Parameterized Algorithmics and Computational Experiments for Finding 2-Clubs

Sepp Hartung[1], Christian Komusiewicz[1], and André Nichterlein[1]

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany
{sepp.hartung, christian.komusiewicz, andre.nichterlein}@tu-berlin.de

Abstract. Given an undirected graph $G = (V, E)$ and an integer $\ell \geq 1$, the NP-hard 2-Club problem asks for a vertex set $S \subseteq V$ of size at least ℓ such that the subgraph induced by S has diameter at most two. In this work, we extend previous parameterized complexity studies for 2-Club. On the positive side, we give polynomial kernels for the parameters "feedback edge set size of G" and "size of a cluster editing set of G" and present a direct combinatorial algorithm for the parameter "treewidth of G". On the negative side, we first show that unless NP $\subseteq$ coNP/poly, 2-Club does not admit a polynomial kernel with respect to the "size of a vertex cover of G". Next, we show that, under the strong exponential time hypothesis, a previous $O^*(2^{|V|-\ell})$ search tree algorithm [Schäfer et al., Optim. Lett. 2012] cannot be improved and that, unless NP $\subseteq$ coNP/poly, there is no polynomial kernel for the dual parameter $|V| - \ell$. Finally, we show that, in spite of this lower bound, the search tree algorithm for the dual parameter $|V| - \ell$ can be tuned into an efficient exact algorithm for 2-Club that substantially outperforms previous implementations.

1 Introduction

Finding cohesive subnetworks in a network is an important task in graph-based data mining and social network analysis. The natural cohesiveness requirement is to demand that the subnetwork is a complete graph, that is, a clique. However, this requirement is often too restrictive and thus relaxed versions such as s-cliques [1], s-plexes [24], and s-clubs [19] have been proposed. An s-club is a graph with diameter at most s, and s-clubs are thus a distance-based relaxation of cliques (which are exactly the graphs of diameter 1). For a constant integer $s \geq 1$, the problem of finding large s-clubs is defined as follows.

s-Club
Input: An undirected graph $G = (V, E)$ and an integer $\ell \geq 1$.
Question: Is there a vertex set $S \subseteq V$ of size at least ℓ such that $G[S]$ has diameter at most s?

Clearly, 1-Club is equivalent to the well-known Clique problem. In this work, we consider 2-Club, the most basic variant of s-Club that is different from Clique. Furthermore, 2-Club is also an important special case concerning the applications: For biological networks, 2-clubs and 3-clubs have been identified as the most reasonable diameter-relaxations of Clique [21], further applications of 2-Club arise in the analysis of social networks [18]. Consequently, experimental evaluations concentrate on finding 2-clubs and 3-clubs [7, 8, 17].

Related Work. For all $s \geq 1$, s-Club is NP-complete on graphs of diameter $s+1$ [3]; 2-Club is NP-complete even on split graphs and, thus, also on chordal graphs [3]. For all $s \geq 1$, s-Club is NP-hard to approximate within a factor of $n^{1/2-\epsilon}$ [2]; a simple approximation algorithm obtains a factor of $n^{1/2}$ for even $s \geq 2$ and a factor of $n^{2/3}$ for odd $s \geq 3$ [2]. Several heuristics [7] and integer linear programming formulations [3, 7] for s-Club have been proposed and experimentally evaluated [17]. 1-Club is equivalent to Clique and thus W[1]-hard with respect to ℓ. In contrast, for $s \geq 2$, s-Club is fixed-parameter tractable with respect to ℓ [8, 22, 23]. s-Club can be solved in $O^*(2^{n-\ell})$ time by a search tree algorithm [8, 22, 23][1]. s-Club can be formulated in monadic second order logic and thus is fixed-parameter tractable with respect to the treewidth of G [23]. Moreover, s-Club does not admit a polynomial kernel with respect to ℓ (unless NP $\subseteq$ coNP/poly), but a so-called *Turing*-kernel with at most k^2 vertices for even s and at most k^3 vertices for odd s [22]. 2-Club is solvable in polynomial time on bipartite graphs, on trees, and on interval graphs [23].

Our Contribution. We make progress towards a systematic classification of the complexity of 2-Club with respect to structural parameters of the input graph. In Section 2, we give an $O(k^2)$-vertex kernel for the parameter "size of a cluster editing set" and an $O(k)$-size kernel for the parameter "feedback edge set size". The kernelization results for these rather large parameters are motivated by our negative results: We show that 2-Club does not admit a polynomial kernel with respect to the size of a vertex cover of the underlying graph, unless NP $\subseteq$ coNP/poly. This excludes polynomial kernels for many prominent structural parameters such as "feedback vertex set size", pathwidth, and treewidth. In Section 3, we give a direct combinatorial algorithm solving 2-Club in $2^{O(2^\omega)}n^2$ time

[1] Schäfer et al. [22] actually considered finding an s-club of size *exactly* ℓ. The claimed fixed-parameter tractability with respect to $n-\ell$ however only holds for the problem of finding an s-club of size *at least* ℓ. The other fixed-parameter tractability results hold for both variants.

on graphs of treewidth ω. Notably, up to a constant in the exponent, this is also the current best running time for the parameter vertex cover size (which we present in Theorem 4). In Section 4, we study s-Club, $s \geq 2$, parameterized by the dual parameter $k' := n - \ell$. We prove that unless the *Strong Exponential Time Hypothesis* (SETH)[2] fails, s-Club cannot be solved in $O^*((2-\epsilon)^{k'})$ time for all $\epsilon > 0$. This is evidence that the previous search tree algorithm [22] is optimal with respect to the parameter k'. Moreover, the presented reduction also implies that s-Club does not admit a polynomial kernel with respect to k' unless NP $\subseteq$ coNP/poly.

Having explored the limits of parameterized algorithmics for the dual parameter k' on the theoretical side, in Section 5 we examine its usefulness for solving 2-Club in practice. To this end, we implemented the search tree strategy for the dual parameter together with data reduction rules that are partially deduced from our findings in Section 2. We show that our implementation outperforms all previously implemented exact algorithms for 2-Club on random and on large-scale real world graphs. Especially on large graphs the concept of Turing kernelization turns out to be the most efficient technique in our "parameterized toolbox".

Preliminaries. We only consider undirected and simple graphs $G = (V, E)$ where $n := |V|$ and $m := |E|$. For a vertex set $S \subseteq V$, let $G[S]$ denote the *subgraph induced by* S and $G - S := G[V \setminus S]$. We use $\text{dist}_G(u, v)$ to denote the *distance between* u *and* v in G, that is, the length of a shortest path between u and v. For a vertex $v \in V$ and an integer $t \geq 1$, denote by $N_t(v) := \{u \in V \setminus \{v\} \mid \text{dist}(u, v) \leq t\}$ the set of vertices within distance at most t to v. Moreover, set $N_t[v] := N_t(v) \cup \{v\}$, $N[v] := N_1[v]$, and $N(v) := N_1(v)$. Two vertices v and w are *twins* if $N(v) \setminus \{w\} = N(w) \setminus \{v\}$. The following simple observation will be used throughout this work.

Observation 1. Let S be an s-club in a graph $G = (V, E)$ and let $u, v \in V$ be twins. If $u \in S$ and $|S| > 1$, then $S \cup \{v\}$ is also an s-club in G.

For the relevant notions of parameterized complexity refer to [11, 12, 20]. A more recent concept not presented in these monographs is *Turing kernelization*. Roughly speaking, in Turing kernelization one creates polynomially many problem kernels instead of one problem kernel. Then, the solution to the parameterized problem can be computed by solving the problem separately on each of these problem kernels. Throughout this

[2] The SETH fails if the satisfiability problem for boolean formulas in conjunctive normal form, the so-called Cnf-Sat problem, is solvable in $O^*((2-\epsilon)^n)$ time for some $\epsilon > 0$ where n denotes the number of variables; a recent survey on the (S)ETH is given by Lokshtanov et al. [16].

work, we assume that, unless stated otherwise, the structural parameter under consideration is provided as an additional input of the 2-Club instance. Due to the space restrictions, most of the proofs are deferred to a long version of this article.

2 Kernelization Algorithms and Lower Bounds

In this section, we provide polynomial-size problem kernels for 2-Club parameterized by "cluster editing set size" and "feedback edge set size", respectively. While these parameters can often be rather large, we show that for the (also relatively large) parameter "vertex cover size of G", there exists no polynomial-size problem kernel (unless NP $\subseteq$ coNP/poly).

A Quadratic-Vertex Kernel for the Parameter Cluster Editing Set Size. A cluster editing set of G is a set of edge additions and deletions that transforms G into a vertex-disjoint union of cliques. Let $G = (V, E)$, an integer ℓ, and a cluster editing set D be an instance of 2-Club; the parameter is $k := |D|$. Denote by $V(D)$ the set of all endpoints of the edges in D and observe that $G - V(D)$ is a cluster graph. The following rules yield an $O(k^2)$-vertex kernel for 2-Club. The first reduction rule is obvious.

