MASTER

Sparsification upper and lower bounds for graph problems and not-all-equal SAT

Pieterse, A.

Award date:
2015

Link to publication
Sparsification Upper and Lower Bounds for Graph Problems and Not-All-Equal SAT

Astrid Pieterse (0743073)
Supervisor: dr. Bart. M. P. Jansen

Master Thesis
Committee: Bart. M. P. Jansen, Jesper Nederlof, and Michel. A. Westenberg

EINDHOVEN UNIVERSITY OF TECHNOLOGY
August 31, 2015
Abstract

Many problems we care for are known to be NP-complete. Therefore, we do not think they can be solved in polynomial time, so only an exponential time algorithm is known. To speed up the search for a solution, we want to start by preprocessing an input instance. You can try to make the input instance smaller in polynomial time, without changing the answer. After doing so, we can run the slow algorithm on a smaller input. This thesis investigated whether there exist such algorithms that reduce the number of edges in graph problems, and the number of clauses for logical formulas. Such an algorithm is called a sparsification.

We will show that for a number of decision problems on graphs, polynomial-time algorithms cannot compress instances of such problems to equivalent instances, of a possibly different problem, whose encoding size is sub-quadratic in the number of vertices. Here we only want to maintain the solution (YES/NO), therefore P = NP would imply that any NP-complete problem can be compressed in polynomial time to a single bit, indicating whether it was a YES- or a NO-instance. For example look at the 4-colorability problem, which asks if we can color all vertices in a graph in such a way that the endpoints of any edge in the graph have distinct colors. Assuming that NP is not contained in coNP/poly, we show that instances of 4-COLORING cannot be compressed to a sub-quadratic encoding in polynomial time. We obtain similar results for a number of other graph problems.

Finally we consider NOT-ALL-EQUAL SAT (NAE-SAT). This is a variant of the well-known satisfiability problem, that plays a central role in the theory of NP-completeness. We show that an instance of NAE-SAT with n variables and d literals per clause can not be compressed to an equivalent instance of size $O(n^{d-1-\varepsilon})$ for any $\varepsilon > 0$, unless NP $\subseteq$ coNP/poly. Furthermore we present a generalized kernel that matches this lower bound.
Contents

1 Introduction 4
2 Preliminaries 9
3 Feedback Arc Set 14
4 4-Coloring 16
5 Planar List Coloring 24
6 Hamiltonian Cycle 26
7 Dominating Set 33
8 d-Hypergraph 2-Colorability and d-NAE-Sat 40
   8.1 Lower bound 40
   8.2 Kernel 41
9 Conclusion 45

List of Figures

1 General idea for a degree-2 cross-composition 12
2 Reduction from VERTEX COVER to FEEDBACK ARC SET. 14
3 Tree gadgets with example colorings for Lemmas 4.2 and 4.3 16
4 Triangular gadget 17
5 Gadgets constructed to prove Lemma 4.6 19
6 Colorings of the gadgets constructed to prove Lemma 4.6 20
7 Graph G′ constructed in Theorem 4.7 to prove a sparsification lower bound for 4-COLORING. 21
8 Path gadget 26
9 Instance G′ for DIRECTED HAMILTONIAN CYCLE, constructed to prove Theorem 6.2 28
10 Reduction from DIRECTED HAMILTONIAN CYCLE to HAMILTONIAN CYCLE. 32
11 Graph G constructed in the proof of Theorem 7.2 for (CONNECTED) DOMINATING SET. 34
1 Introduction

Background Preprocessing has proven to be useful in many applications, such as when dealing with large scheduling problems with constraints. In this case we can seek to reduce the number of constraints, by removing redundant ones. Furthermore we can sharpen existing constraints such that we do not look into partial schedules that can never lead to a good final scheduling.

In practice, algorithms that seem unfeasibly slow in theory can work surprisingly well on large inputs, but this behavior can usually not be explained by the normal worst-case running time analysis. In this type of analysis, we only consider the running time depending on the total size of the input. There may however exist many large inputs for which even an NP-hard problem can be solved much quicker than expected. To analyze this behavior, the notion of a parameterized problem was introduced, where apart from the total input size there is a parameter \( k \in \mathbb{N} \) that measures the complexity of a given input instance ([13], [15]). We hope that if the parameter is small, the instance should now be relatively easy to solve. We say that a parameterized decision problem is fixed-parameter tractable if there is an algorithm that, given any input \( x \) of length \(|x|\) with parameter value \( k \), determines the \textsc{Yes}/\textsc{No} answer to the instance and whose running time is bounded by a polynomial in \(|x|\), multiplied by an arbitrary (usually exponential) function in the parameter \( k \). Therefore, we can make a distinction between parameterized problems that are FPT, and parameterized problems that are not believed to be FPT.

To classify problems, methods were designed to find FPT algorithms or prove a parameterized problems is likely to not be FPT. For example, the method of iterative compression (for example used in [22]) can be used to find FPT algorithms for minimization problems. The main idea is to use a compression routine, that given the input instance and a (too large) solution tries to find a smaller solution or output that this is not possible. To make a distinction between FPT problems and problems that are not believed to be FPT, a hierarchy of complexity classes was introduced in [11]. The main distinction is between class FPT, containing problems that can be solved efficiently when the parameter is small, and problems that are hard for \( W[1] \). Classes FPT and \( W[1] \) are the parameterized analogues of \( P \) and \( NP \), respectively. By their definition, we know \( \text{FPT} \subseteq \text{W}[1] \). It is widely believed that \( \text{FPT} \neq \text{W}[1] \), similar to the statement that \( P \neq NP \). Under this assumption, problems that are hard for \( W[1] \) are not fixed parameter tractable.

It was proven that \textsc{Independent Set} parameterized by the solution size is \( W[1] \)-hard, while \textsc{Vertex Cover} with the same parameter is fixed parameter tractable. Note that an independent set in a graph is a subset of the vertices, such that none are connected by an edge. Vertex cover on the other hand asks for a subset of the vertices such that for every edge at least one of its endpoints is in the set. It is thereby not hard to see that a graph has a vertex cover of size \( k \), if and only if it has an independent set of size \( n - k \).
It turns out that the framework of parameterized complexity analysis makes it possible to rigorously analyze the power of polynomial-time preprocessing algorithms, using the concept of kernelization. A kernelization is a polynomial-time algorithm that transforms a parameterized input $x$ with parameter $k$ for a decision problem, into a new input $x'$ with parameter $k'$, such that the following two conditions hold: (1) the answer to $(x, k)$ is YES if and only if the answer to $(x', k')$ is YES, and (2) there is a function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that the size of $x'$ and the new parameter value $k'$ are both bounded by $f(k)$. The function $f$ is called the size of the kernelization: it gives a guarantee on the amount of data reduction that is achieved in terms of the parameter $k$. A kernel whose size $f$ is bounded by a polynomial, is called a polynomial kernel. It is known that every problem in FPT has a (not necessarily polynomial) kernel [2]. To apply these kernels in practice, it is desirable to find kernels that are as small as possible.

Methods were introduced to be able to find kernels, or prove that a problem does not have a kernel of a certain size. First, the focus was on deciding whether certain problems had polynomial kernels [3]. Under certain assumptions we can prove a kernel of polynomial size does not exist by giving a cross-composition. This is a polynomial time algorithm that transforms a large number of inputs into a single output of the goal problem that acts as a logical OR of the given inputs. With the right bounds on the parameter value of the output instance, kernel lower bounds can be proven.

It is easy to see some assumptions are needed in order to prove that a problem does not have a kernel of small size. Since many of the problems that we consider are NP-complete, $P = NP$ would imply every problem in NP has a trivial polynomial kernel. The preprocessing procedure can in this case decide whether it is given a YES- or a NO-instance. Depending on the result, it can then output a trivial, constant size, YES/NO-instance of the same problem. To actually achieve kernelization lower bounds we currently need an even stronger assumption than $P \neq NP$, which is $NP \not\subseteq coNP/poly$. As the reader may know, a problem is in coNP if it can be solved by a co-nondeterministic Turing machine. coNP/poly is a complexity class containing some more problems, namely the ones that can be solved by a co-nondeterministic Turing machine that is allowed to use a polynomial size advice. This polynomial advice is an additional input string, given on an extra tape, that may only depend on the input length $n$. Its length should be polynomial in $n$. There are multiple reasons why we believe $NP \not\subseteq coNP/poly$. Intuitively, coNP and NP are somehow incomparable. For problems in NP we can give a certificate to validate a YES-instance, such as a satisfying truth assignment for a Boolean formula. For problems in coNP on the other hand, validating NO-instances is easy with the right certificate. For example, if we ask if a certain Boolean formula is a tautology, a truth assignment for which the formula is false would be a certificate for a NO-instance. It seems like allowing a polynomial amount of advice is not enough to change the situation.

\[1\] As is customary in the literature, we will sometimes abbreviate kernelization by kernel.
The framework for proving lower bounds can be further refined ([17], [5]), to be able to also prove bounds on the size of polynomial kernels, such as proving a kernel of bitsize $O(n^{d-\varepsilon})$ is unlikely to exist for $\varepsilon > 0$. For example, it was proven that VERTEX COVER, parameterized by the solution size $k$, does not have a kernel of size $O(k^{2-\varepsilon})$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$ [8].

Most instances for classical graph problems can trivially be stored in $O(n^2)$ bits by using an adjacency matrix, when the input is a graph on $n$ vertices. One may wonder if it is possible to reduce the number of edges in the graph in polynomial time, without changing the answer, such that it can be stored in a sub-quadratic number of bits. This process is often referred to as the sparsification of a graph. Let us for example consider the vertex cover problem again. Suppose we want to find a vertex cover of size at most $k$, and the graph contains a vertex of degree strictly larger than $k$. If we do not choose this vertex, we would be forced to pick all its neighbors. However, that would exceed the size limit of our vertex cover. As such, we could pick the vertex in the cover, delete the vertex from the graph and decrease $k$ by one. The original graph has a vertex cover of size $k$ if and only if the smaller graph has a vertex cover of size $k - 1$. In this way, we have removed at least $k + 1$ edges and one vertex. We would like to know whether a combination of such clever reductions can lead to a significant decrease of the size of our instance.

The goal of this thesis is to analyze whether sparsification is possible for a number of classic NP-complete problems in graph theory and logic. We can use the concept of kernelization to answer this question, by choosing the number of vertices to be the parameter. The question whether an instance can be compressed to an equivalent one of sub-quadratic size, therefore turns into the question of whether the parameterization by the number of vertices $n$ in the graph has a kernel with bitsize $O(n^{2-\varepsilon})$ for some $\varepsilon > 0$. In this way, we obtain lower bounds stating that no kernel of size at most $O(n^{2-\varepsilon})$ exists, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$, where $n$ is the number of vertices in the graph. For logic problems we will use the number of variables as the parameter to obtain sparsification lower bounds.

**Results** We prove that for several graph-theoretical problems, there is no preprocessing algorithm that transforms an $n$-vertex instance into an equivalent sparse instance, i.e., one with $O(n^{2-\varepsilon})$ edges. This is established by considering a more general type of preprocessing algorithm called a generalized kernel (see Section 2, Definition 2.2), and proving that even this more general type of reduction algorithm cannot exist unless $\text{NP} \subseteq \text{coNP}/\text{poly}$. Furthermore we study a variant of satisfiability which is $\text{NOT-ALL-EQUAL SAT}$ and give a sparsification lower bound and a generalized kernel that matches the lower bound.

First of all we consider FEEDBACK ARC SET. Given a directed graph, this problem asks if it is possible to find $k$ arcs, whose removal results in an acyclic graph. In the next section we use a linear parameter transformation from VERTEX COVER to obtain a
sparsification lower bound for the problem, which shows that there is no polynomial
time algorithm that reduces an instance with \( n \) vertices to an equivalent one of bitsize
\( \mathcal{O}(n^{2-\varepsilon}) \) for any \( \varepsilon > 0 \), unless \( \text{NP} \subseteq \text{coNP}/\text{poly} \).

In Section 4 we consider the \( 4\text{-COLORING} \) problem. The problem asks if it is possible
to color the vertices of a graph with 4 colors, such that the endpoints of any edge receive
distinct colors. We prove that, under the same complexity assumption, no polynomial-
time reduction to an equivalent instance of sub-quadratic size is possible. The proof
is related to a construction by Jansen and Kratsch, who proved that an instance of the
related Chromatic Number problem (can the given graph be properly colored with \( q \)
 colors?) that is large, but whose graph is simple because it has a small vertex cover,
cannot be reduced to an equivalent instance whose size is polynomial in that of the
vertex cover [18]. For our sparsification lower bound, we use some of the ideas of Jansen
and Kratsch, but we introduce several new gadgets.

We then consider \( \text{LIST COLORING} \), in list coloring every vertex has a list of allowed
colors and we want to find a proper coloring such that each vertex gets a color from its
list. In general, \( \text{LIST COLORING} \) is a more general problem than \( 4\text{-COLORING} \), which
is just a special case where each list contains colors \{1,2,3,4\}. By the previous result,
a sub-quadratic sparsification for \( \text{LIST COLORING} \) is unlikely to exist. Therefore, we
consider what happens when restricting the input graphs to be planar. Since the number
of edges in a planar graph is always linear in the number of vertices [9], it may not seem
surprising that we obtain a kernel of sub-quadratic size. However, since every vertex
contains a (arbitrarily long) list of possible colors, we need to bound the size of these
lists in our kernel.

To obtain a sparsification lower bound for \( \text{HAMILTONIAN CYCLE} \), which asks if there
is a cycle in the graph that visits every vertex exactly once, we first consider the directed
version of this problem. In Section 6 we use \( \text{DIRECTED HAMILTONIAN CYCLE} \), which
makes it easier to obtain a cross-composition, where we start from \( \text{HAMILTONIAN PATH} \)
on a restricted graph class. The kernelization lower bound for \( \text{HAMILTONIAN CYCLE} \) is
now obtained by showing how to transform an instance of \( \text{DIRECTED HAMILTONIAN CYCLE} \) to one of \( \text{HAMILTONIAN CYCLE} \), in such a way that the lower bound carries over.

In Section 7 we obtain a lower bound for \( \text{DOMINATING SET} \). An input to the problem
consists of a graph \( G \) and integer \( k \). The problem asks if there exists a subset of the
vertices \( S \) of size at most \( k \), such that every vertex in \( V(G) \setminus S \) has a neighbor in \( S \). We
use a combination of gadgets that were used to obtain lower bounds for a different
parameterization of \( \text{DOMINATING SET} \) [10]. The same cross-composition can be used to
obtain lower bounds for \( \text{CONNECTED DOMINATING SET} \), which as the name suggests
asks for a dominating set that is connected. We show that these problems do not have
a generalized kernel of size \( \mathcal{O}(n^{2-\varepsilon}) \), unless \( \text{NP} \subseteq \text{coNP}/\text{poly} \). These results carry over
to the dual problems, which has interesting implications, since both problems have a
kernel with a linear number of vertices, when parameterized by the solution size ([7],

\begin{align*}
\end{align*}
Our results imply that it is very unlikely that we can further reduce the number of edges in these kernels to a sub-quadratic amount.

