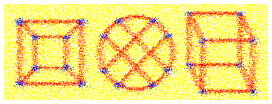
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 0, no. 0, pp. 0–0 (0)
DOI:

Parameterized Algorithmics and Computational Experiments for Finding 2-Clubs

Sepp Hartung[1] *Christian Komusiewicz*[1] *André Nichterlein*[1]

[1]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany

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-size problem 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-size problem kernel with respect to the *size of a vertex cover of* G. Next, we show that, under the strong exponential time hypothesis, a previous $\mathcal{O}(2^{|V|-\ell} \cdot |V||E|)$-time search tree algorithm [Schäfer et al., Optim. Lett. 2012] cannot be improved and that, unless NP $\subseteq$ coNP/poly, there is no polynomial-size problem 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 outperforms previous implementations.

Submitted:	Reviewed:	Revised:	Reviewed:	Revised:
July 2013	-	-	-	-
	Accepted:	Final:	Published:	
	-	-	-	
	Article type: Regular paper		Communicated by: A. Editor	

Research supported by Grant xxxx-xxxx-xxxx

E-mail addresses: sepp.hartung@tu-berlin.de (Sepp Hartung) christian.komusiewicz@tu-berlin.de (Christian Komusiewicz) andre.nichterlein@tu-berlin.de (André Nichterlein)

1 Introduction

Finding cohesive subnetworks 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. This requirement is often too restrictive and thus relaxed versions such as s-cliques [2], s-plexes [38], and s-clubs [33] 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 the subgraph induced by S has diameter at most s?

Clearly, 1-Club is equivalent to the well-known Clique problem where one has to find a clique of size ℓ. 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 [5, 35]; further applications of 2-Club arise in the analysis of social networks [32]. Consequently, experimental evaluations concentrate on finding 2-clubs and 3-clubs [3, 5, 13, 14, 15, 23, 31]. From a theoretical viewpoint it is also interesting that, in contrast to "being a clique", the graph property "being an s-club" for $s \geq 2$ is not *hereditary*. That is, it is not closed under vertex deletion. This seems to be a keypoint in causing the differences in the computational complexity of the corresponding problems.

1.1 Related Work

For all $s \geq 1$, s-Club is NP-complete even on graphs of diameter $s + 1$ [5]; 2-Club is NP-complete even on split graphs and, thus, also on chordal graphs [4]. 2-Club can be solved in polynomial time on bipartite graphs, on trees, and on interval graphs [37]. For all $s \geq 1$, s-Club can be solved in polynomial time on chordal bipartite, strongly chordal, and distance-hereditary graphs [22]. On a superclass of these graph classes, called weakly chordal graphs, it is NP-hard for even s and can be solved in polynomial time for odd s [22].

For all $s \geq 1$, s-Club is NP-hard to approximate within a factor of $n^{1/2-\epsilon}$ [4]; 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 $s \geq 3$ [4]. Several heuristics [12, 14, 15] and integer linear programming (ILP) formulations [3, 5, 13] for s-Club have been proposed and experimentally evaluated [3, 5, 13, 14, 15, 23, 31].

The 1-Club problem 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 [36, 37]. For the *dual parameter* $n - \ell$, the following simple search tree algorithm [36, 37] solves s-Club in $\mathcal{O}(2^{n-\ell} \cdot nm)$ time: As long as there is a

vertex pair whose distance is at least $s+1$, branch into the cases to either delete the first or the second vertex.[1] The same search tree algorithm can be analyzed to run in $\mathcal{O}(\alpha^n \cdot nm)$ time where α is the golden ratio $\alpha < 1.62$ [15]. s-Club can be formulated in monadic second order logic. Thus, it is fixed-parameter tractable with respect to the treewidth of G [37]. 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 ℓ^2 vertices for even s and at most ℓ^3 vertices for odd s [36].

In companion work [24] (see [25] for an extended version) we provided a systematic study of 2-Club with respect to a hierarchy of well-known structural graph parameters. For example, 2-Club remains NP-hard on 6-degenerate graphs, on graphs with a dominating set of size two, and on graphs that become bipartite by deleting only one vertex. On the positive side, the problem becomes fixed-parameter tractable when parameterized by the number of vertices that have to be deleted to transform the input graph into a cograph. On the negative side, it is W[1]-hard with respect to the h-index[2] of the graph but there is a so-called XP-algorithm, that is, an algorithm running in polynomial time for constant parameter values.

1.2 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 $\mathcal{O}(k^2)$-vertex kernel for the parameter *size of a cluster editing set* and an $\mathcal{O}(k)$-size kernel for the parameter *feedback edge set size*. The kernelization results for these rather large parameters are motivated by our negative results: 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 $2^{\mathcal{O}(2^\omega)}n^2$ time 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 5 in Section 3).

In Section 4, we prove that unless the *Strong Exponential Time Hypothesis* (SETH) fails, s-Club cannot be solved in $\mathcal{O}((2-\epsilon)^{n-\ell} \cdot |G|^{\mathcal{O}(1)})$ time for all $\epsilon > 0$. This is evidence that the above-mentioned search tree algorithm [36] is optimal with respect to the parameter $n-\ell$. To prove this, we give a reduction from Cnf-Sat to s-Club where the value of $n-\ell$ in the resulting s-Club instance is equal to the number of variables in the Cnf-Sat-instance. The reduction also implies that s-Club does not admit a polynomial kernel with respect to $n-\ell$, unless NP $\subseteq$ coNP/poly.

[1] Schäfer et al. [36] 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.

[2] The largest number k such that the graph has at least k vertices of degree at least k.

Having explored the theoretical limits of fixed-parameter algorithms for the dual parameter $n - \ell$, we then examine its usefulness for solving 2-Club in practice (in Section 5). 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 explore the effectiveness of our algorithm on random as well as on large-scale real world graphs and 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".

1.3 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 $\mathrm{dist}_G(u, v)$ to denote the *distance between* u *and* v in G, that is, the length of a shortest path between u and v; we omit the subscript if the graph is clear from the context. For a vertex $v \in V$ and an integer $t \geq 1$, denote by $N_t(v) := \{u \in V \setminus \{v\} \mid \mathrm{dist}(u, v) \leq t\}$ 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 $N(v) := N_1(v)$. Finally, let $N(V') := \bigcup_{v \in V'} N(v) \setminus V'$ and $N[V'] := N(V') \cup V'$ denote the open and closed neighborhood of a vertex set $V' \subseteq V$. A *cut-vertex* of a graph is a vertex whose deletion increases the number of connected components by at least one.

A *vertex cover* of a graph $G = (V, E)$ is a vertex subset $V' \subseteq V$ such that $\forall \{v, u\} \in E : \{v, u\} \cap V' \neq \emptyset$. A *feedback edge set* F of a graph is an edge set whose deletion leads to a forest. A *cluster editing set* of G is a set of edge additions and deletions that transforms G into a vertex-disjoint union of cliques. Formally, for a graph $G = (V, E)$ an edge set $D \subseteq V^2$ is a *cluster editing set* for G if $G^D := (V, (E \setminus D) \cup (D \setminus E))$ is a cluster graph, that is, a graph in which every connected component is a clique (called *cluster*).

Two vertices v and w are *twins* if $N(v) \setminus \{w\} = N(w) \setminus \{v\}$. Note that u and v may be non-adjacent. 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.*

Parameterized Algorithmics. We provide a short introduction to parameterized algorithmics. A problem is *fixed-parameter tractable* (FPT) with respect to a parameter k if there is a computable function f such that any instance (I, k) can be solved in $f(k) \cdot |I|^{\mathcal{O}(1)}$ time. A *kernelization algorithm* reduces any instance (I, k) in polynomial time to an equivalent instance (I', k') with $|I'|, k' \leq g(k)$ for some computable g. The instance (I', k') is called *kernel* of size g and in the special case of g being a polynomial it is a polynomial kernel. A kernel is often described by the application of several so-called *data reduction rules*, that are,

polynomial time algorithms that transform an instance (I, k) into an equivalent instance (I', k'). The equivalence of (I, k) and (I', k') is called the *correctness of the data reduction rule*. We say that an instance is *reduced* with respect to a data reduction rule if its application would not change the instance.

The monographs of Downey and Fellows [19], Flum and Grohe [20], Niedermeier [34] contain a more comprehensive introduction. However, a recent concept not presented in these monographs is *Turing kernelization*. Roughly speaking, in Turing kernelization one creates many problem kernels instead of just 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 work, we assume that, unless stated otherwise, the structural parameter (for example, a feedback edge set) under consideration is provided as an additional input of the 2-Club instance.

The Strong exponential time hypothesis (SETH) states that the satisfiability problem for n-variable boolean formulas $\mathcal{F}$ in conjunctive normal form, called Cnf-Sat, cannot be solved in $\mathcal{O}((2-\epsilon)^n \cdot |\mathcal{F}|^{\mathcal{O}(1)})$ time for any $\epsilon > 0$; refer to [30] for a survey on (S)ETH-based lower bounds.

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.

2.1 A Quadratic-Vertex Kernel for the Parameter Cluster Editing Set Size

We show how to obtain an $\mathcal{O}(k^2)$-vertex kernel when k is the size of a (not necessarily minimum-cardinality) cluster editing set. Note that if a cluster editing set D is not given, then we can use a polynomial-time 2.5-factor approximation algorithm for Cluster Editing [39]. The parameterized complexity of Cluster Editing has been extensively studied [6].

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. Then, the graph $G - V(D)$ is a cluster graph, that is, all connected components of $G - V(D)$ are cliques. Observe that $N(v) = N(w)$ for any two vertices v, w in the same clique of $G - V(D)$. The following two rules yield an $\mathcal{O}(k^2)$-vertex kernel for 2-Club.

Rule 1 *Let C be a cluster in $G - V(D)$ and set $D^C = N(v) \cap V(D)$ for some $v \in C$. If C or $N[C \cup D^C]$ is a 2-club of size at least ℓ, then reduce to a constant-size yes-instance. Otherwise, if $|D^C| \leq 1$, then delete C.*

Lemma 1 *Rule 1 is correct and can be exhaustively applied in $\mathcal{O}(n^2m)$ time. Furthermore, the resulting graph G has at most k clusters in $G - V(D)$ and each of them has size less than ℓ.*

Proof: We first prove correctness. Observe that $N(v) = N(w)$ for any two vertices v and w in a cluster C of $G-V(D)$. Clearly, if C or $N[C \cup D^C]$ are 2-clubs of size at least ℓ, it is correct to reduce to a yes-instance. Otherwise, the rule deletes C if $|D^C| \leq 1$. If $D^C = \emptyset$, it is correct to delete C, because C is an isolated clique with $|C| < \ell$. In the remaining case $|D^C| = 1$. Consequently, $C \cup D^C$ is a clique. Thus the set $N[C \cup D^C]$ is a 2-club of size less than ℓ and it is the largest 2-club containing any vertex from C. Therefore, it is correct to delete C.

After exhaustive application of Rule 1, $|D^C| > 1$ for each cluster C in $G-V(D)$. Since $|V(D)| \leq 2k$ this implies that the number of clusters in $G-V(D)$ is at most k and each of them has size less than ℓ. The running time follows from applying for each cluster an all-pair-shortest path algorithm. Clearly, clusters in $G - V(D)$ can be identified in $\mathcal{O}(n + m)$ time. □

Since each cluster in $G - V(D)$ has size at most $\ell - 1$ (Lemma 1) it follows that if $\ell \leq 2k + 1$, then there are at most $2k^2 + 2k$ vertices left and we are done.

We next provide a second data reduction rule that is applied after Rule 1 if $\ell > 2k + 1$. 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 an inclusion-maximal 2-club either contains all or no vertices from C. 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 2 *If $\ell > 2k + 1$, then delete $\ell - 2k - 1$ arbitrary vertices in each cluster C of $G - V(D)$ and set $\ell := 2k + 1$.*

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

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

Proof: Let $I = (G, \ell, D)$ be an instance where Rules 1 and 2 have been applied. Since I is reduced with respect to Rule 1, there are at most k clusters each of size less than ℓ in $G - V(G)$. Finally, $\ell \leq 2k + 1$ due to Rule 2. Altogether this implies that G contains at most $2k^2 + 2k$ vertices.

