The 1st Workshop on Symmetric-key Cryptanalysis Automation and Modelling (SKCAM 2025)

Fast Software Encryption (FSE) 2025 conference

Hybrid and Parallel Algorithms for SAT and MILP

Shaowei Cai

Institute of Software, Chinese Academy of Sciences

2025.3.15

Outline

- Brief Introduction to SAT
- Hybrid SAT Algorithm
- Parallel SAT and MILP Algorithms

Constraint Solvers in Symmetric-key Crypt-analysis

SAT: Boolean Satisfiability

$(A \vee B) \wedge (\neg C \vee \neg B) \wedge (\neg C \vee A)$

MILP: (Mixed**)** Integer Linear Programming

Min z=30x+40y

S.t. 2x+3y≤100

x+2y≤150

x≥0 y≥0 x,y∈Z

Also: Pseudo Boolean Optimization (PBO)

SMT: Satisfiability Modulo Theories

(2^b = c) ∧ (A[3] ≠ A[c-b] ∨ s =10)

CP: Constraint Programming

$$AllDifferent(x_i, x_j) \wedge |x_i - x_j| \neq |i - j|$$

SAT

Definition [Boolean Satisfiability]
Given a Boolean formula φ, test whether there is an assignment to the variables that makes φ true.

- Boolean variables: $x_1, x_2, \ldots$
- A literal is a Boolean variable x (positive literal) or its negation $\neg x$ (negative literal)
- A clause is a disjunction ($\vee$) of literals

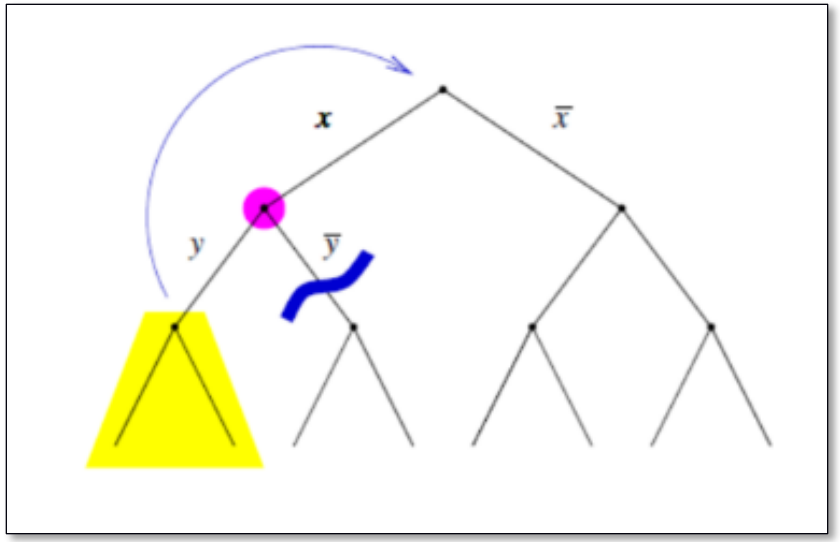
  $x_2 \vee x_3$,

  $\neg x_1 \vee \neg x_3 \vee x_4$

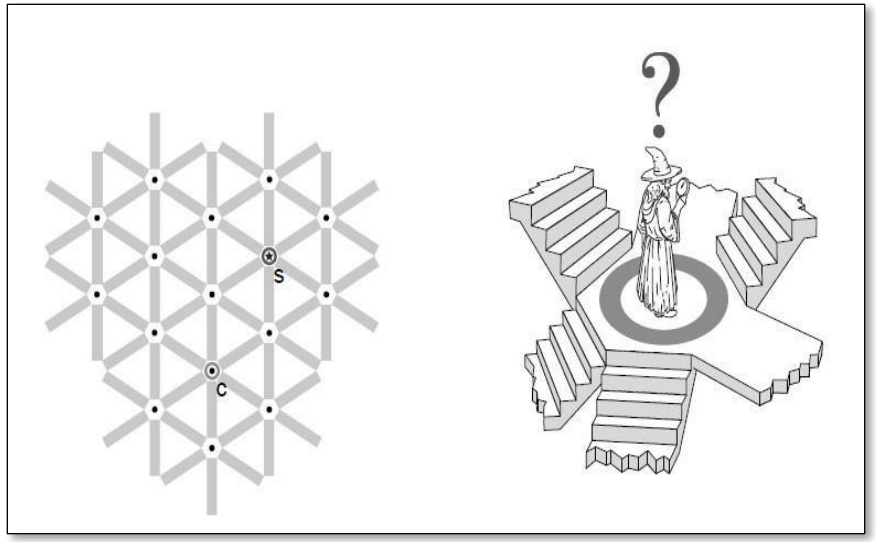
- A Conjunctive Normal Form (CNF) formula is a conjunction ($\wedge$) of clauses.

e.g., $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4)$

Two Methods for Solving SAT



conflict-driven clause learning (CDCL)



