**Article:**

Kanesh, L., Madathil, J., Roy, S. orcid.org/0000-0003-3633-542X et al. (2 more authors) (2023) Further Exploiting c-Closure for FPT Algorithms and Kernels for Domination Problems. SIAM Journal on Discrete Mathematics, 37 (4). pp. 2626-2669. ISSN 0895-4801

https://doi.org/10.1137/22m1491721

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


**Lawqueen Kanesh** ✉
National University of Singapore, Singapore

**Jayakrishnan Madathil** ✉
Chennai Mathematical Institute, Chennai, India

**Sanjukta Roy** ✉
Pennsylvania State University

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

**Saket Saurabh** ✉
The Institute of Mathematical Sciences, HBNI, Chennai, India, and 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 and SIDMA '22] 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, when parameterized either by $k+c$ or by $k$ 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 when parameterized by $k + cl(G)$, 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.



**AMS Subject Classification** 05C85, 68W01, 68Q25.

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

**Related Version** An preliminary version of this work was published in the proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022) [37]. The current version contains all the missing proofs and refined analyses of the running times for the kernelization algorithm for Perfect Code and the FPT algorithm for Connected Dominating Set.

**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 the Chennai Mathematical Institute and the Infosys Foundation.
*Saket Saurabh*: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 819416), and Swarnajayanti Fellowship (no. DST/SJF/MSA01/2017-18).

**Acknowledgements** We thank the anonymous reviewers for their thoughtful feedback.

## 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 study 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, when parameterized by the solution size, are W[1]-hard on general graphs [13, 20, 21], and their complexities on various restricted graph classes have been studied extensively [4, 17, 24, 31, 32, 33, 36, 47, 50].

Fox et al. [27, 28] 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 [14].) And in an attempt to capture the spirit of "social-network-like" graphs without relying on probabilistic models, Fox et al. [28] "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 reasonable 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. [28, Table A.1], and later Koana et al. [43, Table 1], showed that several social networks and biological networks are indeed $c$-closed for rather small values of $c$.

Fox et al. [28] 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 when parameterized by $c$. Notice that CLIQUE, when parameterized by $k$, is W[1]-complete on general graphs [20].

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 [20], fixed-parameter tractable on $c$-closed graphs? Koana et al. [43, 45] 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 when parameterized by $k + c$. In fact, by applying a "Buss-like" reduction rule [10], 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 (when parameterized by either $k + c$ or $k$ 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 algorithm for TDS that

[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 [53].

runs in time $(ck)^{\mathcal{O}(rk)}n^{\mathcal{O}(1)}$. A key ingredient in all these results was a polynomial bound for the Ramsey number on $c$-closed graphs. Koana et al. [43] 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 are fixed-parameter tractable 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. [43].

**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 $cl(G) = 1 + \max\{0, |N(u) \cap N(v)| \mid u, v \in V(G), uv \notin E(G)\}$, 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 $cl(G) \leq \Delta(G) + 1$. But the comparability ends there. As noted by Koana et al. [43], 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 [23, 29], 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, 55].

**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 $|D \cap N[v]| \geq 1$ 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 $|N[V']| \geq t$.

[2] Koana et al. [43] 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 [9, p. 486]. But for that matter, so is the term $k$-closure [8]. 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 an 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 $\mathcal{O}(k^{3(2^c-1)})$ vertices on the family of $c$-closed graphs.
2. CDS admits an algorithm that runs in time $2^{\mathcal{O}(c+k\log(ck))}n^{\mathcal{O}(1)}$. 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. [50].)
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 12). **(P2)** The dominating set $D$ must intersect every "large" maximal clique (Corollary 8). 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 $\ell$ distinct "large" maximal cliques, then $G$ contains an independent set of size $\ell$ as well (Lemma 9). 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. [43]. **(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 $M$, where $L$ is the set of vertices that belong to at least one large maximal clique and $M$ the set of remaining vertices. Thus, $L$ is the union (not necessarily disjoint) of at most $k$ large cliques, and the subgraph $G[M]$ contains no large clique or no independent set of size $k+1$. Therefore, by (R1), we will have $|M| = $ poly$(c,k)$. So we can guess the set of vertices from $M$ 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 $M$ 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 [34, 50, 51, 52]; 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 [11]. The inadmissibility of a polynomial kernel for CDS follows from a result due to Misra et al. [50], 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 11). 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 must point out that properties (P1) and (P2) have been used by Koana et al. [43] 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 11, 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 11 to fashion a reduction rule (Reduction Rule 44), which we use to bound the size of independent sets while designing our kernel for Perfect Code. The idea behind Reduction Rule 44 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 [21] on general graphs, and W[2]-hard even on bipartite graphs (and hence on triangle-free graphs) [56], but it is fixed-parameter tractable on graphs of girth at least 5 [56], planar graphs [1, 2, 26, 38], graphs of bounded genus [22], map graphs [18], $H$-minor free graphs [19] and graphs of bounded degeneracy [3]. The CDS problem is also known to be W[2]-hard on general graphs [21], but admits a polynomial kernel on planar graphs, and more generally, on apex-minor-free graphs [24, 32, 47]. The problem is FPT on graphs of bounded degeneracy [31]. Cygan et al. [16] showed that CDS has no polynomial kernel even on 2-degenerate graphs unless NP $\subseteq$ co-NP/poly. Misra et al. [50] 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. [25] showed that both Dominating Set and CDS admit linear kernels on graphs with excluded topological minors. We refer the reader to [25] 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 [13, 20], and remains W[1]-hard even on bipartite graphs of girth 4 [36], but admits a polynomial kernel on planar graphs [33] and graphs of girth at least 5 [36]. Dawar and Kreutzer [17] 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 [49].

The Partial Vertex Cover (PVC) problem, the "partial variant" of the Vertex Cover problem, asks if $t$ edges of a graph can be covered using $k$ vertices. Both PVC and PDS have been studied under the two natural parameterizations: by $k$ and by $t$. When parameterized by $k$, unlike the widely-studied Vertex Cover, PVC is W[1]-hard on general graphs [34], 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 is fixed-parameter tractable 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. Recently, Koana et al. [41] showed that PVC admits a kernel of size $k^{\mathcal{O}(c)}$ on $c$-closed graphs. 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 (parameterized by $k$) extends to PDS as well. In contrast to Dominating Set, PDS remains W[1]-hard even on graphs of bounded degeneracy [31]. But Amini et al. [4] showed that PDS is FPT on planar graphs, graphs of bounded genus and graphs of bounded maximum degree; these results, in fact, hold for a more general problem called Weighted Partial-$(k, r, t)$-Center. When parameterized by $t$, both PVC and PDS are FPT on general graphs [7, 12, 39, 40].

**Related work on $c$-closed graphs.** As mentioned earlier, Fox et al. [28] 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 [35] 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 $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. See the paper by Behera et al. [6] for an updated version of these results. This was soon followed by the work of Koana and Nichterlein [46], 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 [46] 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 [30] when parameterized by $c$, 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 $\ell, i$ 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 $\gamma$-closed graphs.** Along with $c$-closed graphs, Fox et al. [28] had also introduced a larger class of graphs called weakly $\gamma$-closed graphs. For a positive integer $\gamma$, a graph $G$ is weakly $\gamma$-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. [42] extended their result for Independent Set in [43] to weakly $\gamma$-closed graphs. They showed that Independent Set admits a polynomial kernel on weakly $\gamma$-closed graphs as well. In fact, they showed that a similar result holds for the $\mathcal{G}$-Subgraph problem, for every family $\mathcal{G}$ of graphs that is closed under subgraphs; in the $\mathcal{G}$-Subgraph problem, 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. [42] also showed that two variants of Dominating Set, namely, Independent Dominating Set and Dominating Clique, are FPT on weakly $\gamma$-closed graphs. But they left open the complexity of Dominating Set on weakly $\gamma$-closed graphs, which was recently shown to be FPT by Lokshtanov and Surianarayanan [48]. Koana et al. [42] also gave bounds and enumeration algorithms for various choices of "dense subgraphs" in weakly $\gamma$-closed subgraphs. See [42, Table 1] for an overview of their results. In a companion work, Koana et al. [44] studied Capacitated Vertex Cover, Connected Vertex Cover, and Induced Matching and obtained kernels of size $k^{\mathcal{O}(\gamma)}$, $k^{\mathcal{O}(\gamma)}$, and $(\gamma k)^{\mathcal{O}(\gamma)}$, respectively. They showed a kernel with $O(ck^2)$ vertices for Connected Vertex Cover, and showed lower bounds for the kernelization of Capacitated Vertex Cover, Independent Set, and Dominating Set.

## 2 Preliminaries

This section is divided into three parts. In Section 2.1, we introduce some notation and terminology that we will be using throughout the paper. We use Section 2.1 only to collect the frequently used notation and terms in one place. We will recap the definitions introduced here when we use them later on in the paper. In Section 2.2, we summarise the results due to Fox et al. [28] and Koana et al. [43] that we will be using. In Section 2.3, we prove a few preliminary lemmas that we will be relying on to prove our main results.

### 2.1 Notation and Terminology

**Sets and functions.** For a positive integer $\ell$, we denote the set $\{1, \ldots, \ell\}$ by $[\ell]$. Let $X, Y$ be two sets. By $X \setminus Y$ we denote the set $\{x \in X \mid x \notin Y\}$. 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 $a, b \in \mathbb{N}$.

**Graphs.** 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 neighbourhood and closed neighbourhood of $G$. 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., $N(V') = (\bigcup_{v \in V'} N_G(v)) \setminus V'$ 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 $x, y \in V(G)$, $\text{dist}_G(x,y)$ 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 $G$. A clique in $G$ is a complete subgraph of $G$. An independent set in $G$ is a set of pairwise non-adjacent vertices. 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 $\ell$. We also define two vertex subsets as follows: $L^\ell(G) = \bigcup_{Q \in \mathcal{Q}^\ell(G)} V(Q)$, and $M^\ell(G) = V(G) \setminus L^\ell(G)$. 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 $\ell$, and $M^\ell(G)$ contains the remaining vertices. Notice that $\{L^\ell(G), M^\ell(G)\}$ is a partition of $V(G)$ (with one of the parts possibly being empty).

Let $G$ be a graph and $\mathcal{H}$ a family of subgraphs of $G$. By $I^2(\mathcal{H})$, we denote the set of vertices in $G$ that belong to at least two graphs in $\mathcal{H}$, i.e., $I^2(\mathcal{H}) = \bigcup_{\substack{H_1, H_2 \in \mathcal{H} \\ H_1 \neq H_2}} (V(H_1) \cap V(H_2))$. With a slight abuse of terminology, we say that the family $\mathcal{H}$ is disjoint if the graphs in $\mathcal{H}$ are pairwise vertex-disjoint, i.e., if $I^2(\mathcal{H}) = \emptyset$.

We assume a basic familiarity with concepts in parameterized complexity such as fixed-parameter tractability, kernelization and W[1]-hardness. We do not define these terms here, and refer the reader to the book by Cygan et al. [15] for an introduction to parameterized complexity.

## 2.2 Summary of Results from [28] and [43]

In this section, we briefly summarise the results due to Fox et al. [28] and Koana et al. [43] that we will be using throughout this paper. Following the notation of Koana et al. [43], 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** ([43]). *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 [43, 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. [43] 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** ([43]). *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 $k$.*

Note that Lemma 3 immediately leads to an algorithm to solve the Independent Set problem on $c$-closed graphs in time $2^{\mathcal{O}(k\log(ck))}n^{\mathcal{O}(1)}$.

▶ **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.*

**Proof.** Given $G$ and $k$, we first run the polynomial time algorithm in Lemma 3 and compute $G'$, as described in Lemma 3. If $|V(G')| > ck^2$, then we return any maximal independent set in $G'$, which can be found in polynomial time. Otherwise $|V(G')| \leq ck^2$, and we do as follows. We go over all $k$-sized subsets of $V(G')$, and check if any of them is an independent set; and if there exists an independent set $I \subseteq V(G')$ with $|I| = k$, then we return $I$, and otherwise we return that $G$ has no independent set of size $k$. Note that since $G'$ has at most $ck^2$ vertices, the last step only takes time $\binom{ck^2}{k} \cdot (ck^2)^{\mathcal{O}(1)} = c^k \cdot k^{2k} \cdot (ck^2)^{\mathcal{O}(1)} = 2^{k\log c} \cdot 2^{2k\log k} \cdot (ck^2)^{\mathcal{O}(1)} = 2^{\mathcal{O}(k\log(ck))}(ck^2)^{\mathcal{O}(1)}$. Thus the total runtime of the procedure is bounded by $n^{\mathcal{O}(1)} + 2^{\mathcal{O}(k\log(ck))}(ck^2)^{\mathcal{O}(1)} \leq 2^{\mathcal{O}(k\log(ck))}n^{\mathcal{O}(1)}$.

The correctness of the procedure follows from property (ii) in the statement of Lemma 3, and the fact that since $G'$ is an induced subgraph of $G$, any independent set in $G'$ is also an independent set in $G$ and vice versa. ◀

Fox et al. [28] 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** ([28]). *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 $G$.*

## 2.3 Some Preliminary Lemmas

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

▶ **Lemma 6.** *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., $|N(v) \cap V(Q)| \leq c-1$.*

**Proof.** If $v \in V(G) \setminus V(Q)$ has at least $c$ neighbours in $V(Q)$, then for any $u \in V(Q) \setminus N(v)$, $u$ and $v$ have at least $c$ common neighbours. This implies that $u$ and $v$ must be adjacent for every $u \in V(Q)$, which contradicts the maximality of $Q$. ◀

Lemma 6 immediately implies that two maximal cliques in a $c$-closed graph can intersect in at most $c-1$ vertices.

▶ **Corollary 7.** *Let $G$ be a $c$-closed graph, and let $Q_1$ and $Q_2$ be two distinct maximal cliques in $G$. Then, $|V(Q_1) \cap V(Q_2)| \leq c-1$.*

**Proof.** Since $Q_1$ and $Q_2$ are distinct maximal cliques, there exists a vertex $v \in V(Q_1) \setminus V(Q_2)$. Now, if $|V(Q_1) \cap V(Q_2)| \geq c$, it would imply that $|N(v) \cap V(Q_2)| \geq c$, which by Lemma 6 is not possible. ◀

Another immediate consequence of Lemma 6 is that in a $c$-closed graph $G$, every "small" dominating set of $G$ must intersect every "large" clique in $G$. We formally prove this below.

▶ **Corollary 8.** *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, $D \cap V(C) \neq \emptyset$.*

**Proof.** Since $D$ is a dominating set of $G$, $D$ dominates every vertex of $G$. In particular, $D$ dominates $V(C)$. By Lemma 6, every vertex $v \in D \setminus V(C)$ can dominate at most $c-1$ vertices of $C$. Since $|D \setminus V(C)| \leq |D| \leq k$, $D \setminus V(C)$ dominates at most $(c-1)k$ vertices of $C$. And since $|V(C)| \geq (c-1)k+1$, we must have $D \cap V(C) \neq \emptyset$. ◀

We now show that if a $c$-closed graph $G$ contains sufficiently many large maximal cliques, then $G$ contains a sufficiently large independent set as well. Recall that for $\ell > 0$, $\mathcal{Q}^{\ell}(G)$ denotes the set of all maximal cliques of size at least $\ell$ in $G$; and for integers $a$ and $b$, we defined $\beta(a,b) = 2[(a-1)(b-1)+1]$.

▶ **Lemma 9.** *Let $\ell$ 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 $\ell$. 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 $\ell$-sized independent set in $G$.*

**Proof.** Let $Q_1, Q_2, \ldots, Q_\ell \in \mathcal{Q}^{\beta(c,\ell)}(G)$ be distinct. For each $j \in [\ell]$, let $X_j = \{v \in V(Q_j) \mid v \in V(Q_i) \text{ for some } i \in [\ell] \setminus \{j\}\}$. That is, $X_j = \bigcup_{i \in [\ell] \setminus j} (V(Q_i) \cap V(Q_j))$. Note that by Corollary 7, we have $|X_j| \leq (c-1)(\ell-1)$.

We construct an $\ell$-sized independent set $I$ as follows. Pick an arbitrary vertex $v_1$ from $V(Q_1) \setminus X_1$ into $I$. For $j = 2, 3, \ldots, \ell$, pick a vertex $v_j$ from $V(Q_j) \setminus (X_j \cup \bigcup_{i<j} N(v_i))$. Note that for each $j$, we have $|X_j| \leq (c-1)(\ell-1)$, and for each $i < j$, by Lemma 6, $|N(v_i) \cap V(Q_j)| \leq c-1$, and therefore, $|(\bigcup_{i<j} N(v_i)) \cap V(Q_j)| \leq (c-1)(i-1) \leq (c-1)(\ell-1)$. Thus, $|X_j \cup \bigcup_{i<j} (N(v_i) \cap V(Q_j)| \leq 2(c-1)(\ell-1)$. We thus have $V(Q_j) \setminus (X_j \cup \bigcup_{i<j} N(v_i)) \neq \emptyset$, as $|V(Q_j)| \geq \beta(c,\ell) > 2(c-1)(\ell-1)$, and therefore, we can always pick a $v_j$ as required. Moreover, by definition, $v_j \notin N(v_i)$ for $i < j$, and thus the set $I = \{v_1, v_2, \ldots, v_\ell\}$ we constructed is indeed an independent set.

Finally, observe that the procedure described above to construct $I$ can be executed in polynomial time, when $G$ and the cliques $Q_1, Q_2, \ldots, Q_\ell$ are given as input, which leads to the algorithm required by the statement of the lemma. (The fact that $G$ contains an independent set of size $\ell$ implies that $\ell \leq |V(G)|$, and therefore the dependence of the runtime on $\ell$ is also bounded by a polynomial function of $|V(G)|$.) ◀

▶ **Lemma 10.** *Let $\ell$ 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, every independent set in $G$ has size at most $\ell$.*

**Proof.** Let $I \subseteq V(G)$ be an independent set in $G$. Note that for every $i \in [\ell]$, we have $|I \cap V_i| \leq 1$, as $V_i$ induces a clique, and $I$ is an independent set. Then, as $V(G) = \bigcup_{i \in [\ell]} V_i$, we get $I = \bigcup_{i \in [\ell]} (I \cap V_i)$, which implies that $|I| \leq \ell$. ◀

The following lemma says that given a $c$-closed graph $G$ and an integer $\ell$, in polynomial time, we can either find an independent set of size $\ell$ or conclude that every independent set has size $\mathcal{O}(c \cdot \ell^2)$. Recall that we defined $\beta(c,\ell) = 2[(c-1)(\ell-1)+1]$; $\mathcal{Q}^{\beta(c,\ell)}(G)$ to be the set of all maximal cliques of size at least $\beta(c,\ell)$ in $G$; $L^{\beta(c,\ell)}(G)$ to be the set of all vertices in $G$ that belong to at least one maximal clique of size at least $\beta(c,\ell)$, i.e., $L^{\beta(c,\ell)}(G) = \bigcup_{Q \in \mathcal{Q}^{\beta(c,\ell)}(G)} V(Q)$; and $M^{\beta(c,\ell)}(G) = V(G) \setminus L^{\beta(c,\ell)}(G)$.

▶ **Lemma 11.** *There is an algorithm that, given an $n$-vertex $c$-closed graph $G$ and a positive integer $\ell$ as input, runs in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$, and either returns an independent set of size at least $\ell$, or correctly reports 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 $\ell$ as input, our algorithm works as follows. We first use the algorithm in Lemma 5 to construct $\mathcal{Q}^{\beta(c,\ell)}(G)$ in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$. If $|\mathcal{Q}^{\beta(c,\ell)}(G)| \geq \ell$, then we return an $\ell$-sized independent set constructed using the algorithm in Lemma 9.

Otherwise we construct the sets $L^{\beta(c,\ell)}(G)$ and $M^{\beta(c,\ell)}(G)$. By the definition of the sets $L^{\beta(c,\ell)}(G)$ and $M^{\beta(c,\ell)}(G)$, the induced subgraph $G' = G[M^{\beta(c,\ell)}(G)]$ 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 $\ell$. Thus, if $|V(G')| \geq R_c(\beta(c,\ell),\ell)$, then 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 $\ell$.