As to the running time, Rule 1 can be performed in $\mathcal{O}(n^2m)$ time (Lemma 1). The clusters in $G - V(D)$ can be computed in $\mathcal{O}(n + m)$ time and deleting $\ell - 2k - 1$ vertices in C (Rule 2) is clearly doable in $\mathcal{O}(|C|^2)$ time. Hence, the corresponding running time is dominated by the application of Rule 1. □

The correctness of Rule 2 is based on Observation 2 which is only valid for 2-clubs. For $s > 2$ it is open whether there exists a polynomial kernel for s-Club parameterized by the size of a cluster editing set. However, the more general case of Weighted s-Club, where the vertices have positive weights ω and the task is to find an s-club S such that $\omega(S) := \sum_{v \in S} \omega(v) \geq \ell$, has even a linear vertex kernel. It is motivated by the fact that all vertices in a cluster C of $G - V(D)$ are twins and thus by Observation 1 either all or non of them are contained in a maximum s-club.

The linear vertex kernel for Weighted s-Club is an adaption of the quadratic vertex kernel for 2-Club (see Theorem 1). We first adapt Rule 1 for weights. Therefore, note that the proof of Lemma 1 uses the restrictions of 2-clubs only in case $N[C \cup D^C]$ is a 2-club.

Rule 3 *If there is a cluster C in $G - V(D)$ with $\omega(C) \geq \ell$, then reduce to a constant-size yes-instance. Otherwise, if $N(C) \cap V(D) = \emptyset$, then delete C.*

After exhaustively applying Rule 3, there are at most $2k$ remaining clusters in G as each cluster has at least one neighbor in $V(G)$ and $|V(G)| \leq 2k$. The next data reduction rule uses the weights on the vertices in order to merge the remaining clusters in $G - V(D)$ into one vertex.

Rule 4 *For each cluster C in $G - V(D)$, delete all but one vertex in C and set the weight of the remaining vertex to $\omega(C)$.*

The correctness of the data reduction rule follows from Observation 1. After applying Rule 4, each cluster C has size one and, hence, the application of Rules 3 and 4 leads to the following.

Theorem 2 *The* Weighted s-Club *problem parameterized by the size k of a cluster editing set admits an $\mathcal{O}(n + m)$-time computable $4k$-vertex kernel.*

Clearly, assigning weight one to each vertex is a trivial reduction from s-Club to Weighted s-Club. Thus, Theorem 2 implies a single-exponential time algorithm solving (weighted) s-Club parameterized by the size of a cluster editing set.

Corollary 1 (Weighted) s-Club *for $s \geq 2$ parameterized by the size k of a cluster editing set can be solved in $\mathcal{O}(6.86^k \cdot k^3 + n + m)$ time.*

Proof: First, apply Rules 3 and 4 and thus by Theorem 2 the remaining instances consists of at most $4k$ vertices. Hence, applying the search tree algorithm running in $\mathcal{O}(1.62^n \cdot nm)$ time [15], yields a $\mathcal{O}(6.86^k \cdot k^3 + n + m)$-time algorithm. □

2.2 A Linear Kernel for the Parameter Feedback Edge Set Size

A *feedback edge set* F of a graph G is an edge set whose deletion transforms the graph G into a forest. In this section, we study 2-Club parameterized by the size of F. Given an instance $(G = (V, E), \ell)$ of 2-Club, we first compute a minimum-size feedback edge set F of G in linear time. This can be done by computing a spanning forest T of G in $\mathcal{O}(n + m)$ time and then adding to F each edge of E that is not contained in T. In the following, let $k := |F|$ denote the parameter feedback edge set size. We present two reduction rules for 2-Club and show that exhaustively applying these rules yields a problem kernel with at most $5k + 1$ vertices and at most $6k$ edges.

The first rule checks whether G contains a trivial 2-club of size at least ℓ. The correctness of the rule follows from the fact that for each vertex v the set $N[v]$ is a 2-club. Note that we can check in linear time whether the rule is applicable.

Rule 5 *If there is a vertex $v \in V$ with $|N[v]| \geq \ell$, then replace (G, ℓ) by a yes-instance with one vertex.*

If the rule applies at least once, then we have obtained the claimed problem kernel as the resulting graph has only one vertex. Thus, assume in the following that Rule 5 does not apply. The second rule identifies vertices that are only in trivial 2-clubs. As Rule 5 does not apply, these trivial 2-clubs are to small. Such vertices may thus be safely removed from the graph.

As we will show, in a non-trivial 2-club, any vertex needs to be on a path between two feedback edges of the 2-club. These paths and the corresponding vertices are defined as follows.

Definition 1 *Let $G = (V, E)$ be a graph with a feedback edge set F. For a feedback edge $\{u, v\} \in F$ the path $P_{\{u,v\}}$ between u and v in $T = (V, E \setminus F)$ is called* feedback edge path. *If a vertex w lies on $P_{\{u,v\}}$, then $\{u, v\}$ is a* spanning feedback edge *of w. We also say that $\{u, v\}$* spans *w.*

Lemma 2 *Let (G, ℓ) be an instance of* 2-Club *to which Rule 5 does not apply and let S be a 2-club of size at least ℓ in G. Then, for each vertex $w \in S$, $G[S]$ contains a feedback edge $\{u, v\}$ that spans w.*

Proof: Let S be a 2-club and let $w \in S$ be a vertex w such that $G[S]$ does not contain a spanning feedback edge of w. We show that this implies $|S| < \ell$. Since $G[S]$ does not contain any spanning feedback edge of w, the vertex w is not contained in any cycle in $G[S]$. Therefore, either w has degree one in $G[S]$ or w is a cut-vertex in $G[S]$. In the first case, S is completely contained in the neighborhood of w's neighbor in $G[S]$. In the second case, $S \subseteq N[w]$. Since Rule 5 does not apply, this implies $|S| < \ell$. □

We now exploit the above lemma by identifying those vertices that are not close enough to any spanning feedback edge. We define this closeness as follows.

Definition 2 *Let G be a graph with a feedback edge set F. A vertex w is* satisfied by a feedback edge $\{u, v\} \in F$ *in G if $\{u, v\}$ spans w and if w has in G distance at most two to u and to v.*

If there exists at least one feedback edge that satisfies w, then w is called satisfied. The rule now deletes vertices that are too far away from all spanning feedback edges.

Rule 6 *Let (G, ℓ) be an instance to which Rule 5 does not apply. If G contains a vertex w that is not satisfied, then delete w.*

Lemma 3 *Rule 6 is correct and can be exhaustively applied in $\mathcal{O}(n^2m)$ time.*

Proof: We first prove correctness by showing that w is not contained in a 2-club of size at least ℓ. Let S be a 2-club containing w. Since w is not satisfied it has in G distance at least three to at least one endpoint of every feedback edge that spans w. Thus, S contains at most one endpoint of every feedback edge that spans w. Thus, $G[S]$ contains no feedback edge that spans w. By Lemma 2, this implies $|S| < \ell$. Therefore, removing w from G yields an equivalent instance.

The running time can be achieved as follows. First, by starting a breadth-first search from each vertex in G, compute in $\mathcal{O}(nm)$ time the set $N_2(v)$ for all $v \in V$. Then, in $\mathcal{O}(nm)$ time, store this information in a matrix. This allows to test whether $u \in N_2(v)$ in constant time. For all feedback edges e, compute in $\mathcal{O}(n)$ time the set of vertices that are on the feedback edge path P_e. Now check for each vertex v on each P_e whether both endpoints of e are in $N_2(v)$. If this is the case, then label v as satisfied. Finally, remove all unlabeled vertices in $\mathcal{O}(n+m)$ time. Altogether, one application of the rule runs in $\mathcal{O}(nm)$ time and the rule can be applied at most n times. □

The kernelization algorithm now first checks once whether Rule 5 applies and then exhaustively applies Rule 6. This algorithm produces in $\mathcal{O}(n^2m)$ time an instance that is reduced with respect to Rules 5 and 6. It remains to bound the size of the reduced instance. In order to obtain a tight bound on the vertex number, we analyze the relationship between feedback edges. To this end, we introduce the following definition (see Figure 1 for an illustration).

Definition 3 *Let G be a graph with spanning forest T and feedback edge set F. We say that a feedback edge e* covers *a feedback edge e' if the path P_e completely contains the path $P_{e'}$.*

In the proof of the size bound, we iteratively add the edges of F to T to obtain G. The following lemma states that this can be done in such a way that the new edge does not cover any other edge in the current graph.

Lemma 4 *Let G be a graph with spanning forest T and feedback edge set F. Then there is a set of graphs $T_0, \ldots, T_k$ such that $T_0 = T$, $T_k = G$, and T_i, $i > 0$, is obtained from T_{i-1} by adding one edge e such that T_{i-1} contains all feedback edges that cover e and no feedback edge that is covered by e.*

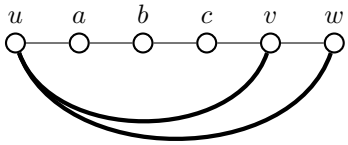

Figure 1: Illustration of the definitions used in the proof of Theorem 3. Herein, bold edges are feedback edges, regular edges are edges of T. The path $P_{\{u,v\}}$ connects the endpoints of the feedback edge $\{u,v\}$ in T. The feedback edge $\{u,w\}$ covers the feedback edge $\{u,v\}$; the feedback edge $\{u,v\}$ is u-minimal; the u-neighbor of $\{u,v\}$ is a, the v-neighbor is c; the vertices a, b and c are all satisfied.

Proof: Start with $T_0 := T$ and iteratively add a feedback e from G such that P_e has maximum length. If an edge e is added in step i, then all edges that cover e are in T_{i-1} since their feedback edge paths are longer. Similarly, for each edge in T_{i-1}, the feedback edge path is at least as long as P_e. Thus, T_{i-1} does not contain an edge that is covered by e. Since in each step one feedback edge is added and since $|F| = k$ we have $T_k = G$. □

Now we can prove the size bound on the kernel.

Theorem 3 2-Club *parameterized by the size k of a feedback edge set admits a kernel with at most $5k+1$ vertices and at most $6k$ edges. This kernel can be computed in $\mathcal{O}(n^2m)$ time.*

Proof: Let (G,ℓ) be an instance that is reduced with respect to Rules 5 and 6. If Rule 5 was successful, then G has only one vertex and the size bound trivially holds. Otherwise, the number of vertices in 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 feedback edge has size at most $3k$. Note that all vertices in G are satisfied since the instance is reduced with respect to Rule 6.

The main idea of the proof is to consider a sequence of graphs $T_0, \ldots, T_k$ that fulfills the properties of Lemma 4 and to bound the number of satisfied vertices in each T_i by $3i$. Since $T_k = G$ this implies $|Y| \leq 3k$. To this end, we show how to construct a set $Y_i \subseteq \{(v,e) \mid v \in V, e \in F\}$ that contains pairs of satisfied vertices and spanning feedback edges to which they are attributed. The central property for Y_i is that each satisfied vertex in T_i is contained in at least one pair of Y_i. Thus, the number of satisfied vertices in T_i is at most $|Y_i|$. We show this property by induction. We maintain one further property of Y_i that is needed in the proof of the inductive step. To formulate this property we introduce some further terminology (see Figure 1): A feedback edge $\{u,v\}$ is called *u-minimal* in T_i if there is in T_i no feedback edge $\{u,w\}$ that is covered by $\{u,v\}$. A vertex w is a u-neighbor of a feedback edge $\{u,v\}$ if $\{u,v\}$ spans w and w is in T a neighbor of u.