Finally, in Section 8, we look at $d$-NOT-ALL-EQUAL SAT and the less well-known problem $d$-HYPERGRAPH 2-COLORABILITY. Given a hypergraph where every edge contains at most $d$ vertices, the latter problem asks for a 2-coloring of the vertices such that each edge contains at least one vertex of each color. It is known that NOT-ALL-EQUAL-SAT does not have a generalized kernel of size $O(n^{d-1-\epsilon})$, unless NP $\subseteq$ coNP/poly [18], and using a linear parameter reduction we show that this also holds for $d$-HYPERGRAPH 2-COLORABILITY. Since every clause may contain $d$ literals in $d$-NAE-Sat, it is not immediately clear why this lower bound is tight. We present a kernel for $d$-HYPERGRAPH 2-COLORABILITY using $2 \cdot n^{d-1}$ hyperedges, which matches the lower bound. This results in a generalized kernel for $d$-NAE-SAT. This is surprising when we compare with the “normal” satisfiability problem, since $d$-CNF-SAT is not believed to have a kernel of size $O(n^{d-\epsilon})$ for any $\epsilon > 0$ [8].

Related work A lot of work on proving kernelization lower bounds with different parameterizations has already been done. FEEDBACK ARC SET is mostly studied when given a tournament as an input, which is a directed graph $T$ such that exactly one of the arcs $(u, v)$ or $(v, u)$ is in $T$ for every pair of vertices $u$ and $v$. The problem is known to have a kernel whose number of vertices is linear in the parameter $k$ [1]. Note that, since this problem asks for a subset of arcs, $k$ is in this case bounded by the number of edges, not by the number of vertices.

For $q$-Coloring, which asks if it is possible to properly color the vertices of a graph with at most $q$ colors, a study was done on the size of kernels for a number of different parameterizations [18]. For example the parameterization by the size $k$ of a vertex cover in the graph is considered. It is shown that for $q$ at least four, there is no kernel of size $O(k^{q-1-\epsilon})$. Furthermore they obtain a kernel with $O(k^{q})$ vertices that can be encoded in $O(k^{q})$ bits.

LONG PATH and LONG CYCLE asking for a simple path, respectively cycle, of length at least $k$ are also studied under various parameterizations [4]. Parameterized by the size of a vertex cover both problems yield a kernel of quadratic size. When parameterized by the solution size existence of a polynomial kernel would imply NP $\subseteq$ coNP/poly.

The construction of a cross-composition does not always have to be deterministic in order to prove lower bounds. This idea was first used in [20] to prove that a Ramsey-type problem does not have a polynomial kernel, parameterized by $k$. This problem asks if a given graph contains an independent set or a clique of size at least $k$. In this thesis, all constructions will be deterministic.

It was already known that $d$-NAE-SAT does not have a kernel of size $O(n^{d-1-\epsilon})$ unless NP $\subseteq$ coNP/poly [18]. By our results it follows that this bound is in some way tight, we can reduce the number of clauses to $O(n^{d-1})$.  

8
2 Preliminaries

For a (hyper)graph $G$ let $V(G)$ denote its set of vertices and $E(G)$ its set of (hyper)edges. Let a $d$-hypergraph be a hypergraph where every edge contains at most $d$ vertices. We call a hypergraph $d$-uniform if every edge contains exactly $d$ vertices, by this definition a normal graph is exactly the same as a 2-uniform hypergraph. For a set of vertices $S \subseteq V(G)$ we use $G[S]$ to denote the graph induced by $S$, more formally $G[S]$ has vertex set $S$ and set of edges $\{\{u,v\} \in S \times S \mid \{u,v\} \in E(G)\}$. Graphs occurring in this thesis are undirected, unless specified otherwise. They do not contain self-loops and there are no duplicate edges. A graph $G$ is bipartite if there exists a partitioning of its vertices into sets $S$ and $T$, such that $G[S]$ and $G[T]$ are edgeless. This means all edges in $G$ have one endpoint in $S$ and one in $T$.

Let $[r]$ be defined as $[r] := \{x \in \mathbb{N} \mid 1 \leq x \leq r\}$.

There are many possible parameters for a given problem. For example the size of the required solution, or the size of a minimum vertex cover in the graph. Once the parameterization is chosen, upper and lower bounds on kernel sizes (see Definition 2.2) can be derived. Note that in the definition above $f(k)$ can be any computable function, resulting in an unfeasibly slow algorithm for even relatively small values of $k$.

In this thesis, the number of vertices in the graph will be used as the parameter. It is obvious that the problem will then be FPT and have a polynomial kernel, but this choice allows us to prove sparsification lower bounds using the framework of cross-composition.

Let us start by giving the formal definition of a (generalized) kernel.

► Definition 2.2 ((Generalized) Kernelization [5]). Let $Q, Q' \subseteq \Sigma^* \times \mathbb{N}$ be parametrized problems and let $h: \mathbb{N} \to \mathbb{N}$ be a computable function. A generalized kernelization for $Q$ into $Q'$ of size $h(k)$ is an algorithm that on input $(x,k) \in \Sigma^* \times \mathbb{N}$ takes time polynomial in $|x| + k$ and outputs an instance $(x',k')$ such that the following hold:

- The size of $x'$ and the parameter value $k'$ are bounded by $h(k)$
- The instance $(x',k')$ is YES for $Q'$ and only if $(x,k)$ is YES for $Q$

The algorithm is a kernelization, or in short a kernel for $Q$ if $Q = Q'$.

It is known that every problem in FPT has a kernel (cf. [2]). Note that for generalized kernels, there are no real restrictions on the output problem $Q'$. In particular, $Q'$ does
not have to be decidable. The preprocessing algorithms given in this thesis are actually “useful”, they return an instance for the starting problem or some other NP-complete problem. The lower bounds we present even hold against generalized kernels, which is stronger than lower bounds for normal kernels.

To prove kernelization lower bounds, one possibility is to give a \textsc{AND}/\textsc{OR}-cross-composition (Definition 2.4), where we are asked to combine instances of an NP-hard problem into a single instance of the goal problem, that will act as a logical \textsc{AND} or \textsc{OR} on the given inputs. When giving a \textsc{AND}/\textsc{OR}-cross-composition it is often easy to assume that the given input instances are somehow similar. Therefore, the notion of a polynomial equivalence relation is introduced.

\textbf{Definition 2.3 (Polynomial equivalence relation [5]).} An equivalence relation $\mathcal{R}$ on $\Sigma^*$ is called a polynomial equivalence relation if the following two conditions hold:

- There is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether $x$ and $y$ belong to the same equivalence class in time polynomial in $|x| + |y|$.
- For any finite set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of $S$ into a number of classes that is polynomially bounded in the size of the largest element of $S$.

For graph problems, a polynomial equivalence relation can define input instances to be equivalent if they share the same number of vertices. It can also be useful to require that all instances in the same equivalence class ask for a solution of the same size, such as for a dominating set of size $k$. For bipartite graphs a polynomial equivalence relation can be used to ensure that the sizes of the two partite sets are the same.

Let us give an example equivalence relation $\mathcal{R}$ that ensures some of these properties and verify that it a polynomial equivalence relation. Suppose our inputs are instances for \textsc{Dominating Set} on bipartite graphs. This problems gets a bipartite graph $G$ and integer $k$ as input and asks if there exists a set $S \subseteq V(G)$ of size at most $k$, such that every vertex in $G$ has a neighbor in $S$ or is contained in $S$. Given two instances $(G, k)$ and $(G', k')$ where $G$ is bipartite with $V(G) = S \cup T$ and similarly $G'$ is bipartite with $V(G') = S' \cup T'$ and these partitions are given on input. Let these instances be equivalent under $\mathcal{R}$ if and only if $|S'| = |S|$, $|T'| = |T|$ and $k = k'$. It is easy to see that $\mathcal{R}$ is an equivalence relation and that we can decide in polynomial time whether two strings are equivalent under $\mathcal{R}$. It remains to show that the second condition is satisfied, to do this we will first define how the inputs are stored. Inputs are given as the adjacency matrix of the graph, followed by the indices of all vertices in one side of the partition, followed by the parameter $k$ (in unary). Suppose we are given a set $S$ of input strings, and let the largest element have length $x$, we need to show that the number of equivalence classes in $S$ is polynomial in $x$. We know that any instance in $S$ has at most $x$ vertices and parameter $k$ bounded by $x$. As such, the number of different parameter values occurring in $S$ is at most $x$. The number of combinations of the size of partition sets $S$ and $T$ is bounded by $x^2$. It follows that the number of equivalence classes under $\mathcal{R}$ is bounded by $x^3$, which is polynomial as desired.
Definition 2.4 (AND/OR-cross-composition [5]). Let $L \subseteq \Sigma^*$ be a language, let $\mathcal{R}$ be a polynomial equivalence relation on $\Sigma^*$, and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. An OR-cross-composition of $L$ into $Q$ (with respect to $\mathcal{R}$) of cost $f(t)$ is an algorithm that, given $t$ instances $x_1, \ldots, x_t \in \Sigma^*$ of $L$ belonging to the same equivalence class of $\mathcal{R}$, takes time polynomial in $\sum_{i=1}^{t} |x_i|$ and outputs an instance $(y, k) \in \Sigma^* \times \mathbb{N}$ such that the following hold:

- **CB** The parameter $k$ is bounded by $O(f(t) \cdot (\max_{i=1}^{t} |x_i|^c))$, where $c$ is some constant independent of $t$.
- **OR** The instance $(y, k)$ is YES for $Q$ if and only if at least one instance $x_i$ is YES for $L$.
- **AND** The instance $(y, k)$ is YES for $Q$ if and only if all instances $x_i$ are YES for $L$. ▲

Depending on the cost of a given AND/OR-cross-composition, it is possible to derive lower bounds for the size of generalized kernels of a parameterized problem. We will use AND/OR-cross-compositions of the following cost to obtain our lower bounds.

Definition 2.5 (Degree-$d$ AND/OR-cross-composition). Let a degree-$d$ AND/OR-cross-composition be an AND/OR-cross-composition of bounded cost where $f(t) \in O(t^{1/d+o(1)})$. We will also use degree-$d$ cross-composition to refer to the OR variant. ▲

Theorem 2.6 ([5]). Let $L \subseteq \Sigma^*$ be a language, let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem and let $d, \varepsilon$ be positive reals. If $L$ has a degree-$d$ OR-cross-composition into $Q$ with cost $f(t) = t^{1/d+o(1)}$, where $t$ denotes the number of instances, and $Q$ has a generalized kernel with size bounded by $O(k^{d-\varepsilon})$, then $L \in \text{coNP} / \text{poly}$. If, additionally, $L$ is NP-hard, then $\text{NP} \subseteq \text{coNP} / \text{poly}$. ▲

As such, a degree-2-OR-cross-composition can be used to show that a parameterized problem does not have a kernel of subquadratic size, unless $\text{NP} \subseteq \text{coNP} / \text{poly}$.

Suppose we want to give a degree-2 AND/OR cross-composition to a problem on graphs, such as 4-COLORING. Let us for now use a simple equivalence relation that defines all input graphs with the same number of vertices to be equivalent. Suppose we are given input graphs $G_1, \ldots, G_t$, each consisting of $n$ vertices.

Consider the most basic cross-composition, let the output graph $G$ simply be the union of all input graphs, meaning $V(G) = \bigcup_{i \in [t]} V(G_i)$ and $E(G) = \bigcup_{i \in [t]} E(G_i)$. It is easy to see that we have now satisfied property AND in Definition 2.4, so this is an AND-cross composition. However, it is not a degree-2 AND-cross composition, since the constructed graph $G$ has $\sum_{i \in [t]} V(G_i) = n \cdot t$ vertices. The definition of a degree-2 AND-cross-composition only allows us to use $O(n^{c} \cdot \sqrt{t})$ vertices for some constant $c$. Thereby, we can not preserve all vertices of all input instances.

The constructions given in this paper, all rely on the same basic idea that allows us to keep all edges of all input instances, while forgetting about most of the vertices. As a starting problem, choose an appropriate NP-hard problem on a restricted class of graphs.
This graph class is chosen such that the vertices of all input instances can be partitioned into two sets, such that we know the structure of the graph induced by either partite set. For example, a bipartite graph, where each of the sets induces an edgeless graph. Given \( t \) instances of your favorite NP-hard problem on bipartite graphs, we construct a graph for our output problem by first creating \( 2\sqrt{t} \) independent sets of appropriate size. We represent an input instance \( x \) by connecting two such independent sets such that the graph they induce is exactly \( x \). It then remains to construct additional gadgets to obtain a logical OR of all inputs. See Figure 1 for an example of this construction using 9 input graphs.

![Figure 1](image_url)

**Figure 1.** The basic idea used to keep all edges in a degree-2 cross-composition, demonstrated with nine small bipartite input graphs.

In order to use obtained lower bounds to prove lower bounds for other parameterized problems, a linear parameter transformation is defined as follows.

**Definition 2.7 (Linear parameter transformation [17]).** Let \( L_1 \) and \( L_2 \) be two parameterized problems. We say that \( L_1 \) is linear parameter reducible into \( L_2 \), written \( L_1 \leq_{\text{lpt}} L_2 \), if there exists a polynomial time computable function \( f \) mapping instances of \( L_1 \) into instances of \( L_2 \), such that for all \((x, k) \in \Sigma^* \times \mathbb{N}\), if \((x', k') = f(x, k)\) then:

- \((x, k) \in L_1 \iff (x', k') \in L_2\)
- \(k' = O(k)\).

In some cases an existing NP-hardness reduction can be used as a linear parameter transformation. We give an example of this in Section 3, where we use the NP-completeness proof of FEEDBACK ARC SET by Karp [19] to obtain a sparsification lower bound for this problem.

**Theorem 2.8 ([17],[18]).** Let \( L_1 \) and \( L_2 \) be two parameterized problems, and let \( d \in \mathbb{N} \) be some constant. If \( L_1 \leq_{\text{lpt}} L_2 \) and \( L_2 \) has a generalized kernel of size \( O(k^d) \), then \( L_1 \) also has a generalized kernel of size \( O(k^d) \).
Theorem 2.8 will be used to show that if parameterized problem \( L_1 \) does not have a kernel of size \( O(k^d) \), and \( L_1 \leq_{ipt} L_2 \), then neither does \( L_2 \).
3 Feedback Arc Set

In this section we will show a kernelization lower bound for Feedback Arc Set (FAS). In the Feedback Arc Set problem we are given a directed graph \( G \) and an integer \( k \). We ask whether it is possible to remove at most \( k \) arcs from \( G \), such that the result is a directed acyclic graph. To prove a kernelization lower bound, we will use a linear parameter transformation from Vertex Cover. A vertex cover of a graph is a subset \( S \) of its vertices, such that for every edge in the graph at least one of its endpoints is contained in \( S \), we also say that this edge is covered by (some vertex in) \( S \). In the Vertex Cover problem we ask if there exists a vertex cover in the graph that has size at most \( k \).

Dell et al. showed a stronger result of the following Lemma, in Theorem 2 in [8].

▶ Lemma 3.1. Vertex Cover parameterized by the number of vertices \( n \) does not have a generalized kernel of size \( O(n^2-\varepsilon) \) for any \( \varepsilon > 0 \), unless \( \text{NP} \subseteq \text{coNP}/\text{poly} \). ◁

The reduction given below is equal to the NP-hardness proof of FAS given by Karp in [19].

▶ Theorem 3.2. Feedback Arc Set parameterized by the number of vertices \( n \) does not have a generalized kernel of size \( O(n^2-\varepsilon) \), unless \( \text{NP} \subseteq \text{coNP}/\text{poly} \).

