Parallel QBF Solving: State of the Art Techniques and Future Perspectives

Florian Lonsing

Knowledge-Based Systems Group, Vienna University of Technology, Austria
http://www.kr.tuwien.ac.at/staff/lonsing/

First Workshop on Parallel Constraint Reasoning (PCR'17)
August 6, 2017, Gothenburg, Sweden
Collocated with CADE-26

This work is supported by the Austrian Science Fund (FWF) under grant S11409-N23.

Introduction

Quantified Boolean Formulas (QBF):

- Existential ($\exists$) / universal ($\forall$) quantification of propositional variables.
- Checking QBF satisfiability: PSPACE-complete.
- Potentially more succinct encodings than propositional logic.

Example

- QBF $\psi := \hat{Q}.\phi$ in *prenex conjunctive normal form (PCNF)*.
- $\psi = \underbrace{\forall u \exists x.}_{\text{quantifier prefix}} \underbrace{(\bar{u} \vee x) \wedge (u \vee \bar{x})}_{\text{propositional CNF}}.$
- Recursive semantics:
  - Assign variables in prefix ordering, recurse on simplified formula $\psi[A]$ under assignment A.
  - Base cases: $\bot$ is unsatisfiable, $\top$ is satisfiable.
  - $\forall u.\psi$ is satisfiable iff $\psi[u/\bot]$ and $\psi[u/\top]$ are satisfiable.
  - $\exists x.\psi$ is satisfiable iff $\psi[x/\bot]$ or $\psi[x/\top]$ is satisfiable.

Introduction: Sequential QBF Solving

QCDCL: [GNT06, Let02, ZM02]

- Extension of *conflict-driven clause learning (CDCL)* for SAT solving.
- Derivation of new *learned clauses* and *cubes* (conjunctions of literals).
- Underlying proof system: *Q-resolution calculus* [KBKF95].

Expansion: [AB02, Bie04, JKMSC16, JM15b, RT15]

- Successively eliminate variables from a QBF.
- Shannon expansion [Sha49]: $\exists x.\phi(x, \ldots) \equiv \phi(\bot, \ldots) \vee \phi(\top, \ldots)$
- Counter example guided abstraction refinement (CEGAR).

QBF Proof Complexity:

- Recent results: orthogonality of proof systems [BCJ15, JM15a].
- Expansion vs. Q-resolution: exponential separation wrt. proof sizes.

Outline

State of the Art Techniques:

- Two main parallel approaches, inspired by parallel SAT solving.
- Examples of solvers illustrating the main approaches.

Challenges:

- Limitations of parallel solving due to prefix ordering.
- Proof generation.

Future Perspectives:

- Proof complexity (orthogonality) as a motivation to parallelize.
- Experiments to highlight performance diversity of sequential solvers.

Main Parallel QBF Solving Approaches

Search Space Handling:

- Given a QBF with n variables, 2^n possible assignments.
- Additionally, prefix ordering and quantifier types must be considered (cf. semantics).
- Approaches differ in how search space is explored.
- Generalizations of approaches to parallel SAT solving.

Example

$\psi = \forall u \exists x . (\bar{u} \vee x) \wedge (u \vee \bar{x})$.

- Out of the 2^2 possible assignments, $\{u, x\}$ and $\{\neg u, \neg x\}$ are sufficient to show the satisfiability of ψ.

Main Parallel QBF Solving Approaches

Portfolio Approach:

- Run several instances of the same solver (or different ones) on the original formula.
- Problem: instances potentially explore same parts of the search space.
- Diversify solver instances by parameter settings.
- With(out) sharing of learned clauses and cubes.

Main Parallel QBF Solving Approaches

Splitting Approach:

- Instances by construction explore different parts of the search space.
- Parts are described by assignments that follow the prefix ordering.
- Instance gets original formula ψ and assignment A, and solves $\psi[A]$.
- Similar to *guiding path method* in parallel SAT solving.

Example

$\psi = \forall u \exists x.(\bar{u} \vee x) \wedge (u \vee \bar{x})$.

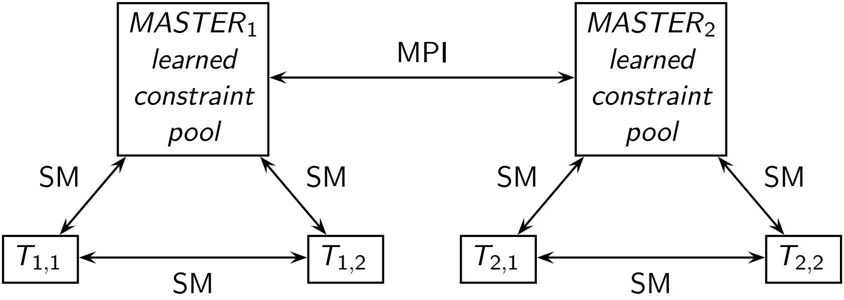
Possible splittings, starting from left end of prefix:

- $A = \{u\}$, solver instance works on $\psi[A] = \exists x.(x)$.
- $A = \{\neg u\}$, solver instance works on $\psi[A] = \exists x.(\neg x)$.