By induction on i we show the main claim:

For each $i \in \{0, \dots, k\}$ there is a set Y_i with the following properties:

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

Base case: Let $Y_0 := \emptyset$. Then, Property 1 holds as $|Y_0| = 0$. Moreover, no vertex in T_0 is satisfied so Property 2 holds. Finally, T_0 does not contain feedback edges so Property 3 holds. Hence, the claim holds for $i = 0$.

Inductive step: By induction, the claim holds for $i - 1$. Let $\{u, v\}$ be the feedback edge that is added from T_{i-1} to T_i. We construct Y_i fulfilling the three properties as follows. Initially, set $Y_i := Y_{i-1}$. Let a be the u-neighbor of $\{u, v\}$ and let c be the v-neighbor of $\{u, v\}$. Add the pairs $(a, \{u, v\})$ and $(c, \{u, v\})$ to Y_i. Next, if there is a vertex b that is in T adjacent to a and c, then add $(b, \{u, v\})$ to Y_i. After these additions, $|Y_i| \leq |Y_{i-1}| + 3$ since we have added at most three pairs. Thus, Y_i fulfills Properties 1 at this point. It also fulfills Property 3 since the only edge that can be minimal in T_i and not in T_{i-1} is $\{u, v\}$ and we have added the pair $(a, \{u, v\})$ and the pair $(b, \{u, v\})$.

Let $Z \subseteq Y$ denote the set of vertices that are satisfied in T_i and not yet contained in any pair of Y_i. Then, add the set of pairs $\{(z, \{u, v\}) \mid z \in Z\}$ to Y_i. Now Y_i fulfills Properties 2 and 3 of the claim but possibly not Property 1. To restore Property 1, we show how to remove $|Z|$ pairs from Y_i without violating Properties 2 and 3.

To this end, we first observe a property of Z. Let $z \in Z$, we show that z is a neighbor of u or v. First, assume towards a contradiction, that z is satisfied by $e = \{u, v\}$. Since $z \notin \{a, b, c\}$, it has in T distance at least three from either u or v, say v. Since z is satisfied by $\{u, v\}$, there is some feedback edge e' such that the length-two path from z to v contains e'. Then one of the endpoints of e' is a neighbor of z in T and thus contained in $P_{\{u,v\}}$, the other endpoint is v. Then, the edge e' is covered by e which contradicts our assumption on the sequence of the T_i's. Thus, z is not satisfied by $\{u, v\}$. In this case, z becomes satisfied because there is in T_i a length-two path to some endpoint of e that is not contained in T_{i-1}. This length-two path contains $\{u, v\}$ which means that z is a neighbor of u or v. This also implies that z is not in $P_{\{u,v\}}$ since $z \notin \{a, c\}$.

Now, partition Z into $Z_u := Z \cap N[u]$ and $Z_v := Z \cap N[v]$. Such a partition is possible since the vertices of Z are not contained in $P_{\{u,v\}}$. Thus, 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 $z_1, \dots, z_q$ denote the vertices in Z_v.

By definition, each z_j is adjacent to v and there is a feedback edge e such that z_j is satisfied by e in T_i but not in T_{i-1}. Consequently, u is one of the endpoints of e, that is, $e = \{u, w_j\}$ for some w_j. Since z_j is not contained in $P_{\{u,v\}}$ but a neighbor of v we have that $P_{\{u,z_j\}}$ contains $P_{\{u,v\}}$. Thus, $\{u, w_j\}$ covers $\{u, v\}$. Furthermore, for all z_j's the w_j's are pairwise distinct. Thus,

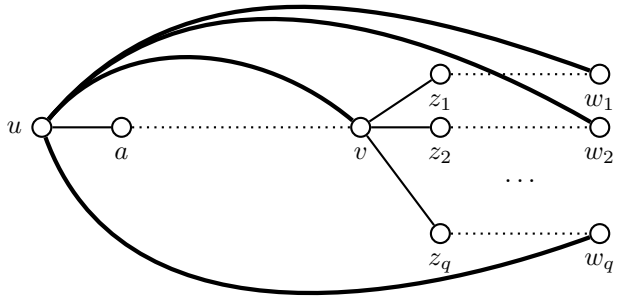

Figure 2: Illustration of the vertices in Z_v and of the corresponding spanning edges.

the situation is as depicted in Figure 2. We now show that for each z_j there is a pair $(a, \{u, p_j\})$ (recall that a is the u-neighbor of $\{u, v\}$) in Y_i that can be removed from Y_i without violating Property 2 or 3. First, note that in T_{i-1} there is for each feedback edge $\{u, w_j\}$ a u-minimal edge $\{u, p_j\}$ that is covered by $\{u, w_j\}$. By construction, T_{i-1} does not contain feedback edges that are covered by $\{u, v\}$. Thus, we have $p_j \neq p_r$ if $j \neq r$. By Property 3, Y_{i-1} contains the pair $(a, \{u, p_j\})$. Since Y_i is, currently, a superset of Y_{i-1} this pair $(a, \{u, p_j\})$ is also contained in Y_i. Now the main observation is that since T_i contains $\{u, v\}$ the edge $\{u, p_j\}$ is not u-minimal in T_i. Recall that the pair $(a, \{u, v\})$ is already contained in Y_i and that $\{u, v\}$ is u-minimal. Consequently, for each $z_j \in Z_v$, the pair $(a, \{u, p_j\})$ can be removed without violating Properties 2 or 3. Hence, we can remove $|Z_v|$ pairs from Y_i without violating either property. Analogously, we can remove $|Z_u|$ further pairs corresponding to the vertices in Z_u without violating either property. The resulting set Y_i of pairs has size at most $|Y_{i-1}| + 3 + |Z| - |Z_v| - |Z_u| = |Y_{i-1}| + 3 \leq 3 \cdot (i-1) + 3 = 3i$. Thus, Y_i fulfills Property 1 as well and the claim also holds for i.

The overall kernel bound now follows from $|Y| \leq |Y_k| \leq 3k$. This implies that $|V| = |X| + |Y| \leq 5k$. The number of edges in T is at most $5k - 1$. Since $|F| = k$, the number of edges in G is thus at most $6k - 1$. □

Applying the $\mathcal{O}(1.62^n \cdot nm)$-time algorithm [15] on the kernel we obtain a single-exponential running time in k.

Corollary 2 2-Club *parameterized by the feedback edge set size k can be solved in $\mathcal{O}(11.1^k \cdot k^3 + n^2 m)$ time.*

2.3 Lower Bounds for Kernelization

We next show that 2-Club does not admit a polynomial kernel with respect to the parameters *bandwidth* of the input graph and *vertex cover size* of the input

graph. The first result implies that 2-Club does not admit a polynomial kernel for the parameter *maximum degree*. The second result implies that 2-Club does not admit a polynomial kernel for parameters such as *feedback vertex set number*, and *treewidth*.

First, we show a lower bound for bandwidth. A graph has *bandwidth* k if it has a linear arrangement of its vertices $v_1, \ldots, v_n$ such that the length $|i - j|$ of each edge $\{v_i, v_j\}$ is at most k. Furthermore, we observe that, given a linear arrangement with bandwidth k, a linear-vertex Turing kernel can be achieved.

Proposition 1 2-Club *parameterized by the bandwidth k does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly; it admits a $2k$-vertex Turing kernel which can be computed in $\mathcal{O}(nm)$ time.*

Proof: We first show that 2-Club does not admit a polynomial problem kernel by showing that it is compositional. A parameterized problem L is compositional if there is a polynomial-time algorithm (a composition) that takes as input instances $(I_1, k), \ldots, (I_t, k)$ of L and computes a new instance (I, k') where k' is upper-bounded by a polynomial in k and (I, k') is a yes-instance if and only if (I_j, k) is a yes-instance for some $1 \leq j \leq t$. Unless NP $\subseteq$ coNP/poly, an NP-hard parameterized problem does not admit a polynomial kernel if it is *compositional* [8]. Note that the parameter bandwidth k is *not* the size of the 2-club. It is, however, sufficient to show a composition for instances that have the same ℓ, since ℓ defines a polynomial equivalence relation on 2-Club instances [10].

Since 2-clubs are connected, taking the disjoint union of t input graphs gives a composition. If the t input graphs have bandwidth k, then the disjoint union has also bandwidth k. Hence, 2-Club does not admit a polynomial kernel.

We next show that 2-Club parameterized by bandwidth has a linear Turing kernel. Let $G = (V, E)$ be a graph with bandwidth k and let the vertices be labeled $v_1, \ldots, v_n$ such that for each edge $\{v_i, v_j\} \in E$ it holds that $|i - j| \leq k$. The Turing kernel can be obtained by creating for each $v_i \in V$ the graph $G[\{v_i, v_{i+1}, \ldots, v_{i+2k}\}]$. The idea of the Turing kernel is simply to try all possibilities to choose the vertex v_i which has the lowest index of the 2-club vertices. By definition of bandwidth, all vertices v_j, $j > i$ within distance two of v_i are in $\{v_{i+1}, v_{i+2}, \ldots, v_{i+2k}\}$ and, hence, $S \subseteq \{v_i, v_{i+1}, \ldots, v_{i+2k}\}$. Thus, the Turing kernel follows. □

The hardness result in Proposition 1 directly implies that 2-Club also has no polynomial kernel for the parameter maximum degree. Next, we show that 2-Club parameterized by *vertex cover* does not admit a polynomial kernel. This implies the same lower bound for many other parameters such as *size of a feedback vertex set*, *treewidth*, *degeneracy*, *distance to cluster graphs*, etc. (see [24, Fig. 1] for a more detailed description of related parameters).

Theorem 4 2-Club *parameterized by the* size of a vertex cover *has no polynomial kernel unless NP $\subseteq$ coNP/poly.*

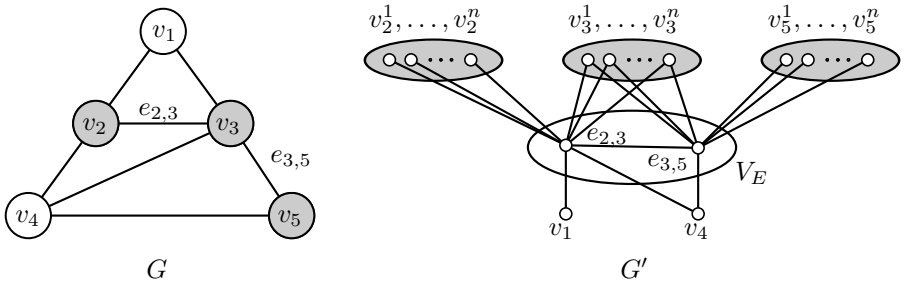

Figure 3: Example of the construction in the proof of Theorem 4. The graph G is an input for Clique parameterized by vertex cover. The gray vertices are a vertex cover in G. The graph G' is constructed as described in the proof of Theorem 4.

Proof: We give a polynomial-time and -parameter reduction [9] from Clique parameterized by *vertex cover* to 2-Club parameterized by *vertex cover*. Unless NP $\subseteq$ coNP/poly, Clique does not admit a polynomial kernel with respect to the size of a *vertex cover* [10]. Let $(G = (V, E), X, h)$ be an instance of Clique that asks for the existence of a clique of size h. Here, X is a vertex cover of G. We construct a graph G' as follows (for an illustration see Figure 3). 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 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-h 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

$$G \text{ has a clique of size } h \Leftrightarrow G' \text{ has a 2-club of size at least } |V_E| + (h-1)n + 1.$$

"$\Rightarrow$": Let K be a size-h clique in G. Observe that $|K \setminus X| \leq 1$ since K is a clique. 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 $|V_E| + hn$. Otherwise, one can add the vertex $v \in K \setminus X$ to S. Then, S has size $|V_E| + (h-1)n + 1$ 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.

