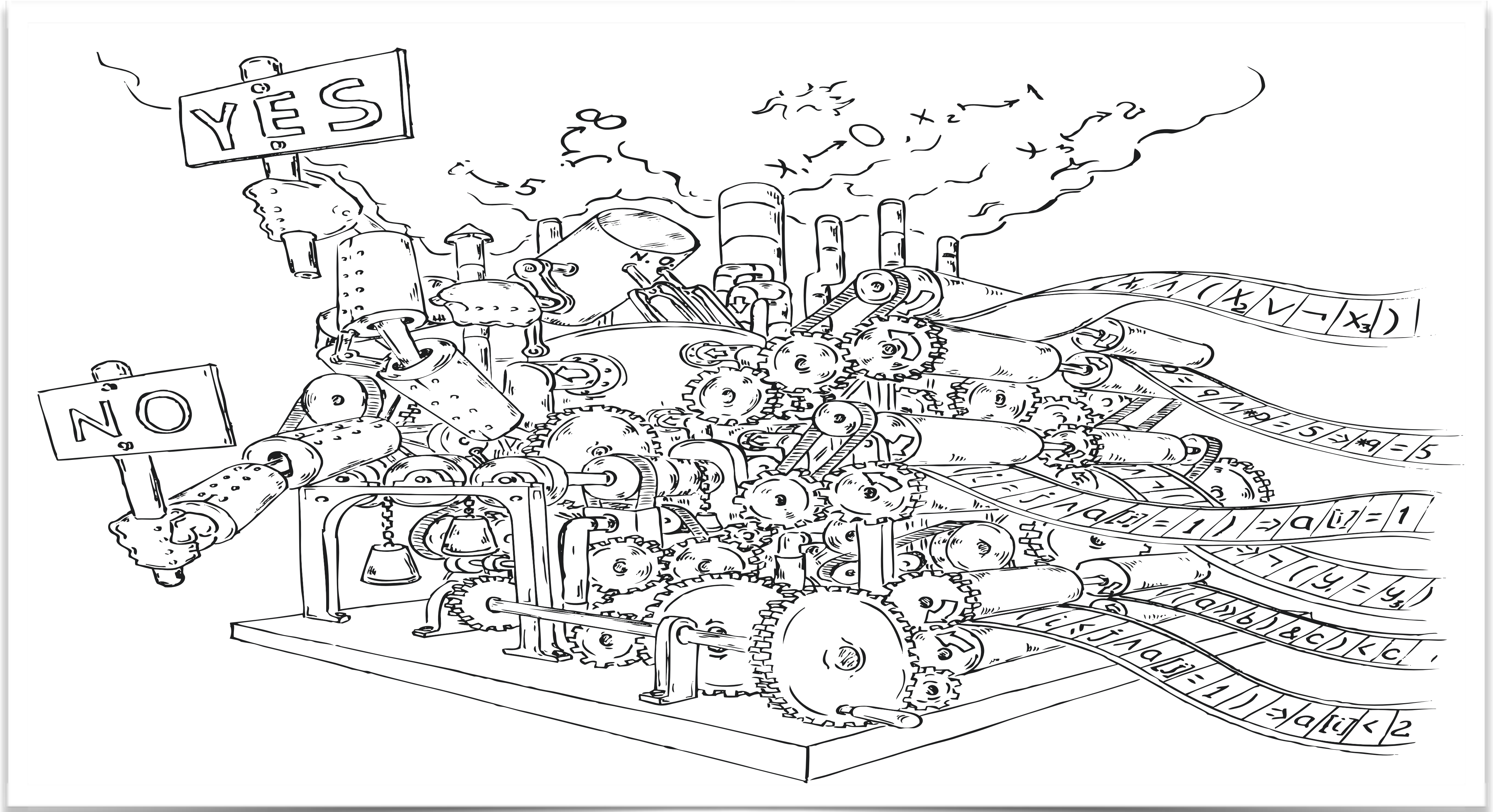
ACM International Conference on the Foundations of Software Engineering
Mon 23 - Fri 27 June 2025 Trondheim, Norway

QSF: Multi-objective Optimization Based Efficient Solving for Floating-Point Constraints

Zhenbang Chen
zbchen@nudt.edu.cn

Joint work with Xu Yang, Wei Dong and Ji Wang

Constraint Solving

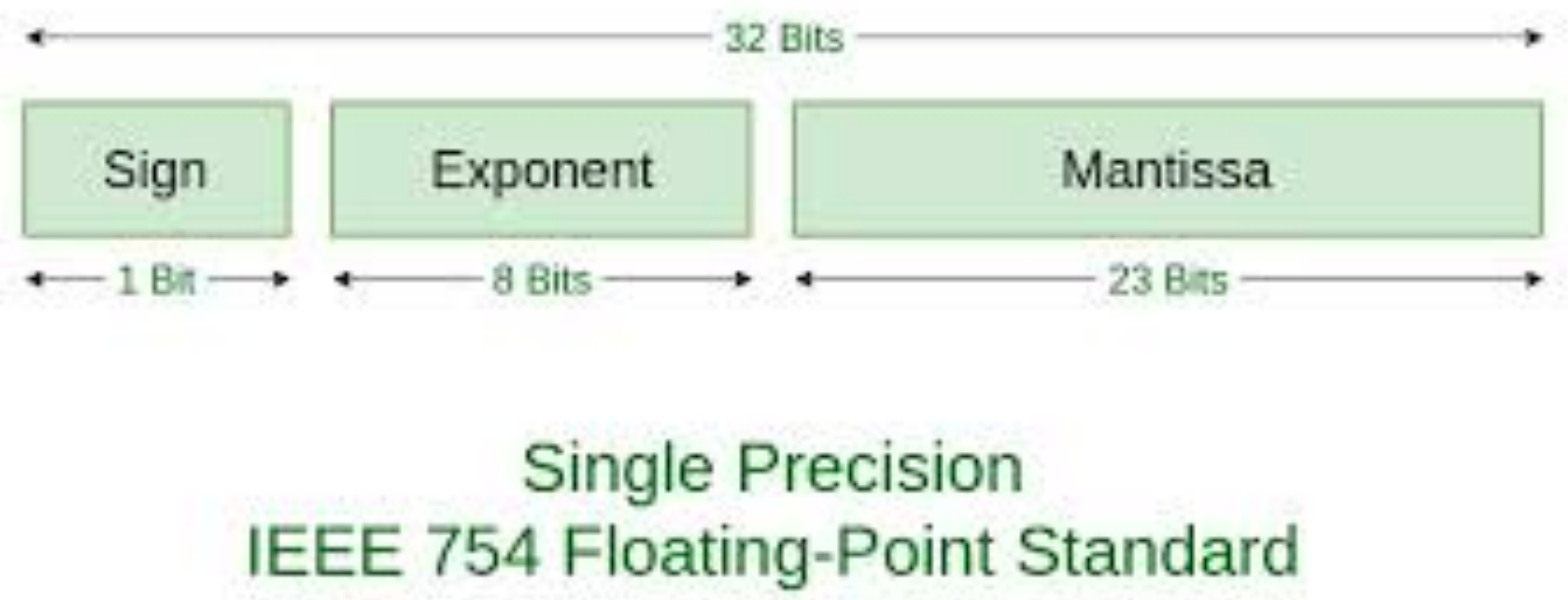


Decision Procedures An Algorithmic Point of View, Second Edition, 2016

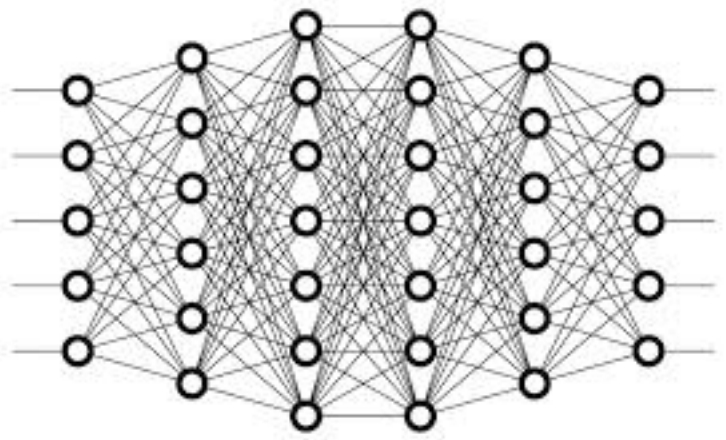
Z3 CVC5 MathSAT 5

Floating-point (FP) Constraint Solving

- Highly demanded for analyzing and verifying FP programs



Explosion of first Ariane 5 flight



SOTA FP Solving Methods

- Bit-blasting Based Approaches
  - Z3, MathSAT5, CVC5, ...

Not scalable

$$a^3 + b^3 > 0$$

- Z3: > 8s
- CVC5: > 2s
- MathSAT5: > 3s

12th Gen Intel(R) Core(TM) i9-12900H CPU@3.0GHz

SOTA FP Solving Methods

- Bit-blasting Based Approaches
  - Z3, MathSAT5, CVC5, ...
- Real Arithmetic Based Approaches
  - dReal, COLIBRI, ...

Not scalable

Not sound

$$(a + b) + c \neq a + (b + c)$$

SAT if *a, b, and c are FP numbers*

SOTA FP Solving Methods

- Bit-blasting Based Approaches
  - Z3, MathSAT5, CVC5, ...
- Real Arithmetic Based Approaches
  - dReal, COLIBRI, ...
- Search-Based Approaches
  - JFS, Xsat, ...

Not scalable

Not sound

Not complete

$a - 1.0 = 1.1$ UNSAT if *a* is a 32-bits FP number

FP Constraint Solving is Challenging

- Precise encoding is expensive

$$a^3 + b^3 > 0$$

  - Z3: > 8s
  - CVC5: > 2s
  - MathSAT5: > 3s

12th Gen Intel(R) Core(TM) i9-12900H CPU@3.0GHz

- Real number encoding is unsound

$$(a+b)+c \neq a+(b+c)$$

SAT if *a, b, and c are FP numbers*

- Search-based method is incomplete

$$a - 1.0 = 1.1$$

UNSAT if *a* is a 32-bits FP number

QSF's Target

- Bit-blasting Based Approaches
  - Z3, MathSAT5, CVC5, ...
- Real Arithmetic Based Approaches
  - dReal, COLIBRI, ...
- Search-Based Approaches
  - JFS, Xsat, ...