Unsound splittings:

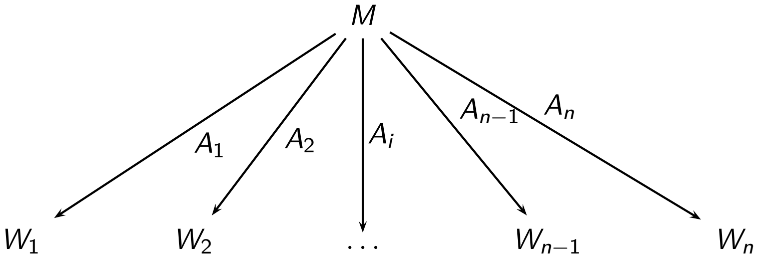
- $A = \{x\}$, solver instance works on $\psi[A] = \forall u.(u)$.
- $A = \{\neg x\}$, solver instance works on $\psi[A] = \forall u.(\neg u)$.

Portfolio Approach Example: HordeQBF



- Massively parallel portfolio to run on distributed architectures [BL16].
- First published parallel portfolio (cf. sequential ones [PT09, SM07]).
- Extension of HordeSat [BSS15], supports any QCDCL solver.
- Hierarchical parallelism: shared memory and message passing.
- Master executes identical but diversified QCDCL solvers (threads).
- Master processes: exchange of learned constraints via MPI.

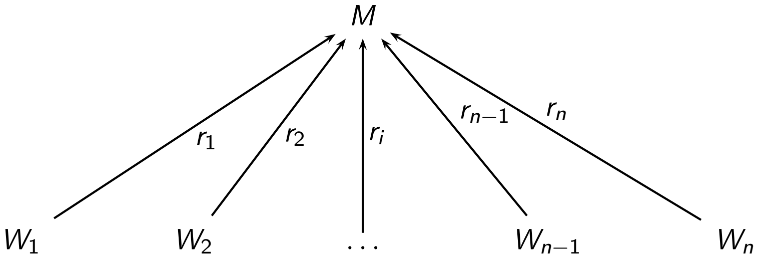
Splitting Approach Example: MPIDepQBF



Master M**:**

- Splits search space by assignments A_i, calling workers W_i.
- Combines results obtained by workers, further splitting.
- Manages exchange of learned constraints (not part of MPIDepQBF).
- Solvers differ in splitting strategy, exchange of learned constraints, shared memory vs. message passing [FMS00, LSB09, LSB$^+$11].

Splitting Approach Example: MPIDepQBF

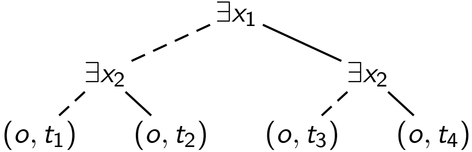


Workers:

- Operate on original formula ψ, receive assignments A_i and timeout.
- Treat A_i as *assumptions* in QCDCL to solve $\psi[A_i]$.
- Solving ψ under assumptions A_i: re-use of learned constraints within each W_i in next run of QCDCL under different A'_i.
- Send result r_i back to master or request increased timeout.
- Exchange of learned constraints either via master or among workers.

Splitting Approach Example

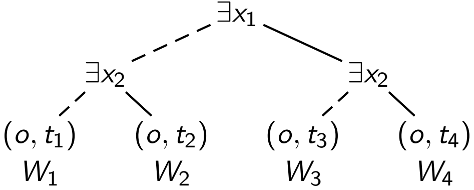
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



Initially 4 idle workers, 4 open leaves (subcases) with individual timeouts t_i.

Splitting Approach Example

PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.

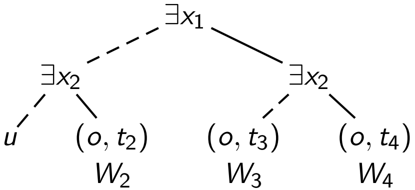


Assign open subcases to idle workers W_i by sending assumptions:

- W_1 works on $\psi[\neg x_1, \neg x_2]$.
- W_2 works on $\psi[\neg x_1, x_2]$.
- W_3 works on $\psi[x_1, \neg x_2]$.
- W_4 works on $\psi[x_1, x_2]$.

