Search-Space Reduction via Essential Vertices

Benjamin Merlin Bumpus
Eindhoven University of Technology, The Netherlands

Bart M. P. Jansen
Eindhoven University of Technology, The Netherlands

Jari J. H. de Kroon
Eindhoven University of Technology, The Netherlands

Abstract

We investigate preprocessing for vertex-subset problems on graphs. While the notion of kernelization, originating in parameterized complexity theory, is a formalization of provably effective preprocessing aimed at reducing the total instance size, our focus is on finding a non-empty vertex set that belongs to an optimal solution. This decreases the size of the remaining part of the solution which still has to be found, and therefore shrinks the search space of fixed-parameter tractable algorithms for parameterizations based on the solution size. We introduce the notion of a c-essential vertex as one that is contained in all c-approximate solutions. For several classic combinatorial problems such as Odd Cycle Transversal and Directed Feedback Vertex Set, we show that under mild conditions a polynomial-time preprocessing algorithm can find a subset of an optimal solution that contains all 2-essential vertices, by exploiting packing/covering duality. This leads to FPT algorithms to solve these problems where the exponential term in the running time depends only on the number of *non-essential* vertices in the solution.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Packing and covering problems; Theory of computation → Linear programming; Theory of computation → Fixed parameter tractability

Keywords and phrases fixed-parameter tractability, essential vertices, covering versus packing

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.30

Related Version *Full Version*: `https://arxiv.org/abs/2207.00386` [7]

Funding Funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803421, ReduceSearch).

1 Introduction

Background and motivation. Due to the enormous potential of preprocessing to speed up the algorithmic solution to combinatorial problems [1, 2, 3, 43, 44], a large body of work in theoretical computer science is devoted to understanding its power and limitations. Using the notion of *kernelization* [4, 20, 21, 25, 29, 33] from parameterized complexity [12, 15] it is possible to formulate a guarantee on the size of the instance after preprocessing based on the parameter of the original instance. Under this model, a good preprocessing algorithm is a *kernelization algorithm*: given a parameterized instance (x, k), it outputs an equivalent instance (x', k') of the same decision problem such that $|x'| + k' \leq f(k)$ for some function f that bounds the size of the kernel. Research into kernelization led to deep algorithmic insights, including connections to protrusions and finite-state properties [5], well-quasi ordering [22], and matroids [30]; these positive results were complemented by impossibility results [13, 16, 30] delineating the boundaries of tractability.

Results on kernelization led to profound insights into the limitations of polynomial-time data compression for NP-hard problems. However, as recently advocated [14], the definition of kernelization only gives guarantees on the *size* of the instance after preprocessing, which does not directly correspond to the running time of subsequently applied algorithms. If the preprocessed instance is not solved by brute force, but via a fixed-parameter tractable algorithm whose running time is of the form $f(k) \cdot n^{\mathcal{O}(1)}$, then the exponential dependence in the running time is on the value of the *parameter* k, which is not guaranteed to decrease via kernelization. In fact, if $P \neq NP$ then no polynomial-time preprocessing algorithm can guarantee to always decrease the parameter of an NP-hard fixed-parameter tractable problem, as iterating the preprocessing algorithm would lead to its solution in polynomial time. In this work, we develop a new analysis of preprocessing aimed at reducing the search space of the follow-up algorithm. We apply this framework to combinatorial optimization problems on graphs, whose goal is to find a minimum vertex-subset satisfying certain properties. The main idea behind the framework is to define formally what it means for a vertex to be *essential* for making reasonable solutions to the problem, and to prove that an efficient preprocessing algorithm can detect a subset of an optimal solution that contains all such essential vertices.

Before stating our results, we introduce and motivate the model. We consider vertex-subset minimization problems on (possibly directed) graphs, in which the goal is to find a minimum vertex subset having a certain property. Examples of the problems we study include Vertex Cover, Odd Cycle Transversal, and Dominating Set. The analysis of such minimization problems, parameterized by the size of the solution, has played an important role in the literature (cf. [9, 17, 27, 38, 40]). Our starting point is the thesis that a good preprocessing algorithm should reduce the *search space*. Since many graph problems are known to be fixed-parameter tractable when parameterized by the size of the solution, we can reduce the search space of these FPT algorithms by finding one or more vertices which are part of an optimal solution, thereby decreasing the size of the solution still to be found in the reduced instance (i.e. the parameter value).

Since in general no polynomial-time algorithm can guarantee to identify at least one vertex that belongs to an optimal solution, the guarantee of the effectiveness of the preprocessing algorithm should be stated in a more subtle way. When solving problems by hand, one sometimes finds that certain vertices v are easily identified to belong to an optimal solution, as avoiding them would force the solution to contain a prohibitively large number of alternative vertices and therefore be suboptimal. Can an efficient preprocessing algorithm identify those vertices that cannot be avoided when making an optimal solution?

Since many NP-hard problems remain hard even when there is a unique solution [42], this turns out to be too much to ask as it would allow instances with unique solutions to be solved in polynomial time, which leads to $NP = RP$. We therefore have to relax the requirements on the preprocessing guarantee slightly, as follows. For an instance of a vertex-subset minimization problem Π on a graph G, we denote the minimum size of a solution on G by $\mathrm{OPT}_\Pi(G)$. For a fixed $c \in \mathbb{R}_{\geq 1}$, we say a vertex $v \in V(G)$ is *c-essential* for Π on G if all feasible solutions $S \subseteq V(G)$ for Π whose total size is at most $c \cdot \mathrm{OPT}_\Pi(G)$ contain v. Based on this notion, we can ask ourselves: can an efficient preprocessing algorithm identify part of an optimal solution if there is at least one c-essential vertex?

Phrased in this way, the algorithmic task becomes more tractable. For example, for the Vertex Cover problem, selecting all vertices that receive the value 1 in an optimal half-integral solution to the LP-relaxation results in a set S which is contained in some optimal solution (by the Nemhauser-Trotter theorem [37], cf [12, §2.5]), and at the same time includes all 2-essential vertices: any vertex $v \notin S$ only has neighbors of value $\frac{1}{2}$ and 1, which implies that the set X of vertices other than v whose value in the LP relaxation is at least $\frac{1}{2}$, forms a feasible solution which avoids v. Its cardinality is at most twice the cost of the LP relaxation and therefore X is a 2-approximation. Hence S contains all 2-essential vertices.

This example shows that a preprocessing step that detects c-essential vertices without any additional information is sometimes possible. However, to be able to extend the scope of our results also to problems which *do not* have polynomial-time constant-factor approximations, we slightly relax the requirements on the preprocessing algorithm as follows. Let Π be a minimization problem on graphs whose solutions are vertex subsets and let $c \in \mathbb{R}_{\geq 1}$.

c-ESSENTIAL DETECTION FOR Π
Input: A graph G and integer k.
Task: Find a vertex set $S \subseteq V(G)$ such that:
G1 if $\mathrm{OPT}_\Pi(G) \leq k$, then there is an optimal solution in G containing all of S, and
G2 if $\mathrm{OPT}_\Pi(G) = k$, then S contains all c-essential vertices.

