DARWIN: Survival of the Fittest Fuzzing Mutators

Patrick Jauernig, Domagoj Jakobovic, Stjepan Picek,
Emmanuel Stapf, Ahmad-Reza Sadeghi

TECHNISCHE UNIVERSITÄT DARMSTADT

SANCTUARY

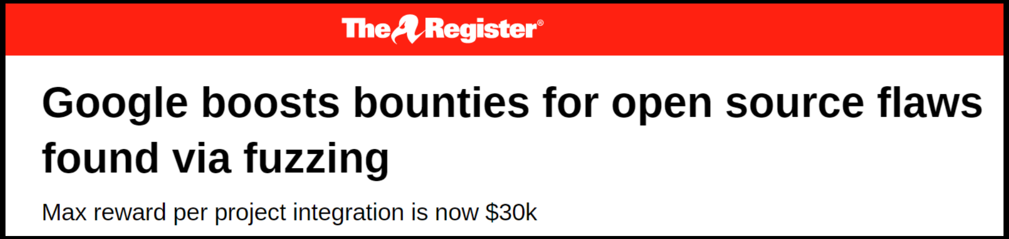
Radboud University

TU Delft

University of Zagreb

Motivation

- Fuzzing research is quite mature



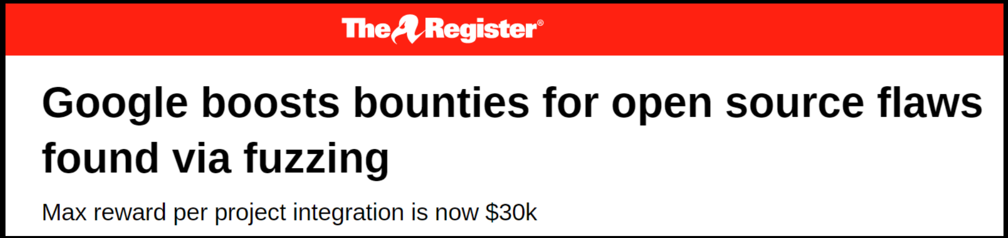
Google boosts bounties for open source flaws found via fuzzing

Max reward per project integration is now $30k

- Key drivers for adoption:
  - Enabling technologies (firmware rehosting, ...)
  - Platforms (OSS-Fuzz, ClusterFuzz)
- Lots of technical improvements (fast snapshots, coverage tracing)

Motivation

- Fuzzing research is quite mature



The Register

Google boosts bounties for open source flaws found via fuzzing

Max reward per project integration is now $30k

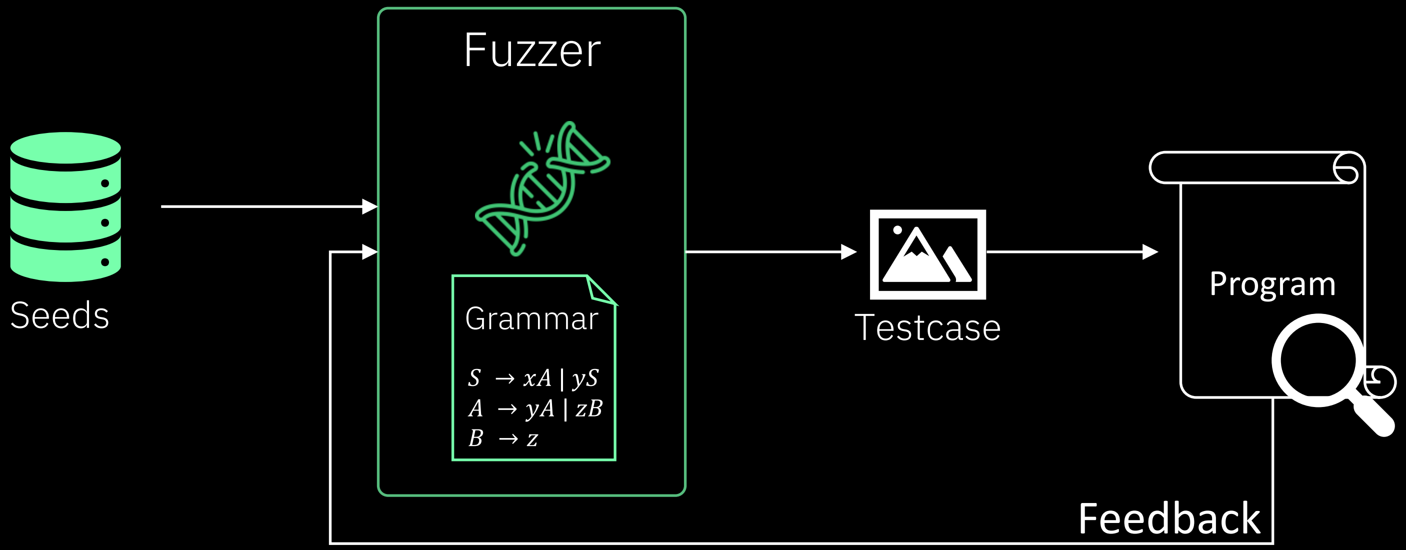
- Key drivers for adoption:
  - Enabling technologies (firmware rehosting, ...)
  - Platforms (OSS-Fuzz, ClusterFuzz)
- Lots of technical improvements (fast snapshots, coverage tracing)

- **Algorithmic improvements can increase efficiency across targets**

Backround – Fuzzing

- Dynamic analysis technique
  - Applies random inputs (testcases) to a target to see if it crashes

- Traditional separation: grammar-based vs. mutational



Seeds

Fuzzer

Grammar

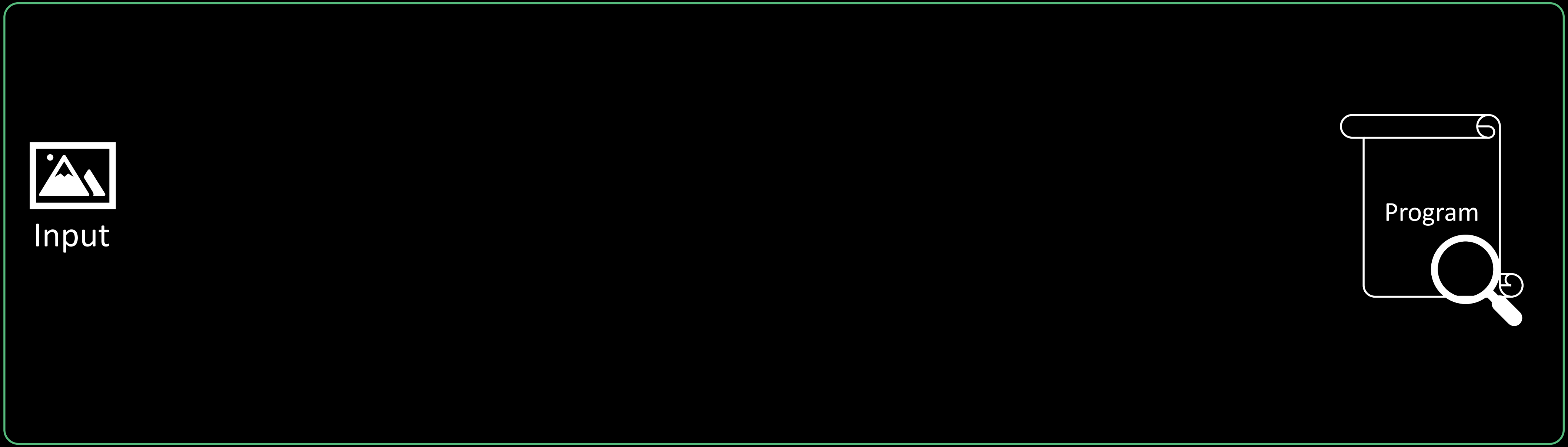
$S \rightarrow xA \mid yS$
$A \rightarrow yA \mid zB$
$B \rightarrow z$

