MPIDepQBF: Towards Parallel QBF Solving without Knowledge Sharing

Charles Jordan[1] Lukasz Kaiser[2] Florian Lonsing[3] Martina Seidl[4]

[1]Division of Computer Science, Hokkaido University, Japan

[2]LIAFA, CNRS & Université Paris Diderot (currently at Google Inc.)

[3]Knowledge-Based Systems Group, TU Wien, Austria

[4]Institute for Formal Models and Verification, JKU Linz, Austria

International Conference on Theory and Applications of Satisfiability Testing (SAT)
July 14 - 17, 2014, Vienna, Austria

Supported by the Austrian Science Fund (FWF) under grants S11408-N23 and S11409-N23, and by the Japan Society for the Promotion of Science (JSPS) as KAKENHI No. 25106501.

Overview (1/2)

Quantified Boolean Formulas (QBF):

- Propositional logic with explicit quantification ($\forall, \exists$) of variables.
- PSPACE-complete decision problem: applications in formal verification, synthesis,...
- Considerable progress in QBF solving techniques: QBF Galleries 2013 and 2014.

Parallel Solving:

- Two paradigms: shared vs. distributed memory.
- Compared to SAT, parallel QBF solving has received little attention recently.

Overview (1/2)

Quantified Boolean Formulas (QBF):

- Propositional logic with explicit quantification ($\forall, \exists$) of variables.
- PSPACE-complete decision problem: applications in formal verification, synthesis,...
- Considerable progress in QBF solving techniques: QBF Galleries 2013 and 2014.

Parallel Solving:

- Two paradigms: shared vs. distributed memory.
- Compared to SAT, parallel QBF solving has received little attention recently.

Overview (2/2)

Related parallel QBF Solvers:

- Shared Memory (multi-threaded): QMiraXT [LSB09].
- Distributed memory (MPI-based): PQSolve [FMS00], PaQuBE [LMS$^+$09, LSB$^+$11].
- Sophisticated scheduling and load balancing.
- Strategies to share learned information (clauses and cubes).

This Work: MPIDepQBF

- MPI-based parallel QBF solver for distributed memory systems.
- Master coordinates workers to solve subproblems: sequential solver DepQBF.
- Search-space partitioning inspired by cube and conquer approach [HKWB11].
- No sharing of learned clauses: learned information is kept only locally in workers.
- Open source: `http://toss.sourceforge.net/develop.html`

Overview (2/2)

Related parallel QBF Solvers:

- Shared Memory (multi-threaded): QMiraXT [LSB09].
- Distributed memory (MPI-based): PQSolve [FMS00], PaQuBE [LMS+09, LSB+11].
- Sophisticated scheduling and load balancing.
- Strategies to share learned information (clauses and cubes).

This Work: MPIDepQBF

- MPI-based parallel QBF solver for distributed memory systems.
- Master coordinates workers to solve subproblems: sequential solver DepQBF.
- Search-space partitioning inspired by cube and conquer approach [HKWB11].
- No sharing of learned clauses: learned information is kept only locally in workers.
- Open source: `http://toss.sourceforge.net/develop.html`

QBF Syntax

QBF in Prenex Conjunctive Normal Form:

- Given a Boolean formula $\phi(x_1, \ldots, x_m)$ in CNF.
- Quantifier prefix $\hat{Q} := Q_1 B_1 Q_2 B_2 \ldots Q_m B_m$.
- Quantifiers $Q_i \in \{\forall, \exists\}$.
- Quantifier block $B_i \subseteq \{x_1, \ldots, x_m\}$ containing variables.
- QBF in prenex CNF (PCNF): $Q_1 B_1 Q_2 B_2 \ldots Q_m B_m.\phi(x_1, \ldots, x_m)$.
- $B_i \leq B_{i+1}$: quantifier blocks are linearly ordered (extended to variables, literals).

Example

- Given the CNF $\phi := (x \vee \neg y) \wedge (\neg x \vee y)$.
- Given the quantifier prefix $\hat{Q} := \forall x \exists y$.
- Prenex CNF: $\psi := \hat{Q}.\phi = \forall x \exists y.\ (x \vee \neg y) \wedge (\neg x \vee y)$.