local search

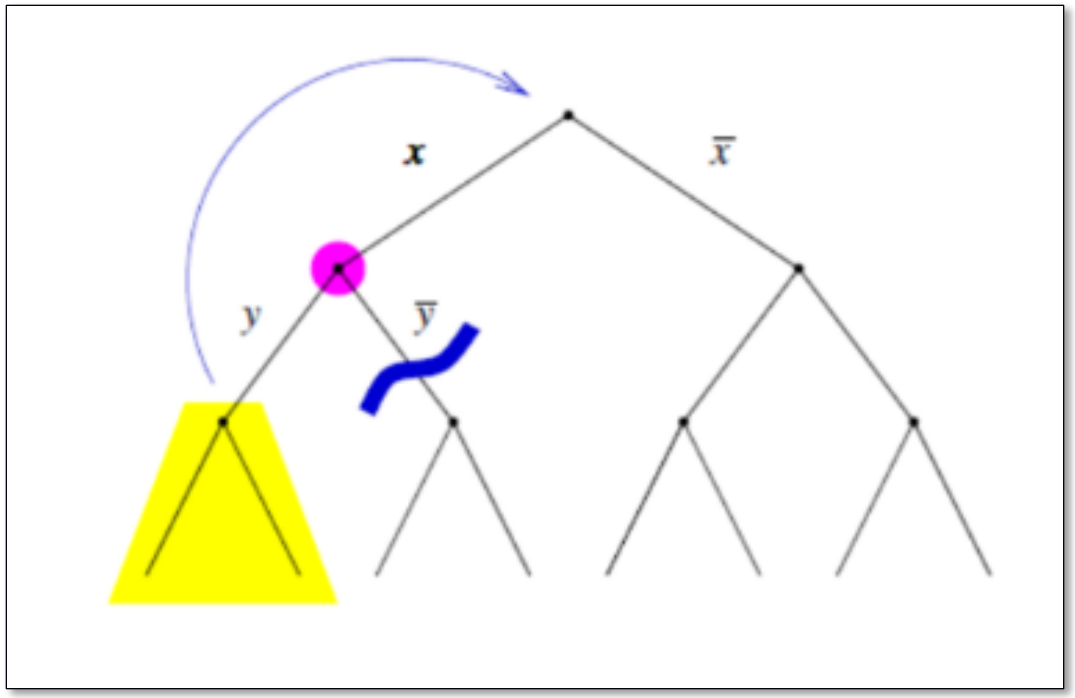
A Simple History of SAT Solvers

- 1960-1990
  - DP, DPLL(1962)
  - Resolution: Stålmarck's Method (1989)
- 1990-2010
  - Local Search (1992): GSAT (1992), WalkSAT (1994)
  - CDCL(1996): GRASP(1999), Chaff(2000), MiniSAT (2003), Glucose (2009), Cryptominisat (2009)
  - Portfolio: SATzilla (2007)
- 2010~today
  - Modern local search: probSAT(2012), CCAnr(2013)
  - Advanced clause management (2009,2015) and simplification
  - Solver engineering: CadiCal, Kissat(2019)
  - Hybridizing CDCL and local search (2020)
  - Efficient parallel solving

CDCL Algorithm Overview

CDCL solver

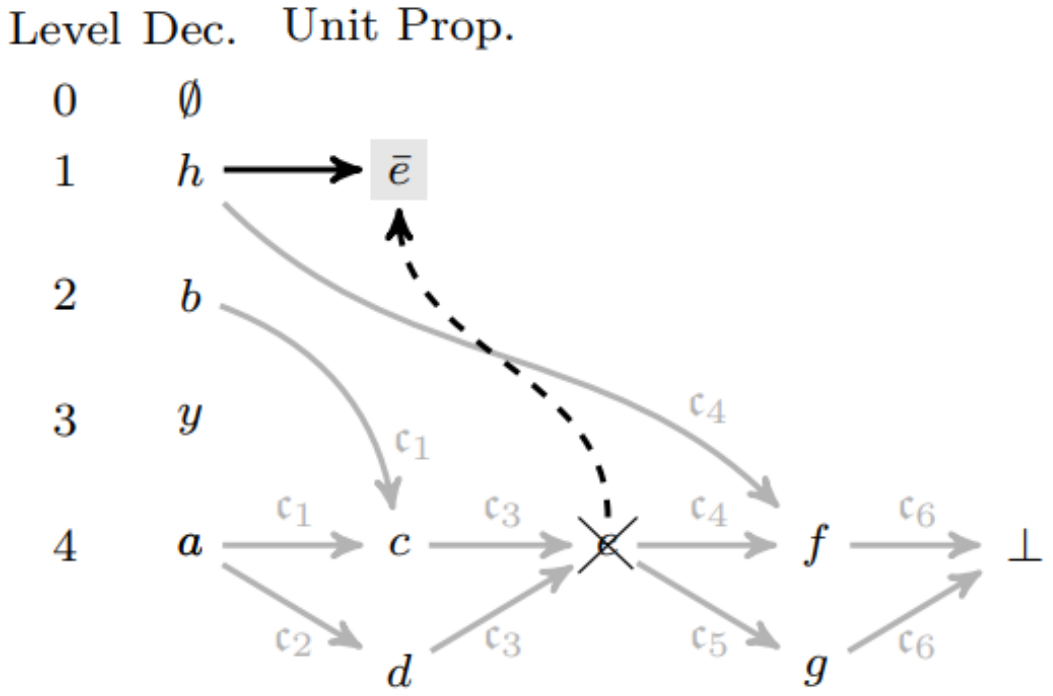
- Decide : Branching strategy and phasing strategy
- Analyze : (non-chronological) backtrack + clause learning



- Clause learning
- Clause management
- Lazy data structures
- Restarting
- Branching
- Phasing
- Mode Switching
- …