Not scalable

Not sound

Improve the efficiency of solving FP constraints

Our Observation

- Only single object function is employed for searching
  - Fuzzing based: #unsatisfied atomic constraints
  - Optimization based: fitness (distance) function's result

Our Observation

- Only single object function is employed for searching
  - Fuzzing based: #unsatisfied atomic constraints
  - Optimization based: fitness (distance) function's result

$$x \geq 64 \wedge y = 64$$

$[x \mapsto 0, y \mapsto 64]$ $f_1 = 1$

$[x \mapsto 63, y \mapsto 63]$ $f_1 = 2$

According to f_1, the first assignment will be prioritized

Our Observation

- Only single object function is employed for searching
  - Fuzzing based: #unsatisfied atomic constraints
  - Optimization based: fitness (distance) function's result

$$f_2 = |x - 64| + |y - 64|$$

$$x \geq 64 \land y = 64$$

$[x \mapsto 0, y \mapsto 64]$	$[x \mapsto 63, y \mapsto 63]$
$f_1 = 1$	$f_1 = 2$
$f_2 = 64$	$f_2 = 2$

According to f_2, the second assignment will be prioritized

Our Key Insight

- Only single object function is employed for searching
  - Fuzzing based: #unsatisfied atomic constraints
  - Optimization based: fitness (distance) function's result

$$x \geq 64 \land y = 64$$

$[x \mapsto 0, y \mapsto 64]$	$[x \mapsto 63, y \mapsto 63]$
$f_1 = 1$	$f_1 = 2$
$f_2 = 64$	$f_2 = 2$

Consider both f_1 and f_2 in the search procedure

Our Key Insight

- Mutation operators in optimization can be customized for FPs

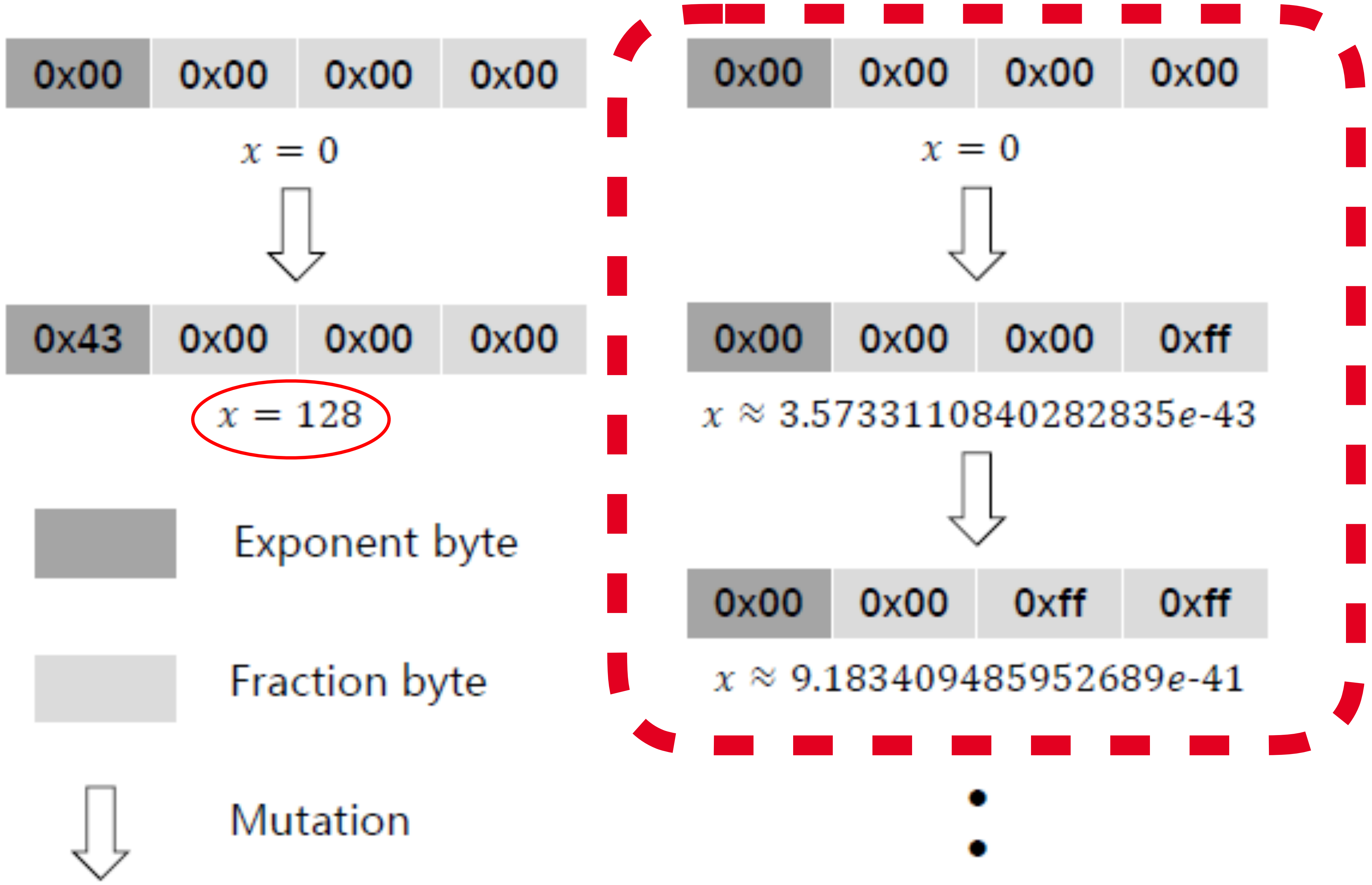
$$x \geq 64 \wedge y = 64$$

$$[x \mapsto 0, y \mapsto 64]$$

Fraction Mutation

Multiple steps

• • •



Our Key Insight

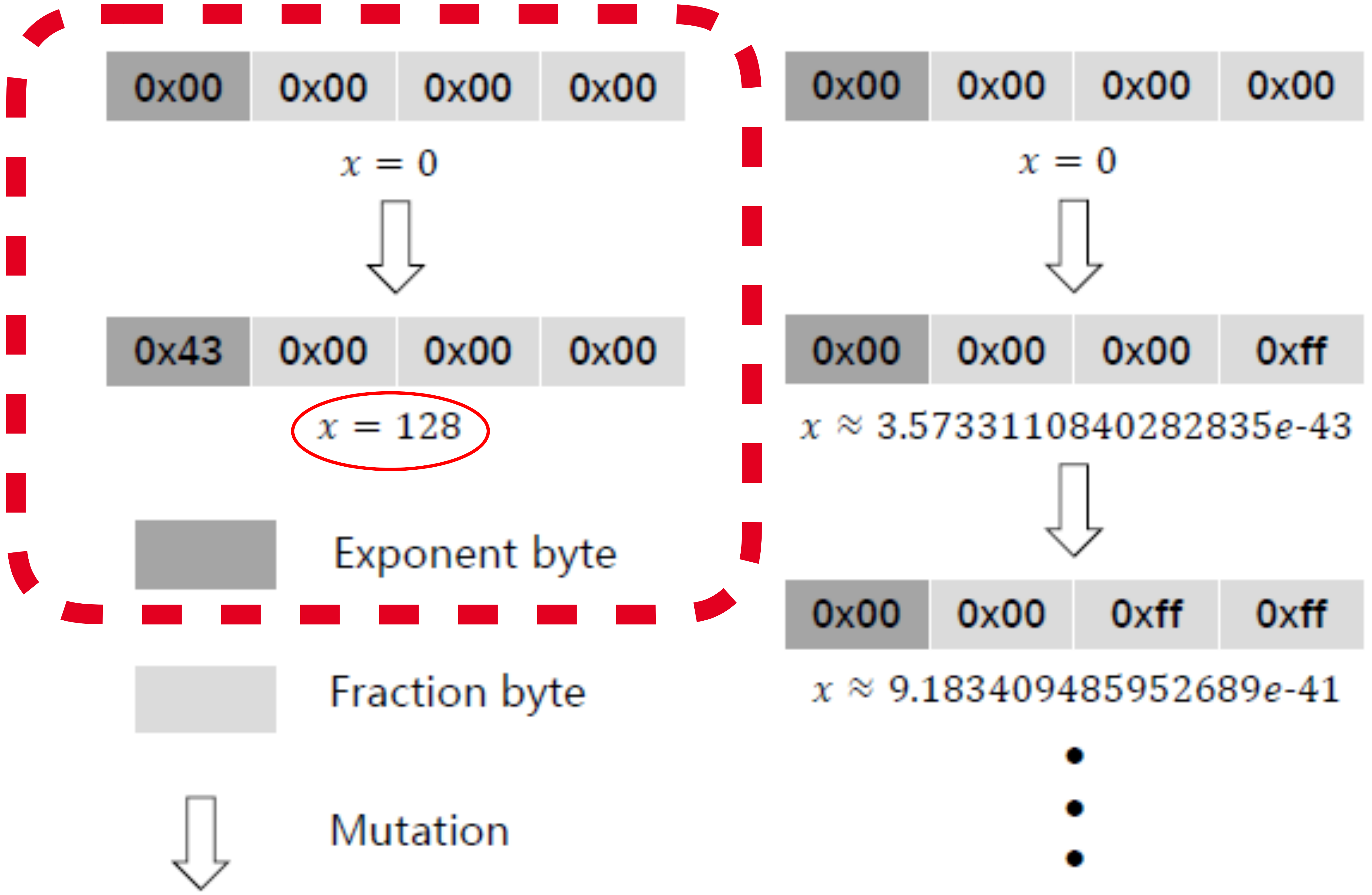
- Mutation operators in optimization can be customized for FPs