Recursive Definition:

- Given a PCNF $\psi := Q_1 B_1 \dots Q_m B_m.\ \phi$.
- Recursively assign the variables in prefix order (from left to right).
- Assignment $A = \{l_1, \dots, l_n\}$: if $l_i \in A$ is a positive (negative) literal, then $var(l_i)$ is assigned to true (false).
- Base cases: the QBF $\top$ ($\bot$) is satisfiable (unsatisfiable).
- $\psi = \forall x \dots \phi$ is satisfiable if $\psi[\neg x]$ and $\psi[x]$ are satisfiable.
- $\psi = \exists x \dots \phi$ is satisfiable if $\psi[\neg x]$ or $\psi[x]$ is satisfiable.
- In $\psi[x]$ ($\psi[\neg x]$), every occurrence of x in ψ is replaced by $\top$ ($\bot$).
- Prerequisite: every variable is quantified in the prefix (no free variables).

QBF Semantics (2/2)

Example (continued)

The PCNF $\psi = \forall x \exists y.(x \vee \neg y) \wedge (\neg x \vee y)$ is satisfiable if

(1) $\psi[x] = \exists y.(y)$ and

(2) $\psi[\neg x] = \exists y.(\neg y)$ are satisfiable.

(1) $\psi[x] = \exists y.(y)$ is satisfiable since $\psi[x, y] = \top$ is satisfiable.
(2) $\psi[\neg x] = \exists y.(\neg y)$ is satisfiable since $\psi[\neg x, \neg y] = \top$ is satisfiable.

QBF Semantics (2/2)

Example (continued)

The PCNF $\psi = \forall x \exists y.(x \vee \neg y) \wedge (\neg x \vee y)$ is satisfiable if

(1) $\psi[x] = \exists y.(y)$ and

(2) $\psi[\neg x] = \exists y.(\neg y)$ are satisfiable.

(1) $\psi[x] = \exists y.(y)$ is satisfiable since $\psi[x, y] = \top$ is satisfiable.
(2) $\psi[\neg x] = \exists y.(\neg y)$ is satisfiable since $\psi[\neg x, \neg y] = \top$ is satisfiable.

QBF Semantics (2/2)

Example (continued)

The PCNF $\psi = \forall x \exists y.(x \lor \neg y) \land (\neg x \lor y)$ is satisfiable if

(1) $\psi[x] = \exists y.(y)$ and

(2) $\psi[\neg x] = \exists y.(\neg y)$ are satisfiable.

(1) $\psi[x] = \exists y.(y)$ is satisfiable since $\psi[x, y] = \top$ is satisfiable.
(2) $\psi[\neg x] = \exists y.(\neg y)$ is satisfiable since $\psi[\neg x, \neg y] = \top$ is satisfiable.

Example (continued)

The PCNF $\psi = \forall x \exists y.(x \vee \neg y) \wedge (\neg x \vee y)$ is satisfiable if

(1) $\psi[x] = \exists y.(y)$ and

(2) $\psi[\neg x] = \exists y.(\neg y)$ are satisfiable.

(1) $\psi[x] = \exists y.(y)$ is satisfiable since $\psi[x, y] = \top$ is satisfiable.

(2) $\psi[\neg x] = \exists y.(\neg y)$ is satisfiable since $\psi[\neg x, \neg y] = \top$ is satisfiable.

QBF Semantics (2/2)

Example (continued)

The PCNF $\psi = \forall x \exists y.(x \lor \neg y) \land (\neg x \lor y)$ is satisfiable if

(1) $\psi[x] = \exists y.(y)$ and

(2) $\psi[\neg x] = \exists y.(\neg y)$ are satisfiable.

(1) $\psi[x] = \exists y.(y)$ is satisfiable since $\psi[x, y] = \top$ is satisfiable.
(2) $\psi[\neg x] = \exists y.(\neg y)$ is satisfiable since $\psi[\neg x, \neg y] = \top$ is satisfiable.