Proof. Suppose we are given an undirected graph \( G = (V, E) \) and ask if it has a vertex cover of size at most \( k \). We construct a directed graph \( G' = (V', E') \) for feedback arc set in the following way. For every vertex in \( v \), add vertices \( \mathit{v_{in}} \) and \( \mathit{v_{out}} \) to \( V' \) and arc \( (\mathit{v_{in}}, \mathit{v_{out}}) \) to \( E' \). For every edge \( \{u, v\} \) in \( E \), we add arcs \( (\mathit{v_{out}}, \mathit{v_{in}}) \) and \( (\mathit{v_{out}}, \mathit{u_{in}}) \) to \( E' \). See Figure 2 for an example. It remains to show validity of this reduction.

\[\begin{align*}
\mathit{u_{in}} & \quad \mathit{v_{in}} & \quad \mathit{w_{in}} \\
\mathit{u_{out}} & \quad \mathit{v_{out}} & \quad \mathit{w_{out}} \\
\mathit{u} & \quad \mathit{v} & \quad \mathit{w}
\end{align*}\]

Figure 2. Reduction from Vertex Cover (left) to Feedback Arc Set (right) on a small graph.

▷ Claim 3.3. \( G \) has a vertex cover of size \( k \) if and only if \( G' \) has a feedback arc set of size \( k \).

Proof. \((\Rightarrow)\) Let \( G \) have a vertex cover \( S \) of size at most \( k \). For every vertex \( v \in S \), add \( (\mathit{v_{in}}, \mathit{v_{out}}) \) to \( S' \), which will be the feedback arc set for \( G' \). Therefore it remains to prove that \( G'[V' \setminus S'] \) is acyclic. Suppose for contradiction that \( G' \) contains a cycle after removing all arcs in \( S' \) from \( E' \). A cycle cannot consist of only arcs of type \( (\mathit{u_{in}}, \mathit{u_{out}}) \), so it must contain an arc \( (\mathit{u_{out}}, \mathit{v_{in}}) \) for some \( u, v \in V' \). Since this implies that \( \{u, v\} \) was an
edge in $E$, either $u$ or $v$ must be in the vertex cover $S$ of $G$. If $u \in S$, then $(u_{\text{in}}, u_{\text{out}})$ was added to $S'$. Therefore, vertex $u_{\text{out}}$ has in-degree 0 in $G'[V' \setminus S']$ and can never be part of a cycle. Similarly, if $v \in S$, then $(v_{\text{in}}, v_{\text{out}}) \notin G'[V' \setminus S']$. Consequently, vertex $v_{\text{in}}$ has out-degree 0 and can not be part of a cycle. So, either $v_{\text{in}}$ or $u_{\text{out}}$ can not be contained in a cycle, implying that the edge $(u_{\text{out}}, v_{\text{in}})$ is not in a cycle. This contradicts the initial assumption, therefore, $G'[V' \setminus S']$ is acyclic.

$(\Leftarrow)$ Now suppose we are given a feedback arc set $S'$ in $G'$ of size at most $k$. If the FAS contains an arc $(u_{\text{out}}, v_{\text{in}})$, we can replace it by the arc $(v_{\text{in}}, v_{\text{out}})$ since any cycle traversing the first arc, must also traverse $(v_{\text{in}}, v_{\text{out}})$. Therefore, we may assume that is contains only arcs of type $(v_{\text{in}}, v_{\text{out}})$. Now we add vertex $v$ to the vertex cover $S$ of $G$ if $(v_{\text{in}}, v_{\text{out}})$ is in the feedback arc set of $G'$. Now suppose for contradiction that $S$ is not a proper vertex cover, so at least one edge $\{u, v\}$ is not covered. This implies that arcs $(u_{\text{in}}, u_{\text{out}})$ and $(v_{\text{in}}, v_{\text{out}})$ are not present in $S'$. Since $\{u, v\} \in E$, it follows that arcs $(u_{\text{out}}, v_{\text{in}})$ and $(v_{\text{out}}, u_{\text{in}})$ are contained in $E'$ and these edges are not present in $S'$ by assumption. But then, $G'[V' \setminus S']$ contains the directed cycle $(u_{\text{in}}, u_{\text{out}}, v_{\text{in}}, v_{\text{out}}, u_{\text{in}})$, which contradicts that $S'$ is a Feedback Arc Set of $G'$. So, $S$ must be a Vertex Cover of $G$. 

The reduction given above satisfies all the properties of a linear-parameter transformation from VERTEX COVER parameterized by the number of vertices $n$, to FEEDBACK ARC SET parameterized by the number of vertices $n$ (Definition 2.7). Suppose FEEDBACK ARC SET does have a sub-quadratic generalized kernel, by Theorem 2.8 this implies that VERTEX COVER also has a sub-quadratic generalized kernel. But then, using Lemma 3.1 it follows that $\text{NP} \subseteq \text{coNP}/\text{poly}$. ▲
4 4-Coloring

In this section we will look at the 4-COLORING problem. A $k$-coloring of a graph $G$ is a function $c: V(G) \rightarrow [k]$. The 4-COLORING problem asks there exists a proper 4-coloring of a given input graph. A graph coloring is proper if the two endpoints of any edge in the graph are assigned distinct colors. We show that 4-COLORING does not have a generalized kernel of size $O(n^{2-\varepsilon})$, by giving a degree-2 cross-composition from an NP-hard problem that will be introduced later. Before giving the construction, we first present some of the gadgets that will be needed and their most important properties.

▶ Definition 4.1 (Treegadget). A treegadget is the graph obtained from a complete binary tree by replacing each vertex $v$ by a triangle on vertices $r_{v}$, $x_{v}$ and $y_{v}$. Let $r_{v}$ be connected to the parent of $v$ and let $x_{v}$ and $y_{v}$ be connected to the left and right subtree of $v$. An example of a treegadget with 8 leaves is shown in Figure 3. If vertex $v$ is the root of the tree, then $r_{v}$ is named the root of the treegadget. If $v$ does not have a left subtree, then $x_{v}$ is a leaf of this gadget, similarly, if $v$ does not have a right subtree then we refer to $y_{v}$ as a leaf of the gadget. Let the height of a treegadget be equal to the height of its corresponding binary tree.

▶ Lemma 4.2. Let $T$ be a treegadget with root $r$ and let $c: V(T) \rightarrow \{1, 2, 3\}$ be a proper 3-coloring of $T$. If $k \in \{1, 2, 3\}$ such that $c(v) \neq k$ for every leaf $v$ of $T$, then $c(r) = k$.

Proof. This will be proven using induction on the structure of a treegadget. For a single triangle, the result is obvious. Suppose we are given a treegadget of height $h$ and that the statement holds for all treegadgets of smaller height. Consider the top triangle $r$, $x$, $y$ where $r$ is the root. For both subtrees of this triangle, it is clear that all leaves in the

- Color the root $r$ with color $k$.
- Color the leaves of the left subtree with colors $1$ and $2$.
- Color the leaves of the right subtree with colors $2$ and $3$.

Figure 3. Treegadgets with example colorings.

It is easy to see that a treegadget is 3-colorable. The important property of this gadget is that if there is a color that does not appear on any leaf in a proper 3-coloring, then this must be the color of the root. Consequently, if the root is not colored with a certain color, then at least one of the leaves must be. See Figure 3a for an illustration. This property will be used to prove that if the graph constructed for the cross-composition has a 4-coloring, at least one input instance was a YES-instance.

▶ Lemma 4.2. Let $T$ be a treegadget with root $r$ and let $c: V(T) \rightarrow \{1, 2, 3\}$ be a proper 3-coloring of $T$. If $k \in \{1, 2, 3\}$ such that $c(v) \neq k$ for every leaf $v$ of $T$, then $c(r) = k$.

Proof. This will be proven using induction on the structure of a treegadget. For a single triangle, the result is obvious. Suppose we are given a treegadget of height $h$ and that the statement holds for all treegadgets of smaller height. Consider the top triangle $r$, $x$, $y$ where $r$ is the root. For both subtrees of this triangle, it is clear that all leaves in the

- Color the root $r$ with color $k$.
- Color the leaves of the left subtree with colors $1$ and $2$.
- Color the leaves of the right subtree with colors $2$ and $3$.

Figure 3. Treegadgets with example colorings.

It is easy to see that a treegadget is 3-colorable. The important property of this gadget is that if there is a color that does not appear on any leaf in a proper 3-coloring, then this must be the color of the root. Consequently, if the root is not colored with a certain color, then at least one of the leaves must be. See Figure 3a for an illustration. This property will be used to prove that if the graph constructed for the cross-composition has a 4-coloring, at least one input instance was a YES-instance.

▶ Lemma 4.2. Let $T$ be a treegadget with root $r$ and let $c: V(T) \rightarrow \{1, 2, 3\}$ be a proper 3-coloring of $T$. If $k \in \{1, 2, 3\}$ such that $c(v) \neq k$ for every leaf $v$ of $T$, then $c(r) = k$.

Proof. This will be proven using induction on the structure of a treegadget. For a single triangle, the result is obvious. Suppose we are given a treegadget of height $h$ and that the statement holds for all treegadgets of smaller height. Consider the top triangle $r$, $x$, $y$ where $r$ is the root. For both subtrees of this triangle, it is clear that all leaves in the
subtree are colored using \( \{1, 2, 3\} \setminus k \). Thus, it follows from the induction hypothesis that the roots of the left and right subtree are colored using \( k \). Hence \( x \) and \( y \) cannot use color \( k \). Since \( x, y, r \) is a triangle, \( r \) has color \( k \) in the 3-coloring.

The following lemma will be used to argue that if one of the input instances is a \textit{YES} instance, we can extend the coloring of this input to a proper 4-coloring of the graph constructed for the cross-composition.

\begin{lemma}
Let \( T \) be a tree gadget with leaves \( L \subseteq V(T) \) and root \( r \). Any 3-coloring \( c' : L \rightarrow \{1, 2, 3\} \) that is proper on \( T[L] \) can be extended to a proper 3-coloring of \( T \). If there is a leaf \( v \in L \) such that \( c'(v) = i \), then such an extension exists with \( c(r) \neq i \).
\end{lemma}

\textit{Proof.} We will prove this by induction on the height of the tree gadget. For a single triangle, the result is obvious. Suppose the lemma is true for all tree gadgets up to height \( h - 1 \) and we are given a tree gadget of height \( h \) with root triangle \( r, x, y \) and with coloring of the leaves \( c' \). Let one of the leaves be colored using \( i \). Without loss of generality assume this leaf is in the left subtree, which is connected to \( x \). By the induction hypothesis, we can extend the coloring restricted to the leaves of the left subtree to a proper 3-coloring of the left subtree such that \( c(r_1) \neq i \). We assign color \( i \) to \( x \). Since \( c' \) restricted to the leaves in the right subtree is a proper 3-coloring of the leaves in the right subtree, by induction we can extend that coloring to a proper 3-coloring of the right subtree. Suppose the root of this subtree gets color \( j \in \{1, 2, 3\} \). We now color \( y \) with a color \( k \in \{1, 2, 3\} \setminus \{i, j\} \), which must exist. Finally, choose \( c(r) \in \{1, 2, 3\} \setminus \{i, k\} \). By definition, the vertices \( r, y, \) and \( x \) are now assigned a different color. Both \( x \) and \( y \) have a different color than the root of their corresponding subtree, thereby \( c \) is a proper coloring. We obtain that the defined coloring \( c \) is a proper coloring extending \( c' \) with \( c(r) = i \). An example is shown in Figure 3b.

\begin{definition}[Triangular gadget]
A triangular gadget is the graph on 12 vertices depicted in Figure 4. Vertices \( u, v, \) and \( w \) are the corners of the gadget, all other vertices are referred to as inner vertices.
\end{definition}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{triangular_gadget.png}
\caption{Triangular gadget.}
\end{figure}

It is easy to see that a triangular gadget is always 3-colorable in such a way that every corner gets a different color. Moreover, we can prove the following lemma.
Lemma 4.5. Let $G$ be a triangular gadget with corners $u,v$ and $w$ and let $c : V(G) \to \{1,2,3\}$ be a proper 3-coloring of $G$. Then $c(v) \neq c(u) \neq c(w) \neq c(v)$. Furthermore, every partial coloring that assigns distinct colors to the three corners of a triangular gadget can be extended to a proper 3-coloring of the entire gadget.

Proof. Suppose there is a proper 3-coloring $c$ of the entire gadget, such that $c(u) = c(v) = i$ for two distinct corners of the gadget and $i \in \{1,2,3\}$. Note that the inner vertices of the gadget consist of three triangles. There is one such triangle for which every of its vertices has either $u$ or $v$ as a neighbor. This implies that color $i$ cannot be used in this triangle, which means that $c$ can never be a proper 3-coloring of the gadget.

Given a partial coloring that assigns distinct colors to the corners, it is easy to extend this coloring to a coloring of the entire gadget, as shown in Figure 4.

This concludes the description of the gadgets that will be used. We now define the source problem for the cross-composition. It is a modification of 3-COLORING WITH TRIANGLE SPLIT DECOMPOSITION, which is given as input a graph $G$ with partition of its vertex set into $X \cup Y$ such that $G[X]$ is an edgeless graph and $G[Y]$ is a disjoint union of triangles. Such a graph is called a triangle split graph, the problem asks if this graph has a proper 3-coloring. The problem we define furthermore requires that the independent set is colored using only 2 colors, as defined below. 3-COLORING WITH TRIANGLE SPLIT DECOMPOSITION was used by Bodlaender et al. to prove kernel lower bounds for CHROMATIC NUMBER parameterized by the size of a vertex cover [5].

2-3-COLORING WITH TRIANGLE SPLIT DECOMPOSITION

**Input:** A graph $G$ with a partition of its vertex set into $X \cup Y$ such that $G[X]$ is an edgeless graph and $G[Y]$ is a disjoint union of triangles.

**Question:** Is there a proper 3-coloring $c : V(G) \to \{1,2,3\}$ of $G$, such that $c(x) \in \{1,2\}$ for all $x \in X$? We will refer to such a coloring as a 2-3-coloring of $G$.

Lemma 4.6. 2-3-COLORING WITH TRIANGLE SPLIT DECOMPOSITION is NP-complete.

Proof. It is easy to verify the problem is in NP. We will show that it is NP-hard by giving a reduction from 3-NOT-ALL-EQUAL-SATISFIABILITY (3-NAE-SAT), which is known to be NP-complete [16]. For the 3-NAE-SAT problem we are given a Boolean formula in conjunctive normal form (CNF). This means we are given a conjunction of clauses, where each clause is a disjunction of exactly three literals. We ask if there exists a truth assignment to the variables such that every clause contains at least one true and at least one false literal.

Suppose we are given formula $F = C_1 \land C_2 \land \ldots \land C_m$ over set of variables $U$. Construct graph $G$ in the following way. For every variable $x \in U$, construct a variable gadget as depicted in Figure 5a. For every clause $C_i$, construct a clause gadget as depicted in Figure 5b. For $C_i = (\ell_1 \lor \ell_2 \lor \ell_3)$ with $i \in [m]$, connect vertex $\ell_j$ for $j \in \{1,2,3\}$ to vertex $v_j$ in gadget $C_i$ in $G$. 18
It is easy to verify that $G$ has a triangle split decomposition. In Figure 5, triangles are shown with white vertices and the independent set is shown in black.

Suppose $G$ is 2-3-colorable with color function $c : V(G) \rightarrow \{1, 2, 3\}$ and let $c(v) \in \{1, 2\}$ for all $v$ in the independent set. Note that in each of the pairs \{x, ¬x\}, \{b_1, b_2\}, and \{u_1, u_2\} the two vertices have distinct colors in any proper 2-3-coloring of $G$. To satisfy $F$, let $x = true$ if and only if $c(x) = 2$. To show that this results in a satisfying assignment, consider any clause $C_i$ for $i \in [m]$. Note that $c(x) = 2 \Leftrightarrow c(¬x) = 1$. Since $c(b_1) \neq c(b_2)$ and $c(b_1), c(b_2) \in \{1, 2\}$ we obtain $c(r_0) = 3$. Therefore, $v_1$ and $u_0$ are colored using colors 1 and 2.