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

To see the correctness of the last step, assume that $|\mathcal{Q}^{\beta(c,\ell)}(G)| \leq \ell - 1$ and $|V(G')| = |M^{\beta(c,\ell)}(G)| \leq R_c(\beta(c,\ell),\ell) - 1$. Note that by definition, $L^{\beta(c,\ell)}(G) = \bigcup_{Q \in \mathcal{Q}^{\beta(c,\ell)}(G)} V(Q)$, i.e., a union of cliques. Therefore, by Lemma 10, 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 $\left\{L^{\beta(c,\ell)}(G), M^{\beta(c,\ell)}(G)\right\}$ is a partition of $V(G)$, for any independent set $I \subseteq V(G)$, we have $|I| = |I \cap L^{\beta(c,\ell)}(G)| + |I \cap M^{\beta(c,\ell)}(G)| \leq (\ell-1) + |M^{\beta(c,\ell)}(G)| \leq (\ell-1) + R_c(\beta(c,\ell),\ell) - 1$.

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

Recall that for $V' \subseteq V(G)$, we defined $CN(V')$ to be the set of common neighbours of the vertices in $V'$, i.e., $CN(V') = \bigcap_{v \in V'} N(v)$. Also, for $V' \subseteq V(G)$ with $|V'| \geq 2$, we defined $N^{[2]}(V')$ to be the union of the sets of common neighbours of every pair of vertices in $V'$, i.e., $N^{[2]}_G(V') = (\bigcup_{\substack{u,v \in V' \\ u \neq v}} CN(\{u,v\})) \setminus V'$. The next lemma says that if $D$ is a dominating set of size at most $k$ and $I$ is an independent set of size $k+1$, then there exists a vertex in $D$ that dominates at least two vertices of $I$. In other words, $D$ must intersect $N^{[2]}(I)$.

▶ **Lemma 12.** *Let $G$ be a graph and $k$ a non-negative integer. Let $I$ be an independent set in $G$ of size $k+1$. For a dominating set $D$ of $G$, if $|D| \leq k$, then $D \cap N^{[2]}(I) \neq \emptyset$. Moreover, if $G$ is $c$-closed, then $|N^{[2]}(I)| \leq (c-1)\binom{k+1}{2}$.*

**Proof.** Assume that $D$ is a dominating set of size at most $k$. Then, since $|I| = k+1$, by the pigeonhole principle, there exists a vertex $v \in D$ and a pair of distinct vertices $u, u' \in I$ such that $v$ dominates both $u$ and $u'$, i.e., $v \in N[u] \cap N[u']$. Note that since $uu' \notin E(G)$ as $I$ is an independent set, it follows that $v \neq u$ and $v \neq u'$. And thus, $v \in N(u) \cap N(u')$, which implies that $v \in N^{[2]}(I)$. Now, if $G$ is $c$-closed, then by the definition of $c$-closed graphs, we have $|N(u) \cap N(u')| \leq c-1$, as $uu' \notin E(G)$. This implies that $|N^{[2]}(I)| \leq |\bigcup_{\substack{u,u' \in I \\ u \neq u'}} N(u) \cap N(u')| \leq (c-1)\binom{k+1}{2}$. ◀

We conclude this section with the following lemma, which says that for a $c$-closed graph $G$ and $Y \subseteq V(G)$ of size at most $c-1$, the common neighbours of $Y$ induces a $(c-|Y|)$-closed graph.

▶ **Lemma 13.** *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.*

**Proof.** Let $G' = G[CN(Y)]$. Consider a pair of distinct vertices $u, v \in V(G')$. Since $G'$ is a subgraph of $G$, we have $N_G(u) \supseteq N_{G'}(u)$ and $N_G(v) \supseteq N_{G'}(v)$, and thus $CN_G(\{u,v\}) \supseteq CN_{G'}(\{u,v\})$. Also, since $u, v \in CN_G(Y)$, we have $CN_G(\{u,v\}) \supseteq Y$. Thus, $CN_G(\{u,v\}) \supseteq CN_{G'}(\{u,v\}) \cup Y$. Also, note that since $V(G') \cap Y = \emptyset$, we have $CN_{G'}(\{u,v\}) \cap Y = \emptyset$, and therefore, $|CN_{G'}(\{u,v\}) \cup Y| = |CN_{G'}(\{u,v\})| + |Y|$.

Now, assume that $|CN_{G'}(\{u,v\})| \geq c - |Y|$. Then, from the previous observations, we get that $|CN_G(\{u,v\})| \geq |CN_{G'}(\{u,v\}) \cup Y| \geq c - |Y| + |Y| = c$. Then, as $G$ is $c$-closed, we have $uv \in E(G)$, which implies that $uv \in E(G')$ as well. ◀

## 3 Perfect Code on c-Closed Graphs

A perfect code of a graph $G$ is a dominating set of $G$ that dominates every vertex of $G$ exactly once. That is, $D \subseteq V(G)$ is a perfect code if $|N[v] \cap D| = 1$, for every $v \in V(G)$. Note that the definition immediately implies that for a perfect code $D$, and for every pair of distinct vertices $x, y \in D$, we have $\text{dist}_G(x,y) \geq 3$. If $xy \in E(G)$, then $x, y \in N[x] \cap D$, and if $G$ contains a path $xvy$ then $x, y \in N[v] \cap D$, neither of which is possible. The PERFECT CODE problem, which we formally define below, asks if a given graph contains a perfect code of a certain size.

> PERFECT CODE **Parameter:** $k + cl(G)$
> **Input:** An undirected graph $G$ and a non-negative integer $k$.
> **Question:** Does $G$ have a perfect code of size at most $k$?

In this section, we show that PERFECT CODE admits an algorithm on $c$-closed graphs that runs in time $2^{\mathcal{O}(c+k\log(ck))}n^{\mathcal{O}(1)}$. Moreover, we show that for each fixed positive integer $c$, the PERFECT CODE problem on $c$-closed graphs admits a kernel with $\mathcal{O}(k^{3(2^c-1)})$ vertices.

To design our algorithm and 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. We 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 $W = \emptyset$.

For future reference, we record below the following observation that will be used throughout this section.

▶ **Observation 14.** *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 first develop some preparatory results that will be useful for both our algorithm and kernel. We begin by introducing a reduction rule, which says that if two vertices have the same closed neighbourhood and have the same colour, then we can safely delete one of them.

▶ **Reduction Rule 15.** *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 $x$.*

▶ **Lemma 16.** *Reduction Rule 15 is safe.*

**Proof.** Informally, the reduction rule is safe because $N_G[x] = N_G[y]$, and therefore, a vertex $v \in V(G)$ dominates $x$ if and only if $v$ dominates $y$. We now prove this formally. Let $x, y \in V(G)$ be such that $x \neq y$ and $N_G[x] = N_G[y]$. Let $x, y \in B$ or $x, y \in W$, and the graph $G' = G - x$ be obtained by a single application of Reduction Rule 15. We prove the safeness of the rule by showing that $((G, B, W), k)$ is a yes-instance of BW-Perfect Code if and only if $((G', B \setminus \{x\}, W \setminus \{x\}), k)$ is a yes-instance of BW-Perfect Code.

Assume that $((G, B, W), k)$ is a yes-instance of BW-Perfect Code, and let $D$ be a bw-perfect code of $(G, B, W)$ of size at most $k$. If $x \notin D$, then clearly $D$ is a perfect code of $G'$ as well. So assume that $x \in D$. This means that $x \in B$, and therefore, by assumption, $y \in B$. Observe that since $N_G[x] = N_G[y]$, we have $xy \in E(G)$. Then, by Observation 14, we have $y \notin D$. We claim that $D' = (D \setminus \{x\}) \cup \{y\}$ is a bw-perfect code of $G'$. Note that for every $v \in V(G') \setminus N_{G'}[y]$, we have $N_G[v] = N_{G'}[v]$. Therefore, $D' \cap N_{G'}[v] = ((D \setminus \{x\}) \cup y) \cap N_G[v] = D \cap N_G[v]$. Now, since $D$ is a bw-perfect code of $G$, we have $|N_G[v] \cap D| = 1$, which implies that $|N_{G'}[v] \cap D'| = 1$. Now, for every $v \in N_{G'}[y]$, note that $D \cap N_G[v] = \{x\}$, and therefore, $(D \setminus \{x\}) \cap N_{G'}[v] = \emptyset$. Thus, $|D' \cap N_{G'}[v]| = |\{y\}| = 1$, which proves that $D'$ is a bw-perfect code of $G'$ of size at most $k$.

Conversely, assume that $((G', B \setminus \{x\}, W \setminus \{x\}), k)$ is a yes-instance of BW-Perfect Code, and let $D''$ be a bw-perfect code of $G'$ of size at most $k$. We claim that $D''$ is a perfect code of $G$ as well. Note that for every vertex $v \in V(G) \setminus \{x\}$, we have $N_{G'}[v] = N_G[v] \setminus \{x\}$, and therefore, $|D'' \cap N_G[v]| = |D'' \cap N_{G'}[v]| = 1$. Now, by the definition of a perfect code, there exists a unique $w \in N_{G'}[y]$ such that $D'' \cap N_{G'}[y] = \{w\}$. And note that since $N_G[x] = N_G[y]$, we have $w \in N_G[x]$. Thus, $|D'' \cap N_G[x]| = |\{w\}| = 1$. This proves that $D''$ is a bw-perfect code of $G$ as well. ◀

▶ Remark 17. Note that Reduction Rule 15 can be applied in polynomial time, and will be applied to an instance $((G, B, W), k)$ at most $|V(G)| - 1$ times. So, from now on, whenever considering an instance of $((G, B, W), k)$ of BW-Perfect Code, we assume that Reduction Rule 15 has been applied exhaustively to $((G, B, W), k)$.

The following lemma says that (when Reduction Rule 15 is no longer applicable), any maximal clique $Q$ in $G$ can contain at most two vertices that do not have neighbours in $V(G) \setminus V(Q)$.

▶ **Lemma 18.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code. *For any maximal clique $Q$ in $G$, we have $|V(Q) \setminus \bigcup_{v \in V(G) \setminus V(Q)} N(v)| \leq 2$.*

**Proof.** By Remark 17, Reduction Rule 15 has been applied exhaustively to $((G, B, W), k)$. Now, assume that the lemma is not true. Let $Q$ be a maximal clique in $G$ such that $|V(Q) \setminus \bigcup_{v \in V(G) \setminus V(Q)} N(v)| \geq 3$. That is, there exist three distinct vertices, say $x_1, x_2, x_3 \in V(Q)$, such that $N[x_i] = V(Q)$ for $i \in [3]$. Note that $N[x_i] = N[x_j]$ for every $\{i, j\} \subseteq [3]$. And at least two of $x_1, x_2$ and $x_3$ must be black or at least two of them must be white. But this is not possible as Reduction Rule 15 has been applied exhaustively to $((G, B, W), k)$. ◀

We now focus specifically on $c$-closed graphs. In the rest of this section, whenever we consider an instance of $((G, B, W), k)$ of BW-Perfect Code, we assume that $G$ is a $c$-closed graph.

Recall that for integers $a$ and $b$, we defined $\alpha(a, b) = (a - 1)b + 1$. In the next three lemmas, we explore how a bw-perfect code of size at most $k$ interacts with "large" maximal cliques. In this section, by a large clique, we mean a clique of size at least $\alpha(c, k)$. We have already shown in Corollary 8 that every dominating set of size at most $k$ must intersect every large maximal clique. The next lemma shows that every bw-perfect code of size at most $k$ must intersect every large maximal clique in exactly one vertex. Recall that $\mathcal{Q}^{\alpha(c,k)}(G)$ denotes the set of all maximal cliques of size at least $\alpha(c, k)$ in $G$.

▶ **Lemma 19.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code*, and $D \subseteq B$ a bw-perfect code of $(G, B, W)$ of size at most $k$. Then, for every $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$, we have $|V(Q) \cap D| = 1$.*

**Proof.** Since $D$ is a bw-perfect code of $G$, by Observation 14, $D$ is a dominating set of $G$. Then, by Corollary 8, $|V(Q) \cap D| \geq 1$. But again by Observation 14, $D$ must be an independent set, and since $Q$ is a clique, $D$ can intersect $Q$ in at most 1 vertex. And the lemma follows. ◀

As an immediate consequence of Lemma 19, we derive the following corollary, which says that if two distinct large maximal cliques intersect, then exactly one vertex from their intersection must belong to every bw-perfect code of size at most $k$.

▶ **Corollary 20.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code*, and $D \subseteq B$ a bw-perfect code of $(G, B, W)$ of size at most $k$. Let $Q_1, Q_2 \in \mathcal{Q}^{\alpha(c,k)}(G)$ be distinct and $V(Q_1) \cap V(Q_2) \neq \emptyset$. Then there exists $v \in V(Q_1) \cap V(Q_2)$ such that $V(Q_1) \cap D = V(Q_2) \cap D = \{v\}$.*

**Proof.** Lemma 19 implies that $|V(Q_i) \cap D| = 1$ for $i \in [2]$. Let $\{v_i\} = V(Q_i) \cap D$, for $i \in [2]$. We claim that $v_1 = v_2$. Suppose not. Note that $V(Q_1) \cap V(Q_2) \neq \emptyset$. Then for every $w \in V(Q_1) \cap V(Q_2)$, we have $v_1, v_2 \in N[w] \cap D$, which, by the definition of a perfect code, is not possible. ◀

The following lemma says that every perfect code of size at most $k$ must necessarily exclude vertices that are endpoints of edges between different large maximal cliques. It is essentially a consequence of property (ii) in Observation 14.

▶ **Lemma 21.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code*, and let $D$ be a bw-perfect code of $(G, B, W)$ of size at most $k$. Let $Q_1, Q_2 \in \mathcal{Q}^{\alpha(c,k)}(G)$. Then, for any $x \in V(Q_1) \setminus V(Q_2)$ and $y \in V(Q_2) \setminus V(Q_1)$ such that $xy \in E(G)$, we have $D \cap \{x, y\} = \emptyset$.*

**Proof.** Since $Q_1, Q_2 \in \mathcal{Q}^{\alpha(c,k)}(G)$, we have $|V(Q_i)| \geq (c-1)k + 1$, for $i \in [2]$. Then, since $D$ is a bw-perfect code of size at most $k$, Lemma 19 implies that $|V(Q_i) \cap D| = 1$ for $i \in [2]$. Let $\{v_i\} = V(Q_i) \cap D$ for $i \in [2]$. Note that to prove the lemma, it is sufficient to prove that $v_1 \neq x$ and $v_2 \neq y$. Assume for a contradiction that $v_1 = x$. Note that $v_1 = x \neq y$, as $v_1 = x \in V(Q_1) \setminus V(Q_2)$. Then, $v_1 y v_2$ is path of length 2 if $v_2 \neq y$, and $(x =) v_1 v_2 (= y)$ is a path of length 1 if $y = v_2$. In either case, we have $\text{dist}(v_1, v_2) \leq 2$, which, by Observation 14, is not possible. By reversing the roles of $Q_1$ and $Q_2$, we can conclude that $y \notin D$ as well. ◀

**Notation.** Consider a bw-graph $(G, B, W)$ and a vertex $v \in V(G)$. By $(G_v, B_v, W_v)$, we denote the bw-graph obtained by deleting $N_G[v]$ from $G$, and by colouring all neighbours of $N_G(v)$ white. That is, $G_v = G - N_G[v]$, $W_v = (W \setminus N_G[v]) \cup N_G(N_G(v))$, and $B_v = V(G_v) \setminus W_v$. Recall that $L^{\alpha(c,k)}(G) = \bigcup_{Q \in \mathcal{Q}^{\alpha(c,k)}(G)} V(Q)$ and $M^{\alpha(c,k)}(G) = V(G) \setminus L^{\alpha(c,k)}(G)$. That is, $L^{\alpha(c,k)}(G)$ contains all the vertices in $G$ that belong to at least one maximal clique of size at least $\alpha(c,k)$, and $M^{\alpha(c,k)}(G)$ contains the remaining vertices. Now, for each $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$, 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 $Q' \in \mathcal{Q}^{\alpha(c,k)}(G)$ and $u \notin V(Q')\}$; and $Z(G) = \bigcup_{Q \in \mathcal{Q}^{\alpha(c,k)}(G)} Z(Q)$. Notice that in the definition of $Z(Q)$, the condition $u \notin V(Q')$, in fact, implies that $Q \neq Q'$. For $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$ and a set $S \subseteq M^{\alpha(c,k)}(G)$ such that $N_G[S] \subseteq M^{\alpha(c,k)}(G)$, let $Y(Q, S) \subseteq V(Q)$ be the set of vertices $u$ in $Q$ such that $u$ has a common neighbour with some vertex in $S$, i.e., $Y(Q, S) = \{u \in V(Q) \mid$ there exist $v \in V(G)$ and $w \in S$ such that $uv, vw \in E(G)\}$; and $Y(G, S) = \bigcup_{Q \in \mathcal{Q}^{\alpha(c,k)}(G)} Y(Q, S)$. We may think of the vertices of $Z(G)$ and $Y(G, S)$ as forbidden vertices—the vertices that cannot belong to a bw-perfect code (that contains $S$); we will prove this formally. The following corollary follows immediately from Lemma 21 and the definition of $Z(G)$.

▶ **Corollary 22.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code*, and let $D$ be a bw-perfect code of $(G, B, W)$ of size at most $k$. Then $Z(G) \cap D = \emptyset$.*

## 3.1 FPT Algorithm for Perfect Code on $c$-Closed Graphs

In this subsection, we focus exclusively on designing our algorithm for Perfect Code. We continue with proving structural results that explore the properties of a bw-perfect code. The first of these results says that if $D$ is a bw-perfect code of size at most $k$, then the intersection of $D$ with $M^{\alpha(c,k)}(G)$ does not dominate any vertex of $L^{\alpha(c,k)}(G)$.

▶ **Lemma 23.** *Let $D$ be a bw-perfect code of $(G, B, W)$ of size at most $k$, and let $S = D \cap M^{\alpha(c,k)}(G)$. Then, $N_G[S] \subseteq M^{\alpha(c,k)}(G)$.*

**Proof.** Suppose not. Then $N_G[S] \cap L^{\alpha(c,k)}(G) \neq \emptyset$. That is, there exists a maximal clique $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$ such that $N_G[S] \cap V(Q) \neq \emptyset$. Let $v \in N_G[S] \cap V(Q)$. Since $v \in N_G[S] \cap L^{\alpha(c,k)}(G)$, we have $v \notin S$, as $S \subseteq M^{\alpha(c,k)}(G)$. Then, since $v \in N_G[S]$, there exists $u \in S$ such that $uv \in E(G)$. Now, by Lemma 19, $|V(Q) \cap D| = 1$. Let $\{w\} = V(Q) \cap D$. Then, $u, w \in N_G[v] \cap D$, which is not possible. ◀

Recall that for a large maximal clique $Q$ and $S \subseteq M^{\alpha(c,k)}(G)$ with $N_G[S] \subseteq M^{\alpha(c,k)}(G)$, we defined $Y(Q, S)$ to be the set of vertices $u \in V(Q)$ such that $u$ has a common neighbour with some vertex in $S$. The next lemma says that no vertex from $Y(Q, S)$ can belong to a bw-perfect code of size at most $k$.

▶ **Lemma 24.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code*, and let $D$ be a bw-perfect code of $(G, B, W)$ of size at most $k$. Let $S = D \cap M^{\alpha(c,k)}(G)$. Then for every $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$, we have $D \cap Y(Q, S) = \emptyset$.*

**Proof.** Observe first that by Lemma 23, $N_G[S] \subseteq M^{\alpha(c,k)}(G)$, and therefore $Y(Q,S)$ is well-defined for every $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$. Assume that the lemma is not true, and let $u \in D \cap Y(Q,S)$ for some $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$. Then there exist vertices $v, w$ such that $w \in S$ and $uv, vw \in E(G)$. Notice that $u \neq w$ as $u \in V(Q) \subseteq L^{\alpha(c,k)}(G)$ and $w \in S \subseteq M^{\alpha(c,k)}(G)$. We thus have two distinct vertices $u, w \in N_G[v] \cap D$, which contradicts the assumption that $D$ is a bw-perfect code. ◀

Recall that for a vertex $v \in V(G)$, we defined $(G_v, B_v, W_v)$ to be the bw-graph obtained from $(G, B, W)$ by deleting $N_G[v]$ and by colouring $N_G(N_G(v))$ white. We will use the following lemma to prove the correctness of our algorithm.