In this model, the preprocessing task is facilitated by supplying an additional integer k in the input. The correctness properties of the output S are formulated in terms of k. If $\mathrm{OPT}_\Pi(G) \leq k$, then the set S is required to be part of an optimal solution. The upper bound on $\mathrm{OPT}_\Pi(G)$ is useful to the algorithm: whenever the algorithm establishes that avoiding v would incur a cost of more than k, it is safe to add v to S. If $\mathrm{OPT}_\Pi(G) = k$, then the algorithm should guarantee that S contains all c-essential vertices. Knowing a lower bound on $\mathrm{OPT}_\Pi(G)$ is useful to the algorithm in case it can establish that any optimal solution containing v can be transformed into one avoiding v whose cost is $(c-1) \cdot k$ larger, which yields a c-approximation if $(c-1) \cdot k \leq (c-1)\,\mathrm{OPT}_\Pi(G)$. Hence vertices for which such a replacement exists are not c-essential and may safely be left out of S.

Results. We present polynomial-time algorithms for c-ESSENTIAL DETECTION FOR Π for a range of vertex-deletion problems Π and small values of c; typically $c \in \{2, 3\}$. Example applications include VERTEX COVER and FEEDBACK VERTEX SET, and also CHORDAL VERTEX DELETION (for which no $O(1)$-approximation is known), ODD CYCLE TRANSVERSAL (for which no $O(1)$-approximation exists, assuming the Unique Games Conjecture [28, 45]), and even DIRECTED ODD CYCLE TRANSVERSAL (which is $W[1]$-hard parameterized by solution size [35]).

The model of c-ESSENTIAL DETECTION FOR Π is chosen such that the detection algorithms whose correctness is formulated with respect to the value of k, can be used as a preprocessing step to optimally solve vertex-subset problems without any knowledge of the optimum. Let $\mathcal{E}^\Pi_c(G)$ denote the set of c-essential vertices in G, which is well-defined since all optimal solutions contain all c-essential vertices. By using a preprocessing step that detects a superset of the c-essential vertices in the solution, we can effectively improve the running-time guarantee for FPT algorithms parameterized by solution size from $f(\mathrm{OPT}_\Pi(G)) \cdot |V(G)|^{\mathcal{O}(1)}$, to $f(\mathrm{OPT}_\Pi(G) - |\mathcal{E}^\Pi_c(G)|) \cdot |V(G)|^{\mathcal{O}(1)}$. This leads to the following results.

▶ **Theorem 1.1.** *For each problem Π with coefficient c and parameter dependence f listed in Table 1 that is not $W[1]$-hard, there is an algorithm that, given a graph G, outputs an optimal solution in time $f(\ell) \cdot |V(G)|^{\mathcal{O}(1)}$, where $\ell := \mathrm{OPT}_\Pi(G) - |\mathcal{E}^\Pi_c(G)|$ is the number of vertices in an optimal solution which are not c-essential.*

Hence the running time for solving these problems does not depend on the total size of an optimal solution, only on the part of the solution that does not consist of c-essential vertices. The theorem effectively shows that by employing c-ESSENTIAL DETECTION FOR Π as a preprocessing step, the size of the search space no longer depends on the total solution size but only on its non-essential vertices.

Table 1 For each problem Π, there is a polynomial-time algorithm for c-Essential detection for the stated value of c. Combined with the state of the art algorithm for the natural parameterization, this leads to an algorithm solving the problem in time $f(\ell) \cdot |V(G)|^{\mathcal{O}(1)}$ where $\ell = \mathrm{OPT}_\Pi(G) - |\mathcal{E}_c^\Pi(G)|$.

Problem	c	$f(\ell)$	Reference
Vertex Cover	2	1.2738^ℓ	[9]
Feedback Vertex Set	2	2.7^ℓ	[31]
Directed Feedback Vertex Set	2	$4^\ell \cdot \ell!$	[10]
Odd Cycle Transversal	2	2.3146^ℓ	[34]
Directed Odd Cycle Transversal	3	W[1]-hard	[35]
Chordal Vertex Deletion	13	$2^{\mathcal{O}(\ell \log \ell)}$	[8]

We also prove limitations to this approach. Assuming FPT $\neq W[1]$, for Dominating Set, Perfect Deletion (in which the goal is to obtain a perfect graph by vertex deletions) and Wheel-Free Deletion, there is no polynomial-time algorithm for c-Essential detection for any $c \in \mathbb{R}_{\geq 1}$. In fact, we can even rule out such algorithms running in time $f(k) \cdot |V(G)|^{\mathcal{O}(1)}$. These results are based on FPT-inapproximability results for Dominating Set [40] and existing reductions [26, 32] to the mentioned vertex-deletion problems.

Techniques. The main work lies in the algorithms for c-Essential detection, which are all based on covering/packing duality for forbidden induced subgraphs to certain graph classes, or variations thereof in terms of (integer) solutions to certain linear programs and their (integer) duals. To understand the relation between detecting essential vertices and covering/packing duality, consider the Odd Cycle Transversal problem (OCT). Following the argumentation for the classic Erdős-Pósa theorem [18], in general there is no constant c such that any graph either has an odd cycle transversal of size $c \cdot k$, or a packing of k vertex-disjoint odd cycles. However, we show that a linear packing/covering relation holds in the following slightly different setting. If $G - v$ is bipartite (so all odd cycles of G intersect v), then the minimum size of an OCT avoiding v equals the maximum cardinality of a packing of odd cycles which pairwise intersect only at v. We can leverage this statement to prove that any vertex v which is not at the center of a flower (see Definition 3.1) of more than $\mathrm{OPT}_{\mathrm{oct}}(G)$ odd cycles, is not 2-essential: for any optimal solution X containing v, the graph $G' := G - (X \setminus \{v\})$ becomes bipartite after removal of v and by assumption does not contain a packing of more than $\mathrm{OPT}_{\mathrm{oct}}(G)$ odd cycles pairwise intersecting at v. So by covering/packing duality on G', it has an OCT X' of size at most $\mathrm{OPT}_{\mathrm{oct}}(G)$ avoiding v, so that $(X \setminus \{v\}) \cup X'$ is a 2-approximation in G which avoids v, showing that v is not 2-essential. Since v is clearly contained in an optimal solution whenever there is a flower of more than $\mathrm{OPT}_{\mathrm{oct}}(G)$ odd cycles centered at v, this yields a method for c-Essential detection when using a known reduction [23] to Maximum Matching to test for a flower of odd cycles.

Organization. After presenting preliminaries in Section 2, we give algorithms to detect essential vertices based on covering/packing duality in Section 3 and based on integrality gaps in Section 4. In Section 5 we show how these detection subroutines can be used to improve the parameter dependence of FPT algorithms parameterized by solution size. The lower bounds are presented in Section 6. The investigation of c-essential vertices has close connections to the area of perturbation stability, which we briefly explore in Section 7. We conclude in Section 8. Due to space limitations, proofs of statements marked (★) are deferred to the full version [7].

