Should Decisions in QCDCL Follow Prefix Order?

Benjamin Böhm
Friedrich Schiller Universität Jena, Germany

Tomáš Peitl
TU Wien, Vienna, Austria

Olaf Beyersdorff
Friedrich Schiller Universität Jena, Germany

Abstract

Quantified conflict-driven clause learning (QCDCL) is one of the main solving approaches for quantified Boolean formulas (QBF). One of the differences between QCDCL and propositional CDCL is that QCDCL typically follows the prefix order of the QBF for making decisions.

We investigate an alternative model for QCDCL solving where decisions can be made in arbitrary order. The resulting system $\mathsf{QCDCL}^{\text{Any}}$ is still sound and terminating, but does not necessarily allow to always learn asserting clauses or cubes. To address this potential drawback, we additionally introduce two subsystems that guarantee to always learn asserting clauses ($\mathsf{QCDCL}^{\text{Uni-Any}}$) and asserting cubes ($\mathsf{QCDCL}^{\text{Exi-Any}}$), respectively.

We model all four approaches by formal proof systems and show that $\mathsf{QCDCL}^{\text{Uni-Any}}$ is exponentially better than QCDCL on false formulas, whereas $\mathsf{QCDCL}^{\text{Exi-Any}}$ is exponentially better than QCDCL on true QBFs. Technically, this involves constructing specific QBF families and showing lower and upper bounds in the respective proof systems.

We complement our theoretical study with some initial experiments that confirm our theoretical findings.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases QBF, CDCL, proof complexity, lower bounds

Digital Object Identifier 10.4230/LIPIcs.SAT.2022.11

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2022/040/`

Supplementary Material *Software (Source Code)*: `https://github.com/fslivovsky/qute` archived at `swh:1:dir:6363a9fc5093a36739e1ad5d8c59ef0fde5351ea`

Funding *Tomáš Peitl*: Austrian Science Fund FWF, grant no. J4361-N.
Olaf Beyersdorff: Carl-Zeiss Foundation and DFG grant BE 4209/3-1.

1 Introduction

SAT solving was revolutionised in the late 1990s by the advent of conflict-driven clause learning (CDCL), which has since been the dominating paradigm in propositional SAT solving [23, 24, 35]. A few years later, the CDCL approach was lifted to the computationally even harder setting of quantified Boolean formulas (QBF) in the form of quantified CDCL (QCDCL) [36]. Though a number of competing approaches to QBF solving exist (cf. [7] for a recent overview), QCDCL is one of most competitive. State-of-the-art implementations include DepQBF [21] and Qute [27].

In comparison to the propositional case, QCDCL poses additional technical challenges, stemming from partitioning the variables into existential and universal (SAT can be viewed as using only existential variables) and the dependencies between the variables imposed by the quantifier prefix. The presence of universal variables entails additional rules for unit propagation (universal reductions), while the variable dependencies imposed by the prefix are typically observed by decision heuristics in the sense that QCDCL follows the prefix order in decision making. The latter is arguably the most severe restriction when transitioning from CDCL to QCDCL. Another difference between CDCL and QCDCL arises from the fact that

unlike in SAT, a satisfying assignment to the QBF matrix does not imply that the QBF is true. Instead, this is witnessed by additionally learning cubes (i.e., conjunctions of literals, also called terms) and producing a cube verification for true QBFs.

Though CDCL and QCDCL are very efficient in practice and in particular on industrial instances (cf. [31] for an overview of QBF solving applications and [15, 22] for experimental studies of solver performance), their success and their inherent limitations are not at all well-understood from a theoretical perspective. The main theoretical approach is through proof complexity [9]. For SAT it is known that CDCL – viewed as a non-deterministic procedure – is equivalent to propositional resolution [1, 3, 29]. In particular, resolution refutations can be efficiently extracted from CDCL runs, whereby lower bounds for resolution proof size imply lower bounds for CDCL running time. However, when using CDCL with practical decision heuristics such as VSIDS [25], the model becomes exponentially weaker than resolution [34].

The situation is even more intricate in QBF. Again, from QCDCL runs, proofs can be efficiently extracted in the format of long-distance Q-Resolution [2, 36].[1] However, QCDCL – even as a non-deterministic procedure – is exponentially weaker than long-distance Q-Resolution and exponentially incomparable to the simpler system of Q-Resolution [6]. Thus it is very interesting, both from a theoretical and practical perspective, to gauge the precise power of QCDCL.

In this paper we introduce and investigate QCDCL models that drop the requirement of making variable decisions along the prefix order. Though it has been recently shown that following the prefix order in QCDCL is not needed for correctness[2], existing prefix-relaxing techniques do not exploit this as much as they could. Dependency schemes [20,28,30,32] work with the assumption that the prefix has to be observed, but notice that certain parts (often called *spurious dependencies*) can be relaxed in preprocessing. With dependency learning [27], a more recent, orthogonal technique, instead of calculating dependencies upfront the solver assumes independence until it runs into a problem, from which it learns a dependency on the fly (dependency learning can be combined with schemes [26]). These strategies are executed differently: with dependency schemes the solver can fully rely on the relaxed prefix and use it for decisions, propagation, and clause/cube learning alike; with dependency learning the solver can only use the relaxed prefix for decisions and propagation and must learn clauses and cubes with the original prefix in order to detect dependencies. However, both approaches share the restriction that once dependencies are found, decisions must respect them.

Our contributions. We propose a new QCDCL model where decisions can ignore quantification entirely; only propagation and clause/cube learning use the prefix information.

When suggesting a new model for solving, there are at least two possible approaches: (1) to give a formal account of the model, prove its correctness, and theoretically quantify the gains on running time; or (2) provide an implementation and experimentally evaluate its practical performance. In this paper, our main focus is to contribute towards (1). While we also perform some initial proof-of-concept experiments, an extensive practical evaluation of the competitiveness of the approach is left for future work (cf. the conclusion).[3]

---

[1] For practical solving, more succinct proof checking formats are used, both for CDCL [13] and QCDCL [14].
[2] It is needed for QDPLL [10, 12], but not for QCDCL (cf. also [6]).
[3] It appears to us that in SAT solving, theoretical analysis has so far been mainly carried out in retrospect, after practical solving developments had already taken place. However, we also see a case for theoretical research providing a-priori *guidance* for practical developments.

Specifically, our contributions are as follows:

1. Formal proof complexity models for QCDCL using arbitrary decisions. We provide a formal proof-complexity model for QCDCL with arbitrary decisions. This follows a recent line of research to formalise and rigorously analyse QCDCL from a proof complexity perspective [6,8].

Our most general model $\mathsf{QCDCL}^{\mathsf{Any}}$ allows arbitrary decisions. Care has to be taken to ensure that we can always learn new clauses and cubes, as otherwise termination of proof search is no longer guaranteed. We ensure this by adding a simple *new constraint condition* (NCC), which forbids making decisions that immediately falsify a clause or satisfy a cube (which is already trivially impossible in prefix-observing QCDCL).

A potential further drawback of not following prefix order is that we can no longer guarantee to learn *asserting* clauses or cubes. In order to address this, we introduce two subsystems of $\mathsf{QCDCL}^{\mathsf{Any}}$ – termed $\mathsf{QCDCL}^{\mathsf{Uni\text{-}Any}}$ and $\mathsf{QCDCL}^{\mathsf{Exi\text{-}Any}}$ – that allow to always learn asserting clauses and cubes, respectively. We prove that all three systems are sound, complete, and terminating.

2. Exponential separations between the QCDCL models. The main contribution of this paper lies in proving that both $\mathsf{QCDCL}^{\mathsf{Uni\text{-}Any}}$ and $\mathsf{QCDCL}^{\mathsf{Exi\text{-}Any}}$ allow for exponentially shorter proofs than the prefix-following QCDCL model. The resulting simulation order is depicted in Figure 2.

To show this we construct two QBF families that exponentially separate the systems. Both employ general constructions – using a "twin" and a "reverse" construction – that could potentially be used for further formulas. Technically, we use the recently developed lower bound approach via the *gauge* of QBFs [8]. However, different from previous work [6,8], which only considered clause learning, our lower bounds work against a more realistic QCDCL system that uses both clause and cube learning. Interestingly, the separation of $\mathsf{QCDCL}^{\mathsf{Uni\text{-}Any}}$ from QCDCL works on false QBFs, while the separation of $\mathsf{QCDCL}^{\mathsf{Exi\text{-}Any}}$ from QCDCL uses true QBFs. The latter is the first dedicated QBF proof-complexity lower bound on true formulas.[4] In fact, we provide a general method how to transform hardness of false QBFs into hardness of true formulas.

3. Proof-of-concept experiments. Though this is not our main focus, we provide initial experiments that confirm our theoretical findings. These experiments are only meant to illustrate that our approach is in principle superior to plain QCDCL, without considering the impact of other techniques like preprocessing or dependency learning, etc. (cf. the discussion of future work in the conclusion).

Organisation. The remainder of this paper is organised as follows. We start in Section 2 with reviewing QBF preliminaries. In Sections 3 and 4 we introduce and formally model the new QCDCL versions. Their proof-complexity analysis and the separations are proven in Section 5. Section 6 describes our proof-of-concept experiments and Section 7 outlines further work.