▶ **Lemma 25.** *Let $(G, B, W)$ be a bw-graph, and let $D \subseteq B$. Then, $D$ is a bw-perfect code of $(G, B, W)$ if and only if $D \setminus \{v\}$ is a bw-perfect code of $(G_v, B_v, W_v)$ for every $v \in D$.*

**Proof.** Fix $v \in D$. Assume first that $D$ is a bw-perfect code of $(G, B, W)$. To prove that $D \setminus \{v\}$ is a bw-perfect code of $(G_v, B_v, W_v)$, we need to prove that $D \setminus \{v\} \subseteq B_v$ and that $D \setminus \{v\}$ dominates every vertex of $G_v$ exactly once, i.e., $|N_{G_v}[w] \cap (D \setminus \{v\})| = 1$ for every $w \in V(G_v)$. Consider $u \in D \setminus \{v\}$. Then $u \in B$, which means that $u \notin W$. And by Observation 14, we have $\text{dist}_G(u, v) \geq 3$, which implies that $u \notin N_G[v] \cup N_G(N_G(v))$. Therefore $u \notin W_v$, which implies that $u \in B_v$. Thus, $D \setminus \{v\} \subseteq B_v$. Now, consider $w \in V(G_v)$. Then, since $D$ is a bw-perfect code of $(G, B, W)$, there exists a unique vertex $x \in D$ such that $x$ dominates $w$, i.e., $N_G[w] \cap D = \{x\}$. Notice that $x \neq v$, as $w \in V(G_v) = V(G) \setminus N_G[v]$; and hence $x \in D \setminus \{v\}$. In fact, $x \notin N_G[v]$, for otherwise, we would have $x, v \in N_G[v] \cap D$, which, by the definition of a bw-perfect code, is not possible. Thus $x \in N_{G_v}[w]$; that is, $x$ dominates $w$ in the graph $G_v$ as well. Since $G_v$ is a subgraph of $G$, we have $N_{G_v}[w] \subseteq N_G[w]$. We thus have $N_{G_v}[w] \cap (D \setminus \{v\}) = \{x\}$. As $w$ is an arbitrary element of $V(G_w)$, we can conclude that $D \setminus \{v\}$ is a perfect code of $(G_v, B_v, W_v)$.

Conversely, assume that $D \setminus \{v\}$ is a bw-perfect code of $(G_v, B_v, W_v)$. By assumption, $D \subseteq B$. Therefore, to prove that $D$ is a perfect code of $(G, B, W)$, we only need to prove that $D$ dominates every vertex of $G$ exactly once. So consider $w' \in V(G)$. We will prove that $|N_G[w'] \cap D| = 1$. Suppose first that $w' \notin N_G[v]$. Then $w' \in V(G_v)$, and there exists a unique vertex $y \in D \setminus \{v\}$ that dominates $w'$. That is, $N_{G_v}[w'] \cap (D \setminus \{v\}) = \{y\}$. If $N_G[w'] = N_{G_v}[w]$, then since $w' \notin N_G[v]$, we can immediately conclude that $N_G[w'] \cap D = \{y\}$. So suppose that there exists $y' \in N_G[w'] \setminus N_{G_v}[w']$. We claim that $y' \notin D$, which will imply that $N_G[w'] \cap D = \{y\}$. By the definitions of $G_v$ and $y'$, we have $y' \in N_G[v]$. Then $y' \notin D \setminus \{v\}$ as $D \setminus \{v\} \subseteq B_v \subseteq V(G_v) = V(G) \setminus N_G[v]$. Since $w' \notin N_G[v]$, we can conclude that $y' \neq v$, which implies that $y' \notin D$. We thus have $N_G[w'] \cap D = \{y\}$. Now, suppose that $w' \in N_G[v]$. We will show that $v$ is the only vertex in $D$ that dominates $w'$. First, since $D \setminus \{v\}$ is a bw-perfect code of $(G_v, B_v, W_v)$, we have $D \setminus \{v\} \subseteq B_v$, and by the definition of $B_v$, we have $B_v \cap N_G[v] = \emptyset$. Therefore, $w' \notin D \setminus \{v\}$. Now, consider $w'' \in N_G(w')$. If $w'' \in N_G[v]$, then again, we have $w'' \notin D \setminus \{v\}$. So suppose that $w'' \in N_G(w') \setminus N_G[v]$. Then, $w'' \in N_G(N_G(v))$, which implies that $w'' \in W_v$, and therefore, $w'' \notin D \setminus \{v\}$. Therefore, $N_G[w'] \cap (D \setminus \{v\}) = \emptyset$ and hence $|N_G[w'] \cap D| = |\{v\}| = 1$. ◀

We now prove the following lemma, which says that if $I$ is an independent set of size $k+1$ in $G$, then every bw-perfect code of $(G, B, W)$ must contain a vertex that dominates at least 2 vertices of $I$. Recall that 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., $N_G^{[2]}(V') = (\bigcup_{\substack{u,v \in V' \\ u \neq v}} (N_G(u) \cap N_G(v)) \setminus V'$.

▶ **Lemma 26.** *Let $((G,B,W),k)$ be an instance of* BW-Perfect Code*, and let $I$ be an independent set of size $k+1$ in $G$. Then, $((G,B,W),k)$ is a yes-instance of* BW-Perfect Code *if and only if $((G_v,B_v,W_v),k-1)$ is a yes-instance for some $v \in N^{[2]}(I) \cap B$.*

**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 $k$. Then, by Observation 14, $D$ is a dominating set of $G$, and therefore, by Lemma 12, $D \cap N^{[2]}(I) \neq \emptyset$. Let $v \in D \cap N^{[2]}(I)$. Then, $|D \setminus \{v\}| \leq k-1$, and by Lemma 25, $D \setminus \{v\}$ is bw-perfect code of $(G_v,B_v,W_v)$, which proves that $((G_v,B_v,W_v),k-1)$ is a yes-instance.

Conversely, assume that $((G_v,B_v,W_v),k-1)$ is a yes-instance of BW-Perfect Code for some $v \in N^{[2]}(I)$, and let $D' \subseteq B_v$ be a bw-perfect code of $(G_v,B_v,W_v)$ of size at most $k-1$. Then again, by Lemma 25, $D' \cup \{v\}$ is a bw-perfect code of $(G,B,W)$ of size at most $k$, which proves that $((G,B,W),k)$ is a yes-instance. ◀

The following lemma says that if $Q_1, Q_2$ are two distinct large maximal cliques that intersect each other, then every bw-perfect code of $G$ must contain a vertex from the intersection of $Q_1$ and $Q_2$.

▶ **Lemma 27.** *Let $((G,B,W),k)$ be an instance of* BW-Perfect Code*, and let $\{Q_1,Q_2\} \subseteq \mathcal{Q}^{\alpha(c,k)}(G)$ such that $V(Q_1) \cap V(Q_2) \neq \emptyset$. Then, $((G,B,W),k)$ is a yes-instance of* BW-Perfect Code *if and only if $((G_v,B_v,W_v),k-1)$ is a yes-instance for some $v \in V(Q_1) \cap V(Q_2) \cap B$.*

**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 $k$. Then, by Corollary 20, there exists $v \in V(Q_1) \cap V(Q_2)$ such that $V(Q_1) \cap D = V(Q_2) \cap D = \{v\}$. Then, $|D \setminus \{v\}| \leq k-1$, and by Lemma 25, $D \setminus \{v\}$ is a bw-perfect code of $(G_v,B_v,W_v)$, which proves that $((G_v,B_v,W_v),k-1)$ is a yes-instance.

Conversely, assume that $((G_v,B_v,W_v),k-1)$ is a yes-instance of BW-Perfect Code for some $v \in V(Q_1) \cap V(Q_2)$, and let $D' \subseteq B_v$ be a bw-perfect code of $(G_v,B_v,W_v)$ of size at most $k-1$. Then again, by Lemma 25, $D' \cup \{v\}$ is a bw-perfect code of $(G,B,W)$ of size at most $k$, which proves that $((G,B,W),k)$ is a yes-instance. ◀

**Definitions of good and bad instances.** We say that an instance $((G,B,W),k)$ is *bad* if any of the following three conditions hold.

**(i)** There exist three distinct cliques $Q_1,Q_2,Q_3 \in \mathcal{Q}^{\alpha(c,k)}(G)$ such that $V(Q_1) \cap V(Q_2) \neq \emptyset$, $V(Q_2) \cap V(Q_3) \neq \emptyset$, but $V(Q_1) \cap V(Q_3) = \emptyset$.

**(ii)** There exist distinct cliques $Q_1,Q_2 \in \mathcal{Q}^{\alpha(c,k)}(G)$ such that $V(Q_1) \cap V(Q_2) \neq \emptyset$, but $V(Q_1) \cap V(Q_2) \subseteq W$.

**(iii)** There exists $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$ such that $(V(Q) \setminus Z(Q)) \subseteq W$.

If none of these three conditions occur, then we say that $((G,B,W),k)$ is a *good* instance. We will show that a bad instance is necessarily a no-instance of BW-Perfect Code. It follows from Lemma 27 that $((G,B,W),k)$ is a no-instance if conditions (i) or (ii) hold; similarly, Corollary 22 implies that $((G,B,W),k)$ is a no-instance if condition (iii) holds.

▶ **Lemma 28.** *Let $((G,B,W),k)$ be an instance of* BW-Perfect Code*. If $((G,B,W),k)$ is a bad instance, then it is a no-instance of* BW-Perfect Code*.*

**Proof.** Let $((G,B,W),k)$ be a bad instance. Assume for a contradiction that $((G,B,W),k)$ is a yes-instance, and let $D$ be a bw-perfect code of $(G,B,W)$ of size at most $k$. Since $((G,B,W),k)$ is a bad instance, at least one of the three conditions in the definition of a bad instance must hold. We show that each of the three conditions will lead to a contradiction. Suppose that there exist three distinct cliques $Q_1, Q_2, Q_3 \in \mathcal{Q}^{\alpha(c,k)}(G)$ such that $V(Q_1) \cap V(Q_2) \neq \emptyset$, $V(Q_2) \cap V(Q_3) \neq \emptyset$, but $V(Q_1) \cap V(Q_3) = \emptyset$. By Corollary 20, there exist $v_{12} \in V(Q_1) \cap V(Q_2) \cap D$ and $v_{23} \in V(Q_1) \cap V(Q_2) \cap D$. Note that $v_{12} \neq v_{23}$, as $v_{12} \in V(Q_1)$ and $v_{23} \in V(Q_3)$, and $V(Q_1) \cap V(Q_3) = \emptyset$. But then $v_{12}, v_{23} \in V(Q_2) \cap D$, which by Lemma 19 is not possible. Now, suppose that there exist distinct cliques $Q_1, Q_2 \in \mathcal{Q}^{\alpha(c,k)}(G)$ such that $V(Q_1) \cap V(Q_2) \neq \emptyset$, but $V(Q_1) \cap V(Q_2) \subseteq W$. By assumption, $D \subseteq B$. And by Corollary 20, there exists $v \in V(Q_1) \cap V(Q_2) \cap D$, which implies that $V(Q_1) \cap V(Q_2) \cap B \neq \emptyset$. This contradicts the assumption that $V(Q_1) \cap V(Q_2) \subseteq W$. Finally, suppose that there exists $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$ such that $V(Q) \setminus Z(Q) \subseteq W$. By Lemma 19, there exists $w \in V(Q) \cap D$. By Corollary 22, $D \cap Z(Q) = \emptyset$, which implies that $w \in V(Q) \setminus Z(Q)$. But since $D \subseteq B$, we get that $(V(Q) \setminus Z(Q)) \cap B \neq \emptyset$, which contradicts the assumption that $V(Q) \setminus Z(Q) \subseteq W$. ◀

**Definition of a feasible set.** Consider an instance $((G, B, W), k)$ of BW-PERFECT CODE such that $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint. Let $k_{\mathcal{Q}} = |\mathcal{Q}^{\alpha(c,k)}(G)|$. We say that a set $S \subseteq M^{\alpha(c,k)}(G) \cap B$ is *feasible* if

**(a)** $|S| \leq k - k_{\mathcal{Q}}$,
**(b)** $N_G[S] \subseteq M^{\alpha(c,k)}(G)$,
**(c)** $S$ is a bw-perfect code for the bw-graph $(N_G[S], B \cap N_G[S], W \cap N_G[S])$,
**(d)** $(V(Q) \cap B) \setminus (Z(Q) \cup Y(Q, S)) \neq \emptyset$ for every $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$, and
**(e)** $(N(v) \cap L^{\alpha(c,k)}(G) \cap B) \setminus (Z(G) \cup Y(G, S)) \neq \emptyset$ for every $v \in M^{\alpha(c,k)}(G) \setminus N_G[S]$.

Informally, a set $S \subseteq M^{\alpha(c,k)}(G)$ is feasible if we can potentially extend $S$ to a bw-perfect code of $G$ by adding $k_Q$ vertices from $L^{\alpha(c,k)}(G)$. Since by Corollary 22 and Lemma 24, $Z(Q)$ and $Y(Q, S)$ cannot intersect a bw-perfect code that contains $S$, condition (d) says that in every large clique $Q$ contains a vertex that can potentially belong to a bw-perfect code (that contains $S$). Similarly, condition (e) says that for every vertex $v \in M^{\alpha(c,k)}(G)$ that is not dominated by $S$, there exists a vertex that can potentially belong to a bw-perfect code (that contains $S$) and dominate $v$. The next two lemmas prove properties of a feasible set. Recall that we say the family $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint if the elements of $\mathcal{Q}^{\alpha(c,k)}(G)$ are pairwise vertex-disjoint.

▶ **Lemma 29.** *Let $((G, B, W), k)$ be an instance of* BW-PERFECT CODE *such that $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, and let $D$ be a bw-perfect code of $(G, B, W)$ of size at most $k$. Let $S = D \cap M^{\alpha(c,k)}(G)$. Then (i) $|D \setminus S| = k_{\mathcal{Q}}$ and (ii) $|S| \leq k - k_{\mathcal{Q}}$, where $k_{\mathcal{Q}} = |\mathcal{Q}^{\alpha(c,k)}(G)|$.*

**Proof.** First, since $D$ is a bw-perfect code of $G$ of size at most $k$, by Lemma 19, we have $|D \cap V(Q)| = 1$ for every $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$. Second, since $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, the cliques in $\mathcal{Q}^{\alpha(c,k)}(G)$ are pairwise vertex-disjoint. Thus $\{V(Q) \mid Q \in \mathcal{Q}^{\alpha(c,k)}(G)\}$ is a partition of $L^{\alpha(c,k)}(G)$. Therefore, $|D \cap L^{\alpha(c,k)}(G)| = \sum_{Q \in \mathcal{Q}^{\alpha(c,k)}(G)} |D \cap V(Q)| = |\mathcal{Q}^{\alpha(c,k)}(G)| = k_{\mathcal{Q}}$. Now, since $S = D \cap M^{\alpha(c,k)}(G)$ and $\{L^{\alpha(c,k)}(G), M^{\alpha(c,k)}(G)\}$ is a partition of $V(G)$, we have $D \setminus S = D \cap L^{\alpha(c,k)}(G)$. Thus, $|D \setminus S| = |D \cap L^{\alpha(c,k)}(G)| = k_{\mathcal{Q}}$.

For proving assertion (ii) of the lemma, note that since $\{L^{\alpha(c,k)}(G), M^{\alpha(c,k)}(G)\}$ is a partition of $V(G)$, we have $|D| = |D \cap L^{\alpha(c,k)}(G))| + |D \cap M^{\alpha(c,k)}(G)| = |D \setminus S| + |S|$. Since $|D| \leq k$ and since $|D \setminus S| = k_{\mathcal{Q}}$, we can conclude that $|S| \leq k - k_{\mathcal{Q}}$. ◀

▶ **Lemma 30.** *Let $((G, B, W), k)$ be an instance of* BW-PERFECT CODE *such that $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, and let $D$ be a bw-perfect code of $(G, B, W)$ of size at most $k$. Then $S = D \cap M^{\alpha(c,k)}(G)$ is a feasible set.*

**Proof.** Let $D$ and $S$ be as defined in the lemma. To show that $S$ is a feasible set, we show that $S$ satisfies each of the five conditions in the definition of a feasible set.

First, by Lemma 29, $|S| \leq k - k_{\mathcal{Q}}$, where $k_{\mathcal{Q}} = |\mathcal{Q}^{\alpha(c,k)}(G)|$. Thus condition (a) holds. Next, by Lemma 23, we get $N_G[S] \subseteq M^{\alpha(c,k)}(G)$, and thus condition (b) holds.

Since $S \subseteq D$ and $D$ is a bw-perfect code of $(G, B, W)$, we get that $S \subseteq B$, and for each $v \in N_G[S]$, $|N_G[v] \cap S| = 1$. Hence, $S$ is a bw-perfect code for the bw-graph $(N_G[S], B \cap N_G[S], W \cap N_G[S])$, and thus condition (c) holds.

Next, to prove that condition (d) holds, consider $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$. Then, by Lemma 19, we have $|D \cap V(Q)| = 1$. Let $u$ be the unique vertex of $D \cap V(Q)$. Since $D \subseteq B$, we have $u \in V(Q) \cap B$. By Corollary 22, $D$ does not intersect $Z(G)$, and therefore, in particular, $D$ does not intersect $Z(Q$. Thus $u \notin Z(Q)$. Similarly, by Lemma 24, $D$ does not intersect $Y(Q, S)$. Thus $u \notin Y(Q, S)$. We thus have $u \in (V(Q) \cap B) \setminus (Z(Q) \cup Y(Q, S))$, which shows that condition (d) holds.

Finally, to prove that condition (e) holds, consider $v \in M^{\alpha(c,k)}(G) \setminus N_G[S]$. Since $D$ is a bw-perfect code of $(G, B, W)$, there exists a vertex $w \in D$ that dominates $v$. Thus $w \in N_G[v]$. Notice that $w \notin S$, because $v \notin N_G[S]$. As $S = D \cap M^{\alpha(c,k)}(G)$, we can conclude that $w \in L^{\alpha(c,k)}(G)$, which also implies that $w \neq v$, and thus $w \in N(v)$. We thus have $w \in N(v) \cap L^{\alpha(c,k)}(G)$. Now, by Corollary 22, $D$ does not intersect $Z(G)$ and thus $w \notin Z(G)$. Similarly, by Lemma 24, $D$ does not intersect $Y(G, S)$, and thus $w \notin Y(G, S)$. We thus have $w \in (N(v) \cap L^{\alpha(c,k)}(G)) \setminus (Z(G) \cup Y(G, S))$, which shows that condition (e) holds. ◀

With respect to each feasible set $S$, we now construct an instance of the Exact Hitting Set problem. We will have the guarantee that $(G, B, W)$ has a bw-perfect code $D$ of size at most $k$ with $D \cap M^{\alpha(c,k)}(G) = S$ if and only if $D \setminus S$ is a solution for the Exact Hitting Set instance corresponding to $S$.

