal distribution
a point in the solution space	position	a distribution (= probabilities of mutation operators)
particle	position	(a probability of) mutation operator
swarm	a set of particles	a set of particles (= a distribution)
# of swarm	1	n



multiple swarms



Customized PSO algorithm in MOPT

- Solution Representation: a swarm x_i^t
  where, *j*th particle is a mutation operator with its **probability (that the operator will be selected)** $x_{i,j}^t$ and **velocity** $v_{i,j}^t$

- Initialization: $x_{i,j}^0, v_{i,j}^0 \in [lb, ub]$

- Solution update
  - swarm update
    - prob. update (for all particle): $x_{i,j}^{(k+1)} = x_{i,j}^k + v_{i,j}^{(k+1)}$
  - velocity update:

$$v_{i,j}^{(k+1)} = w v_{i,j}^k + r_1(p_{i,j} - x_{i,j}^k) + r_2(g_j - x_{i,j}^k)$$

$$where\ w\ is\ a\ inersia\ weight\ and\ r_1, r_2 \sim U(0, 1)$$

Customized PSO algorithm in MOPT

- Solution Representation: a swarm x_i^t
  where, j^{th} particle is a mutation operator with its **probability (that the operator will be selected)** $x_{i,j}^t$ and **velocity** $v_{i,j}^t$

- Initialization: $x_{i,j}^0, v_{i,j}^0 \in [lb, ub]$

- Solution update
  - swarm update
    - prob. update (for all particle): $x_{i,j}^{(k+1)} = x_{i,j}^k + v_{i,j}^{(k+1)}$
  - velocity update:

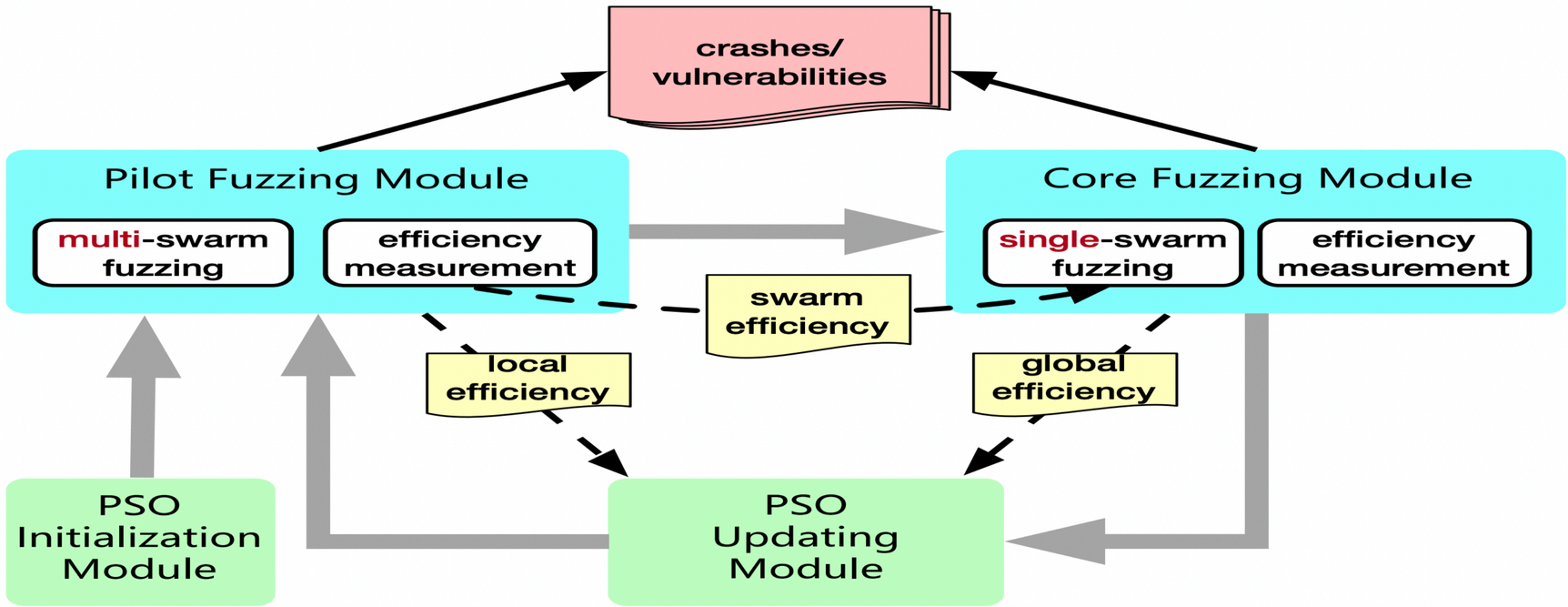
local best probability of j^{th} particle in i^{th} swarm

global best probability of j^{th} particles in all swarms

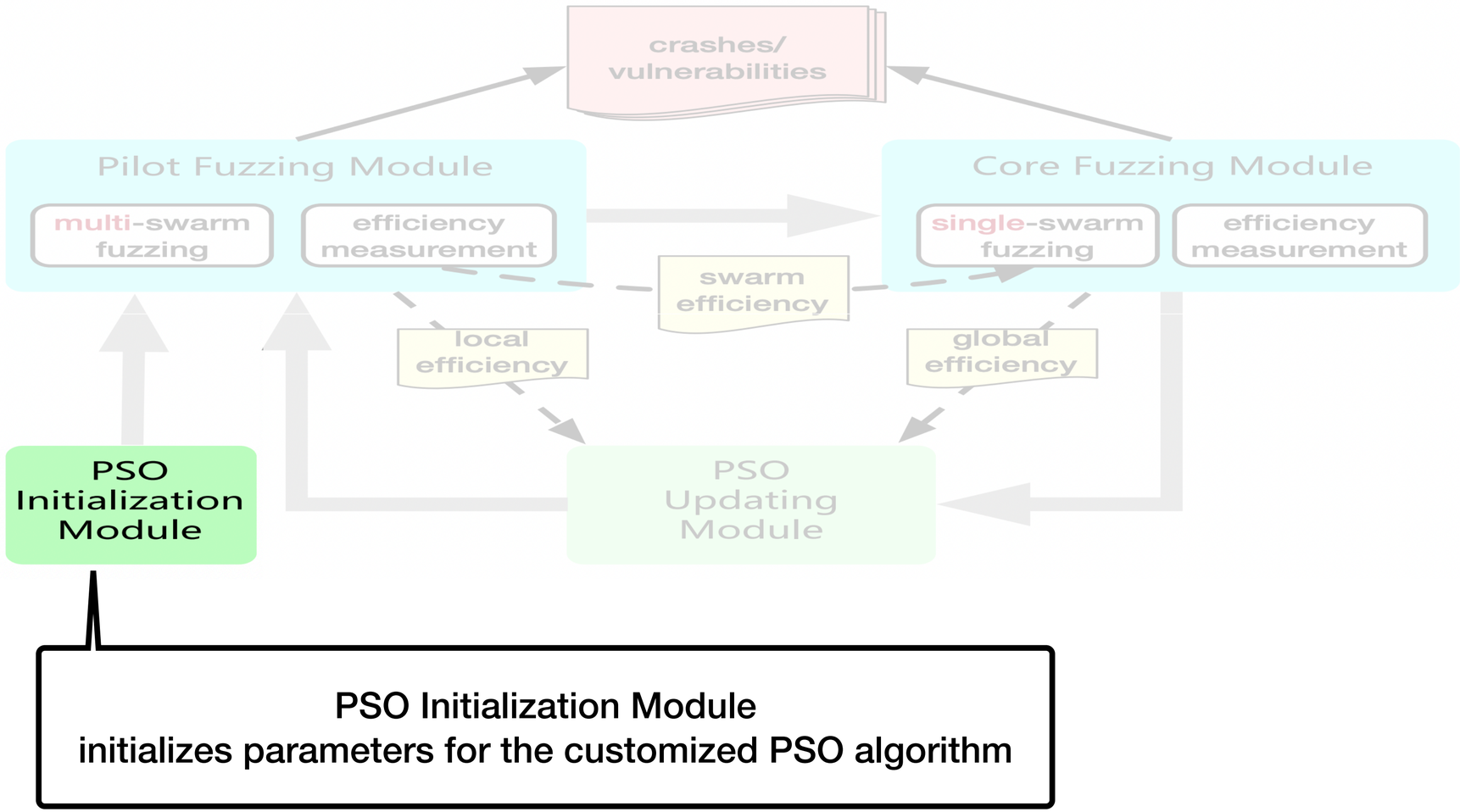
$$v_{i,j}^{(k+1)} = wv_{i,j}^k + r_1(p_{i,j} - x_{i,j}^k) + r_2(g_j - x_{i,j}^k)$$

