Further Exploiting c-Closure for FPT Algorithms and Kernels for Domination Problems

Lawqueen Kanesh ✉
National University of Singapore, Singapore

Jayakrishnan Madathil ✉
Chennai Mathematical Institute, India

Sanjukta Roy ✉
Algorithms and Complexity Group, TU Wien, Austria

Abhishek Sahu ✉
The Institute of Mathematical Sciences, HBNI, Chennai, India

Saket Saurabh ✉
The Institute of Mathematical Sciences, HBNI, Chennai, India
University of Bergen, Norway

Abstract

For a positive integer c, a graph G is said to be c-closed if every pair of non-adjacent vertices in G have at most $c-1$ neighbours in common. The closure of a graph G, denoted by $cl(G)$, is the least positive integer c for which G is c-closed. The class of c-closed graphs was introduced by Fox et al. [ICALP '18 and SICOMP '20]. Koana et al. [ESA '20] started the study of using $cl(G)$ as an additional structural parameter to design kernels for problems that are W-hard under standard parameterizations. In particular, they studied problems such as Independent Set, Induced Matching, Irredundant Set and (Threshold) Dominating Set, and showed that each of these problems admits a polynomial kernel, either w.r.t. the parameter $k+c$ or w.r.t. the parameter k for each fixed value of c. Here, k is the solution size and $c = cl(G)$. The work of Koana et al. left several questions open, one of which was whether the Perfect Code problem admits a fixed-parameter tractable (FPT) algorithm and a polynomial kernel on c-closed graphs. In this paper, among other results, we answer this question in the affirmative. Inspired by the FPT algorithm for Perfect Code, we further explore two more domination problems on the graphs of bounded closure. The other problems that we study are Connected Dominating Set and Partial Dominating Set. We show that Perfect Code and Connected Dominating Set are fixed-parameter tractable w.r.t. the parameter $k + cl(G)$, whereas Partial Dominating Set, parameterized by k is W[1]-hard even when $cl(G) = 2$. We also show that for each fixed c, Perfect Code admits a polynomial kernel on the class of c-closed graphs. And we observe that Connected Dominating Set has no polynomial kernel even on 2-closed graphs, unless NP $\subseteq$ co-NP/poly.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases c-closed graphs, domination problems, perfect code, connected dominating set, fixed-parameter tractable, polynomial kernel

Digital Object Identifier 10.4230/LIPIcs.STACS.2022.39

Funding *Lawqueen Kanesh*: Supported in part by NRF Fellowship for AI grant [R-252-000-B14-281] and by Defense Service Organization, Singapore.
Jayakrishnan Madathil: Supported by Chennai Mathematical Institute and the Infosys Foundation.
Sanjukta Roy: Supported by the WWTF research grant VRG18012.
Saket Saurabh: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416), and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

Acknowledgements We thank two anonymous reviewers for their suggestions that helped improve the presentation of this article.

39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022).
Editors: Petra Berenbrink and Benjamin Monmege; Article No. 39; pp. 39:1–39:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

For a positive integer c, a graph G is said to be c-closed if every pair of non-adjacent vertices in G have at most $c-1$ neighbours in common. That is, for distinct vertices u and v of G, $|N(u) \cap N(v)| \leq c-1$ if $uv \notin E(G)$. In this paper, we investigate the parameterized complexity of domination problems on the class of c-closed graphs. The problems that we study are Perfect Code, Connected Dominating Set and Partial Dominating Set. All these problems are W[1]-hard (w.r.t. standard parameters) on general graphs [12, 18, 19], and their complexities on various restricted graph classes have been studied extensively [4, 15, 22, 29, 30, 31, 34, 41, 44].

Fox et al. [25, 26] introduced the class of c-closed graphs in 2018 as a "distribution-free" model of social networks. While the literature abounds with models that attempt to capture the structure of social networks, they are all probabilistic models. (See, for instance, the survey by Chakrabarti and Faloutsos [13].) And in an attempt to capture the spirit of "social-network-like" graphs without relying on probabilistic models, Fox et al. [26] "turn[ed] to one of the most agreed upon properties of social networks – triadic closure, the property that when two members of a social network have a friend in common, they are likely to be friends themselves." It is easy to see that the definition of c-closed graphs is a reasoned approximation of this property. In a c-closed graph, every pair of vertices with at least c common neighbours are adjacent to each other. Fox et al. [26, Table A.1], and later Koana et al. [39, Table 1], showed that several social networks and biological networks are indeed c-closed for rather small values of c.

Fox et al. [26] showed that an n-vertex c-closed graph contains at most $3^{c/3} \cdot n^2$ maximal cliques.[1] This bound, when coupled with an algorithm for enumerating all maximal cliques in a graph, yields a $2^{\mathcal{O}(c)} \cdot \text{poly}(n)$ time algorithm that enumerates all maximal cliques in c-closed graphs. Observe that an algorithm that *enumerates all maximal cliques* in a graph can be used to determine if a graph *contains a clique of a given size* as well. Thus, the Clique problem, which, given a graph G and an integer k as input, asks if G contains a clique of size k, is fixed-parameter tractable with respect to the parameter c. Notice that Clique, when parameterized by k, is W[1]-complete on general graphs [18], and therefore does not admit a fixed-parameter tractable algorithm unless FPT=W[1].

In light of this result, we could very well ask: How do other problems that are W-hard on general graphs fare on the class of c-closed graphs? In particular, is Independent Set, another canonical W[1]-complete problem [18], fixed-parameter tractable on c-closed graphs? Koana et al. [39] showed that Independent Set, which takes a graph G and an integer k as input, and asks if G contains an independent set of size k, is indeed fixed-parameter tractable w.r.t. the parameter $k+c$. In fact, by applying a "Buss-like" reduction rule [9], they showed that the problem admits a kernel with ck^2 vertices. Motivated by this example, they studied the (kernelization) complexity of three more problems – Induced Matching, Irredundant Set and Threshold Dominating Set (TDS) – and showed that these problems admit polynomial kernels (either w.r.t. the parameter $k+c$, or w.r.t. the parameter k for each fixed c.) TDS is a variant of Dominating Set in which each vertex needs to be dominated at least r times for a given integer r. The kernels for the first two of these problems have size $\text{poly}(c,k)$ whereas the kernel for TDS has size $k^{\mathcal{O}(cr)}$. They also designed an FPT algorithm for TDS that runs in time $3^{c/3} + (ck)^{\mathcal{O}(rk)} n^{\mathcal{O}(1)}$. A key ingredient in all

[1] Note that the classic Moon-Moser theorem only guarantees an upper bound of $3^{n/3}$ for the number of maximal cliques in an n-vertex graph [47].

these results was a polynomial bound for the Ramsey number on c-closed graphs. Koana et al. [39] proved that every c-closed graph with $\mathcal{O}(cb^2 + ab)$ vertices contains either a clique of size a or an independent set of size b, and predicted that this bound could be useful in settling the parameterized complexity of other problems as well. In this paper, we use this bound, and show that two variants of Dominating Set admit fixed-parameter tractable algorithms on c-closed graphs. In particular, we show that Perfect Code is FPT on c-closed graphs, and thus settle a question left open by Koana et al. [39].

Closure of a graph. Recall that a graph G is said to be c-closed if every pair of non-adjacent vertices have at most $c-1$ neighbours in common. The *closure*[2] of a graph G, denoted by $cl(G)$, is the least positive integer c for which G is c-closed. Notice that $cl(G) = 1 + \max\{|N(u) \cap N(v)| \mid u, v \in V(G), uv \notin E(G)\}$, and therefore $cl(G)$ can be computed in polynomial time. In this paper, we study the parameterized complexity of some of the widely-studied problems on graphs of bounded closure, and thus attempt to present a more comprehensive answer to the following questions. How good a structural parameter is $cl(G)$ when it comes to the tractability of domination problems? And in this regard, how does $cl(G)$ differ from some of the other widely-studied structural parameters such as maximum degree, degeneracy and treewidth? Observe that if the maximum degree of graph G is $\Delta(G)$, then $cl(G) \le \Delta(G) + 1$. But the comparability ends there. As noted by Koana et al. [39], an n-vertex clique is 1-closed, but has degeneracy and treewidth $n-1$. On the other hand, the complete bipartite graph $K_{2,n-2}$ has treewidth and degeneracy 2, but $cl(K_{2,n-2}) = n-1$. Thus, closure is incomparable with degeneracy and treewidth. We also note that when parameterized by $cl(G)$ alone, most of the widely-studied problems, with the exception of Clique, would be para-NP-hard. This applies to problems such as Vertex Cover, Independent Set, Dominating Set, Connected Dominating Set and Perfect Code, as all these problems are NP-hard on graphs of maximum degree 4 [21, 27], and therefore NP-hard on 5-closed graphs. So this parameter alone is too small to yield tractability results, and therefore, has to be used in combination with some other parameter, for example, the solution size. But this is often the case with other structural parameters such as degeneracy and maximum degree as well; they are often combined with the solution size [3, 48].

Our results and methods. Let us first define the concept of domination in graphs. Consider a graph G. We say that a vertex in G dominates itself and all its neighbours. That is, a vertex v dominates $N[v]$. And for a set $V' \subseteq V(G)$, V' dominates $N[V']$. A *dominating set* of a graph is a set of vertices $D \subseteq V(G)$ that dominates the entire vertex set, i.e., $N[D] = V(G)$. Or equivalently, $D \subseteq V(G)$ is a dominating set of G if $|D \cap N[v]| \ge 1$ for every vertex $v \in V(G)$. A dominating set $D \subseteq V(G)$ is said to be a *connected dominating set* of G if $G[D]$ is a connected subgraph of G. A *perfect code* of G is a dominating set of G that dominates every vertex exactly once. That is, $D \subseteq V(G)$ is a perfect code of G if $|D \cap N[v]| = 1$ for every vertex $v \in V(G)$. For a non-negative integer t, a set of vertices $V' \subseteq V(G)$ is said to be a *t-partial dominating set* of G if V' dominates at least t vertices of G, i.e., if $|N[V']| \ge t$.