Testcase

Program

Feedback

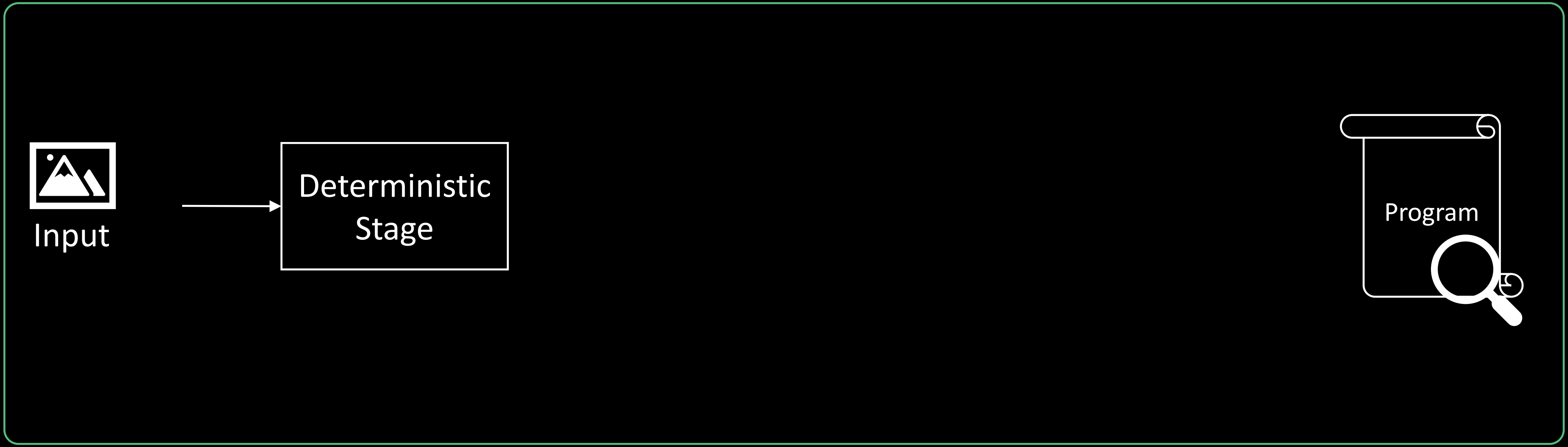
Background – Mutational Fuzzers



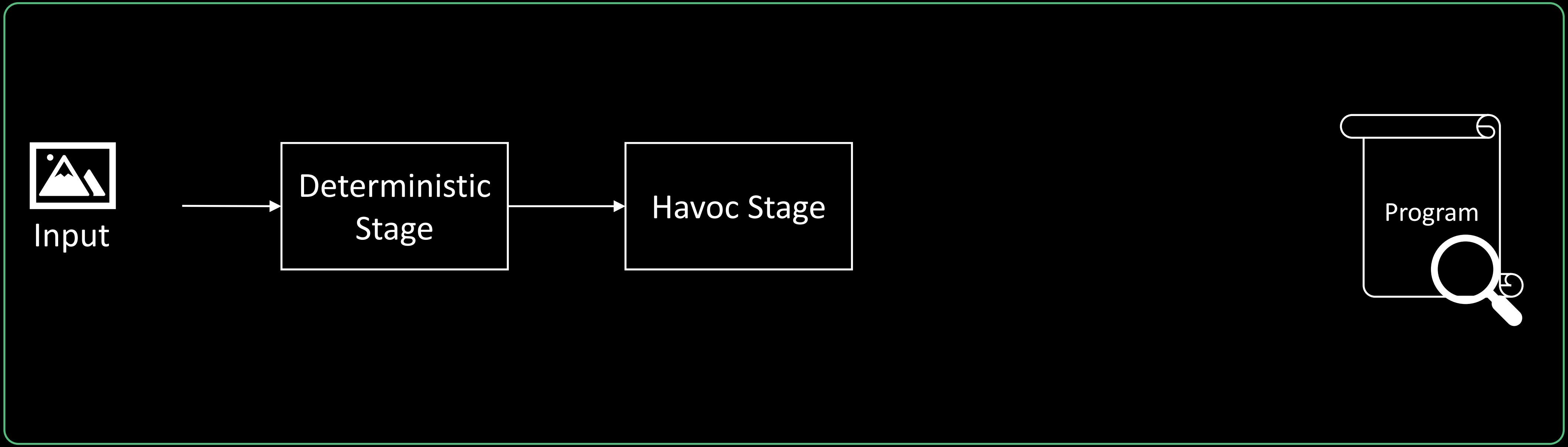
Input

Program

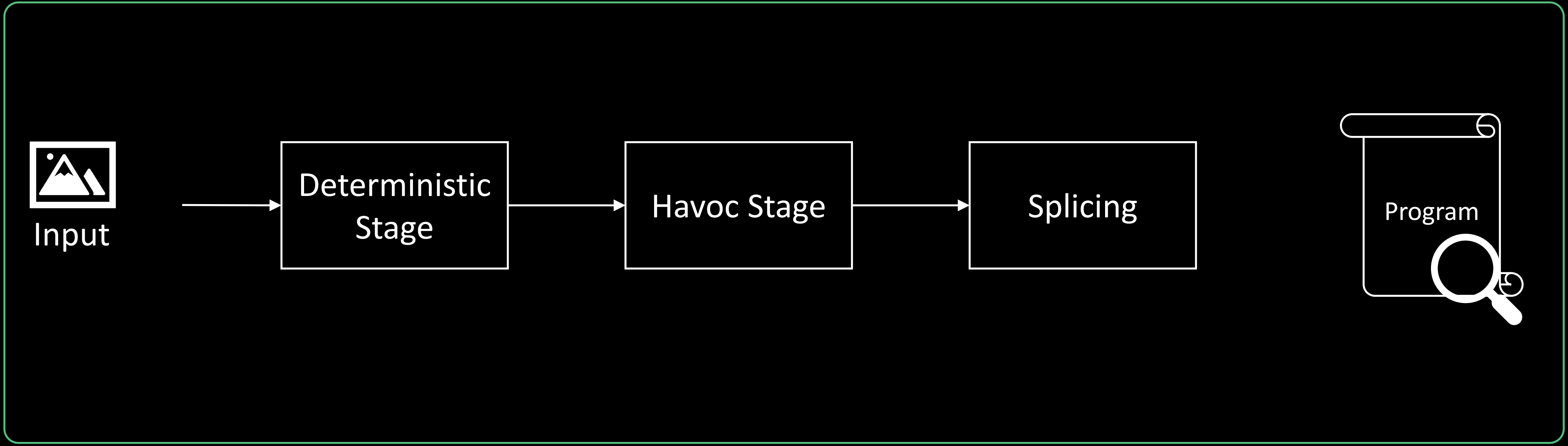
Background – Mutational Fuzzers



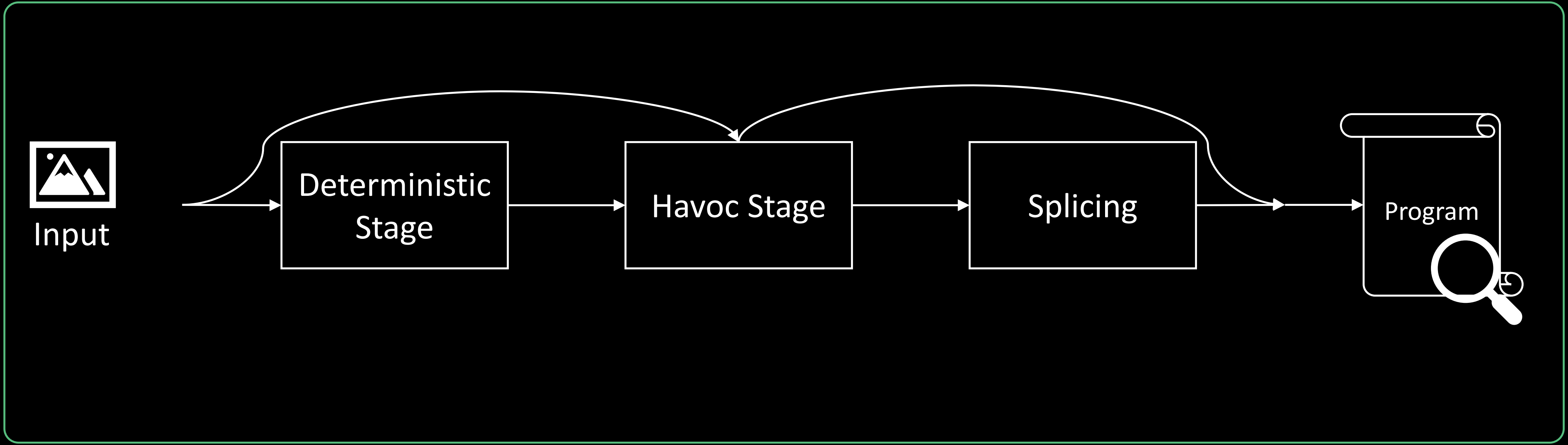
Background – Mutational Fuzzers



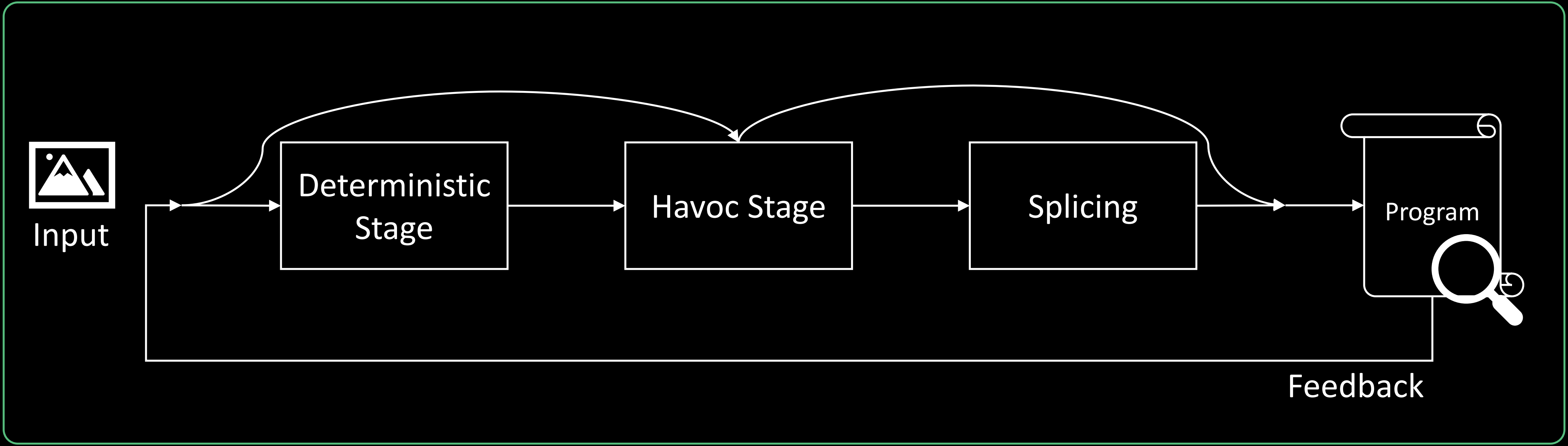
Background – Mutational Fuzzers



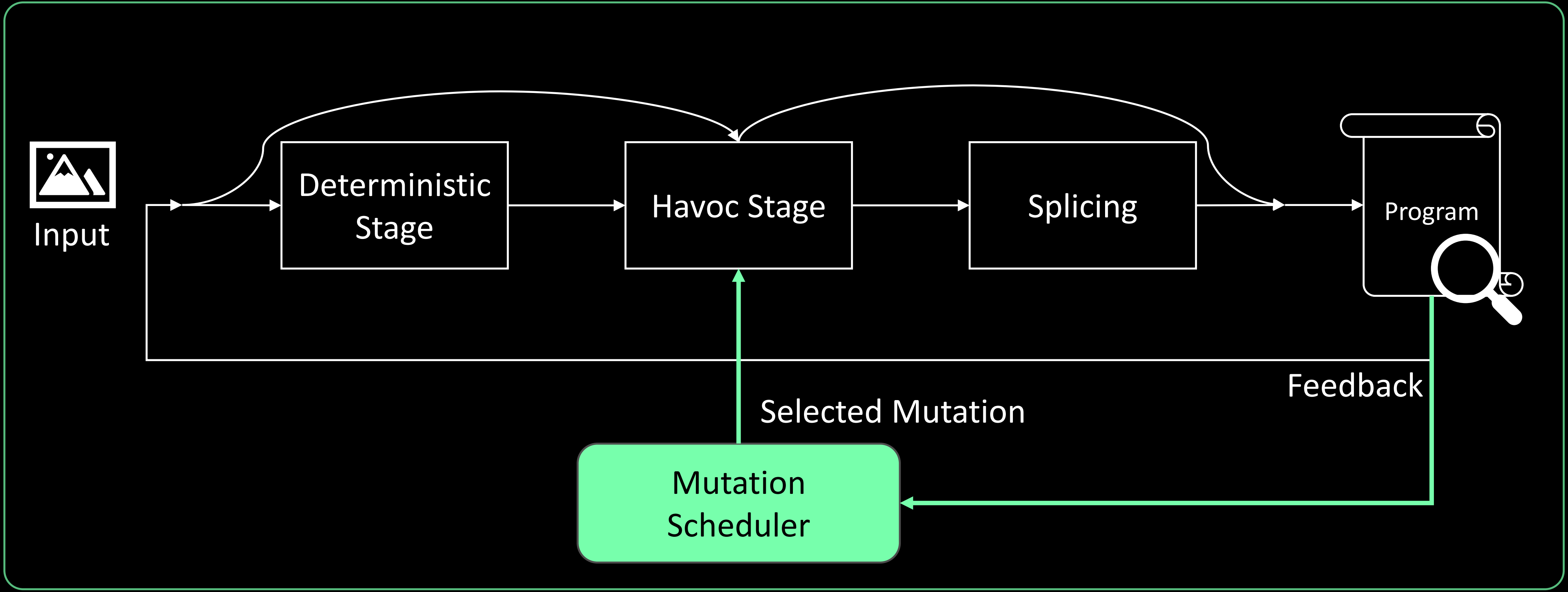
Background – Mutational Fuzzers



Background – Mutational Fuzzers



Background – Mutation Scheduling



Related Work in Algorithmic Improvements

Related Work in Algorithmic Improvements

Mutation Schedulers

MOPT [Lyu et al., USENIX Security 2019]
Fuzzergym [Drozd et al., arXiv 2018]
[Böttinger et al., SPW 2018]

Related Work in Algorithmic Improvements

Mutation Schedulers

MOPT [Lyu et al., USENIX Security 2019]
Fuzzergym [Drozd et al., arXiv 2018]
[Böttinger et al., SPW 2018]

Location Optimization

FairFuzz [Lemieux et al., ASE 2018]
Steelix [Li et al., ESEC/FSE 2017]
[Rajpal et al., arXiv 2017]

Related Work in Algorithmic Improvements

Mutation Schedulers

MOPT [Lyu et al., USENIX Security 2019]
Fuzzergym [Drozd et al., arXiv 2018]
[Böttinger et al., SPW 2018]

Location Optimization

FairFuzz [Lemieux et al., ASE 2018]
Steelix [Li et al., ESEC/FSE 2017]
[Rajpal et al., arXiv 2017]

Other Scheduling Approaches

EcoFuzz [Yue et al., USENIX Security 2020]
NeuFuzz [Wang et al., IEEE Access 2019]

Related Work in Algorithmic Improvements

