Verifying RUP Proofs of Propositional Unsatisfiability

Allen Van Gelder
Computer Science Department
University of California
Santa Cruz, CA, USA

avg@cs.ucsc.edu
http://www.cse.ucsc.edu/∼avg/Papers/

Overview — This talk is about propositional logic only

- Review of Conflict Graphs Used in Sat Solvers
- Practicality of Writing Explicit Resolution Refutations
  - A Pitfall to Avoid
  - TVR and PUP Strategies
- Shorter Alternatives to Explicit Proofs
  - The Complexity Issue — Deterministic Log Space vs. P–Complete
  - Proof "Traces" — Much Shorter than Explicit Proofs
  - Reverse Unit Propagation (RUP) Proof Format
  - Converting RUP to Explicit Resolution
- Practicality of Independently Verifying Explicit Resolution Refutations
  - First Experimental Results — Comparison with Goldberg and Novikov
  - SAT 2007 Competition — Certified Unsat Track: 166 gigabyte proof verified
- Current Limits and Capabilities

Brief Summary of Results

Reverse Unit Propagation (%RUP format) described and evaluated.

- TVR and PUP strategies described and briefly compared.
- %RUP format is easiest to implement; can be retrofitted to existing solvers.
- Supported and used in Certified Unsat Track of SAT 2007 Competition.

First large-scale verifications in which verifier (`checker3`) and solvers (various) were developed by independent researchers.

First large-scale verifications of explicit resolution proofs (%RES format)

- Numerous engineering issues
- `checker3` has verified a %RES proof of 163 gigabytes.
- C library function `mmap` was essential for implementation simplicity.
- Should scale to nearly 1 terabyte, then chip's 40-bit address limit reached.
- Compressed formats up to 1000 time shorter.