[4] Also in SAT, basically all lower bounds are for unsatisfiable formulas.

2 Preliminaries

Propositional and quantified formulas. Variables x and negated variables $\bar{x}$ are called *literals*. We denote the corresponding variable as $\mathrm{var}(x) := \mathrm{var}(\bar{x}) := x$.

A *clause* is a disjunction of literals and a *cube* is a conjunction of literals. We will sometimes interpret clauses and cubes as sets of literals on which we can perform set-theoretic operations.

A *unit clause* (ℓ) is a clause that consists of only one literal. The *empty clause* consists of zero literals, denoted $(\bot)$. We sometimes paraphrase $(\bot)$ as a unit clause with the "empty literal" $\bot$. A clause C is called *tautological* if $\{\ell, \bar{\ell}\} \subseteq C$ for some literal ℓ. If C is a set of literals with the same property, then we will also call it *tautological*.

We define a *unit cube* of a literal ℓ, denoted by $[\ell]$, and the *empty cube* $[\top]$ with "empty literal" $\top$. A cube D is *contradictory* if $\{\ell, \bar{\ell}\} \subseteq D$ for some literal ℓ. If C is a clause or a cube, we define $\mathrm{var}(C) := \{\mathrm{var}(\ell) : \ell \in C\}$. The negation of a clause $C = \ell_1 \vee \ldots \vee \ell_m$ is the cube $\neg C := \overline{C} := \bar{\ell}_1 \wedge \ldots \wedge \bar{\ell}_m$.

A *(total) assignment* σ of a set of variables V is a non-tautological set of literals such that for all $x \in V$ there is some $\ell \in \sigma$ with $\mathrm{var}(\ell) = x$. A *partial assignment* σ of V is an assignment of a subset $W \subseteq V$. A clause C is *satisfied* by an assignment σ if $C \cap \sigma \neq \varnothing$. A cube D is *falsified* by σ if $\neg D \cap \sigma \neq \varnothing$. A clause C that is not satisfied by σ can be *restricted* by σ, defined as $C|_\sigma := \bigvee_{\ell \in C, \bar{\ell} \notin \sigma} \ell$. Similarly we can restrict a non-falsified cube D as $D|_\sigma := \bigwedge_{\ell \in D \setminus \sigma} \ell$. Intuitively, an assignment sets all its literals to true.

A *CNF* (conjunctive normal form) is a conjunction of clauses and a *DNF* (disjunctive normal form) is a disjunction of cubes. We restrict a CNF (resp. DNF) ϕ by an assignment σ as $\phi|_\sigma := \bigwedge_{C \in \phi \text{ non-satisfied}} C|_\sigma$ (resp. $\phi|_\sigma := \bigvee_{D \in \phi \text{ non-falsified}} D|_\sigma$). For a CNF (DNF) ϕ and an assignment σ, if $\phi|_\sigma = \varnothing$, then ϕ is *satisfied* (*falsified*) by σ.

A *QBF* (quantified Boolean formula) $\Phi = \mathcal{Q} \cdot \phi$ consists of a propositional formula ϕ, called the *matrix*, and a *prefix* $\mathcal{Q}$. A *prefix* $\mathcal{Q} = \mathcal{Q}'_1 V_1 \ldots \mathcal{Q}'_s V_s$ consists of non-empty and pairwise disjoint sets of variables $V_1, \ldots, V_s$ and quantifiers $\mathcal{Q}'_1, \ldots, \mathcal{Q}'_s \in \{\exists, \forall\}$ with $\mathcal{Q}'_i \neq \mathcal{Q}'_{i+1}$ for $i \in [s-1]$. For a variable x in $\mathcal{Q}$, the *quantifier level* is $\mathrm{lv}(x) := \mathrm{lv}_\Phi(x) := i$, if $x \in V_i$. For $\mathrm{lv}_\Phi(\ell_1) < \mathrm{lv}_\Phi(\ell_2)$ we write $\ell_1 <_\Phi \ell_2$, while $\ell_1 \leqslant_\Phi \ell_2$ means $\ell_1 <_\Phi \ell_2$ or $\mathrm{lv}_\Phi(\ell_1) = \mathrm{lv}_\Phi(\ell_2)$.

For a QBF $\Phi = \mathcal{Q} \cdot \phi$ with ϕ a CNF (DNF), we call Φ a *QCNF* (*QDNF*). We define $\mathfrak{C}(\Phi) := \phi$ (resp. $\mathfrak{D}(\Phi) := \phi$). Φ is an *AQBF* (augmented QBF), if $\phi = \psi \vee \chi$ with CNF ψ and DNF χ. Again we write $\mathfrak{C}(\Phi) := \psi$ and $\mathfrak{D}(\Phi) := \chi$.

We restrict a QCNF (QDNF) $\Phi = \mathcal{Q} \cdot \phi$ by an assignment σ as $\Phi|_\sigma := \mathcal{Q}|_\sigma \cdot \phi|_\sigma$, where $\mathcal{Q}|_\sigma$ is obtained by deleting all variables from $\mathcal{Q}$ that appear in σ. Analogously, we restrict an AQBF $\Phi = \mathcal{Q} \cdot (\psi \vee \chi)$ as $\Phi|_\sigma := \mathcal{Q}|_\sigma \cdot (\psi|_\sigma \vee \chi|_\sigma)$.

If L is a set of literals (e.g., an assignment), we can get the negation of L, which we define as $\neg L := \overline{L} := \{\bar{\ell} \mid \ell \in L\}$.

(Long-distance) Q-resolution and Q-consensus. Let C_1 and C_2 be two clauses (cubes) from a QCNF (QDNF) or AQBF Φ. Let ℓ be an existential (universal) literal with $\mathrm{var}(\ell) \notin \mathrm{var}(C_1) \cup \mathrm{var}(C_2)$. The *resolvent* of $C_1 \vee \ell$ and $C_2 \vee \bar{\ell}$ over ℓ is defined as

$$(C_1 \vee \ell) \overset{\ell}{\bowtie}_\Phi (C_2 \vee \bar{\ell}) := C_1 \vee C_2$$

(resp. $(C_1 \wedge \ell) \overset{\ell}{\bowtie}_\Phi (C_2 \wedge \bar{\ell}) := C_1 \wedge C_2$).

Let $C := \ell_1 \vee \ldots \vee \ell_m$ be a clause from a QCNF or AQBF Φ such that $\ell_i \leqslant_\Phi \ell_j$ for all $i < j$, $i, j \in [m]$. Let k be minimal such that $\ell_k, \ldots, \ell_m$ are universal. Then we can perform a *universal reduction* step and obtain

$$\mathrm{red}^\forall_\Phi(C) := \ell_1 \vee \ldots \vee \ell_{k-1}.$$

Analogously, we perform *existential reduction* on cubes. Let $D := \ell_1 \wedge \ldots \wedge \ell_m$ be a cube of a QDNF or AQBF Φ with $\ell_i \leqslant_\Phi \ell_j$ for all $i < j$, $i, j \in [m]$. Let k be minimal such that $\ell_k, \ldots, \ell_m$ are existential. Then $\mathrm{red}^{\exists}_\Phi(D) := \ell_1 \wedge \ldots \wedge \ell_{k-1}$.

If it is clear that C is a clause or a cube, we can just write $\mathrm{red}_\Phi(C)$ or even $\mathrm{red}(C)$, if the QBF Φ is also obvious. We will write $\mathrm{red}(\Phi) = \mathrm{red}_\Phi(\Phi)$, if we reduce all clauses and cubes of the AQBF Φ according to its prefix.

As defined by Kleine Büning et al. [19], a Q-resolution (Q-consensus) proof π from a QCNF (QDNF) or AQBF Φ of a clause (cube) C is a sequence of clauses (cubes) $\pi = (C_i)_{i=1}^m$, such that $C_m = C$ and for each C_i one of the following holds:

- *Axiom:* $C_i \in \mathfrak{C}(\Phi)$ (resp. $C_i \in \mathfrak{D}(\Phi)$);
- *Resolution:* $C_i = C_j \overset{x}{\bowtie}_\Phi C_k$ with x existential (universal), $j, k < i$, and C_i non-tautological (non-contradictory);
- *Reduction:* $C_i = \mathrm{red}^{\forall}_\Phi(C_j)$ (resp. $C_i = \mathrm{red}^{\exists}_\Phi(C_j)$) for some $j < i$.

We call C the *root* of π. In [2], an extension of Q-resolution (Q-consensus) proofs to long-distance Q-resolution (long-distance Q-consensus) proofs was introduced by replacing the resolution rule by

- *Resolution (long-distance):* $C_i = C_j \overset{x}{\bowtie} C_k$ with x existential (universal) and $j, k < i$. The resolvent C_i is allowed to contain tautologies such as $u \vee \bar{u}$ (resp. $u \wedge \bar{u}$), if u is universal (existential). If there is such a universal (existential) $u \in \mathrm{var}(C_j) \cap \mathrm{var}(C_k)$, then we require $x <_\Phi u$.

Furthermore, a Q-resolution (Q-consensus) or long-distance Q-resolution (long-distance Q-consensus) proof π from Φ of the empty clause ($\bot$) (the empty cube ($\top$)) is called a *refutation* (*verification*) of Φ. In that case, Φ is called *false* (*true*). We will sometimes interpret π as a set of clauses (or cubes).