▶ **Construction 31** (Construction of an Exact Hitting Set instance). *In the* Exact Hitting Set *problem, given a universe $U$, a family $\mathcal{A}$ of subsets of $U$, and a non-negative integer $\ell$, we ask if there exists a set $X \subseteq U$ of size at most $\ell$ such that $|A \cap X| = 1$ for every $A \in \mathcal{A}$; we call such a set $X$ a solution for the* Exact Hitting Set *instance $(U, \mathcal{A}, \ell)$. With respect to each feasible set $S \subseteq M^{\alpha(c,k)}(G) \cap B$, we construct an instance $(U_S, \mathcal{F}_S, k_{\mathcal{Q}})$ of the* Exact Hitting Set *problem as follows. We take $U_S = (L^{\alpha(c,k)}(G) \cap B) \setminus (Z(G) \cup Y(G, S))$, $\mathcal{F}_S = \mathcal{F}_S^1 \cup \mathcal{F}_S^2$, where $\mathcal{F}_S^1 = \left\{(V(Q) \cap B) \setminus (Z(Q) \cup Y(Q, S)) \mid Q \in \mathcal{Q}^{\alpha(c,k)}(G)\right\}$ and $\mathcal{F}_S^2 = \{(N(v) \cap (L^{\alpha(c,k)}(G) \cap B) \setminus (Z(G) \cup Y(G, S)) \mid v \in M^{\alpha(c,k)}(G) \setminus N_G[S]\}$, and $k_{\mathcal{Q}} = |\mathcal{Q}^{\alpha(c,k)}(G)|$.*

The next lemma says that to solve the instance $((G, B, W), k)$ of BW-Perfect Code, it is enough to solve the instance $(U_S, \mathcal{F}_S, k_{\mathcal{Q}})$ of Exact Hitting Set corresponding to each feasible set $S$.

▶ **Lemma 32.** *Let $((G, B, W), k)$ be an instance of* BW-Perfect Code *such that $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, and let $S \subseteq M^{\alpha(c,k)}(G)$ be a feasible set. Then, for $D \subseteq V(G)$ with $|D| \leq k$, $D$ is a bw-perfect code of $(G, B, W)$ with $D \cap M^{\alpha(c,k)}(G) = S$ if and only if $D \setminus S \subseteq U_S$ and $D \setminus S$ is a solution for the* Exact Hitting Set *instance $(U_S, \mathcal{F}_S, k_{\mathcal{Q}})$.*

**Proof.** Let $D \subseteq V(G)$ be such that $|D| \leq k$ and $D \cap M^{\alpha(c,k)}(G) = S$. First, recall that since $\left\{L^{\alpha(c,k)}(G), M^{\alpha(c,k)}(G)\right\}$ is a partition of $V(G)$, we have $D = (D \cap L^{\alpha(c,k)}(G)) \cup (D \cap M^{\alpha(c,k)}(G))$. And since $D \cap M^{\alpha(c,k)}(G) = S$, we have $D \cap L^{\alpha(c,k)}(G) = D \setminus S$.

Assume now that $D$ is a bw-perfect code of $(G, B, W)$. Observe that the following properties hold: (i) $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, (ii) $D$ is a bw-perfect code of $(G, B, W)$ of size at most $k$, and (iii) $D \cap M^{\alpha(c,k)}(G) = S$. Therefore, using Lemma 29, we can conclude that $|D \setminus S| = k_{\mathcal{Q}}$.

Now, we show that $D \setminus S \subseteq U_S$. Since $D$ is a bw-perfect code, $D \setminus S \subseteq B$. And by Corollary 22 $(D \setminus S) \cap Z(G) = \emptyset$, and by Lemma 24, $(D \setminus S) \cap Y(G, S) = \emptyset$. Therefore, $D \setminus S \subseteq (L^{\alpha(c,k)}(G) \cap B) \setminus (Z(G) \cup Y(G, S)) = U_S$.

Finally, to see that $D \setminus S$ is a solution for the Exact Hitting Set instance $(\mathcal{U}_S, \mathcal{F}_S, k_{\mathcal{Q}})$, consider $F \in \mathcal{F}_S$. We will show that $|F \cap (D \setminus S)| = 1$.

Suppose that $F \in \mathcal{F}_S^1$. Then, $F = (V(Q) \cap B) \setminus (Z(Q) \cup Y(Q, S))$ for some maximal clique $Q \in \mathcal{Q}^{\alpha(c,k)}((G))$. Lemma 19 implies that $|V(Q) \cap D| = 1$. Let $\{x\} = V(Q) \cap D$. Since $D \subseteq B$, we get that $x \in B$. By Corollary 22, $D$ does not intersect $Z(Q)$, and by Lemma 24, $D$ does not intersect $Y(Q, S)$, and therefore, $x \notin Z(Q) \cup Y(Q, S)$. We thus have $x \in (V(Q) \cap B) \setminus (Z(Q) \cup Y(Q, S)) = F$. Since $x$ is the only element of $V(Q)$ that belongs to $D$, and since $F \subseteq V(Q)$, we can conclude that $F \cap D = \{x\}$. Since $F \subseteq V(Q) \subseteq L^{\alpha(c,k)}(G)$, $F$ does not intersect $S$, and therefore, we can conclude that $F \cap (D \setminus S) = \{x\}$.

Suppose now that $F \in \mathcal{F}_S^2$. Then $F = (N(v') \cap L^{\alpha(c,k)}(G) \cap B) \setminus (Z(G) \cup Y(G, S))$ for some $v' \in M^{\alpha(c,k)}(G) \setminus N_G[S]$. Again, since $D$ is a bw-perfect code of $(G, B, W)$, $|N_G[v'] \cap D| = 1$. Let $\{x'\} = N_G[v'] \cap D$. Since $D \subseteq B$, we have $x' \in B$. But since $v' \notin N_G[S]$, and $x'v' \in E(G)$, we have $x' \notin S$. Then, $x' \in D \setminus S$. Also, note that $x' \neq v'$, as $x' \in D \setminus S \subseteq L^{\alpha(c,k)}(G)$, and $v' \in M^{\alpha(c,k)}(G)$. We can thus conclude that $\{x'\} = (N_G(v') \cap L^{\alpha(c,k)}(G) \cap B) \cap (D \setminus S)$. Since $D \setminus S \subseteq U_S$, we get that $x' \notin Z(G) \cup Y(G, S)$. Thus, $\{x'\} \subseteq F \cap (D \setminus S) \subseteq N_G[v'] \cap D = \{x'\}$, which proves that $|F \cap (D \setminus S)| = |\{x'\}| = 1$.

Thus, $D \setminus S$ is a solution for the Exact Hitting Set instance $(U_S, \mathcal{F}_S, k_Q)$. We have thus proved that if $D$ is a bw-perfect code of $(G, B, W)$, then $D \setminus S \subseteq U_S$, and $D \setminus S$ is a solution for $(U_S, \mathcal{F}_S, k_{\mathcal{Q}})$.

Conversely, assume that $D \setminus S \subseteq U_S$, and that $D \setminus S$ is a solution for the Exact Hitting Set instance $(U_S, \mathcal{F}_S, k_{\mathcal{Q}})$. To see that $D$ is a perfect code of $(G, B, W)$, consider a vertex $v \in V(G)$. We will show that $|N_G[v] \cap D| = 1$. Note first that $N_G[v] \cap D = (N_G[v] \cap S) \cup (N_G[v] \cap (D \setminus S))$.

Suppose that $v \in L^{\alpha(c,k)}(G)$. Since $S$ is feasible, $N_G[S] \subseteq M^{\alpha(c,k)}(G)$, and therefore $v \notin N_G[S]$. That is, $S$ does not dominate $v$. We now show that $|N_G[v] \cap (D \setminus S)| = 1$. Note that since $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, $v \in V(Q)$ for exactly one clique $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$. Since $S$ is feasible, $F' = (V(Q) \cap B) \setminus (Z(Q) \cup Y(Q, S)) \in \mathcal{F}_S^1$. And since $D \setminus S$ is a solution for the Exact Hitting Set instance $(U_S, \mathcal{F}_S, k_{\mathcal{Q}})$, we have $|F' \cap (D \setminus S)| = 1$. But note that $F' \subseteq V(Q) \subseteq N_G[v]$, and thus $|N_G[v] \cap (D \setminus S)| \geq |F' \cap (D \setminus S)| = 1$. Now, to show that $|N_G[v] \cap (D \setminus S)| = 1$, we will show that $N_G[v] \setminus F'$ does not intersect $D \setminus S$. Let $u \in N_G[v]$. We claim that if $u \notin F'$, then $u \notin D \setminus S$, which will imply that $|N_G[v] \cap (D \setminus S)| = 1$. So assume that $u \notin F'$. There are three possible cases: (a) $u \in V(Q)$, (b) $u \in L^{\alpha(c,k)}(G) \setminus V(Q)$ and (c) $u \in M^{\alpha(c,k)}(G)$. In each case, we will show that $u \notin D \setminus S$. First, if $u \in V(Q)$, then we must have $u \in W$ or $u \in Z(Q)$ or $u \in Y(Q, S)$, for otherwise we would have $u \in F'$. But $U_S$ does not intersect $W$, $Z(G) \supseteq Z(Q)$ or $Y(G, S) \supseteq Y(Q, S)$. Thus $u \notin U_S$, and therefore, $u \notin D \setminus S$, as $D \setminus S \subseteq U_S$. If $u \in L^{\alpha(c,k)}(G) \setminus V(Q)$, then $u \in Z(G)$, as $v \in V(Q)$ and $uv \in E(G)$. In this case also, $u \notin U_S$, and therefore $u \notin D \setminus S$. Now, if $u \in M^{\alpha(c,k)}(G)$, then clearly, $u \notin D \setminus S$, as $D \setminus S \subseteq L^{\alpha(c,k)}(G)$. These arguments prove that $|N_G[v] \cap D| = 1$.

Now, suppose that $v \in M^{\alpha(c,k)}(G)$. First, we consider the case when $v \in N_G[S]$. Then, since $S$ is a bw-perfect code for $(N_G[S], B \cap N_G[S], W \cap N_G[S])$, we have $|(N_G[v] \cap N_G[S]) \cap S| = 1$, i.e., $|N_G[v] \cap S| = 1$. Observe that to prove that $|N_G[v] \cap D| = 1$, it is now sufficient to prove that $u \notin D$ for every $u \in N_G[v] \setminus N_G[S]$. Consider such a vertex $u \in N_G[v] \setminus N_G[S]$. Then, $u \neq v$, as $v \in N_G[S]$. Therefore $u \in N_G(v) \setminus N_G[S]$. Note first that if $u \in M^{\alpha(c,k)}(G)$, then $u \notin D$, as $D \cap M^{\alpha(c,k)}(G) = S$, and $u \notin S$. On the other hand, if $u \in L^{\alpha(c,k)}(G)$, then $u \in Y(G, S)$ as $uv \in E(G)$, $v \in N[S]$, and $N[S] \subseteq M^{\alpha(c,k)}(G)$ (as $S$ is feasible). Therefore,

$u \notin U_S$, which implies that $u \notin D \setminus S$. These observations prove that $u \notin D$.

Now, consider the case when $v \in M^{\alpha(c,k)}(G) \setminus N_G[S]$. Then, $N_G[v] \cap S = \emptyset$. We will now show that $|N_G[v] \cap (D \setminus S)| = 1$. Note that $v \notin D$, as $v \notin S$, and $v \notin L^{\alpha(c,k)}(G)$ ($\supseteq (D \setminus S)$). Since $S$ is feasible, $(N(v) \cap L^{\alpha(c,k)}(G) \cap B) \setminus (Z(G) \cup Y(G,S)) \neq \emptyset$; and by Construction 31, there exists $F'' \in \mathcal{F}_S^2$ such that $F'' = (N(v) \cap L^{\alpha(c,k)}(G) \cap B) \setminus (Z(G) \cup Y(G,S))$. And since $D \setminus S$ is a solution for the Exact Hitting Set instance $(U_S, \mathcal{F}_S, k_Q)$, we have $|F'' \cap (D \setminus S)| = 1$, which implies that $|N_G[v] \cap (D \setminus S)| \geq |F'' \cap (D \setminus S| = 1$. Now, to complete the proof, it is sufficient to prove that $w \notin D \setminus S$ for every $w \in N_G[v] \setminus F''$. Consider $w \in N_G[v] \setminus F''$. Suppose $w \in L^{\alpha(c,k)}(G)$. Then $w \in W$ or $w \in Z(G)$ or $w \in Y(G,S)$ for otherwise, we would have $w \in F''$ as $w \in N(v)$. Hence $w \notin U_S$, and hence $w \notin D \setminus S$. Suppose now that $w \in M^{\alpha(c,k)}(G)$. Then clearly $w \notin D \setminus S \subseteq L^{\alpha(c,k)}(G)$. These arguments prove that $|N_G[v] \cap D| = 1$. ◀

In the next lemma we prove some size bounds based on the definition of a feasible set and by using the construction of the Exact Hitting Set instance.

▶ **Lemma 33.** *Let $((G,B,W),k)$ be an instance of* BW-Perfect Code *such that $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, and $G$ has no independent set of size $k+1$. Then the following statements are true.*

**(i)** $|M^{\alpha(c,k)}(G)| \leq R_c(\alpha(c,k), k+1) - 1 \leq 2(c-1)k^2$.

**(ii)** *$G$ contains at most $(2(c-1)k^2)^k$ feasible sets.*

**(iii)** *For every $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$, $|V(Q) \setminus Z(Q)| \leq 2(c-1)^2k^2 + 2$.*

**(iv)** *If $((G,B,W),k)$ is a yes-instance, then $|N(v) \cap L^{\alpha(c,k)}(G)| \leq (c-1)k$ for every $v \in M^{\alpha(c,k)}(G)$.*

**(v)** *If $((G,B,W),k)$ is a yes-instance, then for any feasible set $S$, $|F| \leq 2(c-1)^2k^2 + 2$, for every $F \in \mathcal{F}_S$.*

**Proof.** **(i)** By the definition of $M^{\alpha(c,k)}(G)$, the subgraph $G[M^{\alpha(c,k)}(G)]$ contains no clique of size $\alpha(c,k)$. By assumption, $G$ contains no independent set of size $k+1$; in particular, $G[M^{\alpha(c,k)}(G)]$ contains no independent set of size $k+1$. Thus, by Lemma 1, $|M^{\alpha(c,k)}(G)| \leq R_c(\alpha(c,k), k+1) - 1 = (c-1)\binom{k}{2} + (\alpha(c,k)-1)k \leq (c-1)k^2 + ((c-1)k - 1 + 1)k = 2(c-1)k^2$.

**(ii)** By definition, a feasible set has size at most $k$, and is contained in $M^{\alpha(c,k)}(G)$. Therefore, by assertion (i), we get that the number of feasible sets is at most $\binom{|M^{\alpha(c,k)}(G)|}{k} \leq (2(c-1)k^2)^k$.

**(iii)** Consider $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$. Note that every vertex in $V(Q)$ has a neighbour in $L^{\alpha(c,k)}(G) \setminus V(Q)$ or has a neighbour in $M^{\alpha(c,k)}(G)$ or has no neighbour in $V(G) \setminus V(Q)$. Let $A_1$ be the set of vertices in $Q$ that have a neighbour in $L^{\alpha(c,k)}(G) \setminus V(Q)$, $A_2$ be the set of vertices in $Q$ that have a neighbour in $M^{\alpha(c,k)}(G)$ and $A_3$ be the set of vertices in $Q$ that have no neighbour in $V(G) \setminus V(Q)$. That is, $V(Q) = A_1 \cup A_2 \cup A_3$. But notice that $A_1 = Z(Q)$. So to bound $|V(Q) \setminus Z(Q)|$, we only need to bound $|A_2|$ and $|A_3|$, as $V(Q) \setminus Z(Q) = V(Q) \setminus A_1 \subseteq A_2 \cup A_3$.

To bound $|A_2|$, notice that $A_2 = \bigcup_{v \in M^{\alpha(c,k)}(G)} N(v) \cap V(Q)$. By Lemma 6, we have $|N(v) \cap V(Q)| \leq c-1$ for every $v \in M^{\alpha(c,k)}(G)$. And by assertion (i), $|M^{\alpha(c,k)}(G)| \leq 2(c-1)k^2$. Thus $|A_2| \leq (c-1)(2(c-1)k^2)$.

To bound $|A_3|$, notice that $A_3 = V(Q) \setminus \bigcup_{v \in V(G) \setminus V(Q)} N(v)$; and by Lemma 18, we have $|V(Q) \setminus \bigcup_{v \in V(G) \setminus V(Q)} N(v)| \leq 2$.

We thus have $|V(Q) \setminus Z(Q)| \leq |A_2| + |A_3| \leq 2(c-1)^2k^2 + 2$.

**(iv)** Assume that $((G,B,W),k)$ is a yes-instance, and let $D$ be a bw-perfect code of $(G,B,W)$ of size at most $k$. Then, by Lemma 19, we have $|V(Q) \cap D| = 1$ for every $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$. And since $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, we have $|\mathcal{Q}^{\alpha(c,k)}(G)| \leq |D| \leq k$. Also, note that since $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, $L^{\alpha(c,k)}(G)$ is a disjoint union of the cliques in $\mathcal{Q}^{\alpha(c,k)}(G)$. Now, consider $v \in M^{\alpha(c,k)}(G)$. Then, by the definition of $M^{\alpha(c,k)}(G)$, $v \notin V(Q)$ for any $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$. Therefore, by Lemma 6, $|N(v) \cap V(Q)| \leq c-1$. Thus, $|N(v) \cap L^{\alpha(c,k)}(G)| = |\biguplus_{Q \in \mathcal{Q}^{\alpha(c,k)}(G)} N(v) \cap V(Q)| \leq (c-1)|\mathcal{Q}^{\alpha(c,k)}(G)| \leq (c-1)k$.

**(v)** Assume that $((G,B,W),k)$ is a yes-instance, and let $S \subseteq M^{\alpha(c,k)}(G)$ be a feasible set. Consider $F \in \mathcal{F}_S$. If $F \in \mathcal{F}^1_S$, then, $F \subseteq V(Q) \setminus Z(Q)$ for some $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$, and therefore, by assertion (iii), we have $|F| \leq 2(c-1)^2k^2 + 2$. If $F \in \mathcal{F}^2_S$, then, $F \subseteq N(v) \cap L^{\alpha(c,k)}(G)$ for some $v \in M^{\alpha(c,k)}(G)$, and therefore, by assertion (iv), we have $|F| \leq (c-1)k \leq 2(c-1)^2k^2 + 2$.

◀

For future reference, we now state the following observation, which follows immediately from the definitions of a good instance and a feasible set.

▶ **Observation 34.** *Let $((G,B,W),k)$ be an instance of* BW-Perfect Code.

**(i)** *Using the algorithm in Lemma 5, we can construct $\mathcal{Q}^{\alpha(c,k)}(G)$ in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. And once $\mathcal{Q}^{\alpha(c,k)}(G)$ is constructed, by brute force, we can check whether or not $((G,B,W),k)$ is a good instance, and whether or not $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, in time $\binom{|\mathcal{Q}^{\alpha(c,k)}(G)|}{3}n^{\mathcal{O}(1)} = 2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$.*

**(ii)** *For a set $S \subseteq M^{\alpha(c,k)}(G)$, we can check in polynomial time whether $S$ is feasible or not.*

**(iii)** *For a feasible set $S \subseteq M^{\alpha(c,k)}(G)$, we can construct the* Exact Hitting Set *instance $(U_S, \mathcal{F}_S, k_{\mathcal{Q}})$ in polynomial time.*

Finally, before we start describing the algorithm, we state the following result about Exact Hitting Set.

▶ **Lemma 35** (folklore). *There is an algorithm that, given an instance $(U, \mathcal{F}, \ell)$ of* Exact Hitting Set *as input, runs in time $d^\ell \cdot |U|^{\mathcal{O}(1)}$, where $d = \max_{F \in \mathcal{F}} |F|$, and correctly decides whether $(U, \mathcal{F}, \ell)$ is a yes-instance or a no-instance of* Exact Hitting Set.

We are now ready to describe our algorithm. We first informally discuss the idea behind the three main steps of the algorithm. The algorithm consists of two branching procedures followed by a brute-force procedure. We are given an instance $((G,B,W),k)$. In the first stage, we find an independent set $I$ of size $k+1$ (if it exists), and branch on the common black neighbours of $I$. Once there is no independent set of size $k+1$, in the second step, we enumerate all maximal cliques, and branch on the black vertices in the intersection of two large maximal cliques. And once this step is also fully executed, (i) $M^{\alpha(c,k)}(G)$ has no independent set of size $k+1$ and no clique of size $\alpha(c,k)$, and therefore will have size at most $R_c(\alpha(c,k), k+1) - 1$, and (ii) large cliques are pairwise vertex disjoint. In the third step, we guess which subset of $M^{\alpha(c,k)}(G)$ will go into the solution, and also guess one vertex each from the large maximal cliques that will go into the solution; and check if the guessed vertices make a bw-perfect code of size at most $k$. The third step can be executed by creating an Exact Hitting Set instance corresponding to each subset of $M^{\alpha(c,k)}(G)$.

