Efficient Clause Learning for Quantified Boolean Formulas via QBF Pseudo Unit Propagation

Florian Lonsing[1] Uwe Egly[1] Allen Van Gelder[2]

[1]Vienna University of Technology
http://www.kr.tuwien.ac.at/staff/{egly,lonsing}

[2]University of California at Santa Cruz, USA
http://www.cse.ucsc.edu/~avg

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

Conflict-Driven Clause Learning (CDCL): [SS96]

- Crucial for the performance of modern SAT solvers.
- Resolution proofs, trimming the search space.
- Extensions of CDCL for SAT to QBF: QCDCL.

Traditional QCDCL for QBF: [ZM02, GNT02, GNT06, Let02]

- Like CDCL is based on resolution, QCDCL is based on Q-resolution.
- Q-resolution derivation of the clause to be learned.
- Tautological resolvents must be avoided explicitly.

Problem:

- Common approach to avoiding tautologies in traditional QCDCL has an exponential worst case [VG12].
- The derivation of a single learned clause might have an exponential number of intermediate resolvents.

Overview (1/2)

Conflict-Driven Clause Learning (CDCL): [SS96]

- Crucial for the performance of modern SAT solvers.
- Resolution proofs, trimming the search space.
- Extensions of CDCL for SAT to QBF: QCDCL.

Traditional QCDCL for QBF: [ZM02, GNT02, GNT06, Let02]

- Like CDCL is based on resolution, QCDCL is based on Q-resolution.
- Q-resolution derivation of the clause to be learned.
- Tautological resolvents must be avoided explicitly.

Problem:

- Common approach to avoiding tautologies in traditional QCDCL has an exponential worst case [VG12].
- The derivation of a single learned clause might have an exponential number of intermediate resolvents.

Overview (2/2)

Our Work: efficient polynomial time procedure for QCDCL.

- QCDCL based on *QBF Pseudo Unit Propagation (QPUP)* [VG12]: carefully select the order of resolution steps in QCDCL to avoid tautologies.
- Learn a single non-tautological clause in polynomial time.
- QPUP-based QCDCL is compatible with other approaches (e.g. Alexandra's talk).
- Implementation in the search-based QBF solver DepQBF.

Quantified Boolean Formulae (QBF)

Syntax

- Prenex CNF: quantifier-free CNF over quantified Boolean variables.
- PCNF $\psi := Q_1 x_1 \dots Q_n x_n. \ \phi$, where $Q_i \in \{\exists, \forall\}$, no free variables.
- $Q_i x_i \leq Q_{i+1} x_{i+1}$: variables are linearly ordered.

Example

A CNF: $(x \vee \neg y) \wedge (\neg x \vee y)$, and a PCNF: $\forall x \exists y. \ (x \vee \neg y) \wedge (\neg x \vee y)$.

Search-based QBF Solving with Clause Learning:

- Implicitly enumerate paths in a semantic tree by recursive variable instantiation.
- Terminology "QCDCL": conflict-driven clause learning (CDCL) for QBF.
- Learn clauses at unsatisfiable (i.e. conflicting) branches in the search tree.
- Like CDCL in SAT: QCDCL is based on resolution for QBF.

Q-Resolution:

- Combination of universal reduction and propositional resolution.
- Sound and refutational-complete proof system for QBF: Q-resolution proofs.

Definition ([BKF95])

Given a clause C, *universal reduction (UR)* on C produces the clause

$$UR(C) := C \setminus \{l \in L_{\forall}(C) \mid \forall l' \in L_{\exists}(C) : var(l') < var(l)\},$$

where $<$ is the linear variable ordering given by the quantifier prefix.

- Universal reduction deletes trailing universal literals from clauses.

Definition ([BKF95])

- Let C_1, C_2 be non-tautological clauses where $v \in C_1, \neg v \in C_2$ for an $\exists$-variable v.
- *Tentative Q-resolvent* of C_1 and C_2: $C_1 \otimes C_2 := (UR(C_1) \cup UR(C_2)) \setminus \{v, \neg v\}$.
- If $\{x, \neg x\} \subseteq C_1 \otimes C_2$ for some variable x, then no Q-resolvent exists.
- Otherwise, the non-tautological *Q-resolvent* is $C := UR(C_1 \otimes C_2)$.

Boolean Constraint Propagation for QBF (QBCP)

- Generate assignments by assumptions, unit clause rule, **universal reduction (UR).**
- Like BCP for SAT: antecedent clauses and implication graphs.
- Like CDCL for SAT: QCDCL is based on the implication graph given by QBCP.