$$where\ w\ is\ a\ inersia\ weight\ and\ r_1, r_2 \sim U(0, 1)$$

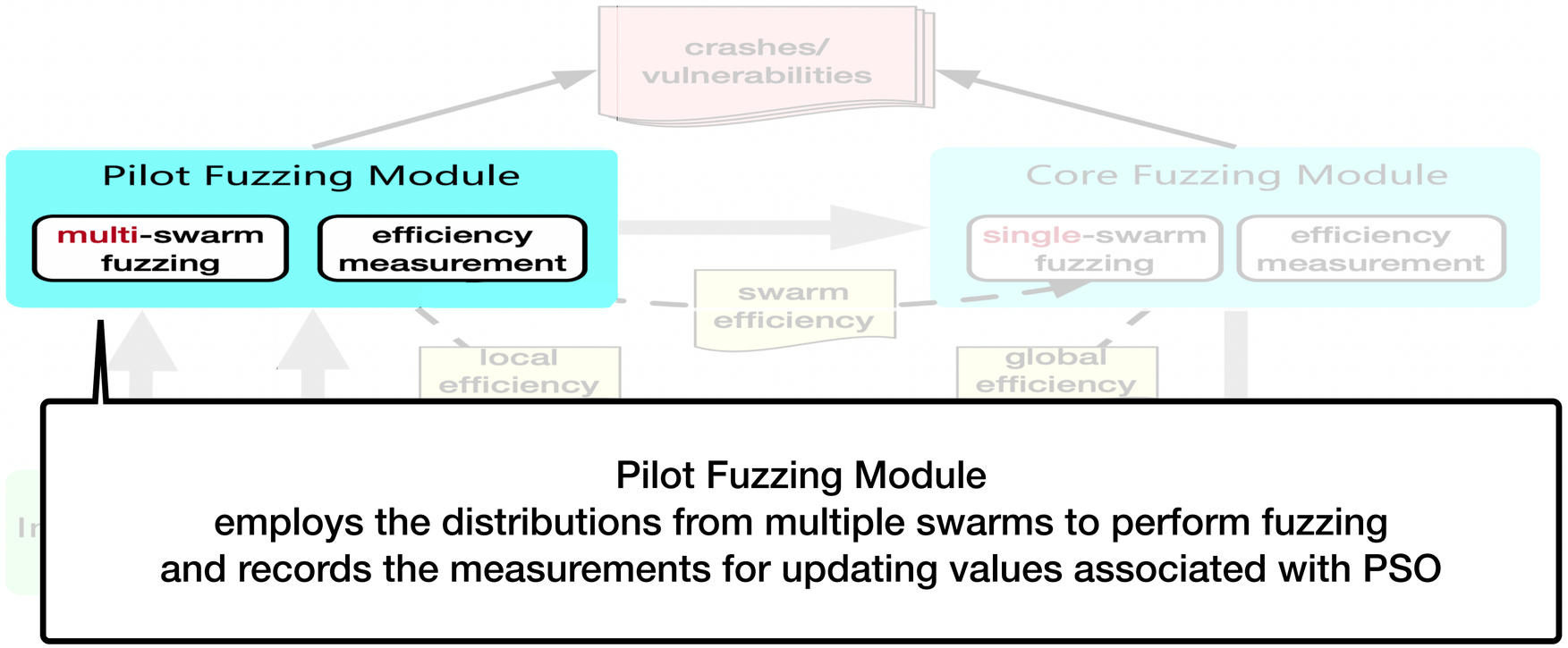
MOpt framework



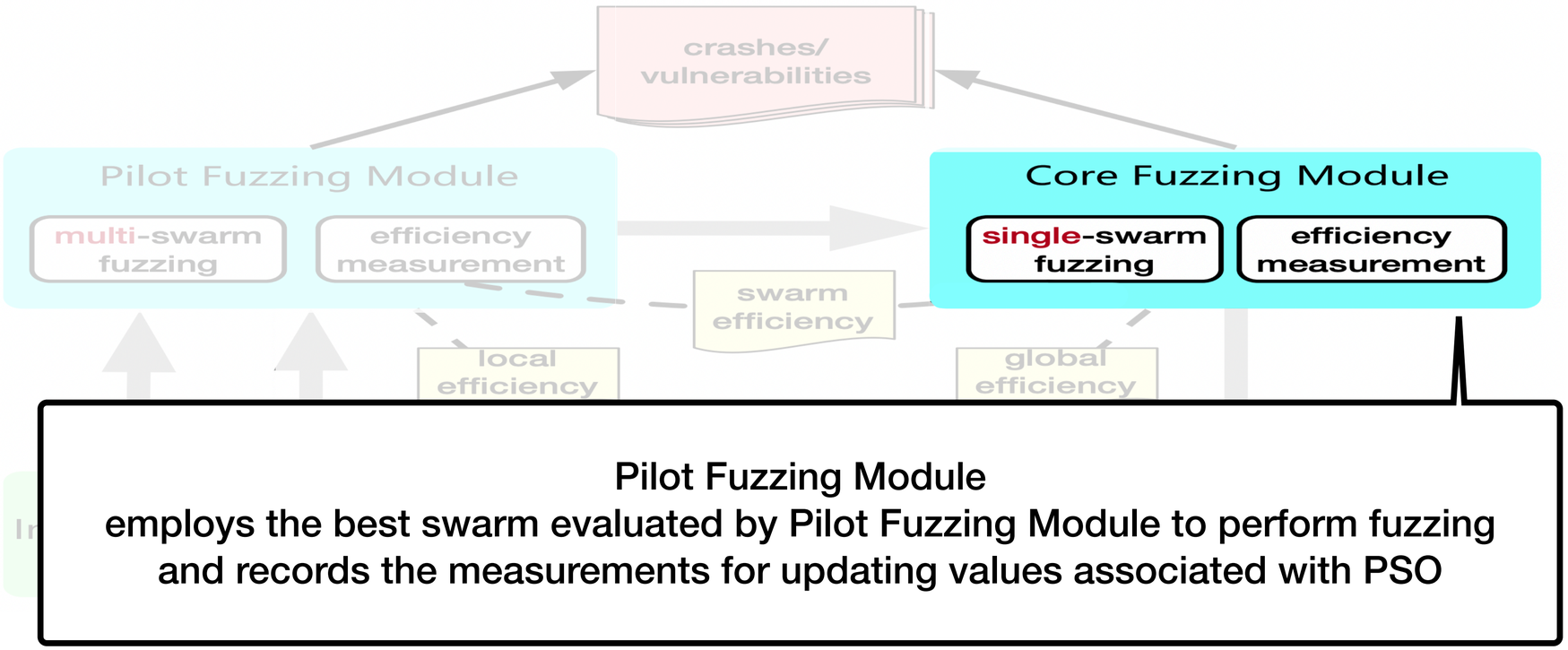
MOpt framework



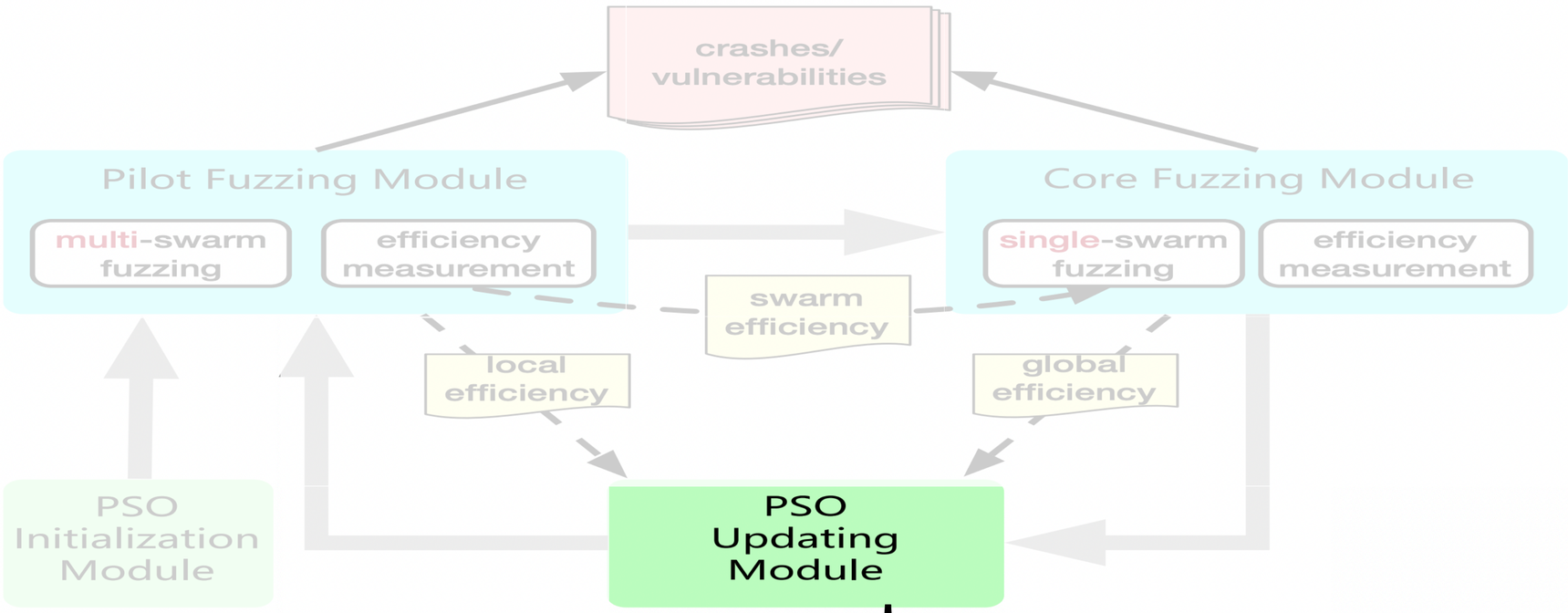
MOpt framework



MOpt framework

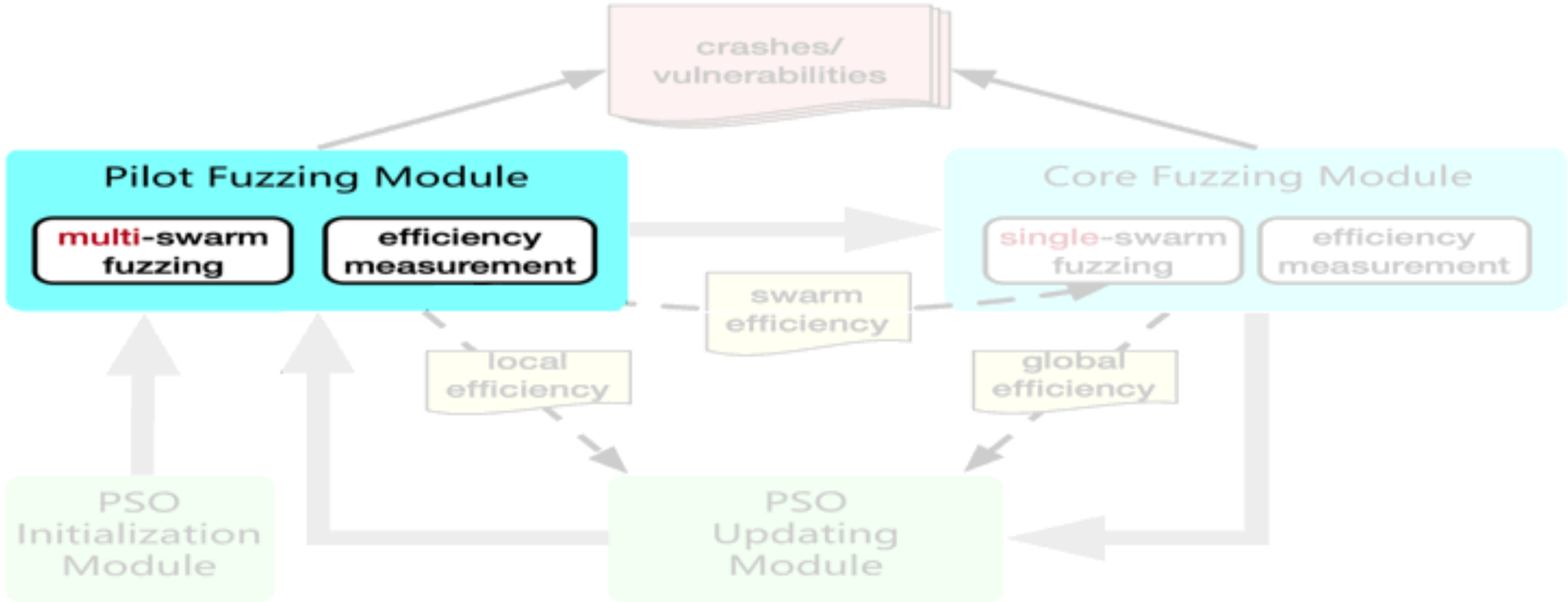


MOpt framework



PSO Updating Module
updates the distribution of each swarm
with the measurements from the two Fuzzing Modules

MOpt framework



And, MOpt continuously fuzzes target program
by iteratively executing the three modules;
Pilot Fuzzing, Core Fuzzing, and PSO Updating Modules

Pacemaker fuzzing mode

- Deterministic stage
  - shows good performance at the beginning of fuzzing