Splitting Approach Example

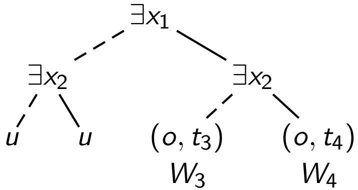
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



W_1 returns "unsat" for subcase $\psi[\neg x_1, \neg x_2]$ and becomes idle.

Splitting Approach Example

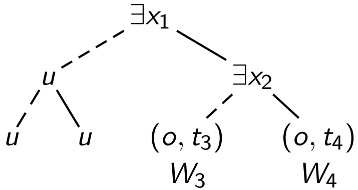
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



W_2 returns "unsat" for subcase $\psi[\neg x_1, x_2]$ and becomes idle.

Splitting Approach Example

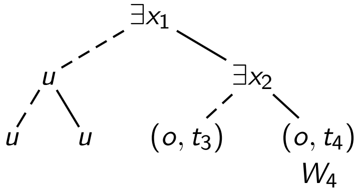
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



Since $\psi[\neg x_1, \neg x_2]$ and $\psi[\neg x_1, x_2]$ unsatisfiable, also $\psi[\neg x_1]$ unsatisfiable.

Splitting Approach Example

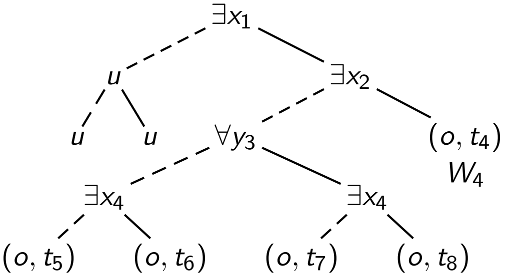
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



W_3 times out, W_1, W_2 are idle, only 2 open leaves: generate new subcases.

Splitting Approach Example

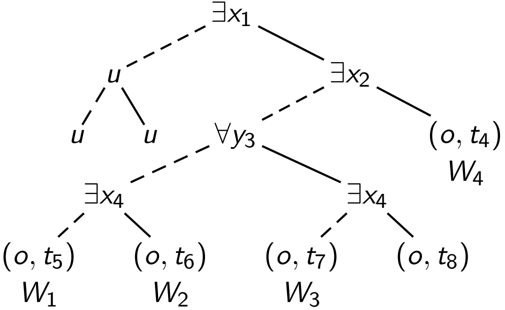
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



Replace open leaf (o, t_3) by binary tree based on $\forall y_3$ and $\exists x_4$.

Splitting Approach Example

PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.

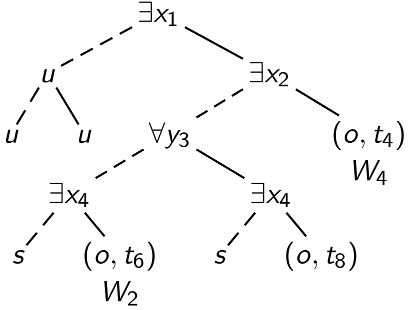


Assign open subcases to idle workers W_1, W_2, and W_3 by assumptions:

- W_1 works on $\psi[x_1, \neg x_2, \neg y_3, \neg x_4]$.
- W_2 works on $\psi[x_1, \neg x_2, \neg y_3, x_4]$.
- W_3 works on $\psi[x_1, \neg x_2, y_3, \neg x_4]$.

Splitting Approach Example

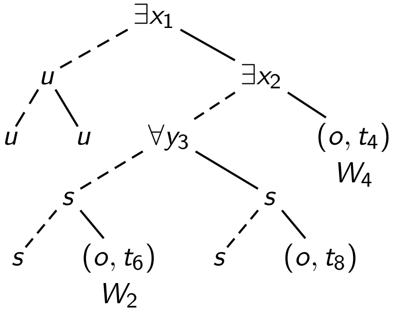
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



W_1 and W_3 return "sat" for $\psi[x_1, \neg x_2, \neg y_3, \neg x_4]$ and $\psi[x_1, \neg x_2, y_3, \neg x_4]$.

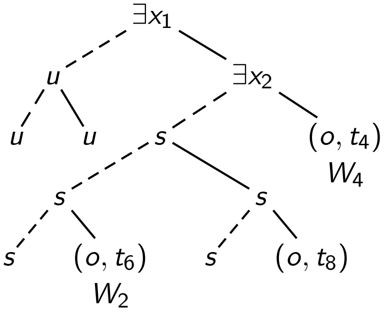
Splitting Approach Example

PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



Subcases $\psi[x_1, \neg x_2, \neg y_3]$ and $\psi[x_1, \neg x_2, y_3]$ are satisfiable.