Example (assignments, implication graphs)

```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

Implication graph:

Assignment $A := \{\}$.

- Assumption: $A := A \cup \{1\}$.
- Clause (-1 2) is unit under A
  $A := A \cup \{2\} = \{1, 2\}$
  $ante(2) :=$ (-1 2)
- Clause (3 5 -2) is unit under A and UR.
  $A := A \cup \{3\} = \{1, 2, 3\}$
  $ante(3) :=$ (3 5 -2)
- Clause (4 5 -2) is unit under A and UR.
  $A := A \cup \{4\} = \{1, 2, 3, 4\}$
  $ante(4) :=$ (4 5 -2)
- Clause (-3 -4) is conflicting under A.
  $ante(\emptyset) :=$ (-3 -4)

- Generate assignments by assumptions, unit clause rule, **universal reduction (UR).**
- Like BCP for SAT: antecedent clauses and implication graphs.
- Like CDCL for SAT: QCDCL is based on the implication graph given by QBCP.

Example (assignments, implication graphs)

```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

Implication graph:

1

Assignment $A := \{\}$.

- Assumption: $A := A \cup \{1\}$.
- Clause (-1 2) is unit under A
  $A := A \cup \{2\} = \{1, 2\}$
  $ante(2) :=$ (-1 2)
- Clause (3 5 -2) is unit under A and UR.
  $A := A \cup \{3\} = \{1, 2, 3\}$
  $ante(3) :=$ (3 5 -2)
- Clause (4 5 -2) is unit under A and UR.
  $A := A \cup \{4\} = \{1, 2, 3, 4\}$
  $ante(4) :=$ (4 5 -2)
- Clause (-3 -4) is conflicting under A.
  $ante(\emptyset) :=$ (-3 -4)

Boolean Constraint Propagation for QBF (QBCP)

- Generate assignments by assumptions, unit clause rule, **universal reduction (UR).**
- Like BCP for SAT: antecedent clauses and implication graphs.
- Like CDCL for SAT: QCDCL is based on the implication graph given by QBCP.

Example (assignments, implication graphs)

```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

Implication graph:

$1 \longrightarrow 2$

Assignment $A := \{\}$.

- Assumption: $A := A \cup \{1\}$.
- Clause (-1 2) is unit under A
  $A := A \cup \{2\} = \{1, 2\}$
  $ante(2) :=$ (-1 2)
- Clause (3 5 -2) is unit under A and UR.
  $A := A \cup \{3\} = \{1, 2, 3\}$
  $ante(3) :=$ (3 5 -2)
- Clause (4 5 -2) is unit under A and UR.
  $A := A \cup \{4\} = \{1, 2, 3, 4\}$
  $ante(4) :=$ (4 5 -2)
- Clause (-3 -4) is conflicting under A.
  $ante(\emptyset) :=$ (-3 -4)

Boolean Constraint Propagation for QBF (QBCP)

- Generate assignments by assumptions, unit clause rule, **universal reduction (UR).**
- Like BCP for SAT: antecedent clauses and implication graphs.
- Like CDCL for SAT: QCDCL is based on the implication graph given by QBCP.

Example (assignments, implication graphs)

```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

Implication graph:

$1 \longrightarrow 2 \longrightarrow 3$

Assignment $A := \{\}$.

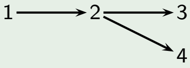
- Assumption: $A := A \cup \{1\}$.
- Clause (-1 2) is unit under A
  $A := A \cup \{2\} = \{1, 2\}$
  $ante(2) :=$ (-1 2)
- Clause (3 5 -2) is unit under A and UR.
  $A := A \cup \{3\} = \{1, 2, 3\}$
  $ante(3) :=$ (3 5 -2)
- Clause (4 5 -2) is unit under A and UR.
  $A := A \cup \{4\} = \{1, 2, 3, 4\}$
  $ante(4) :=$ (4 5 -2)
- Clause (-3 -4) is conflicting under A.
  $ante(\emptyset) :=$ (-3 -4)

- Generate assignments by assumptions, unit clause rule, **universal reduction (UR).**
- Like BCP for SAT: antecedent clauses and implication graphs.
- Like CDCL for SAT: QCDCL is based on the implication graph given by QBCP.

Example (assignments, implication graphs)

```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

Implication graph:



Assignment $A := \{\}$.

- Assumption: $A := A \cup \{1\}$.
- Clause (-1 2) is unit under A
  $A := A \cup \{2\} = \{1, 2\}$
  $ante(2) :=$ (-1 2)
- Clause (3 5 -2) is unit under A and UR.
  $A := A \cup \{3\} = \{1, 2, 3\}$
  $ante(3) :=$ (3 5 -2)
- Clause (4 5 -2) is unit under A and UR.
  $A := A \cup \{4\} = \{1, 2, 3, 4\}$
  $ante(4) :=$ (4 5 -2)
- Clause (-3 -4) is conflicting under A.
  $ante(\emptyset) :=$ (-3 -4)

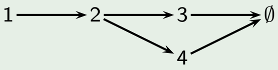
Boolean Constraint Propagation for QBF (QBCP)

- Generate assignments by assumptions, unit clause rule, **universal reduction (UR).**
- Like BCP for SAT: antecedent clauses and implication graphs.
- Like CDCL for SAT: QCDCL is based on the implication graph given by QBCP.

Example (assignments, implication graphs)

```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

Implication graph:



Assignment $A := \{\}$.