A proof system S *p-simulates* a system S', if every S' proof can be transformed in polynomial time into an S proof of the same formula.

3 Our QCDCL models

To analyse the complexity of QCDCL procedures, we need to fully formalise them as proof systems. This approach was initiated in [6] and [8], and we follow that framework. We will only sketch this formalization here.

We store all relevant information of a QCDCL run in *trails*. Since QCDCL uses several runs and potentially also restarts, a QCDCL proof will typically consist of many trails.

▶ **Definition 1** (trails). *A* trail $\mathcal{T}$ *for a QCNF or AQBF* Φ *is a (finite) sequence of pairwise distinct literals from* Φ*, including the empty literals* $\bot$ *and* $\top$*. In general, a trail has the form*

$$\mathcal{T} = (p_{(0,1)}, \ldots, p_{(0,g_0)}; \mathbf{d_1}, p_{(1,1)}, \ldots, p_{(1,g_1)}; \ldots; \mathbf{d_r}, p_{(r,1)}, \ldots, p_{(r,g_r)}), \tag{1}$$

where the d_i *are* decision literals *and* $p_{(i,j)}$ *are* propagated literals*. A trail* $\mathcal{T}$ *has* run into a conflict *if* $\bot \in \mathcal{T}$ *or* $\top \in \mathcal{T}$*.*

We write $x <_\mathcal{T} y$ *if* $x, y \in \mathcal{T}$ *and* x *is left of* y *in* $\mathcal{T}$*. We will use* $x \leqslant_\mathcal{T} y$ *if* $x <_\mathcal{T} y$ *or* $x = y$*.*

Each propagated literal $p_{(i,j)}$ *belongs to an antecedent clause (for existential literals) or cube (for universal literals)* $\mathrm{ante}_\mathcal{T}(p_{(i,j)})$ *that became unit at the point we made the propagation. We will denote the trail up to (and excluding) the propagation* $p_{(i,j)}$ *as* $\mathcal{T}[i,j]$*. The subtrail up to the decision* d_i *is denoted by* $\mathcal{T}[i,0]$*.*

We state some general facts about trails and antecedent clauses/cubes one should keep in mind.

▶ Remark 2. Let $\mathcal{T}$ be a trail, $\ell \in \mathcal{T}$ a propagated literal and $A := \text{ante}_{\mathcal{T}}(\ell)$.

- If ℓ is existential, then $\ell \in A$ and for each existential literal $x \in A$ with $x \neq \ell$ we need $\bar{x} <_{\mathcal{T}} \ell$.
- If ℓ is universal, then $\bar{\ell} \in A$ and for each universal literal $u \in A$ with $u \neq \bar{\ell}$ we need $u <_{\mathcal{T}} \ell$.

An essential element of QCDCL is clause and cube learning. This guarantees to make "progress" after each trail (at least under some conditions that we will specify later).

▶ **Definition 3** (learnable constraints). *Let $\mathcal{T}$ be a trail for Φ of the form (1) with $p_{(r,g_r)} \in \{\bot, \top\}$. Starting with $ante_{\mathcal{T}}(\bot)$ (resp. $ante_{\mathcal{T}}(\top)$) we reversely resolve over the antecedent clauses (cubes) that were used to propagate the existential (universal) variables, until we stop at some arbitrarily chosen point. The clause (cube) we so derive is a* learnable constraint. *We denote the set of learnable constraints by $\mathfrak{L}(\mathcal{T})$.*

We can also learn cubes from trails that did not run into conflict. If $\mathcal{T}$ is a total assignment of the variables from Φ, then we define the set of learnable constraints as the set of cubes $\mathfrak{L}(\mathcal{T}) := \{red^{\exists}_{\Phi}(D) |\ D \subseteq \mathcal{T}$ and D satisfies $\mathfrak{C}(\Phi)\}$.

In QCDCL, our goal is to make "progress" in each run/trail. Thus, we have to ensure that we can always learn new clauses or cubes from a constructed trail. Since we want to work with QCDCL models that do not necessarily follow the prefix order for decision making, it is not guaranteed that we can even learn new constraints from each trail. As we will show later, we need the following condition to prevent such a situation, which could easily lead to a loop in practical solving.

▶ **Definition 4.** *A trail $\mathcal{T}$ for a formula Φ fulfils the* New Constraint Condition *(*NCC *for short), if for each decision d_i the formula $red(\Phi|_{\mathcal{T}[i,0]\cup\{d_i\}})$ does not contain the empty clause or cube.*

Intuitively, this means that a decision must not lead to a conflict immediately. It will become clear later, why we can always find a decision that does not violate the NCC. In fact, classical QCDCL automatically fulfils this condition.

We will now formally define our four QCDCL proof systems, namely QCDCL, QCDCL$^{\text{Any}}$, QCDCL$^{\text{Uni-Any}}$, and QCDCL$^{\text{Exi-Any}}$.

▶ **Definition 5** (QCDCL proof systems). *Let S be one of* QCDCL, QCDCL^{Any}, QCDCL$^{Uni\text{-}Any}$, QCDCL$^{Exi\text{-}Any}$. *An S proof ι from a QCNF $\Phi = \mathcal{Q} \cdot \phi$ of a clause or cube C is a sequence of triples*

$$\iota := [(\mathcal{T}_i, C_i, \pi_i)]_{i=1}^{m},$$

where $C_m = C$, each $\mathcal{T}_i$ is a trail for Φ_i (we define Φ_i as the updated formula Φ before the i^{th} trail) that fulfils the NCC, each $C_i \in \mathfrak{L}(\mathcal{T}_i)$ is one of the constraints we can learn from each trail, and π_i is the long-distance Q-resolution *or* long-distance Q-consensus *proof of C_i from Φ_i that we obtain by performing the steps in Definition 3. We will denote the set of trails in ι as $\mathfrak{T}(\iota)$.*

The four systems differ in how decisions are made:

- QCDCL*: For each decision d_i we have that $lv_{\Psi|_{\mathcal{T}[i,0]}}(d_i) = 1$. I.e., decisions are level-ordered.*

- QCDCL$^{\text{Any}}$: *Decisions can be made arbitrarily as long as the NCC is fulfilled.*
- QCDCL$^{\text{Uni-Any}}$: *An existential decision d_i can only be made if all universal variables that are quantified left of d_i were already assigned in $\mathcal{T}$. Universal decisions can be made in any order as long as the NCC is fulfilled.*
- QCDCL$^{\text{Exi-Any}}$: *A universal decision d_i can only be made if all existential variables that are quantified left of d_i were already assigned in $\mathcal{T}$. Existential decisions can be made in any order as long as the NCC is fulfilled.*

After each trail, we will backtrack to some arbitrary previous point in the trail and continue to decide or propagate from that point.

If $C = C_m = (\bot)$, then ι is called an S refutation *of Φ. If $C = C_m = [\top]$, then ι is called an S* verification *of Φ. The proof ends once we have learned $(\bot)$ or $[\top]$.*

If C is a clause, we can stick together the long-distance Q-resolution *derivations from $\{\pi_1, \ldots, \pi_m\}$ and obtain a* long-distance Q-resolution *proof from Φ of C, which we call $\mathfrak{R}(\iota)$. Similarly, if C is a cube, we can stick together the* long-distance Q-consensus *derivations and obtain a* long-distance Q-consensus *proof $\mathfrak{R}(\iota)$ from Φ of C.*

The size *of ι is defined as $|\iota| := \sum_{i=1}^{m} |\mathcal{T}_i|$. Obviously, we have $|\mathfrak{R}(\iota)| \in \mathcal{O}(|\iota|)$.*

Our formalisation above is based on [6, 8]. However, since in the present paper cube learning is always included, our plain model QCDCL now includes clause *and* cube learning (while in [6, 8], QCDCL denotes a system with just clause learning, but without learning cubes).

The concept behind the two models QCDCL$^{\text{Uni-Any}}$ and QCDCL$^{\text{Any}}$ was already introduced in [6] (albeit defined slightly differently, they were called QCDCL$^{\text{Ass-R-Ord}}_{\text{Red}}$ and QCDCL$^{\text{Any-Ord}}_{\text{No-Red}}$ in those papers). However, since we include cube learning now, our models here match practical solving much better.

▶ Remark 6. In QCDCL, decision making can never violate the NCC if we create the trails "naturally" (i.e., decisions are only made if and only if there are no more propagations on the same decision level left, and conflicts must be detected immediately if there are any).

We still have to make sure to fulfil NCC when backtracking, though. We will explain later how this is achieved.

The next result states simulations between systems, cf. Figure 2. They all follow by definition.

▶ **Proposition 7.** *Each* QCDCL *proof is also a* QCDCL$^{\text{Uni-Any}}$ *and* QCDCL$^{\text{Exi-Any}}$ *proof, and each* QCDCL$^{\text{Uni-Any}}$ *or* QCDCL$^{\text{Exi-Any}}$ *proof is also a* QCDCL$^{\text{Any}}$ *proof.*

4 Learning asserting constraints

We recall the notion of an asserting clause (or cube). The concept originates from SAT solving [24], but directly lifts to QBF [12, 36]. Intuitively, asserting constraints are learnable constraints that become unit after backtracking. We give a more liberal definition as we do not refer to specific asserting constraints (such as UIP clauses).