Rule 1. If there is a cluster C in $G - V(D)$ with $|C| \geq \ell$, then reduce to a trivial yes-instance.

It follows that any 2-club of size at least ℓ has a nonempty intersection with $V(D)$, implying the correctness of the following data reduction rule.

Rule 2. If there is a cluster C in $G - V(D)$ such that $N(v) \cap V(D) = \emptyset$ for all $v \in C$, then delete C.

After exhaustive application of Rule 2, at most $|V(D)| \leq 2k$ clusters remain in $G - V(D)$. Since Rule 1 has been applied, each cluster in $G - V(D)$ has size at most $\ell - 1$. Hence, if $\ell \leq 2k + 1$, then there are at most $4k^2 + 2k$ vertices left and we are done. Now, consider the case where $\ell > 2k + 1$. To bound the size of the clusters in $G - V(D)$ we use the following observation. Its correctness follows from the fact that two vertices in different clusters of $G - V(D)$ are not adjacent and have no common neighbor.

Observation 2. For every 2-club S in G there is at most one cluster C in $G - V(D)$ such that S has a nonempty intersection with C.

Observation 2 implies that every 2-club of size at least ℓ contains at least $\ell - 2k$ vertices from exactly one cluster C of $G - V(D)$. Since all vertices

in C are twins, Observation 1 now implies that in an inclusion-maximal 2-club either all or no vertices from C or are contained. Hence, for $\ell > 2k+1$ decreasing ℓ and the size of each cluster C by $\ell - 2k - 1$ produces an equivalent instance. This leads to the following data reduction rule.

Rule 3. Delete $\ell - 2k - 1$ arbitrary vertices in each cluster C of $G - V(D)$ and set $\ell := 2k + 1$.

Note that in case $|C| \leq l - 2k - 1$ we simply delete all vertices of C. After exhaustive application of Rule 3 for each cluster C it holds that $1 \leq |C| < 2k + 1$. Thus, we arrive at the following.

Theorem 1. 2-Club *parameterized by the cluster editing set size k admits an $O(k^2)$-vertex kernel that can be computed in $O(n + m)$ time.*

A Linear Kernel for the Parameter Feedback Edge Set Size. A *feedback edge set* of a graph is an edge set whose deletion leads to a forest. Let $F \subset E$ be a feedback edge set for $G = (V, E)$. Furthermore, let $T := (V, E \setminus F)$ be the forest obtained by deleting F from G. The correctness of the first data reduction rule follows from the fact that for each vertex v the set $N[v]$ is a 2-club.

Rule 4. If there is a vertex $v \in V$ with $|N[v]| \geq \ell$, then reduce to a trivial yes-instance.

In the following, we exploit that after application of Rule 4 all 2-clubs of size at least ℓ have to "use" feedback edges. The next rule removes all vertices that are not on paths between the endpoints between feedback edges. These vertices are defined as follows.

Definition 1. For a feedback edge $\{u, v\} \in F$ the path $P_{\{u,v\}}$ between u and v in T is called *feedback edge path*. If a vertex w lies on the path $P_{\{u,v\}}$, then the edge $\{u, v\}$ is a *spanning feedback edge* of w.

Rule 5. Let (G, ℓ) be reduced with respect to Rule 4. Then, delete all vertices that do not lie on any feedback edge path.

The final rule removes vertices that are too far away from feedback edges.

Rule 6. If there is a vertex v that has in G distance at least three to at least one endpoint of every spanning feedback edge, then remove v.

Applying these data reduction rules exhaustively results in a linear kernel:

Theorem 2. *The* 2-Club *problem parameterized by the size k of a feedback edge set admits a kernel of size $6k$ that can be computed in $O(n^4)$ time.*

A Kernelization Lower Bound for the Parameter Vertex Cover. We next show that 2-Club does not admit a polynomial kernel with respect to the parameter vertex cover size. This result implies that 2-Club does not admit a polynomial kernel for many structural graph parameters such as feedback vertex set number or treewidth.

Theorem 3. 2-Club *parameterized by vertex cover has no polynomial kernel unless* $NP \subseteq coNP/poly$.

3 Fixed-Parameter Tractability with respect to Treewidth

In this section, we show that 2-Club is fixed-parameter tractable when parameterized by treewidth. To demonstrate the principle idea behind the algorithm, we first describe a fixed-parameter algorithm for the parameter vertex cover.

Theorem 4. s-Club *is solvable in* $O(2^k \cdot 2^{2^k} \cdot nm)$ *time where* k *denotes the size of a vertex cover.*

Extending the ideas behind Theorem 4, we now give a direct combinatorial algorithm for the parameter treewidth which uses the following lemma.

Lemma 1. *Let* G *be a graph and let* S *be a 2-club in* G*. Then, for any tree-decomposition of* G *there is at least one vertex* $v \in S$ *such that there is a bag that contains* $N[v] \cap S$*.*

Proof. Let $T = (X_1 \cup \ldots \cup X_r, E)$ be a tree-decomposition of G. Fix an arbitrary vertex $u \in S$ and denote by X^u any bag in T that contains u. Now, choose a vertex $w \in S$ such that the length of the path from X^u to the first bag that contains w is maximum. Denote this bag by X^w. We show that $N(w) \cap S \subseteq X^w$. Suppose there is a neighbor $v \in N(w) \cap S$ that is not contained in X^w. Since v and w are adjacent and thus are together contained in at least one bag, v is not contained in any bag on the path between X^u and X^w. This implies that the path from X^u to the first bag that contains v is longer than the path between X^u and X^w; a contradiction to the choice of w. □

Theorem 5. 2-Club *is solvable in* $2^{O(2^\omega)} \cdot n^2$ *time where* ω *denotes the treewidth.*

4 Optimality of Dual Parameter Algorithm

In this section, we prove algorithmic lower bounds for s-Club when parameterized by the dual parameter $k' := n - \ell$. We first show that there is a reduction from Cnf-Sat to s-Club with certain properties that allows to infer these lower bounds.

Lemma 2. *There is a parameterized reduction from* Cnf-Sat *to* s-Club *where the dual parameter in the constructed* s-Club *instance is equal to the number of variables in the boolean formula of the* Cnf-Sat *instance.*

Denoting the number of variables in a boolean formula by n, the Strong Exponential Time Hypothesis (SETH) fails if Cnf-Sat can be solved in $O^*((2-\epsilon)^n)$ time for some $\epsilon > 0$ [15]. Thus, by Lemma 2 an algorithm for s-Club running in $O^*((2-\epsilon)^{k'})$ time for some $\epsilon > 0$ would disprove the SETH. This bound is tight since s-Club can be solved in $O^*(2^{k'})$ time [22].

Corollary 1. *Unless the SETH fails,* s-Club *parameterized by the dual parameter* $k' := n - \ell$ *cannot be solved in* $O^*((2-\epsilon)^{k'})$ *time for all* $s \geq 2$.

Chen et al. [9] showed that Cnf-Sat does not admit a polynomial kernel unless coNP $\subseteq$ NP/poly. Since Lemma 2 provides a polynomial time and parameter transformation [6] this lower bound result transfers to 2-Club.

Corollary 2. s-Club *parameterized by the dual parameter* k' *does not admit a polynomial problem kernel unless coNP* $\subseteq$ *NP/poly.*

5 Implementation and Experiments

Search tree. We implemented the following search tree strategy to find a maximum 2-club S in a given graph $G = (V, E)$: If G is not a 2-club, then find a vertex $v \in V$ such that $|N_2(v)|$ is minimum among all vertices. Then, branch into the cases to either delete v from G or to mark v to be contained in S and subsequently delete all vertices in $V \setminus N_2[v]$. During branching we maintain a lower bound, that is, the size ℓ' of a largest 2-club found so far; this lower bound is initialized by the maximum degree plus one. Branching is aborted if the current graph has less than ℓ' vertices. After exploring all branches, we output the current lower bound (along with a 2-club of this size).

The above search tree strategy was introduced by Bourjolly et al. [7] and has been already experimentally evaluated [8]. By simple recursion analysis the running time can be bounded by $O^*(\alpha^n)$ where α is the

golden ratio with $\alpha \approx 1.62$ [8]. Schäfer et al. [22] showed that for the dual size parameter k' this search tree strategy runs in $O^*(2^{k'})$ time (if branching is aborted if more than k' vertices have been removed). Note that by Corollary 1, the search tree size measured by k' cannot be improved unless the SETH fails.