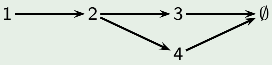
- Assumption: $A := A \cup \{1\}$.
- Clause (-1 2) is unit under A
  $A := A \cup \{2\} = \{1, 2\}$
  $ante(2) :=$ (-1 2)
- Clause (3 5 -2) is unit under A and UR.
  $A := A \cup \{3\} = \{1, 2, 3\}$
  $ante(3) :=$ (3 5 -2)
- Clause (4 5 -2) is unit under A and UR.
  $A := A \cup \{4\} = \{1, 2, 3, 4\}$
  $ante(4) :=$ (4 5 -2)
- Clause (-3 -4) is conflicting under A.
  $ante(\emptyset) :=$ (-3 -4)

- Start at conflicting clause, resolve on existential variables *in reverse assignment order* until the resolvent is asserting (i.e. will be unit after backtracking).
- Resolve on existential variables which were assigned as unit literals, using clauses (i.e. antecedents) which became unit during QBCP.
- Tautological resolvents might occur but must be avoided by "resolving around": $\Rightarrow$ deviate from strict reverse assignment order [GNT06].
- Worst case exponential number (in $|IG|$) of intermediate resolvents [VG12].

Example (continued)

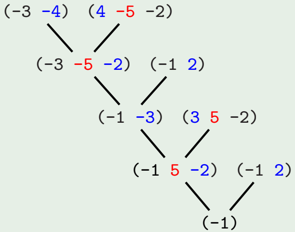
```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

Clause (-3 -4) conflicting:



Assignment $A = \{1, 2, 3, 4\}$
Assignment order: 1, 2, 3, 4
Resolve on: 4, 2, 3, 2.

Derivation of learned clause (-1):

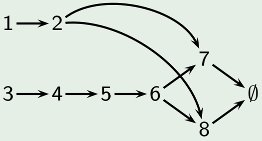


QBF Pseudo Unit Propagation (QPUP): [VG12]

- Basic idea: given an implication graph (IG), associate the conflict node $\emptyset$ and each variable x assigned by the unit literal rule with a "QPUP clause" $qpup(x)$.
- Walking through the entire IG in assignment ordering, compute $qpup(x)$ by resolving $ante(x)$ with already computed $qpup(y)$ s.t. $\neg y \in ante(x)$.
- Resolve in assignment ordering: tautologies cannot occur by construction.
  - Compare: traditional QCDCL resolves in reverse assignment ordering.
- Finally, the non-tautological and asserting QPUP clause $qpup(\emptyset)$ related to the conflict node $\emptyset$ can be learned.

Example (to be continued)

Assumptions: 1, 3
Assignment order: 1, 2, . . . , 8.

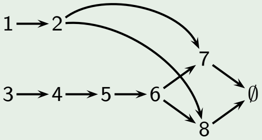



```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

Towards Efficient QCDCL (2/3)

Example (continued; computing QPUP clauses)

Assumptions: 1, 3
Assignment order: 1, 2,. . . , 8.




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

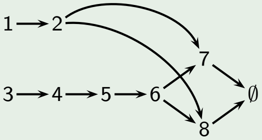
qpup(2) = (-1 2)
qpup(4) = (-3 4)
qpup(5) = (-3 5)
qpup(6) = (-3 6)
qpup(7) = (-1 -3 7)
qpup(8) = (-1 -3 8)
qpup(∅) = (-1 -3)

The clause *qpup*(∅) = (-1 -3) is non-tautological and asserting and can be learned.

Towards Efficient QCDCL (2/3)

Example (continued; computing QPUP clauses)

Assumptions: 1, 3
Assignment order: 1, 2,. . . , 8.




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

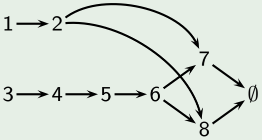
qpup(2) = (-1 2)
qpup(4) = (-3 4)
qpup(5) = (-3 5)
qpup(6) = (-3 6)
qpup(7) = (-1 -3 7)
qpup(8) = (-1 -3 8)
qpup(∅) = (-1 -3)

qpup(2) := *ante*(2) = (-1 2)

The clause *qpup*(∅) = (-1 -3) is non-tautological and asserting and can be learned.