Mutation Schedulers

MOPT [Lyu et al., USENIX Security 2019]
Fuzzergym [Drozd et al., arXiv 2018]
[Böttinger et al., SPW 2018]

- Fail to show improvements in practice
- Introduce per-target parameters

Location Optimization

FairFuzz [Lemieux et al., ASE 2018]
Steelix [Li et al., ESEC/FSE 2017]
[Rajpal et al., arXiv 2017]

Other Scheduling Approaches

EcoFuzz [Yue et al., USENIX Security 2020]
NeuFuzz [Wang et al., IEEE Access 2019]

Related Work in Algorithmic Improvements

Mutation Schedulers

MOPT [Lyu et al., USENIX Security 2019]
Fuzzergym [Drozd et al., arXiv 2018]
[Böttinger et al., SPW 2018]

- Fail to show improvements in practice
- Introduce per-target parameters

Location Optimization

FairFuzz [Lemieux et al., ASE 2018]
Steelix [Li et al., ESEC/FSE 2017]
[Rajpal et al., arXiv 2017]

- Optimize location, but not the associated operation
- Expensive (to integrate)

Other Scheduling Approaches

EcoFuzz [Yue et al., USENIX Security 2020]
NeuFuzz [Wang et al., IEEE Access 2019]

Related Work in Algorithmic Improvements

Mutation Schedulers

MOPT [Lyu et al., USENIX Security 2019]
Fuzzergym [Drozd et al., arXiv 2018]
[Böttinger et al., SPW 2018]

- Fail to show improvements in practice
- Introduce per-target parameters

Location Optimization

FairFuzz [Lemieux et al., ASE 2018]
Steelix [Li et al., ESEC/FSE 2017]
[Rajpal et al., arXiv 2017]

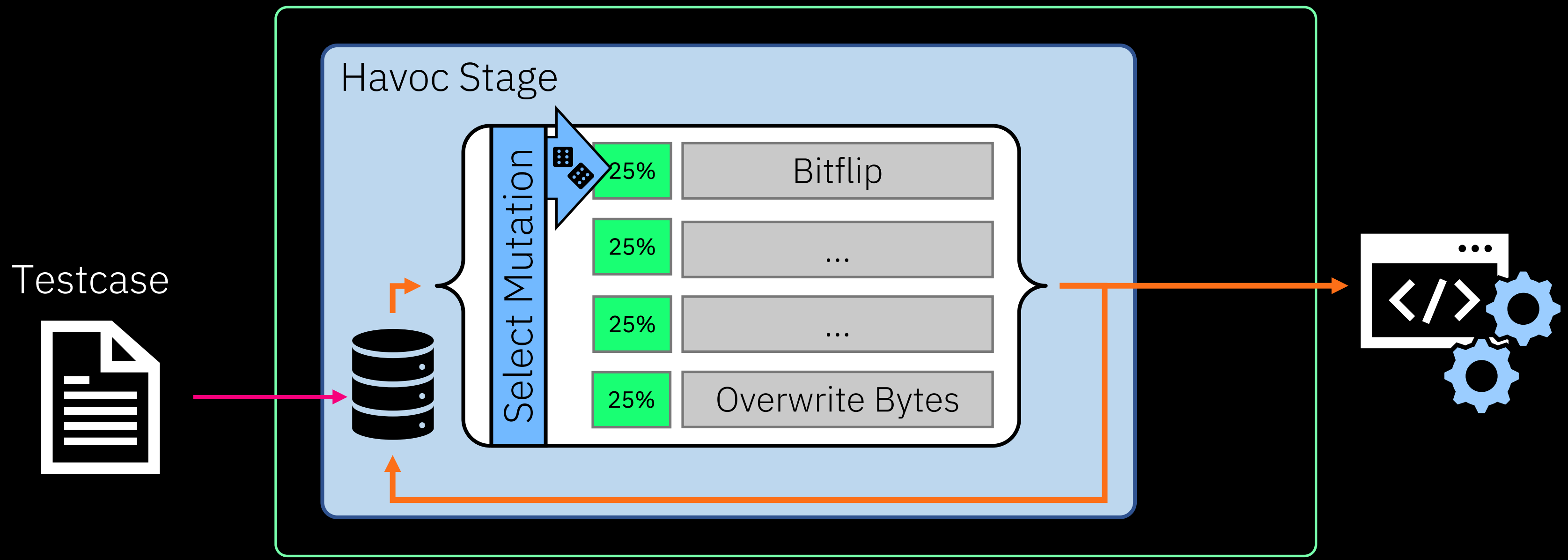
- Optimize location, but not the associated operation
- Expensive (to integrate)