QBF Solving under Assumptions

Definition

Let $\psi := Q_1 B_1 \dots Q_m B_m.\ \phi$ be a QBF. A set $A = \{l_1, \dots, l_n\}$ of *assumptions* is an assignment such that every assigned variable is from the leftmost block B_1: $\forall l_i \in A : var(l_i) \in B_1$.

- Solve the QBF ψ under assumptions A: solve $\psi[A]$.
- Necessary for correctness: restriction to variables from leftmost block B_1.

Implementation of Assumptions in DepQBF :

- Inspired by (incremental) SAT solving under assumptions as in MiniSAT [ES03].
- All information learned under assumptions can be kept across different solver calls.
- Similar to incremental solving by QuBE (bounded model checking of partial designs) [MMLB12] and incremental solving by DepQBF [LE14].
- MPIDepQBF: search-space partitioning by assumptions.

QBF Solving under Assumptions

Definition

Let $\psi := Q_1 B_1 \dots Q_m B_m.\ \phi$ be a QBF. A set $A = \{l_1, \dots, l_n\}$ of *assumptions* is an assignment such that every assigned variable is from the leftmost block B_1:
$\forall l_i \in A : var(l_i) \in B_1$.

- Solve the QBF ψ under assumptions A: solve $\psi[A]$.
- Necessary for correctness: restriction to variables from leftmost block B_1.

Implementation of Assumptions in DepQBF :

- Inspired by (incremental) SAT solving under assumptions as in MiniSAT [ES03].
- All information learned under assumptions can be kept across different solver calls.
- Similar to incremental solving by QuBE (bounded model checking of partial designs) [MMLB12] and incremental solving by DepQBF [LE14].
- MPIDepQBF: search-space partitioning by assumptions.

MPIDepQBF at a Glance

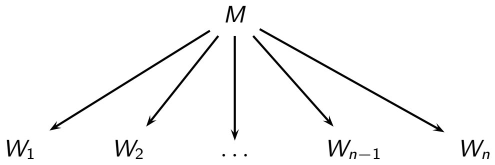
M

W_1 W_2 $\ldots$ W_{n-1} W_n

Framework:

- Coordination of master and worker processes.
- MPI-based, written in OCaml.
- Originated from experiments with reduction finding [JK13].
- Open source: `http://toss.sourceforge.net/develop.html`.

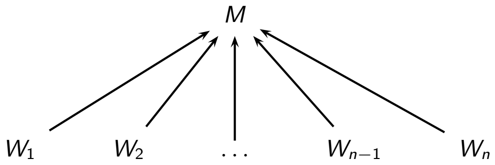
MPIDepQBF at a Glance



Master:

- Search space partitioning by assumptions.
- Assumptions: fixed variable assignments sent to the workers, including a timeout.
- Combines results obtained by workers, further partitioning.
- Similar to PaQuBE, but uses a different partitioning strategy.

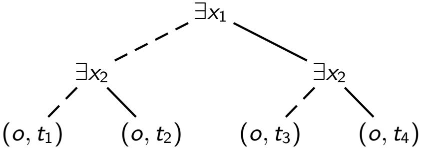
MPIDepQBF at a Glance



Workers:

- Solve the formula under assumptions received from master using DepQBF.
- Timeout: master may send same problem to worker with an increased timeout.
- No communication among workers, no global sharing of learned clauses and cubes.
- Worker keeps all learned clauses and cubes locally across different calls of DepQBF.

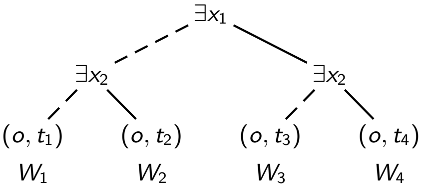
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



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

MPIDepQBF by Example

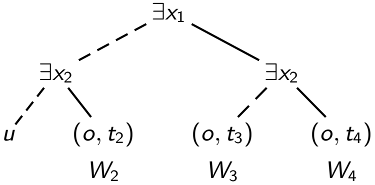
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



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 $\psi[x_1, x_2]$.