Example (continued; computing QPUP clauses)

Assumptions: 1, 3
Assignment order: 1, 2,. . . , 8.




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

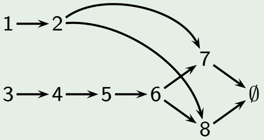
qpup(2) = (-1 2)
qpup(4) = (-3 4)
qpup(5) = (-3 5)
qpup(6) = (-3 6)
qpup(7) = (-1 -3 7)
qpup(8) = (-1 -3 8)
qpup(∅) = (-1 -3)

qpup(4) := *ante*(4) = (-3 4)

The clause *qpup*(∅) = (-1 -3) is non-tautological and asserting and can be learned.

Example (continued; computing QPUP clauses)

Assumptions: 1, 3
Assignment order: 1, 2,. . . , 8.




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

qpup(2) = (-1 2)
qpup(4) = (-3 4)
qpup(5) = (-3 5)
qpup(6) = (-3 6)
qpup(7) = (-1 -3 7)
qpup(8) = (-1 -3 8)
qpup(∅) = (-1 -3)

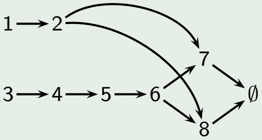
ante(5) = (-4 5) (-3 4) = *qpup*(4)

(-3 5)

The clause *qpup*(∅) = (-1 -3) is non-tautological and asserting and can be learned.

Example (continued; computing QPUP clauses)

Assumptions: 1, 3
Assignment order: 1, 2,. . . , 8.




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

qpup(2) = (-1 2)
qpup(4) = (-3 4)
qpup(5) = (-3 5)
qpup(6) = (-3 6)
qpup(7) = (-1 -3 7)
qpup(8) = (-1 -3 8)
qpup(∅) = (-1 -3)

ante(6) = (-5 6) (-3 5) = *qpup*(5)

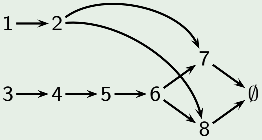
(-3 6)

The clause *qpup*(∅) = (-1 -3) is non-tautological and asserting and can be learned.

Towards Efficient QCDCL (2/3)

Example (continued; computing QPUP clauses)

Assumptions: 1, 3
Assignment order: 1, 2,. . . , 8.




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

qpup(2) = (-1 2)
qpup(4) = (-3 4)
qpup(5) = (-3 5)
qpup(6) = (-3 6)
qpup(7) = (-1 -3 7)
qpup(8) = (-1 -3 8)
qpup(∅) = (-1 -3)

ante(7) = (7 10 -2 -6) (-1 2) = *qpup*(2)

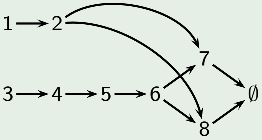
(-1 7 10 -6) (-3 6) = *qpup*(6)

(-1 -3 7)

The clause *qpup*(∅) = (-1 -3) is non-tautological and asserting and can be learned.

Example (continued; computing QPUP clauses)

Assumptions: 1, 3
Assignment order: 1, 2,. . . , 8.




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

qpup(2) = (-1 2)
qpup(4) = (-3 4)
qpup(5) = (-3 5)
qpup(6) = (-3 6)
qpup(7) = (-1 -3 7)
qpup(8) = (-1 -3 8)
qpup(∅) = (-1 -3)

ante(8) = (8 -10 -2 -6)   (-1 2) = *qpup*(2)

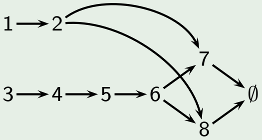
(-1 8 -10 -6)   (-3 6) = *qpup*(6)

(-1 -3 8)

The clause *qpup*(∅) = (-1 -3) is non-tautological and asserting and can be learned.

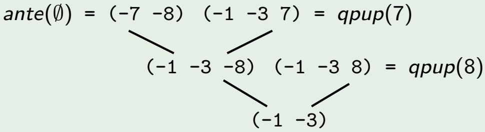
Example (continued; computing QPUP clauses)

Assumptions: 1, 3
Assignment order: 1, 2,. . . , 8.




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

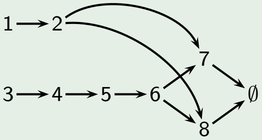
qpup(2) = (-1 2)
qpup(4) = (-3 4)
qpup(5) = (-3 5)
qpup(6) = (-3 6)
qpup(7) = (-1 -3 7)
qpup(8) = (-1 -3 8)
qpup(∅) = (-1 -3)



The clause *qpup*(∅) = (-1 -3) is non-tautological and asserting and can be learned.