▶ **Definition 8** (asserting constraints)**.** *Let $\mathcal{T}$ be a trail for a QCNF Φ that contains r decision literals. A clause (cube) $C \in \mathfrak{L}(\mathcal{T})$ is called* asserting*, if there exists some point $[i, j]$ such that $red^{\forall}_{\Phi}(C|_{\mathcal{T}[i,j]})$ is a unit clause (resp. $red^{\exists}_{\Phi}(C|_{\mathcal{T}[i,j]})$ is a unit cube). Furthermore, we require that we backtrack by at least one decision level, i.e., $i < r$ or $j = 0$.*

Learning asserting clauses might be advantageous as it guarantees new unit propagations after backtracking to a suitable point. In addition, asserting clauses are always new.

▶ **Proposition 9.** *If $\mathcal{T}$ is a trail in a* QCDCL$^{\text{Uni-Any}}$ *(resp.* QCDCL$^{\text{Exi-Any}}$*) proof of a formula Φ, and if $\bot \in \mathcal{T}$ (resp. $\top \in \mathcal{T}$), then there exists a new asserting or empty clause (cube) $C \in \mathfrak{L}(\mathcal{T})$.*

Furthermore, if C is non-empty, there exists a point $[i, j]$ in the trail to which we can backtrack after learning C such that the NCC continues to hold.

A similar result holds for the any-order model, albeit with the difference that we might not be able to learn asserting constraints. But at least we can guarantee to learn a new clause/cube.

▶ **Proposition 10.** *If $\mathcal{T}$ is a trail in a* QCDCL$^{\text{Any}}$ *proof for a formula Φ, that has run into a conflict or in which we assigned all variables, then $\mathfrak{L}(\mathcal{T})$ contains a new clause or cube C that is not contained in Φ. Further, if C is non-empty, there exists a point $[i, j]$ in the trail to which we can backtrack after learning C such that the NCC continues to hold.*

▶ Remark 11. To illustrate the importance of the NCC, we give an example of a QCDCL$^{\text{Any}}$ trail – violating the NCC – from which we cannot learn a new clause. Consider the trail $\mathcal{T} = (\mathbf{x}, \bot)$ for the false QCNF $\forall u \exists x \cdot (u \vee x) \wedge (u \vee \bar{x}) \wedge (\bar{u} \vee x) \wedge (\bar{u} \vee \bar{x})$. The trail violates the NCC, as we got a conflict directly after the decision x. The only learnable clause is $\text{ante}_{\mathcal{T}}(\bot) = \bar{u} \vee \bar{x}$, which is obviously already known.

Another example illustrates the case where we can learn a new clause but no asserting clause. Let the trail be $\mathcal{U} := (\mathbf{x}, y; \mathbf{u}, \bar{z}, \bot)$ for the false QCNF $\forall u \exists x, y, z \cdot (\bar{x} \vee y) \wedge (x \vee y) \wedge (u \vee \bar{y} \vee \bar{z}) \wedge (\bar{u} \vee \bar{y} \vee \bar{z}) \wedge (u \vee \bar{y} \vee z) \wedge (\bar{u} \vee \bar{y} \vee z)$. There are two new clauses we could learn: $\bar{u} \vee \bar{y}$ or $\bar{u} \vee \bar{x}$. None of the two can become unit after backtracking since we used the decision heuristic for QCDCL$^{\text{Exi-Any}}$, although we followed the NCC.

As a special case we obtain for our base model QCDCL the following situation.

▶ **Corollary 12.** *[Folklore, cf. [20]] If $\mathcal{T}$ is a trail in a* QCDCL *proof for a formula Φ, that has run into a conflict, then $\mathfrak{L}(\mathcal{T})$ contains an asserting or empty clause or cube. If $\mathcal{T}$ has not run into a conflict, but we have assigned all variables in $\mathcal{T}$, then $\mathfrak{L}(\mathcal{T})$ contains at least a new cube C.*

Furthermore, if C is non-empty, there exists a point $[i, j]$ in the trail to which we can backtrack after learning C such that the NCC continues to hold.

Figure 1 provides an overview of the four systems and their ability to learn asserting clauses and cubes. As a consequence of always learning new constraints, we infer that our models are all complete and terminating proof methods.

▶ **Theorem 13.** QCDCL, QCDCL$^{\text{Any}}$, QCDCL$^{\text{Uni-Any}}$ *and* QCDCL$^{\text{Exi-Any}}$ *are sound and complete proof systems.*[5] *Additionally, as long as we follow the rules of decision making (especially the NCC), we will always learn the empty clause or cube at some point, no matter what decisions were made.*

[5] I.e., they are proof systems for the language of false and true QBFs in the setting of [11]. Technically, in order not to trivialise the notion of such a proof system, we could consider proof systems for the language L of the marked union of true and false QBFs, i.e., $L = \{0\Phi \mid \Phi \text{ is a false QBF}\} \cup \{1\Phi \mid \Phi \text{ is a true QBF}\}$. In this way, L is still PSPACE complete.

	asserting clauses	only new clauses
asserting cubes	QCDCL	QCDCL$^{\text{Exi-Any}}$
only new cubes	QCDCL$^{\text{Uni-Any}}$	QCDCL$^{\text{Any}}$

Figure 1 Overview of guaranteed learnable constraints after a trail conflict in the corresponding models.

5 Separations of QCDCL systems

In this section, we will exponentially separate our three new models – where decisions do not necessarily follow the prefix order – from the plain model QCDCL. We omit some of the proofs.We will use the *gauge lower bound technique*, introduced in [8], which we will first review. This technique works on Σ_3^b QCNFs. To ease notation, we will assume that prefixes of Σ_3^b QCNFs have the form $\exists X \forall U \exists T$, for sets of literals X, U, T, and we will use the notions of X- U- and T-variables and -literals. Further, we define certain types of clauses:

- X-clauses consist of X-literals only (analogously we define U-clauses and T-clauses),
- XT-clauses consist of at least one X- and at least one T-literal, but no U-literals,
- XUT-clauses consist of at least one X-, U- and T-literal, respectively.

The gauge lower bound method works for a specific class of Σ_3^b QCNFs with the *XT-property*.

▶ **Definition 14** ([6]). *We say that Φ fulfills the* XT-property*, if $\mathfrak{C}(\Phi)$ contains no XT-clauses, no T-clauses that are unit (or empty) and no two T-clauses from $\mathfrak{C}(\Phi)$ are resolvable.*

The XT-property extends to entire QCDCL proofs, as stated in the next lemma.

▶ **Lemma 15** ([6]). *If Φ is a Σ_3^b QCNF that fulfills the XT-property, then it is not possible to derive XT-clauses or new T-clauses via* long-distance Q-resolution *from Φ.*

The gauge lower-bound method from [8] uses the next two notions of fully reduced and primitive proofs (they were implicit in [8] and stated explicitly in [5]).

▶ **Definition 16** (fully reduced proofs [5, 8]). *A* long-distance Q-resolution *refutation π of a QCNF Φ is* fully reduced*, if for each clause $C \in \pi$ that contains universal literals that are reducible, the reduction step is performed immediately and C is not used otherwise in the proof.*

Fully reduced proofs are not much of a limitation. In fact, all long-distance Q-resolution proofs that we extract from a QCDCL run are already fully reduced by default. Also, we can always shorten a given long-distance Q-resolution proof by making it fully reduced.

▶ **Definition 17** (primitive proofs [5,8]). *A* long-distance Q-resolution *proof π from a Σ_3^b formula is* primitive*, if there are no two XUT-clauses in π that are resolved over an X-variable.*

Unlike the fully reduced property, not all proofs extracted from QCDCL are primitive, in general.

Our lower bound method will not work for all QCDCL proofs, but needs *fully reduced primitive* Q-resolution proofs, which are better suited for a proof-complexity analysis. Later, the challenge will be to show that certain extracted proofs from QCDCL are primitive. Note that fully reduced primitive long-distance Q-resolution proofs are always Q-resolution proofs.

The main measure for the lower bound technique is the *gauge* of a formula, defined in [8].

▶ **Definition 18** ([8]). *Let Φ be a Σ_3^b QCNF with prefix $\exists X \forall U \exists T$. We define W_Φ as the set of all* Q-resolution *proofs π from Φ of X-clauses C_π, such that π consists of resolutions over T-literals and reductions only. We define*

$$gauge(\Phi) := \min\{|C_\pi| \,:\, C_\pi \text{ is the root of some } \pi \in W_\Phi\}.$$

Intuitively, gauge(Φ) is the minimal number of X-literals that are piled up during the process of deriving an X-clause without using resolutions over X-literals. In other words: to get rid of all T-literals from Φ, we have to pile up at least gauge(ϕ) many different X-literals.

All notions we introduced so far are combined into the following lower bound method:

▶ **Theorem 19** ([8]). *Each fully reduced primitive* Q-resolution *refutation of a Σ_3^b QCNF Φ that fulfils the XT-property has size $2^{\Omega(gauge(\Phi))}$.*

Our goal is to find formulas that separate QCDCL from QCDCL$^{\text{Uni-Any}}$ and QCDCL from QCDCL$^{\text{Exi-Any}}$, respectively. We start with the latter.