"⇐": Let S be a 2-club of size at least $|V_E| + (h-1)n + 1$ 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 $h - 1$. 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 h, then there is a size-h clique in G. It thus remains to consider the case where a largest twin-free set F is of size $h - 1$. 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 $v_i \in F$ 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 v in G. Consequently, $F \cup \{v\}$ is a size-h clique in G. □

3 A Fixed-Parameter Algorithm for Treewidth

In this section, we show that 2-Club is fixed-parameter tractable when parameterized by treewidth. Although, this was already shown by Schäfer [37] via a monadic second order logic formulation we here present a direct combinatorial algorithm. Surprisingly, up to some constants in the exponent, this is, so far, even the best algorithm for much larger parameters such as *size of a vertex cover*. To demonstrate the principle idea behind the algorithm, we first describe its principles for the parameter vertex cover.

Theorem 5 *s-Club is solvable in $\mathcal{O}(2^k \cdot 2^{2^k} \cdot nm)$ time where k denotes the* size of a vertex cover.

Proof: Let $(G = (V, E), X, s, \ell)$ be an s-Club instance where X is a vertex cover of G with $|X| = k$. 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 add them to the s-club takes 2^{2^k} time. Finally, check in $\mathcal{O}(nm)$ time whether the resulting graph forms an s-club. In total, s-Club can be solved in $\mathcal{O}(2^k \cdot 2^{2^k} \cdot nm)$ time. □

Extending the ideas behind Theorem 5, we now give a direct combinatorial algorithm for the parameter treewidth. More specifically, we provide a dynamic programming algorithm on a *nice tree decomposition* of the input graph G (see Niedermeier [34, Chapter 10] for more details about nice tree decompositions): It is a tree decomposition T, that is, a tree with vertices $X_1, \ldots, X_r$ called *bags* such that $\bigcup_i X_i = V(G)$ and for each $\{u, v\} \in E(G)$ there is a bag X_i with $\{u, v\} \in X_i$. Additionally, for each $v \in V(G)$ the bags containing v induce a connected component in T. In a nice-tree decomposition, each bag X_i is either a *join node* (it has exactly two children with $X_i = X_j = X_l$), an *introduce node*

($|X_i \setminus X_l| = 1$ for the only child X_l), a *forget node* ($|X_l \setminus X_i| = 1$ for the only child X_l), or a *leaf node* (no children). An arbitrary tree decomposition can be restructured in linear time into a nice tree decomposition without an increase in the size of the largest bag [27]. Deciding whether a graph has treewidth ω and (in case of its existence) constructing a corresponding tree decomposition can be done in $2^{\mathcal{O}(\omega^3)} \cdot n$ time [7].

Lemma 5 *Let $G = (V, E)$ be a graph and let $S \subseteq V$. 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 $v \in S$ such that the length of the path from X^u to the first bag that contains v is maximum. Denote this bag by X^v. We show that $N(v) \cap S \subseteq X^v$. Suppose there is a neighbor $w \in N(v) \cap S$ that is not contained in X^v. Since w and v are adjacent they are together contained in at least one bag. Since X^v is the first bag containing v on the path from X^v to X^u and the bags containing w induce a connected component, from $w \notin X^v$ and $w \in N(v)$ it follows that the path from X^u to the first bag containing w is via X^v and thus longer; a contradiction to the choice of v. □

Theorem 6 2-Club *is solvable in $2^{\mathcal{O}(2^\omega)} \cdot n$ time where ω denotes the treewidth the input graph.*

Proof: Let (G, k) be an input instance of 2-Club and let $(X_1 \cup \ldots \cup X_r, E)$ be a nice tree decomposition for G of width ω. Fix a maximum-size 2-club S. By Lemma 5 there is a vertex $v \in S$ such that $N[v] \cap S \subseteq X$ for a bag X. Let $N_v := N[v] \cap S \subseteq X$. First, since $r = \mathcal{O}(n)$ there are $\mathcal{O}(n \cdot \omega \cdot 2^\omega)$ cases for choosing the bag X, the vertex v and its neighbors N_v in X. After having done this, we root the nice tree decomposition in X. Furthermore, we may assume that $X = N_v$, as otherwise one can add a path of forget nodes that starts in X and step-wisely deletes all vertices in $X \setminus N_v$. Next, we describe a bottom-up dynamic programming algorithm.

Denote by 2^P the set of all subsets of a set P. Let X_i be an arbitrary bag. We have for each combination of some $P \subseteq X_i$ and some $\mathcal{T} \subseteq 2^P$ an integer table entry $\mathrm{Tab}_i(P, \mathcal{T})$ which is the size of the largest set $K_i(P, \mathcal{T})$ fulfilling the following properties:

1. $K_i(P, \mathcal{T})$ is a subset of the vertices in the subtree rooted in X_i and $K_i(P, \mathcal{T}) \cap X_i = P$,
2. each *type* $T \subseteq P$ is in $\mathcal{T}$ if and only if there is a vertex $v \in K_i(P, \mathcal{T}) \setminus P$ of type T, that is, $N(v) \cap P = T$, and
3. in $G[K_i(P, \mathcal{T})]$ each vertex in $K_i(P, \mathcal{T})$ has distance at most two to all vertices in $K_i(P, \mathcal{T}) \setminus P$.

Intuitively, $K_i(P, \mathcal{T})$ is almost a 2-club as only the vertices from P are allowed to have distance more than two and all vertices not in X_i are of one of the types in $\mathcal{T}$. Clearly, the table has at most $2^\omega \cdot 2^{2^\omega}$ entries per bag and we show how to compute each entry in $\mathcal{O}(2^{2^\omega})$ time. This implies, together with first step to guess X and N_v, the claimed running time of $2^{\mathcal{O}(2^\omega)} \cdot n$.

Before showing how to compute the table entries, we prove the claim that a largest value $\text{Tab}(N_v, \mathcal{T})$ for any $\mathcal{T}$ in the table of the root X is equal to the size of a largest 2-club in G that contains N_v:

First, consider a table entry $\text{Tab}(N_v, \mathcal{T})$. By Property 1 it follows that $N_v = K(N_v, \mathcal{T}) \cap X$ and since $N_v \subseteq N[v]$ it follows by Property 3 that $K(N_v, \mathcal{T})$ is a 2-club. Hence, $|S| \geq |K(N_v, \mathcal{T})|$. In the other direction, let S' be a 2-club in G with $N_v \subseteq S'$. Let $\mathcal{T}$ be the set containing all $T \subseteq N_v$ where a vertex $u \in S' \setminus N_v$ with $N(u) \cap N_v = T$ exists. By definition, S' fulfills Properties 1 and 2 for $K(N_v, \mathcal{T})$ and since S' is a 2-club also Property 3, implying that $|S'| \leq K(N_v, \mathcal{T})$.

Computation of the table entries. We now specify how to compute the table $\text{Tab}_i(P, \mathcal{T})$ for a bag X_i by distinguishing whether X_i is a leaf, an introduce node, a join node, or a forget node. Table entries where a corresponding set fulfilling Properties 1, 2 and 3 do not exists are set to -∞.

Leaf node: Let X_i be a leaf in the tree decomposition. Clearly, since $K(P, \mathcal{T})$ has to be a subset of vertices in the subtree of X_i (Property 1), by Property 2 we only have to consider the case where $\mathcal{T} = \emptyset$. Then $K(P, \mathcal{T})$ is equal to P, hence, we set $\text{Tab}_i(P, \mathcal{T}) = |P|$ if $\mathcal{T} = \emptyset$ and otherwise $\text{Tab}_i(P, \mathcal{T}) = -\infty$.

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, assume $u \notin P$. Then, by Property 1 it is correct to set $\text{Tab}_i(P, \mathcal{T})$ to the value of $\text{Tab}_\ell(P, \mathcal{T})$.

Second, assume $u \in P$. By Property 1 all vertices in $K_i(P, \mathcal{T}) \setminus P$ are not in X_i and since $u \in X_i \setminus X_\ell$, if there is a type $T \in \mathcal{T}$ with $u \in T$, then we set $\text{Tab}_i(P, \mathcal{T}) = -\infty$ as Property 2 cannot be fulfilled. Additionally, if there is a type $T \in \mathcal{T}$ with $T \cap N(u) = \emptyset$, then also set $\text{Tab}_i(P, \mathcal{T}) = -\infty$ as Property 3 cannot be fulfilled. In all other cases, observe that $K_\ell(P \setminus \{u\}, \mathcal{T}) \cup \{u\}$ fulfills all properties for $K_i(P, \mathcal{T})$, hence, it is correct to set $\text{Tab}_i(P, \mathcal{T}) = \text{Tab}_\ell(P \setminus \{u\}, \mathcal{T}) + 1$.

Forget node: Let X_i be a forget node with the child node X_ℓ and $u \in X_\ell \setminus X_i$. Since $u \notin P$ and because of Properties 1 and 2, $u \in K(P, \mathcal{T})$ can be true only if $T_u = N(u) \cap P \in \mathcal{T}$. Hence, if $T_u \notin \mathcal{T}$, then set $\text{Tab}_i(P, \mathcal{T})$ equal to $\text{Tab}_\ell(P, \mathcal{T})$.

Consider the remaining case where $T_u \in \mathcal{T}$. Let $K_i^u(P, \mathcal{T})$ be a maximum set fulfilling Properties 1, 2 and 3 for $K_i(P, \mathcal{T})$ such that $u \in K_i^u(P, \mathcal{T})$. Then, $K_i^u(P, \mathcal{T})$ either contains only vertex u as T_u-type vertex or at least two vertices of type T_u. If there are at least two, then $|K_i^u(P, \mathcal{T})| = |K_\ell(P \cup \{u\}, \mathcal{T})|$. In case of one T_u-type vertex, the size of K_i^u can be at most equal to those of $K_\ell(P \cup \{u\}, \mathcal{T} \setminus \{T_u\})$. However, to ensure Property 3 in this case one has to additionally check whether vertex u has in $G[K_\ell(P \cup \{u\}, \mathcal{T} \setminus \{T_u\})]$ distance at most two to all vertices in P (by Property 2 this can be checked by just knowing P and $\mathcal{T}$). If this check was positive, then we have $|K_i^u(P, \mathcal{T})| = \max\{|K_\ell(P \cup \{u\}, \mathcal{T})|, |K_\ell(P \cup \{u\}, \mathcal{T} \setminus \{T_u\})|\}$, otherwise $|K_i^u(P, \mathcal{T})| = |K_\ell(P \cup \{u\}, \mathcal{T})|$.

Finally, since either $u \in K_i(P, \mathcal{T})$ or not, it follows in case of $T_u \in \mathcal{T}$ that

$$\mathrm{Tab}_i(P, \mathcal{T}) = |K_i(P, \mathcal{T})| = \max\{|K_i^u(P, \mathcal{T})|, \mathrm{Tab}_\ell(P, \mathcal{T})\}.$$

Join node: Let X_i be a join node and let X_ℓ and X_j be the two child nodes with $X_i = X_\ell = X_j$. We call two subsets $\mathcal{T}_\ell, \mathcal{T}_j \subseteq \mathcal{T}$ *consistent*, if $\mathcal{T}_\ell \cup \mathcal{T}_j = \mathcal{T}$ and for any two types $T, T' \in \mathcal{T}$ with $T \cap T' = \emptyset$ it either holds that $T, T' \in \mathcal{T}_\ell$ or $T, T' \in \mathcal{T}_j$. We prove that it is correct to set

$$\mathrm{Tab}_i(P, \mathcal{T}) = \max_{\forall \text{ consistent } \mathcal{T}_\ell, \mathcal{T}_j \subseteq \mathcal{T}} \mathrm{Tab}_\ell(P, \mathcal{T}_\ell) + \mathrm{Tab}_j(P, \mathcal{T}_j) - |P|.$$

Let K_i^ℓ (K_i^j) be the intersection of $K_i(P, \mathcal{T})$ with the vertices in the subtree rooted in X_ℓ (X_j, respectively). Clearly, $K_i^\ell \cap K_i^j = P$. Additionally, let $\mathcal{T}_\ell \subseteq \mathcal{T}$ ($\mathcal{T}_j \subseteq \mathcal{T}$) be the types of the vertices in K_i^ℓ (K_i^j, respectively). Observe that, by Property 3 each vertex $u \in K_i^\ell \setminus P$ has distance at most two to any $v \in K_i^j \setminus P$. However, by the properties of a tree decomposition u and v cannot be adjacent and thus $N(u) \cap N(v) \subseteq P$. Hence the types of u and v have a non-empty intersection, implying that $\mathcal{T}_\ell$ and $\mathcal{T}_j$ are consistent. Moreover, K_i^ℓ fulfills all properties for $K_\ell(P, \mathcal{T}_\ell)$ and K_i^j fulfills all properties for $K_j(P, \mathcal{T}_j)$. This holds because removing from $K_i(P, \mathcal{T})$ a set of vertices A with either $A \subseteq K_i^\ell \setminus P$ or $A \subseteq K_i^j \setminus P$ may violate only Property 2. Hence, there are consistent $\mathcal{T}_\ell$ and $\mathcal{T}_j$ such that $|K_i(P, \mathcal{T})| \leq |K_\ell(P, \mathcal{T}_\ell)| + |K_j(P, \mathcal{T}_j)| - |P|$.