$$x \geq 64 \wedge y = 64$$

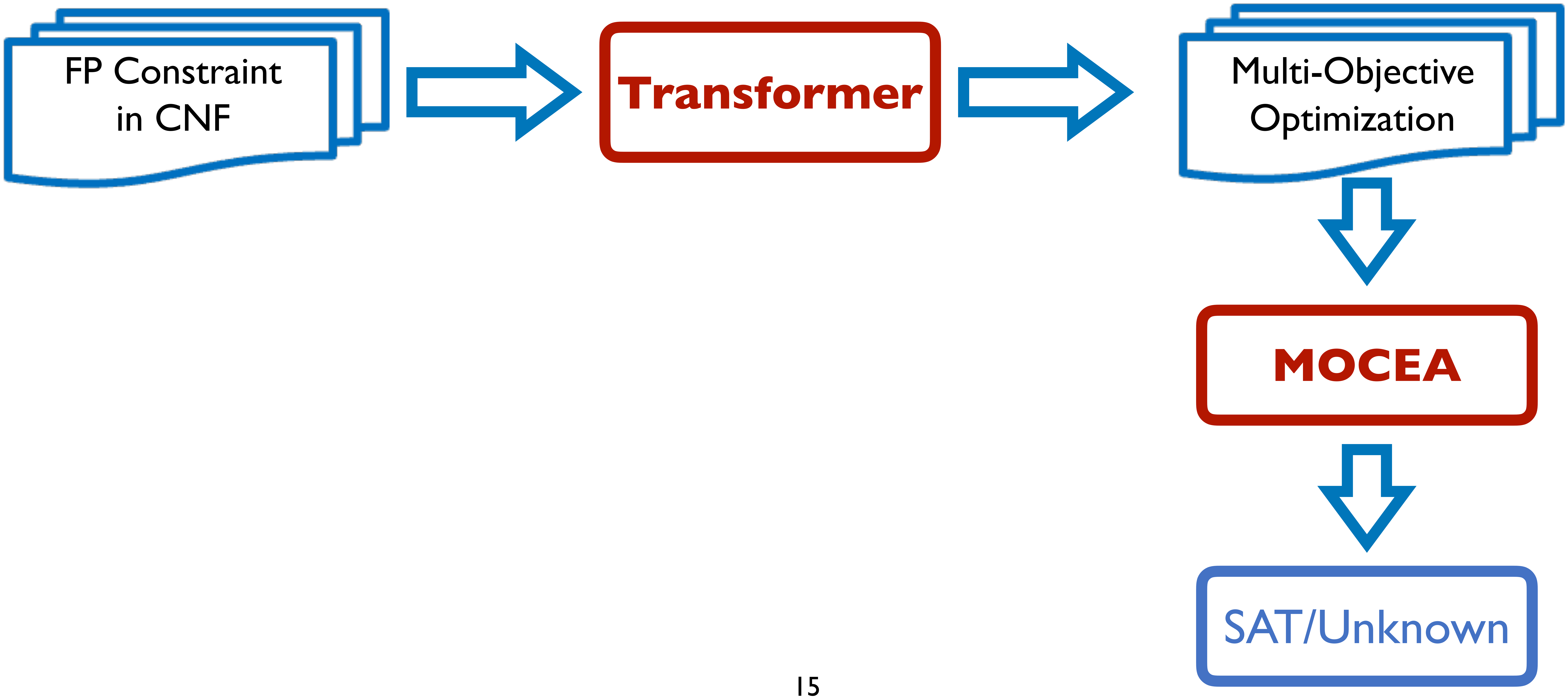
$$[x \mapsto 0, y \mapsto 64]$$

Exponent Mutation ⇩ **Single step**

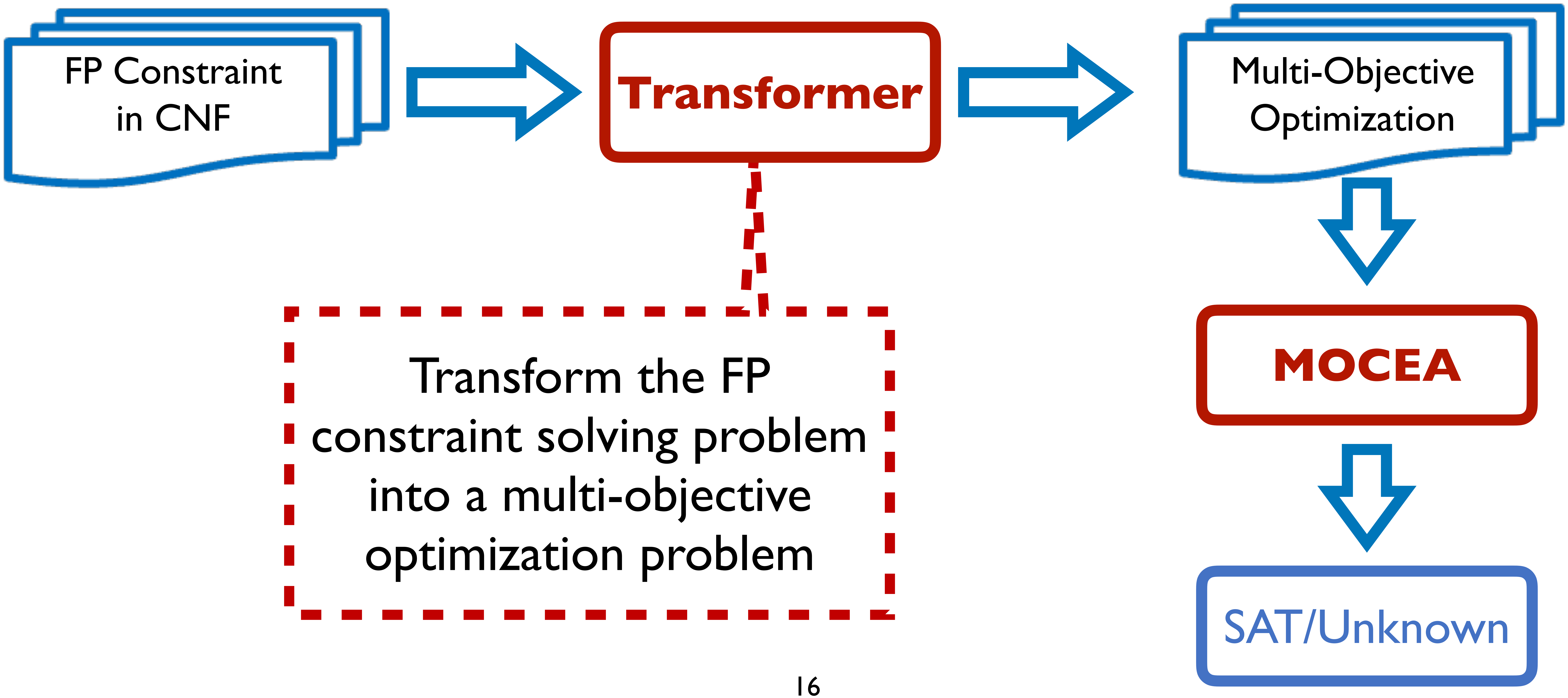
$$[x \mapsto 128, y \mapsto 64]$$



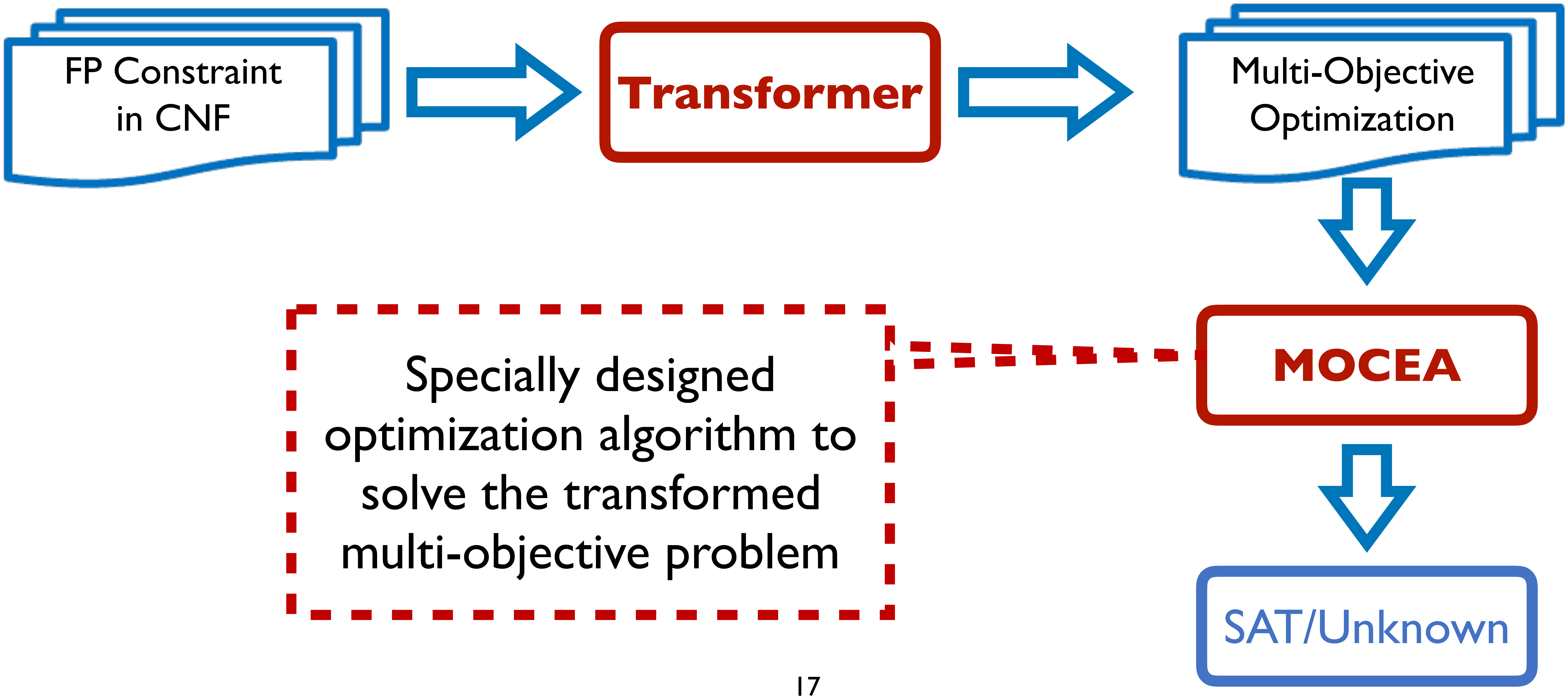
Framework


FP Constraint in CNF
Transformer
Multi-Objective Optimization
MOCEA
SAT/Unknown

Framework


FP Constraint in CNF
Transformer
Multi-Objective Optimization
Transform the FP constraint solving problem into a multi-objective optimization problem
MOCEA
SAT/Unknown

Framework


FP Constraint in CNF
Transformer
Multi-Objective Optimization
MOCEA
SAT/Unknown
Specially designed optimization algorithm to solve the transformed multi-objective problem

Transformation Details

CNF $\psi := \bigwedge_{i \in I} \bigvee_{j \in J_i} e_{i,j} \bowtie_{i,j} e'_{i,j}.$

The number of constraints that are not satisfied under α

$$\begin{cases} \min & F^{\psi}(\alpha) := \left(\frac{1}{n} f_1^{\psi}(\alpha), \frac{1}{n} f_2^{\psi}(\alpha)\right) \\ \text{s.t.} & \alpha \in \mathbb{S} \end{cases}$$

$$f_1^{\psi}(\alpha) := \sum_{i \in I} \prod_{j \in J} \mathbb{I}[\neg \alpha(e_{i,j} \bowtie_{i,j} e'_{i,j})]$$

$$f_2^{\psi}(\alpha) := \sum_{i \in I} \min_{j \in J} \delta(e_{i,j} \bowtie_{i,j} e'_{i,j}, \alpha)$$

Multi-objective optimization problem

Sum of the violations of the constraints under α

Transformation Details

$$\delta(e_1 \bowtie e_2, \alpha) := \begin{cases} \mathbb{I}[\neg\alpha(e_1 \bowtie e_2)] & \bowtie \in \{\neq\} \\ \frac{H(b_{\alpha(e_1)}, b_{\alpha(e_2)})}{m} & \bowtie \in \{=\} \\ \frac{|int(\alpha(e_1)) - int(\alpha(e_2))| + \kappa}{2^m - 1} & \text{otherwise} \end{cases}$$

