EPiC Series in Computing
Volume 100, 2024, Pages 408–425
Proceedings of 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning

Sometimes Hoarding is Harder than Cleaning: NP-hardness of Maximum Blocked-Clause Addition

Bernardo Subercaseaux

Carnegie Mellon University
Pittsburgh, PA, USA
bersub@cmu.edu

Abstract

Adding blocked clauses to a CNF formula can substantially speed up SAT-solving, both in theory and practice. In theory, the addition of blocked clauses can exponentially reduce the length of the shortest refutation for a formula [17, 19]. In practice, it has been recently shown that the runtime of CDCL solvers decreases significantly for certain instance families when blocked clauses are added as a preprocessing step [10, 22]. This fact is in contrast to, but not in contradiction with, prior results showing that Blocked-Clause Elimination (BCE) is sometimes an effective preprocessing step [14, 15]. We suggest that the practical role of blocked clauses in SAT-solving might be richer than expected. Concretely, we propose a theoretical study of the complexity of Blocked-Clause Addition (BCA) as a preprocessing step for SAT-solving, and in particular, consider the problem of adding the maximum number of blocked clauses of a given arity k to an input formula F. While BCE is a *confluent* process, meaning that the order in which blocked clauses are eliminated is irrelevant, this is not the case for BCA: adding a blocked clause to a formula might *unblock* a different clause that was previously blocked. This order-sensitivity turns out to be a crucial obstacle for carrying out BCA efficiently as a preprocessing step. Our main result is that computing the maximum number of k-ary blocked clauses that can be added to an input formula F is NP-hard for every $k \geq 2$.

1 Introduction

Redundant clauses (with respect to a formula F) are defined by the fact that their addition or removal does not affect the satisfiability of F. Nonetheless, adding or removing redundant clauses can still dramatically affect how hard it is in practice to determine whether F is satisfiable or not. Indeed, CDCL-based solvers often alternate between adding redundant clauses (which can make future deductions easier) and removing redundant clauses (which makes the formula smaller). Figure 1 shows an execution of the Kissat solver [4], and illustrates how redundant clauses fluctuate throughout the solving process, sometimes reaching 80% of the total clauses.

As checking for redundancy is a computationally expensive task in general (the empty clause is redundant with respect to a formula F if and only if F is unsatisfiable!), significant attention has been centered around particular forms of redundancy that can be efficiently checked [23].

In this paper we focus on *blocked* clauses, a particular form of redundant clauses introduced by Kullman in 1999 [19]. A definition of blocked clauses based on *resolution* is presented in Section 3, but for now consider the following:

Definition 1 (Preliminary definition)**.** A clause C is blocked with respect to a formula F on a literal $\ell \in C$ if every clause $C' \in F$ containing $\overline{\ell}$ also contains a literal, different from $\overline{\ell}$, whose negation is in C.

Importantly, blocked clauses are redundant [19], and their purely syntactical definition allows us to efficiently check whether a clause is blocked with respect to a formula F. As a result, given a formula F it is possible to efficiently *eliminate* some, or even all, blocked clauses from it. This process is known as *Blocked-Clause Elimination* (BCE) [15], and can be used as a preprocessing step in modern SAT-solvers such as CaDiCal [3], although we remark that BCE is turned off by default. The efficiency of running BCE relies not only on *clause blockedness* being an efficiently checkable property, but also on the fact that the order in which blocked clauses are eliminated is irrelevant, a property known as *confluence* [15]. In this article, we will study *Blocked-Clause Addition* (BCA), the converse of BCE. In a nutshell, we will use the lack of confluence of BCA to derive hardness results. In particular, our main result is the following:

Theorem (Informal statement)**.** For any $k \geq 2$, given a formula F, it is NP-hard to compute the largest set of k-ary blocked clauses that can be added to F.

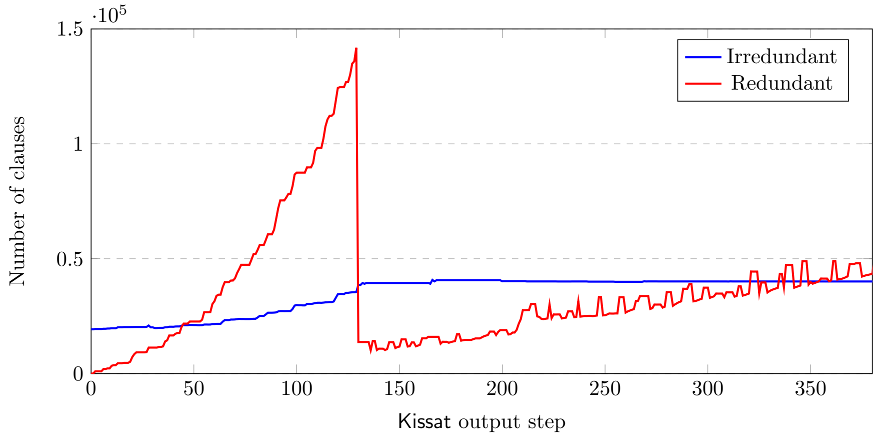

Figure 1: Evolution of the number of redundant and irredundant clauses over a 2-minute run of the Kissat solver [4]. The notion of redundancy used by Kissat corresponds to RUP [18]. The specific instance used for this plot is publicly available at `https://pastebin.com/K6AnWuVx`.

Organization. Section 2 discusses in more detail the role of blocked clauses in (Max)SAT-solving, showing that both BCE and BCA can be helpful depending on the particular application. Then, Section 3 presents the definitions required to state *Maximum* Blocked Clause Addition as a computational problem whose complexity can be analyzed. In Section 4 we introduce BC graphs, the directed graphs that capture the relevance of the order in which blocked clauses can be added to an input formula F. Next, Section 5 presents a partial characterization of BC graphs, describing classes of graphs that can arise as BC graphs. We leverage that characterization to derive hardness results in Section 6, and also show that counting the number of models for a blocked set is #P-hard, thus answering an open question of Heule and Biere [2,12]. Section 7 presents a simple approximation algorithm that achieves a $1/6$-factor of the maximum number of binary blocked clauses that can be added to a formula. Then, Section 8 provides further insight into BC graphs for $k \geq 4$. Finally, we present concluding remarks and open problems in Section 9.

2 Should they Stay or Should they Go?

The seminal paper of Kullmann [19, Lemma 8.10] showed that there are unsatisfiable formulas for which the addition of blocked clauses, even without any new variables, exponentially reduces the length of the shortest resolution refutation. However, when introducing BCE as a general preprocessing technique, Jarvisalo et al. [15, Section 4] remarked that despite the theoretical results of Kullmann, running BCE often resulted in shorter and easier formulas in practice. Reality seems to be more nuanced, as recent empirical evidence has revealed. In 2023, Subercaseaux and Heule determined the *packing chromatic number* of the infinite square grid through SAT-solving, and found that adding merely 85 *blocked clauses* to large formulas provided significant improvements in runtimes and proof sizes [22, Table 3]. Moreover, Fleury and Kaufmann recently showed that BCE was detrimental in 12 out of 15 SAT competitions (2009–2023) [10, Table 1]. We remark that the results reported by Fleury and Kaufmann also suggest that BCA can be beneficial: if the runtime for $F \setminus \textsc{BlockedClauses}(F)$ was longer than for F, that means that taking $G := F \setminus \textsc{BlockedClauses}(F)$, and $G' := G \cup \textsc{BlockedClauses}(F)$, the runtime for G' was shorter than for G, and it is easy to show that any blocked clause w.r.t. F is a blocked clause w.r.t. G, which implies that G' is indeed a potential result of running BCA over G.

Why should blocked clauses go? A simple argument for BCE is that as a rule of thumb, smaller encodings are more efficient. Given that most SAT solvers iterate over clauses in their main loop, each iteration becomes faster if the number of clauses is reduced. In simple words, any clause that is not *helping* the solving process must be *hurting* it. A concrete example in which BCE significantly improved runtimes is the Pythagorean Triples problem, where it removed roughly 50% of the (occurring) variables and 20% of the clauses [14].

Why should we add more? The main reason for adding blocked clauses is that they can help the solver make deductions. We have referred already to theoretical evidence [19] and empirical evidence [10, 22], but we will now discuss a self-contained example illustrating the power of adding blocked clauses. Consider the $\textsc{MinimumVertexCover}$ problem, in which the input is an undirected graph $G = (V, E)$ and an integer $k \geq 0$, and the question is to decide whether there is a set $S \subseteq V$ such that $|S| \leq k$ and for every edge $\{u, v\} \in E$ at least one of u or v is in S. A direct encoding for this problem consists of creating variables x_v for every vertex $v \in V$, representing whether $v \in S$ or not, and then constructing the formula F:

$$(x_u \lor x_v), \quad \text{for every edge} \quad \{u, v\} \in E \qquad \text{(Covering Constraints)}$$

$$\bigwedge_{X \subseteq V, |X| = k+1} \left(\bigvee_{u \in X} \overline{x_u} \right) \qquad \text{(Cardinality Constraints)}$$

Now consider the following graph G, depicted in Figure 2.