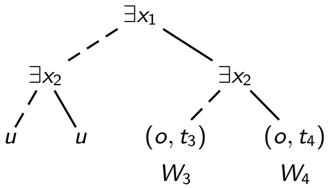
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



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

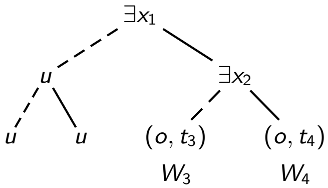
MPIDepQBF by Example

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



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

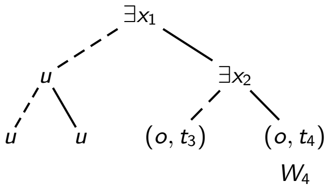
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



Since $\psi[\neg x_1, \neg x_2]$ and $\psi[\neg x_1, x_2]$ are unsatisfiable, the subcase $\psi[\neg x_1]$ is unsatisfiable.

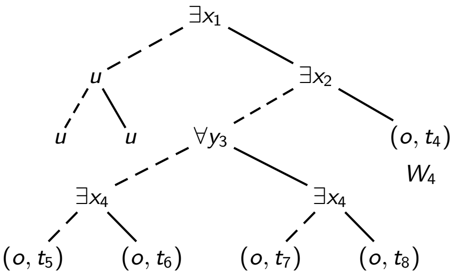
MPIDepQBF by Example

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.

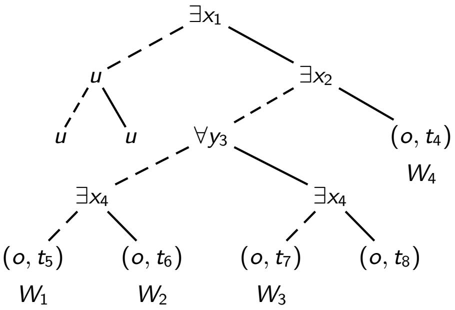
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \dots \phi$.



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

MPIDepQBF by Example

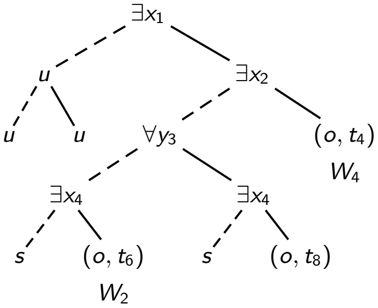
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



Assign open subcases to idle workers W_1, W_2, and W_3 by sending 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 $\psi[x_1, \neg x_2, y_3, \neg x_4]$.

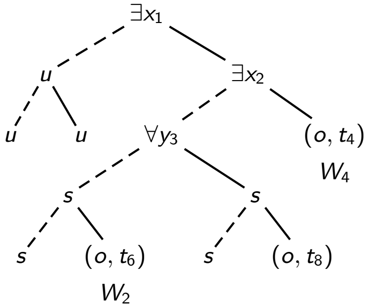
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



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

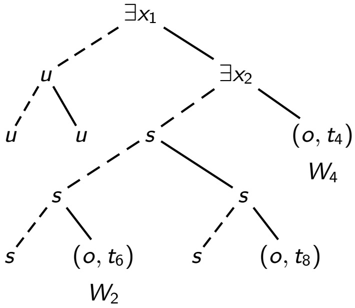
MPIDepQBF by Example

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



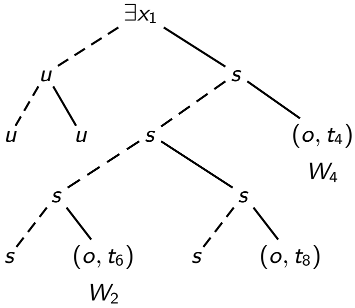
Subcases $\psi[x_1, \neg x_2, \neg y_3]$ and $\psi[x_1, \neg x_2, y_3]$ are satisfiable.