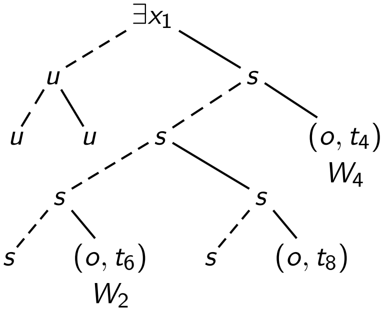
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



Subcase $\psi[x_1, \neg x_2]$ is satisfiable.

Splitting Approach Example

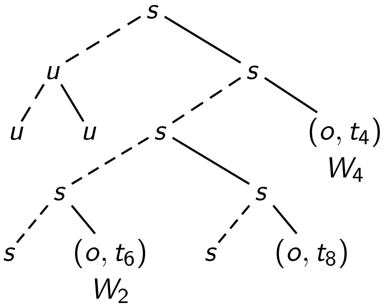
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



Subcase $\psi[x_1]$ is satisfiable.

Splitting Approach Example

PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \dots \phi$.



Finally, conclude that ψ is satisfiable.

Parallel QBF Solvers: Historical Perspective

parallel QBF solver	base solver	QCDCL-based	portfolio	information sharing	process management	most recent paper
Aspvall et al.	n.i.					1996 [ALLS96]
pcaqe	caqe	×	×	×	Pthreads	–
hiqqerfork	DepQBF	✓	✓	×	`fork`	–
HordeQBF	DepQBF[1]	✓	✓	✓	MPI	2016 [BL16]
MPIDepQBF	DepQBF	✓	×	×	MPI	2014 [JKLS14]
par-pd-depqbf	DepQBF	✓	✓	×	`fork`	–
PAQuBE	QuBE	✓	×	✓	MPI	2011 [LSB$^+$11]
PQSolve	QSolve	∼[3]	×	×	MPI	2000 [FMS00]
PQSAT	QSAT	×	×	×	MPI	2010 [MNS10]
PQUABS	quabs	×	×	×	Pthreads	2016 [Ten16]
QMiraXT	MiraXT[2]	✓	×	✓	Pthreads	2009 [LSB09]

✓ yes/supported × no/not supported – unpublished n.i. not implemented
[1] any QCDCL solver can be used [2] parallel SAT-solving framework [3] DPLL-based

- Comprehensive list of solvers (joint work w. Martina Seidl, JKU Linz).
- Publicly available: pcaqe, HordeQBF, MPIDepQBF, PQUABS.

Parallel QBF Solvers: Historical Perspective

parallel QBF solver	base solver	QCDCL-based	portfolio	information sharing	process management	most recent paper
Aspvall et al.	n.i.					1996 [ALLS96]
pcaqe	caqe	×	×	×	Pthreads	–
hiqqerfork	DepQBF	✓	✓	×	`fork`	–
HordeQBF	DepQBF[1]	✓	✓	✓	MPI	2016 [BL16]
MPIDepQBF	DepQBF	✓	×	×	MPI	2014 [JKLS14]
par-pd-depqbf	DepQBF	✓	✓	×	`fork`	–
PAQuBE	QuBE	✓	×	✓	MPI	2011 [LSB+11]
PQSolve	QSolve	∼[3]	×	×	MPI	2000 [FMS00]
PQSAT	QSAT	×	×	×	MPI	2010 [MNS10]
PQUABS	quabs	×	×	×	Pthreads	2016 [Ten16]
QMiraXT	MiraXT[2]	✓	×	✓	Pthreads	2009 [LSB09]

✓ yes/supported × no/not supported – unpublished n.i. not implemented
[1] any QCDCL solver can be used [2] parallel SAT-solving framework [3] DPLL-based

- Portfolio solvers.
- par-pd-depqbf: two solver instances, diversification by input formula.

Parallel QBF Solvers: Historical Perspective

parallel QBF solver	base solver	QCDCL-based	portfolio	information sharing	process management	most recent paper
Aspvall et al.	n.i.					1996 [ALLS96]
pcaqe	caqe	×	×	×	Pthreads	–
hiqqerfork	DepQBF	✓	✓	×	`fork`	–
HordeQBF	DepQBF[1]	✓	✓	✓	MPI	2016 [BL16]
MPIDepQBF	DepQBF	✓	×	×	MPI	2014 [JKLS14]
par-pd-depqbf	DepQBF	✓	✓	×	`fork`	–
PAQuBE	QuBE	✓	×	✓	MPI	2011 [LSB+11]
PQSolve	QSolve	∼[3]	×	×	MPI	2000 [FMS00]
PQSAT	QSAT	×	×	×	MPI	2010 [MNS10]
PQUABS	quabs	×	×	×	Pthreads	2016 [Ten16]
QMiraXT	MiraXT[2]	✓	×	✓	Pthreads	2009 [LSB09]

✓ yes/supported × no/not supported – unpublished n.i. not implemented
[1] any QCDCL solver can be used [2] parallel SAT-solving framework [3] DPLL-based

- Splitting solvers.
- Different (more complex) splitting and knowledge sharing strategies.