Then observe that the following clauses are blocked w.r.t. F: $(\overline{x_1} \lor \overline{x_2})$, $(\overline{x_2} \lor \overline{x_3})$, $(\overline{x_6} \lor \overline{x_7})$. If these clauses are added to F, then most solvers are quickly able to deduce that $(x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2})$

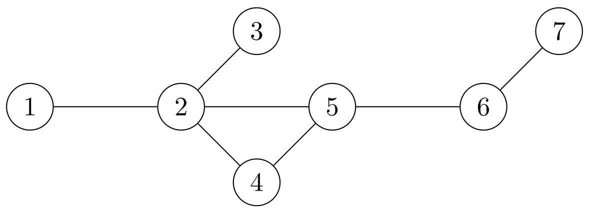

Figure 2: Illustration of the graph G used in the example for the MinimumVertexCover problem.

imply that $\overline{x_1} = x_2$. As the literal x_1 does not appear in any other clauses, the solver can assign $x_1 = 0, x_2 = 1$ without loss of generality, which leads to eliminatating the variable x_1 and the clauses containing it (*Equivalent Literal Substitution* [13]). This simplification would be performed over all vertices of degree 1, therefore implying the solver would have quickly deduced a well-known Vertex Cover preprocessing technique: *"get rid of all vertices of degree* 1 *and add their neighbors to the cover"* [1,9].

3 Preliminaries

We start by introducing some notation and definitions required to formally state the problem at hand. First, we understand a clause as a non-tautological disjunction of literals, which we identify in turn with a set of literals that does not contain both ℓ and $\overline{\ell}$ for any literal ℓ. Given two clauses $A = (a_1 \vee a_2 \vee \ldots \vee a_n)$ and $B = (b_1 \vee b_2 \vee \ldots \vee b_m)$, we use notation $A \vee B$ for the clause $(a_1 \vee a_2 \vee \ldots a_n \vee b_1 \vee b_2 \vee \ldots b_m)$. As a single literal matches our definition of a clause, the previous notation will be used as well for the disjunction of a literal and a clause, identifying $\{\ell\}$ with ℓ. Moreover, we identify CNF formulas with sets of clauses.

Definition 2 (Resolvent)**.** Given clauses $C_1 = \ell \vee C'$ and $C_2 = \overline{\ell} \vee C''$, we define the *resolvent* of C_1, C_2 according to ℓ as $C_1 \otimes_\ell C_2 := C' \vee C''$.

Definition 3 (Blocked literal/clause)**.** A literal ℓ *blocks* a clause $C_1 = \ell \vee C'$ with respect to[1] a formula F when for every clause $C_2 \in F$ of the form $\overline{\ell} \vee C''$, the resolvent $C_1 \otimes_\ell C_2 = C' \vee C''$ is tautological (i.e., contains complementary literals). In such a case we say C is blocked on ℓ w.r.t. F, and also simply that C is blocked w.r.t. F.

The following proposition follows directly from the previous definition and will be used throughout the paper.

Proposition 1 (cf. Heule and Biere [12])**.** *If $G \subseteq F$ and a clause C is blocked w.r.t. F, then C is blocked w.r.t. G.*

From Definition 3 and Proposition 1 it follows that BCE, the process of eliminating all blocked clauses from an input formula F, can be performed in polynomial time [15]. This paper is centered around BCE's opposite, *blocked-clause addition* (BCA), the preprocessing technique introduced by Kullman [19].

Definition 4 (BCA)**.** Given a formula F, and a sequence of distinct clauses $\Gamma = (\Gamma_1, \ldots, \Gamma_n)$ over the variables of F, we say Γ is valid for *blocked-clause addition (BCA)* on F, which we denote by $F \rightsquigarrow_B \Gamma$, if each clause Γ_i is blocked w.r.t. $F \cup \{\Gamma_j \mid j \in \{1, \ldots, i-1\}\}$.

[1]We will use abbreviation *w.r.t.* from now on.

We now have the notation and definitions required to state the computational problems at the heart of this paper, followed by our main result.

PROBLEM:	k-BCA
INPUT:	A formula F and an integer t.
OUTPUT:	Yes if there is a sequence Γ of t clauses of arity k over the variables of F such that $F \rightsquigarrow_B \Gamma$, and No otherwise.

Our main result, listed next, implies that blocked-clause addition is computationally hard to maximize.

Theorem 2. *k-BCA is NP-complete for every $k \geq 2$.*

We remark that the NP-hardness of k-BCA does not directly imply the NP-hardness of k'-BCA for $k' > k$. Indeed, we will have to explicitly prove a technical *lifting* lemma in Section 6 to show that this is the case.

4 BC Graphs and Related Problems

Given a formula F, we denote by $\mathsf{BC}(F)$ the set of clauses that are blocked w.r.t. F and use only variables from F. Moreover, we denote by $\mathsf{BC}_k(F)$ the restriction of $\mathsf{BC}(F)$ to clauses of arity k. Naturally, not all clauses in $\mathsf{BC}(F)$ can belong to a valid BCA sequence on F, as adding a clause $C_1 \in \mathsf{BC}(F)$ can make a clause $C_2 \in \mathsf{BC}(F)$ not blocked anymore. For example, if $F = (x_1 \lor x_2) \land (x_2 \lor x_3)$, then $C_1 = (x_1 \lor \overline{x_3}) \in \mathsf{BC}(F)$ and $C_2 = (\overline{x_1} \lor \overline{x_2}) \in \mathsf{BC}(F)$, but C_2 is not blocked w.r.t. $F \cup \{C_1\}$. The following definitions capture the phenomenon at hand.

Definition 5 (Prevention)**.** Given a formula F, and two clauses $C_1, C_2 \in \mathsf{BC}(F)$. We say the addition of C_1 *prevents* the addition of C_2 when $C_2 \notin \mathsf{BC}(F \cup \{C_1\})$. We denote by $C_1 \rightsquigarrow_p C_2$ the fact that C_1 prevents the addition of C_2.

Definition 6 (BC graph)**.** Given a formula F, we define its *BC graph* $G_{\mathsf{BC}(F)}$ as the directed graph with vertex-set $\mathsf{BC}(F)$, and directed edges $C_1 \to C_2$ when $C_1 \rightsquigarrow_p C_2$. Naturally, $G_{\mathsf{BC}_k(F)}$ denotes the subgraph induced by $\mathsf{BC}_k(F)$.

An example of a BC graph is illustrated in Figure 3. In order to prove Theorem 2 we will only consider BC sets where each clause is blocked on exactly one literal.

Definition 7 (Singly-blocked clauses)**.** A clause C is *singly blocked* w.r.t. a formula F if there is exactly one literal ℓ such that C is blocked on ℓ w.r.t. F.

Remark 1. In the context of singly-blocked clauses, prevention can be seen more easily by the following characterization. If C_1 is singly blocked on ℓ w.r.t. F, then $C_2 \rightsquigarrow_p C_1$ if and only if $\overline{\ell} \in C_2$ and $C_1 \otimes_\ell C_2$ is not tautological.

Note that the definition of BCA does not require that the clauses in the sequence Γ to be singly blocked; we will show that, even in formulas where every clause in their BC set is singly blocked, the BCA problem is NP-hard, which will imply the general hardness. Let us immediately see why we consider singly-blocked clauses.