Other Scheduling Approaches

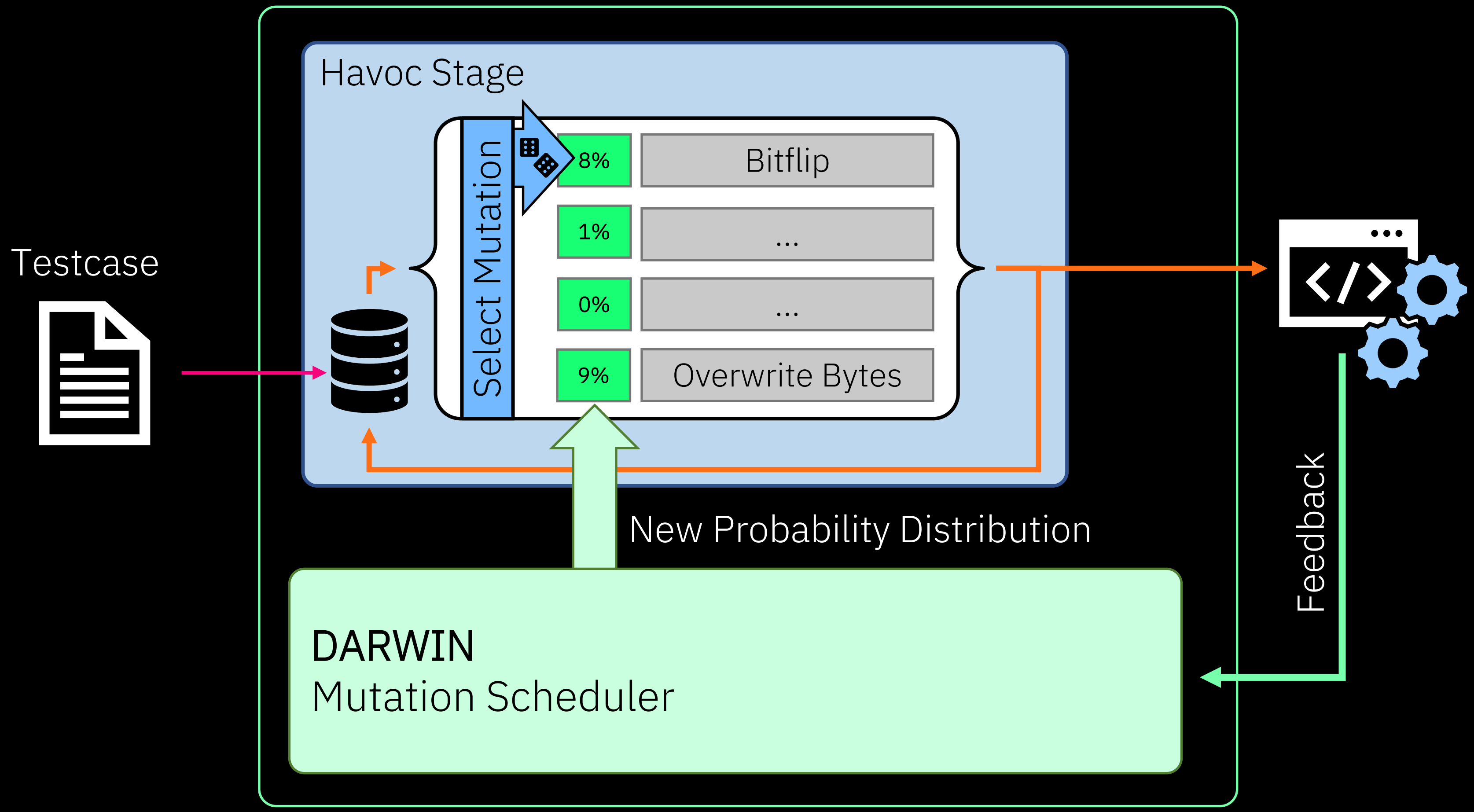
EcoFuzz [Yue et al., USENIX Security 2020]
NeuFuzz [Wang et al., IEEE Access 2019]

- Optimization goal applied very early in fuzzing loop
- Interesting: combining seed selection and mutation scheduling

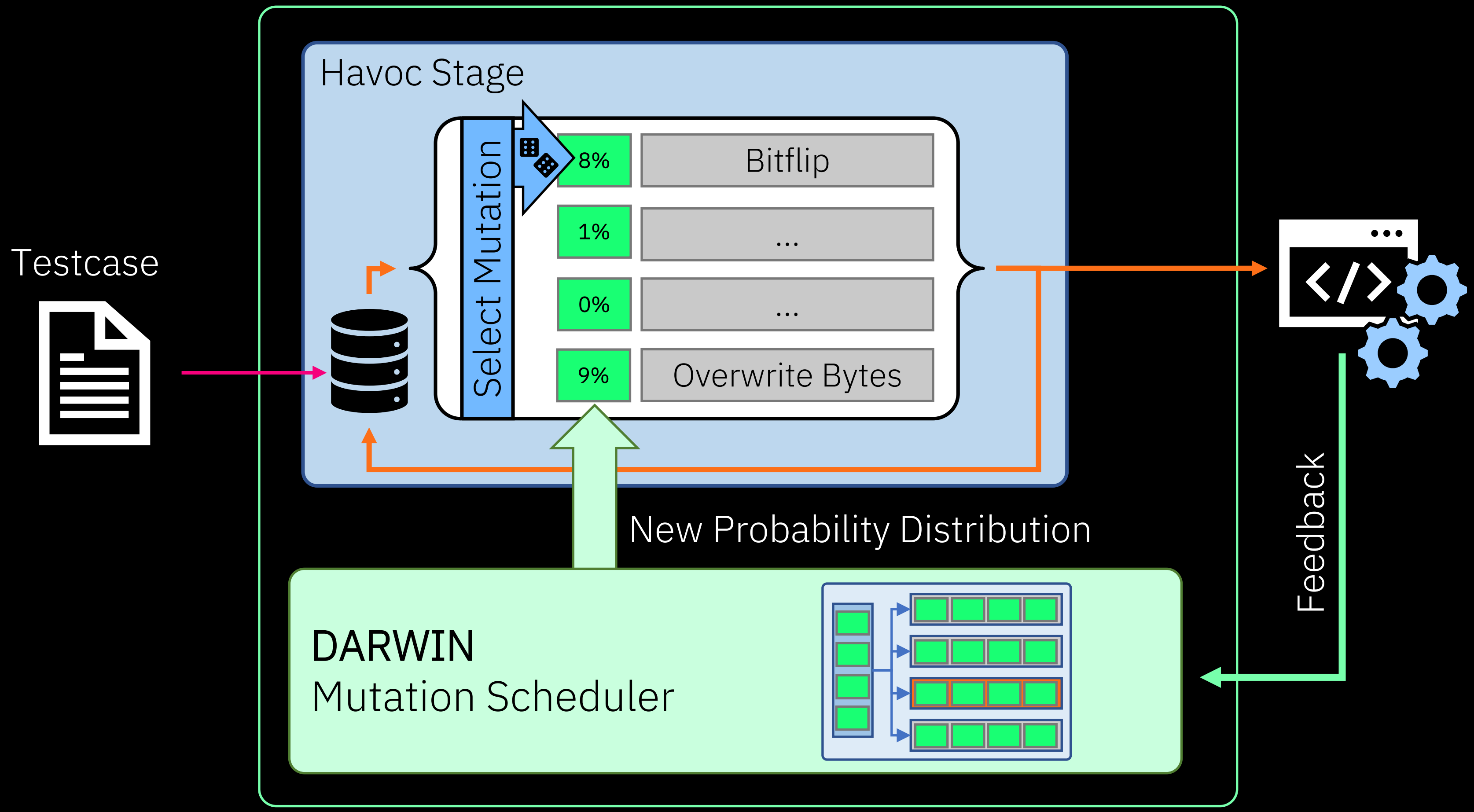
DARWIN


Testcase
Havoc Stage
Select Mutation
25%
Bitflip
25%
...
25%
...
25%
Overwrite Bytes

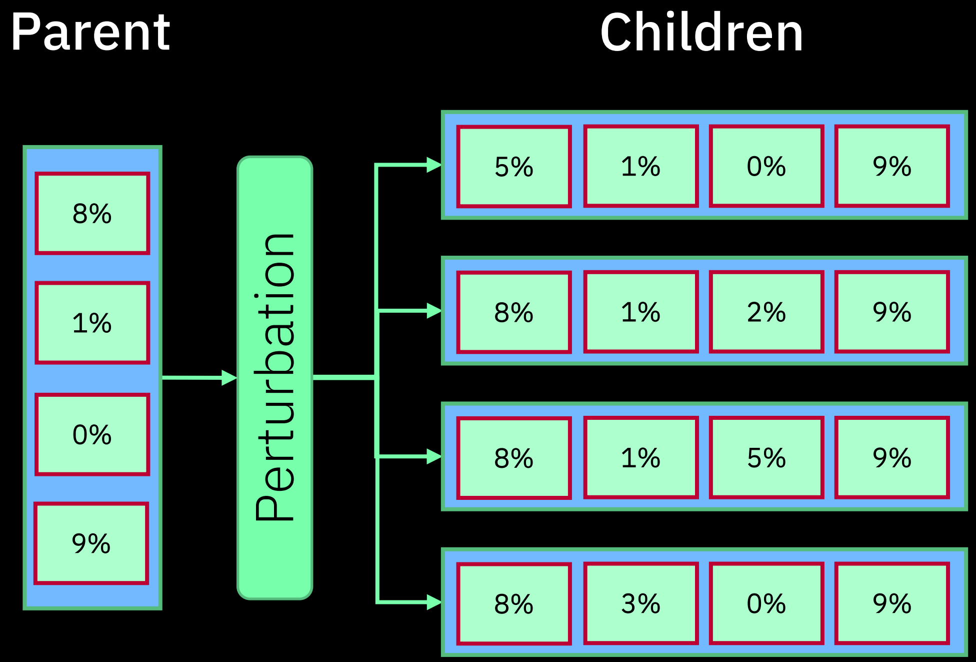
DARWIN


Havoc Stage
Select Mutation
8%
Bitflip
1%
...
0%
...
9%
Overwrite Bytes
Testcase
New Probability Distribution
Feedback
DARWIN
Mutation Scheduler

DARWIN


Testcase
Havoc Stage
Select Mutation
8%
Bitflip
1%
...
0%
...
9%
Overwrite Bytes
New Probability Distribution
DARWIN
Mutation Scheduler
Feedback