Example (continued; computing QPUP clauses)

Assumptions: 1, 3
Assignment order: 1, 2,. . . , 8.




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

qpup(2) = (-1 2)
qpup(4) = (-3 4)
qpup(5) = (-3 5)
qpup(6) = (-3 6)
qpup(7) = (-1 -3 7)
qpup(8) = (-1 -3 8)
qpup(∅) = (-1 -3)

The clause *qpup*(∅) = (-1 -3) is non-tautological and asserting and can be learned.

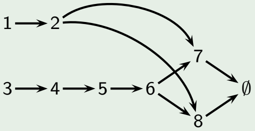
Problem:

- Computing QPUP clauses for every $n \in IG$: total $|IG|$ resolution steps.
- Traversal starts at assumption nodes $\Rightarrow$ full traversal, prohibitive at each conflict.
- Goal: find alternative start points closer to the conflict node $\emptyset$.

Unique Implication Points (UIPs):

- Nodes in the implication graph which are on every path from the most recent assumption to the conflict node $\emptyset$.
- Comprehensive theory in SAT CDCL [SLM09].
- A UIP is a good candidate as a start point to compute QPUP clauses.

Example (continued)



- Node 6 is the first UIP (i.e. closest to $\emptyset$).
- Node 5 is the second UIP.
- Node 4 is the third UIP.
- Node 3 is the fourth UIP.

Efficient QPUP-based QCDCL (1/2)

Two-Phase Algorithm:

- Phase 1: starting at the conflict node $\emptyset$, walk back through the implication graph in reverse assignment order to find suitable start points.
  - Focus on finding UIPs.
  - In general, a single UIP as a start point is not enough.
  - At the latest, phase 1 terminates when reaching the assumption nodes.
- Phase 2: compute the QPUP clauses $qpup(x)$ for all nodes x reachable when walking from the start points found in phase 1 towards the conflict node $\emptyset$.
  - Unlike in traditional QCDCL, here resolutions are done in assignment order.

Goal:

- The non-tautological and asserting QPUP clause $qpup(\emptyset)$ of the conflict node $\emptyset$ computed in phase two will be learned.
- Challenge: what nodes are suitable start points?

Efficient QPUP-based QCDCL (1/2)

Two-Phase Algorithm:

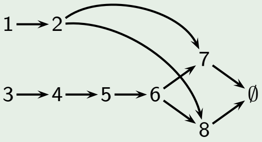
- Phase 1: starting at the conflict node $\emptyset$, walk back through the implication graph in reverse assignment order to find suitable start points.
  - Focus on finding UIPs.
  - In general, a single UIP as a start point is not enough.
  - At the latest, phase 1 terminates when reaching the assumption nodes.
- Phase 2: compute the QPUP clauses $qpup(x)$ for all nodes x reachable when walking from the start points found in phase 1 towards the conflict node $\emptyset$.
  - Unlike in traditional QCDCL, here resolutions are done in assignment order.

Goal:

- The non-tautological and asserting QPUP clause $qpup(\emptyset)$ of the conflict node $\emptyset$ computed in phase two will be learned.
- Challenge: what nodes are suitable start points?

Efficient QPUP-based QCDCL (2/2)

Example (computing QPUP clauses from start points)




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

Node 6 is the 1-UIP, $\{7, 8, \emptyset\}$ reachable, *qpup*($\emptyset$) = (10 -10 -2 -6) tautological.

Node 5 is the 2-UIP, $\{6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-5 10 -10 -2).

Node 4 is the 3-UIP, $\{5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-4 10 -10 -2).

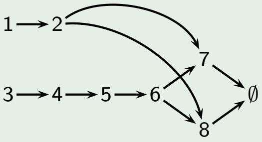
Node 3 is the 4-UIP, $\{4, 5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-3 10 -10 -2).
⇒ **impossible to use a UIP as the single start point.**

Observe:
Node 5 is the 2-UIP, but *qpup*($\emptyset$) = (-5 10 -10 -2) is tautological.
⇒ must eventually resolve on variable 2 to avoid tautology.

Nodes $\{1, 5\}$ are suitable start points:
$\{2, 6, 7, 8, \emptyset\}$ reachable and *qpup*($\emptyset$) = (-1 -5)

Compare:
Using the assumptions $\{1, 3\}$ as trivial start points produces *qpup*($\emptyset$) = (-1 -3).

Example (computing QPUP clauses from start points)




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

Node 6 is the 1-UIP, $\{7, 8, \emptyset\}$ reachable, *qpup*($\emptyset$) = (10 -10 -2 -6) tautological.

Node 5 is the 2-UIP, $\{6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-5 10 -10 -2).

Node 4 is the 3-UIP, $\{5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-4 10 -10 -2).