Turing kernelization. Before starting the search tree algorithm, we use the Turing kernelization introduced by Schäfer et al. [22]. That is, we compute the Turing kernels consisting of $N_2[v]$ for all vertices v of the input graph.[3] We say $N_2[v]$ is the Turing kernel for vertex v. Then, as long as at least one Turing kernel is left, we apply the search tree algorithm to the smallest one, say the one for vertex v, to find the largest 2-club in the Turing kernel of v that contains v. Afterwards, we delete v in all other Turing kernels. The maximum 2-club found during this iteration is the output. Note that this is indeed equivalent to what the search tree algorithm does: In one case v is contained in the maximum 2-club S and thus $S \subseteq N_2[v]$, in the other case v is not contained and can thus be deleted. This observation explains the effectiveness of the search tree algorithms on the considered real-world data from social network analysis: There, the smallest two-neighborhood in the graph is typically much smaller than the entire vertex set, ensuring that all except n nodes in the search tree have limited size.

Heuristic Speed-up. Our main tool for accelerating the search tree algorithm is an extensive application of the following data reduction rules in each branching step. We describe the rules in descending order of observed effectiveness. Herein, let $G = (V, E)$ be the graph of the current branching step.

I1 *Vertex Cover Rule:* Let $G' = (V, E')$ be the graph where two vertices are adjacent iff they have distance at least three in G. Clearly, if a minimum vertex cover of G' has size at least x, then at least x vertex deletions have to be performed in G to obtain a 2-club. We compute a 2-approximate vertex cover C for G' that is disjoint to the marked vertices (as they may not be deleted). If $|V| - \lceil |C|/2 \rceil$ is less than the current lower bound, then abort this branch.

I2 *Cleaning conflicts with marked vertices:* If there is a vertex $v \in V$ that has distance at least three to a vertex that is marked to be contained in the 2-club, then delete v. If v is marked, then abort this branch.

I3 *Common neighbors of marked vertices:* If there are two marked vertices with only one common neighbor v, then mark v.

[3] After applying Rule 4 in advance, $|N_2[v]| \leq \ell^2$.

Table 1. Experimental results on random instances. For each combination of density, n, a, b the other values are the average over 50 instances.

density	[a ; b]	n	m	max deg	avg deg	2-club size	time(s)
0.05	[0.00 ; 0.10]	160	633.60	18.30	7.40	19.30	0.18
0.10	[0.05 ; 0.15]	160	1279.54	28.54	15.46	29.54	2.43
	[0.00 ; 0.20]	160	1276.40	31.68	15.44	33.08	2.79
0.15	[0.10 ; 0.20]	150	1674.28	35.72	21.84	56.98	98.44
	[0.10 ; 0.20]	160	1899.78	38.15	23.26	63.44	372.52
	[0.05 ; 0.25]	160	1906.68	41.04	23.26	77.12	21.54
	[0.00 ; 0.15]	160	1894.08	44.52	23.20	88.60	1.80
0.20	[0.10 ; 0.30]	160	2544.66	50.28	31.36	143.16	0.04

I4 *Degree-one vertices:* Remove each vertex v that has degree one. If v is marked, then abort this branch.

The correctness of Rules I1–I3 is obvious. Rule I4 is correct since we initialized our lower bound by a 2-club formed by a maximum degree vertex and thus a larger 2-club cannot contain degree-one vertices (note that Rule I4 is a special case of Rule 5).

We ran all our experiments on an Intel(R) Core(TM) i3-2130 CPU 3.40GHz machine with 8GB memory under the Debian GNU/Linux 6.0 operating system. The program is implemented in Java and runs under Java 1.6.0.18. The source code is freely available from `http://fpt.akt.tu-berlin.de/two_club/`. We tested our program on random instances as well as on real world data from the 10th DIMACS challenge [10] and compare our running times with recent implementations [8, 17].

Random Instances. As in previous experimental evaluations [8, 17], we use the random graph generator proposed by Gendreau et al. [13] where the density of the resulting graphs is controlled by two parameters, $0 \leq a \leq b \leq 1$, and the expected density is $(a + b)/2$. Table 5 summarizes our findings (see Table 3, appendix, for a full list). As first observed by Bourjolly et al. [7], density 0.15 produces the hardest instances. We solve instances of these types for $n = 150$ typically within 2min; previous implementations needed about 6min [8], or up to an hour [17] for these instances. We observed that the key point for the good behavior of our algorithm on these instances is the *Vertex Cover Rule* that allows quite frequently to prune the search tree.

Real-world Networks. We considered real-world data taken from the 2012 DIMACS challenge [10]. To investigate the usefulness of 2-Club as

natural clique relaxation concept, we ran our algorithm on instances from the *clustering* category; to test our algorithm on large scale social network graphs we ran it on graphs from the *co-author and citation* category. These graphs were obtained by the co-author relationship or the citation relation among authors listed in the DBLP and Citeseer database. In addition to the DIMACS instances, we created a further DBLP coauthor graph, which is the largest instance in our experiments (dblp_thres_1). Table 5 shows the results (see Table 4, appendix, for a full list).

We observe that, since the average degree in real world graphs is small, the Turing kernelization typically produces small graphs for our search tree algorithm. We thus can solve all instances from the clustering category within 10s. This is a significant performance increase in comparison to [17] who needed up to 70min for these instances. Moreover, although the co-author/citation graphs are quite large (up to 715,000 vertices), Turing kernelization enabled us to handle them within roughly 30min.

We observed, however, the unexpected behavior that the largest 2-club is on a majority of the real-world instances "just" a maximum degree vertex together with its neighbors. Thus, the question arises whether the resulting community structures are meaningful. In a first step to examine this, we created from a DBLP coauthor graph subgraphs of the pattern dblp_thres_i where two authors are related by an edge if they coauthored *at least i papers*. We expected that for moderate values of i, say 2 or 3, the resulting (2-club) communities would have a stronger meaning because there are no edges between authors that are only loosely related. Unfortunately, even for values up to $i = 6$ this seems not to be the case. We think the main reason for this is the large gap between the maximum degree vertex (around 1000) and the average degree (less than 10). Thus, there seem to be some authors that dominate the overall structure because of their large number of coauthors. Notably, there are only few of these "dominating" authors: less than 200 authors have more than 200 coauthors.[4]

6 Conclusion

On the theoretical side, we extended existing fixed-parameter tractability results for the 2-Club problem by providing polynomial-size kernels for the parameters cluster editing set size and feedback edge set size.

[4] This implies that the so-called h-index of the real-world instances is low and thus a promising parameter. In companion work, however, we showed that 2-Club is W[1]-hard with respect to the h-index of the input graph [14].

Table 2. Experimental results on instance from the DIMACS implementation [10] taken from the clustering and the coauthor/citation category.

category	name	n	m	max deg	avg deg	2-club size	time(s)
clustering	email	1133	5451	71	9	72	3.27
	hep-th	8361	15751	50	3	51	4.18
	PGPgiantcompo	10680	24316	205	4	206	3.22
	polblogs	1490	16715	351	22	352	9.93
	power	4941	6594	19	2	20	2.53
coauthor	citationCiteseer	268495	1156647	1318	8	1319	429.83
	coAuthorsCiteseer	227320	814134	1372	7	1373	23.04
	coAuthorsDBLP	299067	977676	336	6	337	216.64
	dblp_thres_01	715633	2511988	804	7	805	1742.57
	dblp_thres_02	282831	640697	201	4	202	119.03

We further gave a direct algorithm for the parameter treewidth of G. Complementing these positive results, we showed lower bounds on the kernel size for parameter vertex cover and on the running time as well as on the kernel size for the dual parameter k'. On the practical side, we provide the currently best implementation for 2-Club which solves 2-Club in reasonable time even on large real-world graphs with more than 700,000 vertices.

Still, there are many open questions that deserve further investigations: Is there a substantially better algorithm for the parameter vertex cover than the one for treewidth? Concerning the parameter solution size ℓ, can the, so far impractical, running time or the size of the Turing kernel be improved [22]? Are there stronger parameters than the ones considered here for which 2-Club admits polynomial-size problem kernels? Finally, it would be interesting to transfer our results to 3-Club which is also of interest in practice [17, 21].

Bibliography

[1] R. Alba. A graph-theoretic definition of a sociometric clique. *J. Math. Sociol.*, 3 (1):113–126, 1973.

[2] Y. Asahiro, E. Miyano, and K. Samizo. Approximating maximum diameter-bounded subgraphs. In *Proc. 9th LATIN*, volume 6034 of *LNCS*, pages 615–626. Springer, 2010.

[3] B. Balasundaram, S. Butenko, and S. Trukhanovzu. Novel approaches for analyzing biological networks. *J. Comb. Optim.*, 10(1):23–39, 2005.

[4] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. In *Proc. 17th ESA*, volume 5757 of *LNCS*, pages 635–646. Springer, 2009.

[5] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Cross-composition: A new technique for kernelization lower bounds. In *Proc. 28th STACS*, volume 9 of *LIPIcs*, pages 165–176. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[6] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011.