Parallel QBF Solvers: Historical Perspective

parallel QBF solver	base solver	QCDCL-based	portfolio	information sharing	process management	most recent paper
Aspvall et al.	n.i.					1996 [ALLS96]
pcaqe	caqe	×	×	×	Pthreads	–
hiqqerfork	DepQBF	✓	✓	×	`fork`	–
HordeQBF	DepQBF[1]	✓	✓	✓	MPI	2016 [BL16]
MPIDepQBF	DepQBF	✓	×	×	MPI	2014 [JKLS14]
par-pd-depqbf	DepQBF	✓	✓	×	`fork`	–
PAQuBE	QuBE	✓	×	✓	MPI	2011 [LSB+11]
PQSolve	QSolve	∼[3]	×	×	MPI	2000 [FMS00]
PQSAT	QSAT	×	×	×	MPI	2010 [MNS10]
PQUABS	quabs	×	×	×	Pthreads	2016 [Ten16]
QMiraXT	MiraXT[2]	✓	×	✓	Pthreads	2009 [LSB09]

✓ yes/supported × no/not supported – unpublished n.i. not implemented

[1] any QCDCL solver can be used [2] parallel SAT-solving framework [3] DPLL-based

- Aspvall et al.: not implemented, so far of theoretical interest only.
- Based on algorithm for restricted class of QBFs: only binary clauses.

Parallel QBF Solvers: Historical Perspective

parallel QBF solver	base solver	QCDCL-based	portfolio	information sharing	process management	most recent paper
Aspvall et al.	n.i.					1996 [ALLS96]
pcaqe	caqe	×	×	×	Pthreads	–
hiqqerfork	DepQBF	✓	✓	×	fork	–
HordeQBF	DepQBF[1]	✓	✓	✓	MPI	2016 [BL16]
MPIDepQBF	DepQBF	✓	×	×	MPI	2014 [JKLS14]
par-pd-depqbf	DepQBF	✓	✓	×	fork	–
PAQuBE	QuBE	✓	×	✓	MPI	2011 [LSB+11]
PQSolve	QSolve	∼[3]	×	×	MPI	2000 [FMS00]
PQSAT	QSAT	×	×	×	MPI	2010 [MNS10]
PQUABS	quabs	×	×	×	Pthreads	2016 [Ten16]
QMiraXT	MiraXT[2]	✓	×	✓	Pthreads	2009 [LSB09]

✓ yes/supported × no/not supported – unpublished n.i. not implemented

[1] any QCDCL solver can be used [2] parallel SAT-solving framework [3] DPLL-based

- PQUABS: based on expansion/CEGAR, allows prenex NNF input.
- pcaqe: based on expansion/CEGAR, parallelization of CEGAR.

Challenges

Linear Prefix Ordering:

- Limited potential of search space splitting (cf. QBF semantics).
- Difficult work load balancing (idle resources).

Parallel Proof Generation:

- Generating and checking proofs in splitting/portfolio solvers.
- Sequential solving: QRAT proof system [HSB14, HSB17].

Exploiting Solver Diversity:

- Portfolios: integrate orthogonal approaches (QCDCL vs. expansion).
- Problem: sharing learned information in such hybrid portfolios.
- Alternative diversification strategies: run workers on original/preprocessed formula, primal vs. dual encodings [VG13].

Experiments (1/5)

Benchmark Set:

- Set S_{402}: 402 out of 825 QBFEVAL'16 prenex CNF instances.
- Excluded ones: solved or propositional after preprocessing by Bloqqer.
- Limits: 1800 seconds, 7 GB memory.

Sequential QBF Solvers:

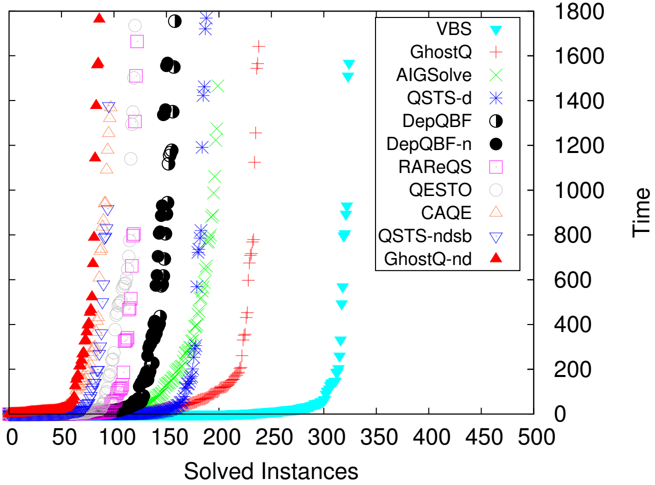
- Ten (variants of) solvers from QBFEVAL'16 that were top ranked.
- Solvers are based on five different solving paradigms/proof systems.