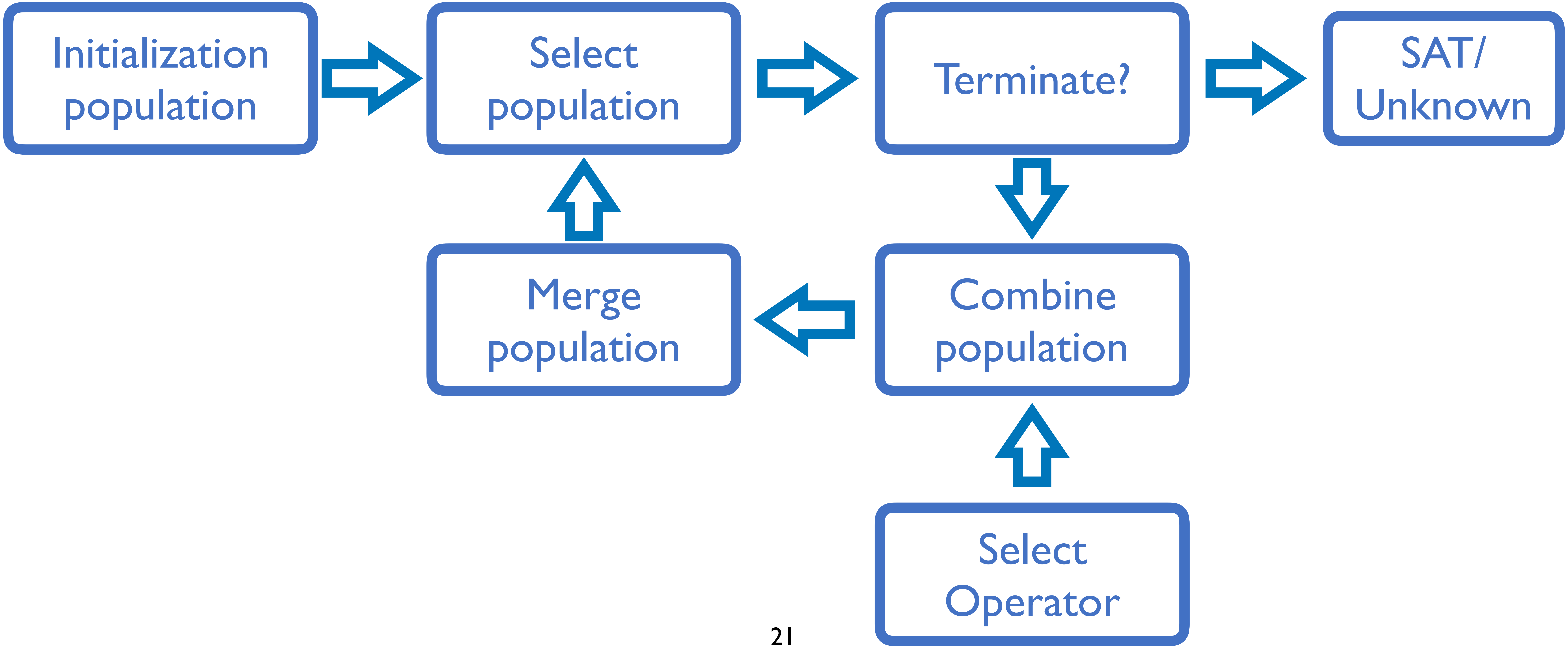
H is hamming distance

Transformation Details

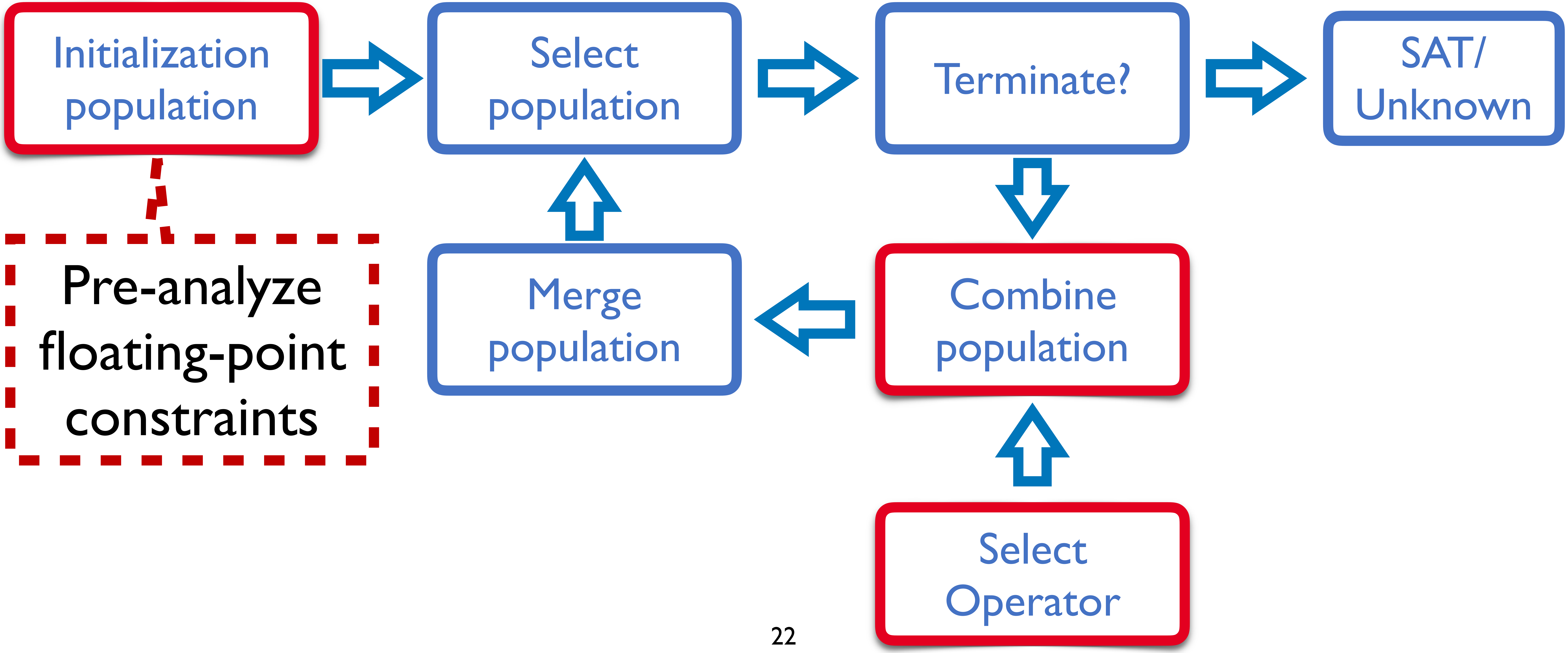
$$\delta(e_1 \bowtie e_2, \alpha) := \begin{cases} \mathbb{I}[\neg\alpha(e_1 \bowtie e_2)] & \bowtie \in \{\neq\} \\ \frac{H(b_{\alpha(e_1)}, b_{\alpha(e_2)})}{m} & \bowtie \in \{=\} \\ \boxed{\frac{|int(\alpha(e_1)) - int(\alpha(e_2))| + \kappa}{2^m - 1} \quad \text{otherwise}} \end{cases}$$

Unsigned integer to avoid floating-point underflow

Optimization Algorithm



Optimization Algorithm



Optimization Algorithm

Optimization Algorithm

Evaluation

- Implementation based on goSAT
  - Z3 as FP constraint's frontend
  - Object function's LLVM IR is generated for JIT
  - MOCEA implementation based on NLopt

Evaluation

- Research Questions
  - Q1: How effective and efficient is QSF in SMT-LIB benchmarks?
  - Q2: How effective and efficient is QSF in analyzing real floating-point programs?
  - Q3: How do different components affect the overall performance of QSF?

Evaluation

- Benchmarks and baselines
  - QF_FP SMTLIB
    - 266 SAT or UNKNOWN
  - Real-world program
    - 3493 SAT or UNKNOWN
  - Timeout: 60s and 600s

Solvers	Version	Technique	Category
Z3 [21]	v4.6.0	Bit-blasting	QF_FP SMT theory
CVC5 [4]	v1.1.2	Bit-blasting	QF_FP SMT theory
MathSAT5 [18]	v5.5.1	Bit-blasting	QF_FP SMT theory
Bitwuzla [48]	v1.0	Bit-blasting	QF_FP SMT theory
COLIBRI [45]	r15172	Interval solving	Real arithmetic SMT
JFS [43]	r5ceecd1	Coverage-guided fuzzing	Search-based
CORAL [57]	v0.7	Meta-heuristic search	Search-based
XSat [29]	2017	Mathematical optimisation	Search-based
goSAT [9]	rb5a423c	Mathematical optimisation	Search-based

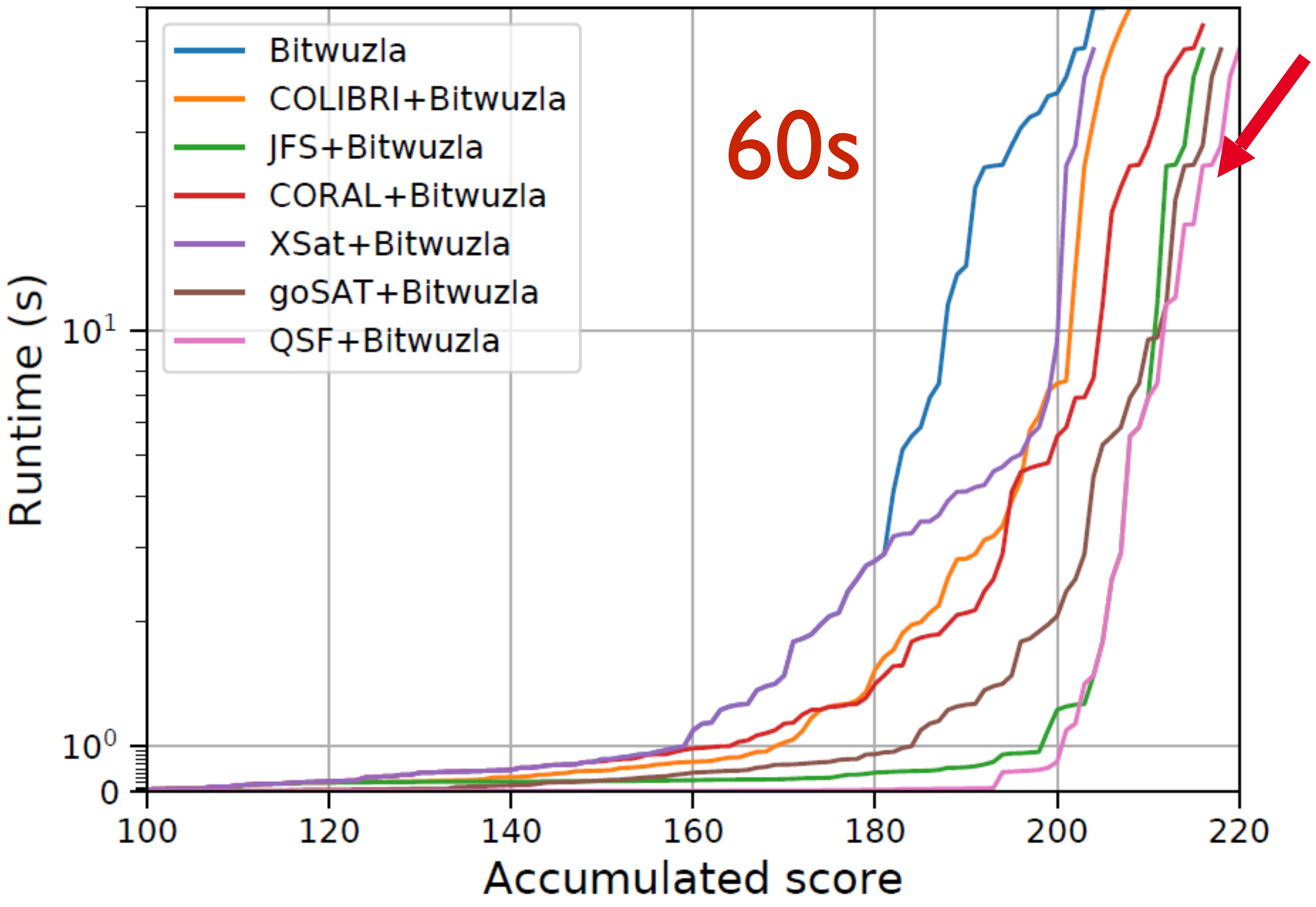
Symbolic Execution of Floating-point Programs: How far are we? *Xu Yang, Guofeng Zhang, Ziqi Shuai, Zhenbang Chen, Ji Wang. Journal of Systems and Software. 2024.*

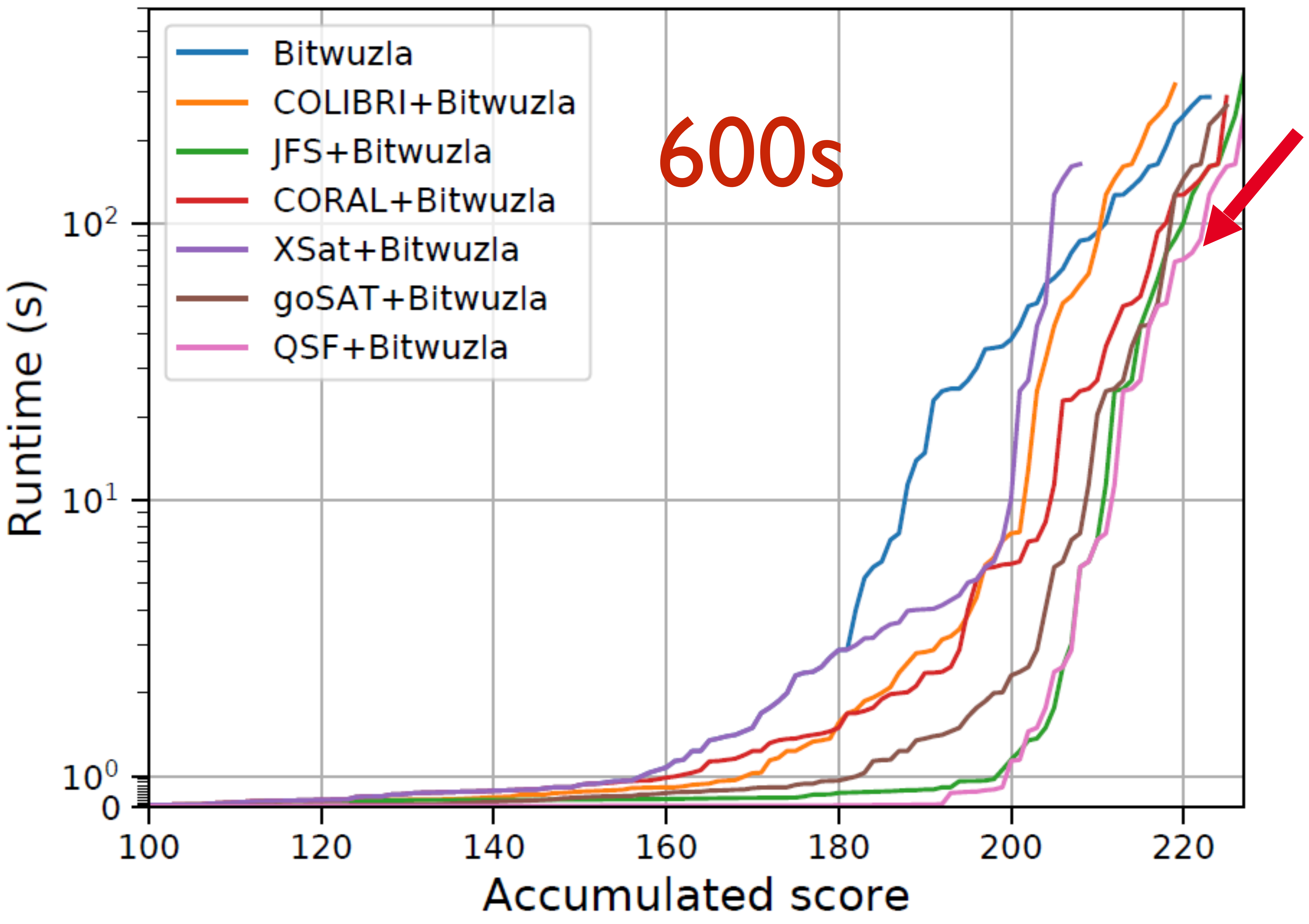
Evaluation: Effectiveness (QF_FP benchmark)