[7] J.-M. Bourjolly, G. Laporte, and G. Pesant. An exact algorithm for the maximum k-club problem in an undirected graph. *European J. Oper. Res.*, 138(1):21–28, 2002.

[8] M. S. Chang, L. J. Hung, C. R. Lin, and P. C. Su. Finding large k-clubs in undirected graphs. In *Proc. 28th Workshop on Combinatorial Mathematics and Computation Theory*, 2011.

[9] Y. Chen, J. Flum, and M. Müller. Lower bounds for kernelizations and other preprocessing procedures. *Theory Comput. Syst.*, 48(4):803–839, 2011.

[10] DIMACS. Graph partitioning and graph clustering, 2012. URL `http://www.cc.gatech.edu/dimacs10/`. Accessed April 2012.

[11] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[12] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[13] M. Gendreau, P. Soriano, and L. Salvail. Solving the maximum clique problem using a tabu search approach. *Ann. Oper. Res.*, 41(4):385–403, 1993.

[14] S. Hartung, C. Komusiewicz, and A. Nichterlein. On structural parameterizations for the 2-club problem, 2012. Manuscript, June 2012.

[15] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. System Sci.*, 63(4):512–530, 2001.

[16] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

[17] F. Mahdavi and B. Balasundaram. On inclusionwise maximal and maximum cardinality k-clubs in graphs. *Discrete Optim.*, 2012. To appear.

[18] N. Memon and H. L. Larsen. Structural analysis and mathematical methods for destabilizing terrorist networks using investigative data mining. In *Proc. 2nd ADMA*, volume 4093 of *LNCS*, pages 1037–1048. Springer, 2006.

[19] R. J. Mokken. Cliques, Clubs and Clans. *Quality and Quantity*, 13:161–173, 1979.

[20] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[21] S. Pasupuleti. Detection of protein complexes in protein interaction networks using n-Clubs. In *Proc. 6th EvoBIO*, volume 4973 of *LNCS*, pages 153–164. Springer, 2008.

[22] A. Schäfer, C. Komusiewicz, H. Moser, and R. Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optim. Lett.*, 6(5), 2012.

[23] A. Schäfer. Exact algorithms for s-club finding and related problems. Diploma thesis, Friedrich-Schiller-Universität Jena, 2009.

[24] S. B. Seidman and B. L. Foster. A graph-theoretic generalization of the clique concept. *J. Math. Sociol.*, 6:139–154, 1978.

A Appendix

A.1 Full experimental results

density	[a ; b]	n	m	FES	max deg	avg deg	h-index	size	time(s)
0.05	[0.00 ; 0.10]	140	478.52	339.52	16.74	6.42	11.58	17.74	0.11
	[0.00 ; 0.10]	150	580.24	431.24	17.78	7.28	12.72	18.78	0.15
	[0.00 ; 0.10]	160	633.60	474.60	18.30	7.40	13.14	19.30	0.18
0.10	[0.05 ; 0.15]	140	965.78	826.78	24.98	13.32	18.28	25.98	0.73
	[0.05 ; 0.15]	150	1116.96	967.96	27.18	14.38	19.68	28.22	1.23
	[0.05 ; 0.15]	160	1279.54	1120.54	28.54	15.46	21.02	29.54	2.43
	[0.00 ; 0.20]	140	978.24	839.24	28.06	13.46	20.00	29.36	0.78
	[0.00 ; 0.20]	150	1117.20	968.20	29.52	14.36	21.34	30.64	1.41
	[0.00 ; 0.20]	160	1276.40	1117.40	31.68	15.44	22.76	33.08	2.79
0.15	[0.10 ; 0.20]	140	1460.28	1321.28	33.18	20.34	25.10	50.60	37.10
	[0.10 ; 0.20]	150	1674.28	1525.28	35.72	21.84	26.74	56.98	98.44
	[0.10 ; 0.20]	160	1899.78	1740.78	38.15	23.26	28.26	63.44	372.52
	[0.05 ; 0.25]	150	1661.28	1512.28	38.28	21.68	27.48	66.08	18.22
	[0.05 ; 0.25]	160	1906.68	1747.68	41.04	23.26	29.64	77.12	21.54
	[0.00 ; 0.15]	150	1686.52	1537.52	41.80	21.96	29.68	82.68	0.95
	[0.00 ; 0.15]	160	1894.08	1735.08	44.52	23.20	31.20	88.60	1.80
0.20	[0.15 ; 0.25]	140	1949.50	1810.50	41.84	27.34	31.66	120.98	0.04
	[0.15 ; 0.25]	150	2235.12	2086.12	44.58	29.34	33.84	133.50	0.04
	[0.15 ; 0.25]	160	2547.14	2388.14	46.78	31.30	35.96	147.26	0.03
	[0.10 ; 0.30]	140	1944.38	1805.38	43.70	27.28	32.30	118.32	0.05
	[0.10 ; 0.30]	150	2235.98	2086.98	46.54	29.26	34.66	130.40	0.05
	[0.10 ; 0.30]	160	2544.66	2385.66	50.28	31.36	37.00	143.16	0.04
	[0.05 ; 0.35]	140	1937.54	1798.54	46.98	27.12	33.48	116.40	0.02
	[0.05 ; 0.35]	150	2236.68	2087.68	50.52	29.38	35.94	127.94	0.02
	[0.05 ; 0.35]	160	2529.82	2370.82	53.40	31.12	38.04	139.18	0.03

Table 3. Experimental results on random instances. For each combination of density, a, b, and n (number of vertices) the other columns indicate the average values over 50 instances. Thereby, m denotes the number of edges, FES the feedback edge size, max deg the maximum degree, avg deg the average degree, and size the maximum size 2-club found by our algorithm.

category	name	n	m	FES	max deg	avg deg	h-index	2-club size	time(s)
clustering	adjnoun	112	425	314	49	7	13	50	0.10
	celegans_metabolic	453	2025	1573	237	8	23	238	0.40
	celegansneural	297	3520	3224	284	23	34	285	0.06
	dolphins	62	159	98	12	5	9	13	0.01
	email	1133	5451	4319	71	9	33	72	3.27
	football	115	613	499	12	10	12	16	0.12
	hep-th	8361	15751	7391	50	3	27	51	4.18
	jazz	198	2742	2545	100	27	41	103	0.13
	karate	34	78	45	17	4	6	18	0.00
	netscience	1589	2742	1154	34	3	19	35	1.52
	PGPgiantcompo	10680	24316	13637	205	4	52	206	3.22
	polblogs	1490	16715	15226	351	22	87	352	9.93
	polbooks	105	441	337	25	8	15	28	0.01
	power	4941	6594	1654	19	2	12	20	2.53
coauthor	citationCiteseer	268495	1156647	888153	1318	8	209	1319	429.83
	coAuthorsCiteseer	227320	814134	586815	1372	7	114	1373	23.04
	coAuthorsDBLP	299067	977676	678610	336	6	132	337	216.64
	dblp_thres_01	715633	2511988	1796356	804	7	208	805	1742.57
	dblp_thres_02	282831	640697	357867	201	4	96	202	119.03
	dblp_thres_03	167006	293796	126791	123	3	62	124	18.89
	dblp_thres_04	112949	168524	55576	88	2	46	89	7.70
	dblp_thres_05	81519	107831	26313	71	2	38	72	4.07
	dblp_thres_06	60989	73407	12419	57	2	31	58	2.25

Table 4. Experimental results on instances from the DIMACS implementation [10] taken from the clustering and the coauthor/citation category.

A.2 Proof 1 (Theorem 1)

Proof. Let $I = (G, \ell, D)$ be an instance that is reduced with respect to Rules 1, 2 & 3. Since I is reduced with respect to Rule 2, there are at most $2k$ cluster in $G - V(G)$. Each of them has size at most $\ell - 1$ since I is reduced with respect to Rule 1. Finally, $\ell \leq 2k + 1$ since I is reduced with respect to Rule 3. Altogether this implies that G contains at most $4k^2 + 2k$ vertices.

As to the running time, the clusters in G and $G - V(D)$ can be computed in $O(n + m)$ time. Having computed the clusters the data reduction rules can be applied exhaustively in linear time: First, checking the size of a cluster C_i or deleting arbitrary vertices in C_i is clearly doable in $O(|C_i|)$ time. Second, the reduction rules can be applied in one pass by applying Rule 1, then Rule 2, and then Rule 3 (note that Rule 3 does not trigger Rule 1 since it reduces l and $|C_i|$ to an equal extent). Hence, the problem kernel can be computed in $O(n + m)$ time. □

A.3 Proof 2 (Proof of correctness for Rule 5)

Proof. Let v be a vertex that does not lie on any feedback edge path. Then, v is not contained in a cycle in G. Suppose otherwise, and let C be a cycle that contains v. Clearly, C contains at least one feedback edge. In case C contains $\ell > 1$ feedback edges, then a cycle C' with $\ell - 1$ feedback edges also containing v is obtained by replacing an arbitrary feedback edge with a path consisting only of edges from $E \setminus F$ (by the minimality of F such a path must exist). In case C contains exactly one feedback edge, v lies on the feedback edge path of this edge, contradicting the initial assumption for v.