  - but requires much more time on a single test case than other stages

- Executing deterministic stage too many times might slow down the overall efficiency of fuzzing
  - AFL handles this by executing deterministic stage only once per test case
  - And, if no new crashes or paths are found, AFL skips deterministic stages and iterates havoc and splicing stages only

This might be thought of as a, very conservative, pacemaking process in AFL, but it is not enough!

Pacemaker fuzzing mode

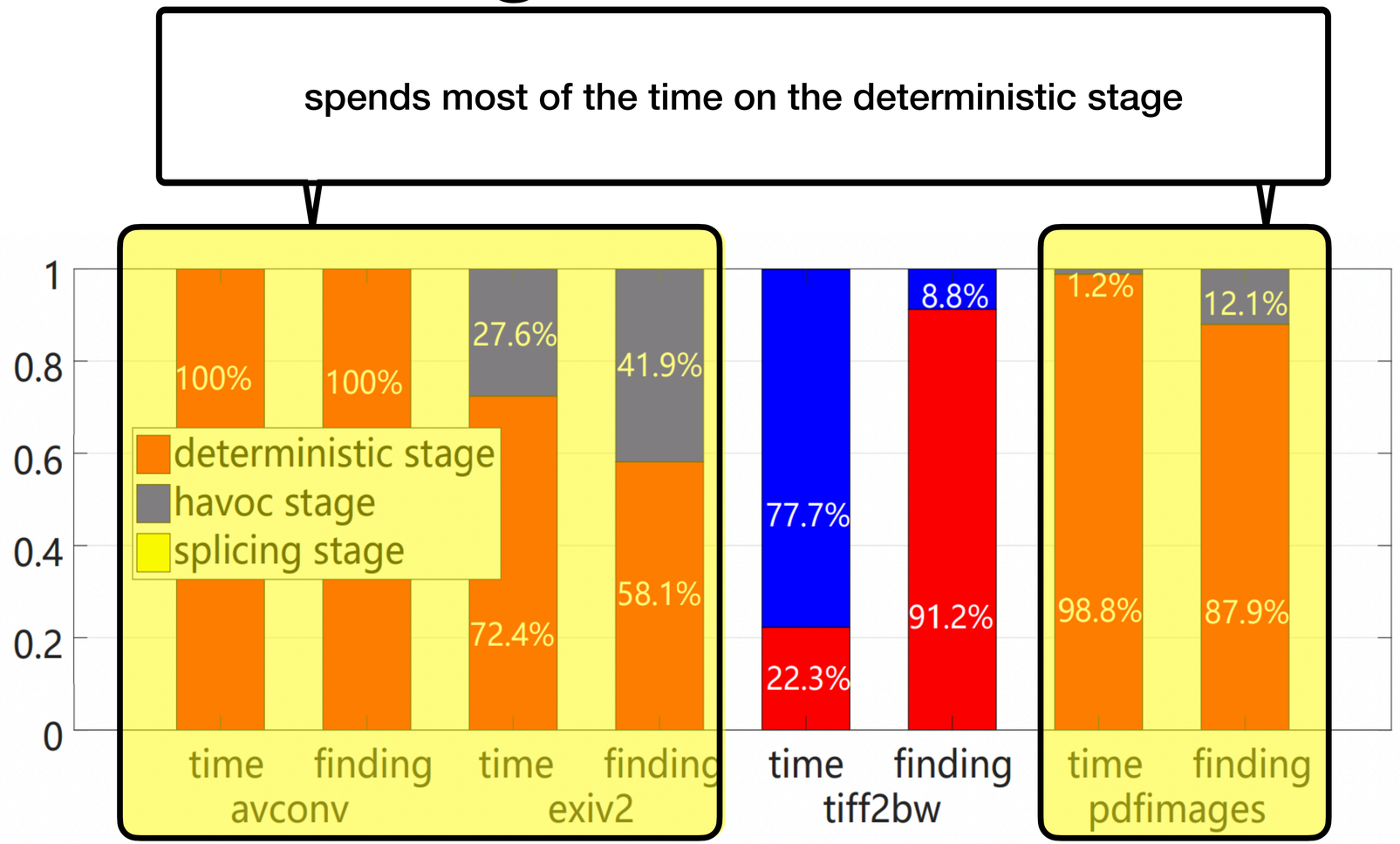


Figure 5: Percentages of time and interesting test cases used and found by the three stages in AFL, respectively.