[2] Koana et al. [39] use the term c-closure instead of closure. But we believe that closure is more appropriate. We must note that the term closure is already used in existing graph theory literature to refer to a certain super-graph of a graph [8, p. 486]. But for that matter, so is the term k-closure [7]. We believe that given the context, there is no room for ambiguity.

In the Perfect Code (resp. Connected Dominating Set (CDS)) problem, the input consists of an n-vertex graph G and a non-negative integer k, and the question is to decide if G contains a perfect code (resp. connected dominating set) of size at most k. In the Partial Dominating Set (PDS) problem, the input consists of an n-vertex graph G and two non-negative integers k and t, and the question is to decide if G contains a t-partial dominating set of size at most k. We show that Perfect Code and CDS, when parameterized by $k + cl(G)$, are fixed-parameter tractable, whereas PDS, when parameterized by k, is W[1]-hard, even for $cl(G) = 2$. Specifically, we prove the following results. (Here, $n = |V(G)|$ and $c = cl(G)$.)

1. Perfect Code admits a fixed-parameter tractable algorithm that runs in time $2^{\mathcal{O}(c+k\log(ck))}n^{\mathcal{O}(1)}$. Moreover, for each fixed $c \geq 1$, Perfect Code admits a kernel with $k^{\mathcal{O}(2^c)}$ vertices on the family of c-closed graphs.
2. CDS admits a fixed-parameter tractable algorithm that runs in time $2^{\mathcal{O}(ck^2\log(ck))}n^{\mathcal{O}(1)}$. But CDS does not admit a polynomial kernel when parameterized by k even when $cl(G) = 2$, unless NP $\subseteq$ co-NP/poly. (The kernelization lower bound follows from a result due to Misra et al. [44].)
3. PDS, when parameterized by k, is W[1]-hard on 2-closed graphs.

Note that a perfect code and a connected dominating set are both dominating sets. Naturally, our algorithms for Perfect Code and CDS rely on three crucial properties of dominating sets and c-closed graphs. Consider a c-closed graph G, and a dominating set D of G of size k. **(P1)** If G contains an independent set I of size $k + 1$, then by the pigeonhole principle, there exists a vertex $v \in D$ that dominates at least two vertices of I. That is, $v \in N(u) \cap N(u')$ for a pair of vertices $u, u' \in I$ (Lemma 11). **(P2)** The dominating set D must intersect every "large" maximal clique (Corollary 7). This follows from the fact that any vertex outside a maximal clique can dominate at most $c - 1$ vertices of the clique (Lemma 6). Thus, if G contains a maximal clique of size $(c - 1)k + 1$, say Q, then we must have $D \cap V(Q) \neq \emptyset$. **(P3)** If G contains ℓ distinct "large" maximal cliques, then G contains an independent set of size ℓ as well (Lemma 8). This again is a consequence of the property that any vertex outside a maximal clique has at most $c - 1$ neighbours in the clique. Here, depending on each problem, we will define an appropriate lower bound on the size of a clique for it to be large. But in both the problems, this bound will be poly(c, k). Finally, we use the following two results due to Koana et al. [39]. **(R1)** Every c-closed graph with $\mathcal{O}(cb^2 + ab)$ vertices contains either a clique of size a or an independent set of size b (Lemma 1). **(R2)** We can find a $(k + 1)$-sized independent set of an n-vertex c-closed graph, if it exists, or correctly conclude that no such set exists, in time $2^{\mathcal{O}(k\log(ck))}n^{\mathcal{O}(1)}$ (Corollary 4).

We now briefly outline how our algorithms exploit these properties. In light of (P1), we first find an independent set I of size $k + 1$ using (R2), and branch on the vertices in $\bigcup_{u,u' \in I} N(u) \cap N(u')$. Note that since $|I| = k + 1$, we have $\binom{k+1}{2} = \mathcal{O}(k^2)$ choices for the pair $\{u, u'\}$. And for each pair $u, u' \in I$, we have $|N(u) \cap N(u')| \leq c - 1$ as G is c-closed. Once this branching step is exhaustively applied, every independent set in G has size at most k. But then (P3) will imply that G contains at most k "large" maximal cliques. Now we partition the vertex set of G into two parts, L and R, where L is the set of vertices that belong to at least one large maximal clique and R the set of remaining vertices. Thus, L is the union (not necessarily disjoint) of at most k large cliques. And the subgraph $G[R]$ contains no large clique or no independent set of size $k + 1$. Therefore, by (R1), we will have $|R| =$ poly(c, k). So we can guess the set of vertices from R that belongs to the "dominating set" that we are looking for, in case (G, k) is indeed a yes-instance. And corresponding to each such guess, we then use the property that L is a union of cliques to solve the problem appropriately. For example, in the case of Perfect Code, we show that once we guess the subset of R that belongs to the solution, the problem then reduces to solving an instance of the d-Exact Hitting Set problem (a variant of Hitting Set in which every set has size at most d and needs to be hit exactly once) for an appropriate choice of d, which can then be solved in time $d^k n^{\mathcal{O}(1)}$. In the case of CDS, we reduce the final step to $2^{\text{poly}(c,k)}$ many instances of the (edge-weighted) Steiner Tree problem, a common technique used in algorithms that seek connected solutions [32, 44, 45, 46]. And we will have the guarantee that our original CDS instance is a yes-instance if and only if at least one of the Steiner Tree instances is a yes-instance. We prove the W-hardness of PDS by designing a parameterized reduction from the Independent Set problem on regular graphs, which is known to be W[1]-complete [10]. The inadmissibility of a polynomial kernel for CDS follows from a result due to Misra et al. [44], which says that CDS admits no polynomial kernel on graphs of girth 5, and the fact that graphs of girth 5 are 2-closed.

To design our kernel for Perfect Code, we bound the size of independent sets and cliques in the input graph by $k^{\mathcal{O}(2^c)}$, and then invoke (R1). The main ingredient in bounding the independent set size is a reduction rule, by which we find a sufficiently large independent set with sufficiently many common neighbours and delete an arbitrary vertex from that independent set. To find this independent set, we design an algorithm that works as follows: Given a c-closed graph G and an integer k, the algorithm will either output an independent set of size k or correctly report that every independent set in G has size $\text{poly}(c,k)$ (Lemma 10). After an exhaustive application of this reduction rule, every independent set in the input graph will have bounded size, and by (P3), the graph will contain only a bounded number of large cliques. Then, we bound the size of each clique as well, which, by (R1), will result in the kernel. We note that our fixed-parameter tractable algorithm and polynomial kernel for Perfect Code do not imply each other. The kernel runs in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(c)}$, and therefore, does not imply a fixed-parameter tractable algorithm w.r.t the parameter $k + c$.

We must point out that properties (P1) and (P2) have been used by Koana et al. [39] in their algorithm and kernel for the TDS problem. But these properties alone are inadequate for Perfect Code and CDS. Hence we introduce (P3), which bounds the number of large maximal cliques in terms of the maximum size of an independent set. We also note that while properties (P1) and (P2) are specific to domination problems, (P3) is a general-purpose bound. Our strategy of partitioning the vertices into L and R (vertices of large cliques and the remaining vertices) is also not specific to domination problems, and could be applicable to other problems as well. So is Lemma 10, which, as mentioned above, gives an algorithm that either outputs an independent set of size k or guarantees an upper bound of $\text{poly}(c,k)$ on the independent set size. We use Lemma 10 to fashion a reduction rule (Reduction Rule 19), which we use to bound the size of independent sets while designing our kernel for Perfect Code. The idea behind Reduction Rule 19 is as follows. To bound the size of any independent in the graph, it is sufficient to bound the size of independent sets within the induced subgraph $G[N(v)]$ for every $v \in V(G)$. Then, to bound the size of independent sets in $G[N(v)]$, it is sufficient to bound the size of independent sets in $G[N(v) \cap N(u)]$ for every $u \in V(G) \setminus \{v\}$. And to bound the size of independent sets in $G[N(v) \cap N(u)]$, it is sufficient to bound the size of independent sets in $G[N(v) \cap N(u) \cap N(w)]$ for every $w \in V(G) \setminus \{v, u\}$ and so on. This strategy of successively bounding the independent sets in stages could be applicable to other problems on c-closed graphs as well. Since G is c-closed, we only need to continue for $c - 1$ stages. That is, we only need to bound the size of independent sets in $G[\cap_{x \in Y} N(x)]$ for all $Y \subseteq V(G)$ with $|Y| = c - 1$.

Related work on domination problems. Domination problems have long been the subject of extensive research in algorithmic graph theory. All the domination problems discussed above are W-hard on general graphs, when parameterized by the solution size. Therefore, a great deal of effort has gone into studying the complexity of these problems on various graph classes. In particular, the classic Dominating Set problem is known to be W[2]-complete [19] on general graphs, and W[2]-hard even on bipartite graphs (and hence on triangle-free graphs) [49], but admits a fixed-parameter tractable algorithm on graphs of girth at least 5 [49], planar graphs [1, 2, 24, 35], graphs of bounded genus [20], map graphs [16], H-minor free graphs [17] and graphs of bounded degeneracy [3]. The CDS problem is also known to be W[2]-hard on general graphs [19], but admits a polynomial kernel on planar graphs, and more generally, on apex-minor-free graphs [22, 30, 41]. The problem is FPT on graphs of bounded degeneracy [29]. Cygan et al. [14] showed that CDS has no polynomial kernel even on 2-degenerate graphs unless NP $\subseteq$ co-NP/poly. Misra et al. [44] studied the effect of the girth of the input graph on the complexity of CDS, and showed that CDS remains W[1]-hard on graphs of girth 3 and 4, admits a fixed-parameter tractable algorithm but no polynomial kernel (unless NP $\subseteq$ co-NP/poly) on graphs of girth 5 and 6, and admits a polynomial kernel on graphs of girth at least 7. Fomin et al. [23] showed that both Dominating Set and CDS admit linear kernels on graphs with excluded topological minors. We refer the reader to [23] for a historical overview of the literature on these problems.