Motivation:

- Highlight the potential of parallel portfolios.
- Disclaimer: presented results are obtained under idealistic conditions.
- Virtual best solver (VBS): ideal portfolio where solving time of fastest sequential solver is attributed to VBS.

Experiments (2/5)

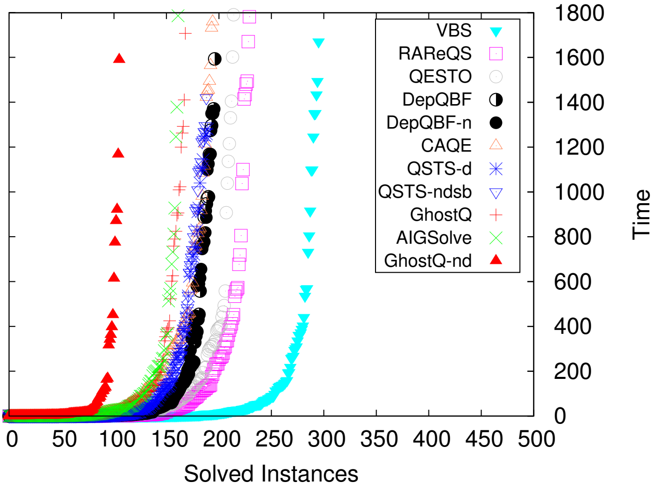
	Solved
VBS	325
GhostQ	239
AIGSolve	200
QSTS-d	189
DepQBF	159
DepQBF-n	152
RAReQS	123
QESTO	121
caqe	98
QSTS-ndsb	96
GhostQ-nd	87



- 402 instances without preprocessing.
- VBS solves 35% more instances than best solver.

Experiments (3/5)

	Solved
VBS	296
RAReQS	230
QESTO	215
DepQBF	197
DepQBF-n	195
caqe	195
QSTS-d	190
QSTS-ndsb	189
GhostQ	169
AIGSolve	162
GhostQ-nd	106



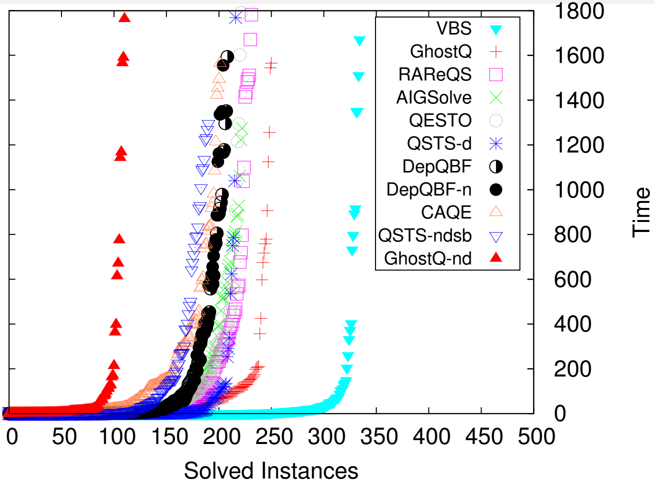
- 402 instances with preprocessing (harmful for some solvers).
- VBS solves 28% more instances than best solver, and becomes worse.

Table : 402 preprocessed instances: instances solved by the virtual best solver (VBS) in classes by number of quantifier blocks (#q) and number of formulas (#f), and relative contribution (%) of solvers to instances solved by VBS.

#q	#f	*VBS*	*RAReQS*	*QESTO*	*DepQBF*	*DepQBF-n*	*caqe*	*QSTS-d*	*QSTS-ndsb*	*GhostQ*	*AIGSolve*	*GhostQ-nd*
2	34	21	19.0	9.5	9.5	**23.8**	4.7	4.7	4.7	0.0	**23.8**	0.0
3	236	168	**49.4**	13.6	2.9	8.9	0.5	0.5	4.7	6.5	12.5	0.0
4–6	24	19	26.3	5.2	10.5	21.0	0.0	0.0	5.2	0.0	**31.5**	0.0
7–9	31	31	16.1	3.2	**35.4**	22.5	0.0	3.2	6.4	0.0	12.9	0.0
10–15	28	21	23.8	4.7	14.2	**38.0**	0.0	0.0	4.7	9.5	0.0	4.7
16–20	30	24	12.5	8.3	12.5	**25.0**	0.0	8.3	**25.0**	0.0	4.1	4.1
21–	19	12	16.6	0.0	0.0	**41.6**	0.0	25.0	16.6	0.0	0.0	0.0
2–3	270	189	**46.0**	13.2	3.7	10.5	1.0	1.0	4.7	5.8	13.7	0.0
4–	132	107	18.6	4.6	17.7	**28.0**	0.0	5.6	11.2	1.8	10.2	1.8
2–	402	296	**36.1**	10.1	8.7	16.8	0.6	2.7	7.0	4.3	12.5	0.6