In the other direction, one can see that for each pair of consistent sets $\mathcal{T}_\ell, \mathcal{T}_j \subseteq \mathcal{T}$ it holds that $K_\ell(P, \mathcal{T}_\ell) \cup K_j(P, \mathcal{T}_j)$ fulfills all properties (except maximality) for $K_i(P, \mathcal{T})$: Since $X_i = X_\ell = X_j$ it is clear that Property 1 is fulfilled. Moreover, as for each type $T \in \mathcal{T}$ it holds that $T \in \mathcal{T}_\ell$ or $T \in \mathcal{T}_j$, also Property 2 is fulfilled. Finally, by the definition of consistency, also Property 3 is fulfilled. □

4 On the Optimality of the Dual Parameter Algorithm

In this section, we prove running time and kernelization 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.

Cnf-Sat
Input: An n-variable boolean formula $\mathcal{F}$ in conjunctive normal form (CNF).
Question: Is there an assignment to the variables in $\mathcal{F}$ that evaluates to true?

As a side result, the reduction also implies that the s-Club Cluster Vertex Deletion problem does not admit a polynomial kernel for all $s \geq 2$. This answers an open question by Liu et al. [29]. The s-Club Cluster Vertex

DELETION problem is to decide for a given graph G and an integer k whether by deleting at most k vertices in G one can obtain a graph whose connected components are 2-clubs.

Lemma 6 *For any $s \geq 2$ there is a polynomial-time reduction from* CNF-SAT *to* s-CLUB *that computes for any n-variable CNF formula an equivalent* s-CLUB*-instance (G, ℓ) with dual parameter $k' = n$ such that there are exactly k' pairwise disjoint vertex pairs each having distance $s + 1$.*

Proof: Let $\mathcal{F}$ be a CNF formula with n variables and 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. We describe how to 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* $\mathcal{V}$ to G, that is, a set that contains for each variable x in $\mathcal{F}$ the two vertices $v_x, \overline{v_x}$. Here, v_x corresponds to x and is called *positive literal vertex*, and $\overline{v_x}$ corresponds to $\neg x$ and is called *negative literal vertex*. Next, add the *clause vertex set* $\mathcal{C}$ that contains for each clause C in $\mathcal{F}$ two *clause vertices* c_1, c_2. Additionally, if the literal x ($\neg x$) occurs in the clause C, then make the corresponding clause vertices c_1 and c_2 adjacent to the positive (negative) literal vertex v_x ($\overline{v_x}$, respectively).

The basic idea of the construction is that for each variable x the two vertices v_x and $\overline{v_x}$ have distance $s + 1$ and thus one has to delete one of them. These are the $k' = n$ vertex pairs corresponding to the requirements on G in Lemma 6. The clause vertices c_1, c_2 have the literal vertices corresponding to the literals in C as common neighbors and there is no path of length at most s between them that avoids all of these common neighbors. Hence, the remaining literal vertices (exactly one for each variable) correspond to a satisfying assignment for $\mathcal{F}$ if and only if for each clause C the clause vertices c_1, c_2 still have at least one literal vertex as a common neighbor.

We extend the graph G to fulfill the above-mentioned properties: For each vertex pair $\{u, w\} \subseteq \mathcal{V} \cup \mathcal{C}$ not equal to $\{v_x, \overline{v_x}\}$ for some variable x and also not equal to $\{c_1, c_2\}$ for some clause C, add a vertex $l_{u,w}$, add a length-$\lfloor \frac{s}{2} \rfloor$ path from $l_{u,w}$ to w, and also a length-$\lfloor \frac{s}{2} \rfloor$ path from $l_{u,w}$ to u. This implies that $\mathrm{dist}(u, w) = s$ in case of even s and $\mathrm{dist}(u, w) = s - 1$ otherwise. Collect all the vertices $l_{u,w}$ to the set L and denote by P all vertices on the $\lfloor \frac{s}{2} \rfloor$-length paths except the endpoints. Clearly, in case of $s \in \{2, 3\}$ all the paths inserted above are only edges between the endpoints and thus $P = \emptyset$.

Finally, if s is odd, then add a distinguished vertex l which is adjacent to each vertex in L and otherwise, if s is even, add edges such that L is a clique.

This ensures that a pair $\{u, w\} \subseteq \mathcal{V} \cup \mathcal{C}$ has distance at most s if and only if there is some vertex $l_{u,w} \in L$ or $\{u, w\}$ is equal to the clause vertices $\{c_1, c_2\}$ for some clause C. We next prove that (G, ℓ) with $\ell = n' - k'$ is a yes-instance of s-CLUB $\Leftrightarrow$ $\mathcal{F}$ has a satisfying assignment.

"$\Rightarrow$": Let D be a vertex subset of G such that $|D| \leq k'$ and $G - D$ is an s-club. Since the positive v_x and negative literal vertex $\overline{v_x}$ of each variable x have distance $s + 1$, either $v_x \in D$ or $\overline{v_x} \in D$ and since there are $k' = n$ variables

it follows that $D \subseteq \mathcal{V}$. Hence all clause vertices are contained in $G - D$ and thus the pair $\{c_1, c_2\}$ for each clause C has distance at most s. This implies that there is a vertex $u \in \mathcal{V}$ such that c_1 and c_2 are adjacent to u. By the construction, u corresponds to a literal in the clause C and thus the remaining literal vertices in $G - D$ correspond to a satisfying assignment of $\mathcal{F}$.

"$\Leftarrow$": Assume that there is a satisfying assignment for $\mathcal{F}$ and let $D \subseteq \mathcal{V}$ be the set of literal vertices whose corresponding literals in $\mathcal{F}$ are assigned to be false. Clearly, $|D| = k' = n$ and it remains to prove that $G - D$ is an s-club:

By construction, for any two vertices $\{u, w\} \subseteq \mathcal{V} \cup \mathcal{C}$ there is either a vertex $l_{u,w} \in L$ or if $\{u, w\}$ corresponds to some $\{c_1, c_2\}$ for some clause C, then there is common literal neighbor in $\mathcal{V}$ not contained in D, implying that in each case $\text{dist}_{G-D}(u, w) \leq s$. To prove the remaining cases observe that each vertex in $v \in \mathcal{V} \cup \mathcal{C} \cup P \cup L \cup \{l\}$ has a length at most $\lfloor \frac{s}{2} \rfloor$ path to at least one vertex in L. Moreover, it holds that the distance from a vertex in $P \cup L \cup \{l\}$ to any vertex in L is at most $\lfloor \frac{s}{2} \rfloor + 1$ in case of odd s and at most $\frac{s}{2}$ for even s, implying that $G - D$ is a s-club. □

The Strong Exponential Time Hypothesis (SETH) fails if Cnf-Sat for n-variable CNF formulas $\mathcal{F}$ can be solved in $\mathcal{O}((2-\epsilon)^n \cdot |\mathcal{F}|^{\mathcal{O}(1)})$ time for some $\epsilon > 0$ [26]. Thus, by Lemma 6 an algorithm for s-Club running in $\mathcal{O}((2-\epsilon)^{k'} \cdot |G|^{\mathcal{O}(1)})$ time for some $\epsilon > 0$ would disprove the SETH. This bound is tight since s-Club can be solved in $\mathcal{O}(2^{k'} \cdot nm)$ time [36].

Corollary 3 *Unless the SETH fails, s-Club on a graph G parameterized by the dual parameter $k' := n - \ell$ cannot be solved in $\mathcal{O}((2-\epsilon)^{k'} \cdot |G|^{\mathcal{O}(1)})$ time for all $s \geq 2$.*

Chen et al. [16] showed that Cnf-Sat parameterized by the number of variables does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. Since Lemma 6 provides a polynomial-time and parameter transformation [11] this lower bound result transfers to 2-Club.

Corollary 4 *s-Club for all $s \geq 2$ parameterized by the dual parameter k' does not admit a polynomial problem kernel unless NP $\subseteq$ coNP/poly.*

Finally, as Lemma 6 states that the graph constructed in the reduction from Cnf-Sat to s-Club has k' pairwise disjoint vertex pairs where from each pair one has to delete at least one vertex, in this special case s-Club Cluster Vertex Deletion is equivalent to s-Club parameterized by the dual parameter k'. Thus by the same argumentation as for Corollary 4 the following holds.

Corollary 5 *s-Club Cluster Vertex Deletion for all $s \geq 2$ does not admit a polynomial problem kernel unless NP $\subseteq$ coNP/poly.*

5 Implementation and Experiments

In this section we present our experimental findings for 2-Club. We implemented a search tree algorithm, multiple data reduction rules, and a Turing kernelization

and we combined them in several ways to get multiple exact algorithms. We then tested them on randomly created instances as well as on a collection of real-world instances taken from the 2nd & 10th DIMACS challenge [17, 18]. We compare our findings to the performance of the Gurobi 5.62 [1] solver running the integer linear programming formulation of Bourjolly et al. [13] and an implementation of Chang et al. [15] of the same basic search tree algorithm.[3] In the following we first describe the algorithms and their variants, the instances and the benchmark setting, and, finally, we describe our experimental findings.

5.1 Implemented Algorithms

Search tree-based algorithm. We implemented the following search tree algorithm, briefly denoted by `ST`, 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 $|N_2(v)|$ (number of vertices within distance two) 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. The 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 algorithm was introduced by Bourjolly et al. [13]. Together with a data structure that maintains the two-neighborhood of all vertices under vertex deletions, this search tree algorithm was implemented and evaluated by Chang et al. [15]. The algorithm has running time $\mathcal{O}(\alpha^n \cdot nm)$ where α is the golden ratio [15]. A different analysis shows that this algorithm runs in $\mathcal{O}(2^{k'} \cdot nm)$ time for the dual parameter $k' = n - \ell$ (if branching is aborted if more than k' vertices have been removed) [36]. By Corollary 3, the search tree size measured by k' cannot be improved unless the SETH fails.

Our implementation of the search tree algorithm `ST` is accelerated in each branching step by the extensive application of the following data reduction rules. We describe the rules in descending order of observed effectiveness. Herein, let $G = (V, E)$ be the graph of the current branching step.

R1 *Vertex Cover Rule:* Let $G' = (V, E')$ be the graph where two vertices are adjacent if and only if they have distance at least three in G. Observe that the size of a minimum vertex cover of G' is a lower bound on the number of vertex deletions that have to be performed to transform G into 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.[4]

[3] The source code and MAKE file of the C++ program of Chang et al. [15] is publicly available (accessed March 2013); we compiled using gcc compiler version 4.7.2 (Debian 4.7.2-5).