Solvers	Timeout (s)	Both	Only QSF	Only other	Neither
Z3	60	156 (58.65%)	**44 (16.54%)**	10 (3.76%)	56 (21.05%)
	600	180 (67.67%)	**20 (7.52%)**	10 (3.76%)	56 (21.05%)
CVC5	60	175 (65.79%)	**25 (9.40%)**	23 (8.65%)	43 (16.17%)
	600	196 (73.68%)	4 (1.50%)	**27 (10.15%)**	39 (14.66%)
MathSAT5	60	172 (64.66%)	**28 (10.53%)**	17 (6.39%)	49 (18.42%)
	600	198 (74.44%)	2 (0.75%)	**20 (7.52%)**	46 (17.29%)
Bitwuzla	60	184 (69.17%)	16 (6.02%)	**21 (7.89%)**	45 (16.92%)
	600	196 (73.68%)	4 (1.50%)	**27 (10.15%)**	39 (14.66%)
COLIBRI	60	177 (66.54%)	**23 (8.65%)**	10 (3.76%)	56 (21.05%)
	600	179 (67.29%)	**21 (7.89%)**	12 (4.51%)	54 (20.30%)
JFS	60	176 (66.17%)	**24 (9.02%)**	4 (1.50%)	62 (23.31%)
	600	180 (67.67%)	**20 (7.52%)**	6 (2.26%)	60 (22.56%)
CORAL	60	58 (21.80%)	**142 (53.38%)**	2 (0.75%)	64 (24.06%)
	600	62 (23.31%)	**138 (51.88%)**	1 (0.38%)	65 (24.44%)
XSat	60	110 (41.35%)	**90 (33.83%)**	8 (3.01%)	58 (21.80%)
	600	110 (41.35%)	**90 (33.83%)**	11 (4.14%)	55 (20.68%)
goSAT	60	141 (53.01%)	**59 (22.18%)**	3 (1.13%)	63 (23.68%)
	600	153 (57.52%)	**47 (17.67%)**	2 (0.75%)	64 (24.06%)

QSF is inferior to Bitwuzla, but is complementary to it

Evaluation: Effectiveness (QF_FP benchmark)





QSF+Bitwuzla has the best performance in combined solvers

Evaluation: Effectiveness (Real-world program benchmark)

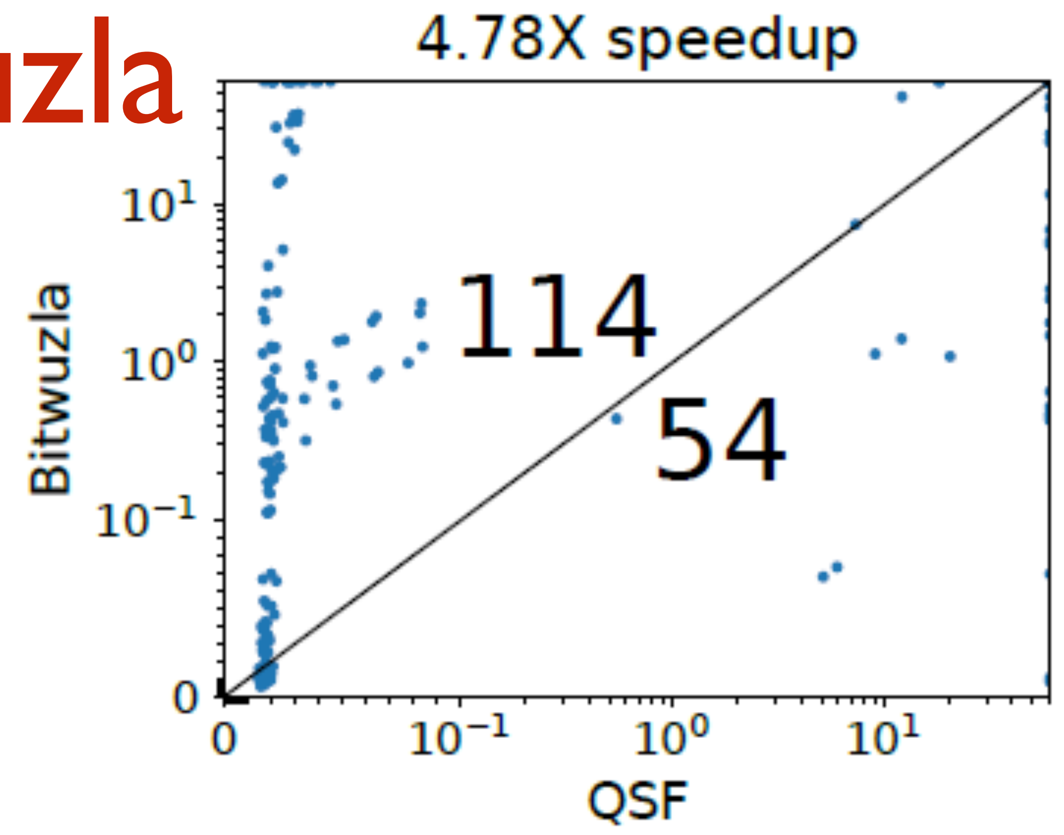
Solvers	Timeout (s)	Both	Only QSF	Only other	Neither
Z3	60	3029 (86.72%)	**182 (5.21%)**	15 (0.43%)	267 (7.64%)
	600	3104 (88.86%)	**114 (3.26%)**	15 (0.43%)	260 (7.44%)
CVC5	60	3092 (88.52%)	**119 (3.41%)**	14 (0.40%)	268 (7.67%)
	600	3167 (90.67%)	**51 (1.46%)**	21 (0.60%)	254 (7.27%)
MathSAT5	60	3069 (87.86%)	**142 (4.07%)**	13 (0.37%)	269 (7.70%)
	600	3161 (90.50%)	**57 (1.63%)**	23 (0.66%)	252 (7.21%)
Bitwuzla	60	3119 (89.29%)	**92 (2.63%)**	15 (0.43%)	267 (7.64%)
	600	3182 (91.10%)	**36 (1.03%)**	24 (0.69%)	251 (7.19%)
COLIBRI	60	2665 (76.30%)	**546 (15.63%)**	9 (0.26%)	273 (7.82%)
	600	2677 (76.64%)	**541 (15.49%)**	3 (0.09%)	272 (7.79%)
JFS	60	3014 (86.29%)	**197 (5.64%)**	31 (0.89%)	251 (7.19%)
	600	3101 (88.78%)	**117 (3.35%)**	35 (1.00%)	240 (6.87%)
goSAT	60	1963 (56.20%)	**1248 (35.73%)**	1 (0.03%)	281 (8.04%)
	600	2186 (62.58%)	**1032 (29.54%)**	1 (0.03%)	274 (7.84%)

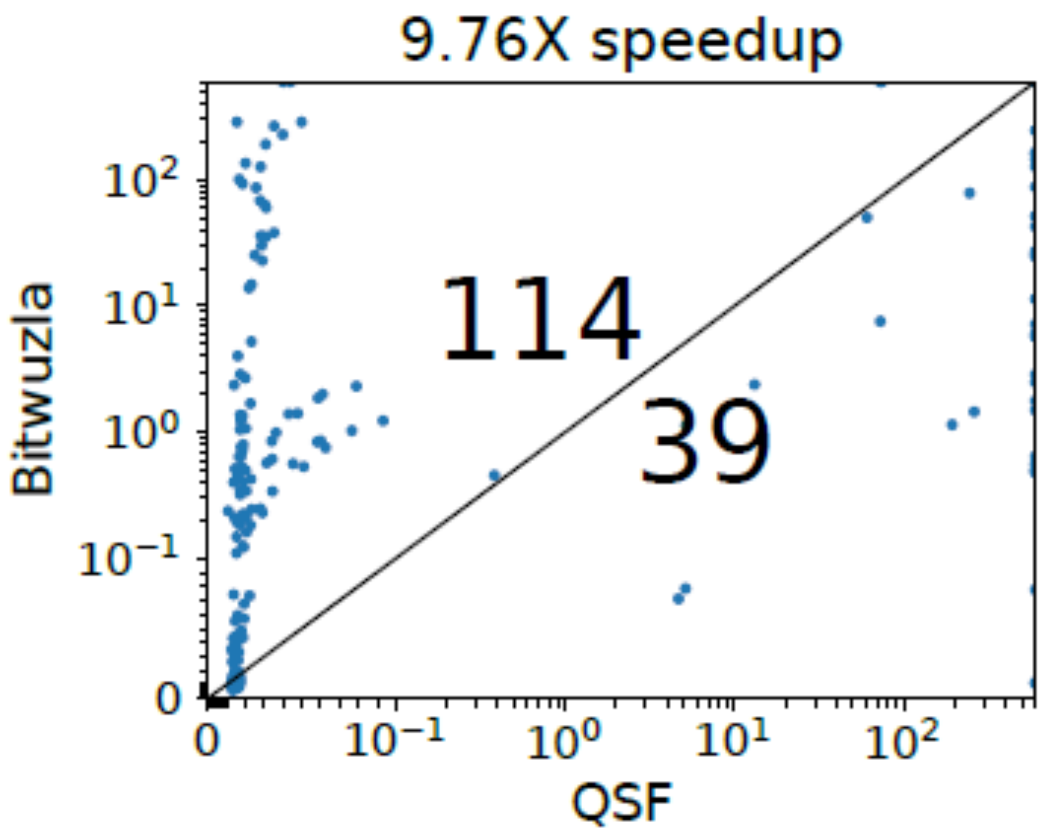
QSF outperforms all compared methods

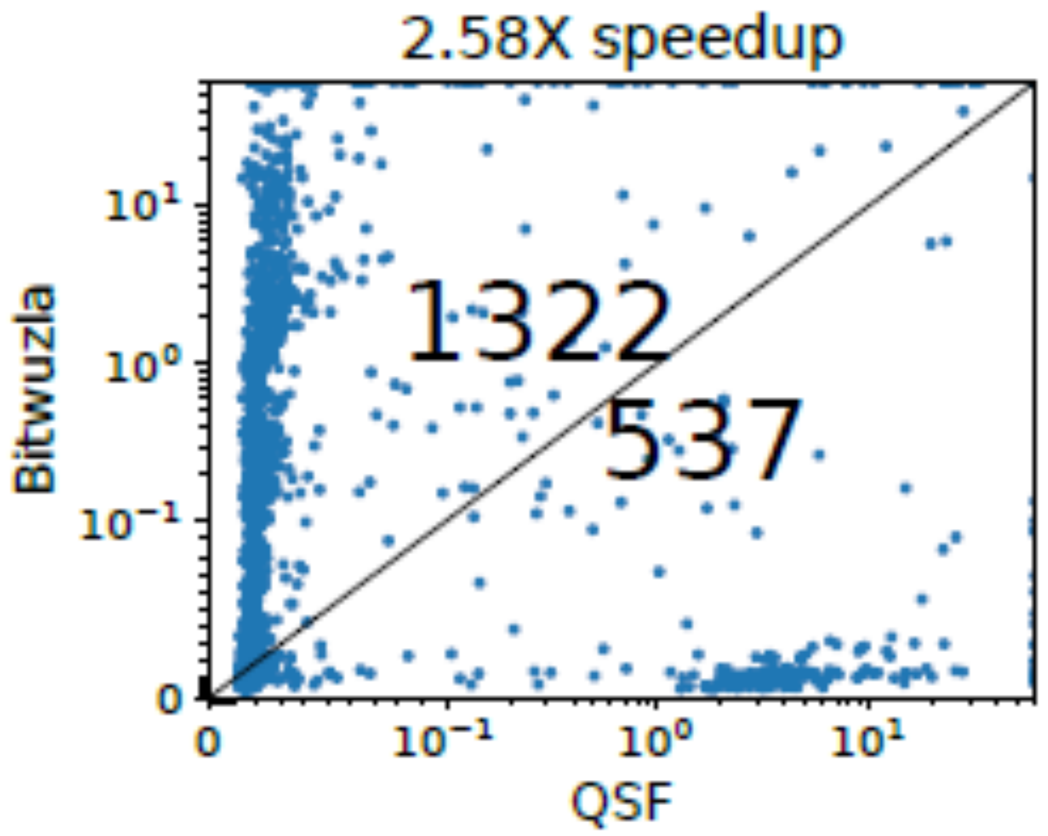
Evaluation: Efficiency (QF_FP & Real-world program)