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



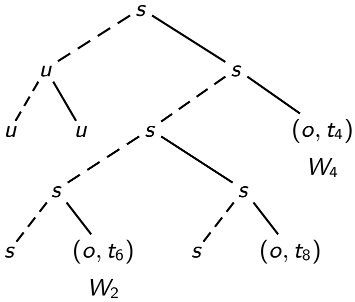
Subcase $\psi[x_1, \neg x_2]$ is satisfiable.

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



Subcase $\psi[x_1]$ is satisfiable.

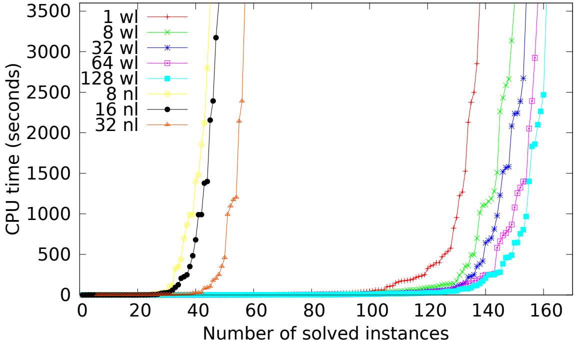
PCNF $\psi := \exists x_1, x_2 \forall y_3 \exists x_4 \ldots \phi$.



Finally, ψ is satisfiable.

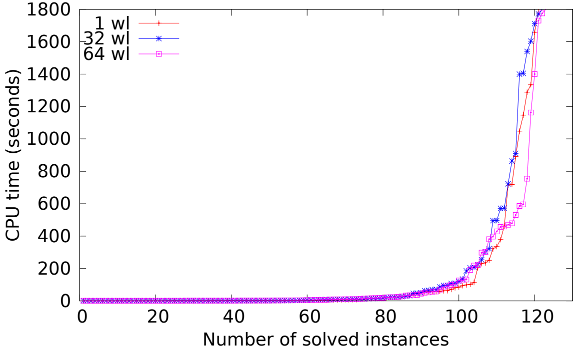
Experiments (1/5)

- Benchmarks: QBFEVAL 2012 Second Round *with* preprocessing by Bloqqer.
- Experiments on Tsubame supercomputer: 8-core 2.93 GHz Xeon 5670 with 30 GB memory per node, 3600s timeout.



- Clause/cube learning is crucial: with (wl) and without (nl) learning.

- Benchmarks: QBFEVAL 2012 Second Round *without* preprocessing by Bloqqer.



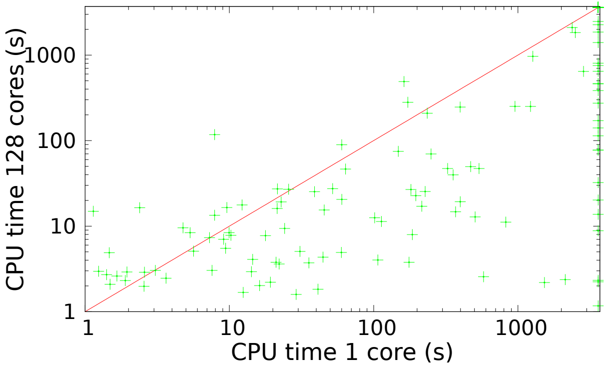
- Preprocessing is crucial.