Suppose $c(v_1) = 1$. In the construction we added an edge from $v_1$ to $\ell_1$ and thereby $c(\ell_1) = 2$. This implies that the first literal of $C_i$ is set to true. By $c(u_0) = 2$, we know $c(u_1) = 1$ and $c(u_2) = 2$. Therefore, $c(r_1) \neq 2$, so either $c(v_2) = 2$ or $c(v_3) = 2$. If $c(v_2) = 2$, then $c(\ell_2) = 1$ which implies that literal $\ell_2$ is false in $C_i$. Similarly, if $c(v_3) = 2$, then $c(\ell_3) = 1$ which implies that literal $\ell_3$ is false in $C_i$. In both cases it follows that clause $C_i$ is NAE-satisfied.

When $c(v_1) = 2$, we can use the same argument with the colors 1 and 2 swapped, to show that $\ell_1$ is false in $C_i$ and $\ell_2$ or $\ell_3$ is true, which again implies that $C_i$ is NAE-satisfied.

Suppose $F$ is a YES-instance, with satisfying truth assignment $S$. Define color function $c : V(G) \rightarrow \{1, 2, 3\}$ as $c(x) := 1$ and $c(¬x) := 2$ if $x$ is set to false in $S$, define $c(x) := 2$ and $c(¬x) := 1$ otherwise. Color the remainder of the variable gadgets consistently.

We now need to show how to color the clause gadgets. Consider any clause $C_i = (\ell_1 \lor \ell_2 \lor \ell_3)$. At least one of the literals is true and one is set to false, by symmetry we only consider four cases. The corresponding colorings are depicted in Figure 6, where red corresponds to 1, green corresponds to 2 and blue corresponds to color 3. It is easy to verify that this leads to a proper 3-coloring that only uses colors 1 and 2 on vertices in the independent set.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{gadget.png}
\caption{The gadgets constructed for the clauses and variables of $F$.}
\end{figure}
Theorem 4.7. \textsc{4-Coloring} parameterized by the number of vertices $n$ does not have a generalized kernel of size $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP} / \text{poly}$.

Proof. By Theorem 2.6 and Lemma 4.6 it suffices to give a degree-2 cross-composition from the 2-3-coloring problem defined above into \textsc{4-Coloring} parameterized by the number of vertices. For ease of presentation, we will actually give a cross-composition into the \textsc{4-List Coloring} problem, whose input consists of a graph $G$ and a list function that assigns every vertex $v \in V(G)$ a list $L(v) \subseteq [4]$ of allowed colors. The question is whether there is a proper coloring of the graph in which every vertex is assigned a color from its list. The \textsc{4-List Coloring} problem reduces to the ordinary \textsc{4-Coloring} by a simple transformation that adds a 4-clique to enforce the color lists, which will prove the theorem. For now, we focus on giving a cross-composition into \textsc{4-List Coloring}.

We start by defining a polynomial equivalence relation on inputs of \textsc{2-3-Coloring with Triangle Split Decomposition}. Let two instances of \textsc{2-3-Coloring with Triangle Split Decomposition} be equivalent under equivalence relation $R$ when they have the same number of triangles and the independent sets have the same size. It is easy to see that $R$ is a polynomial equivalence relation. By duplicating one of the inputs, we can ensure that the number of inputs to the cross-composition is an even power of two; this does not change the value of OR, and increases the total input size by at most a factor four. We will therefore assume that the input consists of $t$ instances of \textsc{2-3-Coloring with Triangle Split Decomposition} such that $t = 2^{2i}$ for some integer $i$, implying that $\sqrt{t}$ and $\log \sqrt{t}$ are integers. Let $t' := \sqrt{t}$. Enumerate the instances as $X_{ij}$ for $1 \leq i, j \leq t'$. Each input $X_{ij}$ consists of a graph $G_{ij}$ and a partition of its vertex set into sets $U$ and $V$, such that $U$ is an independent set of size $m$ and $G_{ij}[V]$ consists of $n$ vertex-disjoint triangles. Enumerate the vertices in $U$ and $V$ as $u_1, \ldots, u_m$ and $v_1, \ldots, v_{3n}$, such that vertices $v_{3\ell-2}, v_{3\ell-1}$ and $v_{3\ell}$ form a triangle, for $\ell \in [n]$. We will create an instance $G'$ of the \textsc{4-List-Coloring} problem, which consists of a graph $G'$ and a list

Figure 6. Colorings of a clause gadget, depending on the coloring of the literals $\ell_1, \ldots, \ell_3$. Note that if the roles of $\ell_2$ and $\ell_3$ are exactly reversed, you can just exchange colors between their parents to get a proper coloring for that situation.
function $L$ that assigns each vertex a subset of the color palette. For ease of presentation we use the letters $\{x, y, z, a\}$ to indicate the four colors, instead on the numbers one to four. Refer to Figure 7 for a sketch of $G'$.

We start the construction by creating the basic graph that keeps all input edges as described in the preliminaries, where we get triangle-split graphs instead of bipartite graphs as inputs.

1. Initialize $G'$ as the graph containing $t'$ sets of $m$ vertices each, called $S_i$ for $i \in [t']$. Label the vertices in each of these sets as $s^i_\ell$ for $i \in [t'], \ell \in [m]$ and let $L(s^i_\ell) := \{x, y, a\}$.

2. Add $t'$ sets of $n$ triangular gadgets each, labeled $T_j$ for $j \in [t']$. Label the corner vertices in $T_j$ as $t^j_\ell$ for $\ell \in [3n]$, such that vertices $t^j_{3\ell-2}, t^j_{3\ell-1}$ and $t^j_{3\ell}$ are the corner vertices of one of the gadgets for $\ell \in [n]$. Let $L(t^j_\ell) := \{x, y, z\}$ and for any inner vertex $v$ of a triangular gadget, let $L(v) := \{x, y, z, a\}$.

3. Connect vertex $s^i_k$ to vertex $t^j_\ell$ if in graph $G_{ij}$ vertex $u_k$ is connected to $v_\ell$, for $k \in [m]$ and $\ell \in [3n]$. By this construction, the subgraph of $G'$ induced by $S_i \cup T_j$
is isomorphic to the graph obtained from $G_{i,j}$ by replacing each triangle with a triangular gadget.

We will now create the additional structure, that allows us to select one instance that should be 4-colorable. First, we add one treegadget that will ensure that at least one of the groups in $S$ is colored using only the colors $x$ and $y$.

4. Add a treegadget $G_S$ with $t'$ leaves to $G'$ and enumerate these leaves as $1, \ldots, t'$; recall that $t'$ is a power of two. Connect the $i$'th leaf of $G_S$ to every vertex in $S_i$. Let the root of $G_S$ be $r_S$ and define $L(r_S) := \{x, y\}$. For every other vertex $v$ in $G_S$ let $L(v) := \{x, y, a\}$.

For $T$ we create a treegadget that ensures that for one group of $T$, the entire group (including the inner vertices of gadgets) is colored using $\{x, y, z\}$. In other groups, the idea is to color the inner vertices using $\{x, y, a\}$ and the corner vertices using color $z$.

5. Add a treegadget $G_T$ with $2t'$ leaves to $G'$ and enumerate these leaves as $1, \ldots, 2t'$. For $j \in [t']$, connect every inner vertex of a triangular gadget in group $T_j$ to leaf number $2j - 1$ of $G_T$. For every leaf $v$ with an even index let $L(v) := \{y, z\}$ and let the root $r_T$ have list $L(r_T) := \{y, z\}$. For every other vertex $v$ of gadget $G_T$ let $L(v) := \{y, z, a\}$.

By construction, $G_T$ has twice as many leaves as $G_S$. It is important to notice that using a similar construction as for $G_S$—having one leaf per group—is not desirable here. A leaf in $G_T$ is connected to all inner vertices of the triangular gadgets in some group. Consider a group in $T$ that will not be part of our solvable input instance, we want to color its inner vertices using $\{x, y, a\}$. This implies the leaf connected to this group is always colored using $z$. However, we have many such groups and two neighboring leaves cannot both get color $z$. Therefore, we use twice as many leaves where the odd leaves connect to the groups, while the even leaves are just there to fill the tree and keep the nice property of a treegadget we proved in Lemma 4.2.

We can now verify that the created graph $G'$ acts as a logical OR of the given input instances.

\begin{claim}
The graph $G'$ is 4-list colorable $\iff$ some input instance $X_{i', j'}$ is 2-3-colorable.
\end{claim}

\begin{proof}
($\Rightarrow$) Suppose we are given a 4-list coloring $c$ for $G'$. Since no vertex in $G_S$ contains color $z$ in its list, the restriction of $c$ to $G_S$ is a 3-coloring with colors $\{x, y, a\}$ where the root is not colored using $a$ by the definition of its list. From Lemma 4.2 it follows that there is a leaf $v$ of $G_S$ such that $c(v) = a$. This leaf is connected to all vertices in some $S_{i'}$ in Step 4, which implies that none of the vertices in $S_{i'}$ are colored using $a$. Therefore all vertices in $S_{i'}$ are colored using $x$ and $y$. Similarly the gadget $G_T$ has at least one leaf $v$ such that $c(v) = a$, note that this must be a leaf with an odd index by the definition of the lists in Step 5. Therefore there exists $T_{j'}$ where all vertices are colored using $x, y$

\end{proof}
or z. Thereby in $S_r \cup T_r$ only three colors are used, such that $S_r$ is colored using only two colors. Using Lemma 4.5 and the fact that $G'[S_r \cup T_r]$ is isomorphic to the graph obtained from $G_{r,j}$ by replacing triangles by triangular gadgets, we conclude that $X_{r,j}$ has a proper 2-3-coloring.

$(\Leftarrow)$ Suppose $c: V(G_{r,j}) \rightarrow \{x,y,z\}$ is a proper 2-3-coloring for $X_{r,j}$. We will construct a 4-list coloring $c': V(G') \rightarrow \{x,y,z,a\}$ for $G'$. For $u_k$ with $k \in [m]$ in instance $X_{r,j}$ let $c'(s_k^i) := c(u_k)$ and for $v_\ell$ for $\ell \in [3n]$ let $c'(t_\ell^j) := c(v_\ell)$. Let $c'(s_k^i) := a$ for $i \neq i^*$ and $\ell \in [n]$, furthermore let $c'(t_\ell^j) := z$ for $j \neq j^*$ and $\ell \in [3m]$. For triangular gadgets in $T_r$, the coloring $c'$ defines all corners to have distinct colors; by Lemma 4.5 we can color the inner vertices consistently using $\{x,y,z\}$. For $T_j$ with $j \in [t']$ and $j \neq j^*$, the corners of triangular gadgets have color z and we can now consistently color the inner vertices using colors $x, y,$ and $a$.

The leaf of gadget $G_S$ that is connected to $S_r$ can be colored using $a$. Every other leaf can use both $x$ and $y$, so we can properly 3-color the leaves such that one leaf has color $a$. From Lemma 4.3 it follows that we can consistently 3-color $G_S$ such that the root $r_S$ does not receive color $a$, as required by $L(r_S)$. Similarly, in triangular gadgets in $T_r$ the inner vertices do not have color $a$. As such, leaf $2j^* - 1$ of $G_T$ can be colored using $a$ and we color leaf $2j^*$ with $y$. For $j \in [t']$ with $j \neq j^*$ color leaf $2j - 1$ with $z$ and leaf $2j$ using $y$. Now the leaves of $G_T$ are properly 3-colored and one is colored $a$. It follows from Lemma 4.5 that we can color $G_T$ such that the root is not colored $a$. This completes the 4-list coloring of $G'$.

It remains to bound the number of vertices of the constructed graph $G'$. Observe that a tree gadget has at least as many leaves as its corresponding binary tree, therefore the graph $G'$ has at most

$$\frac{mt' + 12nt'}{|S|} + \frac{6t'}{|T|} + \frac{12t'}{|G_s|} + \frac{O(t' \cdot (m + n))}{|G_T|} = O(\sqrt{t} \cdot |X_{i,j}|)$$

vertices. Claim 4.8 shows that we have given a degree-2 cross-composition into 4-LIST COLORING. By Theorem 2.6, and Lemma 4.6 it follows that 4-LIST COLORING does not have a generalized kernel of size $O(n^{2-\epsilon})$, unless NP $\subseteq$ coNP/poly. To prove the theorem about 4-COLORING, we give a linear parameter transformation from 4-LIST COLORING. Given an instance $(G, L)$ for 4-LIST COLORING where every list is a subset of $\{c_1, c_2, c_3, c_4\}$ we construct graph $G'$ for 4-COLORING. Initialize $G'$ as $G$. We now add new vertices to $G'$ to simulate the lists. Add a clique on 4 vertices $\{c_1, c_2, c_3, c_4\}$. If for any vertex $v$ in $G'$, some color is not contained in $L(v)$, connect $v$ to the vertex corresponding to this color. As proper colorings of the resulting graph correspond to proper list colorings of $G'$, the resulting graph is 4-colorable if and only if there is a YES-instance among the inputs. Theorem 4.7 now follows from Theorem 2.8.

[\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{example.png}
\caption{Example figure.}
\end{figure}]

23
5 Planar List Coloring

Since most of the results obtained for 4-COLORING will also carry over to LIST COLORING, we now consider a slightly different problem, which is list coloring on planar graphs:

<table>
<thead>
<tr>
<th>PLANAR LIST COLORING</th>
<th>Parameter: The number of vertices $n$</th>
</tr>
</thead>
</table>
| **Input:** A pair $(G,L)$ where $G$ is an undirected planar graph and color function $L : V(G) \to \mathcal{P}(\mathbb{N})$ that maps each vertex $v$ to a list of colors $L(v)$.
| **Question:** Is there a proper coloring $c : V(G) \to \mathbb{N}$ such that $c(v) \in L(v)$ for all $v \in V(G)$ and furthermore such that $c(u) \neq c(v)$ for every edge $\{u,v\} \in E(V)$? |

Storing the graph does not take a quadratic amount of space here, since planar graphs have a linear number of edges. However, we need to store a list of colors for each vertex, that may have an arbitrary length.

**Theorem 5.1.** PLANAR LIST COLORING parameterized by the number of vertices $n$ has a kernel of size $O(n \log n)$.

**Proof.** We present the following simple reduction rule, to reduce the length of the list of a vertex, depending on its degree.

**RM** Let $v$ be a vertex of the graph with a list of colors with length $k$ and degree $d_v$. If $k > d_v + 1$, remove vertex $v$ from the graph.

The following lemma shows that this reduction rule is valid.

**Lemma 5.2.** If instance $(G,L)$ is transformed into $(G',L)$ by applying rule RM on vertex $v^*$, then $G$ has a coloring respecting the lists $L$ if and only if $G'$ has a coloring respecting the lists $L$.

**Proof.** Suppose instance $G$ has a proper coloring respecting the lists $L$. Since $G'$ is a subgraph of $G$ we can simply use the same coloring for $G'$.

Suppose $G'$ has a proper coloring $c$ respecting the lists in $L$. For all vertices except $v^*$, we use this same coloring in $G$ to obtain a partial coloring respecting the lists. It remains to color vertex $v^*$. Since $L(v^*)$ has more colors than $v^*$ has neighbors, there is at least one color in $L(v^*)$ that is not used by any of its neighbors, pick such a color to complete the proper list coloring of $G$.