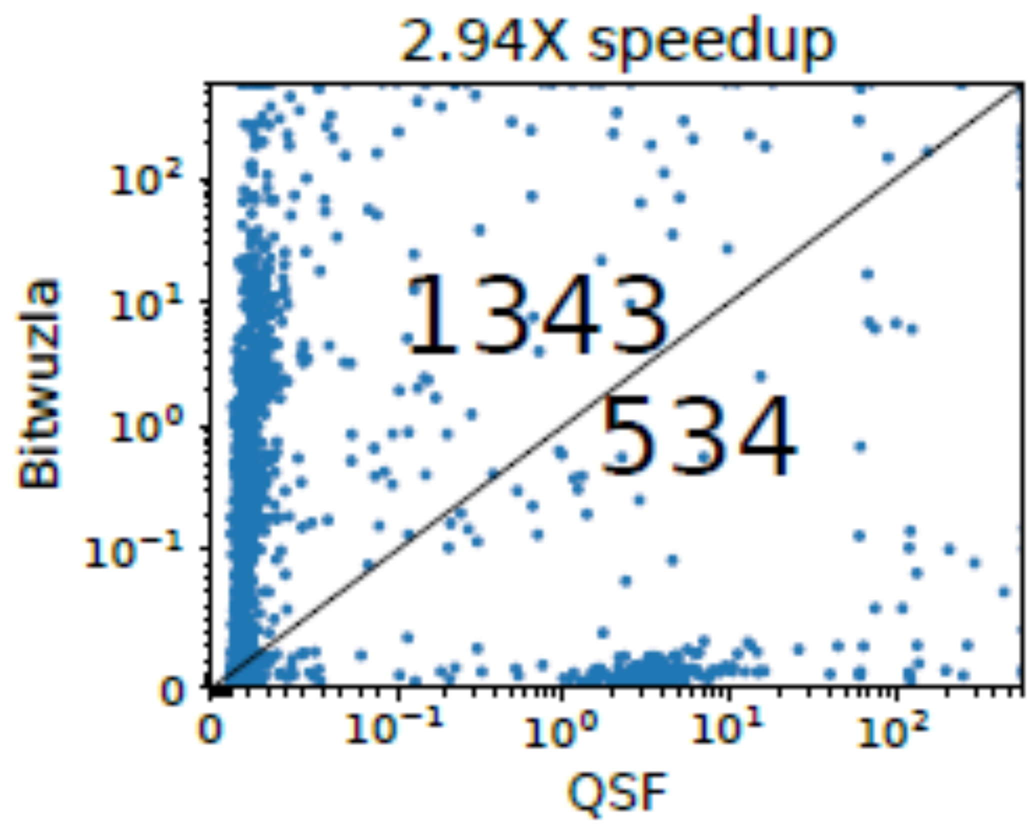
60s | 600s | 60s | 600s

Bitwuzla

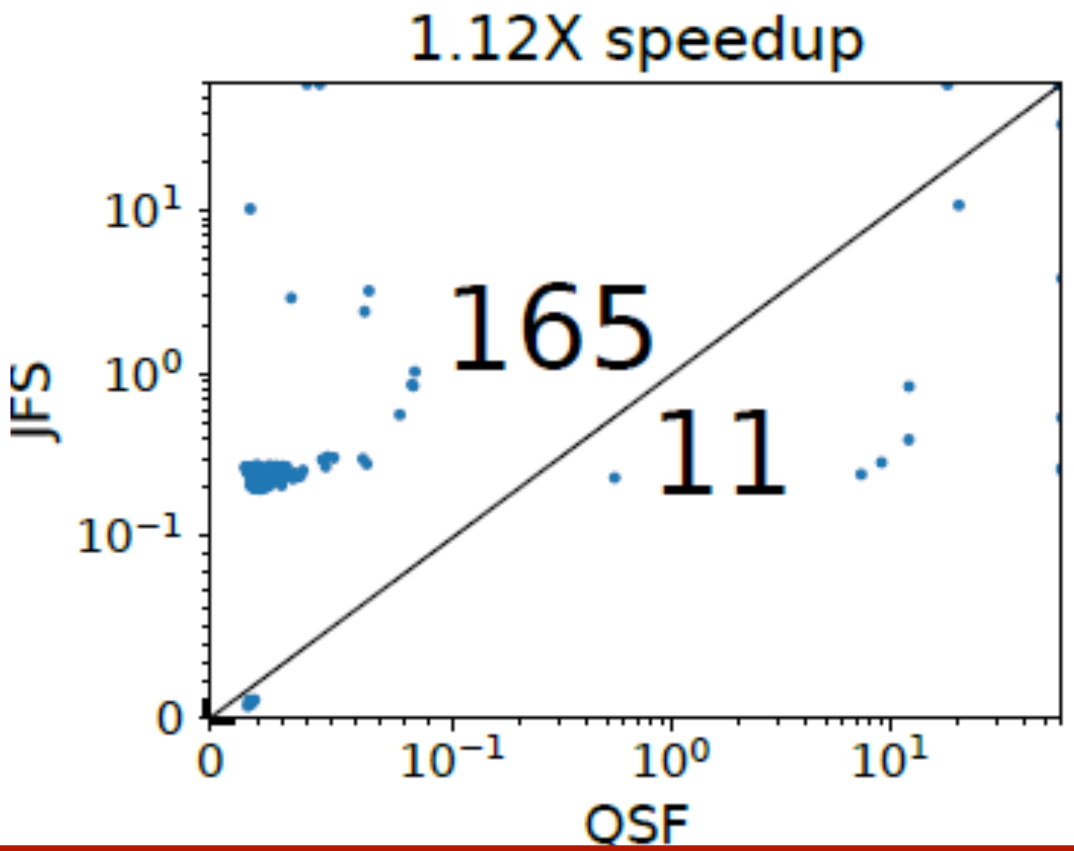


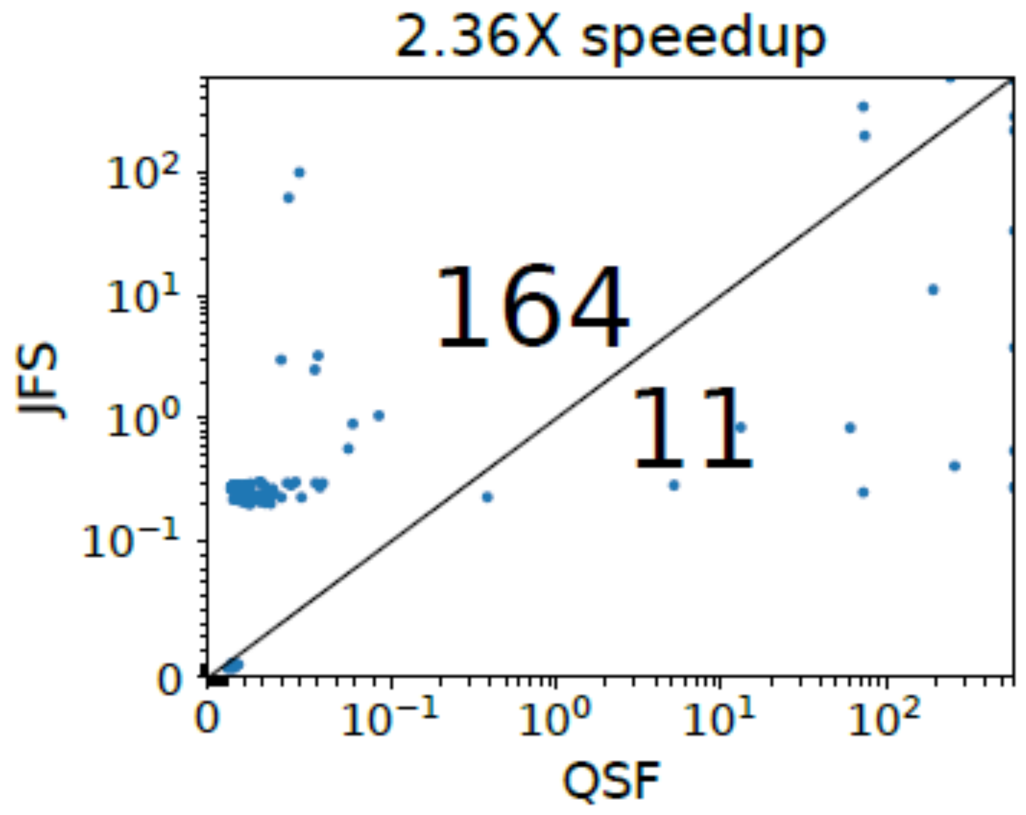


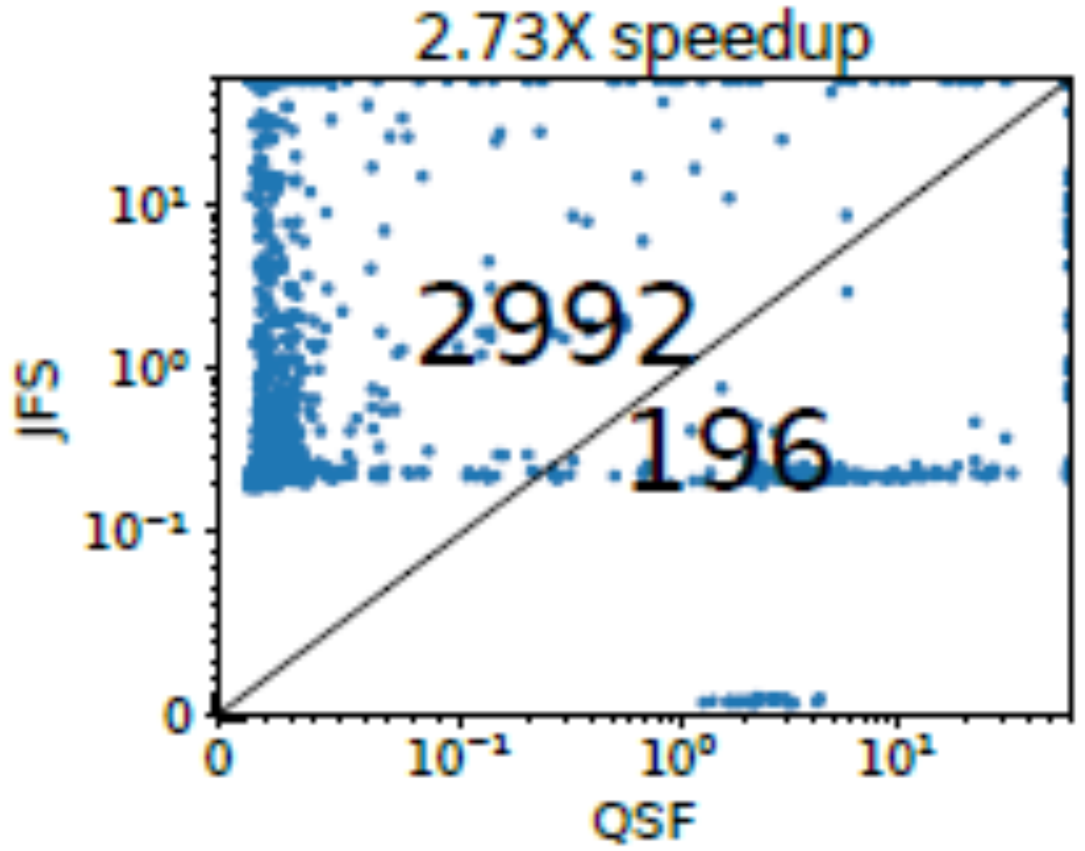


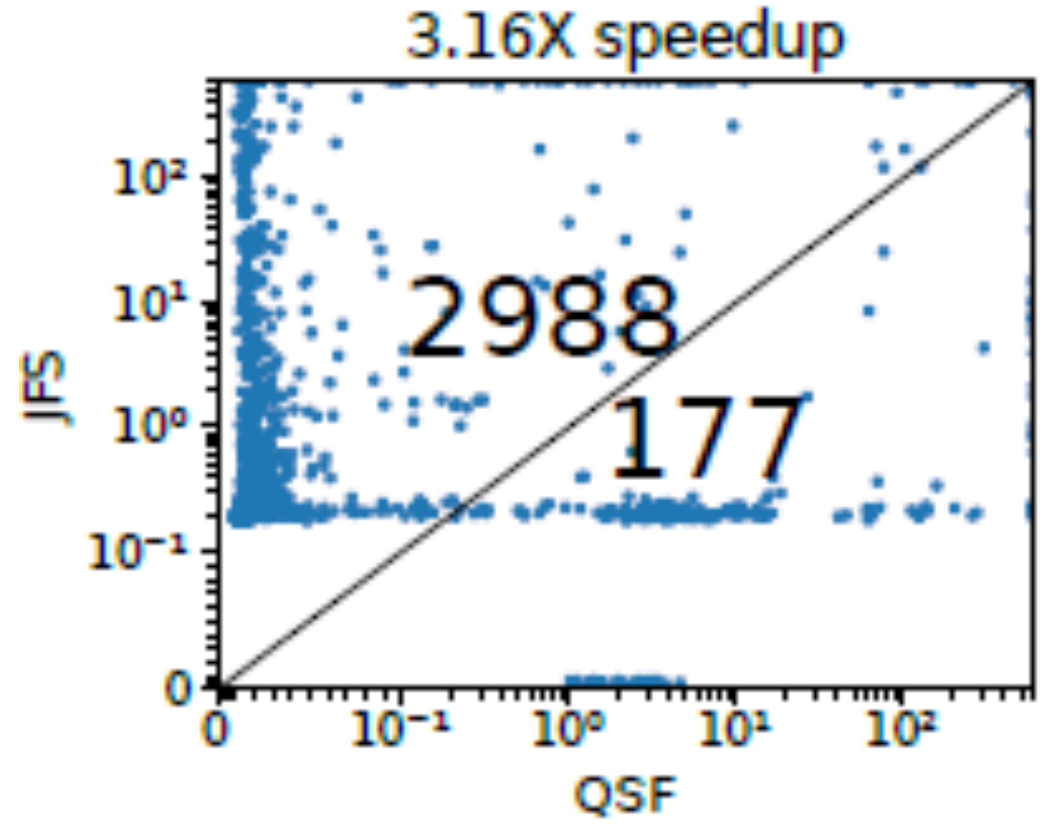


JFS