Pacemaker fuzzing mode

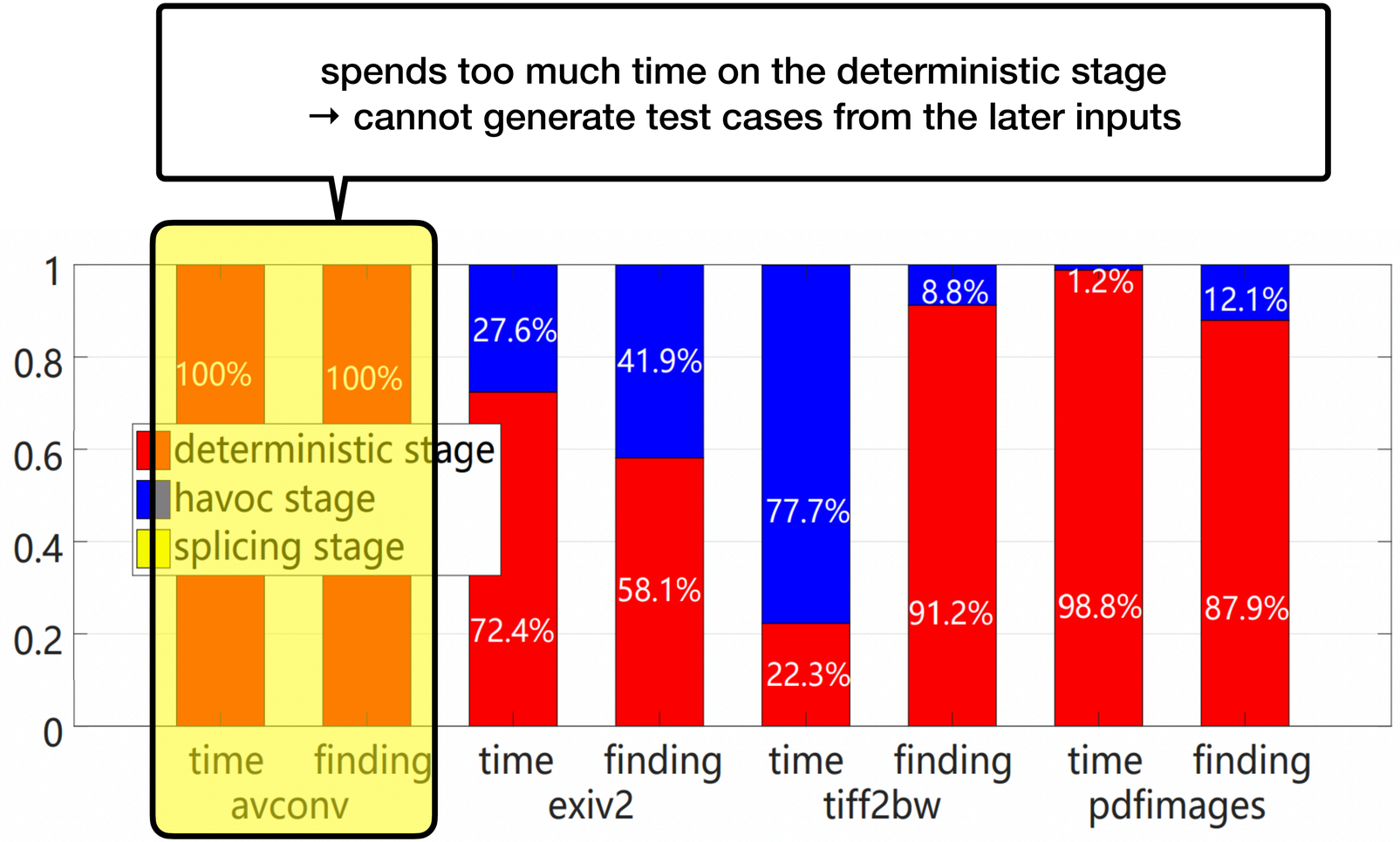


Figure 5: Percentages of time and interesting test cases used and found by the three stages in AFL, respectively.

Pacemaker fuzzing mode

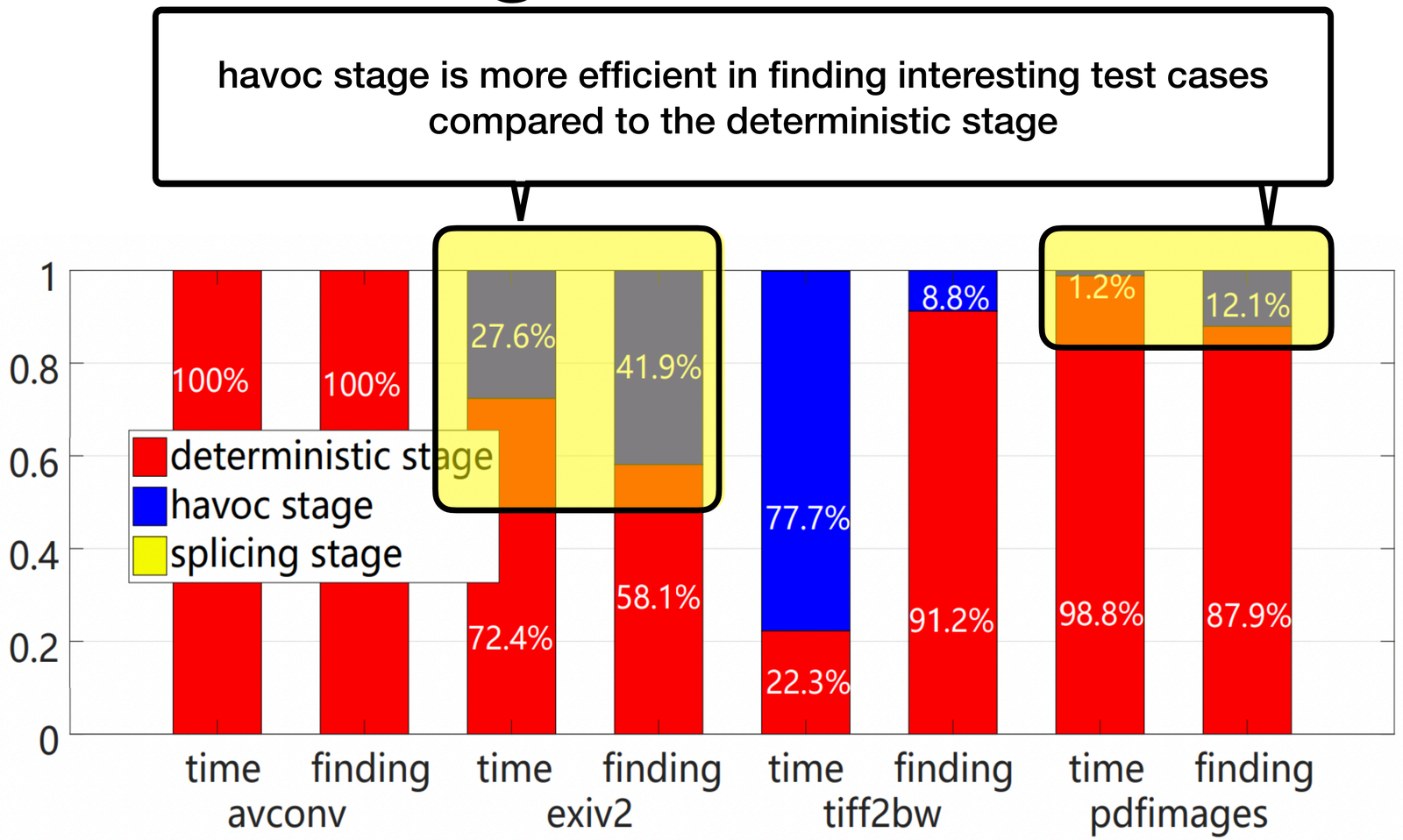


Figure 5: Percentages of time and interesting test cases used and found by the three stages in AFL, respectively.

Pacemaker fuzzing mode

- So, MOpt introduces pacemaker to determine whether it will execute the deterministic stage or not
  - MOpt checks the below condition after it finishes mutating one seed
  - pacemaking condition
    - it has not discovered any new unique crash or path for a long time (= T, set by users)
    - If true, it will disable the deterministic stage for the following seeds

- Pacemaker modes
  - MOPT-AFL-tmp: re-enable the deterministic stage again when the number of new interesting test cases exceeds a predefined threshold
  - MOPT-AFL-ever: never re-enable the deterministic stage

Evaluation settings

- Real world datasets
  - uses 13 open-source linux programs

Table 2: Objective programs evaluated in our experiments.

Target	Source file	Input format	Test instruction
mp42aac	Bento4-1-5-1	mp4	mp42aac @@ /dev/null
exiv2	exiv2-0.26-trunk	jpg	exiv2 @@ /dev/null
mp3gain	mp3gain-1_5_2	mp3	mp3gain @@ /dev/null
tiff2bw	libtiff-4.0.9	tiff	tiff2bw @@ /dev/null
pdfimages	xpdf-4.00	PDF	pdfimages @@ /dev/null
sam2p	sam2p-0.49.4	bmp	sam2p @@ EPS: /dev/null
avconv	libav-12.3	mp4	avconv -y -i @@ -f null -
w3m	w3m-0.5.3	text	w3m @@
objdump	binutils-2.30	binary	objdump –dwarf-check -C -g -f -dwarf -x @@
jhead	jhead-3.00	jpg	jhead @@
mpg321	mpg321_0.3.2	mp3	mpg321 -t @@ /dev/null
infotocap	ncurses-6.1	text	infotocap @@
podofopdfinfo	podofo-0.9.6	PDF	podofopdfinfo @@

Evaluation settings

- Evaluation environments
  - AFL
    - version 2.52b
  - prototypes
    - apply MOPT in the havoc stage of AFL
    - implement two prototypes: MOPT-AFL-tmp and MOPT-AFL-ever
  - machine spec.
    - run on a virtual machine (single 2.40GHz CPU core, 4.5GB RAM and 64-bit Ubuntu 16.04 LTS)
  - initial seed sets
    - randomly select 100 files from the set of files with the corresponding input format

1) The number of unique crashes and paths

Table 3: The unique crashes and paths found by AFL, MOPT-AFL-tmp and MOPT-AFL-ever on the 13 real world programs.

Program	AFL		MOPT-AFL-tmp				MOPT-AFL-ever			
	Unique crashes	Unique paths	Unique crashes	Increase	Unique paths	Increase	Unique crashes	Increase	Unique paths	Increase
mp42aac	135	815	**209**	+54.8%	1,660	+103.7%	199	+47.4%	**1,730**	+112.3%
exiv2	34	2,195	54	+58.8%	2,980	+35.8%	**66**	+94.1%	**4,642**	+111.5%
mp3gain	178	1,430	**262**	+47.2%	**2,211**	+54.6%	**262**	+47.2%	2,206	+54.3%
tiff2bw	4	4,738	**85**	+2,025.0%	**7,354**	+55.2%	43	+975.0%	7,295	+54.0%
pdfimages	23	12,915	357	+1,452.2%	22,661	+75.5%	**471**	+1,947.8%	**26,669**	+106.5%
sam2p	36	531	105	+191.7%	1,967	+270.4%	**329**	+813.9%	**3,418**	+543.7%
avconv	0	2,478	**4**	+4	**17,359**	+600.5%	1	+1	16,812	+578.5%
w3m	0	3,243	**506**	+506	5,313	+63.8%	182	+182	**5,326**	+64.2%
objdump	0	11,565	**470**	+470	19,309	+67.0%	287	+287	**22,648**	+95.8%
jhead	19	478	55	+189.5%	**489**	+2.3%	**69**	+263.2%	483	+1.0%
mpg321	10	123	**236**	+2,260.0%	1,054	+756.9%	229	+2,190.0%	**1,162**	+844.7%
infotocap	92	3,710	340	+269.6%	6,157	+66.0%	**692**	+652.2%	**7,048**	+90.0%
podofopdfinfo	79	3,397	**122**	+54.4%	**4,704**	+38.5%	114	+44.3%	4,694	+38.2%
total	610	47,618	2,805	+359.8%	93,218	+95.8%	**2,944**	+382.6%	**104,133**	+118.7%

run 240 hours for each experiment

1) The number of unique crashes and paths

Table 3: The unique crashes and paths found by AFL, MOPT-AFL-tmp and MOPT-AFL-ever on the 13 real world programs.