[4] We use the well-known 2-approximation that recursively chooses an arbitrary edge $\{u, v\}$, takes u and v into the vertex cover, and deletes all edges incident with u and v.

R2 *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.

R3 *Common neighbors of marked vertices:* If there are two non-adjacent marked vertices with only one common neighbor v, then mark v.

R4 *Degree-one vertices:* Remove each vertex v that has degree one. If v is marked, then abort this branch.

The correctness of Rules R1–R3 is obvious. Rule R4 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 R4 is a special case of Rule 6 on p. 8).

Turing Kernelization. We implemented the Turing kernelization which was introduced by Schäfer et al. [36]. Therein, the basic observation is that for any 2-club S in a graph $G = (V, E)$ it holds that $S \subseteq N_2[v]$ for all $v \in S$. Moreover, after applying Rule 5 (in Subsection 2.2) in advance, $|N_2[v]| \leq \ell^2$. We say that $N_2[v]$ is the Turing kernel for vertex v.

Later on, running an algorithm for 2-Club together with Turing kernelization means that in any step we choose a vertex v, mark v, and run the algorithm on the graph induced by $N_2[v]$. We update the current lower bound ℓ', delete v, delete all vertices u where $|N_2[u]| \leq \ell'$ (they cannot be contained in any 2-club larger than ℓ'), and proceed by choosing the next Turing kernel.[5] We implemented three different strategies to choose the vertex v among all vertices: i) v has minimum degree (`DEG`), ii) a feedback edge set in $G[N_2[v]]$ is of minimum size (`FES`), and iii) the size of $N_2[v]$ is minimum (`N2`).

Note that using Turing kernelization with strategy `N2` 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.

5.2 Results

Setting and Algorithm Variants. We ran all our experiments on an Intel(R) Xeon(R) CPU E5-1620 3.60 GHz machine with 64 GB main memory under the Debian GNU/Linux 7.0 operating system. The program is implemented in Java and runs under the OpenJDK runtime environment in version 1.7.0_03. 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 2nd and 10th DIMACS challenge [17, 18]. We use the name scheme

[5] Deleting all vertices whose two-neighborhood is of size at most the current lower bound can be viewed as a data reduction rule which strengthens R4.

Table 1: Experimental results on random instances with 0.15 density. For each combination of a, b, and n we created 100 instances by the random graph generator proposed by Gendreau et al. [21]. Correspondingly, all other entries namely, m (# edges), Δ_G (maximum degree), $\varnothing_G$ (avg. degree), h-index, 2-club (size of the largest 2-club), and the time in seconds for the solvers `ST` and `CST` are the averages over all these 100 instances.

a	b	n	m	Δ_G	$\varnothing_G$	h-index	2-club	`ST`	`CST`
0	0.3	140	1453	39.3	20.8	27.4	71	0.77	1.0
		150	1668	41.9	22.2	29.4	79.5	1.3	1.9
		160	1899	45.0	23.7	31.2	89.8	1.2	2.1
							$\sum$	**3.27**	**5.0**
0.05	0.25	140	1456	36.7	20.8	26.0	59.5	6.1	7.8
		150	1671	38.9	22.3	27.9	68.1	10.9	15.4
		160	1913	41.3	23.9	29.7	78.5	18.9	31.1
							$\sum$	**35.9**	**54.3**
0.1	0.2	140	1456	33.9	20.8	25.0	50.5	23.9	28.8
		150	1676	35.8	22.3	26.8	57.6	66.5	90
		160	1911	37.7	23.9	28.5	64.7	194	306
							$\sum$	**284**	**425**
0.15	0.15	140	1455	32.6	20.8	24.6	46.1	38.3	44.4
		150	1679	34.7	22.4	26.4	52.9	109	140
		160	1912	36.5	23.9	28.0	59.7	406	628
							$\sum$	**553**	**812**

`SOLVER-STRATEGY` to denote the different variants of the solver, that is, `SOLVER` is one of {`ST`,`ILP`,`CST`} where `ILP` refers to the Gurobi solver running the ILP proposed by Bourjolly et al. [13] and `CST` refers to the search tree implementation of Chang et al. [15]. `STRATEGY` indicates that Turing kernelization is used (if not, it is just omitted), and it states one of the strategies {`DEG`,`FES`,`N2`} used therein. For example, `ST-FES` denotes the solver running Turing kernelization with the strategy `FES` and uses our implementation of the search tree algorithm to solve each Turing kernel.

Random Instances. As in previous experimental evaluations [15, 31], we use the random graph generator due to Gendreau et al. [21] 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 1 summarizes the performance of `ST` and those of `CST` on instances with density 0.15 (see Table 5, appendix, for a full list with different densities). As first observed by Bourjolly et al. [13], density 0.15, $a = 0.15$, and $b = 0.15$ produces the hardest instances. On average the `ST`-solver solves instances of this type for $n = 160$ within ≈ 400 s whereas `CST` needs ≈ 600 s. Note that Mahdavi

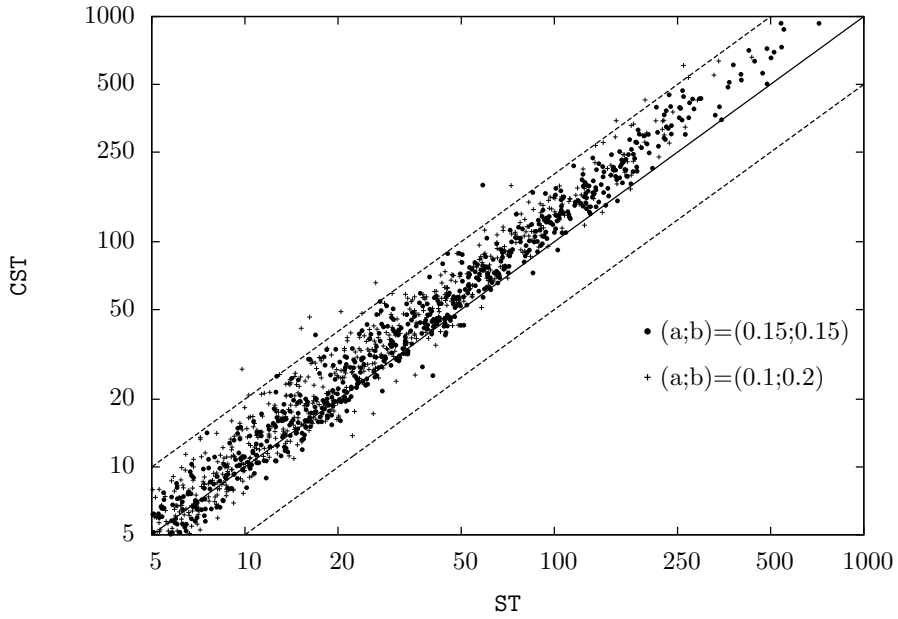

Figure 4: For each combination of $(a, b) = (0.15, 0.15), (0.1, 0.2)$ and $n = 100, 105, \ldots, 155$ we created 100 instances by the random graph generator proposed by Gendreau et al. [21] and depicted the corresponding running times (in seconds) of `ST` and `CST`. The solid line depicts all values where both algorithms would have the same running time. Our `ST`-algorithm outperforms `CST` on most instances; rare exceptions can be found for running times less than 50 s. Notably, the greater the running times get the more `ST` dominates `CST`. However, the performance increase is within a factor of two for most instances (marked by the dashed lines).

and Balasundaram [31] needed about one hour on instances with $n = 150$. A per-instance comparison of `ST` and `CST` (see Figure 4) shows that `ST` achieves a constant speedup of ≈ 1.5 for most of the instances and it is always faster on instances that take more than 250 s for both solvers. We observed that the key point for the good behavior of our algorithm on random instances with density 0.15 is the *Vertex Cover Rule* that allows quite frequently to prune the search tree (see Table 4, appendix, for an extensive comparison of the performance of `ST` with and without data reduction rules). As can be observed in the full list of results, combining the `ST` solver with Turing kernelization does slightly decrease (about 2 %) its performance but still outperforms other combinations, e. g. `ILP` with Turing kernelization. This effect is in stark contrast to the behavior on real-world instances, as there Turing kernelization yields a dramatic performance increase. This is because the largest 2-club contains about 50 % of the vertices, hence, the beneficial decrease in the graph size obtained by

Table 2: Experimental results on random instances with 0.15 density, comparing ILP-based solvers with ST. For each combination of a, b, and n we created 100 instances by the random graph generator proposed by Gendreau et al. [21]. Correspondingly, all other entries namely, m (# edges), Δ_G (maximum degree), 2-Club (size of the largest 2-club), and the time in seconds for the solvers (last five columns) are the average over all these 100 instance.

a	b	n	m	Δ_G	2-club	ST	ILP	ILP-DEG	ILP-N2	ILP-FES
0	0.3	100	743	28.9	41.6	0.1	1.24	1.25	1.25	1.26
		120	1 070	34.4	55.1	0.3	2.2	2.22	2.2	2.21
		130	1 265	37.3	63.7	0.5	2.34	2.32	2.32	2.34
0.05	0.25	100	744	27.1	35.5	0.25	2.32	2.34	2.34	2.35
		120	1 061	31.5	45.6	1.25	10.7	10.9	10.9	10.9
		130	1 252	34.1	52.4	2.71	20.9	20.7	20.6	20.6
0.1	0.2	100	744	25	31.6	0.41	3.79	4.04	4.04	4.04
		120	1 077	29.7	40.6	3.12	42.3	40.7	40.6	40.6
		130	1 255	31.7	44.4	8.66	103.5	98.4	100.3	98.3
0.15	0.15	100	743	24.2	29.6	0.56	7.17	7.31	7.31	7.31
		120	1 068	28.1	37.3	4.16	65.9	64.7	64.6	64.6
		130	1 261	30.4	42.7	14.0	138.2	137.4	139.4	137.2
					$\sum$	36.0	400.6	392.3	395.9	391.7

Turing kernelization is dominated by the time computing it.

Finally, we would like to emphasize that the ILP-solver, as the only program, makes use of the four cores of the processor (all other programs set up only a single thread). However, even when combining the ILP-solver with Turing kernelization it is, against our a-priori intuition, significantly (about 10 times) slower than the search tree-based algorithms ST and CST (see Table 2 for a comparison of its performance on instances with density 0.15). Because of the dramatic performance leak compared to ST and since the results in Table 2 suggest that the ILP-running time exponentially depends on n, we did not perform any further benchmarks for it on random instances.

Real-world Instances. We considered real-world data from the 2nd & 10th DIMACS challenge [17, 18]. To investigate the usefulness of 2-Club as natural clique relaxation concept, we ran our algorithm on instances from the *clustering* category [17]; a standard benchmark for graph clustering and community detection algorithms. To test our algorithm on large scale social network graphs we ran it on graphs from the *co-author and citation* category [17]. These graphs were obtained by the co-author relationship or the citation relation among authors listed in the DBLP and Citeseer database. Moreover, we performed studies on the instances from the clique category for the 2nd DIMACS challenge [18]. In addition to the DIMACS instances, we created a further DBLP coauthor graph,

Table 3: Experimental results on instances from the DIMACS implementation challenges [17, 18]. The first column denotes the name of the instances, n the number of vertices, m the number of edges, Δ_G the maximum degree, and 2-Club denotes the size of the largest 2-club. The last six columns contain for each solver the required time in seconds and "#TK" denotes the number of Turing kernels needed to solve.

name	$n/10^3$	$m/10^3$	Δ_G	2-club	ST-DEG		ST-FES		ILP-DEG	
					time	#TK	time	#TK	time	#TK
10th DIMACS clustering										
email	1.1	5.5	71	72	42.8	171	83.2	234	70.5	162
hep-th	8.4	15.6	50	51	0.94	0	0.94	0	0.93	0
netscience	1.6	2.7	34	35	0.03	0	0.03	0	0.03	0
PGPgiantcompo	10.7	24.3	205	206	1.14	0	1.15	0	1.14	0
power	4.9	6.6	19	20	0.25	0	0.25	0	0.24	0
10th DIMACS co-author citation										
citationCiteseer	268.5	1 156.6	1 318	1 319	79.3	0	78.2	0	78.7	0
coAuthorsCiteseer	227.3	814.1	1 372	1 373	49.4	0	49.5	0	48.8	0
coAuthorsDBLP	299.0	977.7	336	337	85	0	85.2	0	86.7	0
graph_thres_01	715.6	2 512.0	804	805	294	0	275	0	292	0

which is the largest instance in our experiments (dblp_thres_1). Table 3 shows the results (see Table 6, appendix, for a full list).