Conflict Graph: Basic Data Structure in Many Modern SAT Solvers

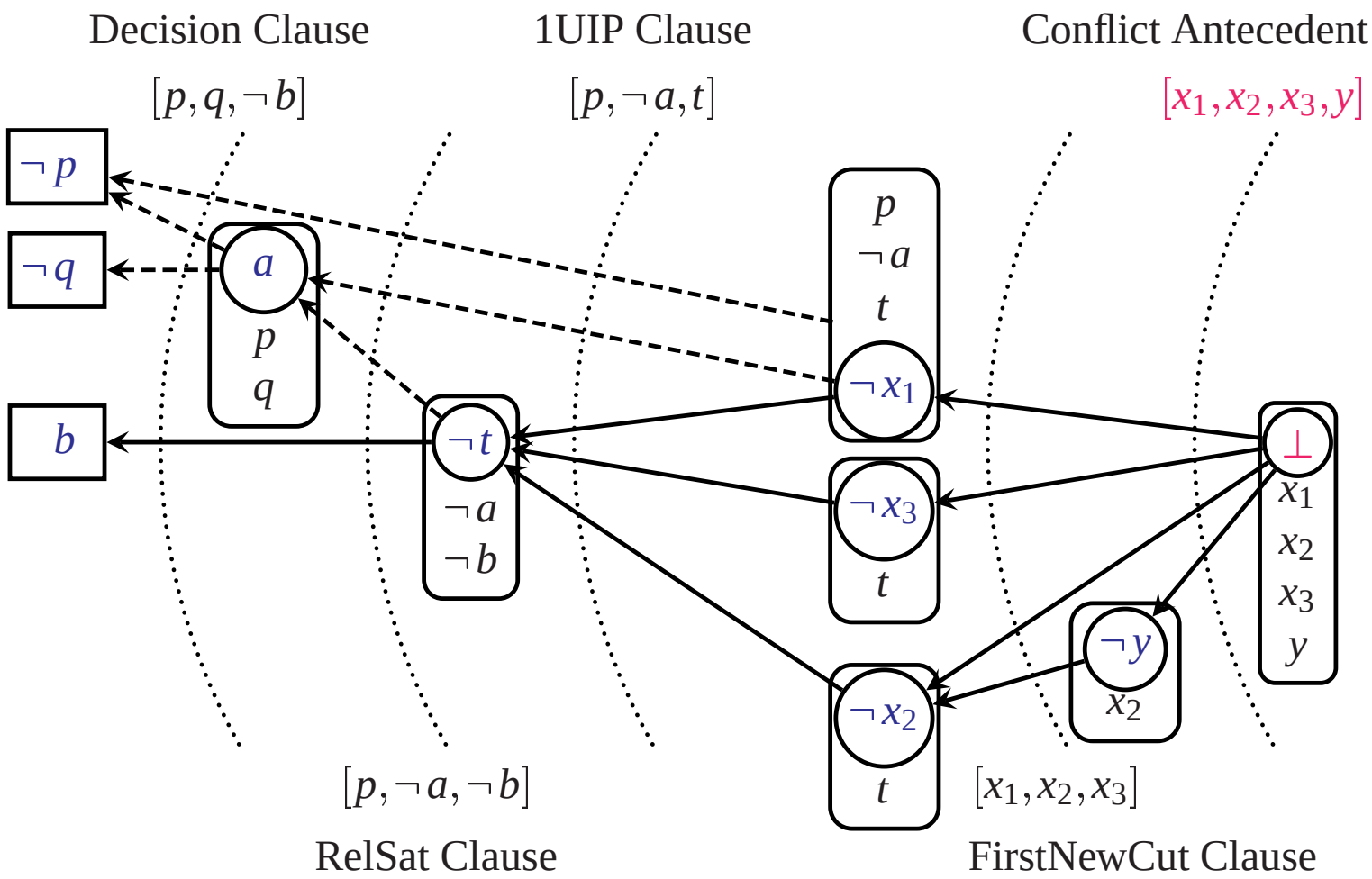
- Introduced in current form in GRASP by Marques-Silva and Sakallah [96,99]
- Adopted into Chaff by Moskewicz, Madigan, Zhao, L. Zhang, and Malik [01]
- Variations in zChaff by L. Zhang, Madigan, Moskewicz, and Malik [01]
- Theoretical analysis by Beame, Kautz, and Sabharwal [*JAIR* 2004]

If x is implied by unit-clause propagation, there is some clause $[x, \neg y_1, \neg y_2, \ldots]$ such that $y_1, y_2, \ldots$ were assumed or implied earlier.

Put arrows from vertex x to vertices $y_1, y_2, \ldots$.

$[x, \neg y_1, \neg y_2, \ldots]$ is called the *antecedent* of x.

Various Cuts in Conflict Graph Yield Different Conflict Clauses



$\bot$ denotes false.

Dashed lines go to vertices at lower (earlier) “decision levels”.