Lemma 3. *Given a formula F, and a sequence of singly-blocked clauses $C_1, \ldots, C_t$ from $\mathsf{BC}(F)$, we have $F \rightsquigarrow_B (C_1, \ldots, C_t)$ if and only if $(C_i, C_j) \notin E(G_{\mathsf{BC}(F)})$ for every $1 \leq i < j \leq t$.*

Formula F

$(x_1 \lor \overline{x_3})$
$(x_3 \lor x_4)$
$(\overline{x_5} \lor x_4)$
$(x_1 \lor \overline{x_2})$
$(x_1 \lor x_2 \lor x_4 \lor \overline{x_5})$
$(x_1 \lor x_4 \lor x_5)$
$(\overline{x_1} \lor \overline{x_2})$
$(x_2 \lor x_3 \lor x_4 \lor \overline{x_5})$
$(x_3 \lor \overline{x_4})$
$(x_3 \lor \overline{x_4} \lor \overline{x_5})$

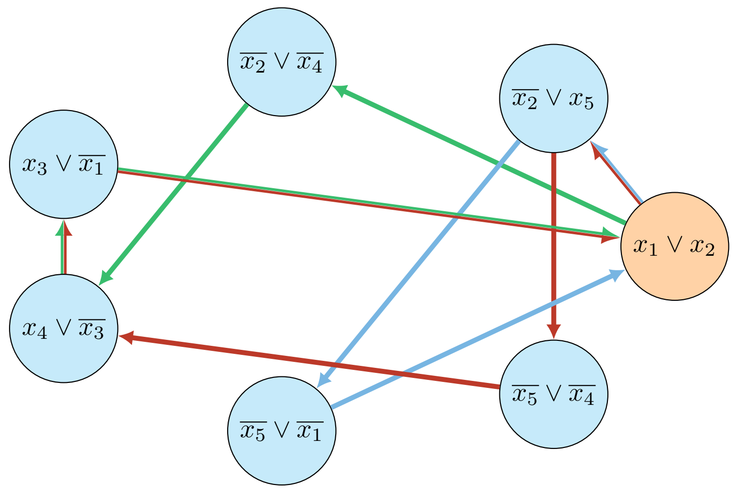
$\overline{x_2} \lor \overline{x_4}$
$\overline{x_2} \lor x_5$
$x_3 \lor \overline{x_1}$
$x_1 \lor x_2$
$x_4 \lor \overline{x_3}$
$\overline{x_5} \lor \overline{x_4}$
$\overline{x_5} \lor \overline{x_1}$

Figure 3: Illustration of the BC graph restricted to $k = 2$ for a formula F. Directed cycles have been colored, and notice that edges that belong to multiple directed cycles receive the colors corresponding to all of them. A feedback vertex set of size 1 is colored in light orange, thus implying by Lemma 6 that there is a valid 2-BCA sequence of length $7 - 1 = 6$ for F.

Proof. For the forward direction assume $F \rightsquigarrow_B (C_1, \dots, C_t)$ and let $1 \leq i < j \leq t$. By definition of $\rightsquigarrow_B$, the clause C_j is blocked w.r.t. $F \cup \{C_1, \dots, C_i, \dots, C_{j-1}\}$, from where using Proposition 1 we deduce that C_j is blocked w.r.t. $F \cup \{C_i\}$, and thus $(C_i, C_j) \notin E(G_{\mathsf{BC}(F)})$.

For the backward direction we assume $(C_i, C_j) \notin E(G_{\mathsf{BC}(F)})$ for every $1 \leq i < j \leq t$ and then proceed inductively. The base case $F \rightsquigarrow_B (C_1)$ is immediate as $C_1 \in \mathsf{BC}(F)$. For the inductive case, assume $F \rightsquigarrow_B (C_1, \dots, C_{j-1})$, and let us show that $F \rightsquigarrow_B (C_1, \dots, C_{j-1}, C_j)$. Using the inductive hypothesis it only remains to show that C_j is blocked w.r.t. $F \cup \{C_1, \dots, C_{j-1}\}$. As C_j is singly blocked on a literal $\ell(C_j)$ w.r.t F, it only remains to show that C_j is blocked on ℓ w.r.t. $\{C_1, \dots, C_{j-1}\}$. Assume, expecting a contradiction, that C_j is not blocked w.r.t. $\{C_1, \dots, C_{j-1}\}$. Then, there must be a clause $C_i \in \{C_1, \dots, C_{j-1}\}$ such that C_i contains $\overline{\ell(C_j)}$ and $C_i \otimes_{\ell(C_j)} C_j$ is not tautological. This implies, by Remark 1, that $C_i \rightsquigarrow_p C_j$, which contradicts the assumption that $(C_i, C_j) \notin E(G_{\mathsf{BC}(F)})$. $\square$

Example 1. For an example illustrating what could happen without the "singly-blocked" restriction, consider the following scenario:

- $F := (x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_4)$.
- $C_1 := (\overline{x_1} \lor \overline{x_2} \lor \overline{x_4})$ is blocked on all $\overline{x_1}, \overline{x_2}$ and $\overline{x_4}$ w.r.t. F.
- $C_2 := (\overline{x_1} \lor x_2 \lor x_3)$ is blocked on x_2 and x_3 w.r.t. F.
- $C_3 := (x_1 \lor \overline{x_2} \lor \overline{x_3})$ is blocked on x_1, $\overline{x_2}$, and $\overline{x_3}$ w.r.t. F.
- $C_4 := (\overline{x_2} \lor \overline{x_3} \lor x_4)$ is blocked on all $\overline{x_2}, \overline{x_3}$ and x_4 w.r.t. F.
- Neither C_2, C_3 nor C_4 prevent C_1.
- C_1 is blocked on $\overline{x_4}$ w.r.t. $F \cup \{C_2, C_3\}$, but is not blocked w.r.t. $F \cup \{C_2, C_3, C_4\}$.

In this scenario, the prevention relation is not enough to properly capture valid sequences for BCA, as neither C_2, C_3 nor C_4 can prevent C_1 but their union does. This is due to C_1 being blocked on multiple literals, as it is only when each of such literals is no longer blocked that C_1 as a whole is no longer blocked. This can be achieved by the set $\{C_2, C_3, C_4\}$, but not by any clause C_i alone.

Next, we show that BCA corresponds naturally to a graph problem when every clause in $\mathsf{BC}_k(F)$ is singly blocked. Given a directed graph $G = (V, E)$, we say a sequence $v_1, \ldots, v_n$ of its vertices is *undominated* if for every $1 \leq i < j \leq n$ we have $(v_i, v_j) \notin E$. In other words, a sequence S is undominated when there are no edges going "forward" in S. This leads to the following computational problem.

PROBLEM:	UndominatedSequence
INPUT:	A directed graph $G = (V, E)$, and an integer $t \geq 1$.
OUTPUT:	Yes if there is a sequence $S = v_1, \ldots, v_t \in V^t$ such that for every $1 \leq i < j \leq t$ we have $(v_i, v_j) \notin E$. No otherwise.

Lemma 4. *Given a formula F, an integer $t \geq 1$, and an integer $k \geq 2$ such that every clause in $\mathsf{BC}_k(F)$ is singly blocked, we have that (F, t) is a Yes-instance of k-BCA if and only if $(G_{\mathsf{BC}_k(F)}, t)$ is a Yes-instance of* UndominatedSequence.

Proof. Immediate from Lemma 3. □

To make progress now, we will require another problem over directed graphs. Consider the following problem, proven to be NP-hard in Karp's seminal paper [16].[2]

PROBLEM:	FeedbackVertexSet
INPUT:	A directed graph $G = (V, E)$, and an integer $t \geq 1$.
OUTPUT:	Yes if there is a set of t vertices $S \subseteq V$ such that their removal (including the edges touching them) leaves G without directed cycles. No otherwise.

In general, we will say that a subset $S \subseteq V$ is a *feedback vertex set* of G if $G[V(G) \setminus S]$ is acyclic. It turns out that UndominatedSequence and FeedbackVertexSet are complementary problems, as the next lemma shows.

Lemma 5. *Given a directed graph G and a subset S of its vertices, the following statements are equivalent: (i) there exists an undominated sequence O_S consisting of an ordering of the elements of S, (ii) $G[S]$ is acyclic, (iii) $V(G) \setminus S$ is a feedback vertex set of G.*

Proof. To prove that (i) implies (ii), assume $O_S = v_1, \ldots, v_{|S|}$ is an undominated sequence, and let $\prec$ be the total order over S defined by $v_i \prec v_j \iff i > j$. By the definition of undominated sequence, every edge $(v_i, v_j) \in E(G[S])$ must hold that $i > j$, and thus $v_i \prec v_j$. This implies that $G[S]$ is a subgraph of $(S, \prec)$, and given $(S, \prec)$ is an acyclic graph as it corresponds to an order, we conclude that $G[S]$ is acyclic. To see that (ii) and (iii) are equivalent, note that $S = V(G) \setminus (V(G) \setminus S)$. Finally, to prove that (ii) implies (i), assume $G[S]$ is

[2] We remark that although some authors refer to FeedbackVertexSet as a problem over undirected graphs, and use something like DirectedFeedbackVertexSet for the version over directed graphs, we adhere to Karp's initial presentation, which is over directed graphs.

acyclic, and therefore it has a topological ordering $\pi = v_1, \ldots, v_{|S|}$, which by definition of topological ordering implies that all edges $(v_i, v_j) \in E(G[S])$ have $i < j$. By reverting π into $O_S = v'_1, \ldots, v'_{|S|}$ through the transformation $v'_i := v_{|S|-i+1}$, we obtain an undominated sequence, as now all edges $(v'_i, v'_j) \in E(G[S])$ have $i > j$. □

Combining Lemma 4 and Lemma 5, we immediately have the following.