The Perfect Code problem, also called Efficient Domination or Perfect Domination, is known to be W[1]-complete [12, 18], and remains W[1]-hard even on bipartite graphs of girth 4 [34], but admits a polynomial kernel on planar graphs [31] and graphs of girth at least 5 [34]. Dawar and Kreutzer [15] showed that Perfect Code is fixed-parameter tractable on effectively nowhere dense graphs. For a summary of results on the (classical) complexity of Perfect Code on various graph classes, see [43].

The Partial Vertex Cover (PVC) problem, the "partial variant" of the widely-studied Vertex Cover problem, asks if t edges of a graph can be covered using k vertices. Both PVC and PDS have been studied w.r.t. the two natural parameters: k and t. When parameterized by k, unlike the widely-studied Vertex Cover, PVC is W[1]-hard on general graphs [32], and remains NP-hard even on bipartite graphs [5]. But Amini et al. [4], using a nuanced branching strategy called implicit branching, showed that PVC admits fixed-parameter tractable algorithms on graph classes with "large independent sets." In particular, they showed that PVC (parameterized by k) is FPT on bipartite graphs, triangle-free graphs, and H-minor free graphs, and thus, in particular, on planar graphs and graphs of bounded genus. As for PDS, note that a PDS instance with $t = n$ is precisely the Dominating Set problem, and therefore, the $W[2]$-hardness of Dominating Set (w.r.t. the parameter k) extends to PDS as well. And in contrast to Dominating Set, PDS remains W[1]-hard even on graphs of bounded degeneracy [29]. But the results due to Amini et al. [4] for a more general problem called Weighted Partial-(k, r, t)-Center showed that PDS, in particular, is FPT on planar graphs, graphs of bounded genus and graphs of bounded maximum degree. When parameterized by t, both PVC and PDS are FPTon general graphs [6, 11, 36, 37].

Related work on c-closed graphs. As mentioned earlier, Fox et al. [26] showed that every n-vertex c-closed graph contains at most $3^{c/3} \cdot n^2$ maximal cliques, and that all maximal cliques can be enumerated in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$. In a preprint announced in 2020, Husic and Roughgarden [33] showed that instead of cliques, other "dense subgraphs" can be enumerated in time $f(c) \cdot \text{poly}(n)$ as well. In particular, they showed that the problems of finding and enumerating subgraphs of bounded co-degree, bounded co-degeneracy and bounded co-treewidth in a c-closed graph admit algorithms that run in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. This result was soon followed by the work of Koana and Nichterlein [40], who investigated the complexity of enumerating all copies of a (small) fixed-graph H in a given c-closed graph. Note that for each fixed graph H, by brute-force, we can detect and enumerate all copies of H in a given n-vertex graph in time $n^{\mathcal{O}(|V(H)|)}$. Nonetheless, Koana and Nichterlein [40] designed significantly better combinatorial algorithms for such problems. They showed that for small graphs (i.e., graphs on 3 or 4 vertices) H, the H-detection and enumeration problems admit "FPT in P" algorithms [28] w.r.t. the parameter c, i.e., algorithms with runtime $\mathcal{O}(c^\ell n^i m^j)$ or $\mathcal{O}(c^\ell n^i + m^j)$, where m and n respectively are the number of edges and vertices of the input graph G, $c = cl(G)$, and ℓ, i and j are small constants independent of c and H. In particular, they designed such algorithms for 11 out of the 15 graphs on 3 or 4 vertices.

Related work on weakly γ-closed graphs. Along with c-closed graphs, Fox et al. [26] had also introduced a larger class of graphs called weakly γ-closed graphs. For a positive integer γ, a graph G is weakly γ-closed if every induced subgraph G' of G has a vertex v such that $|N_{G'}(v) \cap N_{G'}(u)| < \gamma$ for each $u \in V(G')$ with $u \neq v$ and $uv \notin E(G')$. Note that if a graph G is c-closed, then G is weakly c-closed as well. In a subsequent work, Koana et al. [38] extended their result for Independent Set in [39] to weakly γ-closed graphs. They showed that Independent Set admits a polynomial kernel on weakly γ-closed graphs as well. And they showed that a similar result holds for the $\mathcal{G}$-Subgraph problem, for a fixed family of graphs $\mathcal{G}$ that is closed under subgraphs, where the goal is to check if a given graph G contains an induced subgraph on at least k vertices that belongs to $\mathcal{G}$. Notice that Independent Set is a special case of $\mathcal{G}$-Subgraph with $\mathcal{G}$ being the family of all edgeless graphs. Koana et al. [38] also showed that two variants of Dominating Set, namely, Independent Dominating Set and Dominating Clique, are FPT on weakly γ-closed graphs. But they left open the complexity of Dominating Set on weakly γ-closed graphs, which was recently shown to be FPT by Lokshtanov and Surianarayanan [42]. Koana et al. [38] also gave bounds and enumeration algorithms for various choices of "dense subgraphs" in weakly γ-closed subgraphs. See [38, Table 1] for an overview of their results.

Due to space constraints, we only present our kernel for Perfect Code here. We omit other results and the proofs of statements marked with a ♣.

2 Preliminaries

For a positive integer ℓ, we denote the set $\{1, \ldots, \ell\}$ by $[\ell]$. We define the functions $\alpha, \beta : \mathbb{N} \to \mathbb{N}$ as follows: $\alpha(a, b) = (a-1)b+1$ and $\beta(a, b) = 2[(a-1)(b-1)+1]$ for every $a, b \in \mathbb{N}$. All graphs in this paper are simple and undirected. For a graph G, $V(G)$ and $E(G)$ respectively denote the vertex set and edge set of G. For a vertex $v \in V(G)$, $N_G(v)$ and $N_G[v]$ respectively denote the open and closed neighbourhood of v in G. Also, $d_G(v)$ denotes the degree of v in G, i.e., $d_G(v) = |N_G(v)|$. For a set $V' \subseteq V(G)$, $N_G(V')$ and $N_G[V']$ respectively denote the open neighbourhood and closed neighbourhood of V', i.e., $N_G(V') = (\bigcup_{v \in V'} N_G(v)) \setminus V'$ and $N_G[V'] = \bigcup_{v \in V'} N_G[v]$. And $CN_G(V')$ denotes the common neighbours of the vertices in V', i.e., $CN_G(V') = \bigcap_{v \in V'} N_G(v)$. Note that $CN_G(V') \subseteq V(G) \setminus V'$, because for every $v \in V'$, we have $v \notin N_G(v)$, and therefore, $v \notin CN_G(V')$. Also, for $V' \subseteq V(G)$ with $|V'| \geq 2$, by $N_G^{[2]}(V')$, we denote the union of the sets of common neighbours of every pair of vertices in V', i.e., $N_G^{[2]}(V') = (\bigcup_{\substack{u,v \in V' \\ u \neq v}} CN_G(\{u, v\})) \setminus V'$. For a pair of vertices $u, v \in V(G)$, $\text{dist}_G(x, y)$ denotes the length of a shortest path between x and y in G. We may omit the subscript when the graph G is clear from the context.

Consider a graph G. By a maximal clique (resp. maximal independent set) in G, we mean an inclusion-wise vertex maximal clique (resp. independent set) in G. That is, a clique Q (resp. an independent set I) in G is a maximal clique (resp. a maximal independent set) if $G[V(Q) \cup \{v\}]$ is not a clique (resp. $I \cup \{v\}$ is not an independent set) for any $v \in V(G) \setminus V(Q)$ (resp. $v \in V(G) \setminus I$). We say that an independent set I in G is 2-maximal if I is a maximal independent set and $(I \setminus \{v\}) \cup \{u, u'\}$ is not an independent set for every $v \in I$ and $u, u' \in V(G)$. That is, I is 2-maximal if I is maximal and no vertex in I can be replaced by 2 vertices from $V(G) \setminus I$.

We use $\mathcal{Q}(G)$ to denote the family of all maximal cliques in G. For $\ell > 0$, we denote by $\mathcal{Q}^\ell(G)$, the family of all maximal cliques in G of size at least ℓ. We also define two vertex subsets as follows: $L^\ell(G) = \bigcup_{Q \in \mathcal{Q}^\ell(G)} V(Q)$, and $R^\ell(G) = V(G) \setminus L^\ell(G)$. That is, $L^\ell(G)$ is the set of all vertices in G that belong to at least one maximal clique of size at least ℓ, and $R^\ell(G)$ contains the remaining vertices. Notice that $\{L^\ell(G), R^\ell(G)\}$ is a partition of $V(G)$ (with one of the parts possibly being empty).

2.1 Summary of Results From [26] and [39]

In this section, we briefly summarise the results from [26] and [39] that we will be using throughout. Following the notation of Koana et al. [39], for positive integers a, b and c, we let $R_c(a, b) = (c-1)\binom{b-1}{2} + (a-1)(b-1) + 1$.

▶ **Lemma 1** ([39]). *For positive integers a, b and c, every c-closed graph with at least $R_c(a, b)$ vertices contains either a clique of size a or an independent set of size b.*

▶ Remark 2. The proof of the above lemma [39, Proof of Lemma 3.1], in fact, shows that if G is a c-closed graph on at least $R_c(a, b)$ vertices such that G contains no clique of size a, then any 2-maximal independent set in G has size at least b.

Recall that the Independent Set problem takes a graph G and a non-negative integer k as input, and the task is to decide if G has an independent set of size at least k. Koana et al. [39] also showed that the Independent Set problem on c-closed graphs admits a kernel with ck^2 vertices. Specifically, they proved the following.