2 Preliminaries

We consider finite simple graphs, some of which are directed. We use standard notation for graph algorithms; any terms not defined here can be found in the textbook by Cygan et al. [12]. We consider vertex-deletion problems on graphs. For a graph class $\mathcal{C}$, a $\mathcal{C}$-modulator in a graph G is a vertex set $S \subseteq V(G)$ such that $G - S \in \mathcal{C}$. The minimum size of a $\mathcal{C}$-modulator in G is denoted $\mathrm{OPT}_{\mathcal{C}}(G)$. The corresponding minimization problem is defined as follows.

$\mathcal{C}$-Deletion
Input: A graph G.
Task: Find a minimum-size vertex-subset $S \subseteq V(G)$ such that $G - S \in \mathcal{C}$.

Throughout this paper we consider hereditary graph classes $\mathcal{C}$. These can be characterized by a (possibly infinite) set of forbidden induced subgraphs denoted $\mathsf{forb}(\mathcal{C})$. The $\mathcal{C}$-Deletion problem is equivalent to finding a minimum set $S \subseteq V(G)$ such that no induced subgraph of $G - S$ is isomorphic to a graph in $\mathsf{forb}(\mathcal{C})$. We say that such a set S hits all graphs from $\mathsf{forb}(\mathcal{C})$ in G. For these classes the vertex set $V(G)$ is a trivial $\mathcal{C}$-modulator (since the empty graph is in all hereditary classes), which ensures that the task of finding a smallest modulator is always well-defined.

A graph is perfect if for every induced subgraph the size of a largest clique is equivalent to its chromatic number. Equivalently, a graph is perfect if it has no induced cycle of odd length at least five or its edge complement (cf. [24]). A graph is chordal if it has no induced cycle of length at least four. A graph is bipartite if its vertex set can be partitioned into two independent sets, or equivalently, it does not contain an odd-length cycle. Given a graph G and a set $T \subseteq V(G)$, a T-path is a path with at least one edge with both endpoints contained in T. A T-path is odd if it has an odd number of edges. For $u, v \in V(G)$, a (u,v)-separator is a set $S \subseteq V(G) \setminus \{u,v\}$ that disconnects u from v. If G is a directed graph, then in $G - S$ there is no directed path from u to v instead.

3 Positive results via Packing Covering

In this section we provide polynomial-time algorithms for c-Essential detection for Π for various problems Π. The case for the Vertex Cover problem was given in Section 1. The results in this section are all based on packing/covering duality (cf. [11], [41, §73]). Towards this end, we generalize the notion of *flowers*, which played a key role in kernelization for Feedback Vertex Set [6]. While flowers were originally formulated as systems of cycles (forbidden structures for Feedback Vertex Set) pairwise intersecting in a single common vertex, we generalize the notion to near-packings of arbitrary structures here.

▶ **Definition 3.1.** *Let $\mathcal{F}$ be a set of graphs. For a graph G and $v \in V(G)$, a $(v,\mathcal{F})$-*flower with p petals *in G is a set $\{C_1, C_2, \ldots, C_p\}$ of induced subgraphs of G such that each C_i (with $i \in [p]$) is isomorphic to some member of $\mathcal{F}$and such that $V(C_i) \cap V(C_j) = \{v\}$ for all distinct $i, j \in [p]$. The $\mathcal{F}$-*flower number *of a vertex $v \in V(G)$, denoted $\Gamma_{\mathcal{F}}(G,v)$, is the largest integer p for which there is a $(v,\mathcal{F})$-flower in G with p petals.*

We show a general theorem for finding 2-essential vertices for $\mathcal{C}$-deletion if a maximum $(v,\mathsf{forb}(\mathcal{C}))$-flower can be computed in polynomial-time. It applies to those classes $\mathcal{C}$ where graphs with $G - v \in \mathcal{C}$ obey a min-max relation between $\mathcal{C}$-modulators avoiding v and packings of forbidden induced subgraphs intersecting only at v.

▶ **Theorem 3.2.** *Let $\mathcal{C}$ be a hereditary graph class such that, for any graph G and vertex $v \in V(G)$ with $G - v \in \mathcal{C}$, the minimum size of a $\mathcal{C}$-modulator avoiding v in G equals $\Gamma_{\mathsf{forb}(\mathcal{C})}(G, v)$. Suppose there exists a polynomial-time algorithm $\mathcal{A}$ that, given a graph G and vertex $v \in V(G)$, computes $\Gamma_{\mathsf{forb}(\mathcal{C})}(G, v)$. Then there is a polynomial-time algorithm that solves 2-*ESSENTIAL DETECTION FOR *$\mathcal{C}$-*DELETION.*

Proof. Apply algorithm $\mathcal{A}$ to each vertex $v \in V(G)$ and let S be the set of vertices for which it finds that $\Gamma_{\mathsf{forb}(\mathcal{C})}(G, v) > k$. We argue that Properties G1 and G2 are satisfied. If $\mathrm{OPT}_{\mathcal{C}}(G) \leq k$, then every vertex in S is contained in every optimal solution for G since a size-k solution cannot hit a flower of $k + 1$ petals from $\mathsf{forb}(\mathcal{C})$ without using v. Therefore Property G1 is satisfied. Next suppose that $\mathrm{OPT}_{\mathcal{C}}(G) = k$ and let X be an optimal solution. We argue that each vertex $v \notin S$ is not 2-essential. Clearly this holds for any vertex $v \notin X$, so suppose that $v \in X$. Note that for every vertex $v \notin S$ we have $\Gamma_{\mathsf{forb}(\mathcal{C})}(G, v) \leq k$, which implies that $\Gamma_{\mathsf{forb}(\mathcal{C})}(G', v) \leq k$ where $G' := G - (X \setminus \{v\})$. Note that since $G' - v \in \mathcal{C}$, by assumption there exists a $\mathcal{C}$-modulator $X' \subseteq V(G') \setminus \{v\}$ in G' of size $\Gamma_{\mathsf{forb}(\mathcal{C})}(G', v) \leq k$. Observe that $(X \setminus \{v\}) \cup X'$ is a $\mathcal{C}$-modulator in G of size at most $2k$ that avoids v and therefore v is not 2-essential. ◀

Theorem 3.2 allows us to conclude the following result for FEEDBACK VERTEX SET (FVS) and its directed variant (DFVS) using Gallai's theorem and Menger's theorem, respectively.

▶ **Lemma 3.3 (★).** *There are polynomial-time algorithms for* 2-*ESSENTIAL DETECTION FOR* Π *for* $\Pi \in \{FVS, DFVS\}$.