Lemma 6. *Given a formula F, an integer $t \geq 1$, and an integer $k \geq 2$ such that every clause in $\mathsf{BC}_k(F)$ is singly blocked, we have that (F, t) is a Yes-instance of k-BCA if and only if $(G_{\mathsf{BC}_k(F)}, |V(G_{\mathsf{BC}_k(F)})| - t)$ is a Yes-instance of* FeedbackVertexSet.

Proof. Using Lemma 4 it suffices to show that $G_{\mathsf{BC}_k(F)}$ has an undominated sequence of length (at least) t if and only if it has a feedback vertex set of size $|V(G_{\mathsf{BC}_k(F)})| - t$. Indeed, if $G_{\mathsf{BC}_k(F)}$ has an undominated sequence O of length t, and we denote $S_O := \{v \in V(G_{\mathsf{BC}_k(F)}) \mid v \in O\}$, then O is an ordering of S_O and by Lemma 5 we have that $V(G_{\mathsf{BC}_k(F)}) \setminus S_O$ is a feedback vertex set of $G_{\mathsf{BC}_k(F)}$, whose length is exactly $|V(G_{\mathsf{BC}_k(F)})| - t$. The other direction is immediate by using Lemma 5 again. □

It is known that FeedbackVertexSet can be solved in polynomial time for restricted classes of graphs, which in turn implies that UndominatedSequence can also be solved in polynomial time for those classes, and therefore that BCA can be solved in polynomial time as long as $G_{\mathsf{BC}(F)}$ belongs to any of said classes. The ISGCI project [7] lists over 600 such classes of graphs. Furthermore, we remark that efficient parameterized algorithms deciding in time $n^{O(1)} \cdot f(t)$ whether a directed graph G has a feedback vertex set of size at most t are well-known [6], which implies that on formulas F where all but a small number of clauses of $\mathsf{BC}_k(F)$ can be added in a BCA sequence, then such a sequence can be computed efficiently.

On the other direction, however, we cannot yet use the hardness of FeedbackVertexSet (or of UndominatedSequence) to show hardness for k-BCA; such a reduction would require taking a directed graph F as input and then constructing a formula F_G such that $\mathsf{BC}_k(F_G) = G$. Unfortunately, not all directed graphs can arise as BC graphs. Next, in Section 5, we will study the class of graphs that can arise as BC graphs, showing that it is a rich enough class to derive hardness results.

5 Characterization of BC Graphs

In this section, we provide a partial characterization of $\mathsf{BC}(F)_k$ graphs, which will be used to derive complexity results. First, we attack the case of $k = 2$, which requires the following definition.

Definition 8 (k-subdivision)**.** For a positive integer $k \geq 2$, the k-subdivision of a graph G (directed or undirected) consists of replacing every edge $(u, v) \in E(G)$ by a path of length k from u to v.

An example of the 2-subdivision of a directed graph is illustrated in Figure 4.

Lemma 7. *Given any graph G which is the 2-subdivision of a graph H, one can compute in polynomial time (w.r.t. $|G|$) a formula F such that $G_{\mathsf{BC}_2(F)} = G$*[3]*, and moreover, every clause in $\mathsf{BC}_2(F)$ is singly blocked.*

[3]To be fully precise, we mean that $G_{\mathsf{BC}(F)}$ is isomorphic to G, as the vertices of $G_{\mathsf{BC}(F)}$ are clauses, whereas G is an arbitrary graph. For the purpose of this paper we will not make this distinction.

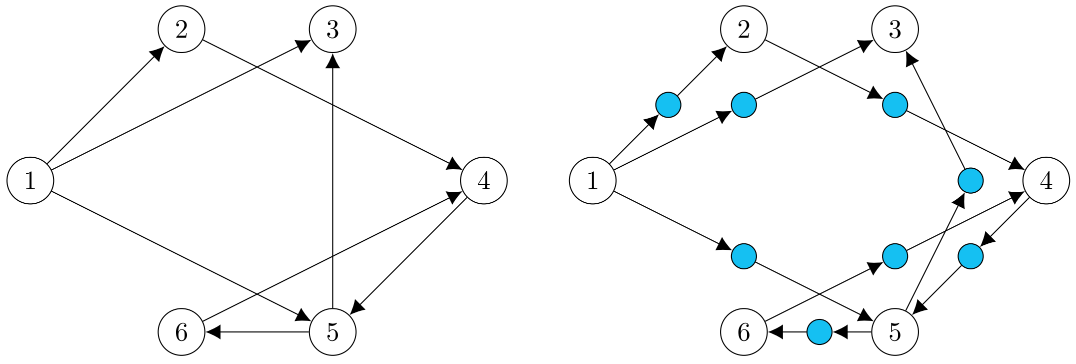

Figure 4: Example of the 2-subdivision of a graph G.

Proof. Let $G = (V_G, E_G)$ be the 2-subdivision of a graph $H = (V_H, E_H)$. Then, we create variables x_u for every $u \in V_H$, and a variable w_u for every $u \in V_H$. For every $u \in V_H$, create the clause $C_u := (x_u \vee \overline{w_u})$. Note now that for every edge $(u, v) \in E_H$, there is exactly one vertex $w \in V_G$ such that $(u, w), (w, v) \in E_G$. We thus can a create well-define clause $W_{u,v} := (w_u \vee \overline{x_v})$ for every edge $(u, v) \in E_H$. We will construct F so that

$$\mathsf{BC}_2(F) = \{C_u \mid u \in V_H\} \cup \{W_{u,v} \mid (u, v) \in E_H\}.$$

To achieve this, we create two new variables y and z, and then add the following sets of clauses to the formula F:

1. (xy-clauses) For every vertex $u \in V_H$, add to F the clauses $(x_u \vee y), (x_u \vee \overline{y})$.

2. (wy-clauses) For every vertex $u \in V_H$, add to F the clauses $(w_u \vee y), (w_u \vee \overline{y})$.

3. (yz-clauses) Add to F the clauses $(y \vee z), (y \vee \overline{z}), (\overline{y} \vee z), (\overline{y} \vee \overline{z})$.

4. (xw-clauses) For every vertex $u \in V_H$, add to F the clause $(\overline{x_u} \vee w_u)$.

5. (out-clauses) For every vertex $u \in V_H$, if $v_1, \ldots, v_m$ are the out-neighbors of u, then add to F the clause $(\overline{w_u} \vee x_{v_1} \vee \ldots \vee x_{v_m})$. Naturally, if u had out-degree 0, the resulting clause will simply be $(\overline{w_u})$.

First, observe that every clause C_u indeed belongs to $\mathsf{BC}_2(F)$ as C_u is blocked on x_u, which can be seen as the only clauses in F where $\overline{x_u}$ appears are the xw-clauses, and their resolvent is tautological by construction. Similarly, every clause $W_{u,v}$ belongs to $\mathsf{BC}_2(F)$ as $W_{u,v}$ is blocked on w_u due to the out-clauses. We now claim that there are no other clauses in $\mathsf{BC}_2(F)$. Indeed, let $C \in \mathsf{BC}_2(F)$ be arbitrary, and then see how the following steps prove that $C = C_u$ for some vertex u or $W_{u,v}$ for some edge $(u, v) \in E_H$.

1. The clause C cannot be blocked on y, as otherwise $C \otimes_y (\overline{y} \vee z)$ being tautological would imply $\overline{z} \in C$, but then $C \otimes_y (\overline{y} \vee \overline{z})$ being tautological would imply $z \in C$, which is a contradiction. Similarly, C cannot be blocked on $\overline{y}$, $\overline{z}$, or z.

2. The clause C cannot be blocked on $\overline{x_u}$ for any $u \in V_H$, as otherwise $C \otimes_{\overline{x_u}} (x_u \vee y)$ being tautological would imply $\overline{y} \in C$, but then $C \otimes_{\overline{x_u}} (x_u \vee \overline{y})$ being tautological would imply $y \in C$, which is a contradiction.

3. The clause C cannot be blocked on $\overline{w_u}$ for any $u \in V_H$, as otherwise $C \otimes_{\overline{w_u}} (w_u \lor y)$ being tautological would imply $\overline{y} \in C$, but then $C \otimes_{\overline{w_u}} (w_u \lor \overline{y})$ being tautological would imply $y \in C$, which is a contradiction.

4. From the previous steps, we either have that C is blocked on x_u for some $u \in V_H$, or on w_u for some $u \in V_H$.

5. If C is blocked on x_u, then because of the xw-clauses, we have that C must contain $\overline{w_u}$, and thus $C = C_u$, given $|C| = 2$. Note that C_u is singly blocked due to step 3.