We repeatedly apply rule RM for all vertices in the graph, until it can no longer be applied. Note that this can be done in polynomial time, since checking if the rule can be applied takes polynomial time and each time it is applied the number of vertices decreases by one. After this process the remaining colors are relabeled from 1 to $m$, where $m$ is the number of remaining colors. It is clear that relabeling does not influence the answer.

The graph is planar, which implies that the number of edges is linear in the number of vertices. Therefore, $O(n \log n)$ storage is sufficient for the structure of the graph using...
an adjacency list encoding. However, we also need to take the lists of colors into account. In a planar graph, the total number of edges is bounded by $3n$, as proven in Corollary 4.2.10 in [9]. Hereby, the total degree of all vertices is bounded by $6n$. After applying reduction rule RM for every vertex, the length of the list of any remaining vertex $v$ is bounded by $d_v + 1$, such that

$$\text{Total length of all lists} \leq \sum_{v \in V} d_v + 1 \leq 7 \cdot n.$$ 

Each element of the list also needs a small amount of storage, but since there are at most $7n$ records in total, there are at most $7n$ different colors, implying that only $O(\log n)$ bits are sufficient to store a single color. Therefore, the total size of all lists is bounded by $O(n \log n)$ bits.
6 Hamiltonian Cycle

In this section we will show that Hamiltonian Cycle does not have a generalized kernel of size $O(n^{2-\varepsilon})$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$, by giving a degree-2 cross-composition. It is easy to give a linear parameter reduction from Directed Hamiltonian Cycle to Hamiltonian Cycle. Therefore, we will first prove that Directed Hamiltonian Cycle does not have a generalized kernel of sub-quadratic size, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.

<table>
<thead>
<tr>
<th>(Directed) Hamiltonian Cycle</th>
<th>Parameter: The number of vertices $n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input: A (directed) graph $G$.</td>
<td>Question: Does there exist a (directed) cycle in $G$ that visits every vertex in $V(G)$ exactly once?</td>
</tr>
</tbody>
</table>

The construction will make extensive use of the path gadget depicted in Figure 8. Such a gadget has the following property, that will be very useful for giving a cross-composition.

**Lemma 6.1.** If a directed graph $G$ contains a path gadget as an induced subgraph, such that the remainder of $G$ only connects to the path gadget at vertices $IN^0$ and $IN^1$, then any directed Hamiltonian cycle in $G$ traverses the gadget via Path 0 or Path 1, as depicted in Figure 8.

**Proof.** Any Hamiltonian cycle in $G'$ should visit the center vertex of the path gadget. Since $IN^0$ and $IN^1$ are its only two neighbors in $G'$, the only option is to visit them consecutively, Path 0 and Path 1 are the only two options to do this. □

![Figure 8. Path gadget.](image)

The following problem will be used as a starting problem for the degree-2 cross-composition.

<table>
<thead>
<tr>
<th>Hamiltonian $s-t$ Path on Bipartite Graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> An undirected bipartite graph $G$ with partite sets $A$ and $B$ such that $</td>
</tr>
<tr>
<td><strong>Question:</strong> Does $G$ have a Hamiltonian path from $b_1$ to $b_n$?</td>
</tr>
</tbody>
</table>

It is known that Hamiltonian path is NP-complete on bipartite graphs [16] and it is easy to see that it remains NP-complete when fixing a degree 1 start- and endpoint.
Theorem 6.2. (Directed) Hamiltonian Cycle parameterized by the number of vertices $n$ does not have a generalized kernel of size $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP/poly}$.

Proof. We first define an equivalence relation $\mathcal{R}$ on instances of Hamiltonian $s-t$ Path on Bipartite Graphs. Let two instances be equivalent under $\mathcal{R}$, when they have the same number of vertices. By definition, their independent sets now also have the same size. It is easy to verify that $\mathcal{R}$ is a polynomial equivalence relation.

Suppose we are given $t$ instances $X_1, \ldots, X_t$. Duplicate one of the input instances until $\sqrt{t}$ is an integer. This will multiply the number of instances by at most four and it does not affect the degree-2 cross-composition. Thereby we can relabel the given input instances as $X_{i,j}$, where $i, j \in [\sqrt{t}]$.

Each instance $X_{i,j}$ consists of a graph $G_{i,j}$ and a partition of its vertex set into independent sets $A^*_{i,j}$ and $B^*_{i,j}$. Let $|A^*_{i,j}| = m$ and $|B^*_{i,j}| = n = m + 1$ for each $i, j \in [\sqrt{t}]$. For each instance, label all elements in $A^*_{i,j}$ as $a^*_1, \ldots, a^*_m$ and all elements in $B^*_{i,j}$ as $b^*_1, \ldots, b^*_n$ such that $b^*_1$ and $b^*_n$ have degree 1.

Using the following steps, we create an instance $G'$ of Directed Hamiltonian Cycle that acts as the logical OR of the inputs.

1. First of all construct $\sqrt{t}$ groups of $m$ path gadgets each. Refer to these groups as $A_i$, for $i \in [\sqrt{t}]$, and label the gadgets within group $A_i$ as $a^i_1, \ldots, a^i_m$. Let the union of all created groups $A_i$ be named $A$.

2. Similarly, construct $\sqrt{t}$ groups of $n$ path gadgets each. Refer to these groups as $B_j$, for $j \in [\sqrt{t}]$, and label the gadgets within group $B_j$ as $b^j_1, \ldots, b^j_n$. Let $B$ be the union of all $B_j$ for $j \in [\sqrt{t}]$.

3. For every input instance $X_{i,j}$, for each edge $\{a^*_k, b^*_\ell\}$ in $G_{i,j}$ with $k \in [m], \ell \in [n]$, do the following.
   - Add an arc from $\text{IN}^0$ of $a^i_k$ to $\text{IN}^1$ of $b^j_\ell$
   - and an arc from $\text{IN}^0$ of $b^j_\ell$ to $\text{IN}^1$ of $a^i_k$.

If some $X_{i,j}$ has a Hamiltonian $s-t$ path, it can be mimicked by the combination of $A_i$ and $B_j$, where for each vertex in $X_{i,j}$ we traverse its path gadget in $G'$, following path 1. The following construction steps are needed to extend such a path to a Hamiltonian cycle in $G'$. Therefore, we create a path through each group $A_i$ for $i \in [\sqrt{t}]$, that traverses all gadgets in this group from $a^i_1$ up to $a^i_m$, using Path 0. For groups in $B$, a similar path is created.

4. Add an arc from the $\text{IN}^1$ terminal of $a^i_\ell$ to the $\text{IN}^0$ terminal of $a^i_{\ell+1}$ for all $\ell \in [m-1]$ and all $i \in [\sqrt{t}]$.

5. Similarly, add an arc from the $\text{IN}^1$ terminal of $b^j_\ell$ to the $\text{IN}^0$ terminal of $b^j_{\ell+1}$ for all $\ell \in [n-1]$ and all $i \in [\sqrt{t}]$. 

27
We create an additional structure that allows us to select exactly $\sqrt{t} - 1$ groups from $A$ and $\sqrt{t} - 1$ groups from $B$, for which all gadgets are traversed in order via Path 0. This leaves exactly one group $A_i$ and one group $B_j$ to be traversed using Path 1.

6. Add a vertex START and a vertex END and the arc $(\text{END, START})$.

7. Let $r := \sqrt{t} - 1$, add $2r$ tuples of vertices, $x_i, y_i$ for $i \in [2r]$ and connect START to $x_1$. Furthermore, add the arcs $(y_i, x_{i+1})$ for $i \in [2r - 1]$.

8. For $i \leq r$ we add arcs from $x_i$ to the IN$^0$ terminal of the gadgets $a_j^i, j \in [\sqrt{t}]$. Furthermore we add an arc from IN$^1$ of $a_m^j$ to $y_i$ for all $j \in [\sqrt{t}]$ and $i \in [r]$. When $i > r$ add arcs from $x_i$ to the IN$^0$ terminal of $b_j^i$ for $j \in [\sqrt{t}]$ and connect IN$^1$ of $b_m^j$ to $y_i$.

9. Add a vertex NEXT and the arc $(y_{2r}, \text{NEXT})$ and an arc from NEXT to the IN$^1$ terminal of all gadgets $b_j^i$ for $j \in [\sqrt{t}]$.

10. Furthermore, add arcs from IN$^0$ of all gadgets $b_j^i$ to END for $j \in [\sqrt{t}]$. So for each $B_j$ exactly one vertex has an outgoing arc to END and one has an incoming arc from NEXT.

This completes the construction of $G'$. A sketch of $G'$ is shown in Figure 9.
In Steps 4 and 5 of the construction a path through each group of gadgets in \( A \) and \( B \) is created. The following lemma shows how this path is used by a Hamiltonian cycle.

\[ \textbf{Lemma 6.3.} \text{ When any Hamiltonian cycle in } G' \text{ enters the path gadget of } a_i^1 \text{ at } IN_0, \text{ the cycle then visits the gadgets of } a_2^i, a_3^i, \ldots, a_m^i \text{ in order without visiting other vertices in between. Similarly, if any Hamiltonian cycle in } G' \text{ enters the path gadget of } b_i^1 \text{ at } IN_0, \text{ the cycle then visits the gadgets of } b_2^i, b_3^i, \ldots, b_m^i \text{ in order without visiting other vertices in between.} \]

\[ \text{Proof.} \text{ Consider a Hamiltonian cycle in } G' \text{ that enters path gadget } a_i^1 \text{ at } IN_0. \text{ By Lemma 6.1 the cycle follows Path 0 and continues to the } IN^1 \text{ terminal of the path gadget. Since that terminal has only one outgoing arc leaving the gadget, which goes to the } IN_0 \text{ terminal of } a_i^2, \text{ it follows that the cycle continues to that path gadget and enters it at } IN_0. \text{ By repeating this argument, the cycle must continue to traverse gadgets } a_i^3 \text{ up to } a_m^i \text{ using Path 0 and without visiting other vertices in between. The proof when entering group } B_j \text{ at the vertex } IN_0 \text{ of } b_i^1 \text{ is equivalent.} \]

The following lemma shows that if \( G' \) contains a directed Hamiltonian cycle, the subpath from \( x_1 \) to \( y_{2r} \) will be used to traverse exactly \( r - 1 \) groups in \( A \) and \( r - 1 \) groups in \( B \).

\[ \textbf{Lemma 6.4.} \text{ Let } C \text{ be a directed Hamiltonian cycle in } G', \text{ such that its first arc is } \{\text{START, } x_1\}. \text{ There are indices } i^*, j^* \in [\sqrt{1}] \text{ such that subpath } C_{x_1,y_{2r}} \text{ of the cycle between } x_1 \text{ and } y_{2r} \text{ contains exactly the vertices} \]

\[ \overline{A_{i^*}} \cup \overline{B_{j^*}} \cup \{x_i, y_j \mid i \in [2r]\} \]

\[ \text{where } \overline{A_{i^*}} \text{ contains all vertices of all gadgets in } A_i \text{ for } i \neq i^* \text{ and similarly } \overline{B_{j^*}} \text{ contains all vertices of all gadgets in } B_j \text{ for } j \neq j^*. \]

\[ \text{Proof.} \text{ We will first show that when the cycle reaches any } x_i \text{ for } i \in [r], \text{ it traverses exactly one group } A_{\ell} \text{ with } \ell \in [r + 1] \text{ and continues to } y_j \text{ and } x_{j+1} \text{ for some } j \in [r], \text{ without visiting other vertices in between. Similarly, when the cycle reaches any } x_i \text{ for } r < i \leq 2r, \text{ it traverses exactly one group } B_{\ell} \text{ with } \ell \in [r + 1] \text{ and continues to } y_j \text{ for some } r < j \leq 2r. \text{ For } j < 2r, \text{ the cycle then continues to } x_{j+1}, \text{ for } j = 2r \text{ the cycle reached } y_{2r}, \text{ which is the last vertex of this subpath.} \]

By Step 8 in the construction, all outgoing arcs of any \( x_i \) for \( i \in [r] \) connect to gadgets \( a_1^\ell \) for some \( \ell \in [\sqrt{r}] \). So for any \( x_i \) in the cycle there must be a unique \( \ell \in [\sqrt{r}] \) such that the edge from \( x_i \) to the \( IN^0 \) terminal of \( a_1^\ell \) is in \( C \). By Lemma 6.3 the cycle visits all vertices in \( A_{i^*} \) and no other vertices, before reaching gadget \( a_m^\ell \), which is traversed by Path 0 to get to \( OUT^0 \) of this gadget. The only neighbors of \( OUT^0 \) of gadget \( a_m^\ell \) lying outside this gadget are of type \( y_j \) for \( j \in [r] \). As such, the cycle must visit some \( y_j \) next, and its only outgoing arc goes to \( x_{j+1} \).

The proof for \( i > r \) is similar. As such, visiting \( x_i \) for \( i \in [r] \) results in visiting all vertices of exactly one group in \( A \) before continuing via \( y_j \) to some \( x_{j+1} \) without visiting any vertices in between. Visiting \( x_i \) for \( r < i \leq 2r \) results in visiting all vertices of exactly
one group in $B$ and returning via $y_j$ to either the end of the subpath ($j = 2r$) or some $x_{j+1}$.

Every vertex $x_i$ for $i \in [2r]$ must be visited by $C$, it remains to show that it is visited in subpath $C_{x_1,y_{2r}}$. Suppose there exists an $x_i$ for $i \in [2r]$ such that $x_i$ is not visited in the subpath from $x_1$ to $y_{2r}$. As we have seen above, visiting some $x_i$ results in visiting all vertices in some group in $A$ or $B$, continued by visiting some $y_j$ for $j \in [2r]$. Note that no other vertices are visited in between. Hereby, $y_j$ is not in subpath $C_{x_1,y_{2r}}$. This implies $j \neq 2r$ and thus the next vertex in the cycle is $x_{j+1}$. So, for $x_i$ not in subpath $C_{x_1,y_{2r}}$, one can find a new vertex $x_{j+1}$ (where $j + 1 \neq i$), such that $x_{j+1}$ is also not in subpath $C_{x_1,y_{2r}}$. Note that we can not create a loop, by visiting a vertex $x_i$ seen earlier, as this would not yield a Hamiltonian cycle in $G'$. For example, the vertex $\text{START}$ would never be visited. This is however a contradiction since we only have finitely many vertices $x_i$.

Thus in subpath $C_{x_1,y_{2r}}$, exactly $r$ groups of $A$ are visited and exactly $r$ groups of $B$ are visited, and no other vertices than specified. This leaves exactly one group $A_i^r$ and one group $B_j^r$ unvisited in $C_{x_1,y_{2r}}$. $\square$

In the lemma above, one group in $A$ and one group in $B$ remain unvisited. The next lemma shows that these groups are visited in the subpath between $\text{NEXT}$ and $\text{END}$. This result will be useful to show that one of the input instances must have a Hamiltonian path if $G'$ has a Hamiltonian cycle.

$\triangleright$ Lemma 6.5. Let $C$ be a Hamiltonian cycle in $G'$, such that its first arc is $\{\text{START},x_1\}$. Let $i^*$ and $j^*$ satisfy the conditions of Lemma 6.4. Then cycle $C$ visits $b_1^i$ before $b_n^j$. Moreover, the subpath of the cycle $C_{b_1^i,b_n^j}$ between terminal $\text{IN}^1$ of $b_1^i$ and $\text{OUT}^1$ of $b_n^j$ (inclusive) contains all vertices of the gadgets in $A_{i^*}$ and $B_{j^*}$, and no others.