Since both, ST and CST, rely on space-consuming data structures to quickly maintain vertex deletions, they are not suitable for graphs with more than ≈1000 vertices. Hence, we performed all tests with solvers where Turing kernelization is applied. We observed that, since the average degree in real-world graphs is small, Turing kernelization typically produces small graphs enabling the corresponding solvers to solve them quickly. More specifically, in its fastest variant, namely ST-DEG, our search tree algorithm together with Turing kernelization solves all but one instance from the clustering category within 10 seconds.[6] This is a significant performance increase in comparison to Mahdavi and Balasundaram [31] who needed up to 70 min 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 1.5 min (in the preliminary version we needed about 30 min). Interestingly, Turing kernelization with the DEG-strategy turned out to be the most effective one. For example, on the email graph from the clustering category, the DEG-strategy yields a speedup factor of two compared to the other strategies. We conjecture that, since the minimum degree vertex v forming the Turing kernel $N_2[v]$ can be marked, this

[6] The version of the ST-DEG-solver presented in previous work [23] was slightly faster on these instances. This moderate worsening is due to some heuristic improvements leading to significant accelerations on instances from the co-author citation category.

allows the solver to quickly mark a majority of v's neighbors as well, because otherwise v would not have a connection to its rather large two-neighborhood. However, a more careful analysis of this phenomenon is necessary. Not to our surprise after the results on random data, combining `ILP` with Turing kernelization is not competitive with the combination of `ST` and Turing kernelization.

We also performed initial experiments with the combination of `CST` and Turing kernelization. However, its performance is roughly similar to the results for `ST` with Turing kernelization because instances formed by the Turing kernels are only moderately hard and thus they are not suitable to spot the difference between the corresponding solvers. Furthermore, our Turing kernelization is implemented in Java and thus has to write an input file for each Turing kernel in order to invoke the C++ program `CST`. Due to this we did not perform a further systematic study here.

On a majority of the real-world instances we further observed the unexpected behavior that the largest 2-club is "just" a maximum-degree vertex together with its neighbors.[7] From this, 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 (with degree 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.[8]

Altogether, our experiments demonstrate that for the hardest randomly created instances as well as for huge real-world instances our implementation of the search tree algorithm combined with Turing kernelization significantly outperforms previous implementations.

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. We further gave a direct algorithm for the parameter treewidth of G. Complementing these positive

[7] This seems to be the reason why "deleting vertices with too small two neighborhood" is quite successful in Turing kernelization. Indeed, together with the (heuristically chosen) rule that Turing kernelization is disabled as soon as it lowers to graph size by less than 50 %, it causes the effect that the number of Turing kernels that need to be solved by the corresponding solver in {`ST`,`ILP`,`CST`} is zero for most instances.

[8] This implies that the h-index of the real-world instances is low and thus a promising parameter. Unfortunately, 2-Club is W[1]-hard with respect to the h-index [23].

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 $k' = n - \ell$. On the practical side, we provide the currently best implementation for 2-Club which, as demonstrated in the experiments, solves 2-Club in up to five minutes 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 [36]? Are there "stronger parameters" [28] 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 [31, 35].

References

[1] Gurobi 5.6. Software, 2014.

[2] R. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3(1):113–126, 1973.

[3] M. T. Almeida and F. D. Carvalho. Integer models and upper bounds for the 3-club problem. *Networks*, 60(3):155–166, 2012.

[4] 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.

[5] B. Balasundaram, S. Butenko, and S. Trukhanovzu. Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization*, 10 (1):23–39, 2005.

[6] S. Böcker and J. Baumbach. Cluster editing. In *Proc. 9th CIE*, volume 7921 of *LNCS*, pages 33–44. Springer, 2013.

[7] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996. .

[8] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.

[9] 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.

[10] 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.

[11] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, 412(35):4570–4578, 2011.

[12] J.-M. Bourjolly, G. Laporte, and G. Pesant. Heuristics for finding k-clubs

in an undirected graph. *Computers & Operations Research*, 27(6):559–569, 2000.

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

[14] F. D. Carvalho and M. T. Almeida. Upper bounds and heuristics for the 2-club problem. *European Journal of Operational Research*, 210(3):489–494, 2011.

[15] M.-S. Chang, L.-J. Hung, C.-R. Lin, and P.-C. Su. Finding large k-clubs in undirected graphs. *Computing*, 95(9):739–758, 2013.

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

[17] DIMACS'12. Graph partitioning and graph clustering. 10th DIMACS implementation challenge, 2012. URL http://www.cc.gatech.edu/dimacs10/. Accessed April 2012.

[18] DIMACS'93. NP-hard problems: Maximum clique, graph coloring, and satisfiability. 2nd DIMACS implementation challenge, 1993. URL http://dimacs.rutgers.edu/Challenges/. Accessed November 2008.

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

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

[21] M. Gendreau, P. Soriano, and L. Salvail. Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research*, 41 (4):385–403, 1993.

[22] P. A. Golovach, P. Heggernes, D. Kratsch, and A. Rafiey. Finding clubs in graph classes. *Discrete Applied Mathematics*, 174:57–65, 2014.

[23] S. Hartung, C. Komusiewicz, and A. Nichterlein. Parameterized algorithmics and computational experiments for finding 2-clubs. In *Proc. 7th IPEC*, volume 7535 of *LNCS*, pages 231–241. Springer, 2012.

[24] S. Hartung, C. Komusiewicz, and A. Nichterlein. On structural parameterizations for the 2-club problem. In *Proc. 39th SOFSEM*, volume 7535 of *LNCS*, pages 231–241. Springer, 2013.

[25] S. Hartung, C. Komusiewicz, and A. Nichterlein. On structural parameterizations for the 2-club problem. Number arXiv:1305.3735v1, 2013.

[26] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4): 512–530, 2001.

[27] T. Kloks. *Treewidth. Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994. ISBN 978-3-540-58356-1.

[28] C. Komusiewicz and R. Niedermeier. New races in parameterized algorithmics. In *Proc. 37th MFCS*, volume 7464 of *LNCS*, pages 19–30. Springer, 2012.

[29] H. Liu, P. Zhang, and D. Zhu. On editing graphs into 2-club clusters. In *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management (FAW-AAIM 2012)*, volume 7285 of *LNCS*, pages 235–246.

Springer, 2012.
[30] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.
[31] F. Mahdavi and B. Balasundaram. On inclusionwise maximal and maximum cardinality k-clubs in graphs. *Discrete Optimization*, 9:84–97, 2012.
[32] 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.
[33] R. J. Mokken. Cliques, Clubs and Clans. *Quality and Quantity*, 13:161–173, 1979.
[34] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
[35] 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.
[36] A. Schäfer, C. Komusiewicz, H. Moser, and R. Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, 6(5):883–891, 2012.
[37] A. Schäfer. Exact algorithms for s-club finding and related problems. Diploma thesis, Friedrich-Schiller-Universität Jena, 2009.
[38] S. B. Seidman and B. L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6:139–154, 1978.
[39] A. van Zuylen and D. P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. *Mathematics of Operations Research*, 34(3):594–620, 2009.

A Appendix: Full experimental results

Table 4: Experimental results on random instances with 0.15 density. For each combination of a, b, and n we created 100 instances by the random graph generator proposed by Gendreau et al. [21]. Correspondingly, all other entries, for example m (# edges) and Δ_G (maximum degree), are the averages over all these 100 instances. The table provides the time (in seconds), number of recursive steps (RS), and the max. search tree depth (MD) for our ST-solver with (default) and without data reduction rules (DR).

a	b	n	m	Δ_G	ST			ST without DR		
					time	RS	MD	time	RS	MD
0	0.3	75	416	22	0.02	526	48.8	0.05	4 299	50.8
0	0.3	100	741	28.8	0.10	2 272	65.8	0.77	52 259	66.5
0	0.3	110	889	31.3	0.19	3 922	72.2	2.62	160 974	72.6
0	0.3	120	1 065	33.8	0.35	6 906	78.2	8.48	463 762	78.6
0	0.3	130	1 266	37.2	0.52	9 386	80.9	22.5	1 061 549	81.0
				$\sum$	**1.18**	**23 012**	**346**	**34.42**	**1 742 843**	**349**
0.05	0.25	75	416	20.7	0.03	810	50.1	0.08	6 470	52.9
0.05	0.25	100	746	27	0.25	5 724	69.0	2.06	151 652	70.4
0.05	0.25	110	899	29.1	0.55	11 452	77.1	7.88	511 986	77.8
0.05	0.25	120	1 072	31.6	1.19	23 569	84.0	34.1	1 997 022	84.9
0.05	0.25	130	1 257	34.2	2.97	53 754	91.3	173.0	9 072 472	92.0
				$\sum$	**4.99**	**95 309**	**371**	**217.1**	**11 739 602**	**378**
0.1	0.2	75	416	19.3	0.04	1 032	50.4	0.09	7 498	54.4
0.1	0.2	100	741	25.1	0.39	8 881	71.3	3.08	234 315	73.5
0.1	0.2	110	901	27.3	1.16	24 830	79.2	17.5	1 243 749	81.3
0.1	0.2	120	1 065	29.4	3.05	58 953	87.3	84.5	5 410 285	89.2
0.1	0.2	130	1 257	31.5	7.82	135 258	95.4	442.0	24 729 684	96.9
				$\sum$	**12.5**	**228 954**	**384**	**547.2**	**31 625 531**	**395**
0.15	0.15	75	414	19	0.04	1 087	51.2	0.09	7 774	54.9
0.15	0.15	100	742	24.4	0.50	11 445	71.8	3.67	287 616	74.5
0.15	0.15	110	901	26.5	1.49	31 109	79.4	20.8	1 487 177	82.2
0.15	0.15	120	1 075	28.4	4.53	86 955	88.0	139.0	9 029 970	90.3
0.15	0.15	130	1 256	30.5	12.8	218 125	96.3	879.0	51 068 817	98.2
				$\sum$	**19.4**	**348 721**	**387**	**1 043**	**61 881 354**	**400**