6. If C is blocked on w_u, then given that $|C| = 2$, there is another literal $\ell \in C$. Because of the out-clause for u, we have that ℓ must be one literal of the form x_v for some v that is an out-neighbor of u, and thus $C = W_{u,v}$. Note that $W_{u,v}$ is singly blocked due to step 2.

We have proved that $\mathsf{BC}_2(F) = \{C_u \mid u \in V_H\} \cup \{W_{u,v} \mid (u,v) \in E_H\}$, and thus $|\mathsf{BC}_2(F)| = |V_G|$. Note now that every edge e in E_G corresponds to either (u,w) or (w,v) with $w \notin V_H$. If $e = (u,w)$, and its associated edge in E_H is (u,v), then by Remark 1 we have $C_u \rightsquigarrow_p W_{u,v}$. Similarly, if $e = (w,v)$, and its associated edge in E_H is (u,v), then e we have by Remark 1 that $W_{u,v} \rightsquigarrow_p C_v$. We thus have that the mapping $\varphi : V_G \to \mathsf{BC}_2(F)$ determined by

$$\varphi(a) = \begin{cases} C_u, & \text{if } a = u \text{ for some } u \in V_H, \\ W_{u,v}, & \text{if } a = (u,v) \text{ for some } e = (u,v) \in V_G \setminus V_H, \end{cases}$$

is an isomorphism between $G_{\mathsf{BC}_2(F)}$ and G, and thus concludes the proof. □

We now prove a technical lemma that extends the previous characterization to every $k \geq 3$. We will use the following definition.

Definition 9 (Minimally blocked)**.** We say a clause C is minimally blocked w.r.t. a formula F if C is blocked w.r.t. F and for every literal $\ell \in C$, the clause $C \setminus \{\ell\}$ is not blocked w.r.t. F.

Remark 2. Every clause $C \in \mathsf{BC}_2(F)$ for a formula F constructed as in Lemma 7 is minimally-blocked.

Lemma 8 (Lifting)**.** *If $\mathcal{C}$ is a class of graphs such that for any graph $G \in \mathcal{C}$ one can compute in polynomial time a formula F such that $G_{\mathsf{BC}_k(F)} = G$ and every clause in $\mathsf{BC}_k(F)$ is both singly blocked and minimally blocked, then for any graph $G \in \mathcal{C}$ one can compute in polynomial time a formula F' such that $G_{\mathsf{BC}_{k+1}(F')} = G$, with every clause in $\mathsf{BC}_{k+1}(F')$ also being singly blocked and minimally blocked.*

Proof. We can assume that $k \geq 2$, as if $k = 1$ then $G_{\mathsf{BC}_k}(F)$ is a collection of isolated vertices and the result is trivial. Let $G \in \mathcal{C}$ be a graph satisfying the hypotheses of the lemma. As a first step, the hypotheses allow us to construct in polynomial time a formula F such that $G_{\mathsf{BC}_k(F)} = G$. For each clause $C \in \mathsf{BC}_k(F)$, we use notation $\ell(C)$ for the only literal $\ell \in C$ such that C is blocked on ℓ w.r.t. F. Now, let y and z be new variables that are not present in F. With these, we can construct F' by the following procedure:

1. (F-clauses) Add to F' every clause of F.

2. (yz-clauses) Add to F' the clauses $(y \lor z), (y \lor \overline{z}), (\overline{y} \lor z), (\overline{y} \lor \overline{z})$.

3. (Lifting clauses) Add to F' the clauses $(\overline{\ell} \vee \overline{y} \vee \overline{z}), (\overline{\ell} \vee \overline{y} \vee z)$ for any literal ℓ that occurs in F or whose negation occurs in F.

Note immediately that this construction takes polynomial time, and thus we only have to focus on correctness. Before the details, let us give a high-level overview of the proof. We will prove that $\mathsf{BC}_{k+1}(F')$ consists exactly of the clauses of the form $C' := (C \vee y)$ for $C \in \mathsf{BC}_k(F)$. Intuitively, the *lifting clauses* will force y to be in any clause $C' \in \mathsf{BC}_{k+1}(F')$. From there and the fact that no clause can be blocked on y w.r.t. in F' because of the yz-clauses, we will derive that $C_1 \rightsquigarrow_p C_2$ w.r.t. F if and only if $(C_1 \vee y) \rightsquigarrow_p (C_2 \vee y)$ w.r.t. F'. This will imply that $G_{\mathsf{BC}_{k+1}(F')}$ and $G_{\mathsf{BC}_k(F)}$ are isomorphic, the desired result. Let us now proceed with the proof, starting with an almost trivial but useful fact.

Fact 1. If a clause C_1 is blocked on a literal ℓ w.r.t. F, then the clause $C_1 \vee C_2$ is also blocked on ℓ w.r.t. F for any clause C_2.

Proof of Fact 1. Follows immediately from the fact that $C_1 \otimes_\ell C' \subseteq (C_1 \vee C_2) \otimes_\ell C'$, where C' is any clause containing $\overline{\ell}$. □

With this, we are ready to prove one direction of the characterization of $\mathsf{BC}_{k+1}(F')$.

Claim 1. For every $C \in \mathsf{BC}_k(F)$ we have $(C \vee y) \in \mathsf{BC}_{k+1}(F')$

Proof of Claim 1. It suffices to argue that $(C \vee y)$ is blocked w.r.t. all (i) F, (ii) the yz-clauses, and (iii) the lifting clauses. For (i), this follows by C being blocked w.r.t. F and Fact 1. For (ii), this follows from the fact that $\ell(C) \notin \{y, \overline{y}, z, \overline{z}\}$, as the variables y, z do not occurr in F, and for (iii), observe that $(C \vee y) \otimes_{\ell(C)} (\overline{\ell(C)} \vee \overline{y} \vee \overline{z})$ is tautological as it contains both y and $\overline{y}$, and the same applies to $(C \vee y) \otimes_{\ell(C)} (\overline{\ell(C)} \vee \overline{y} \vee z)$. □

Before we show the second and harder direction, we will another simple fact about blocked clauses. For any clause C, and formula $\tilde{F}$, let $C{\downarrow_{\tilde{F}}}$ denote the clause C restricted to the variables of $\tilde{F}$.

Fact 2. Let $\tilde{F}$ be any formula and let C be a clause that is blocked w.r.t. $\tilde{F}$ on a literal $\ell \in C{\downarrow_{\tilde{F}}}$. Then $C{\downarrow_{\tilde{F}}}$ is also blocked on ℓ w.r.t. $\tilde{F}$.

Proof of Fact 2. Let C' be any clause in $\tilde{F}$ that contains $\overline{\ell}$, and let $C^\star := C \setminus C{\downarrow_{\tilde{F}}}$. By hypothesis, we have that $C \otimes_\ell C'$ is tautological, and as $C \otimes_\ell C' = C^\star \vee (C{\downarrow_{\tilde{F}}} \otimes_\ell C')$, we have that $C^\star \vee (C{\downarrow_{\tilde{F}}} \otimes_\ell C')$ is tautological. This implies in turn that $(C{\downarrow_{\tilde{F}}} \otimes_\ell C')$ is tautological, since the set of variables of $C^\star$ is disjoint from that of $C{\downarrow_{\tilde{F}}}$ and that of C'. We thus conclude that $C{\downarrow_{\tilde{F}}}$ is blocked on ℓ w.r.t. $\tilde{F}$. □

Claim 2. Every clause $C' \in \mathsf{BC}_{k+1}(F')$ is of the form $C' = (C \vee y)$ for some $C \in \mathsf{BC}_k(F)$. Moreover, C' is minimally blocked w.r.t. F', and singly blocked on $\ell(C)$, the same literal on which C is singly blocked w.r.t. F.

Proof of Claim 2. Let C' be an arbitrary clause in $\mathsf{BC}_{k+1}(F')$. By the same argument as in Lemma 7 we have that C' cannot be blocked on either $y, \overline{y}, z$, or $\overline{z}$. Then, let us see that $y \in C'$. Indeed, let ℓ be any literal such that C' is blocked on ℓ w.r.t. F'. Given that by the previous argument $\ell \notin \{y, \overline{y}, z, \overline{z}\}$, we have that $C' \otimes_\ell (\overline{\ell} \vee \overline{y} \vee \overline{z})$ must be a tautology. This implies in turn that either $y \in C'$ or $z \in C'$. But by repeating the same argument with respect to $C' \otimes_\ell (\overline{\ell} \vee \overline{y} \vee z)$ we obtain that either $y \in C'$ or $\overline{z} \in C'$, from where it follows that $y \in C'$. We thus have that C' is of the form $C'' \vee y$, with $\ell \in C''$.