Node 3 is the 4-UIP, $\{4, 5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-3 10 -10 -2).
⇒ **impossible to use a UIP as the single start point.**

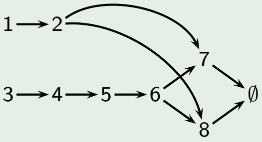
Observe:
Node 5 is the 2-UIP, but *qpup*($\emptyset$) = (-5 10 -10 -2) is tautological.
⇒ must eventually resolve on variable 2 to avoid tautology.

Nodes $\{1, 5\}$ are suitable start points:
$\{2, 6, 7, 8, \emptyset\}$ reachable and *qpup*($\emptyset$) = (-1 -5)

Compare:
Using the assumptions $\{1, 3\}$ as trivial start points produces *qpup*($\emptyset$) = (-1 -3).

Efficient QPUP-based QCDCL (2/2)

Example (computing QPUP clauses from start points)




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

Node 6 is the 1-UIP, $\{7, 8, \emptyset\}$ reachable, *qpup*(∅) = (10 -10 -2 -6) tautological.

Node 5 is the 2-UIP, $\{6, 7, 8, \emptyset\}$ reachable, but *qpup*(∅) = (-5 10 -10 -2).

Node 4 is the 3-UIP, $\{5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*(∅) = (-4 10 -10 -2).

Node 3 is the 4-UIP, $\{4, 5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*(∅) = (-3 10 -10 -2).
⇒ **impossible to use a UIP as the single start point.**

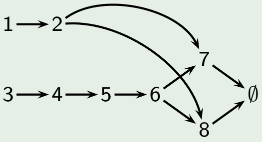
Observe:
Node 5 is the 2-UIP, but *qpup*(∅) = (-5 10 -10 -2) is tautological.
⇒ must eventually resolve on variable 2 to avoid tautology.

Nodes $\{1, 5\}$ are suitable start points:
$\{2, 6, 7, 8, \emptyset\}$ reachable and *qpup*(∅) = (-1 -5)

Compare:
Using the assumptions $\{1, 3\}$ as trivial start points produces *qpup*(∅) = (-1 -3).

Efficient QPUP-based QCDCL (2/2)

Example (computing QPUP clauses from start points)




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

Node 6 is the 1-UIP, $\{7, 8, \emptyset\}$ reachable, $qpup(\emptyset)$ = (10 -10 -2 -6) tautological.

Node 5 is the 2-UIP, $\{6, 7, 8, \emptyset\}$ reachable, but $qpup(\emptyset)$ = (-5 10 -10 -2).

Node 4 is the 3-UIP, $\{5, 6, 7, 8, \emptyset\}$ reachable, but $qpup(\emptyset)$ = (-4 10 -10 -2).

Node 3 is the 4-UIP, $\{4, 5, 6, 7, 8, \emptyset\}$ reachable, but $qpup(\emptyset)$ = (-3 10 -10 -2).

⇒ **impossible to use a UIP as the single start point.**

Observe:

Node 5 is the 2-UIP, but $qpup(\emptyset)$ = (-5 10 -10 -2) is tautological.

⇒ must eventually resolve on variable 2 to avoid tautology.

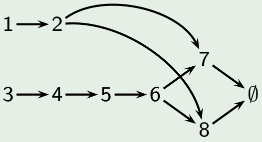
Nodes $\{1, 5\}$ are suitable start points:

$\{2, 6, 7, 8, \emptyset\}$ reachable and $qpup(\emptyset)$ = (-1 -5)

Compare:

Using the assumptions $\{1, 3\}$ as trivial start points produces $qpup(\emptyset)$ = (-1 -3).

Example (computing QPUP clauses from start points)




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

Node 6 is the 1-UIP, $\{7, 8, \emptyset\}$ reachable, *qpup*(∅) = (10 -10 -2 -6) tautological.

Node 5 is the 2-UIP, $\{6, 7, 8, \emptyset\}$ reachable, but *qpup*(∅) = (-5 10 -10 -2).

Node 4 is the 3-UIP, $\{5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*(∅) = (-4 10 -10 -2).

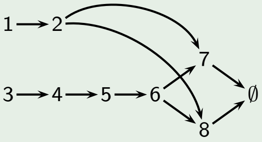
Node 3 is the 4-UIP, $\{4, 5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*(∅) = (-3 10 -10 -2).
⇒ **impossible to use a UIP as the single start point.**

Observe:
Node 5 is the 2-UIP, but *qpup*(∅) = (-5 10 -10 -2) is tautological.
⇒ must eventually resolve on variable 2 to avoid tautology.