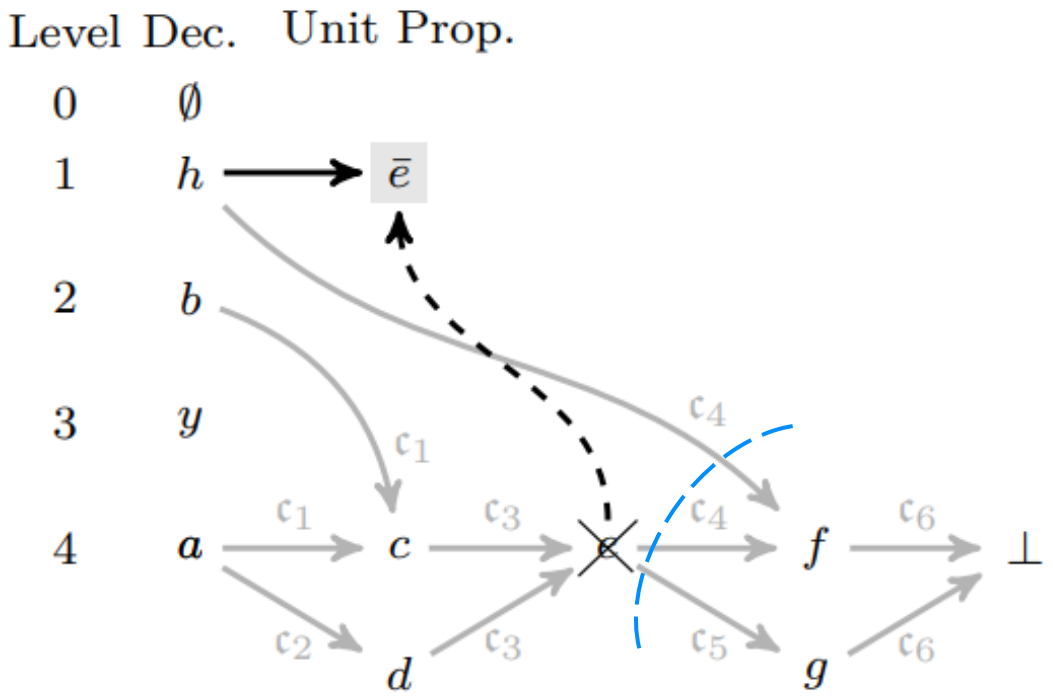
Conflict Driven Clause Learning

$$\mathcal{F}_2 = \mathfrak{c}_1 \wedge \mathfrak{c}_2 \wedge \mathfrak{c}_3 \wedge \mathfrak{c}_4 \wedge \mathfrak{c}_5 \wedge \mathfrak{c}_6$$
$$= (\bar{a} \vee \bar{b} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{c} \vee \bar{d} \vee e) \wedge (\bar{h} \vee \bar{e} \vee f) \wedge (\bar{e} \vee g) \wedge (\bar{f} \vee \bar{g})$$



Conflict Driven Clause Learning

$$\mathcal{F}_2 = \mathfrak{c}_1 \wedge \mathfrak{c}_2 \wedge \mathfrak{c}_3 \wedge \mathfrak{c}_4 \wedge \mathfrak{c}_5 \wedge \mathfrak{c}_6$$
$$= (\bar{a} \vee \bar{b} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{c} \vee \bar{d} \vee e) \wedge (\bar{h} \vee \bar{e} \vee f) \wedge (\bar{e} \vee g) \wedge (\bar{f} \vee \bar{g}) \quad \wedge (\overline{h} \vee \overline{e})$$



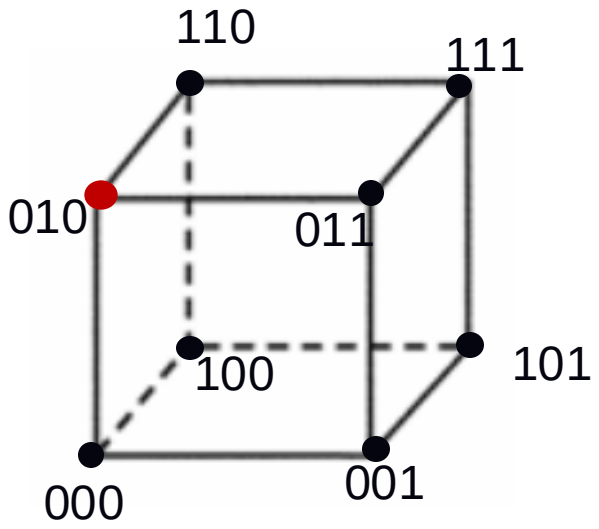
learn a clause $\overline{h} \vee \overline{e}$

Local Search

F={$\neg x_1 \vee \neg x_2,\ x_1 \vee x_2, \neg x_2 \vee \neg x_3, x_2 \vee x_3,\ \neg x_1 \vee x_2 \vee \neg x_3$}

Assignment (x_1, x_2, x_3)	Cost	falsified Clauses
0 0 0	2	$(x_1 \vee x_2), (x_2 \vee x_3)$
0 0 1	1	$(x_1 \vee x_2)$
0 1 0	0	None √
0 1 1	1	$(\neg x_2 \vee \neg x_3)$
1 0 0	1	$(x_2 \vee x_3)$
1 0 1	1	$(\neg x_1 \vee x_2 \vee \neg x_3)$
1 1 0	1	$(\neg x_1 \vee \neg x_2)$
1 1 1	2	$(\neg x_1 \vee \neg x_2), (\neg x_2 \vee \neg x_3)$

a CNF with 3 variables



Search space:
all complete assignments

Organized by neighboring relation

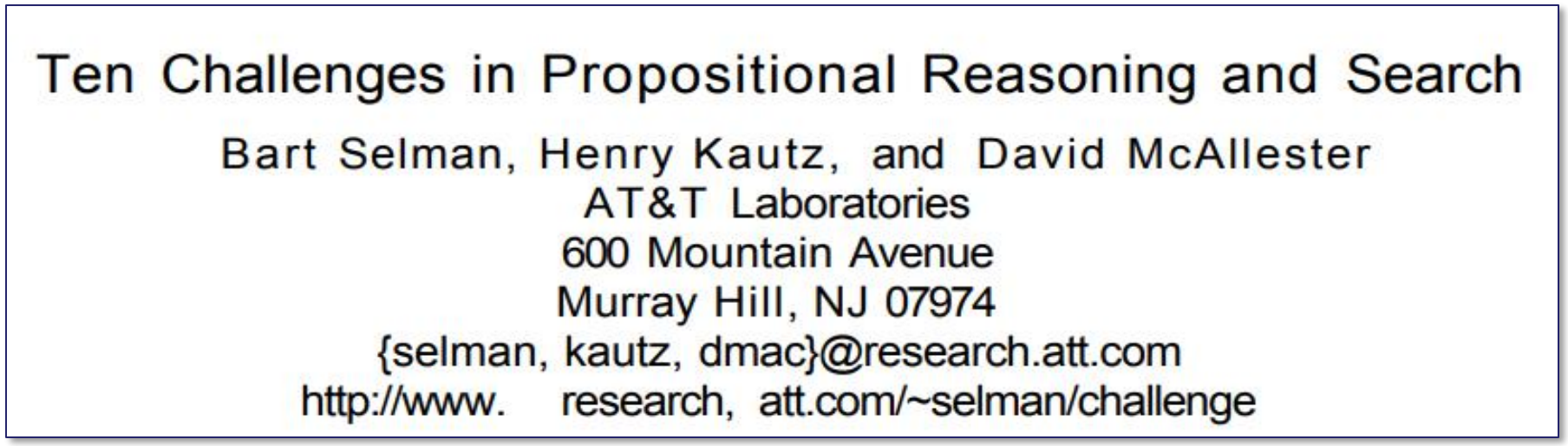
- Local search walks in the search space, trying to visit a satisfying assignment

--- incomplete, cannot prove unsatisfiability.