▶ **Lemma 3** ([39]). *There is an algorithm that, given a graph G and a non-negative integer k as input, runs in polynomial time, and outputs a graph G' such that (i) G' is an induced subgraph of G, (ii) G has an independent set of size k if and only if G' has an independent set of size k, and (iii) if $|V(G')| > ck^2$ then any maximal independent set in G' has size at least k.*

▶ **Corollary 4** (♣). *There is an algorithm that, given an n-vertex c-closed graph G and a non-negative integer k as input, runs in time $2^{\mathcal{O}(k \log(ck))} n^{\mathcal{O}(1)}$, and either returns a k-sized independent set of G if one exists, or correctly reports that no such set exists.*

Note that Corollary 4 follows immediately from Lemma 3. Fox et al. [26] showed that the number of maximal cliques in an n-vertex c-closed graph is bounded by $2^{\mathcal{O}(c)} n^2$. Specifically, they proved the following.

▶ **Lemma 5** ([26]). *Let G be a c-closed graph on n vertices. Then G contains at most $3^{(c-1)/3} n^2$ maximal cliques. Moreover, there is an algorithm that, given G as input, runs in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$, and enumerates all maximal cliques in G.*

2.2 Some Preliminary Lemmas

We now prove a few lemmas that we will be using throughout this paper.

▶ **Lemma 6** ([39]). *Let G be a c-closed graph, and Q a maximal clique in G. Then, for any $v \in V(G) \setminus V(Q)$, v has at most $c-1$ neighbours in $V(Q)$, i.e., $|N(v) \cap V(Q)| \leq c-1$.*

Lemma 6 implies that in a c-closed graph, every "small" dominating set must intersect every "large" clique.

▶ **Corollary 7** (♣). *Let G be a c-closed graph and k a non-negative integer. Let D be a dominating set of G of size at most k, and C a maximal clique in G of size at least $(c-1)k+1$. Then, $D \cap V(C) \neq \emptyset$.*

We now show that if a c-closed graph G contains sufficiently many large cliques, then G contains a sufficiently large independent set as well.

▶ **Lemma 8** (♣). *Let ℓ be a positive integer, and G be a c-closed graph such that $|\mathcal{Q}^{\beta(c,\ell)}(G)| \geq \ell$. Then, G has an independent set of size ℓ. Moreover, there is a polynomial time algorithm that, given a c-closed graph G and distinct $Q_1, Q_2, \ldots, Q_\ell \in \mathcal{Q}^{\beta(c,\ell)}(G)$ as input, returns an ℓ-sized independent set in G.*

▶ **Lemma 9** (♣). *Let ℓ be a positive integer. Let G be a graph and $V_1, V_2, \ldots, V_\ell \subseteq V(G)$ be such that $\bigcup_{i\in[\ell]} V_i = V(G)$, and $G[V_i]$ is a clique for every $i \in [\ell]$. Then, any independent set in G has size at most ℓ.*

▶ **Lemma 10.** *There is an algorithm that, given an n-vertex c-closed graph G and a positive integer ℓ as input, runs in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$, and either returns an independent set of size at least ℓ, or correctly concludes that every independent set in G has size at most $(\ell-1) + R_c(\beta(c,\ell),\ell) - 1 = \mathcal{O}(c \cdot \ell^2)$.*

Proof. Given G and ℓ as input, our algorithm works as follows. We first use the algorithm in Lemma 5 to construct $\mathcal{Q}(G)$ and $\mathcal{Q}^{\beta(c,\ell)}(G)$ in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. If $|\mathcal{Q}^{\beta(c,\ell)}(G)| \geq \ell$, then we return an ℓ-sized independent set constructed using the algorithm in Lemma 8.

Otherwise we construct the sets $L^{\beta(c,\ell)}(G)$, and $R^{\beta(c,\ell)}(G)$. By the definition of the sets $L^{\beta(c,\ell)}(G)$, and $R^{\beta(c,\ell)}(G)$, the induced subgraph $G' = G[R^{\beta(c,\ell)}(G)]$ contains no clique of size $\beta(c,\ell)$. And G', being an induced subgraph of G, is c-closed. So, if $|V(G')| \geq R_c(\beta(c,\ell),\ell)$, then by Lemma 1, G' contains an independent set of size ℓ. And we return a 2-maximal independent set in G', which can be computed in polynomial time, and which, by Remark 2, has size at least ℓ.

Otherwise, if $|\mathcal{Q}^{\beta(c,\ell)}(G)| \leq \ell-1$, and $|V(G')| = |R^{\beta(c,\ell)}(G)| \leq R_c(\beta(c,\ell),\ell)-1$, then we return that every independent set in G has size at most $(\ell-1) + R_c(\beta(c,\ell),\ell) - 1$.

Note that the only time consuming step in this algorithm is the construction of the families $\mathcal{Q}(G)$ and $\mathcal{Q}^{\beta(c,\ell)}(G)$ in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. The rest of the steps run in polynomial time.

To see the correctness of the last step, assume that $|\mathcal{Q}^{\beta(c,\ell)}(G)| \leq \ell-1$ and $|V(G')| = |R^{\beta(c,\ell)}(G)| \leq R_c(\beta(c,\ell),\ell)-1$. Note that by definition, $L^{\beta(c,\ell)}(G) = \bigcup_{Q\in\mathcal{Q}^{\beta(c,\ell)}(G)} V(Q)$. And therefore, by Lemma 9, any independent set in $G[L^{\beta(c,\ell)}(G)]$ has size at most $|\mathcal{Q}^{\beta(c,\ell)}(G)| \leq \ell-1$. Finally, as $\{L^{\beta(c,\ell)}(G), R^{\beta(c,\ell)}(G)\}$ is a partition of $V(G)$, for any independent set $I \subseteq V(G)$, we have $|I| = |I \cap L^{\beta(c,\ell)}(G)| + |I \cap R^{\beta(c,\ell)}(G)| \leq (\ell-1) + |R^{\beta(c,\ell)}(G)| \leq (\ell-1) + R_c(\beta(c,\ell),\ell) - 1$. Hence, the lemma follows. ◀

▶ **Lemma 11** (♣). *Let G be a graph and k a non-negative integer. Let I be an independent set in G of size $k+1$. Then, for any dominating set D of G, if $|D| \leq k$, then $D \cap N^{[2]}(I) \neq \emptyset$. Moreover, if G is c-closed, then $|N^{[2]}(I)| \leq (c-1)\binom{k+1}{2}$.*

▶ **Lemma 12** (♣). *Let G be a c-closed graph, and $Y \subseteq V(G)$ be such that $|Y| \leq c-1$. Then, the graph $G[CN(Y)]$ is $(c-|Y|)$-closed.*

3 A Polynomial Kernel for Perfect Code on c-closed graphs

To design our kernel, we consider a slightly more general version of the problem, which we call BW-Perfect Code. A bw-graph is a graph G along with a partition of $V(G)$ into two parts, B and W. We do not require that both B and W be non-empty. We call the elements of B black vertices and the elements of W white vertices, and for convenience we write that (G, B, W) is a bw-graph. A bw-perfect code of (G, B, W) is a set of vertices $D \subseteq B$ such that $|N[v] \cap D| = 1$ for every $v \in V(G)$. That is, a bw-perfect code is a set of black vertices that dominates every vertex of G exactly once. The definition of a perfect code immediately implies the following observation.

▶ **Observation 13.** *Let (G, B, W) be a bw-graph, and $D \subseteq B$ a bw-perfect code of G. Then,* (i) *D is a dominating set of G, and* (ii) *$dist_G(x, y) \geq 3$ for every pair of distinct vertices $x, y \in D$.*

We now formally define the BW-Perfect Code problem below.

> BW-Perfect Code **Parameter:** $k + cl(G)$
> **Input:** A bw-graph (G, B, W) and a non-negative integer k.
> **Question:** Does (G, B, W) have a bw-perfect code of size at most k?

It is not difficult to see that an instance (G, k) of Perfect Code can be reduced to an equivalent instance $((G, B, W), k)$ of BW-Perfect Code by taking $B = V(G)$ and $W = \emptyset$. We now move to designing a kernel for BW-Perfect Code on c-closed graphs. We first prove that for each fixed positive integer c, the BW-Perfect Code problem on c-closed graphs admits a kernel with $\mathcal{O}(k^{3(2^c-1)})$ vertices. And then argue that an instance of BW-Perfect Code can be reduced in polynomial time to an equivalent instance of Perfect Code, which will give us the required kernel. Specifically, we prove the following theorem.

▶ **Theorem 14.** *Let c be a fixed positive integer. There is an algorithm that, when given an instance $((G, B, W), k)$ of* BW-Perfect Code *as input, where G is an n-vertex c-closed graph, runs in polynomial time, and returns an equivalent instance $((G', B', W'), k')$ of the* BW-Perfect Code *problem such that G' is a c-closed graph and $|V(G')| + k' = \mathcal{O}(k^{3(2^c-1)})$.*

In addition to Theorem 14, we also need the following two intermediate lemmas to prove that Perfect Code admits a kernel. The first of these lemmas deals with the Perfect Code problem on 1-closed graphs, (which are precisely graphs in which every connected component is a clique), and the second one presents a polynomial time reduction from BW-Perfect Code to Perfect Code.

▶ **Lemma 15** (♣). Perfect Code *is polynomial time solvable on* 1*-closed graphs.*

▶ **Lemma 16.** *Let $c > 1$ be a fixed integer. There is an algorithm that given an instance $((G', B', W'), k')$ of* BW-Perfect Code*, runs in polynomial time, and returns an equivalent instance (G'', k'') of* Perfect Code *such that (i) G'' is c-closed if G' is c-closed, (ii) $|V(G'')| = \mathcal{O}(|V(G')|)$, and (ii) $k'' \leq k' + 1$.*

Finally, as a consequence of Theorem 14, Lemmas 15 and 16, we derive the following.