5.1 Separation on true formulas

The advantage of QCDCL$^{\text{Exi-Any}}$ (compared to QCDCL) is to decide existential literals out of order while still learning asserting cubes. Since cubes are important for verifications of true formulas, it makes sense to use true QBFs for the separation.

First, we discuss two generic modifications for QBFs. The twin construction doubles all universal variables. For all clauses with universal variables a copy is created in the twin variables.

▶ **Definition 20** (twin formulas). *Let $\Phi = \exists X \forall U \exists T \cdot \mathfrak{C}(\Phi)$ be a QCNF. Let $U = \{u_1, \ldots, u_m\}$ and let $v_1, \ldots, v_m$ be variables not occuring in Φ. Then the* twin formula *of Φ is the QCNF* $\texttt{Twin}\Phi$ *defined as*

$$\texttt{Twin}\Phi := \exists X \forall (U \cup \{v_1, \ldots, v_m\}) \exists T \cdot \mathfrak{C}(\Phi) \wedge \bigwedge_{C \in \mathfrak{C}(\Phi)} C[u_1/v_1, \ldots, u_m/v_m],$$

where u_i/v_i indicates that all occurrences of u_i are substituted by v_i.

The second modification is the *reversion* of a formula.

▶ **Definition 21.** *If $\Phi = \mathcal{Q}_1 V_1 \mathcal{Q}_2 V_2 \ldots \mathcal{Q}_k V_k \cdot \bigwedge_{j=1}^m C_j$ is a QCNF with $\mathcal{Q}_i \in \{\exists, \forall\}$ and disjoint sets of variables V_i for $i = 1, \ldots, k$, then the* reversion *Rev(Φ) of Φ is the QCNF*

$$\mathcal{Q}'_1 V_1 \mathcal{Q}'_2 V_2 \ldots \mathcal{Q}'_k V_k \forall w \exists c_1, \ldots, c_m \cdot (\bar{c}_1 \vee \ldots \vee \bar{c}_m) \wedge \bigwedge_{j=1}^{m} \bigwedge_{\ell \in C_j} (\bar{\ell} \vee w \vee c_j) \wedge (\bar{\ell} \vee \bar{w} \vee c_j)$$

where $\mathcal{Q}'_i = \forall$ if $\mathcal{Q}_i = \exists$, and $\mathcal{Q}'_i = \exists$ if $\mathcal{Q}_i = \forall$, and $w, c_1, \ldots, c_m$ are new variables not contained in Φ.

It is easy to prove that there exists a duality between the truth values of Φ and Rev(Φ).

▶ **Lemma 22.** *If Φ is a QCNF, then Rev(Φ) is true if and only if Φ is false.*

We will use the reversion to lift hardness from false to true QCNFs. To verify a true formula, we need to create a proof using cubes. We will show that Rev(Φ) is designed such that its initial cubes are basically the negated axiom clauses of Φ. Thus, a verification of Rev(Φ) can be transformed into a refutation of Φ.

Our reversion was inspired by the notion of the *negation* from [18]. The only change we made is adding the variable w. We did this to prevent a direct connection between an X- or U-block and an auxiliary variable c_j from the last block. Our lower bound technique is based on the fact that on certain formulas we cannot have direct connections (hence: cannot directly propagate) between outer and inner quantifier blocks. The added variable w helps to maintain this property.

The next two results shows how we can transform verifications of $\mathrm{Rev}(\Phi)$ into refutations of Φ by interpreting the cubes from the verification as negated clauses of a refutation.

▶ **Lemma 23.** *Let $\Phi = \mathcal{Q} \cdot \bigwedge_{j=1}^{m} C_j$ be a QCNF and let σ be an assignment that satisfies $\mathfrak{C}(Rev(\Phi))$. Then there exists some $C \in \mathfrak{C}(\Phi)$ with $\overline{C} \subseteq \sigma$.*

▶ **Proposition 24.** *If Φ is a false QCNF and ρ is a* long-distance Q-consensus *verification of $Rev(\Phi)$, then ρ can be transformed into a fully reduced* long-distance Q-resolution *refutation π of Φ with $|\pi| \leqslant |\rho|$.*

More precisely, for each clause $C \in \pi$ there is a cube $C' \in \rho$ with $\overline{C} \subseteq C'$. Furthermore, for each two clauses C, D that are resolved in π, the corresponding cubes C', D' are resolved in ρ, as well.

For our next results, we need an even stronger property than the XT-property: We require, that clauses from the formula contain at least one U- and T-literal.

▶ **Lemma 25.** *If Φ is a Σ_3^b QCNF, in which all clauses contain at least one U- and T-literal, then Φ fulfils the XT-property.*

We combine the results above and obtain a new lower bound technique for true formulas, which builds on the gauge technique for false formulas.

▶ **Theorem 26.** *Let Φ be a false Σ_3^b. Additionally, let all clauses $C \in \mathfrak{C}(\Phi)$ contain at least one U- and one T-literal. If the QCNF $\mathtt{Twin}\Phi$ needs fully reduced primitive* Q-resolution *refutations of size s, then* QCDCL *verifications for $Rev(\mathtt{Twin}\Phi)$ also need size s.*

Proof. Let ι be a QCDCL verification for $\mathrm{Rev}(\mathtt{Twin}\Phi)$. We will show that there exists a fully reduced primitive Q-resolution refutation π for $\mathtt{Twin}\Phi$ with $|\pi| \leqslant |\mathfrak{R}(\iota)|$.

Let π be the long-distance Q-resolution refutation of $\mathtt{Twin}\Phi_n$ as described in Proposition 24. Then π is fully reduced. We will show that π is primitive.

Assume not. Then there are two XUT-clauses $B_1, B_2 \in \pi$ that are resolved over some $x \in X$. By the construction of π described in Proposition 24, we can find two cubes $D_1, D_2 \in \mathfrak{R}(\iota)$ such that $\mathrm{var}(D_i) \cap U \neq \varnothing$ and $\mathrm{var}(D_i) \cap T \neq \varnothing$ for $i = 1, 2$ which are resolved over x. One of these cubes was an antecedent cube for x in some trail $\mathcal{T} \in \mathfrak{T}(\iota)$, say $D_1 = \mathrm{ante}_{\mathcal{T}}(x)$ (that means $\bar{x} \in D_1$).

In particular, there is some T-literal $t \in D_1$ such that $t <_{\mathcal{T}} x$ because D_1 must become unit. Remember that t is universal in $\mathrm{Rev}(\mathtt{Twin}\Phi)$ and we can only reduce cubes existentially. Then either t was a regular decision, or a propagation.

Case 1: t was decided.

This is only possible if all U-variables were assigned before. Hence, for each $u \in U$ there is a literal ℓ_u with $\mathrm{var}(\ell_u) = u$ and $\ell_u <_{\mathcal{T}} t <_{\mathcal{T}} x$. Because decisions have to be level-ordered in QCDCL, all ℓ_u had to have been propagated.

Let ℓ_u be the leftmost U-literal in $\mathcal{T}$. Consider its antecedent clause $A := \mathrm{ante}_{\mathcal{T}}(\ell_u)$.

▷ Claim. If ℓ_u is the leftmost U-literal in $\mathcal{T}$, then there exists an $i \in \{1, \ldots, m\}$ such that $c_i \in \mathrm{var}(\mathrm{ante}_{\mathcal{T}}(\ell_u))$ (where $c_1, \ldots, c_m$ are the variables from $\mathrm{Rev}(\mathtt{Twin}\Phi)$ as in Definition 21).

We want to create a contradiction by applying the claim, for which we need to show that A does not contain any literal from $\{c_r, \bar{c}_r \mid r = 1, \ldots, m\}$.

Assume that there is such a literal. That means we can find the leftmost literal $c \in \{c_r, \bar{c}_r \mid r = 1, \ldots, m\}$ in $\mathcal{T}$, hence $c <_{\mathcal{T}} \ell_u <_{\mathcal{T}} t <_{\mathcal{T}} x$. Now, c cannot have been a decision since decisions must be level-ordered. That means that c has been propagated by an antecedent clause $F := \mathrm{ante}_{\mathcal{T}}(c)$. Because c was leftmost, F cannot be the clause $\bar{c}_1 \vee \ldots \vee \bar{c}_m$. It is easy to see that F then has to contain either w or $\bar{w}$ by the structure of a reversion (see Definition 21). W.l.o.g. let $w \in F$. Then we need $\bar{w} <_{\mathcal{T}} c <_{\mathcal{T}} \ell_u$. Because of the quantification order, $\bar{w}$ cannot be a decided literal. Hence $\bar{w}$ must have been propagated by some antecedent cube $E := \mathrm{ante}_{\mathcal{T}}(\bar{w})$. Let ρ be the subproof of E from $\mathfrak{R}(\iota)$. Then there exists an initial cube $G \in \rho$ with $w \in G$, which is not getting resolved away in ρ. Furthermore, G is also an initial cube in $\mathfrak{R}(\iota)$. By Lemma 23, there exists some $H \in \mathfrak{C}(\mathtt{Twin}\Phi)$ such that $\overline{H} \subseteq G$. Since each clause of Φ contains a U-literal, there is such a U-literal $v \in \overline{H} \subseteq G$ and also $v \in E$ because it cannot be resolved or reduced away. This means we need $v <_{\mathcal{T}} \bar{w} <_{\mathcal{T}} \ell_u$, which is a contradiction to the choice of ℓ_u.