DARWIN – Evolution Strategy

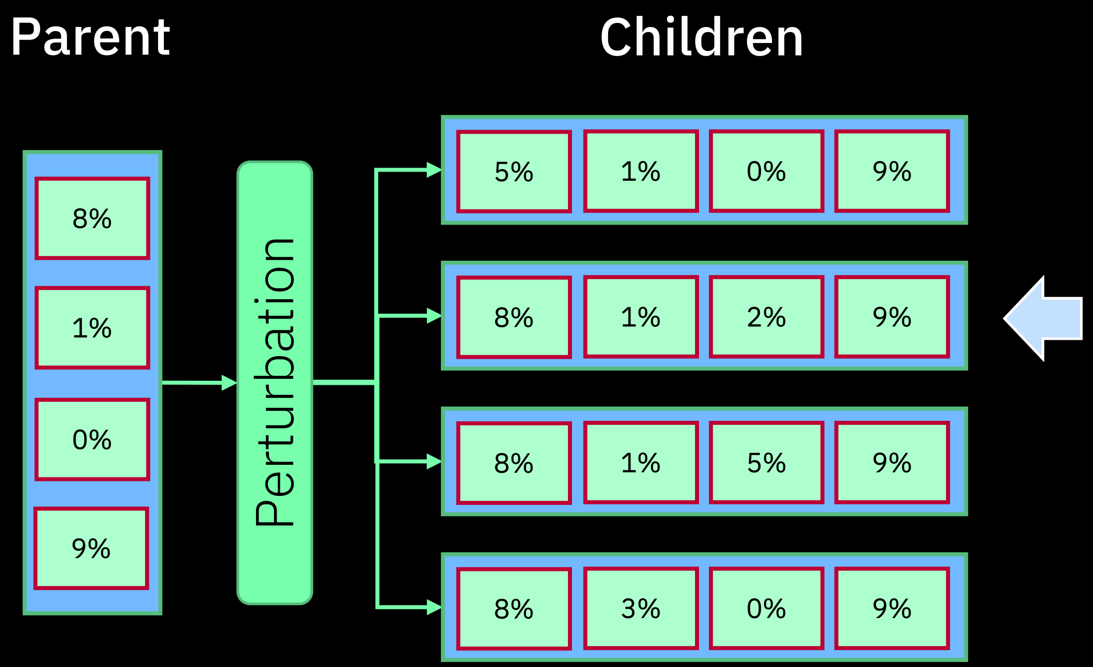


Parent: 8% 1% 0% 9%

Children:

5%	1%	0%	9%
8%	1%	2%	9%
8%	1%	5%	9%
8%	3%	0%	9%

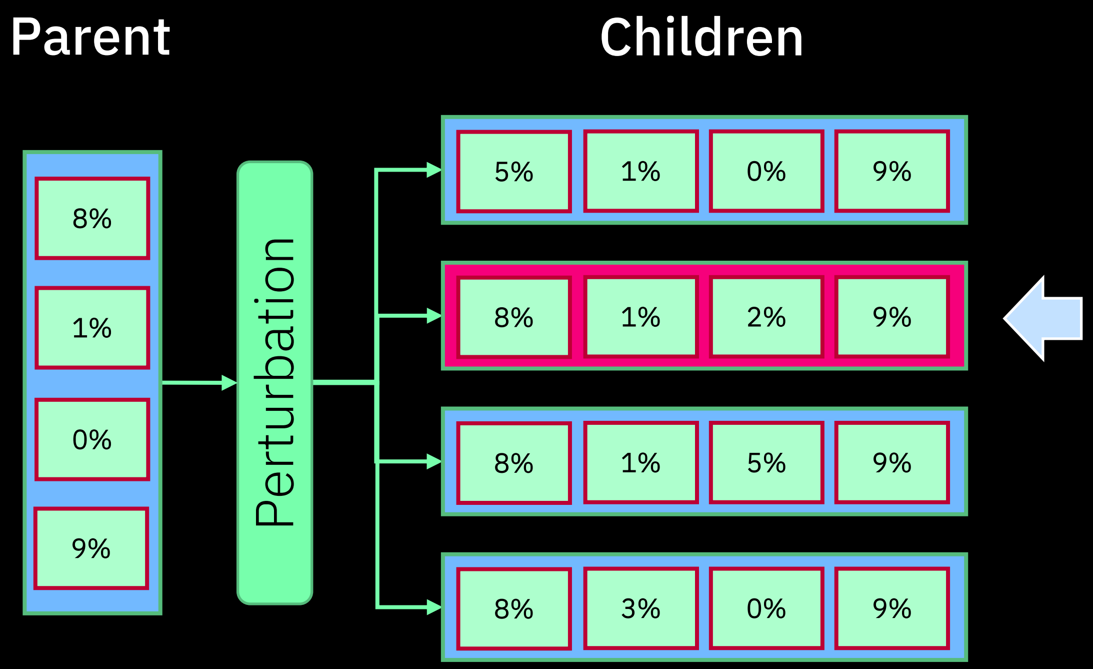
DARWIN – Evolution Strategy



Fitness function determines next parent

fitness = #new unique paths

DARWIN – Evolution Strategy

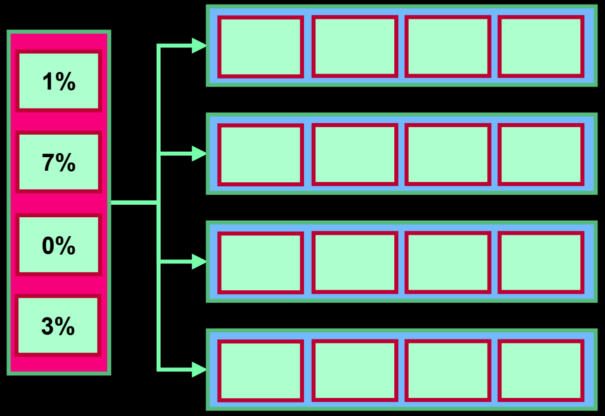


Fitness function determines next parent

fitness = #new unique paths

DARWIN – Evolution Strategy

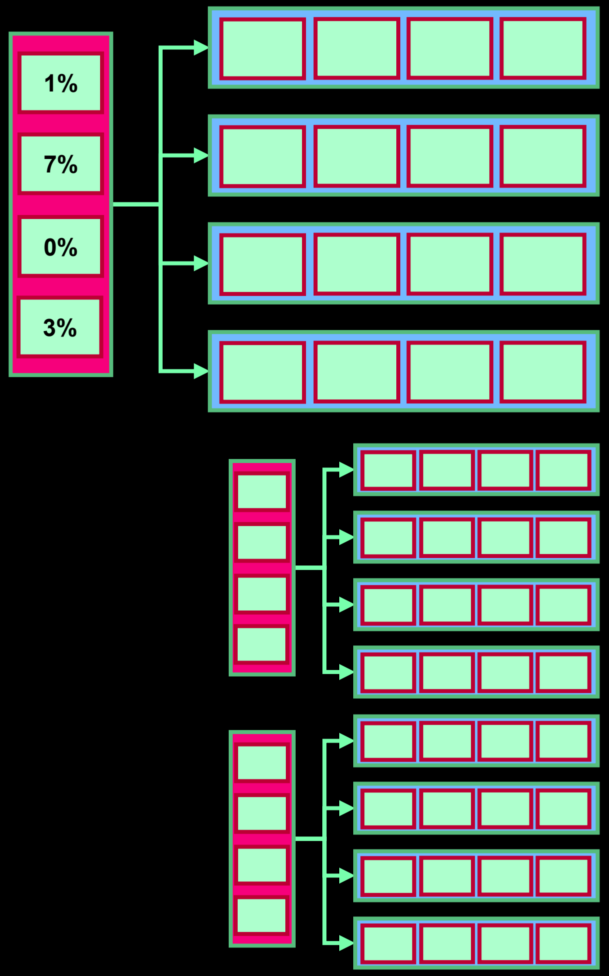
- Very simple and efficient
- Problem: very local algorithm
  - Low probability of escaping local optima



DARWIN – Evolution Strategy

- Very simple and efficient
- Problem: very local algorithm
  - Low probability of escaping local optima

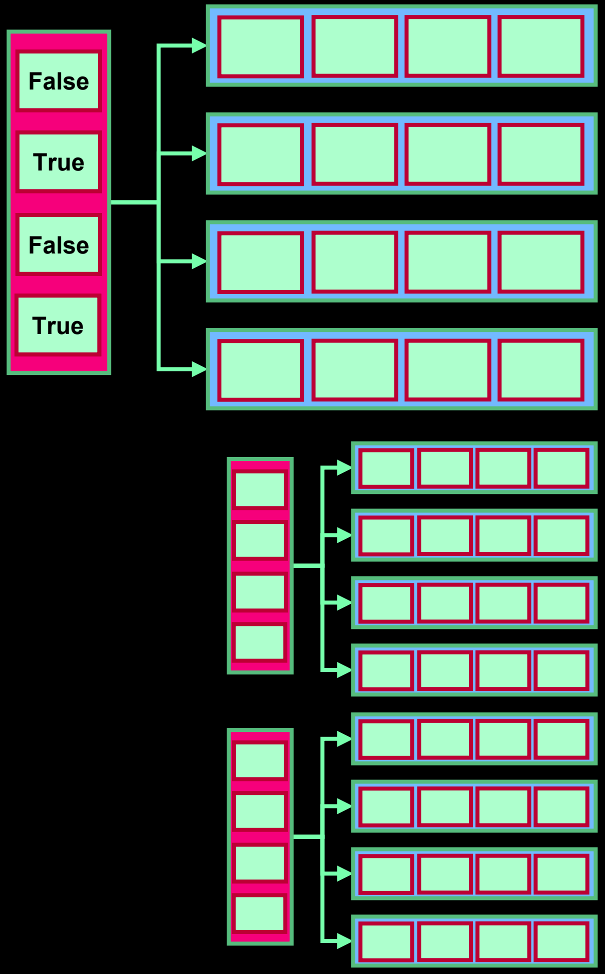
- Solution: multi-parent ES
  - μ parents, λ children
    - 5 parents, 4 children seemed best
  - Cycle through best parent solutions
  - In addition: Binary representation