**Description of our algorithm: Algorithm 1.** We are given an instance $((G,B,W),k)$ of BW-Perfect Code as input.

**Step 1.** First, if $k \geq 0$ and $V(G) = \emptyset$, then we return that $((G, B, W), k)$ is a yes-instance, and terminate. Otherwise, if $k > 0$, then we do as follows. We use the algorithm in Corollary 4 to check if $G$ has an independent set of size $k+1$. If the algorithm in Corollary 4 returns that $G$ has no such independent set, then we proceed to Step 1.1. On the other hand if algorithm in Corollary 4 returns a $(k+1)$-sized independent set $I$, then we branch into $|N^{[2]}(I) \cap B|$ many smaller instances of BW-Perfect Code. *For each $v \in N^{[2]}(I) \cap B$, we create the instance $((G_v, B_v, W_v), k-1)$ and recursively call Step 1 on this instance.* On any branch, at any point if the algorithm in Corollary 4 returns a $(k+1)$-sized independent set $I$ with $N^{[2]}(I) \cap B = \emptyset$, then we discard that branch. On all other branches, we recurse only until $k = 0$ or $V(G) = \emptyset$ or Corollary 4 does not return a $(k+1)$-sized independent set, whichever happens first.

**Step 1.1.** If $k \geq 0$ and $V(G) = \emptyset$, then we return that $((G, B, W), k)$ is a yes-instance, and terminate. Otherwise, if $k > 0$, we proceed as follows. We use the algorithm in Lemma 5 to construct $\mathcal{Q}^{\alpha(c,k)}(G)$. Then, using the algorithm in Observation 34-(i), we check if the instance $((G, B, W), k)$ is good and if $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint. If the instance is good and $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, then we proceed to Step 1.1.1. If the instance is good and $\mathcal{Q}^{\alpha(c,k)}(G)$ is not disjoint, then we choose two cliques $Q_1, Q_2 \in \mathcal{Q}^{\alpha(c,k)}(G)$ such that $V(Q_1) \cap V(Q_2) \neq \emptyset$, and branch into $|V(Q_1) \cap V(Q_2) \cap B|$ many smaller instances of BW-Perfect Code as follows. *For each $v \in V(Q_1) \cap V(Q_2) \cap B$, we create the instance $((G_v, B_v, W_v), k-1)$, and recursively call Step 1.1 on this instance.* On any branch, at any point, if we find that $\mathcal{Q}^{\alpha(c,k)}(G)$ is bad, then we discard that branch. On all other branches, we recurse only until $k = 0$ or $V(G) = \emptyset$ or $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint, whichever happens first.

**Step 1.1.1.** If $k \geq 0$ and $V(G) = \emptyset$, then we return that $((G, B, W), k)$ is a yes-instance, and terminate. Otherwise, if $k > 0$ and $k_\mathcal{Q} > k$, then we discard this branch. Otherwise, if $k > 0$ and if $k_\mathcal{Q} = |\mathcal{Q}^{\alpha(c,k)}(G)| \leq k$, then we do as follows. *For each set $S \subseteq M^{\alpha(c,k)}(G)$ such that $S$ is feasible, we construct the instance $(U_S, \mathcal{F}_S, k_\mathcal{Q})$ of* Exact Hitting Set*. If $|F| \leq 2(c-1)^2k^2 + 2$ for every $F \in \mathcal{F}_S$, then we solve the* Exact Hitting Set *instance $(U_S, \mathcal{F}_S, k_\mathcal{Q})$ using the algorithm in Lemma 35.* If $(U_S, \mathcal{F}_S, k_\mathcal{Q})$ is a yes-instance, then we return that $((G, B, W), k)$ is a yes-instance of BW-Perfect Code, and terminate.

**Step 2.** We return that $(G, B, W, k)$ is a no-instance, and terminate.

This completes the description of the algorithm. The correctness of Step 1 follows from Lemma 26. The correctness of Step 1.1 follows from Lemmas 27 and 28. Note that on any branch, when the algorithm enters Step 1.1.1, the instance $G$ contains no independent set of $k+1$, and $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint. The correctness of considering feasible sets in Step 1.1.1 follows from Lemma 30. The correctness of proceeding only if $|F| \leq 2(c-1)^2k^2 + 2$ for every $F \in \mathcal{F}_S$ follows from Lemma 33-(v). The correctness of returning yes if $(U_S, \mathcal{F}_S, k_\mathcal{Q})$ is a yes-instance of Exact Hitting Set follows from Lemma 32. Note that the algorithm enters Step 2 only if we have not already returned that the input instance is a yes-instance. And Lemmas 26 27, 28, 30, 33-(v) and 32 together imply that if $((G, B, W), k)$ is indeed a yes-instance, then we correctly return yes (in Steps 1, 1.1 or 1.1.1). Hence Step 2 is also correct. These observations show that Algorithm 1 is correct. We now analyse its runtime in the following lemma.

▶ **Lemma 36.** *Algorithm 1 runs in time $2^{\mathcal{O}(c+k\log(ck))}n^{\mathcal{O}(1)}$.*

**Proof.** Let us start with analysing the time taken for one execution of Step 1.1.1. For any set $S \subseteq M^{\alpha(c,k)}(G)$, by Observation 34-(ii), checking whether $S$ is feasible or not can be done in polynomial time. Also, by Observation 34-(iii), we can construct the Exact Hitting Set instance $(U_S, \mathcal{F}_S, k_\mathcal{Q})$ in polynomial time.

For each feasible set $S$, we have $|U_S| \leq |V(G)| = n$. Therefore, by Lemma 35, solving Exact Hitting Set on the instance $(U_S, \mathcal{F}_S, k_\mathcal{Q})$ takes time $(2(c-1)^2k^2+2)^{k_\mathcal{Q}} n^{\mathcal{O}(1)} \leq (2c^2k^2)^k n^{\mathcal{O}(1)}$. Finally, by Lemma 33-(ii), there are at most $(2(c-1)k^2)^k \leq (2ck^2)^k$ many feasible sets. Therefore, one execution of Step 1.1.1 takes time $(2ck^2)^k \cdot (2c^2k^2)^k n^{\mathcal{O}(1)} = (ck)^{\mathcal{O}(k)} n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log(ck))} n^{\mathcal{O}(1)}$.

Now, consider Step 1.1. By Lemma 5, we can construct in $\mathcal{Q}(G)$ and $\mathcal{Q}^{\alpha(c,k)}(G)$ in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$. By Observation 34-(i), we can check, again in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$, whether or not $((G,B,W),k)$ is good and $\mathcal{Q}^{\alpha(c,k)}(G)$ is disjoint. Also, note that in one execution of Step 1.1, at most $|V(Q_1) \cap V(Q_2) \cap B| \leq c-1$ many recursive calls are being made. So the total number of recursive calls made to Step 1.1 is at most $(c-1)^k$.

Finally, consider Step 1. By Corollary 4, finding a $(k+1)$-sized independent set $I$ takes time $2^{\mathcal{O}(k \log(ck))} n^{\mathcal{O}(1)}$. Now, in one execution of Step 1, at most $|N^{[2]}(I)|$ recursive calls to Step 1 are being made. And by Lemma 12, $|N^{[2]}(I)| \leq (c-1)\binom{k+1}{2}$. So the total number of recursive calls made to Step 1 is at most $((c-1)\binom{k+1}{2})^k$.

Therefore, the total runtime of the algorithm is bounded by

$$\begin{aligned}
&\left((c-1)\binom{k+1}{2}\right)^k 2^{\mathcal{O}(k\log(ck))} n^{\mathcal{O}(1)} \cdot (c-1)^k 2^{\mathcal{O}(c)} n^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(k\log(ck))} n^{\mathcal{O}(1)} \\
&\qquad = c^{\mathcal{O}(k)} k^{\mathcal{O}(k)} 2^{\mathcal{O}(k\log(ck))} n^{\mathcal{O}(1)} \cdot c^{\mathcal{O}(k)} 2^{\mathcal{O}(c)} n^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(k\log(ck))} n^{\mathcal{O}(1)} \\
&\qquad = 2^{\mathcal{O}(k\log(ck))} n^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(c+k\log c)} n^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(k\log(ck))} n^{\mathcal{O}(1)} \\
&\qquad = 2^{\mathcal{O}(c+k\log(ck))} n^{\mathcal{O}(1)}.
\end{aligned}$$

◀

We have thus proved the following theorem.

▶ **Theorem 37.** BW-Perfect Code *on $c$-closed graphs admits an algorithm running in time* $2^{\mathcal{O}(c+k\log(ck))} n^{\mathcal{O}(1)}$.

Since we can reduce an instance $(G,k)$ of Perfect Code into an equivalent instance $((G,B,W),k)$ in polynomial time, Theorem 37 implies the following result.

▶ **Theorem 38.** Perfect Code *on $c$-closed graphs admits an algorithm that runs in time* $2^{\mathcal{O}(c+k\log(ck))} n^{\mathcal{O}(1)}$.

## 3.2 A Polynomial Kernel for Perfect Code on c-Closed Graphs

We now move on to designing a kernel for 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. Then we argue that in polynomial time, we can reduce an instance of BW-Perfect Code to an equivalent instance of Perfect Code, which will give us the required kernel. Specifically, we prove the following theorem.

▶ **Theorem 39.** *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 39, 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, and the second one presents a polynomial time reduction from BW-Perfect Code to Perfect Code.

▶ **Lemma 40.** Perfect Code *is polynomial time solvable on* 1*-closed graphs.*

▶ **Lemma 41.** *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 39, Lemmas 40 and 41, we derive the following result.

▶ **Theorem 42.** *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 40 to solve the Perfect Code problem on $(G, k)$. 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 39, 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 41 to obtain from $((G', B', W'), k')$ an equivalent instance $(G'', k'')$ of Perfect Code. Note that as the algorithms in Lemma 40, Theorem 39 and Lemma 41, run in polynomial time, our kernelization algorithm returns $(G'', k'')$ in polynomial time. Since Lemma 41 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 39 and Lemmas 40 and 41. We prove the two lemmas first.

**Proof of Lemma 40.** Let $(G, k)$ be an instance of Perfect Code, where $G$ is a 1-closed graph. Observe first that every connected component of $G$ is a clique. To see this, consider a connected component $C$ of $G$, and let $x, y \in V(C)$. We claim that $xy \in E(G)$. Suppose not. Let $P = xv_1v_2 \ldots v_ry$ be a shortest $x$-$y$ path in $G$. Then, note that $|N(x) \cap N(v_2)| \geq 1$ as $v_1 \in N(x) \cap N(v_2)$. Since $G$ is 1-closed, we must have $xv_2 \in E(G)$, which contradicts the assumption that $P$ is a shortest path between $x$ and $y$.

Since each connected component of $G$ is a clique, any perfect code of $G$ must contain exactly one vertex from each of the connected components. So, if $G$ has more than $k$ connected components, then $(G, k)$ is a no-instance of Perfect Code, and otherwise, $(G, k)$ is a yes-instance of Perfect Code. Thus, to check if $G$ has a perfect code of size at most $k$, we only need to enumerate the connected components of $G$, which can be done in polynomial time. Hence the lemma follows. ◀

**Proof of Lemma 41.** 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

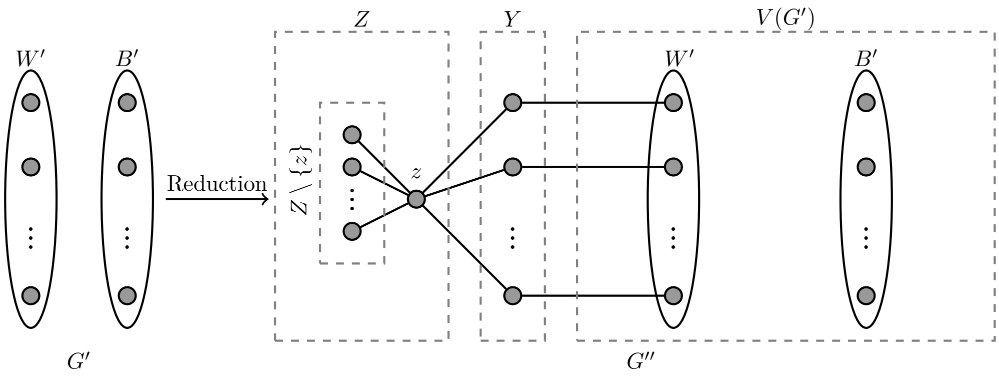


**Figure 1** Polynomial time reduction from BW-Perfect Code to Perfect Code

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 now define the graph $G''$. We take $G''$ to be the supergraph of $G$ obtained by adding $k' + 3 + r$ new vertices $z, z_1, z_2, \ldots, z_{k'+2}, y_1, y_2, \ldots, y_r$. We also add the following new edges to $G''$. We make $z$ adjacent to $z_i$ and $y_j$ for every $i \in [k'+2]$ and $j \in [r]$; also, for every $j \in [r]$, we make $y_j$ adjacent to $v_j$. Thus $V(G'') = V(G') \cup Y \cup Z$, where $Y = \{y_1, y_2, \ldots, y_r\}$ and $Z = \{z, z_1, z_2, \ldots, z_{k'+2}\}$; and $E(G'') = E(G') \cup E_1 \cup E_2 \cup E_3$, where $E_1 = \{v_i y_i \mid i \in [r]\}$, $E_2 = \{y_i z \mid i \in [r]\}$ and $E_3 = \{z z_i \mid i \in [k'+2]\}$. And we set $k'' = k' + 1$. Notice that $G'$ is subgraph of $G'$. Notice also that the set $Y$ is another copy of $W'$. Thus, $\{v_1, v_2, \ldots, v_r\}$ and $Y$ are two copies of $W'$, and the set $E_1$ is a matching in $G''$ between the two copies. See Figure 1.

First, $|V(G'')| = |V(G')| + |Y| + |Z| = |V(G')| + |W'| + (k'+3) = \mathcal{O}(|V(G')|)$.

Second, we 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. 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 $D'' = D' \cup \{z\}$. Notice that $z$ dominates $N_{G''}[z] = Z \cup Y$, and does not dominate vertex $v_i$ for any $i \in [n]$; and no vertex of $D$ dominates $Z \cup Y$ as ($D \subseteq B'$ and hence) no vertex of $D$ is adjacent to any vertex in $Z \cup Y$. Thus $D''$ is a perfect code of $G''$ of size $|D'| + 1 \leq k' + 1 = k''$. Conversely, assume that $(G'', k'')$ is a yes-instance of Perfect Code, and let $D \subseteq V(G'')$ be a perfect code of $G''$ of size at most $k''$. We first claim that $z \in D$. If not, then, since $N_{G''}[z_i] = \{z, z_i\}$, we must have $z_i \in D$ for every $i \in [k'+2]$, which contradicts the assumption that $|D| \leq k'' = k' + 1$. But then as $z$ dominates $Y \cup (Z \setminus \{z\}) = N_{G''}(z)$, we have $Y \cup (Z \setminus \{z\}) \cap (D \setminus \{z\}) = \emptyset$. Therefore $D \setminus \{z\} \subseteq V(G')$. Now, for every $j \in [r]$, since $\text{dist}_{G''}(z, v_j) = 2$, we can conclude that $v_r \notin D$; i.e., $D \cap W' = \emptyset$. Thus $D \setminus \{z\} \subseteq B'$. Since $z$ does not dominate any vertex in $V(G') \subseteq V(G'')$, we can conclude that $D \setminus \{z\}$ dominates every vertex in $V(G')$ exactly once. And we have $|D \setminus \{z\}| = |D| - 1 \leq k'' - 1 = k'$. We have thus shown that $D \subseteq B'$, $D$ dominates every vertex of $G'$ exactly once, and $D$ has size at most $k$; that is, $D$ is a bw-perfect code of $G'$ of size at most $k'$.

Finally, to conclude the proof, we only need to show that if $G'$ is a $c$-closed graph, then so is $G''$. As it is straightforward to verify this, we omit its proof. ◀

The rest of this section is dedicated to proving Theorem 39. Towards that end, we first define two functions $\gamma, \mu : \mathbb{N} \to \mathbb{N}$ as follows. Recall that $\beta(a, b) = 2[(a-1)(b-1)+1]$ and

$R_c(a,b) = (c-1)\binom{b-1}{2} + (a-1)(b-1) + 1$. For $a, b \in \mathbb{N}$, we define $\boldsymbol{\gamma(1,b) = b+1}$, and $\boldsymbol{\gamma(a,b) = b\mu(a-1,b)+1}$; and $\boldsymbol{\mu(a,b) = \gamma(a,b) + R_a(\beta(a,\gamma(a,b)+1), \gamma(a,b)+1) - 1}$. These functions $\gamma$ and $\mu$ will be used to bound the size of independent sets in $G$ when $((G,B,W),k)$ is a yes-instance.

▶ **Observation 43.** *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*

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

**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 44, and in the second part, we bound the size of cliques in $(G,B,W)$ using Reduction Rules 52 and 56 (and Reduction Rule 15). Once the size of cliques and independent sets are bounded, we apply Lemma 1 to derive the kernel. Recall that for a set $Y \subseteq V(G)$, $CN(Y)$ denotes the set of common neighbours of the vertices in $Y$, i.e., $CN(Y) = \bigcap_{v \in Y} N(v)$.

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 44. But the difficulty is 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 11, 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 (which we already did in Reduction Rule 15), 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 52 and 56).

We now formally introduce the following reduction rule.

▶ **Reduction Rule 44.** *For each $i \in [c-1]$, we introduce Reduction Rule 44.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 11 on the graph $G[CN(Y)]$ with $\ell = \gamma(i, k) + 1$. If the algorithm returns an independent set $I$ of size $\ell$, 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 procedure until the algorithm in Lemma 11 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 44.i in the increasing order of $i$. That is, we first apply Reduction Rule 44.1 exhaustively, and for each $i \in [c-1] \setminus \{1\}$, we apply Reduction Rule 44.i only if Reduction Rule 44.$(i-1)$ is no longer applicable.*

We now observe the following fact, which will be useful in establishing the correctness of Reduction Rule 44.

▶ **Observation 45.** *Fix $i \in [c-1]$. For any $Y \subseteq V(G)$ with $|Y| = c - i$, by Lemma 13, the subgraph $G[CN(Y)]$ is $i$-closed. Therefore, after an exhaustive application of Reduction Rule 44.i, by Lemma 11, 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 44.$(c-1)$, for every $v \in V(G)$, every independent set in $G[N(v)]$ has size at most $\mu(c-1, k)$.*

▶ **Lemma 46.** *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 $w_1, w_2 \in I \subseteq CN(Y)$, 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 $w_1 w_2 \in E(G)$, 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 47.** *Fix $i \in [c-1] \setminus \{1\}$. Let $((G, B, W), k)$ be an instance of* BW-Perfect Code *to which Reduction Rule 44.$(i-1)$ 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 45 because of our assumption that Reduction Rule 44.$(i-1)$ 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 48.** *For each $i \in [c-1]$, Reduction Rule 44.i is safe.*

**Proof.** Fix $i \in [c-1]$. Let $((G', B', W'), k)$ be the instance obtained from $((G, B, W), k)$ by a single application of Reduction Rule 44.$i$. 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.