We have now shown that A does not contain any $c_r, \bar{c}_r$, $r \in \{1, \ldots, m\}$. However, this is impossible by our claim. We conclude that Case 1 cannot occur.

Case 2: t was propagated.

Proof sketch. We can prove that the antecedent cube $J := \mathrm{ante}_{\mathcal{T}}(t)$ contains at least one U-literal ℓ due to our precondition that all clauses from Φ contain at least one U-literal. Therefore, $\mathcal{T}$ must contain a leftmost U-literal. By repeating the argument from Case 1, this leads to a contradiction and therefore, Case 2 does not occur either.

We get a contradiction regarding our assumption that π was not primitive. ◀

We now construct specific QBFs that meet the conditions of Theorem 26. We already know from [8] that the equality formulas $\mathtt{Eq}_n$ of [4] have linear gauge and therefore need exponential-size fully reduced primitive Q-resolution refutations. However, not all clauses from $\mathtt{Eq}_n$ contain a U-literal. We modify the formulas by adding an artificial U-literal p to the relevant clauses:

▶ **Definition 27.** *The QCNF* $\mathtt{ModEq}_n$ *consists of the prefix* $\exists x_1, \ldots, x_n \forall u_1, \ldots, u_n, p \exists t_1, \ldots, t_n$ *and the matrix* $x_i \vee u_i \vee t_i$, $\bar{x}_i \vee \bar{u}_i \vee t_i$, $p \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$, $\bar{p} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$ *for* $i = 1, \ldots, n$.

Neither this nor the `Twin` modification changes the gauge of the formulas. Hence we get:

▶ **Proposition 28.** *It holds gauge*$(\mathtt{TwinModEq_n}) = n$. *Hence,* $\mathtt{TwinModEq_n}$ *needs exponential-size fully reduced primitive* Q-resolution *refutations.*

Proof. Since all axiom clauses contain T-literals, we have to get rid of them somehow. The only four clauses that contain T-literals in a negative polarity are the clauses $p \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$, $\bar{p} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$, $q \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$ and $\bar{q} \vee \bar{t}_1 \vee \ldots \vee \bar{t}_n$, where q is the copy of p. Hence, we have to use at least one of them in order to derive an X-clause. In particular, we have to resolve over each t_i. The only four clauses in which t_i occurs in a positive polarity are $x_i \vee u_i \vee t_i$, $\bar{x}_i \vee \bar{u}_i \vee t_i$, $x_i \vee v_i \vee t_i$ and $\bar{x}_i \vee \bar{v}_i \vee t_i$, where v_i is the copy of u_i. In each case we will pile up x_i or $\bar{x}_i$ for each resolution over t_i. Therefore, our X-clause at the end will contain at least n different X-literals.

Hence gauge$(\mathtt{TwinModEq_n}) = n$. The second claim then follows from Theorem 19. ◀

The lower bound for the true QBFs then follows with Theorem 26.

▶ **Corollary 29.** *Rev*($\mathtt{TwinModEq}_n$) *needs exponential-size* QCDCL *verifications.*

We now use a direct construction to show that Rev($\mathtt{TwinModEq}_n$) is easy for QCDCL$^{\text{Exi-Any}}$.

▶ **Proposition 30.** *Rev*($\mathtt{TwinModEq}_n$) *has polynomial-size* QCDCL$^{\text{Exi-Any}}$ *verifications.*

Proof sketch. We first learn those cubes that are negated clauses of $\mathtt{ModEq}_n$ (we do not need to learn the copies) by suitably choosing our decisions. Afterwards, we can basically follow the short QCDCL$^{\text{Uni-Any}}$ proof for $\mathtt{Eq}_n$ from [6] (note that the model was named differently there), where all quantifiers are flipped and cubes are learnt instead of clauses. ◀

▶ **Corollary 31.** QCDCL *and* QCDCL$^{\text{Exi-Any}}$ *are exponentially separated on true formulas.*

5.2 Separation on false formulas

For separating QCDCL and QCDCL$^{\text{Uni-Any}}$, we recall the completion principle $\mathtt{CR}_n$ of [17].

▶ **Definition 32** ([17])**.** *The false QCNF* $\mathtt{CR}_n$ *consists of the prefix* $\exists X \forall U \exists T$ *with*

$$X := \{x_{(i,j)} \mid i,j \in [n]\}, \quad U := \{u\}, \quad T := \{a_i, b_i \mid i \in [n]\}$$

and the matrix

$$x_{(i,j)} \vee u \vee a_i \quad \bar{x}_{(i,j)} \vee \bar{u} \vee b_j \quad \bar{a}_1 \vee \ldots \vee \bar{a}_n \quad \bar{b}_1 \vee \ldots \vee \bar{b}_n$$

for $i, j = 1, \ldots, n$.

For the lower bound, we will use the modification $\mathtt{TwinCR}_n$. As we show, cube learning becomes rather useless with the `Twin` modification. This fact helps us to ensure that QCDCL refutations of $\mathtt{TwinCR}_n$ are primitive, and thus we can apply the gauge lower-bound method.

Similarly as in Proposition 28 we can compute the gauge.

▶ **Lemma 33.** *It holds* $\mathit{gauge}(\mathtt{TwinCR}_n) = n$.

The main work is to check that QCDCL refutations of $\mathtt{TwinCR}_n$ are primitive.

▶ **Proposition 34.** *If* ι *is a* QCDCL *refutation of* $\mathtt{TwinCR}_n$*, then* $\mathfrak{R}(\iota)$ *is fully reduced and primitive.*

Applying Theorem 19 then yields the lower bound.

▶ **Corollary 35.** $\mathtt{TwinCR}_n$ *needs exponential-sized* QCDCL *refutations.*

On the other hand, $\mathtt{TwinCR}_n$ is easy for QCDCL$^{\text{Uni-Any}}$. Basically, we can simulate the Q-resolution refutation of $\mathtt{CR}_n$ from [16], because we can decide universal literals out of order.

▶ **Proposition 36.** $\mathtt{TwinCR}_n$ *has polynomial-sized* QCDCL$^{\text{Uni-Any}}$ *refutations.*

Proof. For each $k = 1, \ldots, n$ we construct the trail

$$\mathcal{T}_k := (\bar{\mathbf{x}}_{(\mathbf{1},\mathbf{k})}; \ldots; \bar{\mathbf{x}}_{(\mathbf{n},\mathbf{k})}; \bar{\mathbf{u}}, a_1, \ldots, a_n, \bot)$$

with antecedent clauses

$$\mathrm{ante}_{\mathcal{T}_k}(a_i) = x_{(i,k)} \vee u \vee a_i, \quad \mathrm{ante}_{\mathcal{T}_k}(\bot) = \bar{a}_1 \vee \ldots \vee \bar{a}_n,$$

for $i = 1, \ldots, n$.

Resolving $\bar{a}_1 \vee \ldots \vee \bar{a}_n$ over each $\text{ante}_{\mathcal{T}_k}(a_i)$ gives us the clause $E_k := x_{(1,k)} \vee \ldots \vee x_{(n,k)}$, which we will learn. Note that the trails and the learned clauses will not affect each other, hence the order in which we construct these n trails does not matter. Next, we construct the trails $\mathcal{U}_1, \ldots, \mathcal{U}_{n-1}$ (in that order). From each $\mathcal{U}_k$ we learn the clause $C_k := \bar{u} \vee b_k$. While constructing $\mathcal{U}_k$, we assume that $C_1, \ldots, C_{k-1}$ were already learned. Then, $\mathcal{U}_k$ looks as follows:

$$\mathcal{U}_k := (\mathbf{u}, b_1, \ldots, b_{k-1}; \mathbf{v}; \bar{\mathbf{b}}_\mathbf{k}, \bar{x}_{(1,k)}, \ldots, \bar{x}_{(n,k)}, \bot)$$

with antecedent clauses

$$\text{ante}_{\mathcal{U}_k}(b_j) = C_j, \quad \text{ante}_{\mathcal{U}_k}(\bar{x}_{(i,k)}) = \bar{x}_{(i,k)} \vee \bar{u} \vee b_k, \quad \text{ante}_{\mathcal{U}_k}(\bot) = E_k,$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, k-1$. Resolving E_k over each $\text{ante}_{\mathcal{U}_k}(\bar{x}_{(i,k)})$ leads to the learnable clause C_k. Having learned the clauses $C_1, \ldots, C_{n-1}$, we continue with the trail $\mathcal{V}$, which will be the last one. It looks as follows:

$$\mathcal{V} := (\mathbf{u}, b_1, \ldots, b_{n-1}, \bar{b}_n, \bar{x}_{(1,n)}, \ldots, \bar{x}_{(n,n)}, \bot)$$

with antecedent clauses

$$\text{ante}_{\mathcal{V}}(b_j) = C_j, \quad \text{ante}_{\mathcal{V}}(\bar{b}_n) = \bar{b}_1 \vee \ldots \vee \bar{b}_n, \quad \text{ante}_{\mathcal{V}}(\bar{x}_{(i,n)}) = \bar{x}_{(i,n)} \vee \bar{u} \vee b_n,$$
$$\text{ante}_{\mathcal{V}}(\bot) = E_n,$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, n-1$. Since we only made a universal decision, we can learn the empty clause ($\bot$) from $\mathcal{V}$ by resolving over everything.