Outline

- Brief Introduction to SAT
- Hybrid SAT Algorithm
- Parallel SAT and MILP Algorithms

Challenge of Combining CDCL and Local Search



Ten Challenges in Propositional Reasoning and Search

Bart Selman, Henry Kautz, and David McAllester
AT&T Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974
{selman, kautz, dmac}@research.att.com
http://www. research, att.com/~selman/challenge

Challenge 7: Demonstrate the successful combination of stochastic search and systematic search techniques, by the creation of a new algorithm that outperforms the best previous examples of both approaches.

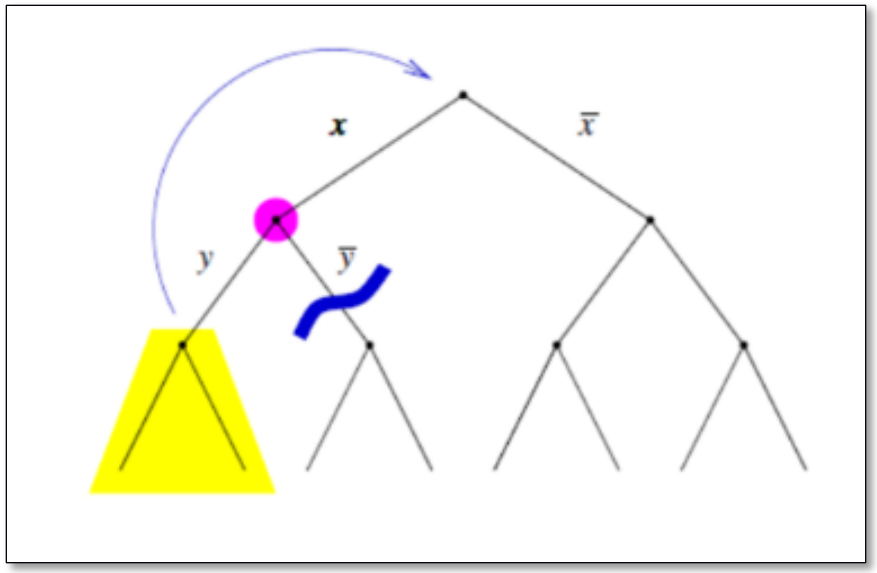
[Bart Selman, Henry Kautz and David McAllester, AAAI 1997]

Challenge of Combining CDCL and Local Search

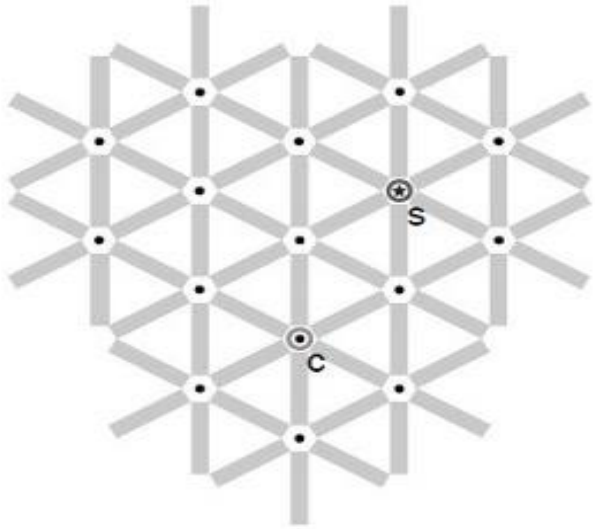
- Local search as main body
  - hybridGM （SAT 2009）, SATHYS （LPAR 2010）
  - GapSAT: use CDCL as preprocessor before local search (SAT 2020)
  - Use resolution in local search (AAAI 1996, AAAI 2005)
- DPLL/CDCL as main body
  - HINOTOS: local search finds subformulas for CDCL to solve (SAT 2008)
  - WalkSatz: calls WalkSAT at each node of a DPLL solver Satz (CP 2002)
  - CaDiCaL and Kissat: a local search solver is called when the solver resets the saved phases and is used immediately after the local search process (2019)
- Sequential portfolio
  - Sparrow2Riss, CCAnr+glucose, SGSeq

Deep Cooperation of CDCL and Local Search

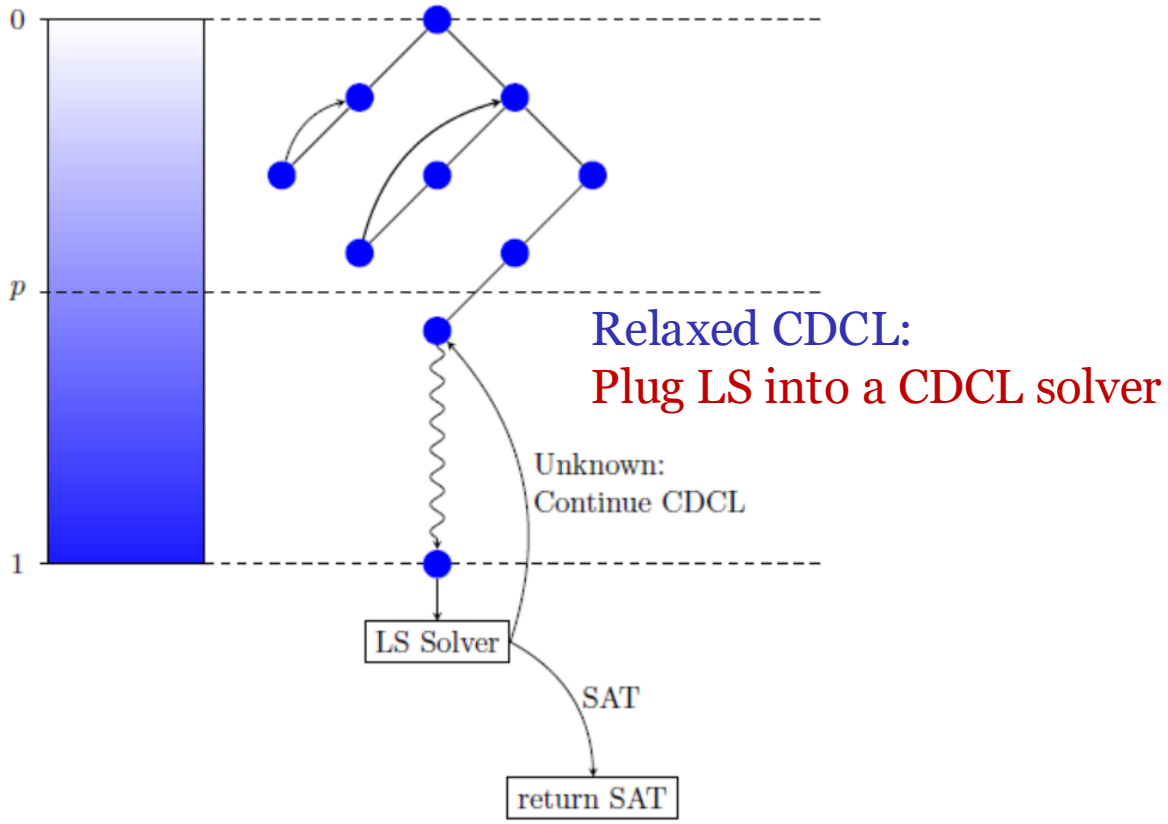
CDCL focuses on a local space in a certain period
→Better to integrate reasoning techniques