Next, we consider ODD CYCLE TRANSVERSAL (OCT), which corresponds to $\mathcal{C}$-DELETION where $\mathcal{C}$ is the class of bipartite graphs and $\mathsf{forb}(C)$ consists of all odd cycles. In order to apply Theorem 3.2 to OCT, we first argue that the class of bipartite graphs satisfies the preconditions. The proof is similar to that of Geelen et al. [23, Lemma 11] who reduce the problem of packing odd cycles containing v to a matching problem. We note that, although their result can be used to obtain a 3-essential detection algorithm, we will show (Lemma 3.7) how to efficiently detect 2-essential vertices as well. If the graph resulting from their construction has a large matching, then there is a large (v,$\mathsf{forb}(\mathcal{C})$)-flower. If on the other hand there is no large matching, then the Tutte-Berge formula is used to obtain a set of size $2k$ that hits all the odd cycles passing through v. We show that if the graph $G - v$ is bipartite instead, then this second argument can be improved to obtain a hitting set of size k by noting that the lack of a large matching implies that there is a small vertex cover due to Kőnig's theorem. This is below, using the viewpoint that odd cycles in G correspond to odd T-paths in $G - v$ for $T = N_G(v)$.

▶ **Lemma 3.4.** *For any undirected graph G and set $T \subseteq V(G)$, a maximum packing of odd T-paths can be computed in polynomial time. Moreover, if G is bipartite then the cardinality of a maximum packing of odd T-paths is equal to the minimum size of a vertex set which intersects all odd T-paths.*

Proof. We reduce to matching as in [23, Lemma 11]. Construct a graph H as follows. For each $v \in V(G) \setminus T$, let $v' \notin V(G)$ be a copy of v. Let $V(H) = V(G) \cup \{v' \mid v \in V(G) \setminus T\}$ and $E(H) = E(G) \cup \{u'v' \mid uv \in E(G - T)\} \cup \{vv' \mid v \in V(G) \setminus T\}$. Note that the graph H consists of the disjoint union of G and a copy of $G - T$, with an added edge between $v \in V(G) \setminus T$ and its copy v'. Geelen et al. [23] mention that there is a 1-1 correspondence between odd T-paths in G and certain augmenting paths in H. For completeness we give a self-contained argument.

▷ Claim 3.5. Graph G contains k vertex-disjoint odd T-paths if and only if H has a matching M of size $|V(G) \setminus T| + k$. Furthermore, given a matching M in H of size $|V(G) \setminus T| + k$ we can compute a set of k vertex-disjoint odd T-paths in polynomial time.

Proof. ($\Rightarrow$) Let $\mathcal{P} = (P_1, \ldots, P_k)$ be a set of k vertex-disjoint odd T-paths in G. Consider a path $P = (v_1, \ldots, v_{2\ell}) \in \mathcal{P}$, where $\ell \geq 1$. First note that we can assume that $V(P) \cap T = \{v_1, v_{2\ell}\}$, since if $v_i \in T$ for some $1 < i < 2\ell$, then either $(v_1, \ldots, v_i)$ or $(v_i, \ldots, v_{2\ell})$ is an odd T-path and we can update $\mathcal{P}$ accordingly. Construct a matching M in H as follows. For any path $P = (v_1, \ldots, v_{2\ell}) \in \mathcal{P}$, add the edges $v_1v_2, v_2'v_3', \ldots, v_{2\ell-1}v_{2\ell}$, alternating between original vertices and copy vertices. This is possible as P is of odd length. For any vertex in $u \in V(G) \setminus T$ that is not contained in an odd T-path, we add uu' to M. Observe that at least $2|V(G) \setminus T| + 2k$ vertices are matched, therefore $|M| \geq |V(G) \setminus T| + k$ as desired.

($\Leftarrow$) Let M be a matching of size $|V(G) \setminus T| + k$. If M contains both uv and $u'v'$ for $u, v \in V(G) \setminus T$, then update M by removing them and inserting uu' and vv' instead. If for $v \in V(G) \setminus T$ only one of v and v' is matched, and it is not matched to its copy, then match it to its copy instead. Afterwards let $E' := \{uv \in E(G) \mid uv \in M \vee u'v' \in M\}$. Observe that in $G[E']$, each vertex in $V(G) \setminus T$ has degree 0 or 2. For each $v \in V(G) \setminus T$ such that v has degree 0 in $G[E']$, add vv' to M if it is not in already. Note that all vertices of $H - T$ are matched. It follows that at least $2k$ vertices in T are matched by M and they have degree 1 in $G[E']$. Observe that $G[E']$ is a collection of paths and cycles with all degree-1 vertices in T. We get that there are k T-paths in G that are of odd length by construction (every even numbered edge in $G[E']$ originated from the copy part of H). Note that we can find them in polynomial time. ◁

Since a maximum matching can be computed in polynomial time, by the claim above we get that a maximum packing of vertex-disjoint odd T-paths can be computed in polynomial time. Next we prove the second part of the statement.

▷ Claim 3.6. If G is bipartite and a maximum matching M in H has size $|V(G) \setminus T| + k$, then there is a set $S \subseteq V(G)$ of size at most k such that $G - S$ has no odd-length T-path.

Proof. Observe that since G is bipartite, the graph H is bipartite as well. By Kőnig's theorem [41, Theorem 16.2], H has a vertex cover X of size $|V(G) \setminus T| + k$. Let $S = (X \cap T) \cup \{u \mid \{u, u'\} \subseteq X\}$. Note that for each $u \in V(G) \setminus T$, at least one of $u \in X$ or $u' \in X$ must hold to cover the edge uu', thereby already accounting for $|V(G) \setminus T|$ vertices of the cover. It follows that $|S| \leq k$. We argue that S hits all odd-length T-paths in G.

For the sake of contradiction suppose there is some odd-length T-path from $t_1 \in T$ to $t_2 \in T \setminus \{t_1\}$ in $G - S$. Let $P = (t_1, v_1, \ldots, v_{2\ell}, t_2)$ be a (not necessarily induced) shortest odd T-path. Note that neither of t_1 and t_2 belong to X by construction of S. Furthermore $P \setminus \{t_1, t_2\} \subseteq V(G) \setminus T$, as otherwise we could construct a shorter odd-length T-path. Consider the vertices $V(P) \cup \{v_1', \ldots, v_{2\ell}'\} \subseteq V(H)$. By construction of S, the vertex cover X has exactly one of v_i and v_i' for each $i \in [2\ell]$. Observe that for it to be a vertex cover, X must contain vertices of $P \setminus \{t_1, t_2\}$ and their copies in an alternating fashion since for each edge v_i, v_{i+1} of P, the graph H contains edges $v_iv_i', v_{i+1}v_{i+1}', v_iv_{i+1}, v_i'v_{i+1}'$. Without loss of generality, let $v_1 \in X$. It follows that $v_{2\ell} \notin X$. But this contradicts that X is a vertex cover as $v_{2\ell}t_2$ is not covered. We conclude that S hits all odd-length T-paths in G. ◁

By Claim 3.5, a maximum packing of k of vertex-disjoint odd T-paths in G implies a matching in H of size $|V(G) \setminus T| + k$. By Claim 3.6, we can create a set of size at most k that intersects all odd T-paths in the bipartite graph G. Clearly such a set has size at least k. It follows that if G is bipartite, then the cardinality of a maximum packing of odd

T-paths is equal to the minimum size of a vertex set which intersects all odd T-paths. (For completeness, we remark that this last property can also be derived from Menger's theorem: in a bipartite graph with bipartition into $A \cup B$, a T-path is odd if and only if its endpoints belong to different partite sets, so a maximum packing of odd T-paths is equivalent to a maximum set of vertex-disjoint paths between $A \cap T$ and $B \cap T$.) ◀

By observing that odd T-paths in $G - v$ directly correspond to flowers with odd cycles pairwise intersecting at v in G, Lemma 3.4 and Theorem 3.2 imply the following.