Let S be a 2-club containing v. Since v is not contained in any cycle in G, v has degree one in $G[S]$ or v is a cut vertex in $G[S]$. In the first case, S is completely contained in the neighborhood of v's neighbor in $G[S]$. In the second case, $S \subseteq N[v]$. Since G is reduced with respect to Rule 4, S thus has size less than ℓ. Consequently, there is no 2-club of size at least ℓ that contains v. Therefore, removing v from G yields an equivalent instance. □

A.4 Proof 3 (Proof of correctness for Rule 6)

Proof. We show that v is not contained in a 2-club of size at least ℓ. Let S be a 2-club containing v. Since v has in G distance at least three to at least one endpoint of every spanning feedback edge, S contains at most one endpoint of every spanning feedback edge. Consequently, either v has degree one in $G[S]$ or v is a cut vertex in $G[S]$. In the first case, S is completely contained in the neighborhood of v's neighbor in $G[S]$. In the second case, $S \subseteq N[v]$. Since G is reduced with respect to Rule 4, S has size less than ℓ. Consequently, there is no 2-club of size ℓ that contains v. Therefore, removing v from G yields an equivalent instance. □

A.5 Proof 4 (Theorem 2)

Proof. Let G be a graph that is reduced with respect to Rules 4, 5, and 6. The size of G can be bounded as follows. There are at most $2k$ vertices incident with feedback edges; let X denote this vertex set. To show the kernel size, we show that the set $Y := V \setminus X$ of vertices *not* incident with any edge in F has size at most $3k$. The main idea underlying our proof is that after the reduction rules have been exhaustively applied each vertex of Y is in T either adjacent to an endpoint of some feedback edge or

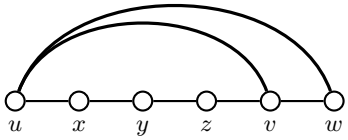

Fig. 1. Illustration of the definitions in the proof of Theorem 2. Herein, bold edges are feedback edges, regular edges are edges of T. The path $P_{\{u,v\}}$ connects the endpoints of the feedback edge $\{u, v\}$ in T. The feedback edge $\{u, w\}$ *covers* the feedback edge $\{u, v\}$; the feedback edge $\{u, v\}$ is u-minimal; the u-neighbor of $\{u, v\}$ is x, the v-neighbor is z; the vertices x, y and z are all *satisfied*.

has distance exactly two to both endpoints of a spanning feedback edge. Using this characterization, a size bound of $5k$ for Y is relatively easy to achieve. In the following, we use a more sophisticated approach to show that $|Y| \leq 3k$.

The main idea of the proof is to iteratively add feedback edges to T, and count the number of vertices for which there is a spanning feedback edge whose endpoints have distance at most two to this vertex. More precisely, let $T_0 := T$ and let $T_k := G$. Furthermore, let T_i be obtained from T_{i-1} by adding a feedback edge e with the following property: all feedback edges e' such that P_e is a subpath of $P_{e'}$ are already contained in T_{i-1}; in the following, we say that $P_{e'}$ *covers* P_e (see Figure 1 for an illustration). Note that finding such an edge e is always possible: If e is covered by an edge e' that is not contained in T_{i-1}, then we consider e'. Clearly the path $P_{e'}$ is longer than the path P_e. Therefore, there has to be some edge whose covering edges are already contained in T_{i-1}.

For each T_i we bound the number of vertices that have distance at most two to both endpoints of a spanning feedback edge; we call these vertices *satisfied*. To this end, we create a set $Y_i \subseteq \{(v, e) \mid v \in V, e \in F\}$. This set contains pairs of satisfied vertices and spanning feedback edges to which they are attributed. The central property for Y_i is that for each satisfied vertex in T_i the set Y_i contains at least one pair that contains this vertex. Then, the aim is to show that $Y_k \leq 3k$.

Before proving the claim we introduce some further terminology concerning the feedback edges that will be used in the proof. A feedback edge $\{u, v\}$ is called *u-minimal* in T_i if there is in T_i no feedback edge $\{u, w\}$ such that the path $P_{\{u,w\}}$ is a subpath of $P_{\{u,v\}}$. A vertex w is an u-neighbor of a feedback edge $\{u, v\}$ if $\{u, v\}$ spans w and w is in T a neighbor of u.

Now, by induction on i we show that for each i there is a set Y_i with the following properties:

– $|Y_i| \leq 3i$,
– each vertex satisfied in T_i is in at least one pair contained in Y_i, and
– if a feedback edge $\{u, v\}$ is u-minimal in T_i, then Y_i contains the pair $(w, \{u, v\})$ where w is the u-neighbor of $\{u, v\}$.

For T_1, the claim can be proven as follows. By definition, T_1 contains one feedback edge $\{u, v\}$. For each satisfied vertex w we add the pair containing $(w, \{u, v\})$ to Y_1. Then Y_1 fulfills the invariant since at most three vertices are satisfied in T_1: the u-neighbor of $\{u, v\}$, the v-neighbor of $\{u, v\}$ and at most one further inner vertex (this is the case when $P_{\{u,v\}}$ contains exactly five vertices).

For the inductive step, assume that the claim holds for $i - 1$, and let $\{u, v\}$ be the feedback edge that is added from T_{i-1} to T_i. We construct Y_i fulfilling the claim as follows. Initially, we set $Y_i := Y_{i-1}$. Then, for the u-neighbor and the v-neighbor of $\{u, v\}$, we add the pair containing $\{u, v\}$ and the respective vertex to Y_i, and, if it exists, a pair containing the vertex that is in T adjacent to both inner neighbors of $\{u, v\}$. After these additions, $|Y_i| \leq |Y_{i-1}| + 3$. We now show that for each further satisfied vertex that is not satisfied in T_{i-1} we can add a pair containing this vertex while also removing another pair from Y_i without violating the invariant.

All further vertices that are satisfied in T_i but not in T_{i-1} are neighbors of u or v since they must "use" $\{u, v\}$ in order to have distance two to some endpoint of a spanning edge. Since the neighbors of u and v that are in $P_{\{u,v\}}$ have already been added to Y_i, only satisfied vertices that are not in $P_{\{u,v\}}$ have to be considered. Let Z denote this set and partition Z into $Z_u := Z \cap N_T[u]$ and $Z_v := Z \cap N_T[v]$. Clearly, such a partition is possible since every vertex of Z is in T a neighbor of either u or v but not of both. Now consider the set Z_v, and let $z_1, \ldots, z_q$ denote the vertices in Z_v.

By definition, each z_j is adjacent to v and there must be a spanning feedback edge e such that in T_i but not in T_{i-1} the vertex z_j has distance at most two to both endpoints of e. Consequently, u is one of the endpoints of e, that is, $e = \{u, w_j\}$ for some w_j. Moreover, $\{u, w_j\}$ covers $\{u, v\}$ and furthermore, for all z_j's the w_j's are pairwise distinct. Thus, the situation is as depicted in Figure 2. We now show that for each z_j there is a pair $(x, \{u, p_j\})$ where x is the u-neighbor of $\{u, v\}$ in Y_i that can be removed from Y_i without violating the invariant. By the invariant, there is for each spanning edge $\{u, w_j\}$ a u-minimal edge e_j that is covered by $\{u, w_j\}$ and such that Y_{i-1} contains the pair (x, e_j). Since so far, Y_i is a superset of Y_{i-1} this pair is also contained in Y_i. Note that since the last

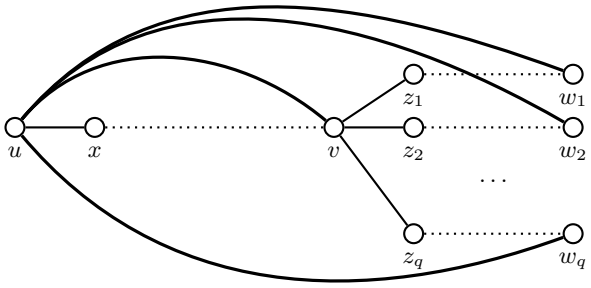

Fig. 2. Illustration of the vertices in Z_v and of the corresponding spanning edges.

edge that was added is $\{u, v\}$ and by the fact that T_i contains no edges that are covered by $\{u, v\}$ these pairs are pairwise distinct: they must cover $\{u, v\}$ and thus their endpoints p_j are on the paths from v to w_j; these paths are, with the exception of v, vertex-disjoint. Now each such pair can be removed, since after adding $\{u, v\}$ each edge $\{u, p_j\}$ is not u-minimal, and we have already added the pair $(x, \{u, v\})$. Consequently, when adding the $|Z_v|$ pairs for the satisfied vertices in Z_v to Y_i we can at the same time safely remove $|Z_v|$ pairs without violating the invariant.