Local search walks in the whole search space
→Better at sampling
→use it as a sampler instead of solver!



Deep Cooperation of CDCL and Local Search



[Cai,Zhang, SAT '21]

A history of this work and similar works independently by Biere

[Cai,Zhang,Fleury,Biere,JAIR '22]

- Call local search in promising branches
- Gather useful information
  - local optimum
  - frequency in unsatified clauses

Improve Branching Heuristics via Local Search

CDCL solvers prefer the variable which may cause conflicts faster (e.g. VSIDS)

Enhance branching heuristic with conflict frequency in local search:

- calculate the conflict frequency: frequency of occurring in unsatisfied clauses
- Normalization: $ls_{confl_{num(\mathrm{x})}}$
- for each variable x, its activity is increased by $ls\_confl\_num(x)$

Local Search Rephasing

Phase selection (assign the picked variable with True or False?) is important for CDCL.

Most modern CDCL solvers utilize the phase saving heuristic [PipatsrisawatDarwiche, SAT'07]

Local search rephasing

- After each restart of CDCL, reset the saved phases of all variables with assignments by local search.

Phase Name	α_longest_LS	α_latest_LS	α_best_LS	no change
Probability	20%	65%	5%	10%

$\alpha\_longest\_LS$: the assignment of the local search procedure in which the initial solution is extended from the longest branch during past CDCL search.
α_best_LS: the assignment with smallest cost among all local search procedures.
α_latest_LS: the assignment of the latest local search procedure.
(the assignment of a local search procedure is the best found assignment)

Deep Cooperation of CDCL and Local Search

Apply the idea to CDCL-based SAT solvers

solver	#SAT	#UNSAT	#Solved	PAR2	#SAT	#UNSAT	#Solved	PAR2
	SC2017(351)				SC2018(400)			
glucose_4.2.1	83	101	184	5220.0	95	95	190	5745.9
glucose+rx	88	95	183	5220.0	113	95	208	5745.9
glucose+rx+rp	112	94	206	4668.5	141	87	228	4438.2
glucose+rx+rp+cf	110	94	204	4668.5	150	91	241	4438.2
Maple-DL-v2.1	101	113	214	4531.0	133	102	235	4533.9
Maple-DL+rx	101	112	213	4531.0	149	101	250	4533.9
Maple-DL+rx+rp	111	103	214	4447.1	158	93	251	4447.1
Maple-DL+rx+rp+cf	116	107	223	4139.4	162	97	259	3927.6
Kissat_sat	115	114	229	3943.5	167	98	265	3748.4
Kissat_sat+cf	113	113	226	4007.0	178	104	282	3407.4
CCAnr	13	N/A	13	9629.9	56	N/A	56	8622.0
	SC2019(400)				SC2020(400)			
glucose_4.2.1	118	86	204	5437.6	68	91	159	6494.6
glucose+rx	120	84	204	5437.6	93	88	181	6494.6
glucose+rx+rp	134	85	219	5066.3	130	85	215	5066.3
glucose+rx+rp+cf	140	85	225	4923.6	134	87	221	4977.9
Maple-DL-v2.1	143	97	240	4601.8	86	104	190	5835.7
Maple-DL+rx	146	93	239	4601.8	121	105	226	4977.9
Maple-DL+rx+rp	155	94	249	4415.3	142	99	241	4389.2
Maple-DL+rx+rp+cf	154	95	249	4377.4	151	106	257	4171.1
Kissat_sat	159	88	247	4293.4	146	114	260	4048.8
Kissat_sat+cf	162	90	252	4293.4	157	113	270	3806.8
CCAnr	13	N/A	13	9678.3	45	N/A	45	8978.7

20 51 9 24 3↓ 17 21 62 9 67 5 10

The hybrid method or similar ideas widely used in winners of main track in SAT Competitions since 2020

#SAT_bonus: solved by hybrid solver, but both original CDCL and LS fail.