▶ **Theorem 17.** *Let c be a fixed positive integer.* Perfect Code *on c-closed graphs admits a kernel with $\mathcal{O}(k^{3(2^c-1)})$ vertices.*

Proof. Let (G, k) be an instance of Perfect Code, where G is a c-closed graph. Our kernelization algorithm returns an equivalent instance (G'', k'') of Perfect Code as follows. If $c = 1$, then we use the algorithm in Lemma 15 to solve the Perfect Code problem on (G, k). And if (G, k) is a yes-instance, we take (G'', k'') to be a trivial yes-instance of Perfect Code with $|V(G'')| + k'' = \mathcal{O}(k)$, and otherwise we take (G'', k'') to be a trivial no-instance of Perfect Code with $|V(G'')| + k'' = \mathcal{O}(k)$, and return (G'', k'').

If $c > 1$, then we create from (G, k), an equivalent instance $((G, B, W), k)$ of BW-Perfect Code by taking $B = V(G)$ and $W = \emptyset$. And then apply the algorithm in Theorem 14, to obtain an equivalent instance $((G', B', W'), k')$ of BW-Perfect Code, where $|V(G')| + k' = \mathcal{O}(k^{3(2^c-1)})$. Finally, we apply the algorithm in Lemma 16 to obtain from $((G', B', W'), k')$ an equivalent instance (G'', k'') of Perfect Code. Note that as the algorithms in Lemma 15, Theorem 14 and Lemma 16, run in polynomial time, our kernelization algorithm returns (G'', k'') in polynomial time. And since Lemma 16 guarantees that $|V(G'')| = \mathcal{O}(|V(G')|)$, and $k'' \leq k' + 1$, we have $|V(G'')| + k'' = \mathcal{O}(k^{3(2^c-1)})$, and the theorem follows. ◀

So now we only need to prove Theorem 14. We first give a sketch of the proof of Lemma 16.

Proof Sketch of Lemma 16. Consider an instance $((G', B', W'), k')$ of BW-Perfect Code. If $W' = \emptyset$, then we take $G'' = G'$ and $k'' = k'$. Note that this choice of G'' and k'' satisfies all the properties stated in the lemma. So, assume that $W' \neq \emptyset$. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$, and without loss of generality let $W' = \{v_1, v_2, \ldots, v_r\}$ for some $r \leq n$. We define the graph G'' as follows: $V(G'') = X \cup Y \cup Z$ and $E(G'') = E_1 \cup E_2 \cup E_3 \cup E_4$, where $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_r\}$ and $Z = \{z, z_1, z_2, \ldots, z_{k'+2}\}$; and $E_1 = \{x_i x_j \mid v_i v_j \in E(G')\}$, $E_2 = \{x_i y_i \mid i \in [r]\}$ and $E_3 = \{y_i z \mid i \in [r]\}$ and $E_4 = \{z z_i \mid i \in [k' + 2]\}$. And we set $k'' = k' + 1$. Note that $G''[X]$ is an isomorphic copy of G'. The set Y is another copy of W'. Thus, $\{x_1, x_2, \ldots, x_r\}$ and Y are two copies of W', and the set E_2 is a matching in G'' between the two copies.

First, $|V(G'')| = |X| + |Y| + |Z| = |V(G')| + |W'| + (k' + 3) = \mathcal{O}(|V(G')|)$. Second, we can show that $((G', B', W'), k')$ is a yes-instance of BW-Perfect Code if and only if (G'', k'') is a yes-instance of Perfect Code. ◀

The rest of this section is dedicated to proving Theorem 14. To that end, we first define two functions $\gamma, \mu : \mathbb{N} \to \mathbb{N}$ as follows. (Recall that $\alpha(a, b) = (a - 1)b + 1$ and $\beta(a, b) = 2[(a-1)(b-1)+1]$.) For $a, b \in \mathbb{N}$, we have $\gamma(1, b) = b+1$, and $\gamma(a, b) = b\mu(a-1, b)+1$; and $\mu(a, b) = \gamma(a, b) + R_a(\beta(a, \gamma(a, b) + 1), \gamma(a, b) + 1) - 1$. These functions γ and μ will be used to bound the size of independent sets in G when $((G, B, W), k)$ is a yes-instance.

▶ **Observation 18.** *Observe that for every fixed $a, i \in \mathbb{N}$, and for $b \in \mathbb{N}$, we have $R_i(a, b) = \mathcal{O}(b^2)$ and $\beta(a, b) = \mathcal{O}(b)$. Therefore, we have*

$$
\begin{array}{ll}
\gamma(1,b) = \mathcal{O}(b) & \mu(1,b) = \mathcal{O}(b) + R_1(\mathcal{O}(b), \mathcal{O}(b)) = \mathcal{O}(b^2) \\
\gamma(2,b) = b\mu(1,b) + 1 = \mathcal{O}(b^3) & \mu(2,b) = \mathcal{O}(b^3) + R_2(\mathcal{O}(b^3), \mathcal{O}(b^3)) = \mathcal{O}(b^6) \\
\gamma(3,b) = b\mu(2,b) + 1 = \mathcal{O}(b^7) & \mu(3,b) = \mathcal{O}(b^7) + R_3(\mathcal{O}(b^7), \mathcal{O}(b^7)) = \mathcal{O}(b^{14}) \\
\ldots & \ldots \\
\gamma(a,b) = \mathcal{O}(b^{2^a-1}) & \mu(a,b) = \mathcal{O}(b^{2(2^a-1)}).
\end{array}
$$

Outline of the kernel. Our kernel for BW-Perfect Code has two parts. In the first part, we bound the size of independent sets in (G, B, W) using Reduction Rule 19, and in the second part, we bound the size of cliques in (G, B, W) using Reduction Rules 27-29. Once the size of cliques and independent sets are bounded, we apply Lemma 1.

To bound the size of independent sets in case $((G, B, W), k)$ is a yes-instance, observe the following fact. Consider an independent set I in G and a bw-perfect code $D \subseteq B$ of size at most k. Then, we can partition I into at most k parts, say, $I_1, I_2, \ldots, I_k$, such that for each $j \in [k]$, there exists a unique vertex $v_j \in D$ that dominates I_j, i.e., $I_j \subseteq N(v_j)$. Thus, to bound $|I|$, we only need to bound $|I_j|$ for every $j \in [k]$. More generally, we only need to bound the size of independent sets contained in $N(v)$ for every $v \in V(G)$. To do this, suppose that for every $Y \subseteq V(G)$ with $|Y| = 2$ we have already managed to bound the size of independent sets contained in $CN(Y)$ by some function of c and k, say, $f(c, k)$. That is, every independent set with at least 2 common neighbours has size at most $f(c, k)$. Now, consider $v \in V(G)$. And let I' be an independent set of size at least $k \cdot f(c, k) + 1$ contained in $N(v)$ and D a bw-perfect code of size at most k. Then, we must have $v \in D$. If not, there exists $u \in D$ that dominates at least $|I'|/k$ vertices of I. That is, there exist $u \in D$ and $I'' \subseteq I'$ such that $|I''| \geq |I'|/k > f(c, k)$ and $I'' \subseteq N(u)$. But note that $I'' \subseteq I' \subseteq N(v)$. Thus, $I'' \subseteq CN(\{u, v\})$ and $|I''| > f(c, k)$, which we have already ruled out to be impossible. By repeating these arguments, we can show that, to obtain the bound of $f(c, k)$ for independent sets with 2 common neighbours, we only need to bound the size of independent sets with 3 common neighbours. This train of arguments only needs to continue until we reach independent sets with $c-1$ common neighbours. Thus, we start with sets Y of size $c - 1$ and bound the size of independent sets contained in $CN(Y)$. Then proceed to sets Y of size $c - 2$ and so on. This idea is formalised in Reduction Rule 19. But the difficulty comes in checking if $CN(Y)$ contains an independent set of the required size, which cannot be done in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$. To overcome this, we use the weaker result of Lemma 10, which causes the bound on the independent set size to increase exponentially in each successive stage. Thus, after $c - 1$ stages, we only manage to obtain a bound of $\mu(c - 1, k) = k^{\mathcal{O}(2^c)}$ for the size of independent sets contained in $N(v)$ for every $v \in V(G)$. And this bound is where the kernel size comes from.

In the second part, bounding the clique size is fairly straightforward. This involves removing twin vertices (Reduction Rule 27), and identifying irrelevant vertices (vertices that cannot belong to any bw-perfect code of size at most k) and colouring them white or removing them (Reduction Rules 28 and 29). (Also, each time we introduce a reduction rule, we apply it exhaustively. So from that point onwards, we would assume that the reduction rule is no longer applicable.)

We now formally introduce the following reduction rule.

▶ **Reduction Rule 19.** *For each $i \in [c - 1]$, we introduce Reduction Rule 19.i as follows. Let $((G, B, W), k)$ be an instance of* BW-Perfect Code*. For each fixed set $Y \subseteq V(G)$ with $|Y| = c - i$, we run the algorithm in Lemma 10 on the graph $G[CN(Y)]$ with $\ell = \gamma(i, k) + 1$. If the algorithm returns an independent set I of size ℓ, then delete a vertex $v \in I$ from G, and colour $N_G(v) \setminus Y$ white. That is, we create a new instance $((G', B', W'), k)$ as follows: $G' = G - v$, $B' = B \setminus (N_G[v] \setminus Y)$ and $W' = V(G') \setminus B' = W \cup (N_G[v] \setminus Y)$. We keep repeating this proceudre until the algorithm in Lemma 10 returns that every independent set in $G[CN(Y)]$ has size at most $(\ell - 1) + R_c(\beta(c, \ell), \ell) - 1$. Also, we apply Reduction Rule 19.i in the increasing order of i. That is, we first apply Reduction Rule 19.*1 *exhaustively, and for each $i \in [c - 1] \setminus \{1\}$, we apply Reduction Rule 19.i only if Reduction Rule 19.$(i - 1)$ is no longer applicable.*

We now observe the following fact, which will be useful in establishing the correctness of Reduction Rule 19.