From Conflict Graph to Resolution Derivation of Conflict Clause

A Pitfall to Avoid

L. Zhang and Malik pseudocode to generate resolution proof of empty clause:

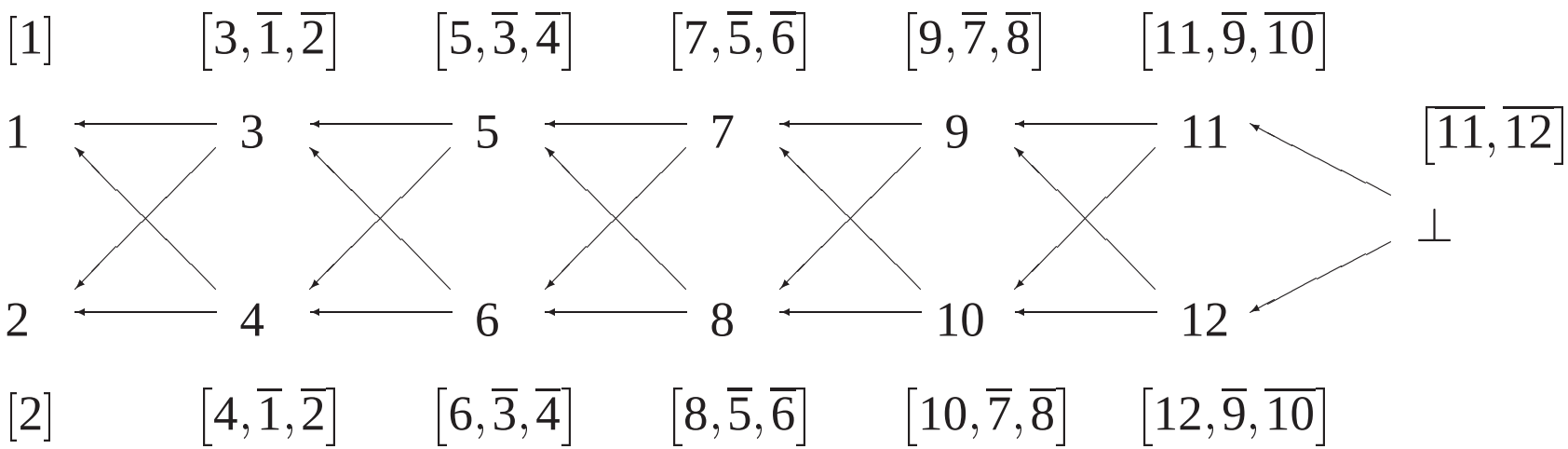
```
1.    cl = final_conflicting_clause;
2.    while (!is_empty_clause(cl)) {
3.        lit = choose_literal(cl);
4.        var = variable_of_literal(lit);
5.        ante_cl = antecedent(var);
6.        cl = resolve(cl, ante_cl); }
```

Note: On line 1, `final_conflicting_clause` is the antecedent of $\perp$, *not* a conflict clause.

Similar pseudocode for deriving nonempty conflict clause in `cl`.

DAG Family with Exponential Worst Case for Foregoing Procedure

- Diagram shows $h = 6$ instance. See Van Gelder, SAT 2007 for details.
- Antecedent clauses are shown in brackets; overbars denote negation.
- Grows as 2^h for "choose smallest variable number".
- Grows as 2^h for "choose smallest DAG height".



`zverify_df` version 2005.11.15 is exponential for reversed variables numbers (thanks to Zhaohui Fu for verifying this experimentally).
Fixed in 2007 version.

What Was the Fix?

Trivial Resolution (TVR) defined by Beame, Kautz, Subharwal

- *Linear* (must use previous resolvent), *Input* (must use an input clause)
- No repeated clashing literal

Their Proposition 4: Conflict clause can be derived by a TVR.

Caveat: Appropriate order must be found.

zchaff was already using TVR for all *nonempty* conflict clauses.

Fix: Do the same to derive the *empty* clause!

Alternate fix: Pseudo-Unit Propagation (PUP)

Experimental Manifestation of Inefficiency

- Sizes are "mega" literals ("mega" = 2^{20}).
- "+?" indicates job was killed when size exceeded stated number.