Proof. Vertex $\text{NEXT}$ is visited directly after $y_{2r}$. The arc from $\text{NEXT}$ to gadget $b_1^i$ must be in the cycle for some $\ell \in [\sqrt{t}]$, since these are the only outgoing edges of $\text{NEXT}$, constructed in Step 9. By Lemma 6.4, all gadgets in all $B_j$ for $j \neq j^*$ are visited in the path from $x_1$ to $y_{2r}$, and thus should not be visited after vertex $\text{NEXT}$. Therefore, the edge from $\text{NEXT}$ to gadget $b_1^i$ is in the cycle, which also implies that $b_1^i$ is visited before $b_n^j$.

It is easy to see that $\{\text{END},\text{START}\}$ is the last arc in $C$. By considering the incoming arcs of $\text{END}$ it follows that some arc from terminal $\text{OUT}^1$ of $b_n^j$ to $\text{END}$ for $\ell \in [\sqrt{t}]$ is in the cycle. Since the vertices in gadgets $b_\ell^j$ for $\ell \neq j^*$ are already visited in $C_{x_1,y_{2r}}$ by Lemma 6.4, it follows that $\{b_n^j,\text{END}\}$ is in $C$.

By Lemma 6.4, none of the terminals of gadgets in $A_{i^*}$ and $B_{j^*}$ is visited in the subpath $C_{x_1,y_{2r}}$, or equivalently in the subpath $C_{\text{START},\text{NEXT}}$. Since $C$ is a Hamiltonian cycle these vertices must therefore be visited in $C_{\text{NEXT},\text{START}}$, which is equivalent to saying that $C_{b_1^i,b_n^j}$ must contain all vertices in $A_{i^*} \cup B_{j^*}$. It is easy to see that this subpath cannot contain any other vertices, as all other vertices are present in $C_{\text{START},\text{NEXT}}$ or $C_{\text{END},\text{START}}$. $\square$

30
Using the previous lemmas we can now show that \( G' \) acts as a logical OR of the given inputs.

\[ \textbf{Claim 6.6.} \text{Graph } G' \text{ has a directed Hamiltonian cycle if and only if at least one of the instances } X_{i,j} \text{ has a Hamiltonian } s-t \text{-path.} \]

\textbf{Proof.} (\( \Leftarrow \)) Suppose \( G' \) has a Hamiltonian cycle \( C \). By Lemma 6.5 there exist \( i^* \), \( j^* \in [\sqrt{t}] \) such that the subpath of \( C \) from gadget \( b_1^{i^*} \) to \( b_n^{j^*} \) visits exactly the gadgets in \( A_i \cup B_j \). Since gadget \( b_1^{i^*} \) is entered at terminal \( \text{IN}^1 \), it is easy to see that all gadgets are traversed using Path 1. We now construct a Hamiltonian path \( P \) for instance \( X_{i^*,j^*} \).

Let \( \{a_k^i(i^*j^*), b_j^i(i^*j^*)\} \in P \) if the arc from \( \text{OUT}^1 \) of \( a_k^i \) to \( \text{IN}^1 \) of \( b_j^i \) is in \( C \). Similarly let \( \{b_k^j(i^*j^*), a_i^j(i^*j^*)\} \in P \) if the arc from \( \text{OUT}^1 \) of \( b_k^j \) to \( \text{IN}^1 \) of \( a_i^j \) is in \( C \), where \( k \in [m] \) and \( \ell \in [n] \). Using that every gadget is visited exactly once via Path 1 in \( C \), we see that \( C \) is a Hamiltonian path.

(\( \Rightarrow \)) Suppose \( X_{i,j} \) has a Hamiltonian \( s-t \) path \( P \). Then we create a Hamiltonian cycle \( C \), for each vertex \( a_k^i \) from instance \( X_{i,j} \) in \( P \) we add Path 1 in path gadget \( a_k^i \) to \( C \) and for each vertex \( b_j^i \) we add Path 1 in path gadget \( b_j^i \) to \( C \). Let \( P \) be ordered such that \( b_1^j \) is its first vertex. Now if \( a_k^i \) is followed by \( b_j^i \) in \( P \), the arc from terminal \( \text{OUT}^1 \) of \( a_k^i \) to \( \text{IN}^1 \) of \( b_j^i \) is added to \( C \). Similarly, if a vertex \( b_j^i \) is followed by \( a_k^i \) in \( P \), the arc from terminal \( \text{OUT}^1 \) of \( b_j^i \) to \( \text{IN}^1 \) of \( a_k^i \) will be added to \( C \). Now the subpath \( C_{b_1^j,b_1^j} \) contains all terminals in all gadgets in \( A_i \cup B_j \).

From \( b_n^{j^*} \) the cycle goes to \( \text{END} \), then to \( \text{START} \) and to \( x_1 \). To visit all groups \( A_i \) for \( i \neq i^* \) and \( B_j \) for \( j \neq j^* \), do the following.

- From \( x_i \) where \( i \leq i^* \), the cycle continues to gadgets \( a_i^1 \), then to \( a_i^2, a_i^3, \ldots, a_i^{m-1} \) following Path 0, and continue to \( y_i, x_{i+1} \).
- From \( x_i \) where \( i^* \leq i \leq r \) it goes to \( a_i^1, a_i^2, \ldots, a_i^{i+1} \) and continues with \( y_i, x_{i+1} \).
- Similarly, from \( x_i \) where \( r \leq i < j^* \), go through gadgets \( b_1^j, \ldots, b_n^j \) and continue to \( y_i, x_{i+1} \).
- From \( x_i \) where \( j^* \leq i \leq 2r \), go to gadgets \( b_1^{i+1}, \ldots, b_n^{i+1} \) and continue to \( y_i \), for \( i \neq 2r \) then add the arc \((y_i, x_{i+1})\).

From \( y_{2r} \), continue to \( \text{NEXT} \), after which the edge \((\text{NEXT}, b_1^{i^*})\) closes the cycle. By definition, no vertex is visited twice, so it remains to check that every vertex of \( G' \) is in the cycle. For vertices \( \text{START}, \text{NEXT}, \text{END} \) and all vertices \( x_i, y_i, z_i \) this is obvious. All vertices in \( A_i \) and \( B_j \) where \( i \neq i^* \) and \( j \neq j^* \) are in the cycle between some \( x_i \) and \( y_i \). All vertices in \( A_i \) and \( B_j \) are visited since \( P \) was a Hamiltonian path on these vertices. \( \Box \)

The number of vertices of \( G' \) is

\[
\frac{3(m+n)\sqrt{t} + 4(\sqrt{t} - 1) + 3}{|A \cup B|} = O(\sqrt{t} \cdot (m+n))
\]
therefore this is a degree-2 cross composition. It is possible to give a linear parameter transformation from \textsc{Directed Hamiltonian Cycle} to \textsc{Hamiltonian Cycle}, using the NP-completeness proof by Karp in [19]. Suppose we are given $G = (V, A)$ for directed Hamiltonian cycle, we construct $G' = (V', E')$ by $V' := \{v_{in}, v_{mid}, v_{out} \mid v \in V\}$ and $E' := \{\{v_{in}, v_{mid}\}, \{v_{mid}, v_{out}\}\} \cup \{\{u_{out}, v_{in}\} \mid (u, v) \in A\}$ as shown in Figure 10. It is easy to verify that this is a linear parameter transformation. Using Theorems 2.6 and 2.8 we conclude that \textsc{(Directed) Hamiltonian Cycle} does not have a generalized kernel of size $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$. \hfill \blacksquare
7 Dominating Set

In this section, we will show that DOMINATING SET and CONNECTED DOMINATING SET do not have a kernel of sub-quadratic size, when parameterized by the number of vertices.

<table>
<thead>
<tr>
<th><strong>DOMINATING SET</strong></th>
<th><strong>Parameter:</strong> The number of vertices $n$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> An undirected graph $G = (V, E)$ and integer $k \in \mathbb{N}$</td>
<td></td>
</tr>
<tr>
<td><strong>Question:</strong> Does there exist a set $S \subseteq V(G)$ with $</td>
<td>S</td>
</tr>
</tbody>
</table>

We will also consider the CONNECTED DOMINATING SET problem which is defined similarly except that there is an extra constraint that $G[S]$ is connected.

A set $S$ with the described properties is called a (connected) dominating set in $G$. We say that a vertex $u$ is dominated by $v$ (with respect to dominating set $S$) if $u$ is a neighbor of $v$ and $v$ is contained in the dominating set.

We will prove the sparsification lower bound using a degree-2 cross-composition, starting from a variation of the COLORED RED-BLUE DOMINATING SET problem (COL-RBDS) as described by Dom et al. in [10].

<table>
<thead>
<tr>
<th><strong>EQUAL-SIZED COLORED RED/BLUE DOMINATING SET (EQ-COL-RBDS)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A bipartite graph $G = (R \cup B, E)$, where $R$ is partitioned into $k$ subsets $R_1, \ldots, R_k$, such that $</td>
</tr>
<tr>
<td><strong>Question:</strong> Is there a set $S \subseteq R$ such that for each $i \in [k]$ the set $S$ contains exactly one vertex of $R_i$, and every vertex in $B$ is adjacent to at least one vertex from $S$?</td>
</tr>
</tbody>
</table>

We will think of the vertices in set $R_i$ as having color $i$ such that the question is if there exists a set $S \subseteq R$ that has exactly one vertex of each color and every vertex in $B$ is adjacent to at least one vertex in $S$.

Lemma 7.1. EQ-COL-RBDS is NP-complete.

Proof. It is easy to give a reduction from COLORED RBDS, without requiring that all color sets have the same size. This problem was proven to be NP-complete by Dom et al. in [10]. Let an input instance on $n$ vertices for Col-RBDS be given, let the largest color set have size $\ell$. We then add isolated vertices to all color other sets until their size is exactly $\ell$. Since the added vertices are isolated, they do not influence the size of a minimum RBDS in the graph. Note that both the number of color sets and their size are bounded by $n$, as such we have created a graph with at most $n^2$ vertices, which is polynomial. ▶

Using EQ-COL-RBDS as our starting problem, we can give a degree-2 cross-composition.
A sketch of $G$, where $t' = 2$, $m = 6$ and $n = 5$. Thereby $K$ should be 5 and $W_{(c_1,c_2)}$ should contain 10 vertices. In this example we show the constructed graph when choosing $K = 1$ for simplicity. Edges from $R$ to $B$ are left out for simplicity.

**Theorem 7.2.** DOMINATING SET and CONNECTED DOMINATING SET parameterized by the number of vertices $n$ do not have a generalized kernel of size $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.

**Proof.** We can use the same degree-2 cross-composition to prove both results.

Define a polynomial equivalence relation $R$ by first of all letting all instances where there is a vertex in $B$ of degree 0 be in the same class, note that these are always no-instances. Let 2 instances $(G = (R \cup B), k)$ and $(G' = (R' \cup B'), k')$ of EQ-COL-RBDS be equivalent if $|R| = |R'|$, $|B| = |B'|$ and $k = k'$. It is easy to see that $R$ indeed is a polynomial equivalence relation.

Suppose we are given $t$ instances of EQ-COL-RBDS, such that $\sqrt{t}$ and $\log \sqrt{t} \in \mathbb{N}$ and such that all given instances are in the same equivalence class of $R$. Let $t' := \sqrt{t}$. If these instances are from the class where $B$ contains a vertex of degree 0, output a constant size no-instance.

Otherwise, label the given instances as $X_{i,j}$ with $i, j \in [t']$. Let instance $X_{i,j}$ have graph $G_{i,j}$, which is bipartite with vertex set $R_{i,j} \cup B_{i,j}$. Let $|R_{i,j}| = m$ and $|B_{i,j}| = n$ and let $R'_{i,j}$ be partitioned into $k$ color classes $R'_{i,j}^p$ for all $i,j \in [t']$ and $p \in [k]$. Label all vertices in $R_{i,j}^p$ as $r_{i,j}^p(i,j)$ with $p \in [k]$ and $q \in [m/k]$, which means that this vertex is the $q$’th vertex of color $p$ from instance $X_{i,j}$. Label vertices in $B_{i,j}$ as $b_{i,j}^0(i,j), \ldots, b_{i,j}^{\sqrt{t}}(i,j)$ arbitrarily. We now create an instance $(G, k)$ for DOMINATING SET using the following steps. A sketch of $G$ can be found in Figure 11.

1. Add vertices $r_{i,j}^p$ for $p \in [k], q \in [m/k]$ and $i \in [t']$. The dominating set problem does not use colored instances, however we will remember the color of these
vertices for simplicity. Let vertex \( r^i_{p,q} \) have color \( p \), for \( i \in [t'] \), \( q \in [m/k] \) and \( p \in [k] \). Define \( R_i := \{ r^i_{p,q} \mid p \in [k], q \in [m/k] \} \) and let \( R := \bigcup_{i \in [t']} R_i \). Give every set \( R_i \) a unique identifier \( ID(R_i) \), which is a subset of \( K := 2 + k + \log t' \) numbers in the range \([2K]\). We can construct this identifier by considering the binary representation \( b \) of \( i - 1 \), which has length \( \log t \). Consider the string \( \bar{b} \bar{b} \), where \( \bar{b} \) contains a zero at a certain position if and only if \( b \) contains a one in this position. String \( \bar{b} \bar{b} \) has exactly \( \log t \) positions with a 1. Now if the \( j \)th position contains a 1, let \( j \in ID(R_i) \). This already ensures that every \( R_i \) gets a unique ID containing \( \log t \) integers in the range \([2\log t']\). Then add \( k + 2 \) distinct integers from the range \( 2\log t + 1, \ldots, 2\log t + 2k + 4 \) to the ID.

2. Add vertices \( b^j_\ell \) for \( \ell \in [n] \) and \( j \in [t'] \). Define \( B_j \) and \( B \) as \( B_j := \{ b^j_\ell \mid \ell \in [n] \} \) and \( B := \bigcup_{j \in [t']} B_j \).

3. Add edges between the vertices \( r^i_{p,q} \) and \( b^j_\ell \) for \( p \in [k], q \in [m/k] \) and \( i,j \in [t'] \) if \( r^i_{p,q}(i,j) \) is connected to \( b^j_\ell(i,j) \) in instance \( X_{i,j} \). This ensures that the graph induced by \( R_i \cup B_j \) is exactly \( G_{i,j} \) and the coloring of vertices in \( R_i \) matches the coloring of \( R_{i,j}^* \).

4. Add vertices \( s' \) and \( s \) and edge \( \{s',s\} \). Furthermore, add edges between \( s \) and all vertices in \( R \). The degree-1 vertex \( s' \) ensures there is a minimum dominating set containing \( s \), which covers all vertices in \( R \) “for free”.

5. In a similar way as given by Dom et al. in [10], for every pair of colors \( (c_1, c_2) \in \{1,\ldots,k\} \times \{1,\ldots,k\} \) with \( c_1 \neq c_2 \) we add a vertex set

\[
W^{(c_1,c_2)} = \{ w_1^{(c_1,c_2)}, \ldots, w_{2k}^{(c_1,c_2)} \}.
\]

For \( x \in [2K] \) connect \( w_x^{(c_1,c_2)} \) to all vertices of color \( c_1 \) in \( R_i \) if \( x \in ID(R_i) \), otherwise connect \( w_x^{(c_1,c_2)} \) to all vertices of color \( c_2 \) in \( R_i \). This construction is used to choose which \( R_i \) is part of a solvable input instance \( X_{i,j} \) for some \( j \in [t'] \). This idea is formalized in Lemmas 7.5 and 7.6.