Experiments (5/5)

	Solved
VBS	335
GhostQ	251
RAReQS	232
AIGSolve	223
QESTO	222
QSTS-d	217
DepQBF	209
DepQBF-n	205
caqe	202
QSTS-ndsb	191
GhostQ-nd	111



- "Best foot forward" analysis (cf. [LSVG16]).
- Select best solving time with(out) preprocessing.
- VBS solves more instances than individual VBS solvers (325 and 296).

Summary and Outlook

State of the Art Techniques:

- Inspiration from parallel SAT: early research on parallel QBF solving.
- QBF semantics as an obstacle: workload balancing, solver correctness.
- Only recent implementations publicly available.

Future Perspectives:

- Proof complexity theory: motivation to combine approaches.
- Experiments: multiple solvers contribute to VBS that are based on different solving paradigms and proof systems.
- Hypothetical parallel portfolio outperforms best solver (+30%).
- Diversification: not only by solver parameters but also by input formula, e.g., with(out) preprocessing,. . .
- Proof generation remains challenging in sequential/parallel solving.

Appendix

[APPENDIX] Experiments

Solver	*S*	$\bot$	$\top$	*Time*
VBS	325	163	162	150K
GhostQ	239	112	127	313K
AIGSolve	200	96	104	389K
QSTS-d	189	88	101	397K
DepQBF	159	90	69	457K
DepQBF-n	152	89	63	467K
RAReQS	123	77	46	513K
QESTO	121	74	47	523K
caqe	98	52	46	564K
QSTS-ndsb	96	54	42	559K
GhostQ-nd	87	53	34	581K

(a) Set S_{402} (not preprocessed).

Solver	*S*	$\bot$	$\top$	*Time*
VBS	296	160	136	214K
RAReQS	230	127	103	335K
QESTO	215	117	98	358K
DepQBF	197	96	101	391K
DepQBF-n	195	98	97	392K
caqe	195	101	94	409K
QSTS-d	190	100	90	406K
QSTS-ndsb	189	100	89	408K
GhostQ	169	79	90	441K
AIGSolve	162	78	84	447K
GhostQ-nd	106	58	48	542K

(b) Set S'_{402} (preprocessed by Bloqqer).

Table : Solved instances (S), solved unsatisfiable ($\bot$) and satisfiable ones ($\top$), and total wall clock time including time outs on sets S_{402} (2a) and S'_{402} (2b).

[APPENDIX] Experiments

Solver	*S*	$\bot$	$\top$	*Time*
VBS	335	170	165	133K
GhostQ	251	115	136	289K
RAReQS	232	128	104	332K
AIGSolve	223	105	118	346K
QESTO	222	120	102	344K
QSTS-d	217	111	106	341K
DepQBF	209	102	107	370K
DepQBF-n	205	101	104	374K
caqe	202	104	98	392K
QSTS-ndsb	191	100	91	404K
GhostQ-nd	111	62	49	536K

- “Best foot forward” analysis.

References

References I

[AB02] Abdelwaheb Ayari and David A. Basin.
QUBOS: Deciding Quantified Boolean Logic Using Propositional Satisfiability Solvers.
In *FMCAD*, volume 2517 of *LNCS*, pages 187–201. Springer, 2002.

[ALLS96] Bengt Aspvall, Christos Levcopoulos, Andrzej Lingas, and Robert Storlind.
On 2-QBF Truth Testing in Parallel.
Information Processing Letters, 57(2):89–93, 1996.

[BCJ15] Olaf Beyersdorff, Leroy Chew, and Mikolás Janota.
Proof Complexity of Resolution-based QBF Calculi.
In *STACS*, volume 30 of *LIPIcs*, pages 76–89. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

[Bie04] Armin Biere.
Resolve and Expand.
In *SAT*, volume 3542 of *LNCS*, pages 59–70. Springer, 2004.

[BL16] Tomas Balyo and Florian Lonsing.
HordeQBF: A Modular and Massively Parallel QBF Solver.
In Nadia Creignou and Daniel Le Berre, editors, *Proc. of the 19th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2016)*, volume 9710 of *LNCS*, pages 531–538. Springer, 2016.

References II

[BSS15] Tomas Balyo, Peter Sanders, and Carsten Sinz.
HordeSat: A Massively Parallel Portfolio SAT Solver.
In Marijn Heule and Sean Weaver, editors, *Proc. of the 18th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2015)*, volume 9340 of *LNCS*, pages 156–172. Springer, 2015.