▶ **Lemma 3.7.** *There is a polynomial-time algorithm for* 2-ESSENTIAL DETECTION FOR *OCT.*

The DOCT problem corresponds to $\mathcal{C}$-deletion where $\mathsf{forb}(C)$ consists of all directed cycles of odd length. Using Menger's theorem on an auxiliary graph, we can detect 3-essential vertices for this problem.

▶ **Lemma 3.8 (★).** *There is a polynomial-time algorithm for* 3-ESSENTIAL DETECTION FOR *DOCT.*

We cannot use the approach based on computing a maximum $(v, \mathcal{F})$-flower for the CHORDAL DELETION problem; a simple reduction[1] from DISJOINT PATHS [39] shows that it is NP-hard to compute a maximum $(v, \mathcal{F})$-flower when $\mathcal{F}$ is the set of chordless cycles of length at least four. In the next section, we will therefore use an approach based on the linear-programming relaxation to deal with CHORDAL DELETION.

4 Positive results via Linear Programming

Consider the following natural linear program for $\mathcal{C}$-DELETION for hereditary graph classes $\mathcal{C}$. The LP corresponding to an input graph G is defined on the variables $(x_u)_{u \in V(G)}$, as follows.

$\mathcal{C}$-DELETION LP
Objective: minimize $\sum_{u \in V(G)} x_u$.
Subject to:
- $\sum_{u \in V(H)} x_u \geq 1$ for each induced subgraph H of G isomorphic to a graph in $\mathsf{forb}(\mathcal{C})$,
- $0 \leq x_u \leq 1$ for each $u \in V(G)$.

In the corresponding integer program, the constraint $0 \leq x_u \leq 1$ is replaced by $x_u \in \{0, 1\}$. We say that a minimization LP has *integrality gap* at most c for some $c \in \mathbb{R}$ if the cost of an integer optimum is at most c times the cost of a fractional optimum. In general, the number of constraints in the $\mathcal{C}$-DELETION LP can be exponential in the size of the graph. Using the ellipsoid method (cf. [41]), this can be handled using a separation oracle: a polynomial-time algorithm that, given an assignment to the variables, outputs a violated constraint if one exists. It is well-known (cf. [41, Thm. 5.10]) that linear programs with an exponential number of constraints can be solved in polynomial time using a polynomial-time separation oracle. To detect essential vertices, the integrality gap of a slightly extended LP will be crucial. We define the v-AVOIDING $\mathcal{C}$-DELETION LP for a graph G and distinguished vertex $v \in V(G)$ as the $\mathcal{C}$-DELETION LP with the additional constraint that $x_v = 0$. Hence its integral solutions correspond to $\mathcal{C}$-modulators avoiding v.

[1] Starting from an instance $(G, (s_1, t_1), \ldots, (s_\ell, t_\ell))$ of DISJOINT PATHS satisfying $s_i t_i \notin E(G)$ for all $i \in [\ell]$ (which is without loss of generality), insert a vertex v adjacent to $A = \bigcup_{i=1}^{\ell} \{s_i, t_i\}$ and insert all edges between vertices in A except $s_i t_i$ for each $i \in [\ell]$.

▶ **Theorem 4.1** (★). *Let $\mathcal{C}$ be a hereditary graph class such that for each graph G and $v \in V(G)$ satisfying $G - v \in \mathcal{C}$, the integrality gap of v-AVOIDING $\mathcal{C}$-DELETION on G is at most $c \in \mathbb{R}_{\geq 1}$. If there is a polynomial-time separation oracle for the $\mathcal{C}$-DELETION LP, then there is a polynomial-time algorithm for $(c+1)$-ESSENTIAL DETECTION FOR $\mathcal{C}$-DELETION.*

Using known results on covering versus packing for chordless cycles in near-chordal graphs, the approach above can be used to detect essential vertices for CHORDAL DELETION. For the class of chordal graphs, the corresponding set of forbidden induced subgraphs is the class hole of all holes, i.e., induced chordless cycles of length at least four.

▶ **Lemma 4.2** (★). *There is a polynomial-time algorithm for 13-ESSENTIAL DETECTION FOR CHORDAL DELETION.*

5 Consequences for Parameterized Algorithms

In this section we show how the algorithms for c-ESSENTIAL DETECTION from the previous section can be used to solve $\mathcal{C}$-DELETION for various classes $\mathcal{C}$, despite the fact that the detection algorithms only work when certain guarantees on k are met. The main theorem connecting the detection problem to solving $\mathcal{C}$-DELETION is the following.

▶ **Theorem 5.1.** *Let $\mathcal{A}$ be an algorithm that, given a graph G and an integer k, runs in time $f(k) \cdot |V(G)|^{\mathcal{O}(1)}$ for some non-decreasing function f and returns a minimum-size $\mathcal{C}$-modulator if there is one of size at most k. Let $\mathcal{B}$ be a polynomial-time algorithm for c-ESSENTIAL DETECTION FOR $\mathcal{C}$-DELETION. Then there is an algorithm that, given a graph G, outputs a minimum-size $\mathcal{C}$-modulator in time $f(\ell) \cdot |V(G)|^{\mathcal{O}(1)}$, where $\ell = OPT_{\mathcal{C}}(G) - |\mathcal{E}_c(G)|$ is the* c-non-essentiality.

Proof. First we describe the algorithm as follows. For each $0 \leq k \leq |V(G)|$, let S_k be the result of running $\mathcal{B}$ on (G, k), let $G_k := G - S_k$, and let $b_k := k - |S_k|$.

Letting L be the list of all triples (G_k, S_k, b_k) sorted in increasing order by their third component b_k, proceed as follows. For each $(G_k, S_k, b_k) \in L$, run $\mathcal{A}$ on (G_k, b_k) to find a minimum $\mathcal{C}$-modulator $S_{\mathcal{A}}$ of size at most b_k, if one exists. If $|S_{\mathcal{A}}| = b_k$, then output $S_{\mathcal{A}} \cup S_k$ as a minimum $\mathcal{C}$-modulator in G.

To analyze the algorithm, we first argue it always outputs a solution. For the call with $k^* = \mathrm{OPT}_{\mathcal{C}}(G)$, both conditions of the detection problem are met. Hence by Property G1 the set S_{k^*} is contained in a minimum modulator in G, so that $\mathrm{OPT}_{\mathcal{C}}(G - S_{k^*}) = \mathrm{OPT}_{\mathcal{C}}(G) - |S_{k^*}| = k^* - |S_{k^*}|$. Therefore graph $G_{k^*} = G - S_{k^*}$ has a modulator of size at most $b_{k^*} = \mathrm{OPT}_{\mathcal{C}}(G - S_{k^*})$ and none which are smaller, so that $\mathcal{A}$ correctly outputs a modulator of size b_{k^*}. In turn, this causes the overall algorithm to terminate with a solution.