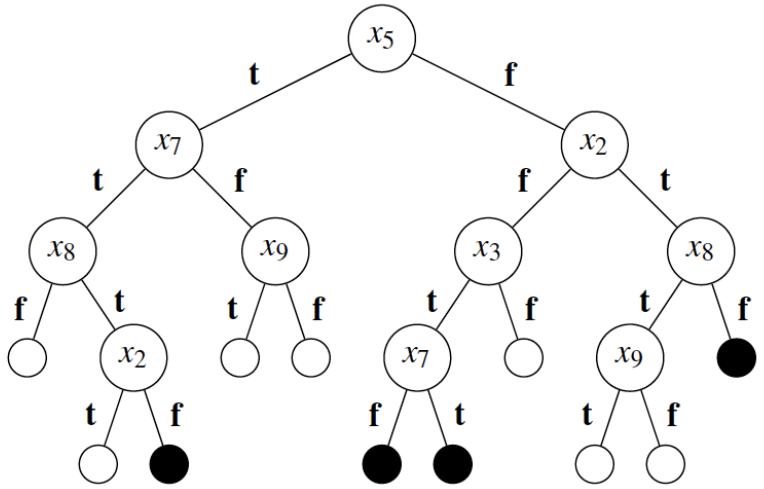
Solver	Analysis for SAT				Analysis for UNSAT	
	#byLS	#SAT_bonus	#LS_call	LS_time(%)	#LS_call	LS_time(%)
	SC2017(351)					
glucose+rx	20	11	24.28	21.66	16.36	5.52
glucose+rx+rp	10	33	17.77	18.46	14.33	4.86
glucose+rx+rp+cf	17	29	22.7	22.19	15.3	5.81
Maple+rx	16	9	13.86	7.52	11.18	2.03
Maple+rx+rp	11	15	9.63	10.43	6.54	2.36
Maple+rx+rp+cf	6	16	12.59	7.49	8.59	2.12
	SC2018(400)					
glucose+rx	50	4	11.27	20.66	29.62	4.94
glucose+rx+rp	47	31	9.46	18.4	21.66	5.64
glucose+rx+rp+cf	53	36	11.43	20.28	20.62	6.64
Maple+rx	52	7	4.8	13.02	11.69	2.81
Maple+rx+rp	56	13	4.84	15.21	8.7	3.04
Maple+rx+rp+cf	51	18	6.52	12.53	15.62	2.94
	SC2019(400)					
glucose+rx	14	8	26.46	10.79	17.42	6.39
glucose+rx+rp	10	26	22.68	8.67	14.59	5.14
glucose+rx+rp+cf	11	26	20.39	11.82	15.51	5.95
Maple+rx	14	7	12.66	2.67	12.94	1.98
Maple+rx+rp	9	14	8.6	3.17	16.59	2.53
Maple+rx+rp+cf	12	15	11.21	3.05	17.23	2.22
	SC2020(400)					
glucose+rx	30	9	14.94	11.75	14.67	10.27
glucose+rx+rp	23	37	13.17	10.79	9.4	9.71
glucose+rx+rp+cf	23	37	12.78	11.67	10.52	10.28
Maple+rx	19	13	14.21	6.69	10.24	5.25
Maple+rx+rp	30	29	8.53	6.62	11.7	6.18
Maple+rx+rp+cf	23	36	10.95	6.05	14.17	5.42

Outline

- Brief Introduction to SAT
- Hybrid SAT Algorithm
- Parallel SAT and MILP Algorithms

Partition

Divide the search space into smaller sub-spaces and solve them separately.



cutoff leaf

refuted leaf

$F_1 := F \wedge (x_5 \wedge x_7 \wedge \neg x_8)$

$F_2 := F \wedge (x_5 \wedge x_7 \wedge x_8 \wedge x_2)$

$F_3 := F \wedge (x_5 \wedge \neg x_7 \wedge x_9)$

$F_4 := F \wedge (x_5 \wedge \neg x_7 \wedge \neg x_9)$

$F_5 := F \wedge (\neg x_5 \wedge \neg x_2 \wedge \neg x_3)$

$F_6 := F \wedge (\neg x_5 \wedge x_2 \wedge x_8 \wedge x_9)$

$F_7 := F \wedge (\neg x_5 \wedge x_2 \wedge x_8 \wedge \neg x_9)$

- Cube and Conquer
  - The key is to choose the variable to partition
    - measurement: result in a large reduction of the formula.
  - Use incremental SAT solver

Portfolio

Parallelly Run different SAT solvers

(could be one solver with different configurations)

- Key: Highly complementary solvers/configurations.

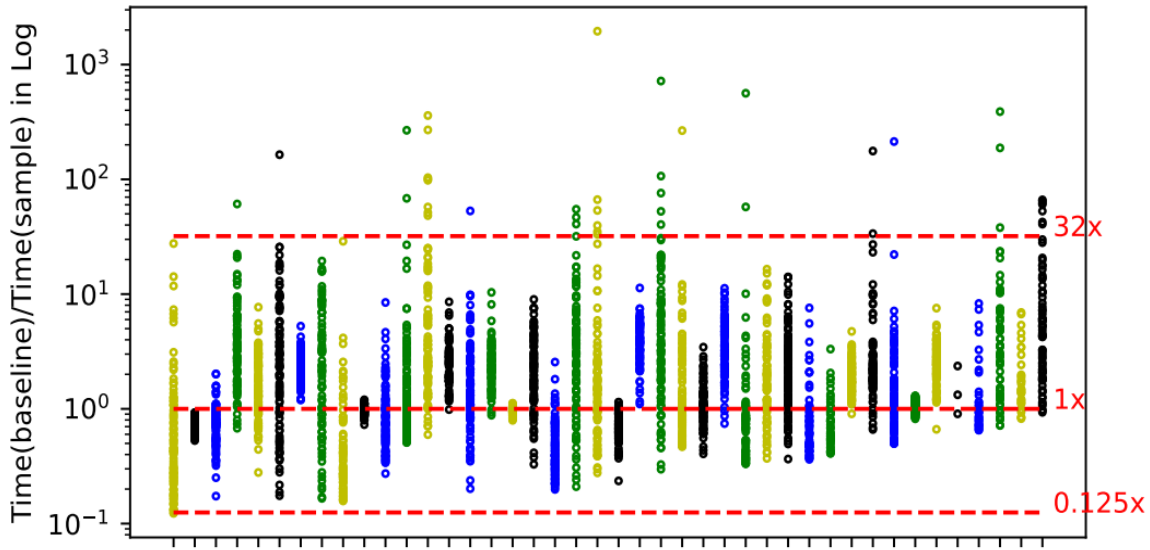
Portfolio

Parallelly Run different SAT solvers

(could be one solver with different configurations)

- Key: Highly complementary solvers/configurations.