Analogously, when adding the pairs for Z_u, we can also remove $|Z_v|$ pairs without violating the invariant. Summarizing, we have

$$|Y_i| \leq |Y_{i-1}| + 3 + |Z_v| - |Z_v| + |Z_u| - |Z_u| = 3 \cdot (i-1) + 3 = 3i.$$

The overall kernel bound now follows from $|Y| \leq |Z_k| \leq 3k$.

We complete the proof by bounding the running time. Clearly, Rule 4 can be exhaustively applied in linear time. The applicability of Rule 5 can be checked in $O(m \cdot n)$ time by considering each edge in F and labeling the vertices on its feedback edge path. After computing an all-pairs shortest path matrix in $O(n^3)$ time, the applicability of Rule 6 can be checked in $O(m \cdot n)$ time. The overall running time now follows from the fact that Rule 5 and Rule 6 can be applied at most n times.

□

A.6 Proof 5 (Theorem 3)

Proof. We give a polynomial time and parameter reduction [4] from Clique parameterized by vertex cover to 2-Club parameterized by vertex

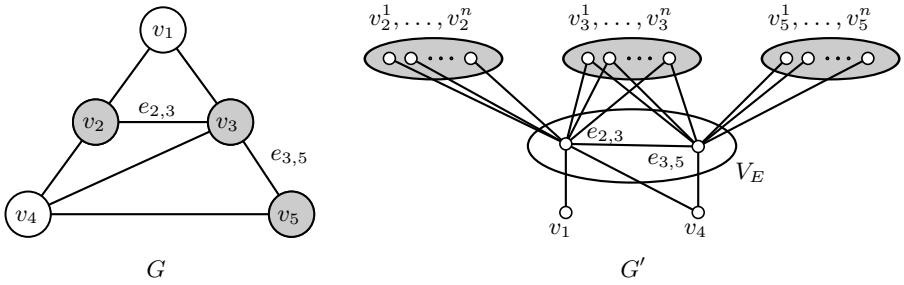

Fig. 3. Example of the construction in the proof of Theorem 3. The graph G is an input for Clique parameterized by vertex cover. The gray vertices are a vertex cover in G. The graph G' is constructed as described in the proof of Theorem 3.

cover. Unless NP $\subseteq$ coNP/poly, Clique does admit a polynomial kernel with respect to the size of a vertex cover [5]. Let $(G = (V, E), X, k)$ be an instance of Clique parameterized by the size of the vertex cover X. We construct a graph G' as follows (for an illustration see Figure 3, appendix). First, add for each vertex $v_i \in X$ exactly n vertices $\{v_i^1, \ldots, v_i^n\}$ to G'. The construction will ensure that these n vertices are twins in G'. Next, add a set V_E set of "edge-vertices" as follows. For each edge $\{v_i, v_j\} \in E$ between two vertex cover vertices $v_i, v_j \in X$ add an edge-vertex $e_{i,j}$ to G' and make $e_{i,j}$ adjacent to all vertices in $\{v_i^1, \ldots, v_i^n, v_j^1, \ldots, v_j^n\}$. Then, add edges such that V_E is a clique. The idea of the construction is to ensure that if a 2-club contains two vertices v_i^x and v_j^y, then v_i and v_j are adjacent in G'. Hence, a 2-club containing such vertices corresponds to a clique in G.

In order to handle the case where a size-k clique K in G contains a vertex from $V \setminus X$, add the vertex set $V \setminus X$ to G'. Then, for each added vertex $v \in V \setminus X$ and each edge-vertex $e_{i,j} \in V_E$ add an edge if v is adjacent to v_j *and* v_i in G. Observe that the construction runs in polynomial time, that it ensures that V_E is a vertex cover for G', and that $|V_E| \leq \binom{|X|}{2}$. Thus, it is a polynomial time and parameter reduction. We complete the proof by showing that G has a clique of size $k \Leftrightarrow G'$ has a 2-club of size at least $|V_E| + (k-1)n + 1$.

"$\Rightarrow$": Let K be a size-k clique in G. Then, every pair of vertices $v_i, v_j \in K \cap X$ has the common neighbor $e_{i,j}$ in G'. Hence, by Observation 1, the vertex set S containing the twins of all vertices in $K \cap X$ and V_E is a 2-club. In case $K \subseteq X$, this 2-club is of size $|V_E| + kn$. Otherwise, one can add the vertex $v \in K \setminus X$ to S. Then, S has size $|V_E| + (k-1)n + 1$ and it is also a 2-club: each vertex $v_i^x \in S \setminus V_E$ has with v the common

neighbor $e_{i,j}$ where v_j is some other vertex in K; similarly, each vertex in V_E has a common neighbor with v.

"$\Leftarrow$": Let S be a 2-club of size at least $|V_E| + (k-1)n + 1$ in G'. A *twin-free set* in S is a subset of $S \setminus (V_E \cup (V \setminus X))$ that does not contain twins. First, since $|V \setminus X| < n+1$, it follows that S contains a twin-free set of size at least $k-1$. Moreover, for each vertex pair $\{v_i, v_j\}$ in a twin-free set, the vertex $e_{i,j}$ has to be contained in S as well: otherwise v_i and v_j have distance greater than two in $G[S]$. Thus, v_i is adjacent to v_j in G. Therefore, a twin-free set in S corresponds to a clique in G. Hence, if there is a twin-free set of size at least k, then there is a size-k clique in G. It thus remains to consider the case where a largest twin-free set T is of size $k-1$. In this case, S contains at least one vertex $v \in V \setminus X$. Since S is a 2-club, there is for each vertex $v_i \in T$ at least one edge-vertex $e_{i,j} \in S$ that is adjacent to v_i and v. By construction, this implies that v_i is adjacent to u in G. Consequently, $T \cup \{v\}$ is a size-k clique in G. □

A.7 Proof 6 (Theorem 4)

Proof. Let $(G = (V, E), X, s, \ell)$ be an s-Club instance where X with $|X| = k$ is a vertex cover of G. First, branch into all possibilities to choose the subset $X' \subseteq X$ that is contained in the desired s-club. Then, remove $X \setminus X'$ and all vertices that are not within distance s to all vertices in X'. Clearly, $V \setminus X$ forms an independent set. Moreover, by Observation 1, it follows that two vertices $u, v \in V \setminus X$ that are twins with respect to $X'$are either both contained in a maximal s-club or none of them. Since there are at most $2^{|X'|} \leq 2^k$ different twins, branching into all possibilities to choose them into the s-club takes 2^{2^k} time. In total, s-Club can be solved in $2^k \cdot 2^{2^k} \cdot nm$ time. □

A.8 Proof 7 (Theorem 5)

Proof. Let (G, k) be an input instance of 2-Club and let $(\langle \{X_i \mid i \in I\} \rangle, E)$ be a nice tree decomposition for G of width ω. By Lemma 1 there is a vertex $v \in S$ such that $N[v] \cap S \subseteq X$ for a bag X. Thus, branch into the $O(n \cdot \omega \cdot 2^\omega)$ cases for choosing the bag X, the vertex v and its neighbors in X. Let $N_v := N[v] \cap S \subseteq X$. Then delete all vertices in $N[v] \setminus N_v$ and, afterwards, all vertices that do not have a neighbor in N_v. (This is correct, since N_v is a dominating set in the desired 2-club.) Set X to be the root of the resulting nice tree decomposition. Next, we describe a bottom-up dynamic programming algorithm on it.

Let X_i be an arbitrary bag and let $v_1^i, v_2^i, \ldots, v_{n_i}^i$, $|X_i| = n_i$, be the vertices in X_i. Furthermore, let $\{T_1^i, T_2^i, \ldots, T_{2^{n_i}}^i\}$ be the set of all possible subsets of X_i. We say a vertex $u \in V$ is of *type* T_j^i if $N(u) \cap X_i = T_j^i$. We store a table Tab_i for X_i that has $2^{n_i+2^{n_i}}$ rows and the rows contain all 0-1 sequences of length $n_i + 2^{n_i}$. Denoting by $r[j]$ the jth entry of row r, we define $V_i(r)$ to be the subset of all vertices in X_i whose entry in r is one, formally $V_i(r) := \{v_j^i \in X_i \mid r[j] = 1\}$, and let $T_i(r) := \{T_j^i \mid r[n_i + j] = 1\}$ be the set of types whose entry is one. Then, the integer value associated to r is the size of a largest vertex set $K_i(r)$ that fulfills the following properties:

1. $K_i(r)$ is a subset of vertices occurring in the subtree rooted by X_i with $X_i \cap K_i(r) = V_i(r)$,
2. each vertex $u \in K_i(r) \setminus V_i(r)$ is of one of the types in $T_i(r)$,
3. for each type in $T_j^i \in T_i(r)$ there is at least one vertex $u \in K_i(r) \setminus V_i(r)$ of type T_j^i, and
4. in $G[K_i(r)]$ each vertex $v \in K_i(r)$ has distance at most two to all vertices in $K_i(r) \setminus V_i(r)$.