Thus we constructed a QCDCL$^{\text{Uni-Any}}$ refutation using $2n + 1$ trails. ◀

▶ **Corollary 37.** QCDCL *and* QCDCL$^{\text{Uni-Any}}$ *are exponentially separated on false formulas.*

We combine both separations into our main result:

▶ **Theorem 38.**

a) QCDCL$^{\text{Uni-Any}}$ *is exponentially stronger than* QCDCL *on false formulas.*

b) QCDCL$^{\text{Exi-Any}}$ *is exponentially stronger than* QCDCL *on true formulas.*

c) QCDCL$^{\text{Any}}$ *is exponentially stronger than* QCDCL *both on false and true formulas.*

Besides $\texttt{TwinCR}_n$, we can find further separations between QCDCL and QCDCL$^{\text{Uni-Any}}$. The QCNFs $\texttt{MirrorCR}_n$ were introduced in [5] as a modification of $\texttt{CR}_n$, where it was shown that the formula is hard for several variants of QCDCL, including our base model QCDCL. It is notable that the matrix of $\texttt{MirrorCR}_n$ is unsatisfiable, and therefore we will never perform cube learning.

6 Experiments

One of the aspirations of proof complexity is to explain and predict solver behaviour, in particular running time. In this section, we evaluate how well our proof-complexity results transfer to the "real world" of QCDCL implemented in a solver.

For our experiments we picked the QCDCL solver Qute [27], and implemented each of the aforementioned QCDCL variants – QCDCL$^{\text{Uni-Any}}$, QCDCL$^{\text{Exi-Any}}$, and QCDCL$^{\text{Any}}$ (Qute could already run in a mode that corresponds to QCDCL). In order to ensure compliance with the NCC (Definition 4), we needed to adapt some of Qute's internal data structures, and so for

the sake of a fair comparison we also report on a version called QCDCL3: algorithmically plain QCDCL but with the new data structures that are required for the other variants (up to 3 watched literals rather than the usual 2, hence the name).

We evaluated each QCDCL variant on the first 100 formulas from each separation family – `TwinCR`, `MirrorCR`, and Rev(`TwinModEq`$_n$) – running the solver with a time limit of 600 seconds on each individual formula on a machine with two 16-core Intel® Xeon® E5-2683 v4@2.10GHz CPUs and 512GB RAM running Ubuntu 20.04.3 LTS on Linux 5.4.0-48, organizing the computation with the help of GNU Parallel [33].

In Figures 3 and 4 we plot running times of the different QCDCL versions as a function of n. Any gaps in the plotted lines indicate the solver timed out at 600 seconds for that particular formula. In general, the proof complexity results are closely mirrored in solver performance, though there is occasionally a bit of surprise.

In Figure 3, we see that for `TwinCR` the configuration QCDCL$^{\text{Uni-Any}}$ is best and scales reasonably well up to $n = 100$. But there are also gaps – for some reason the solver's heuristics appear to be fooled for some particular formulas and fail to navigate towards the short proof. Overall, QCDCL$^{\text{Uni-Any}}$ manages to solve 87 out the first 100 `TwinCR` formulas. QCDCL$^{\text{Any}}$, which should theoretically be at least as good as QCDCL$^{\text{Uni-Any}}$, comes a distant second and fails to solve anything beyond $n = 16$. QCDCL$^{\text{Exi-Any}}$ appears to be off to a good start, but also quickly loses breath solving nothing after $n = 10$. The two vanilla variants QCDCL and QCDCL3 scale exponentially all the way as they should.

The picture on the related `MirrorCR` formulas (Figure 3 below) is boring in comparison and perfectly corresponds to our theoretical results. The two variants that have short proofs – QCDCL$^{\text{Uni-Any}}$ and QCDCL$^{\text{Any}}$ – are also fast in practice, and everything else is dead exponential.

Finally, Rev(`TwinModEq`) in Figure 4 paint a picture somewhat similar to `TwinCR`, though with a different set of peculiarities. The best variant is QCDCL$^{\text{Exi-Any}}$, and unlike QCDCL$^{\text{Uni-Any}}$ on `TwinCR`, it solves all formulas up to $n = 100$ very fast. The second best is QCDCL$^{\text{Any}}$, but once again it drops out relatively early (last solved is $n = 26$) in spite of its theoretical superiority. An interesting thing seems to happen to QCDCL$^{\text{Uni-Any}}$, which appears to be helplessly off to an exponential path, but somehow recovers and solves $n = 15, 16$ fast, only to completely drop out afterwards. The two vanilla variants QCDCL and QCDCL3 are again dead exponential, as they should be.

The recurring theme in Figures 3 and 4 is that the theoretically strongest system QCDCL$^{\text{Any}}$ is outperformed by the specialized version for each formula type. One appealing explanation would be that the specialized systems QCDCL$^{\text{Uni-Any}}$ and QCDCL$^{\text{Exi-Any}}$ profit from their ability to guarantee learning asserting clauses and cubes respectively. But this does not appear to be the real reason: QCDCL$^{\text{Any}}$ also (like QCDCL$^{\text{Uni-Any}}$) learns almost exclusively asserting clauses on `TwinCR` (96% on average, more than 99% in over 70% of cases), and similarly QCDCL$^{\text{Any}}$ (like QCDCL$^{\text{Exi-Any}}$) learns almost exclusively asserting cubes on Rev(`TwinModEq`) (98% on average, more than 99% in over 70% of cases). Thus, the advantage of the specialized systems is unlikely to be explicable solely by the *quantity* of asserting constraints, but rather by their *quality*. This is also supported by the erratic performance of several of the variants on both `TwinCR` and Rev(`TwinModEq`) – it appears that in many cases, there is a short run, but it is hard for the solver to discover. Investigating this properly will probably require much more detailed experiments, which we leave to future work.

7 Conclusion

We have laid the theoretical foundations for new flavours of QCDCL with the ability to ignore all quantification order for decisions. In this paper we focused on proof complexity, showing exponential advantage for the new systems over vanilla QCDCL. We complemented this with a proof-of-concept implementation in Qute, which validates the feasibility of our approach. Our preliminary experiments on crafted formulas already raise some interesting questions about poor solver performance on theoretically easy formulas.

In future work we plan to significantly advance on the practical front. This means polishing and possibly improving the implementation technically, and performing an extensive experimental evaluation: on industrial formulas from recent QBF Evaluations (both PCNF and circuits), combining the approaches presented here with other state-of-the-art techniques like Qute's native dependency learning (and possibly dependency schemes), and using preprocessors, to name a few. We would also like to dive deeper into the analysis of how learning asserting constraints affects solver performance.

References

1 Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *J. Artif. Intell. Res.*, 40:353–373, 2011.

2 Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF certification and its applications. *Form. Methods Syst. Des.*, 41(1):45–65, 2012.

3 Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res. (JAIR)*, 22:319–351, 2004. doi:10.1613/jair.1410.

4 Olaf Beyersdorff, Joshua Blinkhorn, and Luke Hinde. Size, cost, and capacity: A semantic technique for hard random QBFs. *Logical Methods in Computer Science*, 15(1), 2019.

5 Olaf Beyersdorff and Benjamin Böhm. QCDCL with cube learning or pure literal elimination - what is best? *Electron. Colloquium Comput. Complex.*, page 131, 2021. URL: https://eccc.weizmann.ac.il/report/2021/131.

6 Olaf Beyersdorff and Benjamin Böhm. Understanding the relative strength of QBF CDCL solvers and QBF resolution. In *Proc. Innovations in Theoretical Computer Science (ITCS)*, pages 12:1–12:20, 2021.

7 Olaf Beyersdorff, Mikolás Janota, Florian Lonsing, and Martina Seidl. Quantified boolean formulas. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications, pages 1177–1221. IOS Press, 2021.

8 Benjamin Böhm and Olaf Beyersdorff. Lower bounds for QCDCL via formula gauge. In Chu-Min Li and Felip Manyà, editors, *Theory and Applications of Satisfiability Testing – SAT 2021*, pages 47–63, Cham, 2021. Springer International Publishing.

9 Sam Buss and Jakob Nordström. Proof complexity and SAT solving. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications, pages 233–350. IOS Press, 2021.

10 Marco Cadoli, Andrea Giovanardi, and Marco Schaerf. An algorithm to evaluate quantified Boolean formulae. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference (AAAI)*, pages 262–267, 1998.

11 Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

12 Enrico Giunchiglia, Paolo Marin, and Massimo Narizzano. Reasoning with quantified Boolean formulas. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 761–780. IOS Press, 2009.

13 Marijn Heule. Proofs of unsatisfiability. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability, 2nd edition*, Frontiers in Artificial Intelligence and Applications, pages 635–668. IOS press, 2021.

14 Marijn J. H. Heule, Martina Seidl, and Armin Biere. Solution validation and extraction for QBF preprocessing. *J. Autom. Reason.*, 58(1):97–125, 2017.