Nodes $\{1, 5\}$ are suitable start points:
$\{2, 6, 7, 8, \emptyset\}$ reachable and *qpup*(∅) = (-1 -5)

Compare:
Using the assumptions $\{1, 3\}$ as trivial start points produces *qpup*(∅) = (-1 -3).

Example (computing QPUP clauses from start points)




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

Node 6 is the 1-UIP, $\{7, 8, \emptyset\}$ reachable, *qpup*(∅) = (10 -10 -2 -6) tautological.

Node 5 is the 2-UIP, $\{6, 7, 8, \emptyset\}$ reachable, but *qpup*(∅) = (-5 10 -10 -2).

Node 4 is the 3-UIP, $\{5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*(∅) = (-4 10 -10 -2).

Node 3 is the 4-UIP, $\{4, 5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*(∅) = (-3 10 -10 -2).
⇒ **impossible to use a UIP as the single start point.**

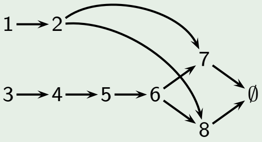
Observe:
Node 5 is the 2-UIP, but *qpup*(∅) = (-5 10 -10 -2) is tautological.
⇒ must eventually resolve on variable 2 to avoid tautology.

Nodes $\{1, 5\}$ are suitable start points:
$\{2, 6, 7, 8, \emptyset\}$ reachable and *qpup*(∅) = (-1 -5)

Compare:
Using the assumptions $\{1, 3\}$ as trivial start points produces *qpup*(∅) = (-1 -3).

Efficient QPUP-based QCDCL (2/2)

Example (computing QPUP clauses from start points)




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

Node 6 is the 1-UIP, $\{7, 8, \emptyset\}$ reachable, *qpup*($\emptyset$) = (10 -10 -2 -6) tautological.

Node 5 is the 2-UIP, $\{6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-5 10 -10 -2).

Node 4 is the 3-UIP, $\{5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-4 10 -10 -2).

Node 3 is the 4-UIP, $\{4, 5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-3 10 -10 -2).
⇒ **impossible to use a UIP as the single start point.**

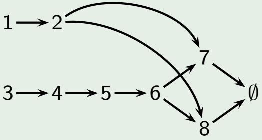
Observe:
Node 5 is the 2-UIP, but *qpup*($\emptyset$) = (-5 10 -10 -2) is tautological.
⇒ must eventually resolve on variable 2 to avoid tautology.

Nodes $\{1, 5\}$ are suitable start points:
$\{2, 6, 7, 8, \emptyset\}$ reachable and *qpup*($\emptyset$) = (-1 -5)

Compare:
Using the assumptions $\{1, 3\}$ as trivial start points produces *qpup*($\emptyset$) = (-1 -3).

Efficient QPUP-based QCDCL (2/2)

Example (computing QPUP clauses from start points)




```
p cnf 10 7
e 1 3 4 5 7 8 0
a 10 0
e 2 6 0
(-1 2),
(-3 4),(-4 5),(-5 6),
(7 10 -2 -6),(8 -10 -2 -6),
(-7 -8)
```

Node 6 is the 1-UIP, $\{7, 8, \emptyset\}$ reachable, *qpup*($\emptyset$) = (10 -10 -2 -6) tautological.

Node 5 is the 2-UIP, $\{6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-5 10 -10 -2).

Node 4 is the 3-UIP, $\{5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-4 10 -10 -2).

Node 3 is the 4-UIP, $\{4, 5, 6, 7, 8, \emptyset\}$ reachable, but *qpup*($\emptyset$) = (-3 10 -10 -2).

$\Rightarrow$ **impossible to use a UIP as the single start point.**

Observe:

Node 5 is the 2-UIP, but *qpup*($\emptyset$) = (-5 10 -10 -2) is tautological.

$\Rightarrow$ must eventually resolve on variable 2 to avoid tautology.

Nodes $\{1, 5\}$ are suitable start points:

$\{2, 6, 7, 8, \emptyset\}$ reachable and *qpup*($\emptyset$) = (-1 -5)

Compare:

Using the assumptions $\{1, 3\}$ as trivial start points produces *qpup*($\emptyset$) = (-1 -3).

Implementation:

- Search-based, clause-learning QBF solver DepQBF.
- Features: traditional QCDCL and QPUP-based QCDCL.
- Our implementation is more sophisticated than the procedure sketched before.
- No QPUP clauses are computed during the search for start points.

Example (formula class with exponential traditional QCDCL [VG12])

Each formula in this class can be decided by learning a single unit clause. The derivation of that learned clause by traditional QCDCL has an exponential number of resolution steps.