Using Fact 2, we have that $C'{\downarrow_F}$ is blocked on ℓ w.r.t. F, and as $C'{\downarrow_F} \subseteq C''$, we have that C'' is blocked on ℓ w.r.t. F.

Let us now see that neither z nor $\overline{z}$ can belong to C''. Indeed, assume expecting a contradiction that $z \in C''$ (the analysis for $\overline{z}$ is analogous). Now, let $C'' := C^\dagger \vee z$, and as $\ell \neq z$, we have $\ell \in C^\dagger$. Because $C' = C^\dagger \vee z \vee y$ is blocked on ℓ w.r.t. F, we have by Fact 2 that $C'{\downarrow_F}$ is blocked w.r.t. F, and as $C'{\downarrow_F} \subseteq C^\dagger{\downarrow_F}$, we deduce that $C^\dagger{\downarrow_F}$ is blocked on ℓ w.r.t. F. Note that $|C^\dagger| = k - 1$, and thus $\left|C^\dagger{\downarrow_F}\right| \leq k - 1$. Then, adding literals of F to $C^\dagger{\downarrow_F}$ until we obtain a clause $C^\star \in \mathsf{BC}_k(F)$ of size k produces a contradiction, since $C^\star$ will be blocked on ℓ w.r.t. F (Fact 1), while $C^\dagger{\downarrow_F}$ is a strict subset of $C^\star$ that is also blocked on ℓ w.r.t F, thus contradicting the minimality of $C^\star$. We thus deduce that neither z nor $\overline{z}$ belong to C''.

We now verify that $C' = (C'' \vee y)$ is minimally blocked w.r.t. F'. Indeed, assume expecting a contradiction that $C^\dagger$ is a proper subset of C' that is blocked w.r.t F'. As any clause in $\mathsf{BC}(F')$ must contain y by the argument above, we have $C^\dagger = (C^\triangle \vee y)$ for some $C^\triangle \subsetneq C''$. Using Fact 1 and Fact 2 we obtain that $C^\triangle$ is blocked w.r.t. F, which contradicts the minimality of $C'' \in \mathsf{BC}_k(F)$.

Similarly, if C' were not singly blocked on $\ell(C'')$, given that C' cannot be blocked on y, we would have that C'' is not singly blocked w.r.t. F, contradicting the hypothesis of $C'' \in \mathsf{BC}_k(F)$ being singly blocked. □

We now claim the desired graph isomorphism.

Claim 3. We have that $C_1 \rightsquigarrow_p C_2$, with $C_1, C_2 \in \mathsf{BC}_k(F)$, if and only if $(C_1 \vee y) \rightsquigarrow_p (C_2 \vee y)$ in F'.

Proof of Claim 3. We will use Remark 1 for both directions. For the forward direction, assume $C_1 \rightsquigarrow_p C_2$. Then, given that C_2 is blocked on $\ell(C_2)$ w.r.t. F, we have that $\overline{\ell(C_2)} \in C_1$ and that no literal $l \in C_2$ with $l \neq \ell(C_2)$ has $\overline{l} \in C_1$. This implies that $(C_1 \vee y) \rightsquigarrow_p (C_2 \vee y)$, as $\overline{\ell(C_2)} \in (C_1 \vee y)$. The backward direction follows from the fact obtained in Claim 2 stating that $(C_2 \vee y)$ is blocked w.r.t. F' on $\ell(C_2)$, the only literal on which C_2 is blocked w.r.t. F. Indeed, if $(C_1 \vee y) \rightsquigarrow_p (C_2 \vee y)$, we have $\overline{\ell(C_2)} \in C_1$, and no literal $l \in C_2$ with $l \neq \ell(C_2)$ has $\overline{l} \in C_1$, from where Remark 1 directly implies $C_1 \rightsquigarrow_p C_2$. □

Combining Claim 1, Claim 2, and Claim 3, we get the desired isomorphism between $G_{\mathsf{BC}_k(F)}$ and $G_{\mathsf{BC}_{k+1}(F')}$, where the conditions of single-blockedness and minimal-blockedness are verified in Claim 2. Given the construction of F' from F takes polynomial time, we conclude the proof. □

6 Hardness Results

In this section we prove that k-BCA is NP-hard for every $k \geq 2$. As a first step, we will show that the FeedbackVertexSet problem is no less hard for 2-subdivision graphs, which arise thanks to Lemma 7.

Proposition 9. FeedbackVertexSet *is* NP-*hard when restricted to* 2-*subdivision graphs.*

Proof. The reduction is directly from FeedbackVertexSet over general directed graphs. Let $G = (V, E)$ be an arbitrary directed graph, and $t \geq 1$ and integer. Then, let G_2 be the 2-subdivision of G. We have that $V(G_2) = V(G) \cup \{v_e \mid e \in E\}$, and

$$E(G_2) = \{(v_e, u), (u, v_e) \mid e = (u, v) \in E\}.$$

Note that G_2 is bipartite under $L := V(G), R := \{v_e \mid e = (u, v) \in E\}$. Clearly, if C is a directed cycle in G, then its 2-subdivision C' is a directed cycle in G_2. Conversely, we claim that if C is any directed cycle in G_2, then C is the 2-subdivision of a directed cycle C' in G, which we denote by $f^{-1}(C)$. Indeed, let $C = (v_1, v_2, \dots, v_n)$ be a directed cycle in G_2. Let $v_i \in C$ be such that $v_i \in L$. Then, let $J = \{i, i+2, i+4, \dots, i+2n\}$, and as

$$(v_j, v_{(j+2)\%2}) \in E(G), \quad \forall j \in J,$$

we have that $C' = f^{-1}(C) = \{v_{j\%n} \mid j \in J\} \subseteq C$ is a directed cycle in G. Now, if G has a feedback vertex set S of size t, then S is also a feedback vertex set of G_2, as otherwise if we assume expecting a contradiction that there is a directed cycle C such that $C \cap S = \varnothing$, then $f^{-1}(C) \cap S = \varnothing$ as $f^{-1}(C) \subseteq C$, thus implying that C is present in $G \setminus S$, which contradicts S being a feedback vertex set. On the other hand, let S be a feedback vertex set of size t in G_2. We claim that from S we can easily construct a feedback vertex set S' for G_2, but such that $S' \subseteq V(G)$. For this, it is enough to replace every vertex v_e in S corresponding to an edge $e = (u, v)$ by u. Indeed, if C is a directed cycle in G_2, we must have that $S \cap C \neq \varnothing$. If $u \in S \cap C$ for some vertex u, then $u \in S'$ as well, and thus $G_2 \setminus S'$ also avoids C. Otherwise, $v_e \in S \cap C$ for some edge $e = (u, v)$, and thus $u \in S'$, which also implies that $G_2 \setminus S'$ avoids C. As this is true for every directed cycle C in G_2, we concldue that S' is a feedback vertex set of G_2. Moreover, we clearly have $|S'| = |S| = t$. By the argument above, we have that S' is also a feedback vertex set for G, and thus we conclude the proof. ☐

We now have all the ingredients to prove our main result, stating that k-BCAis NP-hard.

Proof of Theorem 2. The reduction is from FeedbackVertexSet over 2-subdivision graphs, proved NP-hard in Proposition 9. Let $G = (V, E)$ be a 2-subdivision graph, and let (G, t) be an instance of FeedbackVertexSet. By Lemma 7, we can construct in polynomial time a formula F such that $G_{\mathsf{BC}_2(F)} = G$. Next, using Lemma 8 exactly $(k-2)$ times, we obtain a formula F' such that $G_{\mathsf{BC}_k(F')} = G$. Given k is fixed, this takes polynomial time. Then, using Lemma 6, (G, t) is a Yes-instance of FeedbackVertexSet if and only if $(F', |V(G)| - t)$ is a Yes-instance of k-BCA. As the reduction has been carried out in polynomial time, this concludes the proof. ☐

We finish the section by addressing a conjecture of Heule and Biere [12]. It is well-known that if $\Gamma = (\Gamma_1, \dots, \Gamma_n)$ is a valid sequence for BCA on a formula F, then $F_\Gamma := \Gamma_1 \wedge \dots \wedge \Gamma_n$ is always satisfiable [12], but can we count the number of satisfying assignments of F_Γ in polynomial time? Heule and Biere conjectured a positive answer, and this conjecture was mentioned again by Balyo et al. [2]. Unfortunately, it turns out that this is not the case.

Proposition 10. *Counting the number of satisfying assignments of a formula F_Γ is #P-hard. Equivalently, counting the number of satisfying assignments of a* blocked set *(see [12]) Γ is #P-hard.*

Proof. It is well-known that counting the number of vertex covers of a graph G is #P-hard [20]. We reduce from this problem by defining F as an empty formula, and letting

$$F_\Gamma = \bigwedge_{\{u,v\} \in E(G)} (x_u \vee x_v).$$

Note that Γ is indeed a valid sequence for BCA on F, as all clauses in F_Γ are trivially blocked w.r.t F, and there are no preventions between clauses of F_Γ since all literals are positive.