6. Then, add \( \log t' \) triangles, with vertices \( \{ t^0_\ell, t^1_\ell, t^2_\ell \} \) for \( \ell \in [\log t'] \). Connect \( t^0_\ell \) to all vertices in \( B \), if the \( \ell \)th bit of \( j \) equals 0, connect \( t^1_\ell \) to all vertices in \( B \) if the \( \ell \)th bit of \( j \) equals 1. Define \( T \) to be the union of all these triangles. By choosing exactly one of the vertices \( t^0_\ell \) or \( t^1_\ell \) in a dominating set for each \( \ell \), all groups \( B_j \) except one are dominated automatically. The non-dominated one should then be part of a solvable input instance.

7. Finally, add the edges \( \{\{s,t_\ell\} \mid \ell \in [\log t'], i \in \{0,1\} \} \). Note that \( t^*_\ell \) is not a neighbor of \( s \) and that \( t^0_\ell \) and \( t^1_\ell \) are its only neighbors. This final step of the construction ensures that every vertex in \( T \) contained in the dominating set will
have \( s \) as a neighbor in the dominating set, such that there is always a minimum dominating set that is connected. This result will be formalized in Claim 7.7.

We now make the following observations.

\( \blacktriangleright \textbf{Lemma 7.3.} \) If \( G \) has a dominating set \( D \), then it also has a dominating set \( D' \) of size at most \(|D|\) that does not contain any vertices from \( B \).

\textit{Proof.} Suppose we are given a minimum dominating set \( D \) of \( G \), where vertex \( v \in B \) is present. In any dominating set, \( s \) or \( s' \) must be present. If \( s' \) is present and \( s \) is not, we replace \( s' \) by vertex \( s \), and still obtain a valid dominating set of the same size. As such, all vertices in \( R \) are now dominated by \( s \). Vertices \( t^0_\ell \) and \( t^1_\ell \) with \( \ell \in \log t' \) are dominated by \( s \). Since \( t^2_\ell \) only has neighbors \( t^1_\ell \) and \( t^0_\ell \), at least one of these three vertices is present in \( D \) for every \( \ell \in \log t' \). Hereby every vertex in \( T \) has a neighbor in \( D \).

Since \( B \) is an independent set in \( G \), the vertex \( v \) does not dominate other vertices in \( B \). Since the polynomial equivalence relation ensures that there are no isolated vertices in \( B \), vertex \( v \) has at least one neighbor \( u \) in \( R \). We can safely replace \( v \) by \( u \) to obtain a valid dominating set that has the same size as \( D \) and does not contain any vertices from \( B \). \( \blacktriangleleft \)

\( \blacktriangleright \textbf{Lemma 7.4.} \) Any dominating set of \( G \) of size at most \( k + 1 + \log t' \) contains at least \( 1 + \log t' \) vertices from \( \{ s, s' \} \cup \{ t^0_\ell, t^1_\ell, t^2_\ell \mid \ell \in \log t' \} \) and thus contains at most \( k \) vertices from \( R \).

\textit{Proof.} In a dominating set \( D \) of \( G \), at least \( \log t' \) vertices are needed from \( T \), since \( t^2_\ell \) only has neighbors \( t^1_\ell \) and \( t^0_\ell \), so one of these three vertices must be in \( D \) for each \( \ell \in \log t' \). Furthermore at least one of the vertices \( s' \) or \( s \) must be present, therefore there are \( 1 + \log t' \) vertices in the set that are not from \( R \). \( \blacktriangleleft \)

We obtained that a dominating set of size at most \( k + 1 + \log t' \) contains at most \( k \) vertices from \( R \). This result can now be used to prove that such a dominating set contains exactly one vertex of each color.

\( \blacktriangleright \textbf{Lemma 7.5.} \) Any dominating set of \( G \) of size at most \( k + 1 + \log t' \) uses exactly one vertex of each color from \( R \).

\textit{Proof.} Suppose a dominating set of \( G \) of size at most \( k + 1 + \log t' \) uses less than \( k \) colors from \( R \). If at most \( k - 2 \) colors are used, there must be two colors \( c_1 \) and \( c_2 \) that are not present in the set. However, this implies that all \( 2K \) vertices in \( W^{(c_1,c_2)} \) are not dominated by vertices in \( R \) and must therefore be in the set. This contradicts the maximum size of the dominating set, since \( 2K = 2k + 4 + 2 \log t' \). So, we are left with the possibility of using \( k - 1 \) colors. Consider some color \( c_1 \) that was not used. Look at another color \( c_2 \) that is used exactly once, such a color exists by Lemma 7.4. Suppose the vertex of color \( c_2 \) in the dominating set was from set \( R_i \) for some \( i \in [t'] \). Then for any \( x \in ID(R_i) \) we have that \( w^{(c_1,c_2)}_x \) is not connected to any vertex in the dominating set and therefore must be in

36
the dominating set itself. Since $ID(R_i)$ contains $k$ numbers, there are $K > k + 1 + \log t'$ vertices in $W$ that are not dominated by $R$, which contradicts the maximum size of the dominating set.

The lemma above also implies that there are exactly $k$ vertices from $R$ in any dominating set of $G$ that has size at most $k + 1 + \log t'$. It then follows that none of the vertices in $W$ are in the dominating set. We can now show that the vertices chosen from $R$ are all contained in the same set $R_i$ for some $i$.

\textbf{Claim 7.7.} For any dominating set $D$ of $G$ of size at most $k + 1 + \log t'$, there exists $i \in [t']$ such that all vertices in $D \cap R$ are contained in set $R_i$.

\textbf{Proof.} Suppose there exists two vertices $u, v \in D$ such that $u \in R_i$ and $v \in R_j$ for some $i \neq j$. By Lemma 7.5, $u$ and $v$ have different colors. Suppose $u$ has color $c_u$ and $v$ has color $c_v$. Since $R_i \neq R_j$, there exists $x \in [2K]$ such that $x \in ID(R_i)$ and $x \notin ID(R_j)$. By Step 5 of the construction, this means that none of the neighbors of vertex $w_x^{(c_u,c_v)}$ are contained in the dominating set. However, this vertex is not in $D$ and therefore $D$ is not a dominating set of $G$, which is a contradiction.

Using the previous Lemmas, we can prove that $G'$ acts as a logical OR of the given input instances, for both \textsc{Dominating Set} and \textsc{Connected Dominating Set}.

\textbf{Lemma 7.6.} For any dominating set $D$ of $G$ of size at most $k + 1 + \log t'$, there exists $i \in [t']$ such that all vertices in $D \cap R$ are contained in set $R_i$.

\textbf{Proof.} Suppose there exists two vertices $u, v \in D$ such that $u \in R_i$ and $v \in R_j$ for some $i \neq j$. By Lemma 7.5, $u$ and $v$ have different colors. Suppose $u$ has color $c_u$ and $v$ has color $c_v$. Since $R_i \neq R_j$, there exists $x \in [2K]$ such that $x \in ID(R_i)$ and $x \notin ID(R_j)$. By Step 5 of the construction, this means that none of the neighbors of vertex $w_x^{(c_u,c_v)}$ are contained in the dominating set. However, this vertex is not in $D$ and therefore $D$ is not a dominating set of $G$, which is a contradiction.

Using the previous Lemmas, we can prove that $G'$ acts as a logical OR of the given input instances, for both \textsc{Dominating Set} and \textsc{Connected Dominating Set}.

\textbf{Claim 7.7.}

1. If there is an input $X_{i',j'}$ that has a col-RBDS of size $k$, then $G'$ has a connected dominating set of size $k + 1 + \log t'$.

2. If $G'$ has a (not necessarily connected) dominating set of size $k + 1 + \log t'$, then some input $X_{i',j'}$ has a col-RBDS of size $k$.

\textbf{Proof.} 1. Let $X_{i',j'}$ have a colored RBDS $D$ of size at most $k$, then we can construct a dominating set $D'$ of $G$ in the following way. For any vertex $r_{p,q}^*$ in $D$, add vertex $r_{p,q}^*$ to $D'$.

Furthermore add the vertex $s$ to $D'$. Then add vertex $t_{q}^0$ to $D'$ if the $q$'th bit of $j^*$ is 1, add vertex $t_{q}^1$ otherwise. Now $s'$ is dominated and all vertices in $R$ have neighbor $s$ in $D'$. All vertices in $B_{j'}$ are covered by the vertices in the dominating set from $R_{i'}$, since $D$ was a col-RBDS of $X_{i',j'}$. All vertices in $B_j$ for $j \neq j'$ have neighbor $t_{q}^0$ or $t_{q}^1$ in $D'$ for some $\ell \in \lfloor \log t' \rfloor$, since the bit representation of $j$ must differ from the one of $j^*$ at some position. It now follows from Step 6 of the construction that all vertices in $B_j$ are connected to a vertex in the dominating set.

It remains to verify that all vertices in $W$ have a neighbor in $D'$. Consider $w_{x}^{(c_1,c_2)}$ for $x \in [2K]$ and $c_1, c_2 \in [k]$. If $x \in ID(R_{i'})$, then this vertex is connected to all vertices of color $c_1$ and exactly one of them is contained in $D'$. If $x \notin ID(R_{i'})$, the
vertex $w^{(i)}_2$ is connected to all vertices of color $c_2$ in $R_i$, and again one vertex of this color in $R_i$ is contained in $D'$. So $D'$ is a dominating set of $G$ and it is easy to verify that $|D'| = k + 1 + \log t'$. Furthermore, $D'$ is constructed in such a way that it is connected. We can show this by proving that every vertex in $D'$ is a neighbor of $s$, since we chose $s$ in $D$. Vertices in $D' \cap R$ and $D' \cap T$ are neighbors of $s$, by Steps 4 and 7 of the construction of $G$. The vertex $s'$ and vertices from $W$ and $B$ are not contained in $D'$. Thus, $D'$ is a connected dominating set.

2. Let $D'$ be a dominating set of $G$ of size at most $k + 1 + \log t'$. Using Lemma 7.3 we modify $D'$ such that it chooses no vertices from $B$, without increasing its size. By Lemma 7.5 and 7.6, $D'$ contains exactly $k$ vertices from $R$, all from the same $R_i$, for some $i^*$ and all of different color. $D'$ has size at most $k + 1 + \log t$ of which $k$ are contained in $R$ and one in $\{s, s'\}$. Combined with the fact that for any $\ell \in [\log t']$ vertex $t_j^\ell$ has $t_1^\ell$ and $t_0^\ell$ as its only two neighbors, it follows that exactly one of these three vertices is contained in $D'$ for all $\ell$. Therefore $D'$ contains at most one of the vertices $t_0^\ell$ or $t_1^\ell$ for every $\ell \in [\log t']$.

We can now define $x_j \in \{0, 1\}$ for $\ell \in [\log t']$, such that $t_j^\ell \notin D'$ for all $\ell \in [\log t']$. Consider the index $j^* \in [t]$ given by the binary representation $[x_1 \; x_2 \ldots \; x_{\log t'}]$. It follows from the bit representation of $j^*$ that the vertices in $B_{j^*}$ are not connected to any of the vertices in $D' \cap T$. Since vertices in $R_i$ are only adjacent to vertices in $R$ and vertices of $T$, it follows that every vertex in $B_{j^*}$ has a neighbor in $R$ that is in $D'$. This implies that every vertex in $B_{j^*}$ has a neighbor in $D' \cap R_i$. Since $G[R_i \cup B_{j^*}]$ is isomorphic to the graph of instance $X_{i,j^*}$, it follows that $X_{i,j^*}$ has a col-RBDS of size at most $k$, which are exactly the vertices in $D' \cap R_i$.

Given $t$ instances, the graph $G$ constructed above has

$$\frac{n \cdot t'}{|K|} + \frac{n' \cdot t'}{|B|} + \frac{2}{|T|} + 2 \cdot \log t' + 2 \cdot \left(\frac{k}{2}\right) \cdot 2K = \mathcal{O}(\sqrt{t} \max |X_{i,j}|^3)$$

vertices. It is straightforward to construct $G$ in polynomial time. It follows from Claim 7.7 that $G$ has a dominating set of size $k + 1 + \log t'$, if and only if one of the input instances has a col-RBDS of size $k$. Furthermore, $G$ has a connected dominating set of size $k + 1 + \log t'$ if and only if one of the input instances has a col-RBDS of size $k$. Therefore we have given a degree-2-cross-composition to (CONNECTED) DOMINATING SET. Using Theorem 2.6 it follows that DOMINATING SET and CONNECTED DOMINATING SET do not have a generalized kernel of size $\mathcal{O}(n^{2-\epsilon})$ for any $\epsilon > 0$, unless $\text{NP} \subseteq \text{coNP} / \text{poly}$. ▲

This lower bound has immediate implications for the sparsification lower bounds of the dual problems NONBLOCKER and MAX LEAF SPANNING TREE. It is hard to give an intuition about NONBLOCKER, other than that a graph has a NONBLOCKER of size at least $k$ if and only if it has a DOMINATING SET of size at most $n - k$. MAXIMUM LEAF
Spanning Tree seems to be a more natural problem, where we ask whether a given graph contains a spanning tree with at least $k$ leaves. Using Theorem 7.2 we obtain the following result.

► Corollary 7.8. **Max Leaf Spanning Tree and Nonblocker** parameterized by the number of vertices $n$ do not have a generalized kernel of size $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.

**Proof.** By definition, a graph has a Nonblocker of size $k$ if and only if it has a Dominating Set of size $n - k$. Thereby it is trivial to give a linear parameter transformation from **Dominating Set** to **Nonblocker** and use Theorems 7.2 and 2.8 to obtain the lower bound.

If a graph has a spanning tree with at most $k$ leaves, the non-leaves of this tree correspond to connected dominating set of $G$, that has size $n - k$. Vice versa, given a connected dominating set of size $n - k$, we extend it to a spanning tree with $k$ leaves in a straightforward way. Thus, a graph has a connected dominating set of size $n - k$ if and only if it has a spanning tree with $k$ leaves. Thereby it is trivial to give a linear parameter transformation from **Connected Dominating Set** to **Max Leaf Spanning Tree** and use Theorems 7.2 and 2.8 to obtain the lower bound. ▶

Both **Dominating Set** and **Connected Dominating Set** are not believed to be fixed parameter tractable when parameterized by the solution size ([12], [6]), so no kernels are known. **Nonblocker** and **Maximum Leaf Spanning Tree** are however fixed parameter tractable and both have a linear size kernel. **Nonblocker** has a kernel of $\frac{5}{2}k + 3$ vertices [7] and **Maximum Leaf Spanning Tree** has a kernel of $3.75k$ vertices [14]. Since for these problems input parameter $k$ is bounded by $n$, Theorem 7.2 shows that we cannot reduce the number of edges of these kernels to $O(k^{2-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.

39
8 \textit{d-Hypergraph 2-Colorability and d-NAE-Sat}

In this section we will give a kernelization lower bound of \textit{d-Hypergraph 2-Colorability}, by giving a linear parameter reduction starting from \textit{d-NAE-Sat}. Furthermore we will show that both this problem and \textit{d-NAE-Sat} have a (generalized) kernel consisting of \(O(n^{d-1})\) edges or clauses.

### \textit{d-Hypergraph 2-Colorability}

**Parameter:** The number of vertices \(n\)

**Input:** A hypergraph \(G = (V, E)\), where each hyperedge contains at most \(d\) vertices.

**Question:** Is there a coloring using \{Red, Blue\} of all vertices in \(V\) such that every edge in \(E\) contains at least one Red and one Blue vertex.