Size Parameter	1	2	3	4	5	6	7	8	9	10
Traditional QCDCL	6	14	30	62	126	254	510	1022	2046	4094
QPUP-based QCDCL	6	10	14	18	22	26	30	34	38	42

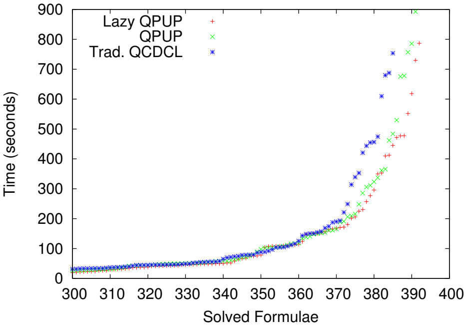
Table: number of resolutions in DepQBF to derive the single learned unit clause.

Benchmarks from Previous QBF Evaluations:

- Improvements with QPUP-based QCDCL.
- Lazy QPUP-based QCDCL: learn a clause without explicitly deriving it.
  - Conservatively predict the set literals definitely in the learned clause.
- Further experimental results: see the QBF Gallery 2013.

QBFEVAL'10 (568 formulas, no preprocessing)	
Lazy QPUP	393 (170 s, 223 u)
QPUP	392 (170 s, 222 u)
Trad. QCDCL	386 (167 s, 219 u)

- Instances solved (sat, unsat).
- Intel Xeon E5450, 3.00 GHz, timeout 900 seconds, 8 GB memory limit.



Conclusions

Traditional QCDCL for QBF:

- Based on implication graphs resulting from QBCP.
- Start at conflict node, resolve on variables in reverse assignment order.
- Tautologies must be avoided explicitly: exponential worst case.

QPUP-based QCDCL:

- Start at internal nodes of the implication graph, resolve on variables in assignment order working towards the conflict node.
- With the right set of start point, tautologies cannot occur by construction.
- For practical efficiency: finding start points close to the conflict node.
- Compatible with other approaches in search-based QBF solving.

Future Work:

- Procedural improvements.
- More detailed comparison of QCDCL variants (traditional, QPUP, lazy QPUP).

New version of DepQBF to be released: `http://lonsing.github.io/depqbf/`

Conclusions

Traditional QCDCL for QBF:

- Based on implication graphs resulting from QBCP.
- Start at conflict node, resolve on variables in reverse assignment order.
- Tautologies must be avoided explicitly: exponential worst case.

QPUP-based QCDCL:

- Start at internal nodes of the implication graph, resolve on variables in assignment order working towards the conflict node.
- With the right set of start point, tautologies cannot occur by construction.
- For practical efficiency: finding start points close to the conflict node.
- Compatible with other approaches in search-based QBF solving.

Future Work:

- Procedural improvements.
- More detailed comparison of QCDCL variants (traditional, QPUP, lazy QPUP).

New version of DepQBF to be released: `http://lonsing.github.com/depqbf/`

Conclusions

Traditional QCDCL for QBF:

- Based on implication graphs resulting from QBCP.
- Start at conflict node, resolve on variables in reverse assignment order.
- Tautologies must be avoided explicitly: exponential worst case.

QPUP-based QCDCL:

- Start at internal nodes of the implication graph, resolve on variables in assignment order working towards the conflict node.
- With the right set of start point, tautologies cannot occur by construction.
- For practical efficiency: finding start points close to the conflict node.
- Compatible with other approaches in search-based QBF solving.

Future Work:

- Procedural improvements.
- More detailed comparison of QCDCL variants (traditional, QPUP, lazy QPUP).

New version of DepQBF to be released: `http://lonsing.github.com/depqbf/`

H. Kleine Büning, M. Karpinski, and A. Flögel.
Resolution for Quantified Boolean Formulas.
Inf. Comput., 117(1):12–18, 1995.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Learning for Quantified Boolean Logic Satisfiability.
In *AAAI/IAAI*, pages 649–654, 2002.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas.
J. Artif. Intell. Res. (JAIR), 26:371–416, 2006.

R. Letz.
Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas.
In U. Egly and C. G. Fermüller, editors, *TABLEAUX*, volume 2381 of *LNCS*, pages 160–175. Springer, 2002.

J. P. Marques Silva, I. Lynce, and S. Malik.
Conflict-Driven Clause Learning SAT Solvers.
In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 131–153. IOS Press, 2009.

João P. Marques Silva and Karem A. Sakallah.
GRASP - a new search algorithm for satisfiability.
In *ICCAD*, pages 220–227, 1996.

Allen Van Gelder.
Contributions to the Theory of Practical Quantified Boolean Formula Solving.
In Michela Milano, editor, *CP*, volume 7514 of *LNCS*, pages 647–663. Springer, 2012.

L. Zhang and S. Malik.
Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation.
In P. Van Hentenryck, editor, *CP*, volume 2470 of *LNCS*, pages 200–215. Springer, 2002.