▶ **Observation 20.** *Fix $i \in [c-1]$. For any $Y \subseteq V(G)$ with $|Y| = c - i$, by Lemma 12, the subgraph $G[CN(Y)]$ is i-closed. Therefore, after an exhaustive application of Reduction Rule 19.i, by Lemma 10, every independent set in $G[CN(Y)]$ has size at most $\gamma(i,k) + R_i(\beta(i,\gamma(i,k)+1), \gamma(i,k)+1) - 1 = \mu(i,k)$. In particular, when $i = c-1$, we get that after an exhaustive application of Reduction Rule 19.$(c-1)$, for every $v \in V(G)$, every independent set in $G[N(v)]$ has size at most $\mu(c-1,k)$.*

▶ **Lemma 21.** *Let $((G,B,W),k)$ be an instance of* BW-Perfect Code*. Let $Y \subseteq V(G)$ be such that $|Y| = c-1$, and $I \subseteq CN(Y)$ be an independent set with $|I| \geq \gamma(1,k)$. Then, for any bw-perfect code $D \subseteq B$ of (G,B,W) with $|D| \leq k$, we have $|D \cap Y| = 1$.*

Proof. Let $D \subseteq B$ be a bw-perfect code of (G,B,W) with $|D| \leq k$. We first claim that $D \cap Y \neq \emptyset$. Assume for a contradiction that $D \cap Y = \emptyset$. Now, since $|I| \geq \gamma(1,k) = k+1$ and $|D| \leq k$, by the pigeonhole principle, there exists a vertex $u \in D$ that dominates at least two vertices of I, say, $w_1, w_2 \in I$. That is, $u \in N[w_1] \cap N[w_2]$. Since I is an independent set, and $uw_1, uw_2 \in E(G)$, we can conclude that $u \neq w_1$ and $u \neq w_2$. Thus, $u \in N(w_1) \cap N(w_2)$. But since since $w_1, w_2 \in I \subseteq CN(Y)$, we get that $Y \subseteq N(w_1) \cap N(w_2)$. Thus, $Y \cup \{u\} \subseteq N(w_1) \cap N(w_2)$. Because of our assumption that $D \cap Y = \emptyset$, we have $u \notin Y$, and thus $|Y \cup \{u\}| = c$. Thus, w_1 and w_2 have at least c common neighbours, and therefore $w_1w_2 \in E(G)$, which is not possible as $w_1$and w_2 belong to the independent set I. Thus, $D \cap Y \neq \emptyset$. Now, if there exist $y_1, y_2 \in D \cap Y$, where $y_1 \neq y_2$, then for any $x \in I$, we have $y_1, y_2 \in N[x] \cap D$, which, by the definition of a bw-perfect code, is not possible. Therefore, we conclude that $|D \cap Y| = 1$. ◀

▶ **Lemma 22.** *Fix $i \in [c-1] \setminus \{1\}$. Let $((G,B,W),k)$ be an instance of* BW-Perfect Code *to which Reduction Rule 19.$(i-1)$ has been applied exhaustively. Let $Y \subseteq V(G)$ be such that $|Y| = c-i$, and $I \subseteq CN(Y)$ be an independent set with $|I| \geq \gamma(i,k)$. Then, for any bw-perfect code $D \subseteq B$ of (G,B,W) with $|D| \leq k$, we have $|D \cap Y| = 1$.*

Proof. Let $D \subseteq B$ be a bw-perfect code of (G,B,W) with $|D| \leq k$. We first claim that $D \cap Y \neq \emptyset$. Assume for a contradiction that $D \cap Y = \emptyset$. Now, since $|I| \geq \gamma(i,k) = k\mu(i-1,k)+1$ and $|D| \leq k$, by the pigeonhole principle, there exists a vertex $u \in D$ that dominates at least $\mu(i-1,k)+1$ vertices of I. Let $I' \subseteq I$ be such that $|I'| \geq \mu(i-1,k)+1$ and u dominates I'. That is, $I' \subseteq N[u]$. Observe first that $u \notin I'$. To see this, suppose that $u \in I'$. Then, for every $w \in I' \setminus \{u\}$, since u dominates w, we must have $uw \in E(G)$, which contradicts the fact that I' is an independent set. So, $u \notin I'$, and therefore, $I' \subseteq N(u)$. And we already have $I' \subseteq I \subseteq CN(Y)$. We can conclude that $I' \subseteq N(u) \cap CN(Y) = CN(Y \cup \{u\})$. Because of our assumption that $D \cap Y = \emptyset$, we have $u \notin Y$, and thus $|Y \cup \{u\}| = c-i+1 = c-(i-1)$. That is, $Y \cup \{u\}$ is a set of size $c-(i-1)$, and I' is an independent set such that $I' \subseteq CN(Y \cup \{u\})$, and $|I'| \geq \mu(i-1,k)+1$. But this conclusion contradicts Observation 20 because of our assumption that Reduction Rule 19.$(i-1)$ has been applied exhaustively. Thus, $D \cap Y \neq \emptyset$. Now, if there exist $y_1, y_2 \in D \cap Y$, where $y_1 \neq y_2$, then for any $x \in I$, we have $y_1, y_2 \in N[x] \cap D$, which, by the definition of a bw-perfect code, is not possible. Therefore, we conclude that $|D \cap Y| = 1$. ◀

▶ **Lemma 23.** *Reduction Rule 19.i is safe.*

Proof. Let $((G', B', W'), k)$ be the instance obtained from $((G, B, W), k)$ by a single application of Reduction Rule 19.i. Then, there exists $Y \subseteq V(G)$ with $|Y| = c - i$, and an independent set $I \subseteq CN(Y)$ with $|I| = \gamma(i, k) + 1$ and a vertex $v \in I$ such that $G' = G - v$, $B' = B \setminus (N_G[v] \setminus Y)$ and $W' = V(G') \setminus B' = W \cup (N_G[v] \setminus Y)$. We shall show that $((G, B, W), k)$ and $((G', B', W'), k)$ are equivalent instances.

First consider the case when $i = 1$. Then, $|Y| = c - 1$, and $|I| = \gamma(1, k) + 1$. Assume that $((G, B, W), k)$ is a yes-instance of BW-Perfect Code, and let $D \subseteq B$ be a bw-perfect code of (G, B, W) of size at most k. Then, by Lemma 21, $|D \cap Y| = 1$. Let $\{y\} = D \cap Y$. But then since $y \in D$, and $I \subseteq CN(Y) \subseteq N(y)$, we have $I \cap D = \emptyset$. In particular $v \notin D$. Also, for any $w \in N_G(v) \setminus Y$, we have $\text{dist}_G(y, w) \leq 2$, and thus, by Observation 13, we have $w \notin D$. Thus, $D \cap (N_G[v] \setminus Y) = \emptyset$, and therefore, $D \subseteq B \setminus (N_G[v] \setminus Y) = B'$. Thus, D is a bw-perfect code of (G', B', W') as well.

Conversely, assume that $((G', B', W'), k)$ is a yes-instance, and let $D' \subseteq B'$ be a bw-perfect code of (G', B', W') with $|D'| \leq k$. We claim that D' is a bw-perfect code of (G, B, W) as well. Note that for any $x \in V(G) \setminus \{v\}$, we have $N_{G'}[x] = N_G[x] \setminus \{v\}$. Therefore, since $v \notin D'$, we have $|D' \cap N_G[x]| = |D' \cap N_{G'}[x]| = 1$. So, now we only need to show that $|D' \cap N_G[v]| = 1$. Note that $N_G[v] = (N_G[v] \setminus Y) \cup (N_G[v] \cap Y)$. First, since $N_G[v] \setminus Y \subseteq W'$, and $D' \subseteq B'$, we get that $D' \cap (N_G[v] \setminus Y) = \emptyset$. So we only need to show that $|D' \cap (N_G[v] \cap Y)| = 1$. Now, observe that as $|I \setminus \{v\}| = \gamma(1, k)$, by Lemma 21, we have $|D' \cap Y| = 1$. Let $\{y'\} = D' \cap Y$. Then, $y' \in D' \cap N_G[v]$, and in fact, $\{y'\} = D' \cap (N_G[v] \cap Y)$. This completes the proof for the case when $i = 1$.

Now, assume that $i > 1$. First, by assumption, Reduction Rule 19.j is not applicable to $((G, B, W), k)$ for any $j \in [i-1]$. And we have $|Y| = c-i$, and $|I| = \gamma(i-1, k)+1$. Assume that $((G, B, W), k)$ is a yes-instance of BW-Perfect Code, and let $D \subseteq B$ be a bw-perfect code of (G, B, W) of size at most k. Then, by Lemma 22, we have $|D \cap Y| = 1$. Let $\{y\} = D \cap Y$. But then since $y \in D$, and $I \subseteq CN(Y) \subseteq N(y)$, we have $I \cap D = \emptyset$. In particular $v \notin D$. Also, for any $w \in N_G(v) \setminus Y$, we have $\text{dist}_G(y, w) \leq 2$, and thus, by Observation 13, we have $w \notin D$. Thus, $D \cap (N_G[v] \setminus Y) = \emptyset$, and therefore, $D \subseteq B \setminus (N_G[v] \setminus Y) = B'$. Thus, D is a bw-perfect code of (G', B', W') as well.