Clearly each distinct satisfying assignment of F_Γ corresponds to a distinct vertex cover of G, which implies that the number of satisfying assignments of F_Γ is equal to the number of vertex covers of G, and thus the result follows. □

7 Approximation Algorithms

In this section we prove that, at least for $k = 2$, the BCA problem is inside the APX complexity class. This is in sharp contrast with FeedbackVertexSet, which does not admit constant factor approximation algorithms unless P = NP [8,11]. To be precise, given an input formula F, we denote by $\mathsf{OPT}(F)$ the maximum number of binary blocked clauses that can be added to F. Then, for an algorithm ALG computing valid BCA-sequences, we denote by $\mathsf{ALG}(F)$ the length of the valid BCA-sequence output by ALG on input F. We say that ALG is a ρ-approximation algorithm for the 2-BCA problem if, for every input formula F, we have $\mathsf{ALG}(F) \geq \rho \cdot \mathsf{OPT}(F)$. Before we jump into the proof, note that as opposed to the hardness reesults presented in Section 6, we cannot focus now only on singly-blocked clauses, but rather a positive result must hold for any formula F.

Theorem 11. *The* 2-BCA *problem admits a* 1/6*-approximation algorithm that runs in polynomial time.*

Proof. Let F be an input formula with variables $x_1, \dots, x_n$. We will split $\mathsf{BC}_2(F)$ into 6 sets:

$$S_1 := \{C \text{ is blocked on } x_i \text{ w.r.t. } F \mid C = (x_i \lor x_j)\}, \tag{1}$$
$$S_2 := \{C \text{ is blocked on } \overline{x_i} \text{ w.r.t. } F \mid C = (\overline{x_i} \lor \overline{x_j})\}, \tag{2}$$
$$S_3 := \{C \text{ is blocked on } x_i \text{ w.r.t. } F \mid C = (x_i \lor \overline{x_j}) \text{ with } i < j\}, \tag{3}$$
$$S_4 := \{C \text{ is blocked on } x_i \text{ w.r.t. } F \mid C = (x_i \lor \overline{x_j}) \text{ with } i > j\}, \tag{4}$$
$$S_5 := \{C \text{ is blocked on } \overline{x_i} \text{ w.r.t. } F \mid C = (\overline{x_i} \lor x_j) \text{ with } i < j\}, \tag{5}$$
$$S_6 := \{C \text{ is blocked on } \overline{x_i} \text{ w.r.t. } F \mid C = (\overline{x_i} \lor x_j) \text{ with } i > j\}. \tag{6}$$

Note that this sets can clearly be computed in polynomial time. We now claim that it suffices to output the largest of these sets. Because $\sum_{i=1}^{6} |S_i| = |\mathsf{BC}_2(F)|$ we have $\max_{i=1}^{6} |S_i| \geq \frac{1}{6}|\mathsf{BC}_2(F)|$. Let us now show that each of the graphs $G_{\mathsf{BC}_2(F)}[S_i]$ is acyclic. First, it is easy to see that $G_{\mathsf{BC}_2(F)}[S_1]$ and $G_{\mathsf{BC}_2(F)}[S_2]$ are acyclic, as they in fact contain no edges (recall Remark 1). Let us now show that $G_{\mathsf{BC}_2(F)}[S_3]$ is acyclic, as the remaining cases are analogous. By Remark 1, any edge $C_1 \to C_2$ in $G_{\mathsf{BC}_2(F)}[S_3]$ must correspond to

$$C_1 := (x_i \lor \overline{x_j}) \rightsquigarrow_p (x_j \lor \overline{x_k}) =: C_2,$$

where by definition $i < j$ and $j < k$. Therefore, any cycle $C_1 \to C_2 \to \dots \to C_m$ in $G_{\mathsf{BC}_2(F)}[S_3]$ must correspond to a sequence

$$(x_i \lor \overline{x_j}) \rightsquigarrow_p (x_j \lor \overline{x_k}) \rightsquigarrow_p (x_k \lor \overline{x_l}) \rightsquigarrow_p \cdots \rightsquigarrow_p (x_{m-1} \lor \overline{x_m}) \rightsquigarrow_p (x_m \lor \overline{x_i}),$$

from where we would deduce $i < j < k < l < \cdots < m < i$, a contradiction. Therefore, $G_{\mathsf{BC}_2(F)}[S_3]$ is acyclic, and the same argument applies to $G_{\mathsf{BC}_2(F)}[S_4]$, $G_{\mathsf{BC}_2(F)}[S_5]$, and $G_{\mathsf{BC}_2(F)}[S_6]$. □

Note that it is not straightforward to extend this algorithm for $k \geq 3$, as the sets S_m in which one could hope to partition $\mathsf{BC}_k(F)$ become much more complicated. For example, for $k = 3$, consider the set S_m containing blocked clauses of the form $(x_i \lor \overline{x_j} \lor x_k)$, blocked on x_i, with $i > j$ but $i < k$. How could we argue that $G_{\mathsf{BC}_3(F)}[S_m]$ is acyclic, or at least that we can easily obtain a large fraction of it? This is left as an open problem.

8 Further Characterization of BC Graphs

Even though Lemma 7 and Lemma 8 provide a partial characterization of BC graphs that was enough to derive a hardness result, we now provide a tighter partial characterization for $k \geq 4$. Let us first introduce two standard graph theory definitions. We will say a directed graph G is C_2-free if it does not contain edges (u, v) and (v, u) for any pair of vertices u, v, and we will say G is k-out-regular if the out-degree of every vertex is exactly k.

Lemma 12. *Given any C_2-free k-out-regular directed graph G, one can compute in polynomial time a formula F such that $G_{\mathsf{BC}_{(k+1)}(F)} = G$.*

Proof. In order to simplify the presentation of the proof, let us present it only for $k = 3$, as its generalization to $k = 3$ does not change the proof but makes the notation heavier. Let $G = (V, E)$ be a C_2-free 3-out-regular graph. We start by creating a variable x_u for every vertex $u \in V$. Then, for every vertex u, create a clause C_u defined as

$$C_u = \left(x_u \vee \left(\bigvee_{(u,v) \in E} \overline{x_v} \right) \right) = \left(x_u \vee \overline{x_{v_1}} \vee \overline{x_{v_2}} \vee \overline{x_{v_3}} \right),$$

where we used x_{v_i} to denote the variable x_v for the i-th out-neighbor v of u in G. We will now construct F to ensure that $\mathsf{BC}_4(F) = \{C_u \mid u \in V\}$. We create two new variables y and z, and then add the following sets of clauses to the formula F:

1. (xy-clauses) For every vertex u, add to F the clauses $(x_u \vee y), (x_u \vee \overline{y})$.

2. (yz-clauses) Add to F the clauses $(y \vee z), (y \vee \overline{z}), (\overline{y} \vee z), (\overline{y} \vee \overline{z})$.

3. (E-clauses) For every edge $(u, v) \in E$, add to F the clause $(\overline{x_u} \vee x_v)$.

First, observe that every clause C_u indeed belongs to $\mathsf{BC}_4(F)$ as C_u is blocked on x_u, which can be seen as the only clauses in F where $\overline{x_u}$ appears are the E-clauses, and their resolvent is tautological by construction. We now claim that there are no other clauses in $\mathsf{BC}_4(F)$. Indeed, let $C \in \mathsf{BC}_4(F)$ be arbitrary, and then see how the following steps prove that $C = C_u$ for some vertex u.

1. The clause C cannot be blocked on y, as otherwise $C \otimes_y (\overline{y} \vee z)$ being tautological would imply $\overline{z} \in C$, but then $C \otimes_y (\overline{y} \vee \overline{z})$ being tautological would imply $z \in C$, which is a contradiction. Similarly, C cannot be blocked on $\overline{y}$, $\overline{z}$, or z.

2. The clause C cannot be blocked on $\overline{x_u}$ for any $u \in V$, as otherwise $C \otimes_{\overline{x_u}} (x_u \vee y)$ being tautological would imply $\overline{y} \in C$, but then $C \otimes_{\overline{x_u}} (x_u \vee \overline{y})$ being tautological would imply $y \in C$, which is a contradiction.

3. From 1. and 2., we deduce that C must be blocked on a literal $x_{u^\star}$ for some $u^\star \in V$.

4. Let v_1, v_2, v_3 be the out-neighbors of $u^\star$ in G. Then, as $C \otimes_{x_{u^\star}} (\overline{x_{u^\star}} \vee x_{v_1})$ is tautological, we must have $\overline{x_{v_1}} \in C$. Similarly, we deduce $\overline{x_{v_2}} \in C$ and $\overline{x_{v_3}} \in C$.