Having established that the algorithm outputs a solution, we proceed to show that it outputs a minimum-size modulator whenever it outputs a solution (which may be in an earlier iteration than for $k^* = \mathrm{OPT}_{\mathcal{C}}(G)$). Let k' be the value of k that is reached when the algorithm outputs a solution $S_{\mathcal{A}} \cup S_{k'}$. Then we know:

1. algorithm $\mathcal{A}$ found a minimum-size modulator $S_{\mathcal{A}}$ in $G_{k'}$ of size at most $b_{k'}$, and
2. the set $S_{\mathcal{A}} \cup S_{k'}$ is a modulator in G, since $S_{\mathcal{A}}$ is a modulator in $G_{k'} = G - S_{k'}$, and therefore $\mathrm{OPT}_{\mathcal{C}}(G) \leq b_{k'} + |S_{k'}| = k'$.

To see that the algorithm is correct, notice that, since $\mathrm{OPT}_{\mathcal{C}}(G) \leq k'$, the set $S_{k'}$ is contained in some minimum-size modulator for G (since $\mathcal{B}$ satisfies Property G1). Hence $\mathrm{OPT}_{\mathcal{C}}(G_{k'}) = \mathrm{OPT}_{\mathcal{C}}(G) - |S_{k'}|$. Since $\mathcal{A}$ outputs a minimum-size modulator if there is one of size at most b_k, we have $|S_{\mathcal{A}}| = \mathrm{OPT}_{\mathcal{C}}(G) - |S_{k'}|$, so that $\mathcal{A}(G_{k'}) \cup S_{k'}$ is a feasible modulator of size $\mathrm{OPT}_{\mathcal{C}}(G)$ and therefore optimal.

Now we prove the desired running-time bound. First of all, notice that we can determine the list L in polynomial time by running $\mathcal{B}$ once for each value of k (which is at most $|V(G)|$). By how we sorted L, we compute $\mathcal{A}(G_k, b_k)$ only when $b_k \leq b_{k^*}$, as we argued above that if the algorithm has not already terminated, it does so after reaching $k^* = \mathrm{OPT}_{\mathcal{C}}(G)$. Hence the calls to algorithm $\mathcal{A}$ are for values of the budget b_k which satisfy $b_k \leq b_{k^*}$. We bound the latter, as follows.

Since $k^* = \mathrm{OPT}_{\mathcal{C}}(G)$, the set S_{k^*} found by $\mathcal{B}$ is a superset of the set $\mathcal{E}_c(G)$ of all of the c-essential vertices in G (Property G2). This means that we have

$$b_{k^*} = \mathrm{OPT}_{\mathcal{C}}(G) - |S_{k^*}| \leq \mathrm{OPT}_{\mathcal{C}}(G) - |\mathcal{E}_c(G)| = \ell,$$

so the parameter of each call to $\mathcal{A}$ is at most ℓ, giving the total time bound $f(\ell) \cdot |V(G)|^{\mathcal{O}(1)}$. ◀

Theorem 1.1 now follows from Theorem 5.1 via the algorithms for c-Essential detection given in the previous sections and the state-of-the-art algorithms for the natural parameterizations listed in Table 1. Although the latter may be originally stated for the decision version, using self-reduction they can easily be adapted to output a minimum solution if there is one of size at most k.

6 Hardness results

Given the positive results we saw in Sections 3 and 4, it is natural to seek problems Π for which c-Essential detection for Π is intractable. Here we show that c-Essential detection for Dominating Set is intractable for any $c \in \mathcal{O}(1)$ and then use this as a starting point to prove similar results for Hitting Set, Perfect Deletion, and Wheel-free Deletion.

A dominating set is a vertex set whose closed neighborhood is the entire graph. The domination number of a graph is the size of a minimum dominating set. The starting point for our reductions is the following result which states that it is $W[1]$-hard to solve Dominating Set parameterized by solution size even on instances which have "solution-size gaps".

▶ **Lemma 6.1** ([40], cf. [19, Thm. 4]). *Let $F, f \colon \mathbb{N} \to \mathbb{N}$ be any computable functions. Assuming* FPT $\neq$ W[1]*, there does not exist an algorithm that, given a graph G and integer k, runs in time $f(k) \cdot |V(G)|^{\mathcal{O}(1)}$ and distinguishes between the following two cases:*
- *Completeness: G has a dominating set of size k.*
- *Soundness: Every dominating set of G is of size at least $k \cdot F(k)$.*

All of our reductions in this section share a leitmotif. We start with a *gap* instance (G, k) of Dominating Set and map it to an instance G' of c-Essential detection for Π (for appropriate Π) equipped with a distinguished vertex v^* with the following property: (1) if G has domination number at most k, then no optimal solution in G' contains v^*; (2) if G has domination number strictly greater than $c \cdot F(k)$ (for some appropriate F), then v^* is contained in every solution of size at most $c \cdot F(k)$ in G'. Thus our hardness results will follow by combining reductions of this kind with Lemma 6.1.

▶ **Lemma 6.2** (★). *There is a polynomial-time algorithm R that, given a graph G and integer k, outputs a graph $R(G, k)$ containing a distinguished vertex v^* such that:*
- *if G has dominating number at most k, then the domination number of $R(G, k)$ is exactly k and every optimal dominating set avoids v^*;*
- *if G has domination number strictly greater than $c \cdot (k + 1)$ for some $c \in \mathbb{R}_{\geq 1}$, then $R(G, k)$ has domination number $k + 1$ and the distinguished vertex v^* is contained in all $R(G, k)$-dominating sets of size at most $c \cdot (k + 1)$.*

Proof sketch. The graph $R(G,k)$ is defined formally as follows:

1. initialize $R(G,k)$ as the graph on vertex set $\{v_i \mid v \in V(G), 0 \leq i \leq k\}$ with edges $\{v_i u_j \mid uv \in E(G), 0 \leq i,j \leq k\}$,
2. for each $i \in [k]$ insert an apex a_i which is adjacent to $\{v_i \mid v \in V(G)\}$,
3. insert a vertex v^* which is adjacent to $\{v_i \mid v \in V(G), 0 \leq i \leq k\}$.

The proof that $R(G,k)$ has the desired properties is given in the full version [7], together with a figure of the construction. ◀

Lemma 6.1 combined with the reduction provided by Lemma 6.2 yields the following.

▶ **Theorem 6.3.** *Unless FPT $= W[1]$, there is no FPT-time algorithm for c-*Essential detection for Dominating Set *parameterized by k for any $c \in \mathbb{R}_{\geq 1}$.*

Proof. Suppose such an algorithm $\mathcal{A}$ exists for c; we will use it with Lemma 6.1 to show FPT $= W[1]$ for the function $F(k) = c(k+1)$.