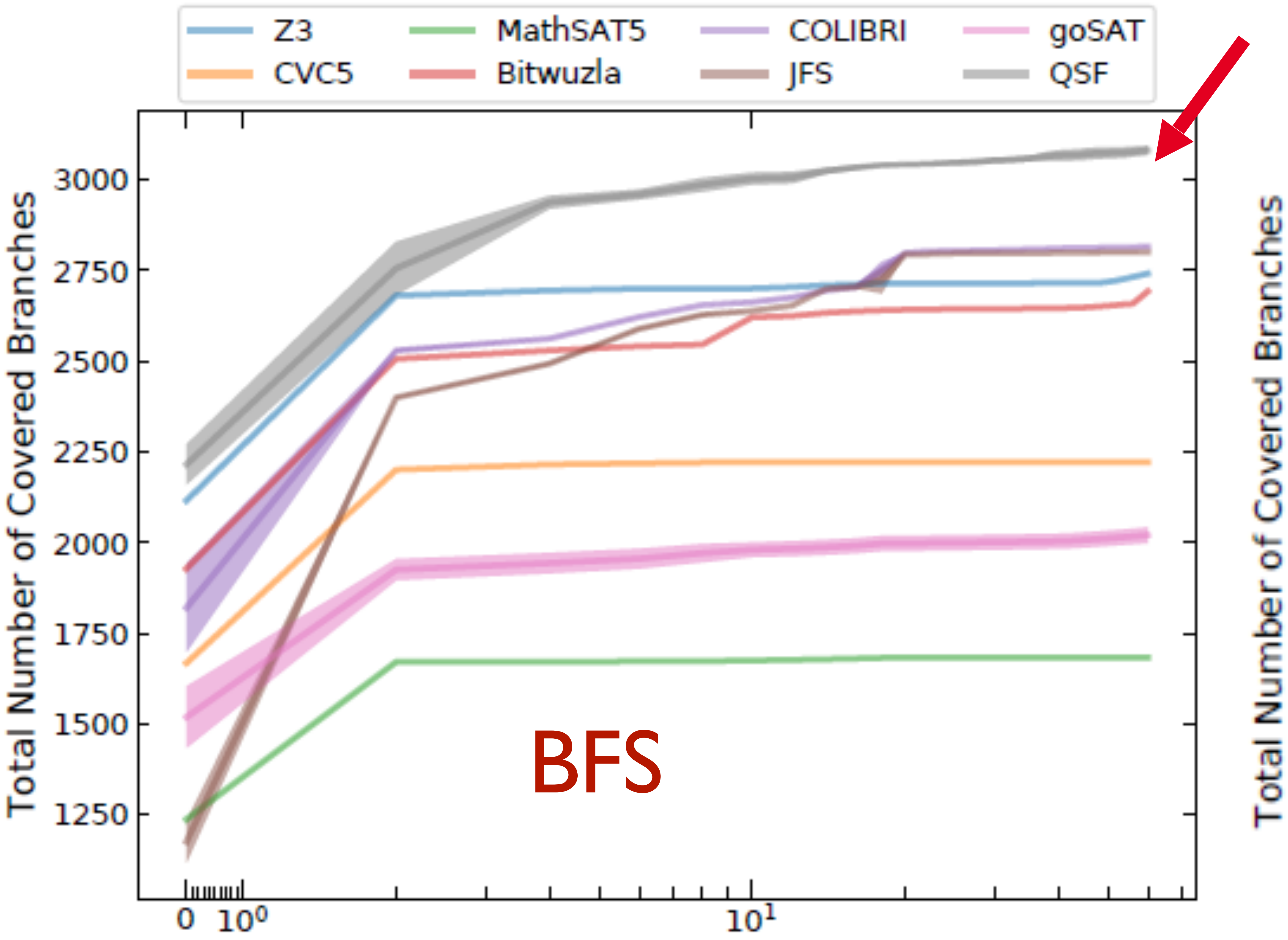


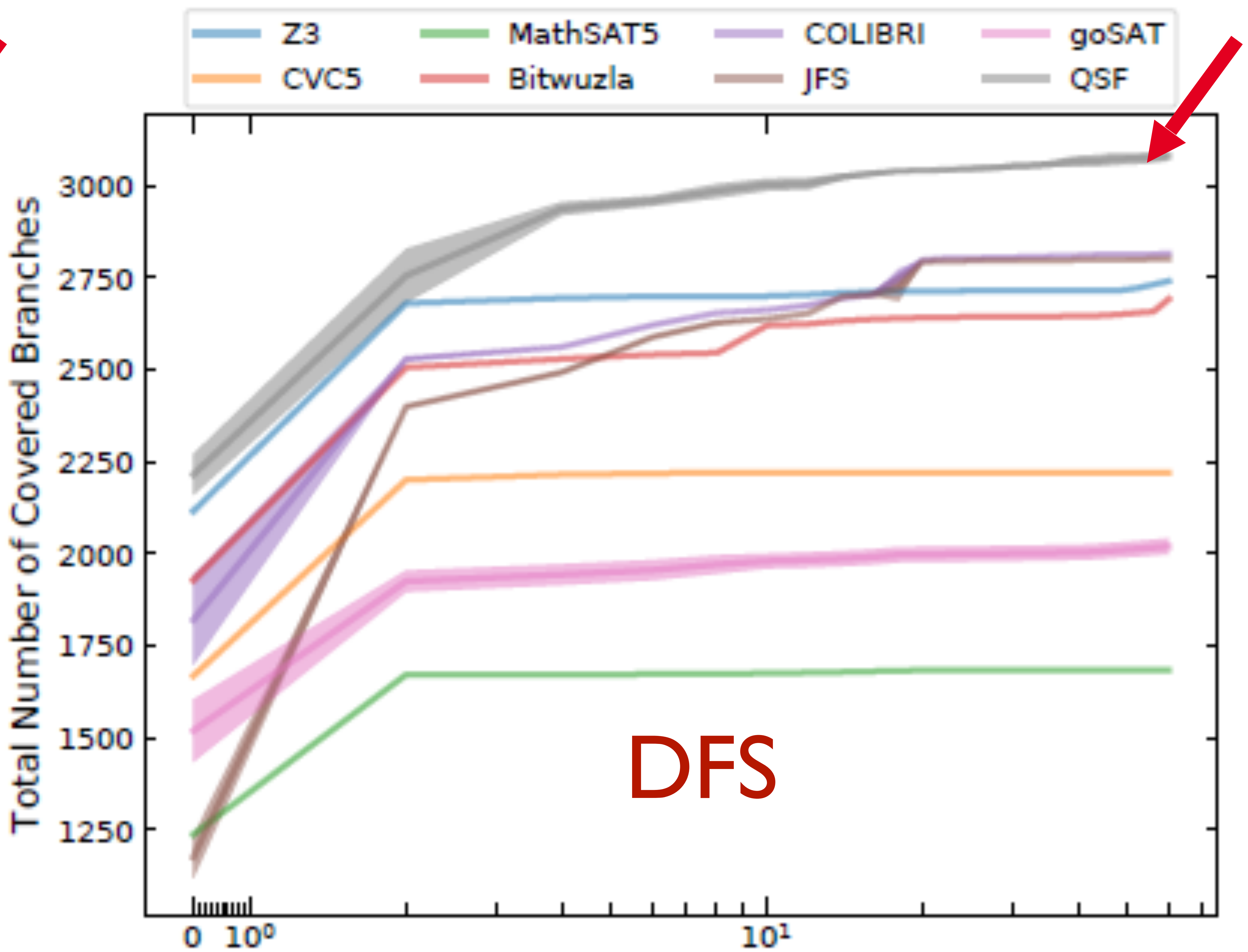




QSF is more efficient than Bitwuzla&JFS on both benchmarks

Evaluation: Application to Symbolic Execution





The number of covered branches obtained by QSF is better than that of other methods