### \textit{d-Not-All-Equal Sat}

**Parameter:** The number of variables \(n\)

**Input:** A CNF formula \(F\) on \(n\) variables with at most \(d\) literals per clause.

**Question:** Is there an assignment for all variables, such that each clause contains at least one true and at least one false literal? If such an assignment exists, we say \(F\) if NAE-satisfiable.

#### 8.1 Lower bound

We can show a lower bound on the kernel size of \textit{d-NAE-Sat}, which we will later use to prove a lower bound for the kernel of \textit{d-Hypergraph 2-Colorability}. The following result was also shown by Jansen \textit{et al.} in [18].

\textbf{Lemma 8.1.} Let \(d \geq 4\) be an integer. Then \textit{d-NAE-Sat} parameterized by the number of variables \(n\) does not have a generalized kernel of size \(O(n^{d-1-\varepsilon})\) for any \(\varepsilon > 0\), unless \(\text{NP} \subseteq \text{coNP}/\text{poly}\).

\textit{Proof.} It is easy to give a linear parameter reduction from \textit{d-CNF-Sat} to \((d + 1)-\text{NAE-Sat}\). By Theorem 2.8 it follows that if \((d + 1)-\text{NAE-Sat}\) has a generalized kernel of size \(O(n^{d-\varepsilon})\), then also \textit{d-CNF-Sat} has a generalized kernel of size \(O(n^{d-\varepsilon})\). In Theorem 1 in [8] Dell \textit{et al.} prove an even stronger statement than the result we need here. Their theorem implies is that \textit{d-CNF-Sat} does not have a generalized kernel of size \(O(n^{d-\varepsilon})\)

So suppose we are given a Boolean formula \(F\) of \textit{d-CNF-Sat}, with variables \(X = x_1, \ldots, x_n\) and clauses \(C_1, \ldots, C_m\). We will now create an instance \(F'\) of \((d + 1)-\text{NAE-Sat}\) with variables \(X' := \{x_1, \ldots, x_n\} \cup y\) and clauses \(C_1', \ldots, C_m'\), where \(C_i' := C_i \cup y\) for \(1 \leq i \leq m\). It follows that every clause \(C_i'\) has size at most \(d + 1\). It remains to verify that \(F'\) is NAE-satisfiable if and only if \(F\) is satisfiable.

Suppose \(F\) is satisfiable, we can extend a satisfying truth \(S\) assignment with \(y := \text{false}\) to NAE-satisfy \(F'\). Every clause now contains one \text{false} literal by definition, and one \text{true} literal since \(S\) was a satisfying truth assignment for \(F\).
Suppose \( F' \) is NAE-satisfied with assignment \( S \), we can then do a case distinction on the value of \( y \). If \( y = \text{false} \) every clause in must \( F' \) contain a \textit{true} literal not equal to \( y \). Therefore, we can use the same truth assignment to satisfy \( F \). If \( y = \text{true} \) then every clause in \( F' \) contains at least one \textit{false} literal not equal to \( y \). Therefore, if we invert the truth assignment by defining \( x := \text{true} \) if \( x \) was \textit{false} in \( S \), then every clause in \( F \) contains at least one \textit{true} literal, thus \( F \) is satisfiable.

\[ \blacktriangleright \textbf{Theorem 8.2.} \ Let \( d \geq 4 \) be an integer. There is a linear parameter transformation from \( d\text{-NAE-SAT} \) to \( d\text{-HYPERGRAPH 2-COLORABILITY} \). It follows that \( d\text{-HYPERGRAPH 2-COLORABILITY} \) does not have a kernel of size \( O(n^{d-1-\epsilon}) \) for any \( \epsilon > 0 \), unless \( \text{NP} \subseteq \text{coNP/poly} \).

\textbf{Proof.} \ Here we will use the result from Lemma 8.1 and give a linear parameter transformation from \( d\text{-NAE-SAT} \) to \( d\text{-HYPERGRAPH 2-COLORABILITY} \). Suppose we are given a Boolean formula \( F \) in CNF-form for \( d\text{-NAE-SAT} \), with variables \( X = x_1, \ldots, x_n \) and clauses \( C_1, \ldots, C_m \). For hypergraph 2-colorability, we now create the set of vertices \( V = \{y_1, \ldots, y_n\} \cup \{z_1, \ldots, z_n\} \). We create hyperedges \( e_i \) for \( i \in [m] \) such that \( y_i \in e_i \) if \( x_i \in C_i \) and \( z_i \in e_i \) if \( \neg x_i \in C_i \). Then the following set of hyperedges is used for \( d\text{-HYPERGRAPH 2-COLORABILITY} \):

\[ E = \{e_i \mid 1 \leq i \leq m\} \cup \{\{y_i, z_i\} \mid 1 \leq i \leq n\} \]

It is easy to verify that every hyperedge in \( E \) has size at most \( d \). Suppose \( F \) has a satisfying truth assignment \( T \). Define color function \( c \) in the following way. Let \( c(y_i) = \text{Red} \), \( c(z_i) = \text{Blue} \) if \( x_i = \text{true} \) in \( T \), and let \( c(y_i) = \text{Blue} \), \( c(z_i) = \text{Red} \) if \( x_i = \text{false} \) according to \( T \). The hyperedges of type \( \{y_i, z_i\} \in E \) are now 2-colored by definition. It is easy to verify that also every hyperedge \( e_i \) for \( 1 \leq i \leq m \) is 2-colored, since \( C_i \) contains at least one \textit{true} and one \textit{false} literal, resulting in one \textit{Red} and one \textit{Blue} vertex in \( e_i \).

Suppose there is a proper 2-coloring \( c: V \rightarrow \{\text{Red}, \text{Blue}\} \) of the hypergraph. Since for every \( i \in [n] \) the hyperedge \( \{y_i, z_i\} \) is present, the variables \( y_i \) and \( z_i \) must have a different color. Consider the following options

- \( y_i \) is \textit{Red} and \( z_i \) is \textit{Blue}. Then to satisfy \( F \), we choose \( x_i = \text{true} \).

- \( z_i \) is \textit{Red} and \( y_i \) is \textit{Blue}. Then to satisfy \( F \), we choose \( x_i = \text{false} \).

Since every edge \( e_i \) contains at least one \textit{Red} vertex, this implies that every clause \( C_i \) for \( i \in [m] \) contains at least one \textit{true} literal using the assignment above. Also, since every hyperedge \( e_i \) contains at least one \textit{Blue} vertex, every \( C_i \) must contain at least one \textit{false} literal. Therefore \( F \) is NAE-satisfied with this assignment.

From the fact that linear-parameter transformations transfer lower bounds (Theorem 2.8) and Lemma 8.1 it now follows that \( d\text{-HYPERGRAPH 2-COLORABILITY} \) cannot have a kernel of size \( O(n^{d-1-\epsilon}) \) for any \( \epsilon > 0 \), unless \( \text{NP} \subseteq \text{coNP/poly} \).

\[ \blacktriangleright \]

### 8.2 Kernel

To prove validity of our kernel we use the following lemma due to Lovász [21]. This lemma was originally developed to prove bounds on the number of hyper edges in
critical 3-chromatic hypergraphs. We say a hypergraph $G$ is critical 3-chromatic if $G$ is not 2-colorable, but the removal of any edge from $G$ yields a 2-colorable hypergraph. Lovász showed that for $r$-uniform critically 3-chromatic hypergraphs the number of edges is bounded by $(\binom{n}{r-1}) = O(n^{r-1})$.

\textbf{Lemma 8.3} ([21]). Let $H$ be an $r$-uniform hypergraph with hyperedges $E_1, \ldots, E_m$. Let $\alpha_1, \ldots, \alpha_m$ be real numbers such that for every $(r-1)$-element subset $A$ of $V(H)$,

\[ \sum_{E_i \supset A} \alpha_i = 0. \]

Then for every partition $\{V_1, V_2\}$ of $V(H)$ the following holds:

\[ \sum_{E_i \subseteq V_1} \alpha_i = (-1)^r \sum_{E_i \subseteq V_2} \alpha_i. \]

\textbf{Theorem 8.4.} $d$-HYPERGRAPH 2-COLORABILITY parameterized by the number of vertices $n$ has a kernel with $2 \cdot n^{d-1}$ hyperedges that can be encoded in $O(n^{d-1} \cdot d \cdot \log n)$ bits.

\textbf{Proof.} Suppose we are given a hypergraph with vertex set $V$ and hyperedges $E$, where each hyperedge contains at most $d$ vertices. We show how to reduce the number of hyperedges without changing the 2-colorability status. Let $E_r \subseteq E$ denote the set of edges in $E$ that contain exactly $r$ vertices. For each $E_r$ we construct a set $E'_r \subseteq E_r$ of representative hyperedges. Enumerate the edges in $E_r$ as $e'_1, \ldots, e'_k$. We construct a $(0,1)$-matrix $M_r$ with $N := \binom{n}{r-1}$ rows and $k$ columns. Consider all possible subsets $A_1, \ldots, A_N$ of size $r-1$ of the set of vertices $V$. Define the elements $m_{ij}$ for $i \in N$ and $j \in k$ of $M_r$ as follows.

\[ m_{ij} := \begin{cases} 1 & \text{if } A_i \subseteq e'_j; \\ 0 & \text{otherwise.} \end{cases} \]

Using Gaussian elimination, compute a base $B$ of the columns of this matrix, which is a subset of the columns that span the column space of $M_r$. Let $E'_r$ contain edge $e'_j$ if the $i$th column of $M_r$ is contained in $B$, and define $E' := \bigcup_{r \in [d]} E'_r$, which forms the kernel. We will now prove validity of the obtained kernel.

\textbf{Lemma 8.5.} $(V, E)$ has a proper 2-coloring $\iff (V, E')$ has a proper 2-coloring.

\textbf{Proof.} $(\Rightarrow)$ Clearly, if $(V, E)$ has a proper 2-coloring then the same coloring is proper for the subhypergraph $(V, E')$ since $E' \subseteq E$.

$(\Leftarrow)$ Now suppose $(V, E')$ has a proper 2-coloring. We say a hyperedge is \textit{monochromatic} with respect to some coloring, if all its vertices receive the same color. We will show that for each $r \in [d]$, no hyperedge of $E_r$ is monochromatic under this coloring. All hyperedges contained in $E'_r$ are 2-colored by definition. Suppose there exists $r \in [d]$,
such that $E_r$ contains a monochromatic hyperedge. Let $E_r = e'_1, \ldots, e'_k$ and let $e'_r$ be a hyperedge in $E_r$ whose vertices all receive the same color.

By reordering the matrix $M_r$, we may assume that the basis $B$ of $M_r$ contains the first $\ell$ columns, thus $i^* > \ell$. Let $m_i$ denote the $i$'th column of $M_r$. Since $m_{i^*}$ is not contained in the basis, there exist coefficients $\alpha_1, \ldots, \alpha_\ell$ such that

$$\sum_{i=1}^\ell \alpha_i \cdot m_i = m_{i^*}.$$

For $i \in [k]$, define:

$$\beta_i := \begin{cases} \alpha_i & \text{if } i \leq \ell; \\
-1 & \text{if } i = i^*; \\
0 & \text{otherwise}. \end{cases}$$

From this definition of $\beta$ it follows that

$$\sum_{i=1}^k \beta_i \cdot m_i = \sum_{i=1}^\ell \alpha_i \cdot m_i - m_{i^*} = 0.$$

Let $A_j$ be any size $(r - 1)$-subset of $V$. Since $m_{i,j} = 1$ exactly when $e_i \supseteq A_j$,

$$\sum_{e_i \supseteq A_j} \beta_i = \sum_{e_i \supseteq A_j} \beta_i m_{i,j} = 0.$$

By Lemma 8.3 we obtain that for any partitioning $V_1 \cup V_2$ of the vertices in $V$,

$$\sum_{\ell \subseteq V_1} \beta_i = (-1)^r \sum_{\ell \subseteq V_2} \beta_i.$$

Consider however the partitioning $(V_1, V_2)$ given by the 2-coloring of the vertices. Then every edge $e_i \in E'_r$ contains at least one vertex of each color and is thereby not fully contained in $V_1$ or $V_2$. As such, these edges contribute 0 to both sides of the equation. The edge $e_{i^*}$ is the only remaining edge with a non-zero coefficient and by assumption, it is contained entirely within one color class. Without loss of generality, let $e_{i^*} \subseteq V_1$. But then $\sum_{\ell \subseteq V_1} \beta_i = -1$ while $(-1)^r \sum_{\ell \subseteq V_2} \beta_i = 0$, which contradicts Formula (1). \hfill $\triangleright$

\textbf{Lemma 8.6.} The obtained kernel contains at most $2 \cdot n^{d-1}$ hyperedges.

\textit{Proof.} Consider matrix $M_r$ for $r \in [d]$. Its rank is bounded by the minimum number of its rows and columns, which is at most $\binom{n}{r-1} \leq n^{r-1}$. As such, we get that $|E'_r| \leq \text{rank}(M_r) \leq n^{r-1}$. Note that $d \leq n$, such that

$$|E'| \leq \sum_{r=1}^d n^{r-1} = n^{d-1} + \sum_{r=1}^{d-1} 2 \cdot n^{d-1}.$$

So $E'$ contains at most $2n^{d-1}$ hyperedges. \hfill $\triangleright$
It follows from Lemma 8.5 and 8.6 that we have presented a kernel consisting of $2 \cdot n^{d-1}$ hyperedges and $n$ vertices. Thereby, we can store the representation of a vertex in $O(\log n)$ bits. Since every hyperedge consists of at most $d$ vertices, we can store this kernel in $O(d \cdot \log n \cdot n^{d-1})$ bits.

As we have seen in Theorem 8.2, we can give a linear parameter transformation from $d$-NAE-SAT to $d$-HYPERGRAPH 2-COLORABILITY. Suppose we are given an instance of $d$-NAE-SAT, we can now obtain a generalized kernel for this problem by transforming it to an instance of $d$-HYPERGRAPH 2-COLORABILITY and then applying the kernel described in Theorem 8.4. This results in an instance of at most $O(n^{d-1})$ hyperedges.
9 Conclusion

We have shown that several well-known graph problems do not have a generalized kernel of size $O(n^{2-\epsilon})$, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$. For example, we showed that 4-coloring does not have a generalized kernel of sub-quadratic size. However, it is left as an open question whether this also holds for the 3-coloring problem, that is also known to be NP-hard. We also showed that a kernel of sub-quadratic size for one of the problems Hamiltonian Cycle, (Connected) Dominating Set, Max Leaf Spanning Tree, Nonblocker, and Feedback Arc Set would imply that $\text{NP} \subseteq \text{coNP}/\text{poly}$.

Most graph problems we considered ask a question about the vertices of the graph, such as 4-coloring which asks if there exists a proper coloring of the vertices. For edge-based problems, such as the maximum cut problem that asks for a cut in the graph of size $k$, it seems harder to find a cross-composition using the same strategy as we used in this thesis. The approach to select one instance and somehow remove the effect of all other instances does not seem to work out. Only for Feedback Arc Set, we have shown a lower bound using a linear parameter transformation.

Furthermore, we showed a lower bound for $d$-NAE-SAT and $d$-Hypergraph 2-colorability and a nontrivial sparsification to match the lower bound, using a lemma by Lovász. It is an interesting open question whether there also exist problems defined on general graphs that allow a non-trivial sparsification, since so far we have only given examples of problems that do not have such a sparsification.
References