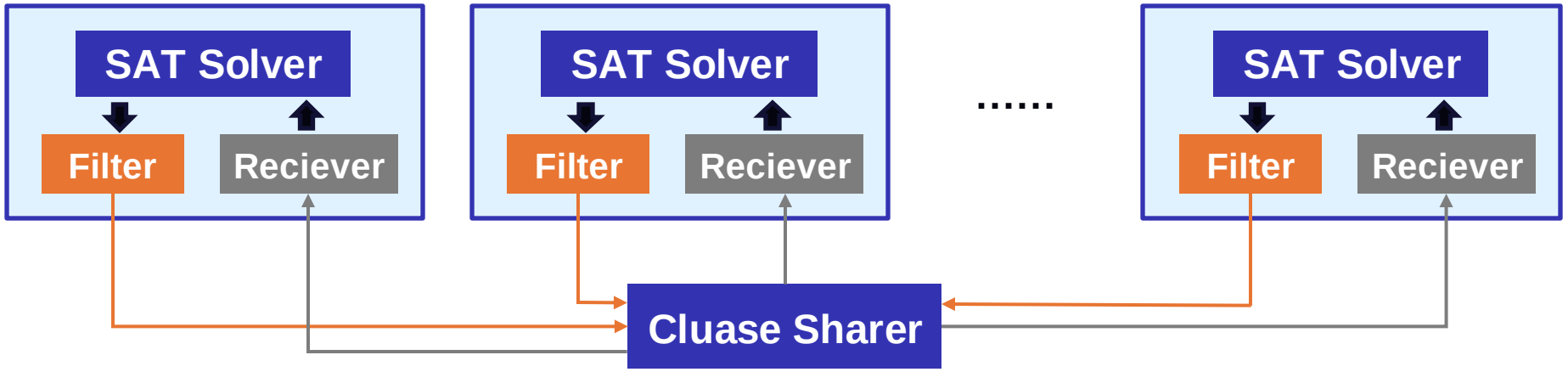
- Random shuffle: For each thread, randomly shuffles the order in which variables enter the heap.



Large variation of run time among different branching orders

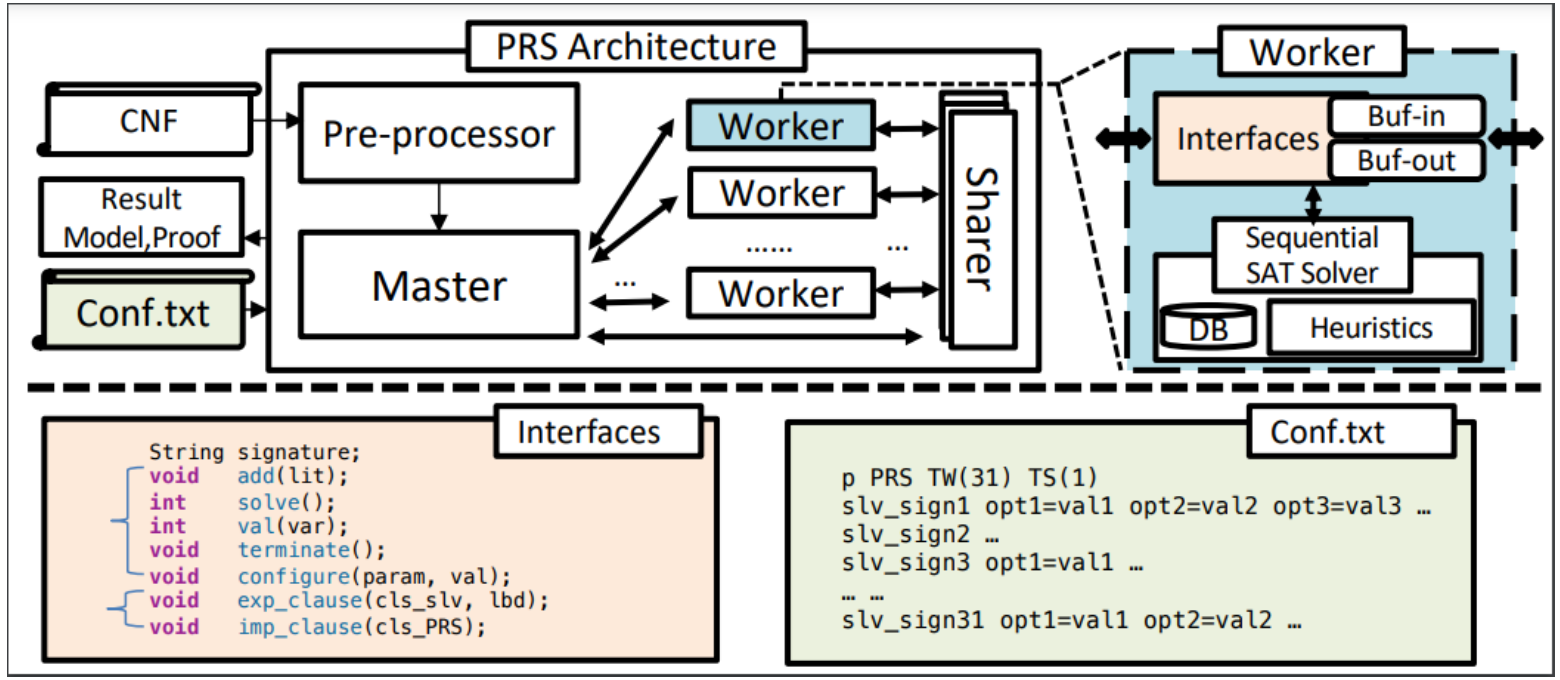
Improve Portfolio: Clause Sharing

- Why Clause Sharing
  - Multiple solvers work independently → **Explore the same search space redundantly**
- Key idea: Share learnt clauses to avoid redundant work