Program	AFL		MOPT-AFL-tmp				MOPT-AFL-ever			
	Unique crashes	Unique paths	Unique crashes	Increase	Unique paths	Increase	Unique crashes	Increase	Unique paths	Increase
mp42aac	135	815	**209**	+54.8%	1,660	+103.7%	199	+47.4%	**1,730**	+112.3%
exiv2	34	2,195	54	+58.8%	2,980	+35.8%	**66**	+94.1%	**4,642**	+111.5%
mp3gain	178	1,430	**262**	+47.2%	**2,211**	+54.6%	**262**	+47.2%	2,206	+54.3%
tiff2bw	4	4,738	**85**	+2,025.0%	**7,354**	+55.2%	43	+975.0%	7,295	+54.0%
pdfimages	23	12,915	357	+1,452.2%	22,661	+75.5%	**471**	+1,947.8%	**26,669**	+106.5%
sam2p	36	531	105	+191.7%	1,967	+270.4%	**329**	+813.9%	**3,418**	+543.7%
avconv	0	2,478	**4**	+4	**17,359**	+600.5%	1	+1	16,812	+578.5%
w3m	0	3,243	**506**	+506	5,313	+63.8%	182	+182	**5,326**	+64.2%
objdump	0	11,565	**470**	+470	19,309	+67.0%	287	+287	**22,648**	+95.8%
jhead	19	478	55	+189.5%	**489**	+2.3%	**69**	+263.2%	483	+1.0%
mpg321	10	123	**236**	+2,260.0%	1,054	+756.9%	229	+2,190.0%	**1,162**	+844.7%
infotocap	92	3,710	340	+269.6%	6,157	+66.0%	**692**	+652.2%	**7,048**	+90.0%
podofopdfinfo	79	3,397	**122**	+54.4%	**4,704**	+38.5%	114	+44.3%	4,694	+38.2%
total	610	47,618	2,805	+359.8%	93,218	+95.8%	**2,944**	+382.6%	**104,133**	+118.7%

the number of unique crashes and paths discovered by AFL

1) The number of unique crashes and paths

Table 3: The unique crashes and paths found by AFL, MOPT-AFL-tmp and MOPT-AFL-ever on the 13 real world programs.

Program	AFL		MOPT-AFL-tmp				MOPT-AFL-ever			
	Unique crashes	Unique paths	Unique crashes	Increase	Unique paths	Increase	Unique crashes	Increase	Unique paths	Increase
mp42aac	135	815	**209**	+54.8%	1,660	+103.7%	199	+47.4%	**1,730**	+112.3%
exiv2	34	2,195	54	+58.8%	2,980	+35.8%	**66**	+94.1%	**4,642**	+111.5%
mp3gain	178	1,430	**262**	+47.2%	**2,211**	+54.6%	**262**	+47.2%	2,206	+54.3%
tiff2bw	4	4,738	**85**	+2,025.0%	**7,354**	+55.2%	43	+975.0%	7,295	+54.0%
pdfimages	23	12,915	357	+1,452.2%	22,661	+75.5%	**471**	+1,947.8%	**26,669**	+106.5%
sam2p	36	531	105	+191.7%	1,967	+270.4%	**329**	+813.9%	**3,418**	+543.7%
avconv	0	2,478	**4**	+4	**17,359**	+600.5%	1	+1	16,812	+578.5%
w3m	0	3,243	**506**	+506	5,313	+63.8%	182	+182	**5,326**	+64.2%
objdump	0	11,565	**470**	+470	19,309	+67.0%	287	+287	**22,648**	+95.8%
jhead	19	478	55	+189.5%	**489**	+2.3%	**69**	+263.2%	483	+1.0%
mpg321	10	123	**236**	+2,260.0%	1,054	+756.9%	229	+2,190.0%	**1,162**	+844.7%
infotocap	92	3,710	340	+269.6%	6,157	+66.0%	**692**	+652.2%	**7,048**	+90.0%
podofopdfinfo	79	3,397	**122**	+54.4%	**4,704**	+38.5%	114	+44.3%	4,694	+38.2%
total	610	47,618	2,805	+359.8%	93,218	+95.8%	**2,944**	+382.6%	**104,133**	+118.7%

the percentage(or the number) of increase
of unique crashes and paths discovered by MOpt-AFL-tmp/-ever

1) The number of unique crashes and paths

Table 3: The unique crashes and paths found by AFL, MOPT-AFL-tmp and MOPT-AFL-ever on the 13 real world programs.

Program	AFL		MOPT-AFL-tmp				MOPT-AFL-ever			
	Unique crashes	Unique paths	Unique crashes	Increase	Unique paths	Increase	Unique crashes	Increase	Unique paths	Increase
mp42aac	135	815	**209**	+54.8%	1,660	+103.7%	199	+47.4%	**1,730**	+112.3%
exiv2	34	2,195	54	+58.8%	2,980	+35.8%	**66**	+94.1%	**4,642**	+111.5%
mp3gain	178	1,430	**262**	+47.2%	**2,211**	+54.6%	**262**	+47.2%	2,206	+54.3%
tiff2bw	4	4,738	**85**	+2,025.0%	**7,354**	+55.2%	43	+975.0%	7,295	+54.0%
pdfimages	23	12,915	357	+1,452.2%	22,661	+75.5%	**471**	+1,947.8%	**26,669**	+106.5%
sam2p	36	531	105	+191.7%	1,967	+270.4%	**329**	+813.9%	**3,418**	+543.7%
avconv	0	2,478	**4**	+4	**17,359**	+600.5%	1	+1	16,812	+578.5%
w3m	0	3,243	**506**	+506	5,313	+63.8%	182	+182	**5,326**	+64.2%
objdump	0	11,565	**470**	+470	19,309	+67.0%	287	+287	**22,648**	+95.8%
jhead	19	478	55	+189.5%	**489**	+2.3%	**69**	+263.2%	483	+1.0%
mpg321	10	123	**236**	+2,260.0%	1,054	+756.9%	229	+2,190.0%	**1,162**	+844.7%
infotocap	92	3,710	340	+269.6%	6,157	+66.0%	**692**	+652.2%	**7,048**	+90.0%
podofopdfinfo	79	3,397	**122**	+54.4%	**4,704**	+38.5%	114	+44.3%	4,694	+38.2%
total	610	47,618	2,805	+359.8%	93,218	+95.8%	**2,944**	+382.6%	**104,133**	+118.7%

MOpt-AFL-tmp/-ever significantly improve the performance of AFL

2) The number of discovered vulnerabilities

- Methodology
  - recompile the evaluated programs with AddressSanitizer
  - re-evaluate them with the discovered crash inputs found in the first evaluation
  - make tuple of the top three source code locations of the stack trace (provided by AddressSanitizer) for each crash input
  - make a set of unique tuples
  - consider the corresponding input of the tuple as a vulnerability that triggers the corresponding crash

2) The number of discovered vulnerabilities

Table 4: Vulnerabilities found by AFL, MOPT-AFL-tmp and MOPT-AFL-ever.

Program	AFL				MOPT-AFL-tmp				MOPT-AFL-ever			
	Unknown vulnerabilities		Known vulnerabilities	Sum	Unknown vulnerabilities		Known vulnerabilities	Sum	Unknown vulnerabilities		Known vulnerabilities	Sum
	Not CVE	CVE	CVE		Not CVE	CVE	CVE		Not CVE	CVE	CVE	
mp42aac	/	1	1	2	/	2	1	3	/	5	1	6
exiv2	/	5	3	8	/	5	4	9	/	4	4	8
mp3gain	/	4	2	6	/	9	3	12	/	5	2	7
pdfimages	/	1	0	1	/	12	3	15	/	9	2	11
avconv	/	0	0	0	/	2	0	2	/	1	0	1
w3m	/	0	0	0	/	14	0	14	/	5	0	5
objdump	/	0	0	0	/	1	2	3	/	0	2	2
jhead	/	1	0	1	/	4	0	4	/	5	0	5
mpg321	/	0	1	1	/	0	1	1	/	0	1	1
infotocap	/	3	0	3	/	3	0	3	/	3	0	3
podofopdfinfo	/	5	0	5	/	6	0	6	/	6	0	6
tiff2bw	1	/	/	1	2	/	/	2	2	/	/	2
sam2p	5	/	/	5	14	/	/	14	28	/	/	28
Total	6	20	7	33	16	58	14	88	30	43	12	85

sum of known/unknown vulnerabilities
discovered by AFL, MOpt-AFL-tmp, and MOpt-AFL-ever.

2) The number of discovered vulnerabilities

Table 4: Vulnerabilities found by AFL, MOPT-AFL-tmp and MOPT-AFL-ever.