DARWIN – Evolution Strategy

- Very simple and efficient
- Problem: very local algorithm
  - Low probability of escaping local optima

- Solution: multi-parent ES
  - μ parents, λ children
    - 5 parents, 4 children seemed best
  - Cycle through best parent solutions
  - In addition: Binary representation



DARWIN - Contributions

- Leveraging Evolution Strategy to optimize mutation scheduling
- Keeping execution speed high
- No target-dependent parameters
- Easy to integrate into mutational fuzzers

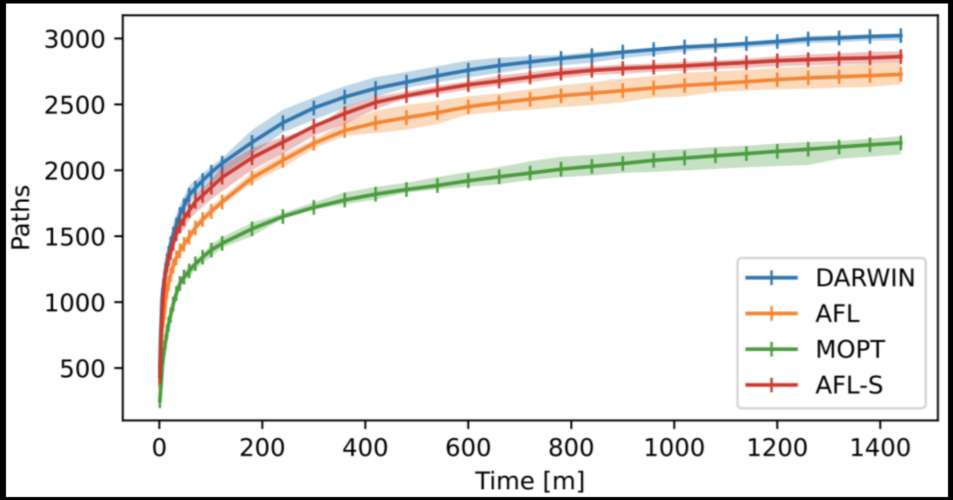
Evaluation

- Is mutation scheduling a dynamic problem?
- Does it make sense to trade in speed for efficiency?
- Is there an improvement in
  - Coverage?
  - Time to coverage?
  - Bugs?

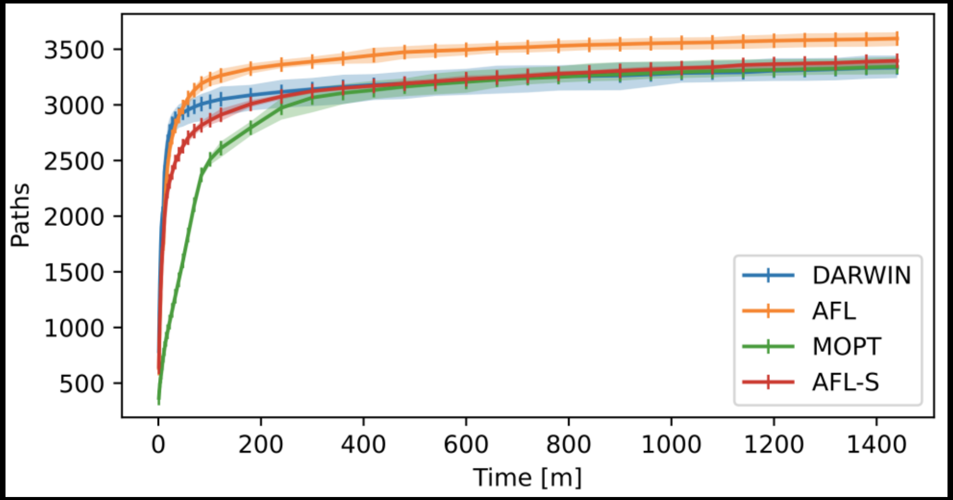
Evaluation - Coverage

- Binutils suite, bsdtar, djpeg, jhead, tcpdump
- Edge coverage: +6.77% vs. MOPT, +1.73% vs. AFL
  - +4.38% vs. static variant (AFL-S)!
- At disadvantage for targets expecting highly-structured input