Conversely, assume that $((G', B', W'), k)$ is a yes-instance, and let $D' \subseteq B'$ be a bw-perfect code of (G', B', W') with $|D'| \leq k$. We claim that D' is a bw-perfect code of (G, B, W) as well. Note that for any $x \in V(G) \setminus \{v\}$, we have $N_{G'}[x] = N_G[x] \setminus \{v\}$. Therefore, since $v \notin D'$, we have $|D' \cap N_G[x]| = |D' \cap N_{G'}[x]| = 1$. So, now we only need to show that $|D' \cap N_G[v]| = 1$. Note that $N_G[v] = (N_G[v] \setminus Y) \cup (N_G[v] \cap Y)$. First, since $N_G[v] \setminus Y \subseteq W'$, and $D' \subseteq B'$, we get that $D' \cap (N_G[v] \setminus Y) = \emptyset$. So we only need to show that $|D' \cap (N_G[v] \cap Y)| = 1$. Now, observe that as $|I \setminus \{v\}| = \gamma(i, k)$, by Lemma 22, we have $|D' \cap Y| = 1$. Let $\{y'\} = D' \cap Y$. Then, $y' \in D' \cap N_G[v]$, and in fact, $\{y'\} = D' \cap (N_G[v] \cap Y)$. This completes the proof for the lemma. ◀

▶ Remark 24. Observe that each application of Reduction Rule 19 can be executed in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$. Also, for each set $Y \subseteq V(G)$ with $|Y| \leq c - 1$, Reduction Rule 19 is applied only at most $|CN(Y)| \leq n$ times. And note that the set Y has at most $\sum_{i=1}^{c-1} \binom{n}{i} = n^{\mathcal{O}(c)}$ choices. Thus, Reduction Rule 19 can be applied exhaustively in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(c)}$. Since c is a fixed constant, we have $2^{\mathcal{O}(c)} n^{\mathcal{O}(c)} = n^{\mathcal{O}(1)}$. That is, we can exhaustively apply Reduction Rule 19 in polynomial time. So, from now on, we assume that Reduction Rule 19 has been applied exhaustively.

The following lemma bounds the size of an independent set in G if $((G, B, W), k)$ is a yes-instance.

▶ **Lemma 25.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *If $((G, B, W), k)$ is a yes-instance, then every independent set in G has size at most $\gamma(c, k) - 1$.*

Proof. Assume that $((G, B, W), k)$ is a yes-instance of BW-Perfect Code, and let $D \subseteq B$ be a bw-perfect code of (G, B, W) of size at most k. Let $I \subseteq V(G)$ be an independent set. Assume for a contradiction that $|I| \geq \gamma(c, k) = k\mu(c-1, k) + 1$. Then, since $|D| \leq k$, by the pigeonhole principle, there exists $v \in D$ such that v dominates at least $\mu(c-1, k)+1$ vertices of I. That is, there exists an independent set I' such that $I' \subseteq N(v)$ and $|I'| \geq \mu(c-1, k)+1$, which, by Observation 20, is not possible, as Reduction Rule 19, and in particular, Reduction Rule 19.$(c-1)$ has been applied exhaustively. ◀

We have thus bounded the size of every independent set in G for yes-instances. This immediately bounds the number of large cliques (by Lemma 8), as well as the number of vertices that do not belong to any large maximal clique (by Lemma 1).

▶ **Lemma 26.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *If $((G, B, W), k)$ is a yes-instance, then*
1. $|\mathcal{Q}^{\beta(c,\gamma(c,k))}(G)| \leq \gamma(c, k) - 1$, *and*
2. $|R^{\beta(c,\gamma(c,k))}(G)| \leq R_c(\beta(c, \gamma(c, k)), \gamma(c, k)) - 1$.

Proof. Assume that $((G, B, W), k)$ is a yes-instance of BW-Perfect Code.
1. If $|\mathcal{Q}^{\beta(c,\gamma(c,k))}(G)| \geq \gamma(c, k)$, then by Lemma 8, G contains an independent set of size $\gamma(c, k)$, which contradicts Lemma 25.
2. By the definition of $R^{\beta(c,\gamma(c,k))}(G)$, the induced subgraph $G[R^{\beta(c,\gamma(c,k))}(G)]$ of G contains no clique of size $\beta(c, \gamma(c, k))$. By Lemma 25, the graph G, and hence the graph $G[R^{\beta(c,\gamma(c,k))}(G)]$, contains no independent set of size $\gamma(c, k)$. The bound then follows from Lemma 1. ◀

In the next three reduction rules we bound the size of every clique in G as well, which, in turn, will help us bound $|L^{\beta(c,\gamma(c,k))}(G)|$. We begin by introducing a reduction rule, which says that if two vertices have same closed neighborhood and the same colour, then we can safely delete one of them.

▶ **Reduction Rule 27.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *Let $x, y \in V(G)$ be distinct vertices such that $N_G[x] = N_G[y]$. If $x, y \in B$ or $x, y \in W$, then delete x.*

Let Q be a maximal clique of size at least $\alpha(c, k)$. By Corollary 7, exactly one vertex from $V(Q)$ is in every bw-perfect code. Therefore, no vertex from $N(V(Q))$ belongs to a bw-perfect code of size at most k. So we color $N(V(Q))$ white in the next reduction rule.

▶ **Reduction Rule 28.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code, *and let $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$. Colour $N(V(Q))$ white. That is, we construct the instance $((G, B', W'), k)$ of* BW-Perfect Code, *where $W' = W \cup N(V(Q))$, and $B' = B \setminus N(V(Q))$.*

Let $Q \in \mathcal{Q}^{\alpha(c,k)+1}(G)$. We define $Z(Q)$ to be the set of vertices in $V(Q)$ that have neighbours in some other maximal clique of size at least $\alpha(c, k)$, i.e., $Z(Q) = \{u \in V(Q) \mid uv \in E(G)$ for some $v \in V(Q'),$ where $Q' \in \mathcal{Q}^{\alpha(c,k)}(G), u \notin V(Q'),$ and $Q' \neq Q\}$. In the following reduction rule we show that we can safely delete $Z(Q)$.

▶ **Reduction Rule 29.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *If there exists $Q \in \mathcal{Q}^{\alpha(c,k)+1}(G)$ and $v \in Z(Q)$, then delete v. That is, we construct the instance $((G', B', W'), k)$ of* BW-Perfect Code, *where $G' = G - v$, $B' = B \setminus \{v\}$, and $W' = W \setminus \{v\}$.*

▶ **Lemma 30** (♣). *Reduction Rules 27, 28, and 29 are safe.*

▶ Remark 31. Observe that given an instance $((G, B, W), k)$ of BW-Perfect Code, using the algorithm in Lemma 5, we can construct $\mathcal{Q}^{\alpha(c,k)}(G)$ (and $\mathcal{Q}^{\alpha(c,k+1)}(G)$) in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. And once we construct these families of cliques, we can then exhaustively apply Reduction Rules 28 in time $|\mathcal{Q}^{\alpha(c,k)}(G)|n^{\mathcal{O}(1)}$ and Reduction Rule 29 in time $|\mathcal{Q}^{\alpha(c,k+1)}(G)|n^{\mathcal{O}(1)}$. Also, observe that we can exhaustively apply Reduction Rule 27 in polynomial time. So from now on, we assume that we have exhaustively applied Reduction Rules 27-29.

▶ **Lemma 32** (♣). *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *If $((G, B, W), k)$ is a yes-instance, then for every $Q \in \mathcal{Q}^{\beta(c,\gamma(c,k))}(G)$, we have*
1. *$Z(Q) = \emptyset$, and*
2. *$|V(Q)| \leq (c-1)[R_c(\beta(c, \gamma(c, k)), \gamma(c, k)) - 1] + 2$.*

Finally, Lemmas 26-(1) and 32-(2) together bound $|L^{\beta(c,\gamma(c,k))}(G)|$, which bounds $|V(G)|$.

▶ **Lemma 33.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *If $((G, B, W), k)$ is a yes-instance, then $|V(G)| = \mathcal{O}(k^{3(2^c-1)})$.*

Proof. Assume that $((G, B, W), k)$ is a yes-instance. Then, by Lemma 26-(1), we have $|\mathcal{Q}^{\beta(c,\gamma(c,k))}(G)| \leq \gamma(c, k) - 1 = \mathcal{O}(k^{2^c-1})$, and by Lemma 32-(2), we have $|V(Q)| \leq (c-1)[R_c(\beta(c, \gamma(c, k)), \gamma(c, k)) - 1] + 2 = \mathcal{O}((\gamma(c, k))^2) = \mathcal{O}(k^{2(2^c-1)})$. Therefore, we have

$$\begin{aligned}
|L^{\beta(c,\gamma(c,k))}(G)| &= |\bigcup_{Q \in \mathcal{Q}^{\beta(c,\gamma(c,k))}(G)} V(Q)| \\
&\leq (\gamma(c, k) - 1) \cdot (c-1)[R_c(\beta(c, \gamma(c, k)), \gamma(c, k)) - 1] + 2 \\
&= \mathcal{O}(k^{2^c-1}) \cdot \mathcal{O}(k^{2(2^c-1)}) \\
&= \mathcal{O}(k^{3(2^c-1)}).
\end{aligned}$$

Also, by Lemma 26-(2), we have $|R^{\beta(c,\gamma(c,k))}(G)| \leq R_c(\beta(c, \gamma(c, k)), \gamma(c, k)) - 1 = R_c(\mathcal{O}(k^{2^c-1}), \mathcal{O}(k^{2^c-1})) = \mathcal{O}(k^{2(2^c-1)})$. Finally, since $\{L^{\beta(c,\gamma(c,k))}(G), R^{\beta(c,\gamma(c,k))}(G)\}$ is a partition of $V(G)$, we conclude that $|V(G)| = \mathcal{O}(k^{3(2^c-1)})$. ◀

Each of our reduction rules is safe and by Remarks 24 and 31, all the reduction rules we introduced can be executed in polynomial time, and are applied only polynomially many times. We have thus proved Theorem 14.

4 Conclusion

We resolved the parameterized complexity of three domination problems – Perfect Code, CDS and PDS– on c-closed graphs. We believe that our results, along with that of Koana et al. [39], make a convincing case for pursuing the closure of a graph as a significant structural parameter. We also believe that the arguments in this paper can be adapted to solve similar problems on c-closed graphs. In particular, our strategy for Perfect Code may be applicable to the Even Dominating Set (resp. Odd Dominating Set) problems, where the goal is to check if a graph G has a dominating set D of size at most k such that D dominates every vertex of G an even (resp. odd) number of times. While we showed that PDS is W[1]-hard even on 2-closed graphs, the status of Partial Vertex Cover on c-closed graphs still remains open. It would be interesting to see if any our results extend to weakly γ-closed graphs (see [26] and [38]) as well.

References

1 Jochen Alber, Hans L. Bodlaender, Henning Fernau, Ton Kloks, and Rolf Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002. doi:10.1007/s00453-001-0116-5.