5. From the previous steps, and recalling that $|C| = 4$, we conclude that $C = C_{u^\star}$.

Having proved that $\mathsf{BC}_4(F) = \{C_u \mid u \in V\}$, we now claim that $C_u \rightsquigarrow_p C_v$ if and only if $(u, v) \in E$. To see this, first note that if $(u, v) \in E$, and we denote by w_1, w_2, w_3 the out-neighbors of v, and by v_1, v_2, the out-neighbors of u that are not v, then

$$C_v \otimes_{x_v} C_u = (x_u \lor \overline{x_{v_1}} \lor \overline{x_{v_2}} \lor \overline{x_{w_1}} \lor \overline{x_{w_2}} \lor \overline{x_{w_2}}),$$

which is tautological if and only if $u \in \{w_1, w_2, w_3\}$, which would imply $(v, u) \in E$ and thus contradict G being C_2-free. For the other direction, if $(u, v) \notin E$, then $\overline{x_v} \notin C_u$, and thus C_u cannot prevent C_v which is blocked on x_v. This concludes the proof. ☐

But do C_2-free k-out-regular graphs exist? The previous result will be vacuously true otherwise! We now show that not only do they exist but also that is easy to construct them.

Lemma 13. *For every $k \geq 3$, there exists a C_2-free k-out-regular graph G_k, and moreover, such a graph can be computed in time $O_k(1)$.*

Proof. The idea of this lemma is fairly standard (cf. [5, 21]). Let $n = 2k + 1$, and let K_n be the complete graph on n vertices, which is clearly $2k$-regular. As every vertex in K_n has even degree, we have that K_n admits an Eulerian cycle $C = e_1, \dots, e_m$, where $m = \binom{n}{2}$, which can be computed in time $O(n^2)$. We construct G as the orientation of K_n according to C, which we describe formally next. For every $i \in \{1, \dots, m\}$, let $e_i = \{u_i, v_i\}$, and let π_i be the single vertex in $e_i \setminus \left(e_i \cap e_{(i+1)\%m}\right)$. This way, we identify C with the sequence $\pi_1, \pi_2, \dots, \pi_m$. Then, to an empty graph on n vertices, add directed edges $(\pi_i, \pi_{(i+1)\%m})$ for every $i \in \{1, \dots, m\}$. Note that the resulting graph G is an orientation of K_n, and thus it is C_2 free. As G is also a directed Eulerian graph, the in-degree of each vertex equals its out-degree, and as those two amounts add up to $2k$, it follows that G is k-out-regular. As the construction of G takes $O(n^2)$ time, and $n = O_k(1)$, we conclude the proof. ☐

9 Conclusion

We started by showing how the role of blocked clauses in SAT-solving is nuanced, and further research will be required to have a better understanding of the impact of adding blocked clauses as a preprocessing step. In particular, a promising direction of future research is to identify subclasses of blocked clauses that are likely to be helpful during solving time, and thus restrict BCA to those clauses. Similarly, identifying subclasses of blocked clauses that are not helpful during solving might result in better BCE. For a concrete example, in the Pythagorean Triple problem [14], BCE is able to eliminate variables, which has a significant impact on solving time. This suggests that one could perform BCE with the hope of eliminating variables, but re-introduce some of the eliminated blocked clauses afterwards.

In terms of complexity, we have shown a fundamental asymmetry between BCE and BCA, proving that BCA is NP-hard for every arity $k \geq 2$. In doing so, we have related BCA to different problems in graph theory, which we hope can result in identifying tractable cases for BCA. Moreover, we have shown a very simple approximation algorithm obtaining a constant factor approximation for 2-BCA. It might be possible that a more clever algorithm can obtain an even better approximation factor. We have shown as well that model counting over blocked sets is computationally hard, answering a question of Heule and Biere [12]. This arguably represents a drawback of BCA, as it can alter the number of models of the original formula in a way that is hard to track.

In terms of open problems, while Sections 5 and 8 provide a decent characterization of BC graphs that is sufficient to derive hardness results, it would be interesting to fully characterize them. In other words, what graphs can arise as BC graphs?

Another interesting open problem is to show that computing a maximum length sequence for BCA, without any restriction on the arity of the blocked clauses, is NP-hard. Finally, a last open question is whether k-BCA admits a constant factor approximation algorithm for $k \geq 3$.

Acknowledgments. I would like to thank Marijn J.H. Heule for introducing me to *Blocked Clause Addition*. I would also like to thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Science Foundation (NSF) under grants CCF-2108521 and CCF-2229099.

References

[1] R. Balasubramanian, Michael R. Fellows, and Venkatesh Raman. An improved fixed-parameter algorithm for vertex cover. *Information Processing Letters*, 65(3):163–168, February 1998.

[2] Tomáš Balyo, Andreas Fröhlich, Marijn J. H. Heule, and Armin Biere. Everything You Always Wanted to Know about Blocked Sets (But Were Afraid to Ask). In Carsten Sinz and Uwe Egly, editors, *Theory and Applications of Satisfiability Testing – SAT 2014*, pages 317–332, Cham, 2014. Springer International Publishing.

[3] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling Entering the SAT Competition 2020. In Tomas Balyo, Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.

[4] Armin Biere and Mathias Fleury. Gimsatul, IsaSAT and Kissat entering the SAT Competition 2022. In Tomas Balyo, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2022 – Solver and Benchmark Descriptions*, volume B-2022-1 of *Department of Computer Science Series of Publications B*, pages 10–11. University of Helsinki, 2022.

[5] Gunnar Brinkmann. Generating regular directed graphs. *Discrete Mathematics*, 313(1):1–7, 2013.

[6] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, and Saket Saurabh. *Parameterized algorithms.* Springer International Publishing, Basel, Switzerland, 1 edition, August 2015.

[7] H. N. de Ridder et al. Information System on Graph Classes and their Inclusions (ISGCI). `https://graphclasses.org/classes/problem_Feedback_vertex_set.html`.

[8] G. Even, J. (Seffi) Naor, B. Schieber, and M. Sudan. Approximating Minimum Feedback Sets and Multicuts in Directed Graphs. *Algorithmica*, 20(2):151–174, February 1998.

[9] Aleksander Figiel, Vincent Froese, André Nichterlein, and Rolf Niedermeier. There and Back Again: On Applying Data Reduction Rules by Undoing Others. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms (ESA 2022)*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 53:1–53:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[10] Mathias Fleury and Daniela Kaufmann. Life span of SAT techniques, 2024.

[11] Venkatesan Guruswami, Rajsekar Manokaran, and Prasad Raghavendra. Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 573–582, 2008.

[12] Marijn J. H. Heule and Armin Biere. Blocked Clause Decomposition. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, Lecture Notes in Computer Science, pages 423–438, Berlin, Heidelberg, 2013. Springer.

[13] Marijn J. H. Heule, Matti Järvisalo, and Armin Biere. Efficient CNF Simplification Based on Binary Implication Graphs. In Karem A. Sakallah and Laurent Simon, editors, *Theory and Applications of Satisfiability Testing - SAT 2011*, pages 201–215, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[14] Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek. Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer. In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability Testing – SAT 2016*, pages 228–245, Cham, 2016. Springer International Publishing.

[15] Matti Järvisalo, Armin Biere, and Marijn Heule. Blocked Clause Elimination. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 129–144, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[16] Richard M. Karp. *Reducibility among Combinatorial Problems*, page 85–103. The IBM Research Symposia Series. Springer US, Boston, MA, 1972.

[17] Benjamin Kiesl, Adrián Rebola-Pardo, and Marijn J. H. Heule. Extended resolution simulates DRAT. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning*, Lecture Notes in Computer Science, page 516–531, Cham, 2018. Springer International Publishing.

[18] Benjamin Kiesl-Reiter and Mike Whalen. Proofs for incremental SAT with inprocessing. In *FMCAD 2023*, 2023.

[19] O. Kullmann. On a generalization of extended resolution. *Discrete Applied Mathematics*, 96–97(1):149–176, October 1999.

[20] J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.

[21] Lex Schrijver. Bounds on the number of Eulerian orientations. *Combinatorica*, 3:375–380, January 1983.

[22] Bernardo Subercaseaux and Marijn J. H. Heule. The packing chromatic number of the infinite square grid is 15. In Sriram Sankaranarayanan and Natasha Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part I*, volume 13993 of *Lecture Notes in Computer Science*, page 389–406. Springer, 2023.

[23] Emre Yolcu. Lower bounds for set-blocked clauses proofs, 2024.