15 Holger H. Hoos, Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Portfolio-based algorithm selection for circuit qbfs. In John N. Hooker, editor, *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*, volume 11008 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 2018. doi:10.1007/978-3-319-98334-9_13.

16 Mikolás Janota. On Q-Resolution and CDCL QBF solving. In *Proc. International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 402–418, 2016.

17 Mikolás Janota and Joao Marques-Silva. Expansion-based QBF solving versus Q-resolution. *Theor. Comput. Sci.*, 577:25–42, 2015.

18 Mikoláš Janota and Joao Marques-Silva. An Achilles' heel of term-resolution. In Eugénio Oliveira, João Gama, Zita Vale, and Henrique Lopes Cardoso, editors, *Progress in Artificial Intelligence*, pages 670–680, Cham, 2017. Springer International Publishing.

19 Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified Boolean formulas. *Inf. Comput.*, 117(1):12–18, 1995.

20 Florian Lonsing. *Dependency Schemes and Search-Based QBF Solving: Theory and Practice*. PhD thesis, Johannes Kepler University Linz, 2012.

21 Florian Lonsing and Uwe Egly. DepQBF 6.0: A search-based QBF solver beyond traditional QCDCL. In *Proc. International Conference on Automated Deduction (CADE)*, pages 371–384, 2017.

22 Florian Lonsing and Uwe Egly. Evaluating QBF solvers: Quantifier alternations matter. In *Proc. Principles and Practice of Constraint Programming (CP)*, pages 276–294. Springer, 2018.

23 João P. Marques Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2021.

24 João P. Marques Silva and Karem A. Sakallah. GRASP - a new search algorithm for satisfiability. In *Proc. IEEE/ACM International Conference on Computer-aided Design (ICCAD)*, pages 220–227, 1996. doi:10.1145/244522.244560.

25 M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: engineering an efficient sat solver. In *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*, pages 530–535, 2001. doi:11.1145/378239.379017.

26 Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Combining resolution-path dependencies with dependency learning. In Mikolás Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 306–318. Springer, 2019. doi:10.1007/978-3-030-24258-9_22.

27 Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Dependency learning for QBF. *J. Artif. Intell. Res.*, 65:180–208, 2019.

28 Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider. Long-distance Q-resolution with dependency schemes. *J. Autom. Reasoning*, 63(1):127–155, 2019.

29 Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artif. Intell.*, 175(2):512–525, 2011. doi:10.1016/j.artint.2010.10.002.

30 Marko Samer and Stefan Szeider. Backdoor sets of quantified Boolean formulas. *Journal of Automated Reasoning*, 42(1):77–97, 2009. doi:10.1007/s10817-008-9114-5.

31 Ankit Shukla, Armin Biere, Luca Pulina, and Martina Seidl. A survey on applications of quantified Boolean formulas. In *Proc. IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 78–84, 2019.

32 Friedrich Slivovsky and Stefan Szeider. Soundness of Q-resolution with dependency schemes. *Theoretical Computer Science*, 612:83–101, 2016.

33 Ole Tange. GNU Parallel - The command-line power tool. *login: The USENIX Magazine*, February:42–47, 2011.

34 Marc Vinyals. Hard examples for common variable decision heuristics. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

35 Lintao Zhang, Conor F. Madigan, Matthew W. Moskewicz, and Sharad Malik. Efficient conflict driven learning in Boolean satisfiability solver. In *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 279–285, 2001.

36 Lintao Zhang and Sharad Malik. Conflict driven learning in a quantified Boolean satisfiability solver. In *Proc. IEEE/ACM International Conference on Computer-aided Design (ICCAD)*, pages 442–449, 2002.

A Appendix

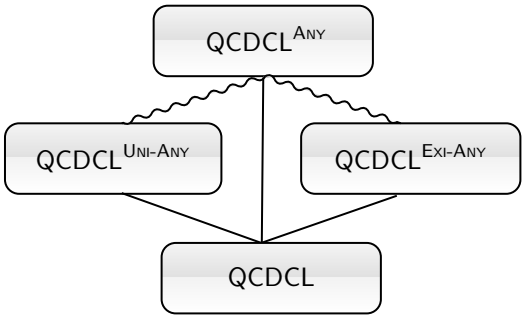

Figure 2 Hasse diagram of the simulation order of QCDCL proof systems. Solid lines represent p-simulations and exponential separations. Waved lines represent p-simulations, for which separations are not known.

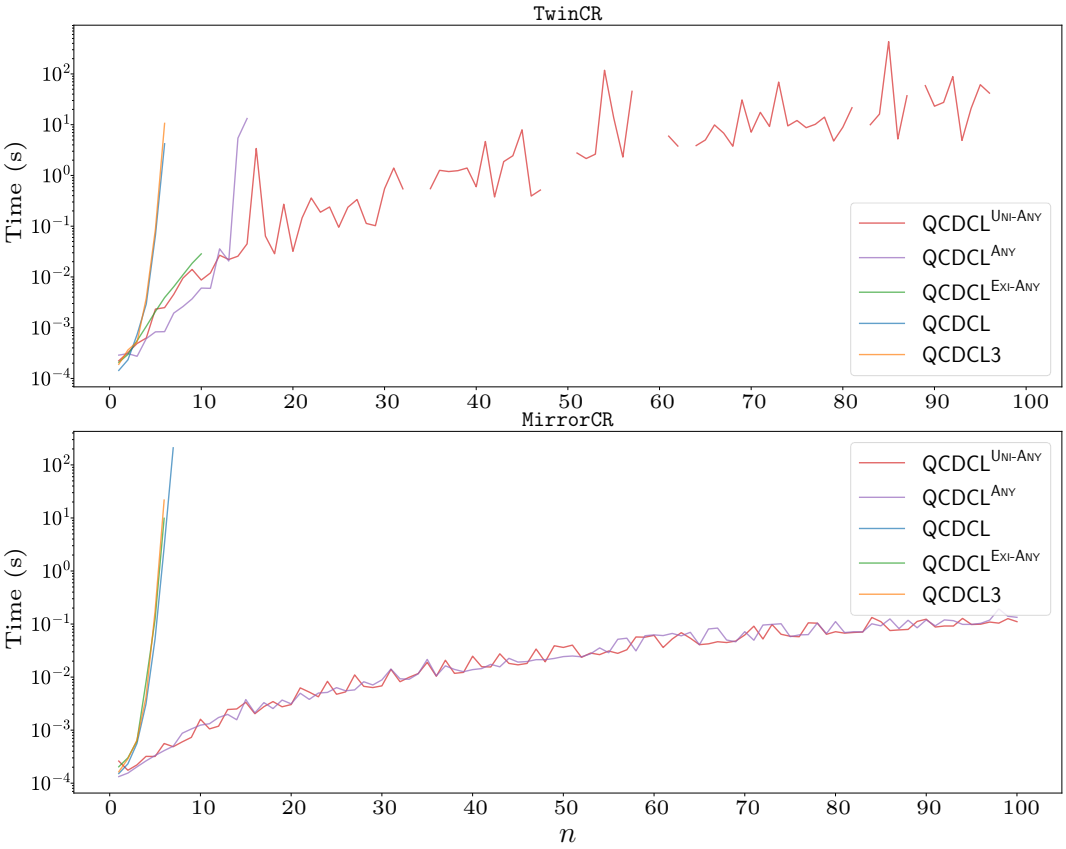

Figure 3 Performance on $\texttt{TwinCR}_n$ (above) and $\texttt{MirrorCR}_n$ (below). Legends are sorted best-to-worst.

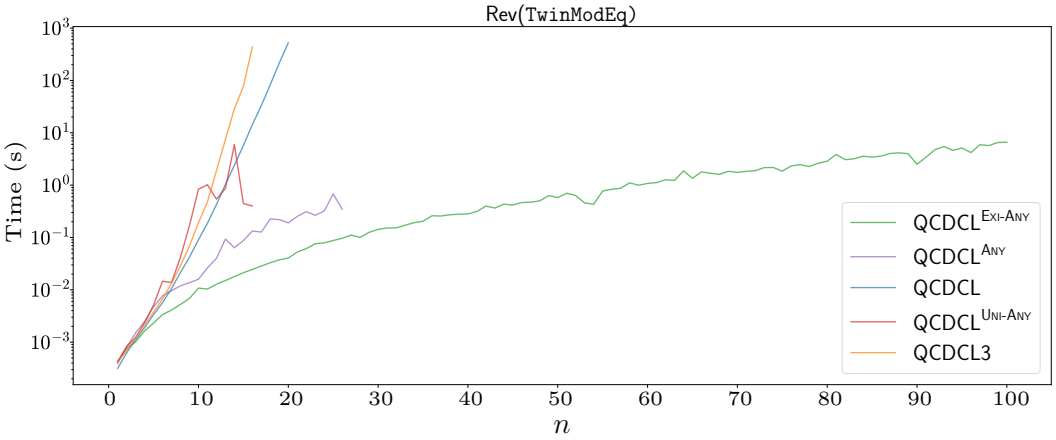

Figure 4 Running time in seconds on $\text{Rev}(\texttt{TwinModEq}_n)$. The legend is sorted from best downwards.