Program	AFL				MOPT-AFL-tmp				MOPT-AFL-ever			
	Unknown vulnerabilities		Known vulnerabilities	Sum	Unknown vulnerabilities		Known vulnerabilities	Sum	Unknown vulnerabilities		Known vulnerabilities	Sum
	Not CVE	CVE	CVE		Not CVE	CVE	CVE		Not CVE	CVE	CVE	
mp42aac	/	1	1	2	/	2	1	3	/	5	1	6
exiv2	/	5	3	8	/	5	4	9	/	4	4	8
mp3gain	/	4	2	6	/	9	3	12	/	5	2	7
pdfimages	/	1	0	1	/	12	3	15	/	9	2	11
avconv	/	0	0	0	/	2	0	2	/	1	0	1
w3m	/	0	0	0	/	14	0	14	/	5	0	5
objdump	/	0	0	0	/	1	2	3	/	0	2	2
jhead	/	1	0	1	/	4	0	4	/	5	0	5
mpg321	/	0	1	1	/	0	1	1	/	0	1	1
infotocap	/	3	0	3	/	3	0	3	/	3	0	3
podofopdfinfo	/	5	0	5	/	6	0	6	/	6	0	6
tiff2bw	1	/	/	1	2	/	/	2	2	/	/	2
sam2p	5	/	/	5	14	/	/	14	28	/	/	28
Total	6	20	7	33	16	58	14	88	30	43	12	85

Both MOpt-AFL-tmp and MOpt-AFL-ever found more vulnerabilities than AFL

3) The number of discovered CVEs

- Methodology
  - check the vulnerability reports of the target program on the CVE website
  - see whether the discovered vulnerabilities correspond to some already existed CVEs
  - If not, submit the vulnerability reports and the Proof of Concepts (PoCs) to the vendors and the CVE assignment team

3) The number of discovered CVEs

Table 4: Vulnerabilities found by AFL, MOPT-AFL-tmp and MOPT-AFL-ever.

Program	AFL				MOPT-AFL-tmp				MOPT-AFL-ever			
	Unknown vulnerabilities		Known vulnerabilities	Sum	Unknown vulnerabilities		Known vulnerabilities	Sum	Unknown vulnerabilities		Known vulnerabilities	Sum
	Not CVE	CVE	CVE		Not CVE	CVE	CVE		Not CVE	CVE	CVE	
mp42aac	/	1	1	2	/	2	1	3	/	5	1	6
exiv2	/	5	3	8	/	5	4	9	/	4	4	8
mp3gain	/	4	2	6	/	9	3	12	/	5	2	7
pdfimages	/	1	0	1	/	12	3	15	/	9	2	11
avconv	/	0	0	0	/	2	0	2	/	1	0	1
w3m	/	0	0	0	/	14	0	14	/	5	0	5
objdump	/	0	0	0	/	1	2	3	/	0	2	2
jhead	/	1	0	1	/	4	0	4	/	5	0	5
mpg321	/	0	1	1	/	0	1	1	/	0	1	1
infotocap	/	3	0	3	/	3	0	3	/	3	0	3
podofopdfinfo	/	5	0	5	/	6	0	6	/	6	0	6
tiff2bw	1	/	/	1	2	/	/	2	2	/	/	2
sam2p	5	/	/	5	14	/	/	14	28	/	/	28
Total	6	20	7	33	16	58	14	88	30	43	12	85

Both MOpt-AFL-tmp and MOpt-AFL-ever found more CVEs than AFL

4) Compatibility analysis

- MOPT prototypes are implemented based on AFL
- But, MOPT can be implemented on other mutation-based fuzzers
- Combine MOPT with AFLFast and VUzzer
  - implement MOPT-AFLFast-ever, MOPT-AFLFast-tmp, and MOPT-VUzzer

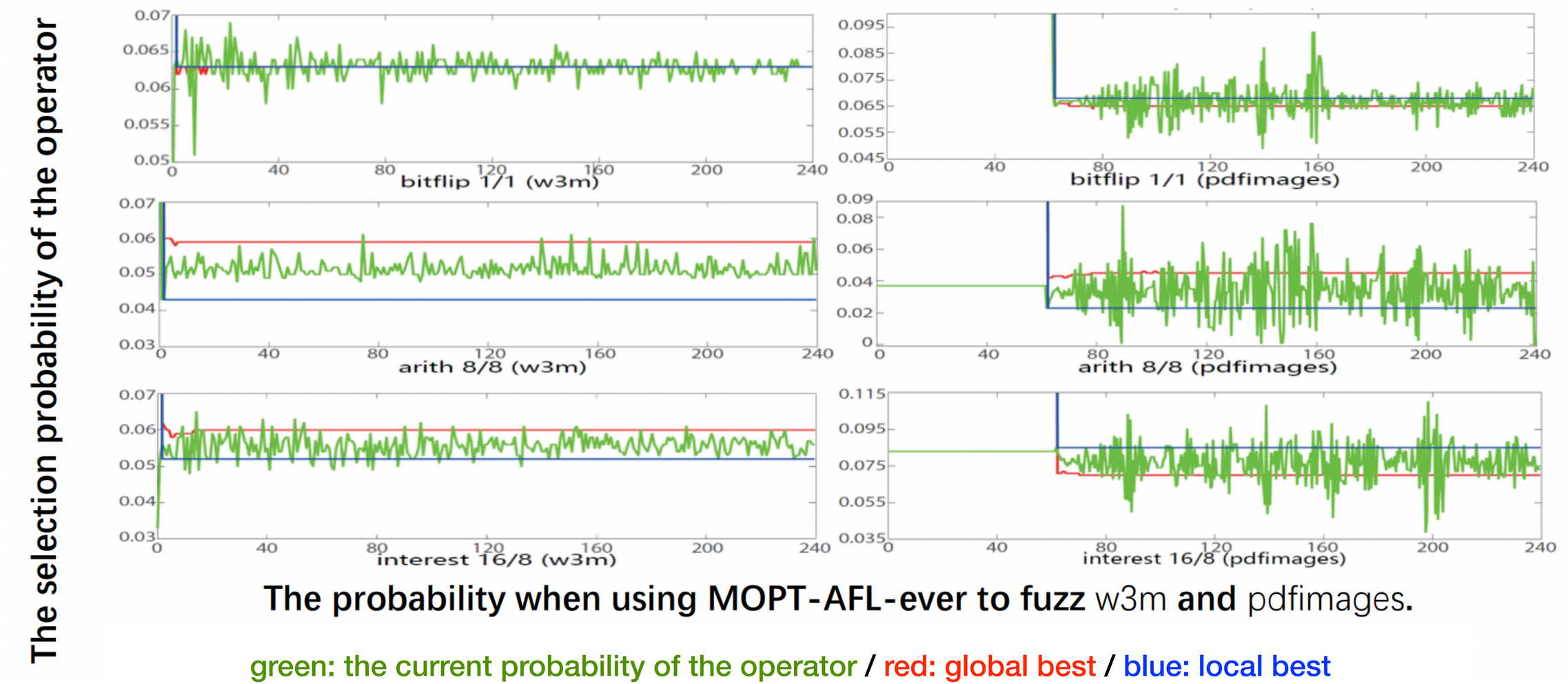
4) Compatibility analysis

Table 6: The compatibility of the MOPT scheme.

		mp42aac	exiv2	mp3gain	tiff2bw	pdfimages	sam2p	mpg321
AFL	Unique crashes	135	34	178	4	23	36	10
	Unique paths	815	2,195	1,430	4,738	12,915	531	123
MOPT-AFL-tmp	Unique crashes	**209**	54	**262**	**85**	357	105	**236**
	Unique paths	1,660	2,980	**2,211**	**7,354**	22,661	1,967	1,054
MOPT-AFL-ever	Unique crashes	199	**66**	**262**	43	**471**	**329**	229
	Unique paths	**1,730**	**4,642**	2,206	7,295	**26,669**	**3,418**	**1,162**
AFLFast	Unique crashes	210	0	171	0	18	37	8
	Unique paths	1,233	159	1,383	5,114	12,022	603	122
MOPT-AFLFast-tmp	Unique crashes	**393**	51	**264**	5	292	**196**	**230**
	Unique paths	**3,389**	2,675	2,017	7,012	24,164	2,587	**1,208**
MOPT-AFLFast-ever	Unique crashes	384	**58**	259	**18**	**345**	114	30
	Unique paths	2,951	**2,887**	**2,102**	**7,642**	**26,799**	**2,623**	160
VUzzer	Unique crashes	12	0	54,500	0	0	13	3,598
	Unique paths	12%	9%	50%	13%	25%	18%	18%
MOPT-VUzzer	Unique crashes	**16**	0	**56,109**	0	0	**16**	**3,615**
	Unique paths	12%	9%	**51%**	13%	25%	18%	18%

In this table, best results are bolded
MOPT-prototypes based on other mutation-based fuzzers
also outperform the corresponding base fuzzers