Given an input (G,k) in which the goal is to distinguish between the completeness and soundness cases, the algorithm proceeds as follows. Using the reduction R of Lemma 6.2, consider the graph $R(G,k)$ and let S be the output of an algorithm for c-Essential detection for Dominating Set on the pair $(R(G,k), k+1)$; note that the solution size for which we ask is $k+1$ rather than k. Without loss of generality we may assume $k \geq 2$, as the distinction can trivially be made otherwise. We will show that in the completeness case we have $v^* \notin S$, while in the soundness case we have $v^* \in S$, which allows us to distinguish between these cases by checking whether v^* belongs to the output of $\mathcal{A}(R(G,k), k+1)$.

For the completeness case, suppose G has domination number at most k. Then, by Lemma 6.2, so does $R(G,k)$. This means that Property G1 holds for the call to $\mathcal{A}(R(G,k), k+1)$, so that there is some optimal solution S' of size k which contains S and hence we have $v^* \notin S$ by Lemma 6.2.

For the soundness case, suppose G has domination number at least $k \cdot F(k) = c(k+1)k > c(k+1)$ (we use $k \geq 2$ here). Then by Lemma 6.2, graph $R(G,k)$ has domination number $k+1$ and v^* is contained in all its dominating sets of size at most $c(k+1)$. In other words: v^* is c-essential in $R(G,k)$. Consequently, $v^* \in S$ by Property G2 since the argument $k+1$ we supplied to $\mathcal{A}$ coincides with the optimum in $R(G,k)$ in this case.

If $\mathcal{A}$ runs in FPT-time, then the overall procedure runs in FPT-time which implies FPT $= W[1]$ by Lemma 6.1. ◀

Combining Lemma 6.2 with the standard reduction S from Dominating Set to Hitting Set (where hyperedges are given by closed neighborhoods) yields a composite mapping $S \circ R$ which relates c-essentiality to gaps in solution quality in much the same way as R did. We leverage this together with known reductions from Hitting Set to Perfect Deletion [26] and Wheel-free Deletion [32] to show the following.

▶ **Theorem 6.4 (★).** *Unless FPT $= W[1]$, both c-*Essential detection for PerfDel *and c-*Essential detection for WheelDel *do not admit FPT-time algorithms parameterized by k for any $c \in \mathbb{R}_{\geq 1}$.*

7 Connections to Perturbation Resilience

In the area of perturbation resilience [36] there is a notion of so-called *c-perturbation resilient instances* to optimization problems. Roughly, these are instances G in which there is a *unique* optimal solution S which far outperforms (by a factor of c) every other solution S' in G.

More formally, for a vertex-weighted graph minimization problem Π whose solution is a vertex-subset, we say that an instance $(G, w \colon V(G) \to \mathbb{N})$ with a *unique* optimal solution S is c-*perturbation resilient* if for any perturbed weight function w' satisfying $w(v) \leq w'(v) \leq c \cdot w(v)$ for all $v \in V(G)$, the instance $(G, w'(x))$ has a unique optimal solution and furthermore this solution is S. (Of course, one can define an analogous notion for maximization problems as well and what follows in this section applies to both.)

Classes of c-perturbation resilient instances have been shown to be "islands of tractability" where many intractable problems become polynomial-time-solvable [36] and the suggestive intuition behind this fact is that perturbation resilient instances have unique optima which "stand out" and are "obvious" in some sense. Viewing stability through the lens of parameterized complexity, it is natural to ask whether one can quantify in an algorithmically useful way how distant an input is from being stable. The following proposition supports our claim that the *non-essentiality* (recall Theorem 5.1) is a good such measure since on $(> c)$-stable inputs, the c-non-essentiality is smallest possible (namely it is 0).

▶ **Proposition 7.1.** *Given constants $c' > c \geq 1$, if G is a c'-stable input to a graph optimization problem Π whose solutions are vertex-subsets, then $\mathcal{E}_c^{\Pi}(G)$ is the unique optimum of G.*

Proof. Consider the unique optimum S for Π on G. If S' is a c-approximation for Π on G, then we must have $S' = S$ (since otherwise G would not be c'-stable). As a consequence we know that every vertex of S must be c-essential and hence we have $S \subseteq \mathcal{E}_c(G) \subseteq S$. ◀

Proposition 7.1 and Theorem 5.1 allow us to immediately deduce that any algorithm for c-Essential detection solves all $(> c)$-perturbation resilient instances exactly.

▶ **Corollary 7.2.** *Given a minimization problem Π, any algorithm for c-Essential detection for Π solves $(> c)$-stable instances exactly.*

8 Conclusion

We introduced the notion of c-essential vertices for vertex-subset minimization problems on graphs, to formalize the idea that a vertex belongs to all *reasonable* solutions. Using a variety of approaches centered around the theme of covering/packing duality, we gave polynomial-time algorithms that detect a subset of an optimal solution containing all c-essential vertices, which decreased the search space of parameterized algorithms from exponential in the size of the solution, to exponential in the number of non-essential vertices in the solution.

Throughout the paper we have restricted ourselves to working with unweighted problems. However, many of the same ideas can be applied in the setting where each vertex has a positive integer weight of magnitude $\mathcal{O}(|V(G)|^{\mathcal{O}(1)})$ and we search for a minimum-weight solution. Since integral vertex weights can be simulated for many problems by making twin-copies of a vertex, our approach can be extended to Weighted Vertex Cover, Weighted Odd Cycle Transversal, and Weighted Chordal Deletion.

Our results shed a new light on which instances of NP-hard problems can be solved efficiently. FPT algorithms for parameterizations by solution size show that instances are easy when their optimal solutions are small. Theorem 1.1 refines this view: it shows that instances with large optimal solutions can still be easy, as long as only a small number of vertices in the optimum is not c-essential.

We remark that there is an alternative route to algorithms for c-Essential detection, which is applicable to $\mathcal{C}$-Deletion problems which admit a constant-factor approximation. If there is a polynomial-time algorithm that, given a graph G and vertex v, outputs a

c-approximation for the problem of finding a minimum-size $\mathcal{C}$-modulator avoiding v, it can be used for c-Essential detection. A valid output S for the detection problem with input (G, k) is obtained by letting S contain all vertices for which the approximation algorithm outputs a v-avoiding modulator of size more than $c \cdot k$. Using this approach (cf. [22]) one can solve $\max_{F \in \mathcal{F}} |V(F)|^{\mathcal{O}(1)}$-Essential detection for $\mathcal{F}$-Minor-Free Deletion for any finite family $\mathcal{F}$ containing a planar graph. As the results for problems for which no constant-factor approximation exists are more interesting, we focused on those.

Our work opens up several questions for future work. Is the integrality gap for v-Avoiding Planar Vertex Deletion constant, when $G - v$ is planar? Can $\mathcal{O}(1)$-Essential detection for Planar Vertex Deletion be solved in polynomial time? Can 2-Essential detection for Chordal Deletion be solved in polynomial time? Can the constant c for which we can detect c-essential vertices be lowered below 2?

Considering a broader horizon, it would be interesting to investigate whether there are less restrictive notions than c-essentiality which can be used as the basis for guaranteed search-space reduction.

References

1 Tobias Achterberg, Robert E. Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. Presolve reductions in mixed integer programming. Technical Report 16-44, ZIB, Takustr.7, 14195 Berlin, 2016. URL: `http://nbn-resolving.de/urn:nbn:de:0297-zib-60370`.