Small Goldberg-Novikov DATE03

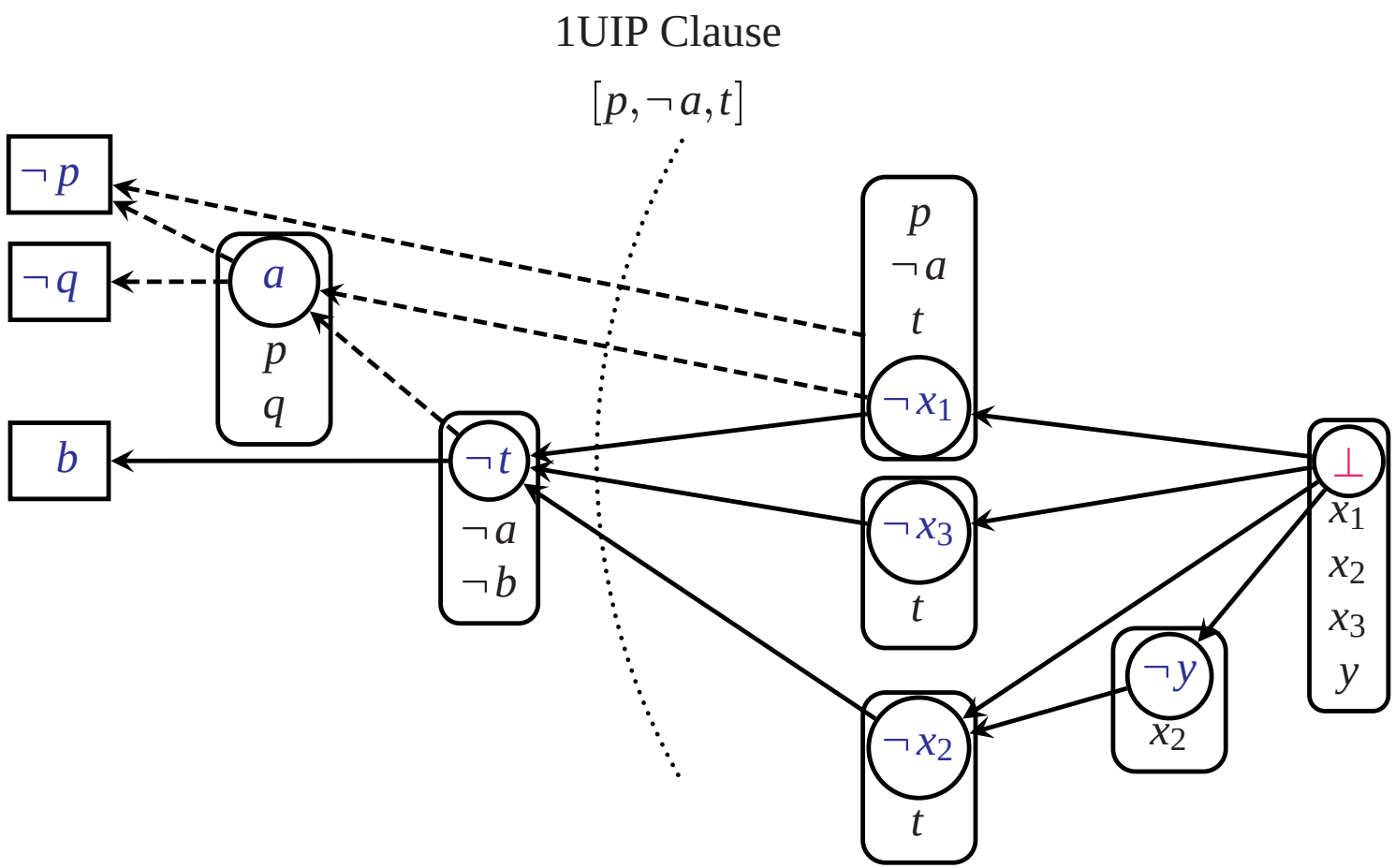
benchmark	2004.11.15	with fix
5pipe	153	15
5pipe_1_ooo	986	92
5pipe_5_ooo	2320	94
6pipe	169	97
6pipe_6_ooo	3072+?	486
7pipe	358	319
9vliw_bp_mc	308	58
barrel7	2754	5
barrel8	3072+?	73
barrel9	3072+?	80
c3540	393	78
c5315	162	9
c7552	2650	22

IBM_FV SAT 2005 Competition

benchmark	2004.11.15	with fix
01_SAT_dat.k10	3	0.134
07_SAT_dat.k30	3	2.582
07_SAT_dat.k35	3	2.859
18_SAT_dat.k10	174	1
18_SAT_dat.k15	102,520+?	14
1_11_SAT_dat.k10	13	0.338
1_11_SAT_dat.k15	128	2
20_SAT_dat.k10	13	0.208
23_SAT_dat.k10	3	0.062
23_SAT_dat.k15	15,259+?	4
26_SAT_dat.k10	0.036	0.036
2_14_SAT_dat.k10	70	1
2_14_SAT_dat.k15	3220	7

TVR and Pseudo-Unit Propagation (PUP) Strategies

"Pseudo-Unit" because the clause looks like a unit clause to the solver.