Experimental Results for Random Instances

dens.	a	b	n	m	Δ_G	$\varnothing_G$	h-index	2-club	ST	ST-DEG	ST-N2	ST-FES	CST
0.05	0	0.1	180	812	20.7	9.0	14.7	21.7	0.26	2.4	3.5	3.8	0.15
			200	984	22.5	9.8	16.1	23.5	0.37	3.6	5.1	5.6	0.21
	0.05	0.05	180	810	17.9	9.0	13.4	18.9	0.28	3.6	4.2	4.7	0.14
			200	994	19.5	9.9	14.6	20.5	0.39	5.0	6.7	7.4	0.21
								$\sum$	**1.3**	**14.6**	**19.5**	**21.5**	**0.72**
0.1	0	0.2	170	1427	33.8	16.8	23.6	35.1	3.5	3.6	3.7	3.7	3.1
			180	1615	35.7	17.9	25.3	37.2	7.1	7.5	7.6	7.6	7.3
			190	1801	38.0	19.0	26.7	39.7	14.6	15.0	15.1	15.1	16.1
			200	1991	39.4	19.9	28.1	41.6	29.2	29.9	29.9	30.0	30.4
	0.05	0.15	170	1435	30.3	16.9	22.0	31.4	2.5	2.6	2.6	2.7	2.1
			180	1612	31.7	17.9	23.4	32.7	5.0	5.1	5.2	5.2	4.2
			190	1802	33.1	19.0	24.5	34.2	9.7	9.9	9.9	9.9	8.8
			200	1980	34.8	19.8	25.7	35.8	15.3	15.6	15.6	15.6	14.8
	0.1	0.1	170	1433	28.1	16.9	21.3	29.2	2.7	2.7	2.7	2.7	1.7
			180	1611	29.7	17.9	22.5	30.7	5.2	5.3	5.3	5.3	3.5
			190	1797	30.7	18.9	23.7	31.7	10.0	10.2	10.2	10.2	7.3
			200	1992	32.3	19.9	24.9	33.3	17.1	17.5	17.5	17.5	13.5
								$\sum$	**122**	**125**	**125**	**126**	**113**
0.15	0	0.3	130	1265	37.3	19.5	25.8	63.7	0.50	0.50	0.51	0.52	0.61
			140	1453	39.3	20.8	27.4	71.0	0.77	0.77	0.78	0.79	1.0
			150	1668	41.9	22.2	29.4	79.5	1.3	1.3	1.3	1.3	1.9
			155	1782	43.6	23.0	30.2	84.8	1.3	1.4	1.4	1.4	2.1
			160	1899	45.0	23.7	31.2	89.8	1.2	1.2	1.2	1.3	2.1
	0.05	0.25	130	1252	34.1	19.3	24.3	52.4	2.7	2.7	2.8	2.8	3.0
			140	1456	36.7	20.8	26.0	59.5	6.1	6.1	6.2	6.2	7.8
			150	1671	38.9	22.3	27.9	68.1	10.9	11.0	11.0	11.0	15.4
			155	1796	39.8	23.2	28.9	73.6	14.8	14.9	14.9	14.9	22.1
			160	1913	41.3	23.9	29.7	78.5	18.9	19.0	19.0	19.0	31.1
	0.1	0.2	130	1255	31.7	19.3	23.3	44.4	8.7	8.7	8.8	8.8	9.0
			140	1456	33.9	20.8	25.0	50.5	23.9	24.2	24.3	24.3	28.8
			150	1676	35.8	22.3	26.8	57.6	66.5	67.1	67.1	67.2	90.0
			155	1790	36.8	23.1	27.7	60.7	114	115	115	115	173
			160	1911	37.7	23.9	28.5	64.7	194	196	196	196	306

dens.	a	b	n	m	Δ_G	$\varnothing_G$	h-index	2-club	ST	ST-DEG	ST-N2	ST-FES	CST
	0.15	0.15	130	1261	30.4	19.4	23.2	42.7	14.0	14.2	14.2	14.2	14.2
			140	1455	32.6	20.8	24.6	46.1	38.3	38.8	38.8	38.8	44.4
			150	1679	34.7	22.4	26.4	52.9	109	110	110	110	140
			155	1792	35.6	23.1	27.2	55.3	243	246	246	246	346
			160	1912	36.5	23.9	28.0	59.7	406	409	410	410	628
			165	2031	37.9	24.6	28.9	63.1	581	586	587	586	763
								$\sum$	**1858**	**1875**	**1876**	**1876**	**2629**
0.2	0.05	0.35	200	3988	66.1	39.9	47.7	186.2	0.02	0.03	0.05	0.10	0.06
	0.1	0.3	200	3988	61.6	39.9	46.0	191.1	0.02	0.03	0.05	0.09	0.05
	0.15	0.25	200	3980	58.3	39.8	44.5	194.4	0.02	0.03	0.04	0.09	0.04
	0.2	0.2	200	3976	55.6	39.8	44.1	195.7	0.02	0.03	0.04	0.09	0.03
								$\sum$	**0.07**	**0.11**	**0.18**	**0.37**	**0.19**

Table 5: Full list of experimental results on random instances with 0.05, 0.1, 0.15, and 0.2 density. For each combination of a, b, and n we created 100 instances by the random graph generator proposed by Gendreau et al. [21]. Correspondingly, all other entries namely, m (# edges), Δ_G (maximum degree), $\varnothing_G$ (avg. degree), h-index, 2-Club (size of the largest 2-club), and the time in seconds for the solvers `ST`, `ST-DEG`, `ST-N2`,`ST-FES`, `CST` are the average over all these 100 instance.

Experimental Results for Real-World Instances

name	n	m	Δ_G	$\varnothing_G$	h-Index	2-club	ST-Deg		ST-N2		ST-FES		ILP-Deg		ILP-N2		ILP-FES	
							time	#TK	time	#TK	time	#TK	time	#TK	time	#TK	time	#TK
10th DIMACS clustering																		
adjnoun	112	425	49	7	13	50	0.01	0	0.01	0	0.00	0	0.00	0	0.01	0	0.00	0
celegans_metabolic	453	2025	237	8	23	238	0.10	0	0.10	0	0.11	0	0.11	0	0.10	0	0.11	0
celegansneural	297	3520	284	23	34	285	0.03	0	0.03	0	0.03	0	0.03	0	0.03	0	0.03	0
dolphins	62	159	12	5	9	13	0.00	0	0.00	0	0.00	0	0.01	0	0.01	0	0.01	0
email	1133	5451	71	9	33	72	42.8	171	110	293	83.2	234	70.5	162	140	293	112	234
football	115	613	12	10	12	16	0.07	5	0.38	66	0.46	68	2.73	4	0.77	66	0.87	68
hep-th	8361	15751	50	3	27	51	0.94	0	0.93	0	0.94	0	0.93	0	0.93	0	0.95	0
jazz	198	2742	100	27	41	103	0.05	0	0.05	0	0.07	0	0.23	0	0.24	0	0.26	0
karate	34	78	17	4	6	18	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0
netscience	1589	2742	34	3	19	35	0.03	0	0.03	0	0.03	0	0.03	0	0.03	0	0.03	0
PGPgiantcompo	10680	24316	205	4	52	206	1.14	0	1.15	0	1.15	0	1.14	0	1.16	0	1.15	0
polblogs	1490	16715	351	22	87	352	9.78	4	44.9	42	21.0	12						
polbooks	105	441	25	8	15	28	0.01	3	0.03	10	0.07	17	0.06	5	0.08	11	0.14	18
power	4941	6594	19	2	12	20	0.25	0	0.25	0	0.25	0	0.24	0	0.24	0	0.25	0
10th DIMACS co-author citation																		
citationCiteseer	268495	1156647	1318	8	209	1319	79.3	0	82.5	0	78.2	0	78.7	0	78.5	0	82.0	0
coAuthorsCiteseer	227320	814134	1372	7	114	1373	49.4	0	49.8	0	49.5	0	48.8	0	48.7	0	49.6	0
coAuthorsDBLP	299067	977676	336	6	132	337	85.0	0	91.6	0	85.2	0	86.7	0	85.0	0	85.3	0
graph_thres_01	715633	2511988	804	7	208	805	294	0	284	0	275	0	292	0	287	0	277	0
graph_thres_02	282831	640697	201	4	96	202	62.4	0	60.6	0	59.5	0	61.3	0	59.9	0	59.2	0
graph_thres_03	167006	293796	123	3	62	124	25.7	0	25.6	0	25.5	0	25.0	0	25.1	0	25.2	0
graph_thres_04	112949	168524	88	2	46	89	14.2	0	14.3	0	14.3	0	13.9	0	13.9	0	13.9	0
graph_thres_05	81519	107831	71	2	38	72	8.93	0	8.91	0	8.97	0	8.74	0	8.70	0	8.77	0
2nd DIMACS clique																		
brock200_2	200	9876	114	98	99	200	0.04	0	0.10	0	0.20	0	0.92	0	0.99	0	1.08	0
brock200_4	200	13089	147	130	128	200	0.03	0	0.15	0	0.26	0	0.86	0	0.97	0	1.09	0
brock400_2	400	59786	328	298	294	400	0.26	0	1.95	0	4.91	0	5.90	0	7.64	0	9.62	0
brock400_4	400	59765	326	298	294	400	0.28	0	1.98	0	4.83	0	4.78	0	6.62	0	8.49	0

name	n	m	Δ_G	$\varnothing_G$	h-Index	2-club	ST-Deg		ST-N2		ST-FES		ILP-Deg		ILP-N2		ILP-FES	
							time	#TK	time	#TK	time	#TK	time	#TK	time	#TK	time	#TK
brock800_2	800	208166	566	520	516	800	2.17	0	15.1	0	28.3	0	96.8	0	110	0	121	0
brock800_4	800	207643	565	519	514	800	2.14	0	14.9	0	28.0	0	83.1	0	96.0	0	107	0
C125.9	125	6963	119	111	107	125	0.01	0	0.06	0	0.09	0	0.08	0	0.13	0	0.16	0
C250.9	250	27984	236	223	219	250	0.09	0	0.57	0	0.95	0	0.81	0	1.24	0	1.62	0
C500.9	500	112332	468	449	441	500	0.63	0	9.32	0	14.7	0	5.55	0	13.8	0	20.0	0
DSJC1000.5	1000	249826	551	499	500	1000	2.24	0	14.4	0	30.4	0	181	0	194	0	208	0
DSJC500.5	500	62624	286	250	250	500	0.38	0	1.99	0	5.63	0	15.7	0	17.5	0	19.9	0
gen200_p0.9_44	200	17910	190	179	173	200	0.04	0	0.24	0	0.37	0	0.39	0	0.60	0	0.74	0
gen200_p0.9_55	200	17910	190	179	173	200	0.04	0	0.24	0	0.37	0	0.40	0	0.59	0	0.74	0
gen400_p0.9_55	400	71820	375	359	347	400	0.31	0	3.42	0	6.60	0	2.70	0	5.92	0	8.09	0
gen400_p0.9_65	400	71820	378	359	349	400	0.32	0	3.42	0	6.52	0	2.68	0	5.85	0	8.08	0
gen400_p0.9_75	400	71820	380	359	349	400	0.31	0	3.38	0	6.48	0	2.68	0	5.92	0	8.13	0
hamming8-4	256	20864	163	163	163	256	0.07	0	0.29	0	0.51	0	2.25	0	2.30	0	2.72	0
keller4	171	9435	124	110	106	171	0.02	0	0.09	0	0.16	0	0.68	0	0.74	0	0.82	0
keller5	776	225990	638	582	565	776	2.68	0	18.7	0	31.4	0	74.7	0	91.3	0	103	0
MANN_a27	378	70551	374	373	364	378	0.30	0	3.71	0	7.39	0	0.48	0	3.58	0	6.81	0
p_hat1500-1(1)	1500	284923	614	379	456	1500	26.8	0	34.6	0	55.0	0	288	0	296	0	314	0
p_hat300-1	300	10933	132	72	90	299	0.06	0	0.13	0	0.30	0	1.50	0	1.47	0	1.73	0
p_hat300-2	300	21928	229	146	148	300	0.09	0	0.41	0	0.65	0	2.76	0	3.01	0	3.53	0
p_hat300-3	300	33390	267	222	208	300	0.12	0	0.67	0	1.37	0	1.88	0	2.47	0	2.89	0
p_hat700-1	700	60999	286	174	207	700	0.74	0	1.61	0	4.88	0	21.7	0	22.7	0	25.2	0
p_hat700-2	700	121728	539	347	352	700	1.02	0	6.56	0	13.3	0	49.2	0	55.4	0	61.3	0
p_hat700-3	700	183010	627	522	486	700	1.70	0	14.5	0	25.4	0	33.7	0	46.7	0	56.9	0

Table 6: Full list of experimental results on instances from the DIMACS implementation challenges [17, 18]. The first column denotes the name of the instances, n the number of vertices, m the number of edges, Δ_G the maximum degree, $\varnothing_G$ the avg. degree, and 2-Club denotes the size of the largest 2-club. The last twelve columns contain for each solver the required time in seconds and "#TK" denotes the number of Turing kernels needed to solve.