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 $k$. We first claim that $|D \cap Y| = 1$. Suppose $i = 1$. Then $|Y| = c-1$ and $|I| = \gamma(1,k)+1$. By Lemma 46, $|D \cap Y| = 1$. Now, suppose that $i > 1$. First, by assumption, Reduction Rule 44.$j$ is not applicable to $((G, B, W), k)$ for any $j \in [i-1]$. And we have $|Y| = c-i$, and $|I| = \gamma(i,k)+1$. Then, by Lemma 47, we have $|D \cap Y| = 1$. In either case, 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 14, 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 46 if $i = 1$ and by Lemma 47 if $i > 1$, 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. ◀

▶ Remark 49. Observe that to apply the rule exhaustively, we need not go over all sets $Y \subseteq V(G)$ of size at most $c-1$. We only need to consider sets $Y \subseteq V(G)$ for which $CN(Y)$ contains at least two non-adjacent vertices, say $x$ and $y$. But then we would have $Y \subseteq CN(\{x,y\})$. Since $|CN(\{x,y\})| \leq c-1$, we only have at most $2^{c-1}$ choices for $Y$. And since there are only at most $\binom{n}{2} = \mathcal{O}(n^2)$ choices for $\{x,y\}$, we can conclude that we only need to go over $2^{c-1}n^2$ sets $Y$ to apply the rule exhaustively. Now, note that each application of Reduction Rule 44 can be executed in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$ as the algorithm in Lemma 11 takes time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. Also, for each set $Y \subseteq V(G)$ with $|Y| \leq c-1$, Reduction Rule 44 is applied only at most $|CN(Y)| \leq n$ times. Thus, we can exhaustively apply Reduction Rule 44 in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$.[3] Recall that $c$ is a fixed constant, and therefore we can exhaustively apply

[3] In the conference version of this paper [37], we had only claimed that we can exhaustively apply

Reduction Rule 44 in polynomial time. So, from now on, we assume that Reduction Rule 44 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 50.** *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 45, is not possible, as Reduction Rule 44, and in particular, Reduction Rule 44.$(c-1)$ 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 9), as well as the number of vertices that do not belong to any large maximal clique (by Lemma 1).

▶ **Lemma 51.** *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. $|M^{\beta(c,\gamma(c,k))}(G)| \leq R_c(\beta(c,\gamma(c,k)),\gamma(c,k))-1$.

**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 9, $G$ contains an independent set of size $\gamma(c,k)$, which contradicts Lemma 50.
2. By the definition of $M^{\beta(c,\gamma(c,k))}(G)$, the induced subgraph $G[M^{\beta(c,\gamma(c,k))}(G)]$ of $G$ contains no clique of size $\beta(c,\gamma(c,k))$. By Lemma 50, the graph $G$, and hence the graph $G[M^{\beta(c,\gamma(c,k))}(G)]$, contains no independent set of size $\gamma(c,k)$. The bound then follows from Lemma 1.

◀

In what follows, we show that the size of cliques in $G$ can be bounded as well, which, in turn, will help us bound $|L^{\beta(c,\gamma(c,k))}(G)|$. Recall that $\alpha(c,k)=(c-1)k+1$.

▶ **Reduction Rule 52.** *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 $B' = B \setminus N(V(Q))$.*

▶ **Lemma 53.** *Reduction Rule 52 is safe.*

**Proof.** Let $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$, and let $((G,B',W'),k)$ be the instance obtained from $((G,B,W),k)$ by a single application of Reduction Rule 52 by colouring $N(V(Q))$ white.

Assume that $((G,B,W),k)$ is a yes-instance, and let $D \subseteq B$ be a bw-perfect code of $((G,B,W),k)$ of size at most $k$. Then, by Lemma 19, $|D \cap V(Q)| = 1$. Let $\{v\} = D \cap V(Q)$.

---

Reduction Rule 44 in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(c)}$. This bound, in particular the term $n^{\mathcal{O}(c)}$, comes from going over all subsets $Y \subseteq V(G)$ of size at most $c-1$. While this is obviously true, we need not go over all subsets $Y$, as we have just explained. We are grateful to an anonymous reviewer who pointed out this fact to us, which led to an improvement in the runtime from $2^{\mathcal{O}(c)}n^{\mathcal{O}(c)}$ to $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$.

Note that as $Q$ is a clique, for any $u \in N(V(Q))$, we have $\text{dist}_G(u, v) \leq 2$, which together with Observation 14, implies that $D \cap N(V(Q)) = \emptyset$. Therefore, $D \subseteq B \setminus N(V(Q)) = B'$. Thus, $D$ is a bw-perfect code of $(G, B', W')$ as well.

For the other direction, note that as $B' \subseteq B$, any bw-perfect code of $(G, B', W')$ is a bw-perfect code of $(G, B, W)$ as well. ◀

▶ Remark 54. Observe that given an instance $((G, B, W), k)$ of BW-Perfect Code, using the algorithm in Lemma 5, we can construct $\mathcal{Q}^{\alpha(c,k)}(G)$ in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. Once $\mathcal{Q}^{\alpha(c,k)}(G)$ is constructed, we can apply Reduction Rule 52 exhaustively, in time $|\mathcal{Q}^{\alpha(c,k)}(G)|n^{\mathcal{O}(1)} \leq 2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$. So, from now on, we assume that Reduction Rule 52 has been applied exhaustively.

▶ **Lemma 55.** *Let $(G, B, W)$ be a bw-graph, and $Q$ a clique (not necessarily maximal) in $G$ such that $N(Q) \subseteq W$. Then, for any bw-perfect code $D$ of $G$, we have $|D \cap V(Q)| = 1$.*

**Proof.** Let $D \subseteq B$ be a bw-perfect code of $(G, B, W)$. Since $N(Q) \subseteq W$, we have $D \cap N(Q) = \emptyset$. And since $D$ dominates $V(Q)$, we must have $D \cap V(Q) \neq \emptyset$. But since $Q$ is a clique and $D$ an independent set, at most one vertex from $V(Q)$ can belong to $D$. We thus have $|D \cap V(Q)| = 1$. ◀

Recall that for each $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$, we defined $Z(Q)$ 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., $Z(Q) = \{u \in V(Q) \mid uv \in E(G) \text{ for some } v \in V(Q'), \text{ where } Q' \in \mathcal{Q}^{\alpha(c,k)}(G), u \notin V(Q'), \text{ and } Q' \neq Q\}$.

▶ **Reduction Rule 56.** *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 57.** *Reduction Rule 56 is safe.*

**Proof.** Let $((G', B', W'), k)$ be obtained from $((G, B, W), k)$ by a single application of Reduction Rule 56. Then there exists $Q \in \mathcal{Q}^{\alpha(c,k)+1}(G)$ and $v \in Z(Q)$ such that $G' = G - v$, $B' = B \setminus \{v\}$, and $W' = W \setminus \{v\}$. Note first that as $\mathcal{Q}^{\alpha(c,k)+1}(G) \subseteq \mathcal{Q}^{\alpha(c,k)}(G)$, we have $Q \in \mathcal{Q}^{\alpha(c,k)}(G)$. By Remark 54, $N_G(V(Q)) \subseteq W$. In fact, as $v \in V(Q)$, we have $N_G(V(Q)) \subseteq W \setminus \{v\} = W'$.

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 Corollary 22, we have $v \notin D$, and therefore $D$ is a bw-perfect code of $(G', B', W')$ as well.

Conversely, 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 $k$. We claim that $D'$ is a bw-perfect code of $(G, B, W)$ as well. First, $D' \subseteq B' \subseteq B$. Also, note that for every vertex $u \in V(G) \setminus \{v\}$, we have $N_{G'}[u] = N_G[u] \setminus v$. Since $v \notin D'$, we get that $D' \cap N_{G'}[u] = D' \cap N_G[u]$, and thus $|D' \cap N_G[u]| = 1$. To complete the proof, we now argue that $|D' \cap N_G[v]| = 1$. Since $Q - v$ is a clique (not necessarily maximal) in $G'$, and $N_{G'}(V(Q - v)) \subseteq N_G(V(Q)) \subseteq W'$, by Lemma 55, we have $|D' \cap V(Q - v)| = 1$. Let $\{w\} = D' \cap V(Q - v)$. Thus $\{w\} = D' \cap (N_G[v] \cap V(Q))$. And as $N_G(v) \setminus V(Q) \subseteq N_G(V(Q)) \subseteq W'$, we get that $D' \cap N_G[v] \setminus V(Q)) = \emptyset$. Thus, we conclude that $|D' \cap N_G[v]| = |\{w\}| = 1$. ◀

▶ Remark 58. Just like Reduction Rule 52, observe that Reduction Rule 56 can be applied in time $2^{\mathcal{O}(c)}n^{\mathcal{O}(1)}$ as well. So from now on, we assume that Reduction Rule 56 has been applied exhaustively.

▶ **Lemma 59.** *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. $|V(Q)| \leq (c-1)[R_c(\beta(c, \gamma(c,k)), \gamma(c,k)) - 1] + 2$.

**Proof.** Assume that $((G, B, W), k)$ is a yes-instance. Consider $Q \in \mathcal{Q}^{\beta(c,\gamma(c,k))}(G)$.

1. Observe that as $\gamma(c,k) \geq \gamma(1,k)$ for every $c \geq 1$, we have $\beta(c, \gamma(c,k)) \geq \beta(c, \gamma(1,k)) = 2[(c-1)(\gamma(1,k)-1)+1] = 2[(c-1)k+1] \geq (c-1)k+2 = \alpha(c,k)+1$. Therefore, $Q \in \mathcal{Q}^{\alpha(c,k)+1}(G)$. Thus, if $Z(Q) \neq \emptyset$, then Reduction Rule 56 would apply, which contradicts Remark 58.
2. We classify the vertices of $V(Q)$ depending on their neighbours in $V(G) \setminus V(Q)$. For every vertex $v \in V(Q)$, there are three possibilities: (i) $v$ has no neighbour in $V(G) \setminus V(Q)$; but by Lemma 18, the number of such vertices $v$ is at most 2. (ii) The vertex $v$ has a neighbour in $L^{\beta(c,\gamma(c,k))}(G) \setminus V(Q)$; but in this case $v \in Z(Q)$, which contradicts the previous assertion that $Z(Q) = \emptyset$. And (iii) $v$ has a neighbour in $M^{\beta(c,\gamma(c,k))}(G)$; the number of such vertices $v$ is at most $(c-1)|M^{\beta(c,\gamma(c,k))}(G)|$, because by Lemma 6, every vertex in $M^{\beta(c,\gamma(c,k))}(G)$ has at most $c-1$ neighbours in $V(Q)$. Now, by Lemma 51, $|M^{\beta(c,\gamma(c,k))}(G)| \leq R_c(\beta(c,\gamma(c,k)), \gamma(c,k)) - 1$ and we thus have $|V(Q)| \leq (c-1)[R_c(\beta(c,\gamma(c,k)), \gamma(c,k)) - 1] + 2$.

◀

Finally, Lemma 51-(1) and Lemma 59-(2) together bound $|L^{\beta(c,\gamma(c,k))}(G)|$, which, in turn, bounds $|V(G)|$.

▶ **Lemma 60.** *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 51-(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 59-(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 51-(2), we have $|M^{\beta(c,\gamma(c,k))}(G)| \leq R_c(\beta(c,\gamma(c,k)), \gamma(c,k)) - 1 = R_c(\mathcal{O}(k^{2^c-1}), \mathcal{O}(k^{2^c-1})) = \mathcal{O}(k^{2(2^c-1)})$. Finally, since $\left\{L^{\beta(c,\gamma(c,k))}(G), M^{\beta(c,\gamma(c,k))}(G)\right\}$ is a partition of $V(G)$, we conclude that $|V(G)| = \mathcal{O}(k^{3(2^c-1)})$. ◀

Each of our reduction rule is safe and by Remarks 49 and 58, all the reduction rules we introduced can be executed in polynomial time, and are applied only polynomially many times. We have thus proved Theorem 39.

## 4 Connected Dominating Set on c-Closed Graphs

Recall that for a graph $G$, a connected dominating set of $G$ is a dominating set $D \subseteq V(G)$ such that $G[D]$ is connected. The CDS problem, which we formally define below, asks if a given graph contains a connected dominating set of a certain size.

| Connected Dominating Set (CDS) | **Parameter:** $k + cl(G)$ |
|---|---|
| **Input:** An undirected graph $G$ and a non-negative integer $k$. | |
| **Question:** Does $G$ have a connected dominating set of size at most $k$? | |

In this section, we show that CDS admits an algorithm on $c$-closed graphs that runs in time $2^{\mathcal{O}(c+k\log(ck))}n^{\mathcal{O}(1)}$. We also argue that CDS has no polynomial kernel on $c$-closed graphs. We address the kernelization question first, for which invoke the following result due to Misra et al. [50].

▶ **Lemma 61** ([50]). *The* CDS *problem, parameterized by $k$, admits no polynomial kernel on the class of graphs with girth 5, unless* NP $\subseteq$ *co*-NP/*poly.*

Observe now that if $G$ is a graph with girth 5, then $G$ is 2-closed. If not, then $G$ contains 2 non-adjacent vertices, say $x, y \in V(G)$ such that $x$ and $y$ have two common neighbours, say, $x'$ and $y'$. But then, note that $G[\{x, y, x', y'\}]$ contains a 4-cycle, which contradicts the assumption that $G$ has girth 5. Lemma 61 thus implies the following result.

▶ **Theorem 62.** CDS *admits no polynomial kernel on 2-closed graphs, unless* NP $\subseteq$ *co*-NP/*poly.*

The rest of this section is dedicated to designing an algorithm for CDS that runs in time $2^{\mathcal{O}(c+k\log(ck))}n^{\mathcal{O}(1)}$. To design the algorithm, we consider a slightly more general version of the problem, which we call CPY-Connected Dominating Set (CPY-CDS, for short). A cpy-graph is a graph $G$ along with a partition of $V(G)$ into three parts, $C, P$ and $Y$ (we allow empty parts) such that for each vertex $v \in P$, $N_G(v) \cap C \neq \emptyset$ and there does not exist an edge $uv \in E(G)$ such that $u \in C$ and $v \in Y$. For convenience we write that $(G, C, P, Y)$ is a cpy-graph. We think of the vertices in these three parts $C, P, Y$ as having colours: $C$ for cyan, $P$ for purple and $Y$ for yellow. So a cpy-graph is a graph in which each purple vertex is dominated by a cyan vertex, and no yellow vertex is dominated by a cyan vertex.

A cpy-connected dominating set of $(G, C, P, Y)$ is a connected dominating set $D$ of $G$ such that $C \subseteq D$. The CPY-CDS problem is formally defined below.

| CPY-Connected Dominating Set (CPY-CDS) | **Parameter:** $k + cl(G)$ |
|---|---|
| **Input:** A cpy-graph $(G, C, P, Y)$ and a non-negative integer $k$. | |
| **Question:** Does $(G, C, P, Y)$ have a cpy-connected dominating set of size at most $k$? | |

It is not difficult to see that an instance $(G, k)$ of CDS can be reduced to an equivalent instance $((G, C, P, Y), k)$ of CPY-CDS by taking $Y = V(G)$ and $C = P = \emptyset$. Informally, our algorithm runs in two steps. In the first step, it reduces the size of a maximum independent set in $G[Y]$ to $k$, by a branching procedure implied by Lemma 12. After the branching procedure, let $(G, C, P, Y)$ be the reduced instance such that $G[Y]$ does not contain any independent set of size $k + 1$. Informally, in the next step, the algorithm constructs a solution ensuring connectivity as follows. (1) The set $C$ is contained in every solution, which dominates $P \cup C$. (2) The set $Y$ is divided into two parts. One part $Y_1$ is the set of vertices that are contained in some maximal clique of $G$ that contains at least $\beta(c, k+1)$ vertices of $Y$, (we call them "large" cliques), and the other part is $Y_2 = Y \setminus Y_1$. (2a) To dominate $Y_1$, we take a vertex from each large clique into the solution. (2b) Guess the set $S \subseteq Y_2$ which is contained in the solution. Now, to dominate $Y_2' = Y_2 \setminus N[S]$, we guess a partition $J_1, J_2, \ldots, J_\ell$ of $Y_2'$ such that all vertices in $X_i$ are dominated by the same vertex in the solution. For each $J_i$, we need to take a vertex from the set of common neighbours of $J_i$ into the solution, while ensuring that the solution is connected. To execute this step, the algorithm generates $f(c,k)$ many instances of the Steiner Tree problem, and invokes a known algorithm for Steiner Tree. In the Steiner Tree problem, given an $n$-vertex graph $G^*$, a weight function $w : E(G^*) \to [\rho]$ for some $\rho \in \mathbb{N}$ and a set of vertices $T \subset V(G^*)$ as input, the objective is to find a minimum weight subgraph $H$ of $G$ such that $H$ is a connected subgraph of $G^*$ and $T \subseteq V(H)$. Here the vertices in $T$ are called terminals. Our algorithm uses the following result due to Nederlof [54] to solve the Steiner Tree instances.

▶ **Lemma 63** ([54, Theorem 3]). *There is an algorithm that, given an instance $(G^*, w, T^*)$ of* Steiner Tree *as input, runs in time $\mathcal{O}(2^{|T^*|} \cdot \rho \cdot n^{\mathcal{O}(1)})$ and outputs a minimum weight connected subgraph $H$ of $G^*$ and $T^* \subseteq V(H)$. Here, $n$ is the number of vertices in $G^*$ and $\rho$ is the maximum weight assigned to any edge of $G$ by $w$.*

Next, we state some observations that follow directly from Lemma 12 and Corollary 8. These observations are stated here for the sake of completeness. Recall that for $V' \subseteq V(G)$ with $|V'| \geq 2$, we defined $N^{[2]}(V')$ to be the union of the sets of common neighbours of every pair of vertices in $V'$, i.e., $N_G^{[2]}(V') = (\bigcup_{\substack{u,v \in V' \\ u \neq v}} (N(u) \cap N(v))) \setminus V'$.

▶ **Observation 64.** *Let $(G, C, P, Y)$ be a cpy-graph and $k$ a non-negative integer. Let $I \subseteq Y$ be an independent set in $G$ of size $k+1$. Then, for any cpy-connected dominating set $D$ of $G$ of size at most $k$, $D \cap N^{[2]}(I) \neq \emptyset$.*

The proof of Observation 64 follows from Lemma 12.

▶ **Observation 65.** *Let $(G, C, P, Y)$ be a c-closed cpy-graph and $k$ a non-negative integer. Let $D$ be a cpy-connected dominating set of $G$ of size at most $k$, and $Q$ a maximal clique in $G$ of size at least $(c-1)k+1$. Then, $D \cap V(Q) \neq \emptyset$.*

The proof of Observation 65 follows from Corollary 8.

**Notation.** Consider a cpy-graph $(G, C, P, Y)$ and a vertex $v \in V(G) \setminus C$. By $(G, C_v, P_v, Y_v)$, we denote the cpy-graph obtained by adding $v$ to $C$, deleting $N_G[v] \cap Y$ from $Y$, and by adding $N_G(v) \cap Y$ to $P$. That is, $C_v = C \cup \{v\}$, $P_v = P \cup (N_G(v) \cap Y)$, $Y_v = Y \setminus (N_G[v] \cap Y)$. Recall that we defined $\beta(a,b) = 2[(a-1)(b-1)+1]$ for $a, b \in \mathbb{N}$. For $\ell \in \mathbb{N}$, we defined $\mathcal{Q}^{\beta(c,k+1)}(G)$ to be the set of all maximal cliques in $G$ of size at least $\ell$. Recall also that $L^{\beta(c,k+1)}(G) = \bigcup_{Q \in \mathcal{Q}^{\beta(c,k+1)}(G)} V(Q)$ and $M^{\beta(c,k+1)}(G) = V(G) \setminus L^{\beta(c,k+1)}(G)$. That is, $L^{\beta(c,k+1)}(G)$ contains all the vertices in $G$ that belong to at least one maximal clique of size at least $\beta(c,k+1)$, and $M^{\beta(c,k+1)}(G)$ contains the remaining vertices. We now define a subfamily of $\mathcal{Q}^{\beta(c,k+1)}(G)$ as follows. By $\mathcal{Q}_Y^{\beta(c,k+1)}(G)$ we denote the set of all cliques $Q \in \mathcal{Q}^{\beta(c,k+1)}(G)$ such that $|V(Q) \cap Y| \geq \beta(c,k+1)$; that is, $\mathcal{Q}_Y^{\beta(c,k+1)}(G) = \{Q \in \mathcal{Q}^{\beta(c,k+1)}(G) \mid |V(Q) \cap Y| \geq \beta(c,k+1)\}$. And we define $L_Y^{\beta(c,k+1)}(G) = \bigcup_{Q \in \mathcal{Q}_Y^{\beta(c,k+1)}(G)} V(Q) \cap Y$ and $M_Y^{\beta(c,k+1)}(G) = Y \setminus L_Y^{\beta(c,k+1)}(G)$. That is, $L_Y^{\beta(c,k+1)}(G)$ contains all the vertices in $Y$ that belong to at least one maximal clique that contains at least $\beta(c,k+1)$ vertices from the set $Y$, and $M_Y^{\beta(c,k+1)}(G)$ contains the remaining vertices of $Y$. For $Z \subseteq V(G)$ and a non-negative integer $\ell$, by $\mathfrak{B}(Z, \ell)$, we denote the set of all partitions of $Z$ into at most $\ell$ parts. For a partition $\mathcal{Z} \in \mathfrak{B}(Z, \ell)$, we define $\mathcal{Z}_{CN} = \{CN_G(X) | X \in \mathcal{Z}\}$. That is, $\mathcal{Z}_{CN}$ is the set of common neighbourhoods of the sets in $\mathcal{Z}$.