PUP: may begin $\mathrm{res}(x_2, [\neg x_2, t], [\neg y, x_2])$ or $\mathrm{res}(x_2, [\neg x_2, t], [x_1, x_2, x_3, y])$.

TVR: may begin $\mathrm{res}(y, [x_1, x_2, x_3, y], [\neg y, x_2])$ but *NOT* $\mathrm{res}(x_2, [\neg x_2, t], [x_1, x_2, x_3, y])$.

Shorter Alternatives to Explicit Proofs

- *Resolve–Trace* proof format given by L. Zhang and Malik [DATE 03].
  - Implemented in `zchaff` and `zverify`
- *Conflict Clause Proof* proposed by Goldberg and Novikov [DATE 03].
  - Implemented in `berkmin` but not publicly available
- *Resolution Proof Trace* (%RPT) offered in SAT 2007 Certified Unsat Track
  - *binary* — expanded by `expandtrace0`

Reverse Unit Propagation (RUP) Proof Format

Prover claims clause C is implied by prior clauses $F \cup D$.

The RUP requirement: $F \cup D \cup \neg(C)$ must derive $\perp$ by *unit clause propagation*.

"Reverse" because units clauses $\neg(C)$ are propagated back into earlier clauses.

Grasp–Chaff–Berkmin conflict clauses are RUP clauses.

- Stated without proof by Goldberg and Novikov [DATE03];
- Proved for a large class of conflict clauses by Beame *et al.* [JAIR 2004]

Other reasoning methods may be able to use the RUP format.

Compression achieved of as much as 1000 to 1 vs. *explicit* proofs (%RES format).

Converting RUP to Explicit Resolution

rupToRes expands %RUP to %RES format.

- Expansion not unique
- rupToRes is *not* trusted ! !

Implementation based on zchaffSE (after fixes).

- Disable usual selection of a branching literal.
- Assert all the unit clauses of $\neg(C)$ at one "decision level."
- If $\perp$ is derived, output the (TVR) resolution proof of C.
- Otherwise, proof is buggy!

Often a clause *stronger than* C is derived!

The Complexity Issue

Quis custodiet ipsos custodes? (Who will guard the guards themselves?)
—Juvenal (first century A.D.)

Ultimately, some verifier must be *trusted* by human inspection.

Proposal: Specify the proof format itself to be in a very low complexity class.

- Easy to independently develop your *own* verifier.
- Reasonable to demand that the verifier be simple before you *trust* it.
- No need to *trust* the solver !

Deterministic Log Space (DLOG) vs. P–Complete

A decision problem is P–Complete if *every* problem that can be solved in deterministic polynomial time can be reduced to this problem.

A decision problem is in DLOG if it can be solved by a program that has a *fixed number* of working storage registers (can re-read the input).

- Each register can hold a number of $\log(L)$ bits for inputs of length L.
- No other memory, no `malloc`, no recursion.

Where do Proof Formats Fall in Terms of Complexity Classes?

Unit clause propagation is P–Complete.

- Requires linear space for known algorithms.
- Corollary: RUP format is P–Complete.

Reachability in a directed graph is believed *not* to be solvable in DLOG.

- Corollary: Resolution proof trace formats (clauses not materialized, as in %RPT and `resolve_trace`) are not in DLOG.

An explicit resolution proof (clauses materialized, operands specified, as in %RES) *is* in DLOG.