size



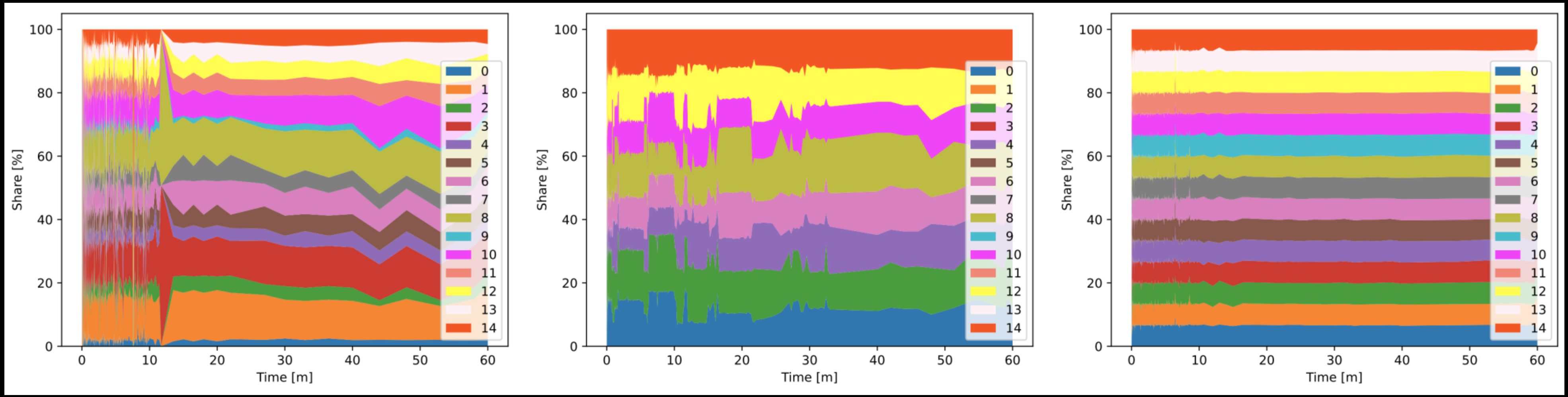
cxxfilt



Evaluation – Mutation Histories

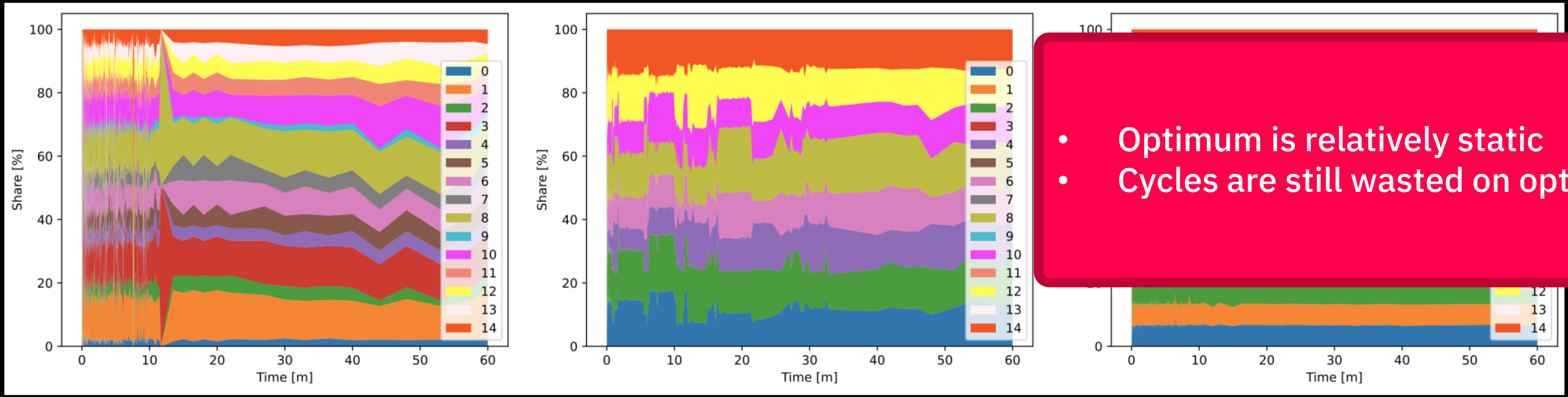
cxxfilt

DARWIN MOPT AFL



Evaluation – Mutation Histories

cxxfilt



DARWIN

MOPT

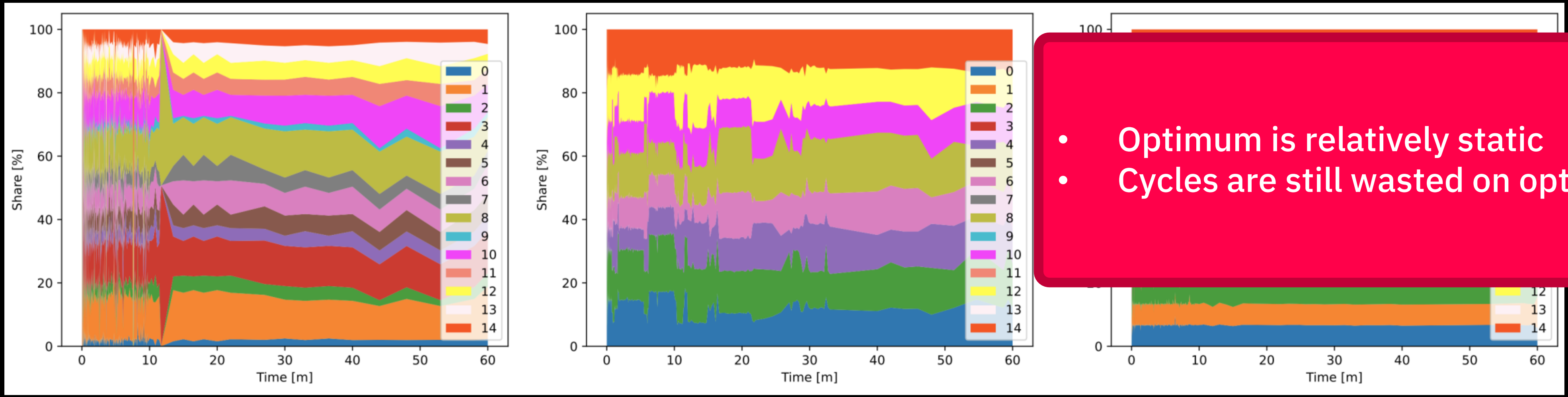
AFL

- Optimum is relatively static
- Cycles are still wasted on optimization

Evaluation – Mutation Histories

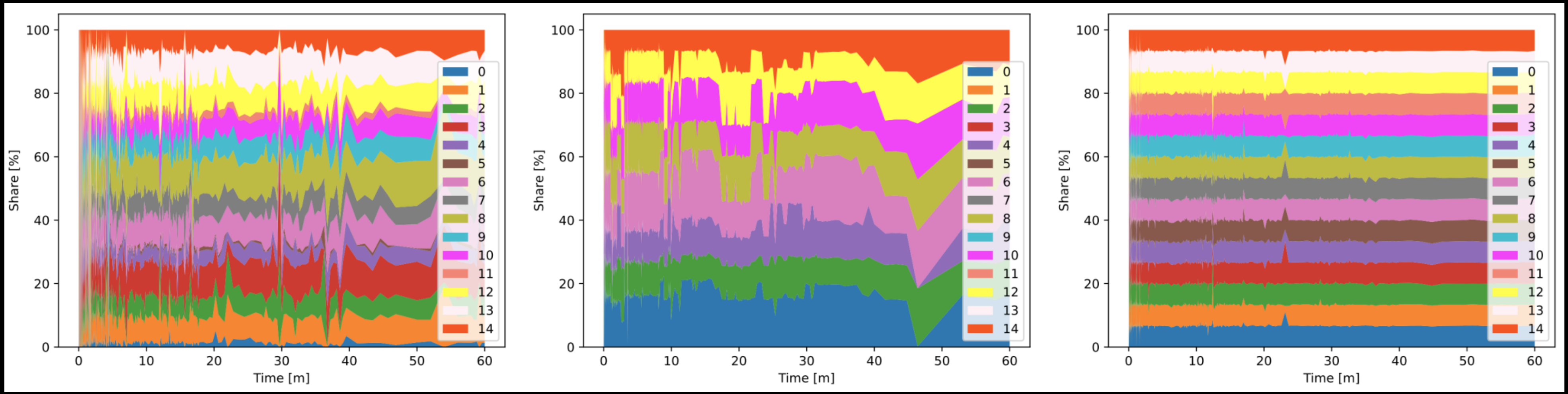
cxxfilt

DARWIN MOPT AFL



- Optimum is relatively static
- Cycles are still wasted on optimization

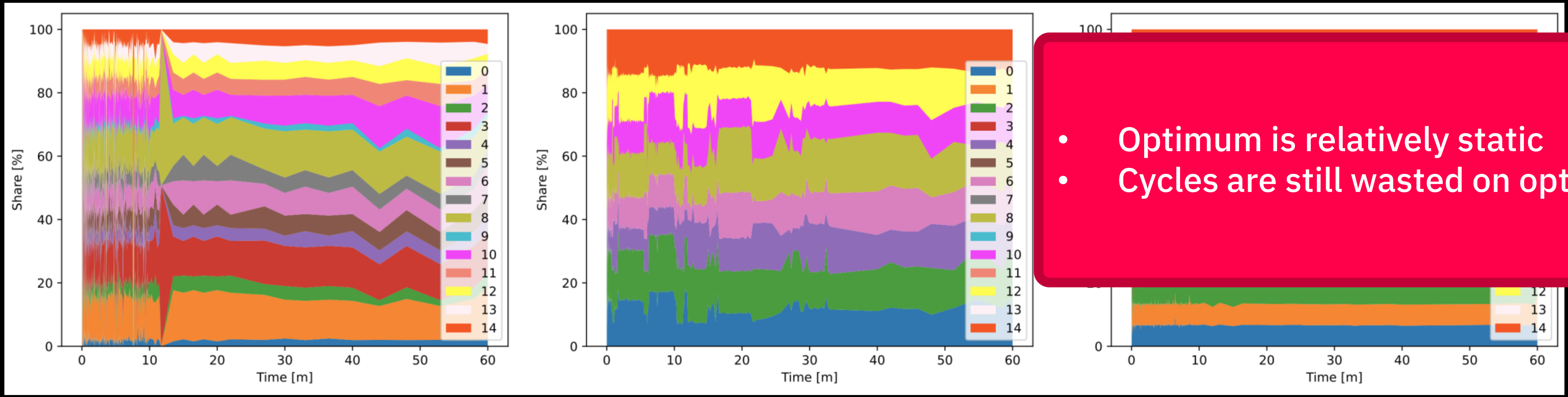
size



Evaluation – Mutation Histories

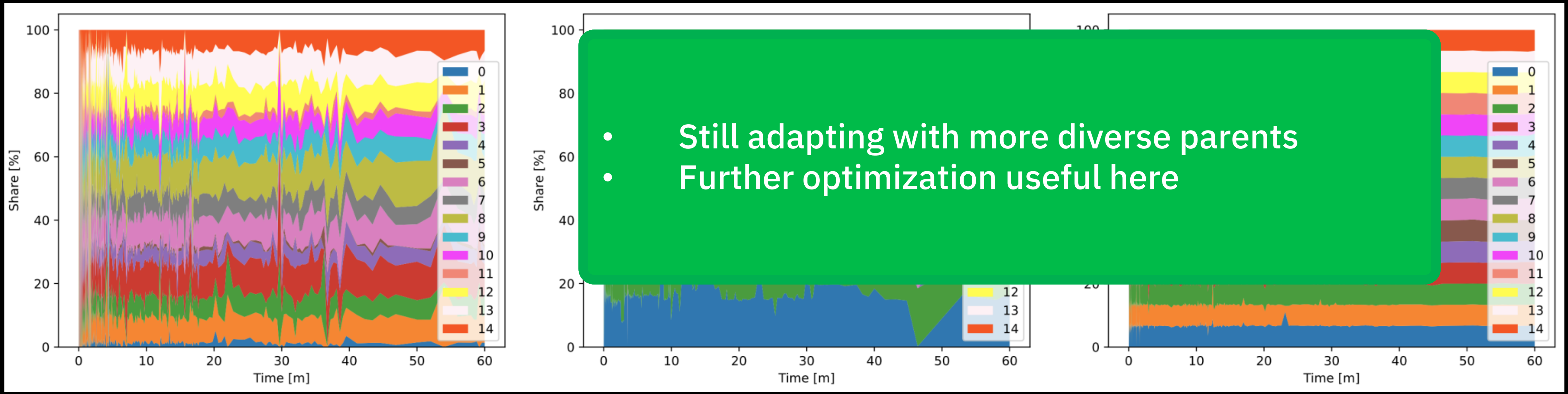
cxxfilt

DARWIN MOPT AFL



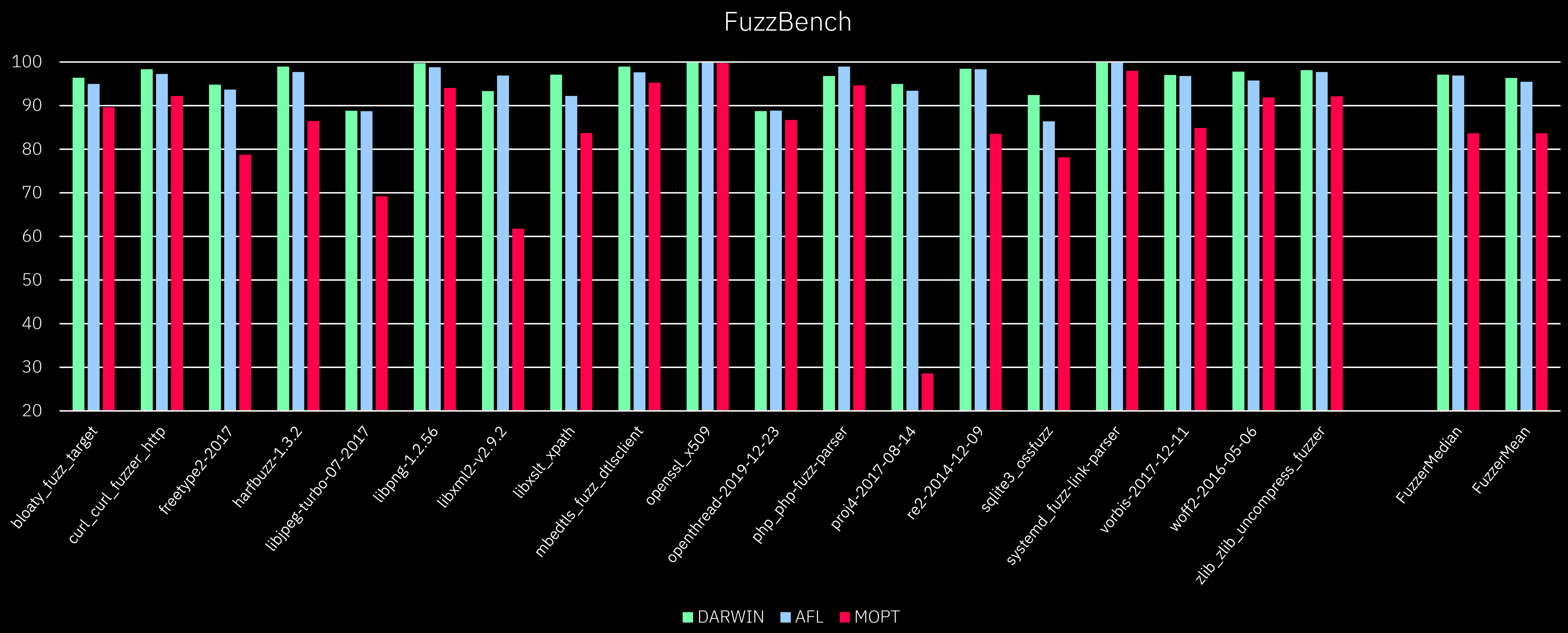
- Optimum is relatively static
- Cycles are still wasted on optimization