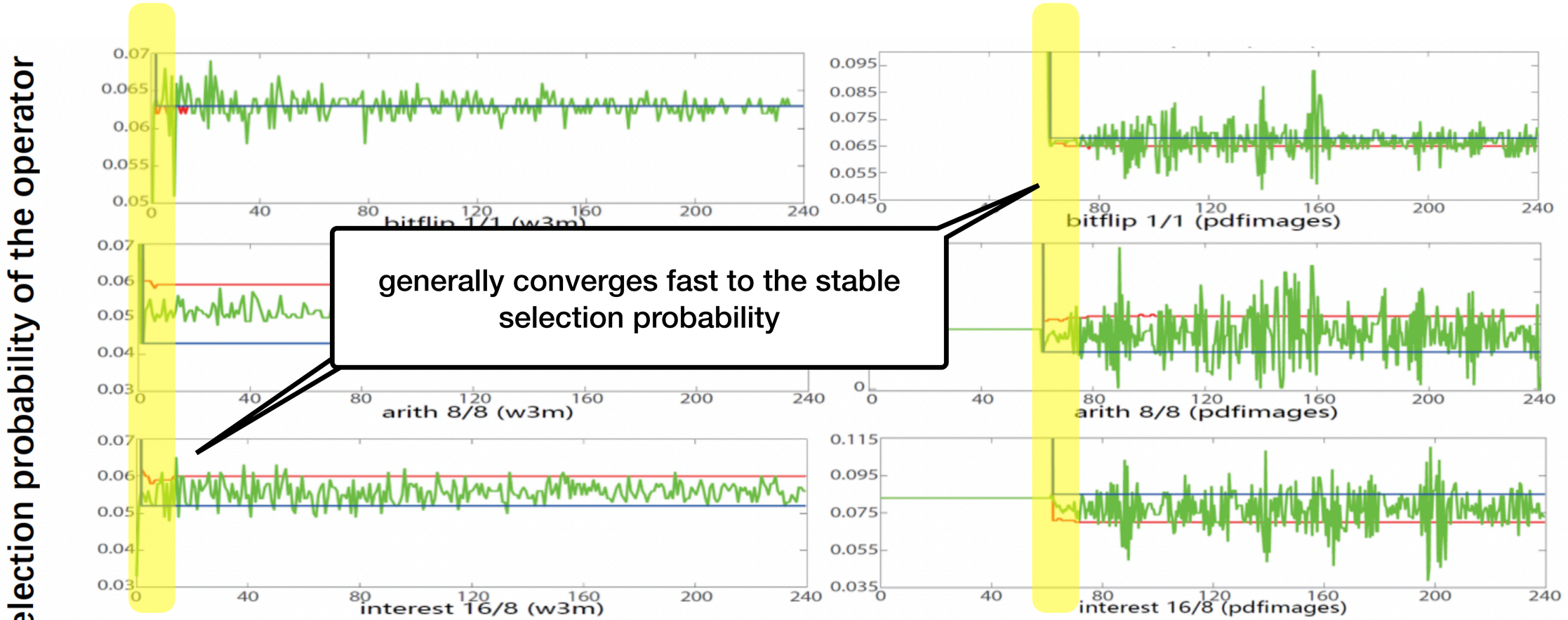
5) Convergence analysis



The probability when using MOPT-AFL-ever to fuzz w3m **and** pdfimages**.**

green: the current probability of the operator / red: global best / blue: local best

5) Convergence analysis



The probability when using MOPT-AFL-ever to fuzz w3m **and** pdfimages**.**

green: the current probability of the operator / red: global best / blue: local best

Conclusion

- Propose a novel mutation scheduling scheme, MOPT
- Apply MOPT to several state-of-the-art fuzzers
- Evaluate MOPT-fuzzers with the extensive experiments
- Demonstrate the high efficiency, compatibility, and steadiness of MOPT

Thank you

Appendix

1-2) The number of unique crashes over time

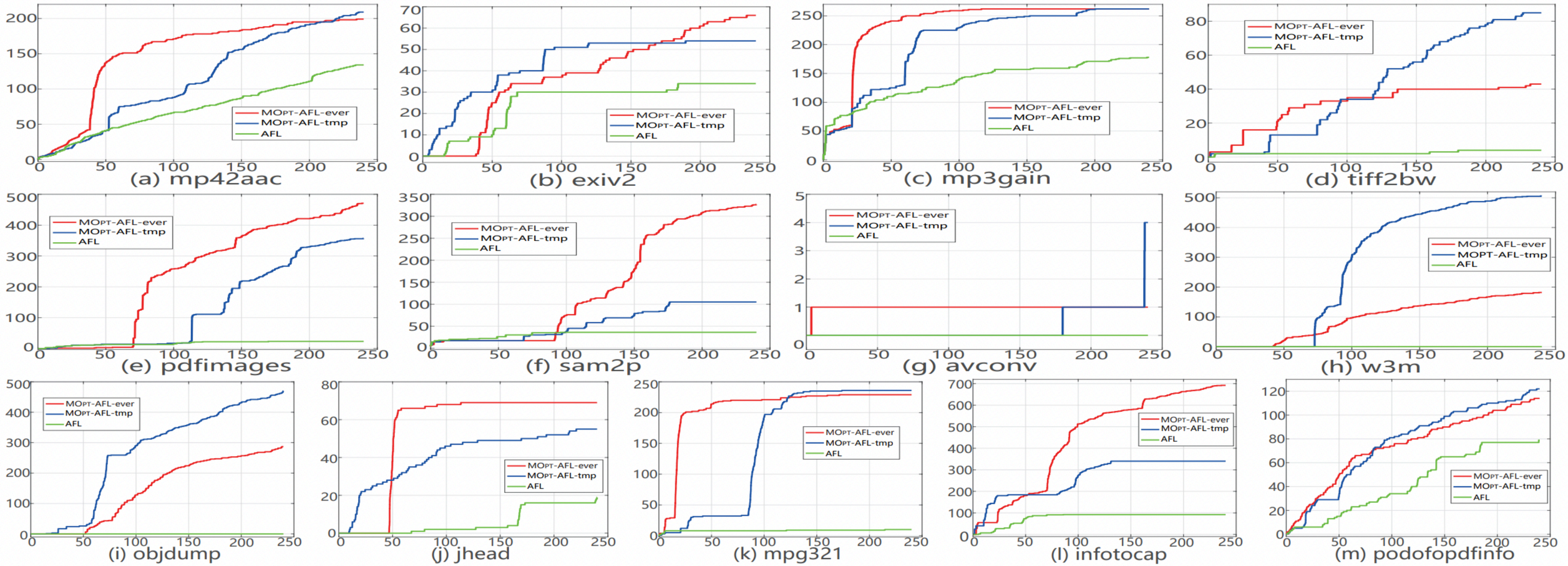


Figure 9: The number of unique crashes discovered by MOPT-AFL-ever, MOPT-AFL-tmp and AFL over 240 hours. X-axis: time (over 240 hours). Y-axis: the number of unique crashes.

1-2) The number of unique crashes over time

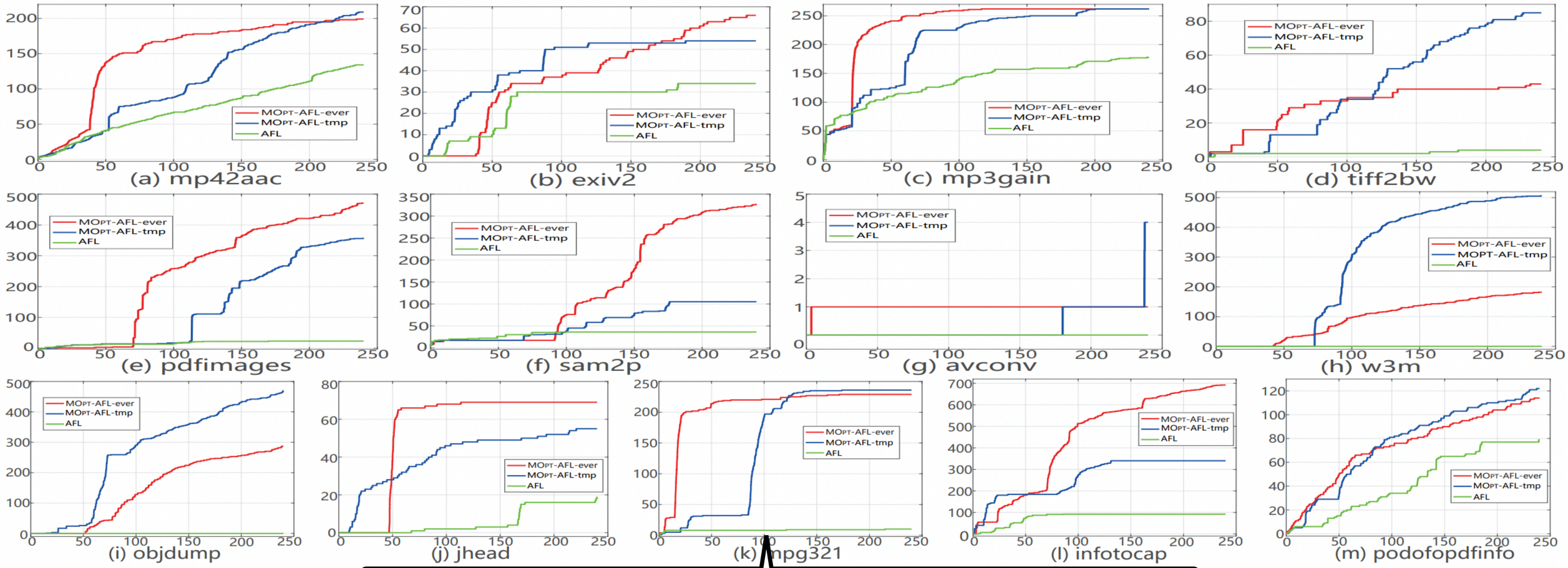


Figure 9: The number of uni AFL over 240 hours. X-axis: time (over 240 hours). Y-axi

red: MOpt-AFL-ever / blue: MOpt-AFL-tmp / green: AFL
Both MOpt prototypes outperform AFL in all experiments