If there is no such vertex set $K_i(r)$ then set $\mathrm{Tab}_i(r) = -\infty$. Furthermore, the row r has to be *consistent*, that is, for every $T_j^i \in T_i(r)$ it holds that $T_j^i \subseteq V_i(r)$. If r is not consistent, then again set $\mathrm{Tab}_i(r) = -\infty$. Once we computed these values in each bag, we look into the root X of the tree decomposition and then take the maximum values of the rows r with $N_v = V(r)$.

Computation of the table entries. We now specify how to compute the table Tab_i by distinguishing the following cases: Whether the bag X_i is a leaf, an introduce node, a join node, or a forget node. First initialize all table entries with $-\infty$. We only look at consistent rows in the following.

Leaf node: Let X_i be a leaf in the tree decomposition. Clearly, since $K_i(r)$ has to be a subset of vertices in the subtree of X_i, we only have to consider the case where $T_i(r) = \emptyset$. Then $K_i(r)$ is equal to $V_i(r)$ and, hence, we set $\mathrm{Tab}_i(r) = |V_i(r)|$.

Introduce node: Let X_i be an introduce node with the child node X_ℓ and let $u \in X_i \setminus X_\ell$ be the introduced vertex. First, consider rows r of Tab_i with $u \notin V_i(r)$. Then, clearly, we can take the value of the corresponding row r' in Tab_ℓ with $V_i(r) = V_\ell(r')$ and $T_i(r) = T_\ell(r')$. Second, consider the rows r in Tab_i with $u \in V_i(r)$. Since $u \notin X_\ell$, we can set $\mathrm{Tab}_i(r) = -\infty$ for all rows r containing a set $T_j^i \in T_i(r)$ with $u \in T_j^i$ or $N(u) \cap T_j^i \cap V_i(r) = \emptyset$ (property 4.). For the other rows r take one

plus the value of the corresponding row r' in Tab_ℓ with $V_i(r) = V_\ell(r')$ and $T_i(r) = T_\ell(r')$.

Next, consider the case that the bag X_i is a join node and X_ℓ and X_q are the two child nodes with $X_i = X_\ell = X_q$. For a row r we define $R_{\ell,q}(r)$ as the set of pairs of rows that "fit" to r:

$$\begin{aligned} R^{\mathrm{join}}_{\ell,q}(r) := \{(r', r'') \mid r' \in \{0,1\}^{n_\ell + 2^{n_\ell}}, r'' \in \{0,1\}^{n_q + 2^{n_q}} \wedge \\ V_i(r) = V_\ell(r') = V_q(r'') \wedge \\ \forall T^\ell_j \in T_\ell(r') \forall T^q_h \in T_q(r'') : T^\ell_j \cap T^q_h \cap V_i(r) \neq \emptyset \wedge \\ T_i(r) = T_\ell(r') \cup T_q(r'')\} \end{aligned}$$

Then, $\mathrm{Tab}_i(r) = \max_{(r',r'') \in R^{\mathrm{join}}_{\ell,q}(r)} \{\mathrm{Tab}_\ell(r') + \mathrm{Tab}_\ell(r'')\} - V_i(r)$. Note that the subtraction of $V_i(r)$ avoids double counting the vertices $V_i(r)$.

Forget node: Let X_i be a forget node with the child node X_ℓ and $u \in X_\ell \setminus X_i$. Then we have to consider the cases that $u \in K_i(r)$ and $u \notin K_i(r)$ for each row r in Tab_i. Again we define the "fitting" rows:

$$\begin{aligned} R^{\mathrm{del}}_\ell(r) := \{r' \mid r' \in \{0,1\}^{n_\ell + 2^{n_\ell}} \wedge \\ V_i(r) \cup \{u\} = V_\ell(r') \wedge \\ \forall T^\ell_j \in T_\ell(r') : (T^\ell_j \setminus \{u\} \in T_i(r)) \wedge \\ \forall T^i_j \in T_i(r) : (T^i_j \in T_\ell(r')) \vee (T^i_j \cup \{u\} \in T_\ell(r')) \vee (u \text{ is of type } T^i_j)\} \\ \cup \{r' \mid r' \in \{0,1\}^{n_\ell + 2^{n_\ell}} \wedge V_i(r) = V_\ell(r) \wedge T_i(r) = T_\ell(r)\} \end{aligned}$$

Observe that the first set in the definition of $R^{\mathrm{del}}_\ell(r)$ catches the cases where $u \in K_i(r)$ and the second set catches the case that $u \notin K_i(r)$. Then, $\mathrm{Tab}_i(r) = \max_{r' \in R^{\mathrm{del}}_\ell(r)} \{\mathrm{Tab}_\ell(r')\}$.

Running time. Clearly, the table Tab_i for a leaf X_i can be computed in $O(2^{O(2^\omega)})$ time.

The computation of the table for an introduce node X_i with child node X_ℓ can be done in $O(2^{2\cdot(\omega+2^\omega)} \cdot 2^{2\omega})$ time: For each of the at most $O(2^{\omega+2^\omega})$ rows in Tab_i we can compute $\mathrm{Tab}_i(r)$ in $O(2^{\omega+2^\omega} \cdot 2^{2\omega})$ time: For a row r where the introduced vertex u is not chosen, that is, $u \notin V_i(r)$ the lookup in the corresponding row in the child bag clearly can be done in the stated time. For a row r where u is chosen the check whether there is a set $T^i_j \in T_i(r)$ with $u \in T^i_j$ or $N(u) \cap T^i_j \cap V_i(r) = \emptyset$ and the lookup in the corresponding row in the child bag also can be done in the stated time.

The computation for a join node X_i with the two child nodes X_ℓ and X_q can be done in $O(2^{3\cdot(\omega+2^\omega)} \cdot 2^{3\omega})$: Simply check for each combination of rows $r \in \mathrm{Tab}_i, r' \in \mathrm{Tab}_\ell, r'' \in \mathrm{Tab}_q$ whether the pair (r', r'') fulfill the conditions to be contained in $R^{\mathrm{join}}_{\ell,q}(r)$ in $O(2^{3\omega})$ and then take the maximum corresponding value of all fitting pairs.

The computation for a forget node X_i with the child node X_ℓ can be done in $O(2^{2\cdot(\omega+2^\omega)} \cdot 2^{2\omega})$ time: Checking for each pair of rows $r \in \mathrm{Tab}_i, r' \in \mathrm{Tab}_\ell$ whether r' fits to r and then take the maximum of all fitting rows. The checking whether r' fits to r can be done in at most $O(2^{2\omega})$ time.

Together with the first step where we have guessed the bag X and N_v, the total running time can be bound by $2^{O(2^\omega)} \cdot n^2$.

Correctness. Since our root X contains the vertex set N_v and N_v is the neighborhood of the vertex v guessed to be contained the desired maximum 2-club S, one can easily verify that S contains $\max_r\{\mathrm{Tab}(r) \mid V(r) = N_v\}$ vertices. (Because of property 4. and since v dominates N_v, each set $K_i(r)$ for a row r with $V(r) = N_v$ is a 2-club.)

It remains to show that the properties 1 to 4 are satisfied in the table of each bag in the tree decomposition. Observe, that the properties are fulfilled for the leafs of the tree decomposition: Since in this case $K_i(r) = V_i(r)$ there are no vertices in $K_i(r) \setminus V_i(r)$.

Next, consider an introduce node X_i with child node X_ℓ, introduced vertex $\{u\} = X_i \setminus X_\ell$, and the table of X_ℓ satisfy the properties 1 to 4. Let r be some row in the table Tab_i. If $u \notin V_i(r)$ the largest vertex set $K_i(r)$ fulfilling the properties 1 to 4 is clearly the set $K_\ell(r')$ with $V_i(r) = V_\ell(r')$ and $T_i(r) = T_\ell(r')$. Thus, the corresponding table entry is $\mathrm{Tab}_i(r) = \mathrm{Tab}_\ell(r')$. Now consider the other case when $u \in V_i(r)$. Let $K_i(r)$ be the largest vertex set fulfilling the properties. Since property 4 is satisfied u has distance at most two to every vertex in $K_i(r) \setminus V_i(r)$. Since $u \notin X_\ell$ this implies for every vertex $w \in K_i(r) \setminus V_i(r)$ that $N(u) \cap N(w) \subseteq V_i(r)$. Thus, every set $T^i_j \in T_i(r)$ has to fulfill $u \notin T^i_j$ and $N(u) \cap T^i_j \cap V_i(r) \neq \emptyset$, otherwise $\mathrm{Tab}_i(r)$ is set to $-\infty$. Assuming that for all sets $T^i_j \in T_i(r)$ it holds that $u \notin T_i(r)$ and $N(u) \cap T^i_j \cap V_i(r) \neq \emptyset$, it is clear that $K_i(r)$ is $K_\ell(r') \cup \{u\}$ where $V_i(r) = V_\ell(r') \cup \{u\}$ and $T_i(r) = T_\ell(r')$. Thus, the computed table Tab_i fulfills the properties.