We first prove a few structural results that explore the properties of a cpy-connected dominating set. In what follows, $((G, C, P, Y), k)$ is an instance of CPY-CDS.

▶ **Observation 66.** *Let $Q$ be a clique in $G$ such that $V(Q) \cap Y \neq \emptyset$. Then $Q$ is a maximal clique in $G$ if and only if $Q$ is a maximal clique in $G[P \cup Y]$.*

Observation 66 follows from that fact that there does not exist any edge between a vertex in $C$ and a vertex in $Y$.

The following observation is a direct consequence of Lemma 9, where we proved that we can construct a $(k+1)$-sized independent set from $k+1$ maximal cliques of size $\beta(c,k+1)$. It is not difficult to see that we can construct a $(k+1)$-sized independent set contained in $Y$, provided that each of the $k+1$ maximal cliques intersect $Y$ in at least $\beta(c,k+1)$ vertices.

▶ **Observation 67.** *If $|\mathcal{Q}_Y^{\beta(c,k+1)}(G)| \geq k+1$, then $G[Y]$ contains an independent set of size $k+1$.*

▶ **Lemma 68.** *If $G[Y]$ does not contain an independent set of size $k+1$, then $|M_Y^{\beta(c,k+1)}(G)| < R_c(\beta(c,k+1), k+1)$.*

**Proof.** By the definition of $M_Y^{\beta(c,k+1)}(G)$, the subgraph $G[M_Y^{\beta(c,k+1)}(G)]$, does not contain any clique of size $\beta(c,k+1)$. And by our assumption, $G[Y]$, and hence $G[M_Y^{\beta(c,k+1)}(G)]$ does not contain an independent set of size $k+1$. The proof follows from Lemma 1. ◀

▶ **Lemma 69.** *Let $((G,C,P,Y),k)$ be an instance of* CPY-CDS*, and let $I \subseteq Y$ be an independent set of size $k+1$ in $G$. Then, $((G,C,P,Y),k)$ is a yes-instance of* CPY-CDS *if and only if $((G,C_v,P_v,Y_v),k)$ is a yes-instance for some $v \in N^{[2]}(I)$.*

**Proof.** Assume that $((G,C,P,Y),k)$ is a yes-instance of CPY-CDS, and let $D$ be a cpy-connected dominating set of $((G,C,P,Y),k)$ of size at most $k$. By Observation 64, $D \cap N^{[2]}(I) \neq \emptyset$. Let $v \in D \cap N^{[2]}(I)$. Then, $C \cup \{v\} \subseteq D$. Recall that $C_v = C \cup \{v\}$, which implies that $D$ is a cpy-connected dominating set of $(G,C_v,P_v,Y_v)$ of size at most $k$. This proves that $((G,C_v,P_v,Y_v),k)$ is a yes-instance of CPY-CDS.

Conversely, assume that $((G,C_v,P_v,Y_v),k)$ is a yes-instance of CPY-CDS for some $v \in N^{[2]}(I)$, and let $D'$ be a cpy-connected dominating set of $((G,C_v,P_v,Y_v),k)$ of size at most $k$. Then again, $C \subseteq C_v \subseteq D$, which implies that $D'$ is a cpy-connected dominating set of $(G,C,P,Y)$ of size at most $k$. This proves that $((G,C,P,Y),k)$ is a yes-instance of CPY-CDS. ◀

Next, we describe how to construct an instance of Steiner Tree from an instance of CPY-CDS.

▶ **Construction 70** (**Construction of a** Steiner Tree **instance**)**.** *Let $((G,C,P,Y),k)$ be an instance of* CPY-CDS *such that $G[Y]$ does not contain an independent set of size $k+1$. For $S \subseteq M_Y^{\beta(c,k+1)}(G)$, let $Z_S = M_Y^{\beta(c,k+1)}(G) \setminus N[S]$. With respect to every $S \subseteq M_Y^{\beta(c,k+1)}(G)$ with $|S \cup C| \leq k$, and every $\mathcal{Z} \in \mathfrak{B}(Z_S,k)$, we construct an instance $I_{(S,\mathcal{Z})} = (G^*,w,T^*)$ of the* Steiner Tree *problem as follows.*

*Informally, for each clique $Q \in \mathcal{Q}_Y^{\beta(c,k+1)}(G)$ (resp. for each set $X \in \mathcal{Z}_{CN}$), we add a terminal $t$ and make it adjacent to exactly all the vertices in $Q$ (resp. $X$), and thus ensure that a vertex from $Q$ (resp. $X$) must go into the solution. Moreover, we assign weight $k+1$ to all edges incident with $t$ to ensure that exactly one edge incident with $t$ goes into the solution. Figure 2 shows the construction. Now, we describe the construction formally.*

*We initialise $V(G^*) = V(G)$, $E(G^*) = E(G)$ and $T^* = C \cup S$. We also set $w(e) = 1$ for each $e \in E(G^*)$. Now, for each $Q \in \mathcal{Q}_Y^{\beta(c,k+1)}(G)$, we add a new vertex $t_Q$ and edges $E_Q = \{t_Q v \mid v \in V(Q)\}$ to $G^*$; and for each $e \in E_Q$, we set $w(e) = k+1$. We also add $t_Q$ to $T^*$. Let $T^1$ be the set of terminals added in this step to $T^*$. Also, for each set $X \in \mathcal{Z}_{CN}$, we add a new vertex $t_X$ and edges $E_X = \{t_X u \mid u \in X\}$ to $G^*$; and for each $e \in E_X$, we set $w(e) = k+1$. Finally, we add $t_X$ to $T^*$. Let $T^2_{(S,\mathcal{Z})}$ be the set of terminals added in this*

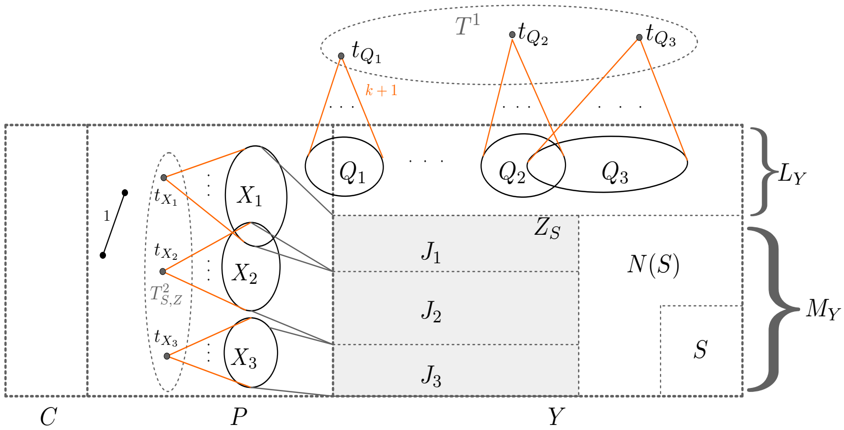


**Figure 2** Depicts the partition of the vertex set of $G$ into $C$, $P$, and $Y$ and construction of the Steiner Tree instance. The set $Y$ is further divided into $L_Y^{\beta(c,k+1)}(G)$ and $M_Y^{\beta(c,k+1)}(G)$, denoted by $L_Y$ and $M_Y$, respectively. Based on the guessed set $S$, $M_Y^{\beta(c,k+1)}(G)$ is further divided. The grey box depicts $Z_S = M_Y^{\beta(c,k+1)}(G) \setminus N[S]$. The tuple $(J_1, J_2, J_3)$ denotes a partition of the set $Z_S$ into 3 parts; and for each $i \in [3]$, $X_i$ denotes the set of common neighbours of $J_i$, i.e., $X_i = CN(J_i)$. It is not necessarily that $X_i \subseteq P$. For clarity of the figure we have depicted $X_i$s being contained in $P$. The sets $Q_1, Q_2$ and $Q_3$ denote "large" cliques. The vertices $t_{X_1}, t_{X_2}, t_{X_3}, t_{Q_1}, t_{Q_2}$, and $t_{Q_3}$ are the terminals created in the construction of the Steiner Tree instance (Construction 70); $t_{X_i}$ and $t_{Q_i}$ are adjacent to every vertex in $X_i$ and $Q_i$, respectively, and every edge (in orange) incident with $t_{X_i}$ and $t_{Q_i}$ has weight $k+1$; and each of the "original edges" of $G$ has weight 1 in the Steiner Tree instance.

*step to $T^*$. Note that given a cpy-graph $(G, C, P, Y)$, an integer $k$, $S \subseteq M_Y^{\beta(c,k+1)}(G)$ and a partition $\mathcal{Z}$ of $Z_S$, where $Z_S = M_Y^{\beta(c,k+1)}(G) \setminus N[S]$, an instance $I_{(S,\mathcal{Z})} = (G^*, w, T^*)$ of* Steiner Tree *can be constructed in polynomial time. Figure 2 shows the construction.*

▶ **Observation 71.** $|T^*| \leq |C \cup S| + k + |\mathcal{Q}_Y^{\beta(c,k+1)}(G)|$.

**Proof.** Observe the following facts:
**(i).** $T^* = C \cup S \cup T^1 \cup T^2_{(S,\mathcal{Z})}$;
**(ii).** $|T^1| \leq |\mathcal{Q}_Y^{\beta(c,k+1)}(G)|$; and
**(iii).** $|T^2_{(S,\mathcal{Z})}| \leq |\mathcal{Z}_{CN}| \leq |\mathcal{Z}| \leq k$.
The proof follows from (i)-(iii). ◀

▶ **Lemma 72.** *Consider an instance $((G, C, P, Y), k)$ of* CPY-CDS *such that $G[Y]$ has no independent set of size $k+1$. Then, $((G, C, P, Y), k)$ is a yes-instance of* CPY-CDS *if and only if there exists some $S \subseteq M_Y^{\beta(c,k+1)}(G)$ with $|C \cup S| \leq k$ and a partition $\mathcal{Z}$ of $Z_S$ into at most $k$ parts, where $Z_S = M_Y^{\beta(c,k+1)}(G) \setminus N[S]$, such that for the instance $I_{(S,\mathcal{Z})} = (G^*, w, T^*)$ of* Steiner Tree *there exists a solution $H$ such that $\sum_{e \in E(H)} w(e) \leq |T'|(k+1) + k - 1$, where $T' = T^1 \cup T^2_{(S,\mathcal{Z})}$, and $I_{(S,\mathcal{Z})}, T^1, T^2_{(S,\mathcal{Z})}$ are as described in Construction 70.*

**Proof.** Assume that $((G, C, P, Y), k)$ is a yes-instance of CPY-CDS, and let $D$ be a cpy-connected dominating set of $(G, C, P, Y)$ of size at most $k$. Then $G[D]$ is connected and $C \subseteq D$. Let $H'$ be a spanning tree of $G[D]$. Let $D \cap M_Y^{\beta(c,k+1)}(G) = S$. Thus $C \cup S \subseteq D$ and therefore $|C \cup S| \leq |D| \leq k$. **(a)** Observe that for each vertex $u \in Z_S = M_Y^{\beta(c,k+1)}(G) \setminus N[S]$, there exist a vertex $v \in D$ such that $v$ dominates $u$. Let $D' = \{v_1, v_2, \ldots, v_\ell\} \subseteq D$ be a minimal set of vertices in $D$ that dominates $Z_S$, i.e., $Z_S \subseteq N[D']$. Note that $\ell \leq |D| \leq k$. Let

$\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_\ell\}$ be a partition of $Z_S$ such that all the vertices in $Z_i$ are dominated by the vertex $v_i \in D'$ for every $i \in [\ell]$. That is, $v_i \in CN_G(Z_i)$. (Note that the partition $\mathcal{Z}$ need not be unique.) For each $i \in [\ell]$, let $X_i = CN_G(Z_i)$. Recall that $\mathcal{Z}_{CN} = \{CN_G(Z_i) \mid Z_i \in \mathcal{Z}\} = \{X_i \mid i \in [\ell]\}$. Observe that $v_i \in X_i \cap D$ for every $i \in [\ell]$. **(b)** By Observation 65, for each $Q \in \mathcal{Q}_Y^{\beta(c,k+1)}(G)$, we have $D \cap V(Q) \neq \emptyset$. Let $v_Q \in D \cap V(Q)$ for each $Q \in \mathcal{Q}_Y^{\beta(c,k+1)}(G)$. Now, consider the Steiner Tree instance $I_{(S,\mathcal{Z})} = (G^*, w, T^*)$ with respect to $S$ and $\mathcal{Z}$, as defined in Construction 70. **(1)** Recall that corresponding to each clique $Q \in \mathcal{Q}_Y^{\beta(c,k+1)}(G)$, the graph $G^*$ contains a terminal $t_Q \in T^*$ and edges $E_Q = \{t_Q v \mid v \in V(Q)\}$. Also, $T^1 = \left\{t_Q \mid Q \in \mathcal{Q}_Y^{\beta(c,k+1)}(G)\right\} \subseteq T^*$. By (b) there exists a vertex $v_Q \in D \cap V(Q)$. This implies that $v_Q$ is adjacent to $t_Q$ in $G^*$. Therefore, we can obtain a connected subgraph $H_1'$ of $G^*$ from $H'$ by adding vertices in $T^1$ and the edges in $\left\{t_Q v_Q \mid \mathcal{Q}_Y^{\beta(c,k+1)}(G)\right\}$ to $H'$. Note that $d_{H_1'}(t_Q) = 1$ for every $t_Q \in T^1$. **(2)** Recall also that corresponding to each set $X \in \mathcal{Z}_{CN}$, the graph $G^*$ contains a terminal $t_X$ and edges $E_X = \{t_X u \mid u \in X\}$. Also, $T^2_{(S,\mathcal{Z})} = \{t_X \mid X \in \mathcal{Z}_{CN}\}$. By (a) there exists a vertex $v_i \in X_i \cap D$ for every $i \in [\ell]$. This implies that $v_i$ is adjacent to $t_{X_i}$ in $G^*$ for every $i \in [\ell]$. We can thus obtain a connected subgraph $H$ of $G^*$ from $H_1'$ by adding the vertices in $T^2_{(S,\mathcal{Z})}$ and the edges $\{t_{X_i} v_i \mid i \in [\ell]\}$ to $H_1'$. Note that $d_{H_1'}(t_X) = 1$ for every $t_X \in T^2_{(S,\mathcal{Z})}$. Recall that $T^* = C \cup S \cup T^1 \cup T^2_{(S,\mathcal{Z})}$ and $(C \cup S) \subseteq D$. By (1) and (2), $T^* \subseteq V(H)$. Now, since $H'$ is a spanning tree of $G[D]$, $w(e) = 1$ for every $e \in E(H)$, and $|D| \leq k$, we have $\sum_{e \in E(H')} w(e) = |D| - 1 \leq k - 1$. Finally, since the vertices in $T' = T^1 \cup T^2_{(S,\mathcal{Z})}$ have degree 1 in $H$, we have $\sum_{e \in E(H) \setminus E(H')} w(e) \leq |T'|(k+1)$. Thus, $\sum_{e \in E(H)} w(e) \leq |T'|(k+1) + k - 1$, and therefore, $H$ is a required solution for the Steiner Tree instance $I_{(S,\mathcal{Z})} = (G^*, w, T^*)$.

Conversely, assume that for $S \subseteq M_Y^{\beta(c,k+1)}(G)$ and a partition $\mathcal{Z} \in \mathfrak{B}(Z_S, k)$ of $Z_S$, where $Z_S = M_Y^{\beta(c,k+1)}(G) \setminus N[S]$, the Steiner Tree instance $I_{(S,\mathcal{Z})} = (G^*, w, T^*)$ has a solution $H$ such that $\sum_{e \in E(H)} w(e) \leq |T'|(k+1) + k - 1$, where $T' = T^1 \cup T^2_{(S,\mathcal{Z})}$. We can assume that $H$ is a tree, for otherwise any spanning tree of $H$ is also a solution for the instance $I_{(S,\mathcal{Z})} = (G^*, w, T^*)$ with weight at most $|T'|(k+1) + k - 1$. **(I)** For each edge $e$ incident with the vertices of $T'$, we have $w(e) = k + 1$, and since $\sum_{e \in E(H)} w(e) \leq |T'|(k+1) + k - 1$, the vertices in $T'$ are leaves in $H$. Hence, $H^* = H[D^*]$ is a tree, where $D^* = V(H) \setminus T'$. Note that by the definition of the instance $I_{(S,\mathcal{Z})}$, we have $w(e) = 1$ for every $e \in E(H^*)$, and therefore, $\sum_{e \in E(H^*)} w(e) \leq k - 1$, which implies that $|D^*| = |E(H^*)| + 1 \leq k$. Since $G$ and $G^*$ differ only by the vertex set $T'$, observe that we have $E(H^*) \subseteq E(G)$, and therefore $H^* = H[D^*]$ is a subgraph of $G[D^*]$. Thus, $G[D^*]$ is connected. Now, we only need to prove that $D^*$ is a dominating set of $G$. **(II)** Consider a clique $Q \in \mathcal{Q}_Y^{\beta(c,k+1)}(G)$. Corresponding to $Q$, there exists a vertex $t_Q \in T^*$ with $N_{G^*}(t_Q) = V(Q)$. Since $t_Q$ is a terminal, and $T'$ does not contain any neighbour of $t_Q$, and since $H$ is connected, $D^* = V(H) \setminus T'$ must contain a neighbour of $t_Q$, which implies that $D^* \cap V(Q) = D \cap N_{G^*}(t_Q) \neq \emptyset$. Let $u_Q \in D^* \cap V(Q)$. Observe that $u_Q$ dominates $V(Q)$. By Observation 66, $Q$ is a maximal clique in $G[P \cup Y]$. Therefore, $u_Q \notin C$. Let $D_1 = \{u_Q \mid Q \in \mathcal{Q}_Y^{\beta(c,k+1)}(G)\}$. Note that $D_1 \subseteq D^*$. The above observations imply that $D_1$ dominates $L_Y^{\beta(c,k+1)}(G) = \bigcup_{Q \in \mathcal{Q}_Y^{\beta(c,k+1)}(G)} V(Q)$ in $G$. **(III)** Consider a set $X_i = CN_G(Z_i) \in \mathcal{Z}_{CN}$. Corresponding to $X_i$, there exists a vertex $t_{X_i} \in T^*$ with $N_{G^*}(t_{X_i}) = X_i$. Again, since $t_{X_i}$ is a terminal, and $T'$ does not contain any neighbour of $t_{X_i}$, and since $H$ is connected, $D^* = V(H) \setminus T'$ must contain a neighbour of $t_{X_i}$, which implies that $D^* \cap X_i = D \cap N_{G^*}(t_{X_i}) \neq \emptyset$. Let $u_{X_i} \in D^* \cap X_i$. Observe that $u_{X_i}$ dominates $Z_i \subseteq N_G(u_{X_i})$. Recall that $Z_i \subseteq Y$ and a vertex in $Y$ is not adjacent to any vertex in $C$, and hence $u_{X_i} \notin C$. Let $D_2 = \{u_{X_i} \mid X_i \in \mathcal{Z}_{CN}\}$. Note that $D_2 \subseteq D^*$. Recall that $\mathcal{Z} \in \mathfrak{B}(Z_S, k)$ is a partition of $Z_S$, where $Z_S = M_Y^{\beta(c,k+1)}(G) \setminus N[S]$, and therefore $D_2$ dominates $Z_S$.