2 Takuya Akiba and Yoichi Iwata. Branch-and-reduce exponential/FPT algorithms in practice: A case study of vertex cover. *Theor. Comput. Sci.*, 609:211–225, 2016. `doi:10.1016/j.tcs.2015.09.023`.

3 Jochen Alber, Nadja Betzler, and Rolf Niedermeier. Experiments on data reduction for optimal domination in networks. *Annals OR*, 146(1):105–117, 2006. `doi:10.1007/s10479-006-0045-4`.

4 Hans L. Bodlaender. Lower bounds for kernelization. In Marek Cygan and Pinar Heggernes, editors, *Parameterized and Exact Computation - 9th International Symposium, IPEC 2014, Wroclaw, Poland, September 10-12, 2014. Revised Selected Papers*, volume 8894 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2014. `doi:10.1007/978-3-319-13524-3_1`.

5 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 629–638. IEEE Computer Society, 2009. `doi:10.1109/FOCS.2009.46`.

6 Hans L. Bodlaender and Thomas C. van Dijk. A cubic kernel for feedback vertex set and loop cutset. *Theory Comput. Syst.*, 46(3):566–597, 2010. `doi:10.1007/s00224-009-9234-2`.

7 Benjamin Merlin Bumpus, Bart M. P. Jansen, and Jari J. H. de Kroon. Search-space reduction via essential vertices. *CoRR*, abs/2207.00386, 2022. `arXiv:2207.00386`.

8 Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016. `doi:10.1007/s00453-015-0014-x`.

9 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010. `doi:10.1016/j.tcs.2010.06.026`.

10 Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5):21:1–21:19, 2008. `doi:10.1145/1411509.1411511`.

11 Maria Chudnovsky, Jim Geelen, Bert Gerards, Luis A. Goddyn, Michael Lohman, and Paul D. Seymour. Packing non-zero A-paths in group-labelled graphs. *Comb.*, 26(5):521–532, 2006. `doi:10.1007/s00493-006-0030-1`.

12 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. `doi:10.1007/978-3-319-21275-3`.

13 Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM*, 61(4):23:1–23:27, 2014. doi:10.1145/2629620.

14 Huib Donkers and Bart M. P. Jansen. Preprocessing to reduce the search space: Antler structures for feedback vertex set. In Lukasz Kowalik, Michal Pilipczuk, and Pawel Rzazewski, editors, *Graph-Theoretic Concepts in Computer Science - 47th International Workshop, WG 2021, Warsaw, Poland, June 23-25, 2021, Revised Selected Papers*, volume 12911 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2021. doi:10.1007/978-3-030-86838-3_1.

15 Rodney G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer Publishing Company, Incorporated, 2012.

16 Andrew Drucker. New limits to classical and quantum instance compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015. doi:10.1137/130927115.

17 Eduard Eiben, Robert Ganian, Thekla Hamm, and O-joung Kwon. Measuring what matters: A hybrid approach to dynamic programming with treewidth. *J. Comput. Syst. Sci.*, 121:57–75, 2021. doi:10.1016/j.jcss.2021.04.005.

18 P. Erdös and L. Pósa. On independent circuits contained in a graph. *Canadian Journal of Mathematics*, 17:347–352, 1965. doi:10.4153/CJM-1965-035-8.

19 Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. doi:10.3390/a13060146.

20 Michael R. Fellows. The lost continent of polynomial time: Preprocessing and kernelization. In *Proc. 2nd IWPEC*, pages 276–277, 2006. doi:10.1007/11847250_25.

21 Fedor Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing.* Cambridge University Press, 2019.

22 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-deletion: Approximation, kernelization and optimal FPT algorithms. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 470–479. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.62.

23 Jim Geelen, Bert Gerards, Bruce A. Reed, Paul D. Seymour, and Adrian Vetta. On the odd-minor variant of Hadwiger's conjecture. *J. Comb. Theory, Ser. B*, 99(1):20–29, 2009. doi:10.1016/j.jctb.2008.03.006.

24 M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* ISSN. Elsevier Science, 2004.

25 Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007. doi:10.1145/1233481.1233493.

26 Pinar Heggernes, Pim van 't Hof, Bart M. P. Jansen, Stefan Kratsch, and Yngve Villanger. Parameterized complexity of vertex deletion into perfect graph classes. *Theor. Comput. Sci.*, 511:172–180, 2013. doi:10.1016/j.tcs.2012.03.013.

27 Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Wlodarczyk. Vertex deletion parameterized by elimination distance and even less. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1757–1769. ACM, 2021. doi:10.1145/3406325.3451068.

28 Subhash Khot. On the power of unique 2-prover 1-round games. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 767–775. ACM, 2002. doi:10.1145/509907.510017.

29 Stefan Kratsch. Recent developments in kernelization: A survey. *Bull. EATCS*, 113, 2014. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/285.

30 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020. doi:10.1145/3390887.

31 Jason Li and Jesper Nederlof. Detecting feedback vertex sets of size k in $O^*(2.7^k)$ time. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 971–989. SIAM, 2020. doi:10.1137/1.9781611975994.58.

32 Daniel Lokshtanov. Wheel-free deletion is W[2]-hard. In Martin Grohe and Rolf Niedermeier, editors, *Parameterized and Exact Computation, Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings*, volume 5018 of *Lecture Notes in Computer Science*, pages 141–147. Springer, 2008. `doi:10.1007/978-3-540-79723-4_14`.

33 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization - Preprocessing with a guarantee. In Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 129–161. Springer, 2012. `doi:10.1007/978-3-642-30891-8_10`.

34 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014. `doi:10.1145/2566616`.

35 Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2181–2200. SIAM, 2020. `doi:10.1137/1.9781611975994.134`.

36 Konstantin Makarychev and Yury Makarychev. Perturbation resilience. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*, pages 95–119. Cambridge University Press, 2020. `doi:10.1017/9781108637435.008`.

37 George L. Nemhauser and Leslie E. Trotter Jr. Vertex packings: Structural properties and algorithms. *Math. Program.*, 8(1):232–248, 1975. `doi:10.1007/BF01580444`.

38 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. `doi:10.1016/j.orl.2003.10.009`.

39 Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. `doi:10.1006/jctb.1995.1006`.

40 Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019. `doi:10.1145/3325116`.

41 A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

42 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986. `doi:10.1016/0304-3975(86)90135-0`.

43 Karsten Weihe. Covering trains by stations or the power of data reduction. In *Algorithms and Experiments (ALEX98)*, pages 1–8, 1998. URL: `https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.57.2173`.

44 Karsten Weihe. On the differences between “Practical” and “Applied”. In Stefan Näher and Dorothea Wagner, editors, *Algorithm Engineering, 4th International Workshop, WAE 2000, Saarbrücken, Germany, September 5-8, 2000, Proceedings*, volume 1982 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2000. `doi:10.1007/3-540-44691-5_1`.

45 Sebastian Wernicke. *On the algorithmic tractability of single nucleotide polymorphism (SNP) analysis and related problems.* diplom.de, 2014.