[FMS00] Rainer Feldmann, Burkhard Monien, and Stefan Schamberger.
A Distributed Algorithm to Evaluate Quantified Boolean Formulae.
In Henry A. Kautz and Bruce W. Porter, editors, *Proc. of the 17th Nat. Conference on Artificial Intelligence and 12th Conference on on Innovative Applications of Artificial Intelligence (AAA/IAAI 2000)*, pages 285–290. AAAI Press / The MIT Press, 2000.

[GNT06] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella.
Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas.
JAIR, 26:371–416, 2006.

[HSB14] Marijn Heule, Martina Seidl, and Armin Biere.
A Unified Proof System for QBF Preprocessing.
In *IJCAR*, volume 8562 of *LNCS*, pages 91–106. Springer, 2014.

References III

[HSB17] Marijn J. H. Heule, Martina Seidl, and Armin Biere.
Solution Validation and Extraction for QBF Preprocessing.
J. Autom. Reasoning, 58(1):97–125, 2017.

[JKLS14] Charles Jordan, Lukasz Kaiser, Florian Lonsing, and Martina Seidl.
MPIDepQBF: Towards Parallel QBF Solving without Knowledge Sharing.
In Carsten Sinz and Uwe Egly, editors, *Proc. of the 17th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2014)*, volume 8561 of *LNCS*, pages 430–437. Springer, 2014.

[JKMSC16] Mikolás Janota, William Klieber, João Marques-Silva, and Edmund Clarke.
Solving QBF with Counterexample Guided Refinement.
Artif. Intell., 234:1–25, 2016.

[JM15a] Mikolás Janota and Joao Marques-Silva.
Expansion-Based QBF Solving versus Q-Resolution.
Theor. Comput. Sci., 577:25–42, 2015.

[JM15b] Mikolás Janota and Joao Marques-Silva.
Solving QBF by Clause Selection.
In *IJCAI*, pages 325–331. AAAI Press, 2015.

References IV

[KBKF95] Hans Kleine Büning, Marek Karpinski, and Andreas Flögel.
Resolution for Quantified Boolean Formulas.
Inf. Comput., 117(1):12–18, 1995.

[Let02] Reinhold Letz.
Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas.
In *TABLEAUX*, volume 2381 of *LNCS*, pages 160–175. Springer, 2002.

[LSB09] Matthew D. T. Lewis, Tobias Schubert, and Bernd Becker.
QmiraXT - A Multithreaded QBF Solver.
In Carsten Gremzow and Nico Moser, editors, *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV)*, pages 7–16. Universitätsbibliothek Berlin, Germany, 2009.

[LSB+11] Matthew Lewis, Tobias Schubert, Bernd Becker, Paolo Marin, Massimo Narizzano, and Enrico Giunchiglia.
Parallel QBF Solving with Advanced Knowledge Sharing.
Fundamenta Informaticae, 107(2-3):139–166, 2011.

[LSVG16] Florian Lonsing, Martina Seidl, and Allen Van Gelder.
The QBF Gallery: Behind the Scenes.
Artif. Intell., 237:92–114, 2016.

References V

[MNS10] Benoit Da Mota, Pascal Nicolas, and Igor Stéphan.
A new parallel architecture for QBF tools.
In *Proc. of the Int. Conference on High Performance Computing and Simulation (HPCS 2010)*, pages 324–330. IEEE, 2010.

[PT09] Luca Pulina and Armando Tacchella.
A self-adaptive multi-engine solver for quantified Boolean formulas.
Constraints, 14(1):80–116, 2009.

[RT15] Markus N. Rabe and Leander Tentrup.
CAQE: A Certifying QBF Solver.
In *FMCAD*, pages 136–143. IEEE, 2015.

[Sha49] Claude Elwood Shannon.
The Synthesis of Two-Terminal Switching Circuits.
Bell System Technical Journal, 28(1):59–98, 1949.

[SM07] Horst Samulowitz and Roland Memisevic.
Learning to Solve QBF.
In *AAAI*, pages 255–260. AAAI Press, 2007.

[Ten16] Leander Tentrup.
Non-prenex QBF Solving Using Abstraction.
In *In Proc. of the 19th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2016)*, volume 9710 of *LNCS*, pages 393–401. Springer, 2016.

[VG13] Allen Van Gelder.
Primal and Dual Encoding from Applications into Quantified Boolean Formulas.
In *CP*, volume 8124 of *LNCS*, pages 694–707. Springer, 2013.

[ZM02] Lintao Zhang and Sharad Malik.
Conflict Driven Learning in a Quantified Boolean Satisfiability Solver.
In *ICCAD*, pages 442–449. ACM / IEEE Computer Society, 2002.