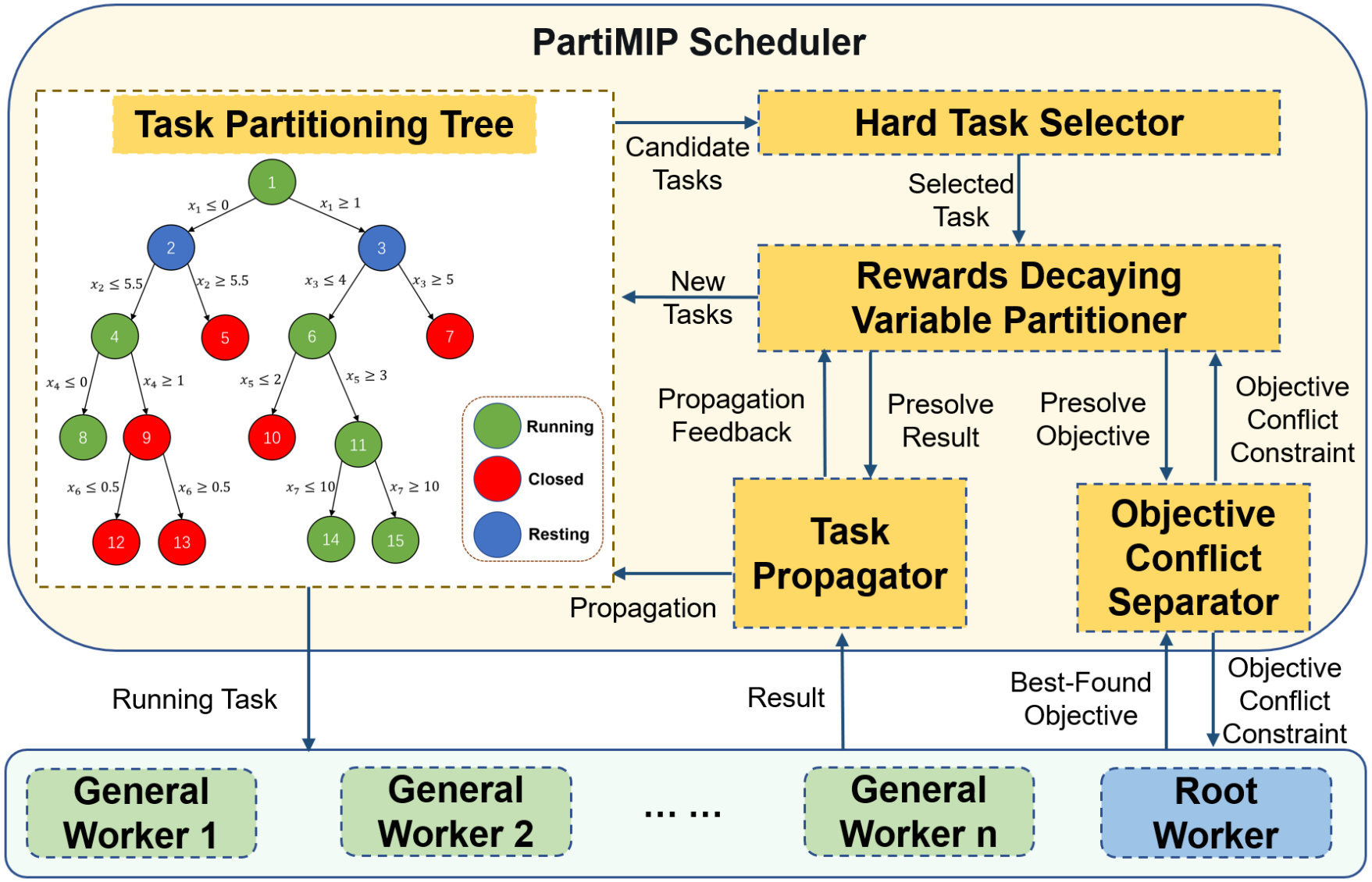


PRS: A State-of-the-art Parallel SAT Solver

- Comprehensive framework that supports preprocessing, dynamic clause sharing, reproducible parallel SAT solving, and parallel UNSAT proof generating.
- Portfolio: Random Shuffle, Diversification, Clause Sharing
- Winner of parallel tracks in SAT competitions 2022 and 2023, and used in winner of 2024.



Parallel MILP by Partitioning



Parallel MILP by Partitioning

- Establish new records for 21 challenging MILP instances on MIPLIB benchmark.
- **Close an open instance “shiftreg5-1” by proving the optimality**

instance	variable number	constraint number	previous	ours
dlr1	9142907	1735470	2708148.96	2708064.137
s82	1690631	87878	-33.78523765	-33.79705762
bmocbd3	403771	152791	-372986719.7	-373286017.2
neos-5151569-mologa	108116	45671	686759699	686750731.3
neos-4232544-orira	87060	180600	5557371.4	5553207.125
dws012-02	51108	26382	122074.2014	121112.0559
shiftreg5-1	48736	31424	522.1155	520.2562365
polygonpack5-15	48163	163429	-55494653.84	-55494686.56
sct5	37265	13304	-228.1172304	-228.1194928
neos-4230265-orari	32980	76374	81765.21022	73755
neos-4292145-piako	32950	75834	29160.50026	28122.49998
adult-regularized	32674	32709	7022.953543	7022.953543
cmflsp40-36-2-10	28152	4266	66452235.08	66452234.49
gmut-76-40	24338	2586	-14169441.78	-14169460.97
supportcase23	24275	40502	-12160.65936	-12160.65936
polygonpack4-7	10788	34529	-51837707.95	-51837712.98
neos-5045105-creuse	3848	252	20.5714291	20.57141059
eva1aprime6x6opt	3514	34872	-16.31528288	-18.10099528
dfn-bwin-DBE	3285	235	73623.79	73623.78816
gsvm2rl11	2001	1500	18121.638	18121.63764
gsvm2rl9	801	600	7438.181168	7438.181021

Future Directions

Specific constraint solvers, are (much) better than general solvers.

Exploit the feature

- What are the major operations?
- Are there some variables more important than others?

Configure the solver

- Tune the parameters, auto-tuning
- Choose the good combination of the strategies
- Automatically optimize the solver by LLM (see AutoSAT on Arxiv)

Use computing resource: Parallel and distributed solvers

(seek for a scaling method)

Suggestions

To improve the power of SAT and MILP solvers for Symmetric-key Cryptanalysis?

- Submit your benchmarks to SAT/PB/SMT competitions and MIPLIB website
- Make more benchmarks available, maintain a library
- Bring people from two communities together
- Go to the constraint solving community to introduce the problems

Welcome to visit us:

Beijing, China

http://solver.ios.ac.cn/en/

caisw@ios.ac.cn

Thank you!