size

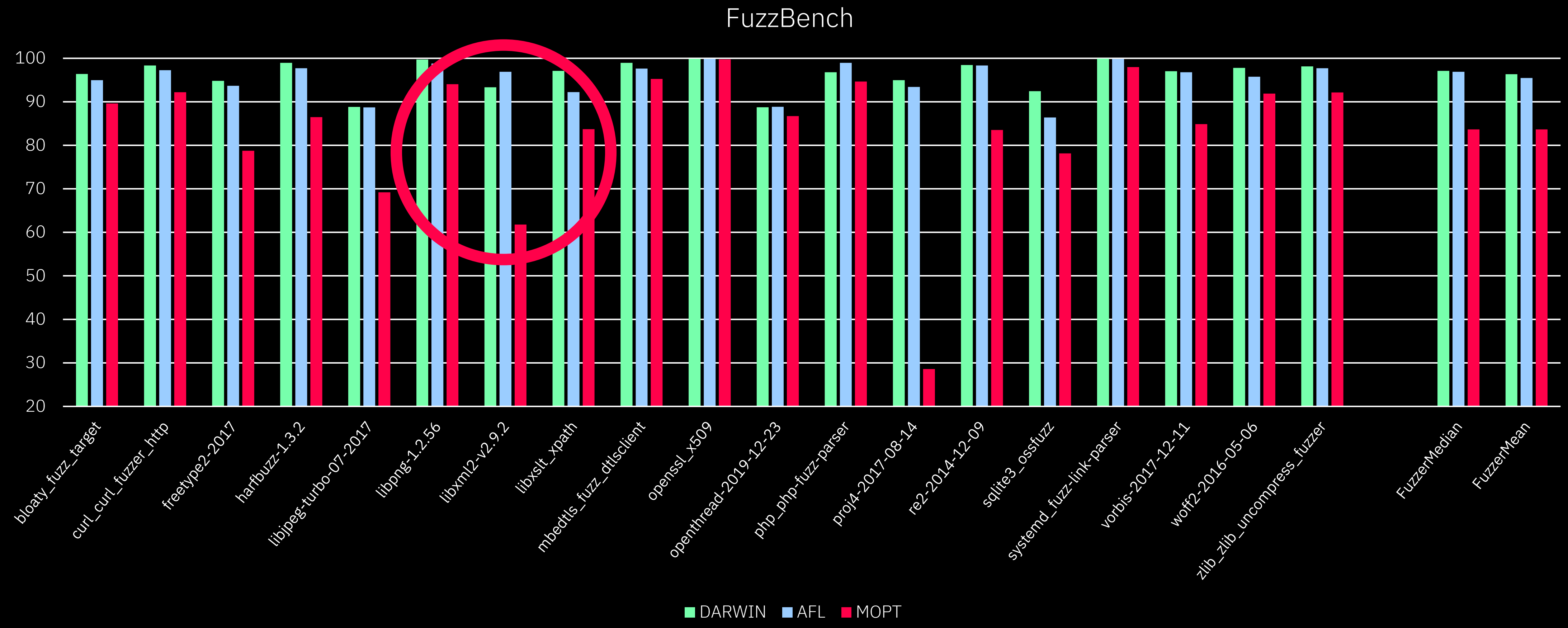


- Still adapting with more diverse parents
- Further optimization useful here

Evaluation - FuzzBench



Evaluation - FuzzBench

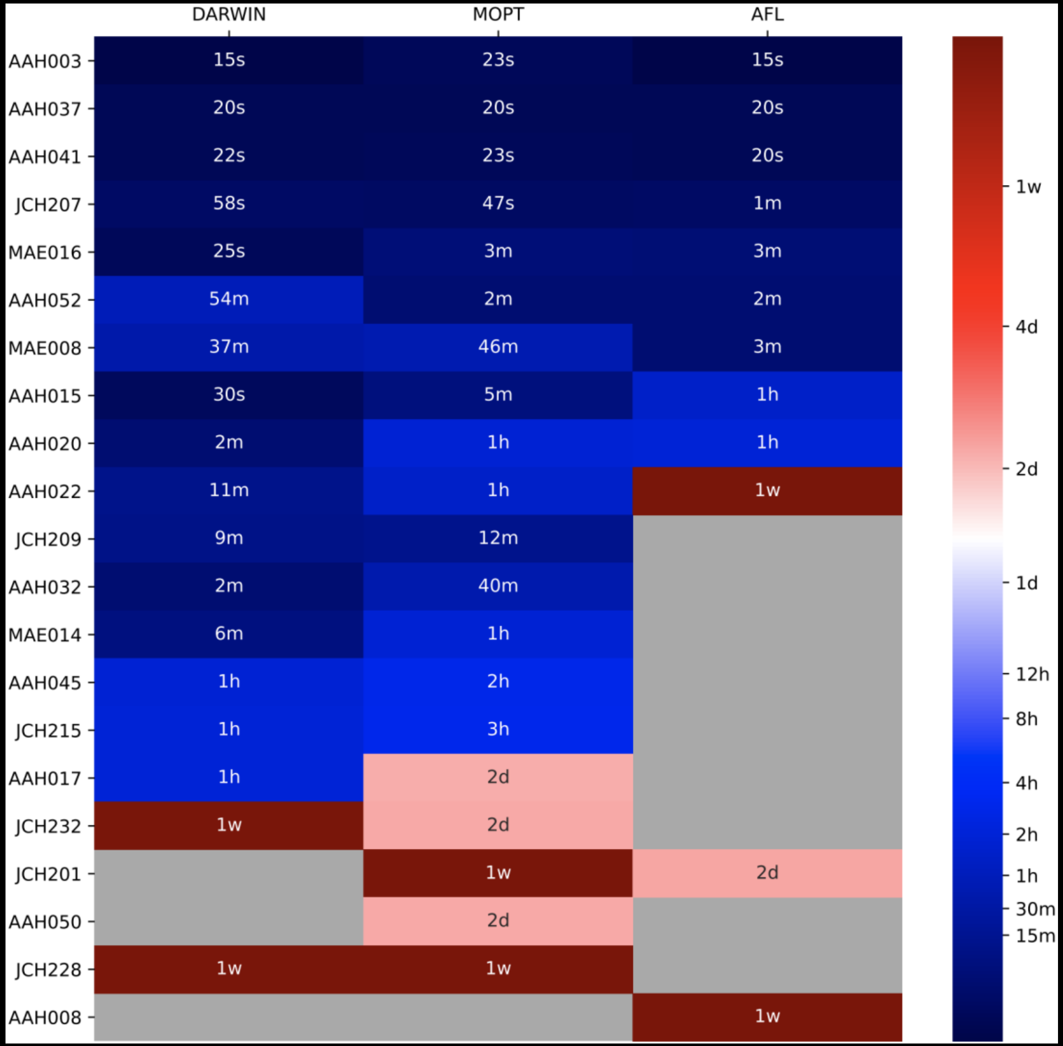


Evaluation - MAGMA

- MAGMA: Benchmark suite to find backported bugs

- Different reports, in this case: survival analysis (“time to bug”)

- DARWIN finds 15/21 bugs fastest



	DARWIN	MOPT	AFL
AAH003	15s	23s	15s
AAH037	20s	20s	20s
AAH041	22s	23s	20s
JCH207	58s	47s	1m
MAE016	25s	3m	3m
AAH052	54m	2m	2m
MAE008	37m	46m	3m
AAH015	30s	5m	1h
AAH020	2m	1h	1h
AAH022	11m	1h	1w
JCH209	9m	12m	
AAH032	2m	40m	
MAE014	6m	1h	
AAH045	1h	2h	
JCH215	1h	3h	
AAH017	1h	2d	
JCH232	1w	2d	
JCH201		1w	2d
AAH050		2d	
JCH228	1w	1w	
AAH008			1w

Magma: A ground-truth fuzzing benchmark, Hazimeh et al., 2020

Evaluation - Crashes

- Crash experiment based on coverage targets
  - Max: unique bugs within one run
  - Uniq: unique bugs over all ten runs

- DARWIN variants outperform MOPT, AFL, EcoFuzz, and AFL-S
- One novel bug in objcopy: memory leak

	DARWIN	**AFL**	**AFL-S**	**MOPT**	**EcoFuzz-D**	**EcoFuzz**
Max	**7**	4	5	1	**18**	1
Unique	**20**	12	12	2	**26**	1

Conclusion

- DARWIN is the first ES-based mutation scheduler
- Adaptive optimization outperforms static optimization
- Significant improvement in bug-finding capabilities

Contact: darwin@sanctuary.dev