Evaluation: Ablation studies

Benchmarks	Solvers	Timeout (s)	Both	Only QSF	Only other	Neither
QF_FP	QSF_f_1^ψ	60	189 (71.05%)	**11 (4.14%)**	0 (0.00%)	66 (24.81%)
		600	189 (71.05%)	**11 (4.14%)**	0 (0.00%)	66 (24.81%)
	QSF_f_2^ψ	60	193 (72.56%)	**7 (2.63%)**	0 (0.00%)	66 (24.81%)
		600	194 (72.93%)	**6 (2.26%)**	0 (0.00%)	66 (24.81%)
	QSF_NSGA-II	60	182 (68.42%)	**18 (6.77%)**	0 (0.00%)	66 (24.81%)
		600	183 (68.80%)	**17 (6.39%)**	0 (0.00%)	66 (24.81%)
	QSF_NoPre	60	174 (65.41%)	**26 (9.77%)**	0 (0.00%)	66 (24.81%)
		600	175 (65.79%)	**25 (9.40%)**	0 (0.00%)	66 (24.81%)
RWP	QSF_f_1^ψ	60	2962 (84.80%)	**249 (7.13%)**	6 (0.17%)	276 (7.90%)
		600	2989 (85.57%)	**229 (6.56%)**	2 (0.06%)	273 (7.82%)
	QSF_f_2^ψ	60	3187 (91.24%)	**24 (0.69%)**	6 (0.17%)	276 (7.90%)
		600	3192 (91.38%)	**26 (0.74%)**	2 (0.06%)	273 (7.82%)
	QSF_NSGA-II	60	2620 (75.01%)	**591 (16.92%)**	1 (0.03%)	281 (8.04%)
		600	2617 (74.92%)	**601 (17.21%)**	1 (0.03%)	274 (7.84%)
	QSF_NoPre	60	2891 (82.77%)	**320 (9.16%)**	2 (0.06%)	280 (8.02%)
		600	2891 (82.77%)	**327 (9.36%)**	2 (0.06%)	273 (7.82%)

Bi-objective guidance is better than single objective guidance

Evaluation: Ablation studies

Benchmarks	Solvers	Timeout (s)	Both	Only QSF	Only other	Neither
QF_FP	QSF_f_1^ψ	60	189 (71.05%)	**11 (4.14%)**	0 (0.00%)	66 (24.81%)
		600	189 (71.05%)	**11 (4.14%)**	0 (0.00%)	66 (24.81%)
	QSF_f_2^ψ	60	193 (72.56%)	**7 (2.63%)**	0 (0.00%)	66 (24.81%)
		600	194 (72.93%)	**6 (2.26%)**	0 (0.00%)	66 (24.81%)
	QSF_NSGA-II	60	182 (68.42%)	**18 (6.77%)**	0 (0.00%)	66 (24.81%)
		600	183 (68.80%)	**17 (6.39%)**	0 (0.00%)	66 (24.81%)
	QSF_NoPre	60	174 (65.41%)	**26 (9.77%)**	0 (0.00%)	66 (24.81%)
		600	175 (65.79%)	**25 (9.40%)**	0 (0.00%)	66 (24.81%)
RWP	QSF_f_1^ψ	60	2962 (84.80%)	**249 (7.13%)**	6 (0.17%)	276 (7.90%)
		600	2989 (85.57%)	**229 (6.56%)**	2 (0.06%)	273 (7.82%)
	QSF_f_2^ψ	60	3187 (91.24%)	**24 (0.69%)**	6 (0.17%)	276 (7.90%)
		600	3192 (91.38%)	**26 (0.74%)**	2 (0.06%)	273 (7.82%)
	QSF_NSGA-II	60	2620 (75.01%)	**591 (16.92%)**	1 (0.03%)	281 (8.04%)
		600	2617 (74.92%)	**601 (17.21%)**	1 (0.03%)	274 (7.84%)
		60	2891 (82.77%)	**320 (9.16%)**	2 (0.06%)	280 (8.02%)
			(82.77%)	**327 (9.36%)**	2 (0.06%)	273 (7.82%)

MOCEA is better than the classic evolutionary algorithm

Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE transactions on evolutionary computation, 2002, 6(2): 182-197.

Evaluation: Ablation studies

Benchmarks	Solvers	Timeout (s)	Both	Only QSF	Only other	Neither
QF_FP	QSF_f_1^{ψ}	60	189 (71.05%)	**11 (4.14%)**	0 (0.00%)	66 (24.81%)
		600	189 (71.05%)	**11 (4.14%)**	0 (0.00%)	66 (24.81%)
	QSF_f_2^{ψ}	60	193 (72.56%)	**7 (2.63%)**	0 (0.00%)	66 (24.81%)
		600	194 (72.93%)	**6 (2.26%)**	0 (0.00%)	66 (24.81%)
	QSF_NSGA-II	60	182 (68.42%)	**18 (6.77%)**	0 (0.00%)	66 (24.81%)
		600	183 (68.80%)	**17 (6.39%)**	0 (0.00%)	66 (24.81%)
	QSF_NoPre	60	174 (65.41%)	**26 (9.77%)**	0 (0.00%)	66 (24.81%)
		600	175 (65.79%)	**25 (9.40%)**	0 (0.00%)	66 (24.81%)
RWP	QSF_f_1^{ψ}	60	2962 (84.80%)	**249 (7.13%)**	6 (0.17%)	276 (7.90%)
		600	2989 (85.57%)	**229 (6.56%)**	2 (0.06%)	273 (7.82%)
	QSF_f_2^{ψ}	60	3187 (91.24%)	**24 (0.69%)**	6 (0.17%)	276 (7.90%)
		600	3192 (91.38%)	**26 (0.74%)**	2 (0.06%)	273 (7.82%)
	QSF_NSGA-II	60	2620 (75.01%)	**591 (16.92%)**	1 (0.03%)	281 (8.04%)
		600	2617 (74.92%)	**601 (17.21%)**	1 (0.03%)	274 (7.84%)
	QSF_NoPre	60	2891 (82.77%)	**320 (9.16%)**	2 (0.06%)	280 (8.02%)
		600	2891 (82.77%)	**327 (9.36%)**	2 (0.06%)	273 (7.82%)

The preprocessing is helpful to QSF

Conclusion

FP Constraint Solving is Challenging

- Precise encoding is expensive

$$a^3 + b^3 > 0$$

  - Z3: > 8s
  - CVC5: > 2s
  - MathSAT5: > 3s

12th Gen Intel(R) Core(TM) i9-12900H CPU@3.0GHz

- Real number encoding is unsound

$$(a + b) + c \neq a + (b + c)$$

SAT if *a*, *b*, *and c are FP numbers*

- Search-based method is incomplete

$$a - 1.0 = 1.1$$

UNSAT if *a* is a 32-bits FP number

7

Our Key Insight

- Only single object function is employed for searching
  - Fuzzing based: #unsatisfied atomic constraints
  - Optimization based: fitness (distance) function's result

$$x \geq 64 \wedge y = 64$$

$[x \mapsto 0, y \mapsto 64]$	$[x \mapsto 63, y \mapsto 63]$
$f_1 = 1$	$f_1 = 2$
$f_2 = 64$	$f_2 = 2$

Consider both f_1 and f_2 in the search procedure

12

Our Key Insight

- Mutation operators in optimization can be customized for FPs

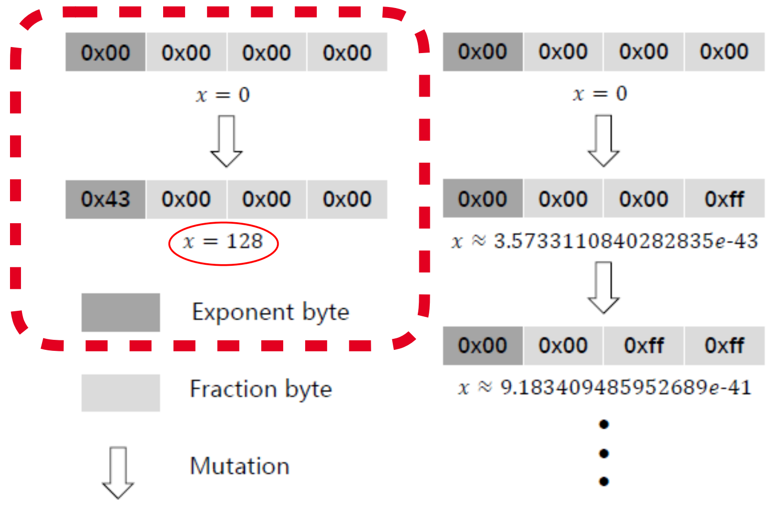
$$x \geq 64 \wedge y = 64$$

$$[x \mapsto 0, y \mapsto 64]$$

Exponent Mutation

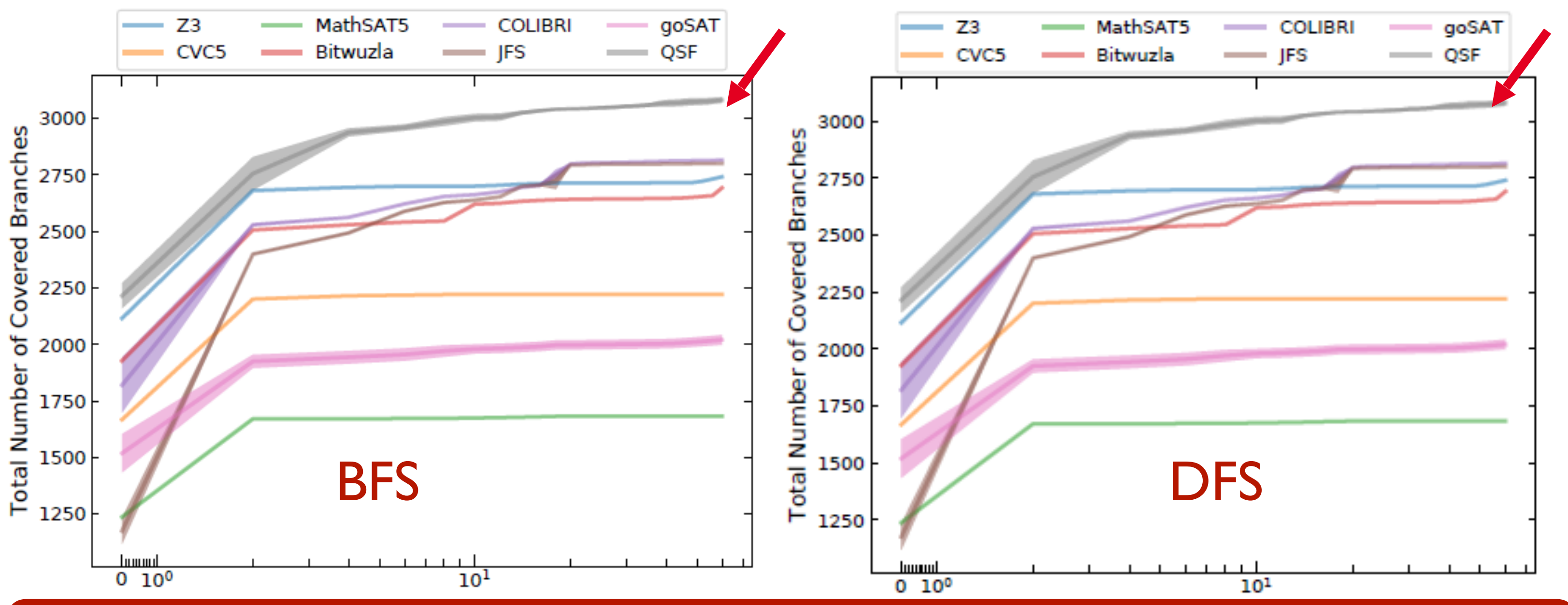
Single step

$$[x \mapsto 128, y \mapsto 64]$$



14

Evaluation: Application to Symbolic Execution



The number of covered branches obtained by QSF is better than that of other methods

32

Thank you!
Q&A

Artifact: https://github.com/zbchen/QSF