Note also that as $S \subseteq T^* \setminus T'$, we have $S \subseteq V(H) \setminus T' = D^*$. These observations imply that $D_2 \cup S$ dominates $M_Y^{\beta(c,k+1)}(G)$ in $G$. **(IV)** Finally, as $C \subseteq T^* \setminus T'$, we also have $C \subseteq V(H) \setminus T' = D^*$. And for each vertex in $P$ there exists a neighbour in $C$, and therefore $C$ dominates $P$. By (II), (III) and (IV), $D' = D_1 \cup D_2 \cup S \cup C$ is a dominating set of $G$ and $D' \subseteq D^*$. Hence, by (I)-(IV), $D^*$ is a cpy-connected dominating set of $(G, C, P, Y)$ of size at most $k$, and thus $((G, C, P, Y), k)$ is a yes-instance of CPY-CDS. This completes the proof. ◀

We now describe our algorithm.

**Description of our algorithm: Algorithm 2.** We are given an instance $((G, C, P, Y), k)$ of CPY-CDS as input.

**Step 1.** First, if $k-|C| \geq 0$, $Y = \emptyset$ and $G[C]$ is connected, then we return that $((G, C, P, Y), k)$ is a yes-instance, and terminate. Otherwise, if $k - |C| > 0$, we do as follows. We use the algorithm in Corollary 4 to check if $G[Y]$ has an independent set of size $k + 1$. If the algorithm in Corollary 4 returns that $G[Y]$ has no such independent set, then we proceed to Step 1.1. On the other hand, if the algorithm in Corollary 4 returns a $(k + 1)$-sized independent set $I$, then we branch into $|N^{[2]}(I)|$ many instances of CPY-CDS. *For each $v \in N^{[2]}(I)$, we create the instance $((G, C_v, P_v, Y_v), k)$ and recursively call Step 1 on this instance.* On any branch, at any point if the algorithm in Corollary 4 returns a $(k + 1)$-sized independent set $I$ with $N^{[2]}(I) = \emptyset$, then we discard that branch. On all other branches, we recurse only until $k-|C| = 0$ or $Y = \emptyset$, whichever happens earlier. We note that on any branch, for each of the instances $((G, C_v, P_v, Y_v), k)$ that we create from $((G, C, P, Y), k)$, we have $|C_v| = |C \cup \{v\}| = |C| + 1$, and therefore, $k - |C_v| < k - |C|$. That is, $k - |C|$ decreases as we proceed along a branch.

**Step 1.1.** Use the algorithm in Lemma 5 to construct $\mathcal{Q}^{\beta(c,k+1)}(G)$. For each set $S \subseteq M_Y^{\beta(c,k+1)}(G)$ with $|S \cup C| \leq k$ and for each $\mathcal{Z} \in \mathfrak{B}(Z_S, k)$, we construct the instance $I_{(S,\mathcal{Z})} = (G^*, w, T^*)$ of Steiner Tree. We solve the Steiner Tree instance $I_{(S,\mathcal{Z})} = (G^*, w, T^*)$ using the algorithm in Lemma 63. Let $H$ be the solution returned by the algorithm in Lemma 63. Let $T^1, T^2_{(S,\mathcal{Z})} \subseteq T^*$ be as defined in the Construction 70 of the instance $I_{(S,\mathcal{Z})} = (G^*, w, T^*)$. Then, if $\sum_{e \in E(H)} w(e) \leq |T'|(k + 1) + k - 1$, where $T' = T^1 \cup T^2_{(S,\mathcal{Z})}$, then we return that $((G, C, P, Y), k)$ is a yes-instance, and terminate.

**Step 2.** We return that $((G, C, P, Y), k)$ is a no-instance, and terminate.

This completes the description of the algorithm. The correctness of Step 1 follows from Lemma 69. The correctness of Step 1.1 follows from Lemma 72. Note that the algorithm enters Step 2 only if we have not already returned that the input instance is a yes-instance. And Lemmas 69 and 72 together imply that if $((G, B, W), k)$ is indeed a yes-instance, then we correctly return yes (in Step 1 or Step 1.1). Hence Step 2 is also correct. These observations show that Algorithm 2 is correct. Now, we analyse its runtime in the following lemma.

▶ **Lemma 73.** *Algorithm 2 runs in time $2^{\mathcal{O}(c+k\log(ck))} n^{\mathcal{O}(1)}$.*

**Proof.** Recall that $\beta(c, k + 1) = 2[(c - 1)k + 1]$. Therefore, $R_c(\beta(c, k + 1), k + 1) = (c - 1)\binom{k}{2} + (2(c - 1)k + 1)(k + 1) = \mathcal{O}(ck^2)$.

Consider Step 1.1. By Lemma 5, we can construct $\mathcal{Q}^{\alpha(c,k)}(G)$ in time $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$. By Lemma 68, we have $|M_Y^{\beta(c,k+1)}(G)| < R_c(\beta(c, k + 1), k + 1) = \mathcal{O}(ck^2)$. For every subset $S \subseteq M_Y^{\beta(c,k+1)}(G)$ with $|S \cup C| \leq k$ and every partition $\mathcal{Z}$ of $Z_S$ into at most $k$ parts, where $Z_S = M_Y^{\beta(c,k+1)}(G) \setminus N[S]$, the algorithm constructs an instance $I_{(S,\mathcal{Z})} = (G^*, w, T^*)$ of Steiner Tree in polynomial time. Note that the number of choices for $S$ is at most

$\sum_{j=0}^{k} \binom{|M_Y^{\beta(c,k+1)}(G)|}{j} = \sum_{j=0}^{k} \binom{\mathcal{O}(ck^2)}{j} \leq (k+1) \cdot (\mathcal{O}(ck^2))^k = 2^{\mathcal{O}(k \log(ck))}$. The number of choices for $\mathcal{Z}$ is at most $|M_Y^{\beta(c,k+1)}(G)|^k = (\mathcal{O}(ck^2))^k = 2^{\mathcal{O}(k \log(ck))}$. Therefore, the number of choices for the pair $(S, \mathcal{Z})$ is at most $2^{\mathcal{O}(k \log(ck))} \cdot 2^{\mathcal{O}(k \log(ck))} = 2^{\mathcal{O}(k \log(ck))}$. Thus, the the algorithm constructs $2^{\mathcal{O}(k \log(ck))}$ many instances of Steiner Tree. By Lemma 63, the algorithm takes $\mathcal{O}(2^{|T^*|} \cdot \rho \cdot n^{\mathcal{O}(1)})$ time for each instance of Steiner Tree. Now we compute the value of $|T^*|$. Observe the following property of the instance CPY-CDS instance $((G, C, P, Y), k)$ when the algorithm enters Step 1.1. The subgraph $G[Y]$ has no independent set of size $k+1$, and therefore, by Observation 67, we have $|\mathcal{Q}_Y^{\beta(c,k+1)}(G)| \leq k$. Recall also that we have $|S \cup C| \leq k$ Now, by Observation 71, the number of terminals in each of the Steiner Tree instances $I_{(S,\mathcal{Z})}$ is at most $|C \cup S| + k + |\mathcal{Q}_Y^{\beta(c,k+1)}(G)| = \mathcal{O}(k)$. Also, in each of the Steiner Tree instances $I_{(S,\mathcal{Z})}$, the maximum weight of any edge is $k+1$. These observations, along with Lemma 63, imply that each of the instances $I_{(S,\mathcal{Z})}$ of Steiner Tree can be solved in time $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$. Therefore, the total time taken by one execution of Step 1.1 is bounded by $2^{\mathcal{O}(c)} n^{\mathcal{O}(1)} + 2^{\mathcal{O}(k \log(ck))} \cdot 2^{\mathcal{O}(k)} n^{\mathcal{O}(1)} \leq 2^{\mathcal{O}(c + k \log(ck))} n^{\mathcal{O}(1)}$.

Now, consider one execution of Step 1. By Corollary 4, finding a $(k+1)$-sized independent set $I$ takes time $2^{k \log(ck)} n^{\mathcal{O}(1)}$. Note that in one execution of Step 1, at most $|N^{[2]}(I)|$ recursive calls to Step 1 are being made; and by Lemma 12, $|N^{[2]}(I)| \leq (c-1)\binom{k+1}{2}$. Note also that recursive calls to Step 1 are made only until $k = 0$. Thus the total number of recursive calls made to Step 1 is bounded by $((c-1)\binom{k+1}{2})^k = 2^{\mathcal{O}(k \log(ck))}$.

Hence the total running time of the algorithm is bounded by $2^{\mathcal{O}(k \log(ck))} \cdot 2^{\mathcal{O}(c + k \log(ck))} n^{\mathcal{O}(1)} = 2^{\mathcal{O}(c + k \log(ck))} n^{\mathcal{O}(1)}$. ◀

We have thus proved the following theorem.

▶ **Theorem 74.** CPY-CDS *on $c$-closed graphs admits an algorithm running in time* $2^{\mathcal{O}(c + k \log(ck))} n^{\mathcal{O}(1)}$.

Since we can reduce an instance $(G, k)$ of CDS into an equivalent instance $((G, R, P, Y), k)$ of CPY-CDS in polynomial time, Theorem 74 implies the following result.

▶ **Theorem 75.** CDS *on $c$-closed graphs admits an algorithm that runs in time* $2^{\mathcal{O}(c + k \log(ck))} n^{\mathcal{O}(1)}$.

## 5 Partial Dominating Set on $c$-Closed Graphs

For a graph $G$ and a non-negative integer $t$, a $t$-partial dominating set of $G$ is a vertex subset $V' \subseteq V(G)$ that dominates at least $t$ vertices of $G$, i.e., $|N_G[V']| \geq t$. In the Partial Dominating Set (PDS) problem, we are given a graph $G$ and two non-negative integers $k$ and $t$ as input, and the task is to decide if $G$ has a $t$-partial dominating set of size at most $k$. In this section, we show that PDS (parameterized by $k$) is W[1]-hard even on 2-closed graphs. We do this by a reduction from Independent Set on regular graphs, which is known to be W[1]-complete [11].

▶ **Lemma 76.** *There is a parameterized reduction from* Independent Set *on regular graphs to* PDS *on 2-closed graphs.*

**Proof.** Let $(G, k)$ be an instance of Independent Set, where $G$ is a regular graph. Assume that $G$ is $r$-regular for some $r \geq 3$. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$ and $E(G) = \{e_1, e_2, \ldots, e_m\}$.

We construct an instance $(G', k', t)$ of PDS as follows. Informally, $G'$ is obtained by subdividing every edge of $G$. More formally, we take $V(G') = X \cup Y$, where $X = \{x_i \mid i \in [n]\}$

and $Y = \{y_i \mid i \in [m]\}$; and $E(G') = \{x_i y_j \mid v_i \text{ is an endpoint of } e_j, i \in [n], j \in [m]\}$. Finally, we set $k' = k$ and $t = k(r+1)$. Note that $G'$ can be constructed in polynomial time, and the reduction preserves the parameter as $k' = k$. Also, observe that $G'$ is 2-closed, as any two distinct vertices in $G'$ have at most 1 common neighbour. For two distinct vertices $x_i, x_j \in X$, $\{i,j\} \subseteq [n]$, if $e_\ell = v_i v_j \in E(G)$, then $x_i$ and $x_j$ have exactly one common neighbour $y_\ell$, and otherwise, they have no common neighbour. For $u \in Y$ and $x_i \in X$, they do not have a common neighbour, since $N(u) \subseteq X$ and $N(x_i) \subseteq Y$. Also, no two vertices in $Y$ have more than one common neighbor by the definition of $E(G')$.

Now, to see that $(G,k)$ and $(G',k',t)$ are equivalent instances, observe the following properties of $G'$, which follow from the definitions of $E(G')$. (i) The sets $X$ and $Y$ are independent sets in $G'$. (ii) For each $x_i \in X$, we have $N_{G'}(x_i) = \{y_j \in Y \mid e_j \text{ is incident to } v_i \text{ in } G\}$, and therefore, $|N_{G'}(x_i)| = d_G(v_i) = r$. (iii) For distinct $x_i, x_j \in X$ such that $v_i v_j \notin E(G)$, we have $N(x_i) \cap N(x_j) = \emptyset$. (iv) For each $x_i \in X$, $d_{G'}(x_i) = |N(x_i)| \overset{\text{(ii)}}{=} r$ and for each $y_i \in Y$, $d_{G'}(y_i) = 2$.

We now claim that $(G,k)$ is a yes-instance of Independent Set if and only if $(G',k',t)$ is a yes-instance of PDS. Assume that $(G,k)$ is a yes-instance of Independent Set, and let $S \subseteq V(G)$ be an independent set in $G$ of size $k$. We define $S' \subseteq V(G')$ as follows: $S' = \{x_i \in X \mid v_i \in S\}$. And we claim that $S'$ is a $t$-partial dominating set of $G'$. We have $N[S'] = \bigcup_{x_i \in S'} N[x_i] = S' \cup \bigcup_{x_i \in S'} N(x_i)$. Since, for each $i \in [n]$, $N(x_i) \subseteq Y$, we have that $S' \cap \bigcup_{x_i \in S'} N(x_i) = \emptyset$. By property (iii) observed above, we have $|N[S']| = |S'| + \sum_{x_i \in S'} |N(x_i)| \overset{\text{(iv)}}{=} k + \sum_{x_i \in S'} r = k(r+1) = t$. Thus, $S'$ is indeed a $t$-partial dominating set of $G'$ of size $k$.

Now, assume that $(G',k',t)$ is a yes-instance of PDS, and let $T' \subseteq V(G')$ be a $t$-partial dominating set of $G'$ of size at most $k$. Note that for every $w \in T'$, by property (iv), $d_{G'}(w) \leq r$. Now, observe that $|T'| = k$, for otherwise, $|N[T']| \leq |T'| + \sum_{w \in T'} d_{G'}(w) \leq (k-1) + (k-1)r < k(r+1) = t$, which contradicts the fact that $T'$ is a $t$-partial dominating set. Observe also that $T' \subseteq X$. Otherwise, suppose that $|T' \cap Y| = \ell$ for some $0 < \ell \leq k$. Thus, since $V(G') = X \cup Y$, we have

$$
\begin{aligned}
|N[T']| &= |N[T' \cap X]| + |N[T' \cap Y]| \\
&\leq |T'| + \sum_{w \in T' \cap X} d_{G'}(w) + \sum_{v \in T' \cap Y} d_{G'}(v) \\
&\overset{\text{(iv)}}{=} k + r(k-\ell) + 2\ell \\
&< k(r+1) \text{ for } r \geq 3.
\end{aligned}
$$

The second last equality follows from the degree bounds in property (iv) observed above. The last inequality is true whenever $\ell > 0$ and $r \geq 3$. Therefore, $|N[T']| < t$, which again, contradicts the fact that $T'$ is a $t$-partial dominating set. Thus, $T' \subseteq X$ and $|T'| = k$. Now, consider the set $T \subseteq V(G)$ defined as follows: $T = \{v_i \mid x_i \in T'\}$. We claim that $T$ is an independent set in $G$. Suppose not. Then, there exist $v_i, v_j \in T$ such that $v_i v_j \in E(G)$. Let $e_\ell = v_i v_j$. But then note that $y_\ell \in N_{G'}(x_i) \cap N_{G'}(x_j)$. Thus, $|N_{G'}(x_i) \cup N_{G'}(x_j)| < |N_{G'}(x_i)| + |N_{G'}(x_j)|$, which implies that $|N_{G'}[T']| = |T' \cup \bigcup_{w \in T'} N_{G'}(w)| < |T'| + \sum_{w \in T'} N_{G'}(w) = k + kr = t$, a contradiction. We have thus shown that $T$ is an independent set of size $k$ in $G$, and therefore, $(G,k)$ is a yes-instance of Independent Set. ◀

Lemma 76, along with the fact that Independent Set on regular graphs is W[1]-complete [11], implies the following result.

▶ **Theorem 77.** PDS *parameterized by the solution size is* W[1]*-hard on 2-closed graphs.*

## 6 Conclusion

We resolved the parameterized complexity of three domination problems—Perfect Code, Connected Dominating Set and Partial Dominating Set—on $c$-closed graphs. In particular, we showed that Perfect Code is fixed-parameter tractable, and that for each fixed $c$, Perfect Code admits a polynomial kernel on $c$-closed graphs, and thus settled a question posed by Koana et al. [43]. We believe that our results, along with that of Koana et al. [43, 45], make a convincing case for continuing the study of closure of a graph as a structural parameter. In the course of proving our results, we exploited several structural and algorithmic properties of $c$-closed graphs. It would be interesting to see if any of these properties can be used to solve other problems on $c$-closed graphs. It would also be interesting to see if any our results extend to weakly $\gamma$-closed graphs (see [28] and [42]).

## 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. Balaram Behera, Edin Husic, Shweta Jain, Tim Roughgarden, and C. Seshadhri. FPT algorithms for finding near-cliques in c-closed graphs. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 17:1–17:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.17.
7. Markus Bläser. Computing small partial coverings. *Inf. Process. Lett.*, 85(6):327–331, 2003. doi:10.1016/S0020-0190(02)00434-9.
8. 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.
9. J. Adrian Bondy and Uppaluri S. R. Murty. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2008. doi:10.1007/978-1-84628-970-5.
10. Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM J. Comput.*, 22(3):560–572, 1993. doi:10.1137/0222038.
11. Leizhen Cai. Parameterized complexity of cardinality constrained optimization problems. *Comput. J.*, 51(1):102–121, 2008. doi:10.1093/comjnl/bxm086.
12. 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.
13. Marco Cesati. Perfect code is W[1]-complete. *Inf. Process. Lett.*, 81(3):163–168, 2002. doi:10.1016/S0020-0190(01)00207-1.
14. Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1):2, 2006. doi:10.1145/1132952.1132954.

15 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

16 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.

17 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.

18 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.

19 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.

20 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.

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

22 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.

23 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.

24 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.

25 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.

26 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.

27 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.

28 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.

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

30 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.

31 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.

**32** 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.

**33** 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.

**34** 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.

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

**36** 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.

**37** Lawqueen Kanesh, Jayakrishnan Madathil, Sanjukta Roy, Abhishek Sahu, and Saket Saurabh. Further exploiting c-closure for FPT algorithms and kernels for domination problems. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 39:1–39:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.STACS.2022.39.

**38** 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.

**39** 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.

**40** 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.

**41** Tomohiro Koana, Christian Komusiewicz, André Nichterlein, and Frank Sommer. Covering many (or few) edges with k vertices in sparse graphs. In *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, pages 42:1–42:18, 2022. doi:10.4230/LIPIcs.STACS.2022.42.

**42** 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.

**43** 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.

**44** Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Essentially tight kernels for (weakly) closed graphs. In *32nd International Symposium on Algorithms and Computation, ISAAC 2021, December 6-8, 2021, Fukuoka, Japan*, pages 35:1–35:15, 2021. doi:10.4230/LIPIcs.ISAAC.2021.35.

**45** Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Exploiting \$c\$-closure in kernelization algorithms for graph problems. *SIAM J. Discret. Math.*, 36(4):2798–2821, 2022. doi:10.1137/21m1449476.

**46** Tomohiro Koana and André Nichterlein. Detecting and enumerating small induced subgraphs in c-closed graphs. *Discret. Appl. Math.*, 302:198–207, 2021. doi:10.1016/j.dam.2021.06.019.

**47** 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.

**48** 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.

**49** 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.

**50** 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.

**51** 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.

**52** 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.

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

**54** Jesper Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, 2013. doi:10.1007/s00453-012-9630-x.

**55** 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.

**56** 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.