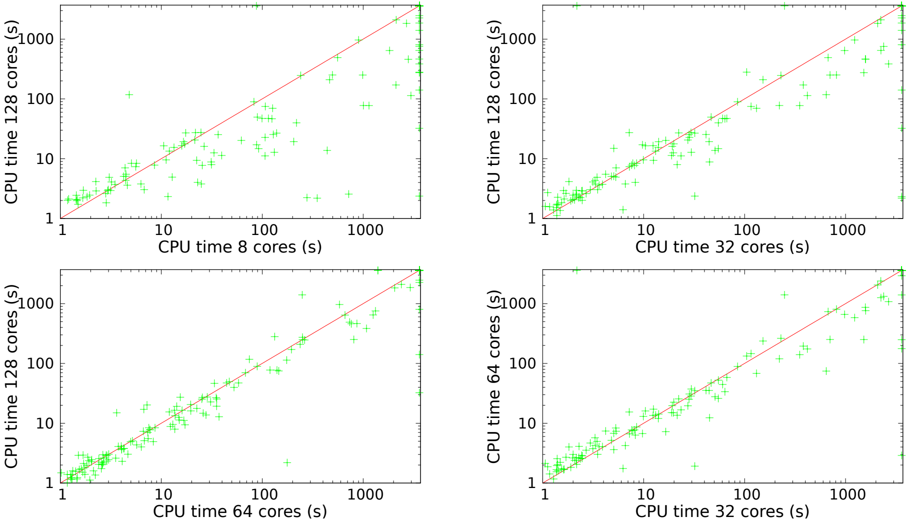
eval12r2-bloqqer (276 formulas)			
# cores	*solved*	*unsatisfiable*	*satisfiable*
1	137	68	69
8	149	77	72
16	150	78	72
32	153	81	72
64	157	84	73
128	160	86	74

- Number of formulas solved when using $x := 1, 8, 16, 32, 64, 128$ cores.

eval12r2-bloqqer (276 formulas)			
# cores (x)	*# solved both x/128*	*avg time (s) x cores*	*avg time (s) 128 cores*
1	137	168.11	62.26
8	148	180.64	64.03
16	149	154.44	76.26
32	151	163.74	79.46
64	155	122.96	98.47

- Run times on formulas solved by both $x := 1, 8, 16, 32, 64$ cores and 128 cores.


CPU time 128 cores (s)
CPU time 1 core (s)
1
10
100
1000
1
10
100
1000



CPU time 128 cores (s)
CPU time 8 cores (s)
CPU time 128 cores (s)
CPU time 32 cores (s)
CPU time 128 cores (s)
CPU time 64 cores (s)
CPU time 64 cores (s)
CPU time 32 cores (s)
1
10
100
1000

Conclusion

MPIDepQBF:

- Search-space partitioning by assumptions.
- Master: schedules workers and maintains a search tree.
- No global sharing of learned clauses/cubes: workers keep learned information locally.
- Promising experimental results (also on desktop computers).

Future Work:

- Strategies to share learned clauses/cubes.
- Generation of proofs and certificates.

Niklas Eén and Niklas Sörensson.
Temporal Induction by Incremental SAT Solving.
Electr. Notes Theor. Comput. Sci., 89(4):543–560, 2003.

R. Feldmann, B. Monien, and S. Schamberger.
A Distributed Algorithm to Evaluate Quantified Boolean Formulae.
In *AAAI/IAAI*, pages 285–290. AAAI Press / The MIT Press, 2000.

Marijn Heule, Oliver Kullmann, Siert Wieringa, and Armin Biere.
Cube and conquer: Guiding cdcl sat solvers by lookaheads.
In Kerstin Eder, João Lourenço, and Onn Shehory, editors, *Haifa Verification Conference*, volume 7261 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2011.

Charles Jordan and Lukasz Kaiser.
Experiments with Reduction Finding.
In Matti Järvisalo and Allen Van Gelder, editors, *SAT*, volume 7962 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2013.

F. Lonsing and U. Egly.
Incremental QBF solving.
In *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 2014. Proceedings*, Lecture Notes in Computer Science. Springer, 2014.
To appear.

M. D. T. Lewis, P. Marin, T. Schubert, M. Narizzano, B. Becker, and E. Giunchiglia.
PaQuBE: Distributed QBF Solving with Advanced Knowledge Sharing.
In Oliver Kullmann, editor, *SAT*, volume 5584 of *LNCS*, pages 509–523. Springer, 2009.

Matthew Lewis, Tobias Schubert, and Bernd Becker.
QMiraXT – a multithreaded QBF solver.
In *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV)*, 2009.

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.

Paolo Marin, Christian Miller, Matthew D. T. Lewis, and Bernd Becker.
Verification of Partial Designs using Incremental QBF Solving.
In Wolfgang Rosenstiel and Lothar Thiele, editors, *DATE*, pages 623–628. IEEE, 2012.