2 Jochen Alber, Hongbing Fan, Michael R. Fellows, Henning Fernau, Rolf Niedermeier, Frances A. Rosamond, and Ulrike Stege. A refined search tree technique for dominating set on planar graphs. *J. Comput. Syst. Sci.*, 71(4):385–405, 2005. doi:10.1016/j.jcss.2004.03.007.

3 Noga Alon and Shai Gutner. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. *Algorithmica*, 54(4):544–556, 2009. doi:10.1007/s00453-008-9204-0.

4 Omid Amini, Fedor V. Fomin, and Saket Saurabh. Implicit branching and parameterized partial cover problems. *J. Comput. Syst. Sci.*, 77(6):1159–1171, 2011. doi:10.1016/j.jcss.2010.12.002.

5 Nicola Apollonio and Bruno Simeone. The maximum vertex coverage problem on bipartite graphs. *Discret. Appl. Math.*, 165:37–48, 2014. doi:10.1016/j.dam.2013.05.015.

6 Markus Bläser. Computing small partial coverings. *Inf. Process. Lett.*, 85(6):327–331, 2003. doi:10.1016/S0020-0190(02)00434-9.

7 J. Adrian Bondy and Vasek Chvátal. A method in graph theory. *Discret. Math.*, 15(2):111–135, 1976. doi:10.1016/0012-365X(76)90078-9.

8 J. Adrian Bondy and Uppaluri S. R. Murty. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2008. doi:10.1007/978-1-84628-970-5.

9 Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM J. Comput.*, 22(3):560–572, 1993. doi:10.1137/0222038.

10 Leizhen Cai. Parameterized complexity of cardinality constrained optimization problems. *Comput. J.*, 51(1):102–121, 2008. doi:10.1093/comjnl/bxm086.

11 Leizhen Cai, Siu Man Chan, and Siu On Chan. Random separation: A new method for solving fixed-cardinality optimization problems. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, volume 4169 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2006. doi:10.1007/11847250_22.

12 Marco Cesati. Perfect code is W[1]-complete. *Inf. Process. Lett.*, 81(3):163–168, 2002. doi:10.1016/S0020-0190(01)00207-1.

13 Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1):2, 2006. doi:10.1145/1132952.1132954.

14 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Kernelization hardness of connectivity problems in d-degenerate graphs. *Discret. Appl. Math.*, 160(15):2131–2141, 2012. doi:10.1016/j.dam.2012.05.016.

15 Anuj Dawar and Stephan Kreutzer. Domination problems in nowhere-dense classes. In Ravi Kannan and K. Narayan Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, volume 4 of *LIPIcs*, pages 157–168. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009. doi:10.4230/LIPIcs.FSTTCS.2009.2315.

16 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. *ACM Trans. Algorithms*, 1(1):33–47, 2005. doi:10.1145/1077464.1077468.

17 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005. doi:10.1145/1101821.1101823.

18 Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: on completeness for W[1]. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995. doi:10.1016/0304-3975(94)00097-3.

19 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

20 John A. Ellis, Hongbing Fan, and Michael R. Fellows. The dominating set problem is fixed parameter tractable for graphs of bounded genus. *J. Algorithms*, 52(2):152–168, 2004. doi:10.1016/j.jalgor.2004.02.001.

21 Michael R. Fellows and Mark N. Hoover. Perfect domination. *Australas. J Comb.*, 3:141–150, 1991. URL: http://ajc.maths.uq.edu.au/pdf/3/ocr-ajc-v3-p141.pdf.

22 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 503–510. SIAM, 2010. doi:10.1137/1.9781611973075.43.

23 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Kernels for (connected) dominating set on graphs with excluded topological minors. *ACM Trans. Algorithms*, 14(1):6:1–6:31, 2018. doi:10.1145/3155298.

24 Fedor V. Fomin and Dimitrios M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.*, 36(2):281–309, 2006. doi:10.1137/S0097539702419649.

25 Jacob Fox, Tim Roughgarden, C. Seshadhri, Fan Wei, and Nicole Wein. Finding cliques in social networks: A new distribution-free model. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 55:1–55:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.55.

26 Jacob Fox, Tim Roughgarden, C. Seshadhri, Fan Wei, and Nicole Wein. Finding cliques in social networks: A new distribution-free model. *SIAM J. Comput.*, 49(2):448–464, 2020. doi:10.1137/18M1210459.

27 Michael R. Garey and David S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W.H. Freeman, New York, 1979.

28 Archontia C. Giannopoulou, George B. Mertzios, and Rolf Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *Theor. Comput. Sci.*, 689:67–95, 2017. doi:10.1016/j.tcs.2017.05.017.

29 Petr A. Golovach and Yngve Villanger. Parameterized complexity for domination problems on degenerate graphs. In Hajo Broersma, Thomas Erlebach, Tom Friedetzky, and Daniël Paulusma, editors, *Graph-Theoretic Concepts in Computer Science, 34th International Workshop, WG 2008, Durham, UK, June 30 - July 2, 2008. Revised Papers*, volume 5344 of *Lecture Notes in Computer Science*, pages 195–205, 2008. doi:10.1007/978-3-540-92248-3_18.

30 Qianping Gu and Navid Imani. Connectivity is not a limit for kernelization: Planar connected dominating set. In Alejandro López-Ortiz, editor, *LATIN 2010: Theoretical Informatics, 9th Latin American Symposium, Oaxaca, Mexico, April 19-23, 2010. Proceedings*, volume 6034 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2010. doi:10.1007/978-3-642-12200-2_4.

31 Jiong Guo and Rolf Niedermeier. Linear problem kernels for NP-hard problems on planar graphs. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 375–386. Springer, 2007. doi:10.1007/978-3-540-73420-8_34.

32 Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Parameterized complexity of vertex cover variants. *Theory Comput. Syst.*, 41(3):501–520, 2007. doi:10.1007/s00224-007-1309-3.

33 Edin Husic and Tim Roughgarden. FPT algorithms for finding dense subgraphs in c-closed graphs. *CoRR*, abs/2007.09768, 2020. arXiv:2007.09768.

34 Minghui Jiang and Yong Zhang. Perfect domination and small cycles. *Discret. Math. Algorithms Appl.*, 9(3):1750030:1–1750030:11, 2017. doi:10.1142/S1793830917500306.

35 Iyad A. Kanj and Ljubomir Perkovic. Improved parameterized algorithms for planar dominating set. In Krzysztof Diks and Wojciech Rytter, editors, *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*, volume 2420 of *Lecture Notes in Computer Science*, pages 399–410. Springer, 2002. doi:10.1007/3-540-45687-2_33.

36 Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Intuitive algorithms and t-vertex cover. In Tetsuo Asano, editor, *Algorithms and Computation, 17th International Symposium, ISAAC 2006, Kolkata, India, December 18-20, 2006, Proceedings*, volume 4288 of *Lecture Notes in Computer Science*, pages 598–607. Springer, 2006. doi:10.1007/11940128_60.

37 Joachim Kneis, Daniel Mölle, and Peter Rossmanith. Partial vs. complete domination: t-dominating set. In Jan van Leeuwen, Giuseppe F. Italiano, Wiebe van der Hoek, Christoph Meinel, Harald Sack, and Frantisek Plasil, editors, *SOFSEM 2007: Theory and Practice of Computer Science, 33rd Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 20-26, 2007, Proceedings*, volume 4362 of *Lecture Notes in Computer Science*, pages 367–376. Springer, 2007. doi:10.1007/978-3-540-69507-3_31.

38 Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Computing dense and sparse subgraphs of weakly closed graphs. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPIcs*, pages 20:1–20:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ISAAC.2020.20.

39 Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Exploiting c-closure in kernelization algorithms for graph problems. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 65:1–65:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.65.

40 Tomohiro Koana and André Nichterlein. Detecting and enumerating small induced subgraphs in c-closed graphs. *CoRR*, abs/2007.12077, 2020. arXiv:2007.12077.

41 Daniel Lokshtanov, Matthias Mnich, and Saket Saurabh. Linear kernel for planar connected dominating set. In Jianer Chen and S. Barry Cooper, editors, *Theory and Applications of Models of Computation, 6th Annual Conference, TAMC 2009, Changsha, China, May 18-22, 2009. Proceedings*, volume 5532 of *Lecture Notes in Computer Science*, pages 281–290. Springer, 2009. doi:10.1007/978-3-642-02017-9_31.

42 Daniel Lokshtanov and Vaishali Surianarayanan. Dominating set in weakly closed graphs is fixed parameter tractable. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, pages 29:1–29:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.29.

43 Chin Lung Lu and Chuan Yi Tang. Weighted efficient domination problem on some perfect graphs. *Discret. Appl. Math.*, 117(1-3):163–182, 2002. doi:10.1016/S0166-218X(01)00184-6.

44 Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. The kernelization complexity of connected domination in graphs with (no) small cycles. *Algorithmica*, 68(2):504–530, 2014. doi:10.1007/s00453-012-9681-z.

45 Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. FPT algorithms for connected feedback vertex set. *J. Comb. Optim.*, 24(2):131–146, 2012. doi:10.1007/s10878-011-9394-2.

46 Daniel Mölle, Stefan Richter, and Peter Rossmanith. Enumerate and expand: Improved algorithms for connected vertex cover and tree cover. *Theory Comput. Syst.*, 43(2):234–253, 2008. doi:10.1007/s00224-007-9089-3.

47 John W Moon and Leo Moser. On cliques in graphs. *Israel journal of Mathematics*, 3(1):23–28, 1965.

48 Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Polynomial kernels for dominating set in graphs of bounded degeneracy and beyond. *ACM Trans. Algorithms*, 9(1):11:1–11:23, 2012. doi:10.1145/2390176.2390187.

49 Venkatesh Raman and Saket Saurabh. Short cycles make W -hard problems hard: FPT algorithms for W -hard problems in graphs with no short cycles. *Algorithmica*, 52(2):203–225, 2008. doi:10.1007/s00453-007-9148-9.