1-2) The number of unique crashes over time

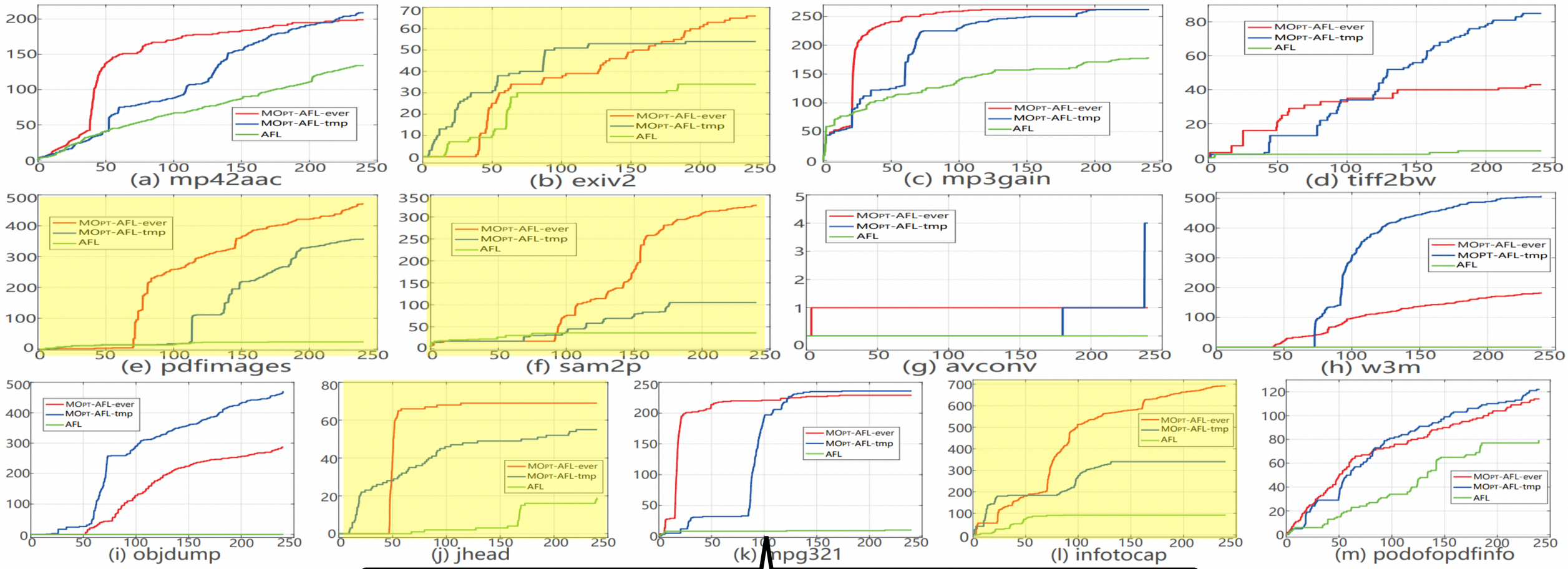


Figure 9: The number of uni AFL over 240 hours. X-axis: time (over 240 hours). Y-axi

red: MOpt-AFL-ever performs best in 5 programs

1-2) The number of unique crashes over time

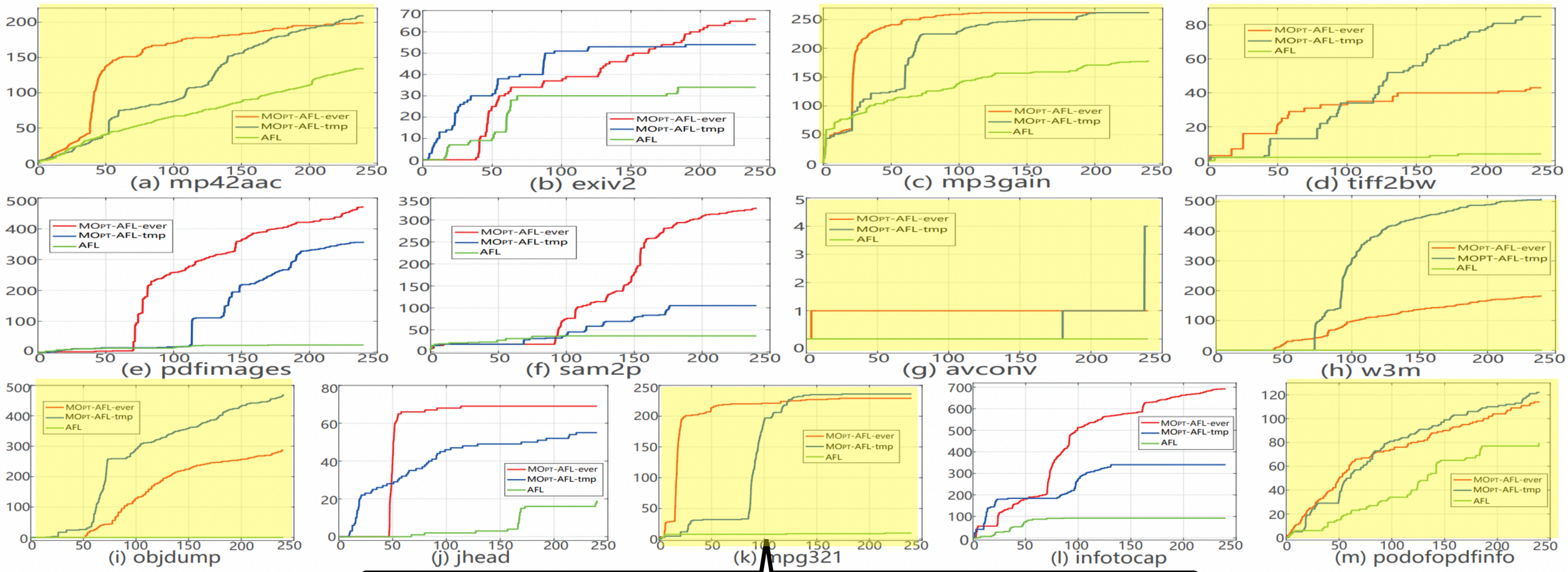


Figure 9: The number of uni AFL over 240 hours. X-axis: time (over 240 hours). Y-axi

blue: MOpt-AFL-tmp performs best in 8 programs

1-2) The number of unique crashes over time

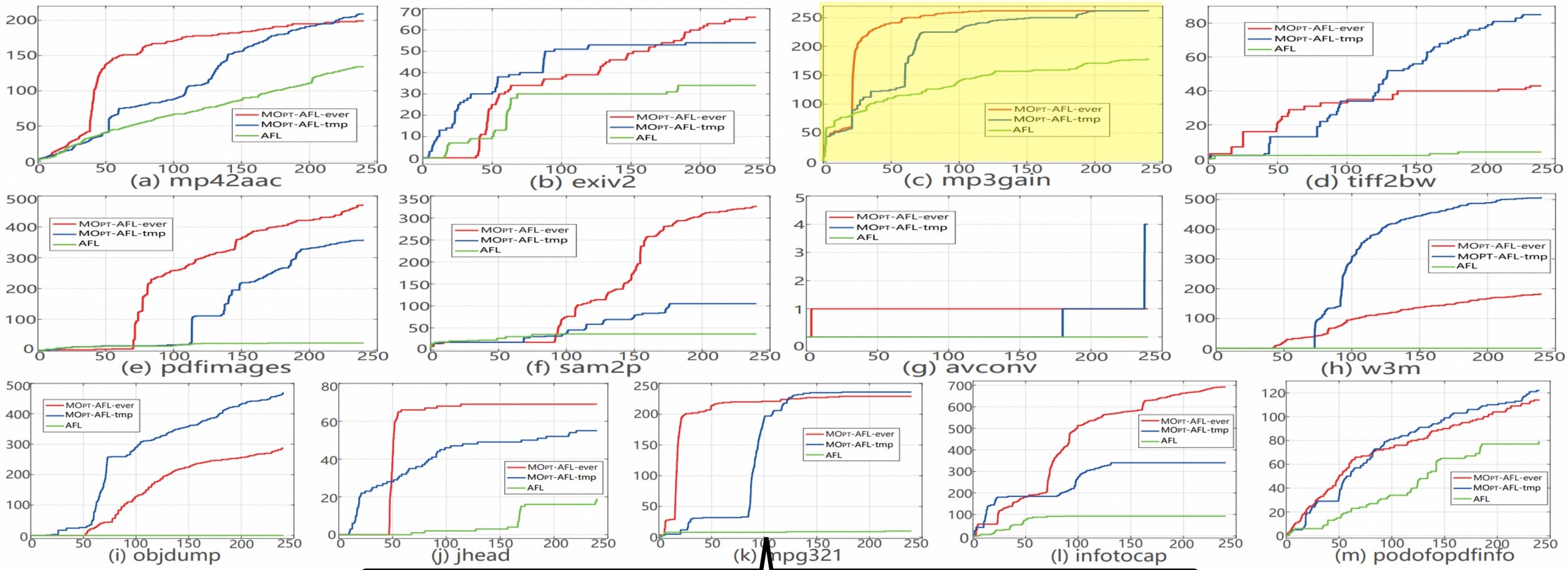


Figure 9: The number of uni … AFL over 240 hours. X-axis: time (over 240 hours). Y-axi…

Both MOpt prototypes found most of discovered crashes very fast at the beginning of fuzzing