Practicality of Independently Verifying Explicit Resolution Refutations

First Experimental Results — Proof Length Comparisons

	Input		Full %RES		Resolve-Trace			RUP			`rupToRes`		
[GN03]	Vars	Cls	lits	cls	nbrs	ratio	cls	lits	ratio	cls	lits	ratio	cls
Benchmark	$\times 10^3$	$\times 10^3$	$\times 10^6$	$\times 10^6$	$\times 10^6$		$\times 10^6$	$\times 10^6$		$\times 10^6$	$\times 10^6$		$\times 10^6$
longmult13	7	20	4005	10.3	55	0.014	0.545	138	0.035	0.545	14140	3.53	30.8
w10_70	33	104	2582	4.3	6	0.002	0.033	8	0.003	0.033	2924	1.13	4.7
fifo8-400	260	708	9149	5.3	16	0.002	0.201	87	0.010	0.201	11905	1.30	6.4

- Ratios are to length of Full Resolution (%RES).
- Benchmarks (27 in all, see table in proceedings) from:

  [GN03] Goldberg and Novikov, Verification of Proofs of Unsatisfiability for CNF Formulas, *DATE* 2003.

Main findings:

- Resolution proofs about 100 times shorter than predicted by above paper.
- Superlinear growth within families not observed.

First Experimental Results — Times in CPU seconds

[GN03] Benchmark	zchaff Time	zchaffSE Increment	checker3 Time	Res.Trace Increment	zverify Time	rupToRes + checker3
5pipe	11	4	2	4	1	7
5pipe_1_ooo	25	12	13	4	2	32
5pipe_5_ooo	29	12	12	11	2	23
6pipe	80	23	12	31	3	61
6pipe_6_ooo	142	58	67	6	4	110
7pipe	254	54	39	16	5	169
9vliw_bp_mc	39	14	8	2	2	53
longmult12	1,102	445	57	143	21	3681
longmult13	1,663	343	89	163	23	4887
longmult14	974	369	44	153	16	3775
longmult15	257	72	69	37	5	625
w10_45	4	5	6	0	0	9
w10_60	27	60	73	10	2	88
w10_70	97	250	391	2	4	394
fifo8-200	136	27	27	41	3	56
fifo8-300	370	76	77	66	6	184
fifo8-400	2,737	1,089	397	75	16	2553

- `zchaff`, `zchaffSE`, and `zverify` version Jan. 2007.
- Easier benchmarks not shown (27 in all, see table in proceedings).

SAT 2007 Competition — Certified Unsat Track

http://www.cse.ucsc.edu/∼avg/ProofChecker/cert-poster.pdf

Pigeon-Hole formulas plus 17 industrial benchmarks from SAT 2007 competition

Resources: 1 CPU hour at 2.6 GHz, 8 GB real memory, about 6 GB swap
36 GB (most cases) on a local disk

The Contestants

Program	Proof Format	Authors	Program	Proof Format	Authors
booleforce	%RES	A. Biere	tts	%RPT	I. Spence
picosat	%RES	A. Biere	zchaffSE07	%RES	Z. Fu, L. Zhang
picosat	%RUP	A. Biere	zchaff_0	%RES	A. Van Gelder, Z. Fu, L. Zhang

- tts clear winner on pigeon-hole (*asymptotically optimal?*).
- booleforce edged out zchaff on industrial.
- zchaff had best balance of industrial and pigeon-hole.
- picosat RUP produced most proofs, but we had too little disk to expand many.

Biggest Verified %RES Proof: IBM_FV_2004_rb30_Sd.k15

booleforce: 163GB picosat RUP: 86GB (from 52MB) zchaff: 36GB.

Conclusions

%RES "language" is recognizable in deterministic log space (DLOG).

%RUP format is easiest to implement.

Current Limits and Capabilities

- `checker3` has verified a %RES proof of 163 gigabytes.
- C library function `mmap` was essential for implementation simplicity.
- Should scale to nearly 1 terabyte, then chip's 40-bit address limit reached.

Summary for GN03 benchmarks (adding up all times):

- Producing `resolve_trace` and checking with `zverify` incurs 11% overhead.
- Producing %RES and checking with `checker3` incurs 38% overhead.
- Producing %RUP, expanding with `rupToRes`, and checking with `checker3` incurs 200% overhead.
- Wide variations from the average.

Take-Home Message: CPU time is the bottleneck anymore.