Next, consider a join node X_i with the two child nodes X_ℓ and X_q, $X_i = X_\ell = X_q$. First observe that for any two vertex sets $K_\ell(r')$ and $K_q(r'')$ fulfilling the properties 1 to 4 it holds that $K_\ell(r') \cap K_q(r'') \subseteq X_i$. Thus, if for a row r the set $K_i(r)$ contains a vertex u that appears somewhere in the subtree rooted by X_ℓ and a vertex w that appears somewhere in the subtree

rooted by X_q and $u, w \notin X_i$, then it is clear that u and w are connected by a vertex in X_i, that is $N(u) \cap N(w) \subseteq X_i$ and $N(u) \cap N(w) \neq \emptyset$. Hence there are two rows r' and r'' such that $K_i(r) = K_\ell(r') \cup K_q(r'')$, it holds that $V_i(r) = V_\ell(r') = V_q(r'')$, for each pair $u \in K_\ell(r') \setminus V_\ell(r)$, $w \in K_q(r'') \setminus V_i(r'')$ there is a vertex $x \in V_i(r)$ with $x \in N(u) \cap N(w)$, and $T_i(r) = T_\ell(r') \cup T_q(r'')$. Note that $(r', r'') \in R^{\text{join}}_{\ell,q}(r)$ and, thus, $\text{Tab}_i(r)$ is indeed the size of a maximum vertex set fulfilling properties 1 to 4.

Finally, consider a forget node X_i with the child node X_ℓ and $\{u\} = X_\ell \setminus X_i$. There are two cases for the set $K_i(r)$: $u \in K_i(r)$ and $u \notin K_i(r)$. First consider the case $u \in K_i(r)$. Then there is a row r' such that $K_i(r) = K_\ell(r')$ and $V_i(r) \cup \{u\} = V_\ell(r')$. Note that the sets $T^i(r)$ and $T^\ell(r')$ have to be somewhat "consistent": The vertex u has to be of some type $T^i_j \in T_i(r)$ because $u \in K_i(r) \setminus V_i(r)$ and for all other sets $T^i_{j'} \in T_i(r)$, $T^i_j \neq T^i_{j'}$, there has to be a vertex in $K_i(r) \setminus (V_i(r) \cup \{u\})$ being of type $T^i_{j'}$. Since $u \notin X_i$, this implies that $T^i_{j'} \in T_\ell(r')$ or $T^i_{j'} \cup \{u\} \in T_\ell(r')$. Reversely, each set $T^\ell_j \in T_\ell(r')$ need some corresponding set in $T_i(r)$, that is, if $u \in T^\ell_j$, then $(T^\ell_j \setminus \{u\}) \in T_i(r)$, otherwise $T^\ell_i \in T_i(r)$. Note that by definition of $R^{\text{del}}_\ell(r)$ this implies $r' \in R^{\text{del}}_\ell(r)$. Hence, the computed entry $\text{Tab}_i(r)$ fulfill the properties 1 to 4. □

A.9 Proof 8 (Lemma 2)

Proof. We give a parameterized reduction from Cnf-Sat parameterized by the number n of variables to s-Club parameterized by the dual parameter k'. Let $\mathcal{F}$ be a formula in conjunctive normal form with n variables. Assume without loss of generality that $\mathcal{F}$ does not contain a clause that contains the positive and the negative literal of the same variable. In the following, we construct an n'-vertex graph G such that for $k' := n$ the graph G has an s-club of size at least $n' - k'$ if and only if $\mathcal{F}$ has a satisfying assignment.

First, add the *literal vertex set* V to G, that is, a set that contains for each variable x in $\mathcal{F}$ two vertices, one corresponding to the positive literal x and the other corresponding to $\neg x$. Next, add for each clause in $\mathcal{F}$ two vertices, one called the *left clause vertex* and the second called the *right clause vertex*, and make them adjacent to all literal vertices that correspond to the literals occurring in the clause. Denote the set of clause vertices by K. We will ensure that G has the three following properties:

1. The distance between the positive literal vertex and the negative literal vertex of a variable is at least $s+1$. All other literal vertices have pairwise distance at most s.
2. For each clause, every path from the left clause vertex v to the right clause vertex w that does not contain any literal vertex from $N(v) \cap N(w)$ has length at least $s+1$. All other clause vertices have pairwise distance at most s.
3. Each literal vertex has a path to each clause vertex that is of length at most s and does not contain any other literal vertex.

From the properties above it is easy to prove the correctness of the reduction: Suppose that by deleting at most k' vertices, G can be transformed into an s-club. Since the positive and negative literal vertex of each variable have distance at least $s+1$ (property 1), at least one of them has to be deleted. This forces $n = k'$ deletions, one for each variable. Consequently, only literal vertices are deleted and the remaining literal vertices correspond to an assignment. Because of property 2, for each left and right vertex of a clause there has to be at least one remaining literal vertex that is a common neighbor. Thus, the remaining literal vertices correspond to a *satisfying* assignment for $\mathcal{F}$. In the other direction, because of property 3, it is straightforward to argue that deleting all literal vertices that do not correspond to a satisfying assignment results in an s-club.

In the following, we will extend our graph G such that properties 1–3 are fulfilled. Moreover, we ensure that the vertices that we add for this purpose have distance at most s to all other vertices, no matter which of the literal vertices will be deleted. Let A be the set containing all vertex pairs of $V \cup K$ except the pairs $\{v_1, v_2\}$ where either v_1 and v_2 refer to the same clause (left and right clause vertex) or to the same variable (positive and negative literal vertex). For each vertex pair $\{v, v'\}$ of A add a new vertex $\ell_{\{v,v'\}}$ to the graph. Denote this newly added vertex set by C. Next, connect each vertex $\ell_{(\{v,v'\}} \in C$ with v and v' by a path of length $\lfloor s/2 \rfloor$ each. In case $2 \leq s \leq 3$, we simply connect the corresponding vertices by an edge. Denote the vertices on the paths without the end points by P. Thus, $V(G) = C \cup P \cup V \cup K$.

To complete the construction, we make a case distinction between odd and even s. If s is odd, then we complete the construction by adding a distinguished vertex x to C that is adjacent to all vertices in C. If s is even, then we make C a clique.

We now prove for the two cases of odd and even s that G fulfills properties 1-3: Clearly, for each vertex pair $v, w \in A$ there is a vertex in $\ell_{\{v,w\}} \in C$ such that v and w have distance $\lfloor s/2 \rfloor$ to $\ell_{\{v,w\}}$, implying

property 3. Moreover, observe that for the positive and negative literal of one variable there is no vertex in C that is reachable within distance $\lfloor s/2 \rfloor$ from both vertices and hence their distance is $s+1$ (property 1). The same argument holds for the left and right vertex of a clause: All paths that do not contain any literal vertex from the common neighborhood contain at least two vertices from C and thus have length at least $s+1$ (property 2).

To complete the proof we show in the following case distinction that the vertices in $P \cup C$ have distance at most s to all other vertices:

Case 1: $s \geq 3$ is odd. Let $v^C \in C$. First, observe that because of vertex x, C is a 2-club. Moreover, each vertex in $P \cup V \cup K$ has distance at most $\lfloor s/2 \rfloor$ to at least one vertex in C and thus v^C has distance at most $2 + \lfloor s/2 \rfloor \leq s$ to each vertex in $V(G)$. Consider a vertex $v^P \in P$. Clearly, v_p has distance at most $\lfloor s/2 \rfloor - 1$ to one vertex in C and thus can reach any other vertex in P via x by a path of length at most $2 + 2(\lfloor s/2 \rfloor - 1) = s - 1$. Note that from this, since each vertex in $V \cup K$ has at least one neighbor in P, it also follows that v^P has distance at most s to each vertex in $V \cup K$.

Case 2: $s \geq 2$ is even. Let $v^C \in C$. Since C is a clique and any vertex in $P \cup V \cup K$ has distance at most $s/2$ to at least one vertex in C, v^C has distance at most $s/2 + 1 \leq s$ to all vertices in $V(G)$. Now, let $v^P \in P$. Clearly, v^P has distance at most $s/2 - 1$ to one vertex in C and thus distance at most $s-1$ to all other vertices in P. Since any vertex in $V \cup K$ has at least one neighbor in P, it thus follows that v^P has distance at most s to each vertex in $V \cup K$. □