

# Exact Algorithms for Finding Well-Connected 2-Clubs in Real-World Graphs: Theory and Experiments\*

Christian Komusiewicz<sup>1,†</sup>    André Nichterlein<sup>2</sup>    Rolf Niedermeier<sup>2</sup>  
Marten Picker<sup>2</sup>

<sup>1</sup> Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Germany,

<sup>2</sup> Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Germany,  
komusiewicz@informatik.uni-marburg.de    {andre.nichterlein,rolf.niedermeier}@tu-berlin.de

## Abstract

Finding large “cliquish” subgraphs is a central topic in graph mining and community detection. A popular clique relaxation are 2-clubs: instead of asking for subgraphs of diameter one (these are cliques), one asks for subgraphs of diameter at most two (these are 2-clubs). A drawback of the 2-club model is that it produces star-like hub-and-spoke structures as maximum-cardinality solutions. Hence, we study 2-clubs with the additional constraint to be well-connected. More specifically, we investigate the algorithmic complexity for three variants of well-connected 2-clubs, all established in the literature: robust, hereditary, and “connected” 2-clubs. Finding these more cohesive 2-clubs is NP-hard; nevertheless, we develop an exact combinatorial algorithm, extensively using efficient data reduction rules. Besides several theoretical insights we provide a number of empirical results based on an engineered implementation of our exact algorithm. In particular, the algorithm significantly outperforms existing algorithms on almost all (sparse) real-world graphs we considered.

## 1 Introduction

The  $s$ -club model, introduced by Mokken [28], is a well-established mathematical model for community mining in graphs [17]. An important special case is the 2-club model. In the corresponding algorithmic problem, the task is to compute a largest 2-club, that is, a maximum-cardinality vertex subset inducing a subgraph of diameter at most two. As a community model, 2-clubs have the drawback that they often form hub-and-spoke structures, that is, they often consist of one vertex that is adjacent to all other vertices of the community plus only few additional edges. Indeed, previous experimental work revealed that most maximum-cardinality 2-clubs in real-world graphs have exactly this hub-and-spoke structure and are thus fairly sparse [21]. This is essentially the opposite of cliques which provide a subgraph model with maximal (edge) density. In this sense, standard 2-clubs are far away from being cliquish. The drawback of cliques, however, is that they are too demanding with respect to density in order to be meaningful in many applications based on mining cohesive subgraphs [31, 24]. Hence, we study three 2-club models that exclude the low degree of connectivity that 2-clubs may have while still providing meaningful community abstractions.

---

\*Parts of this work are based on the last author’s master thesis [32]. Work started when all authors were with TU Berlin.

<sup>†</sup>CK was partially supported by the DFG, project MAGZ (KO 3669/4-1).

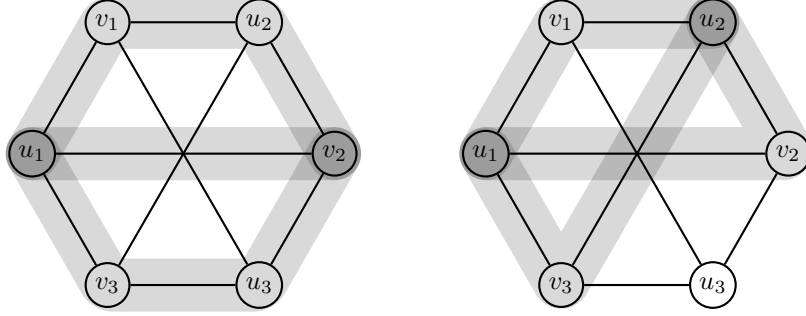


Figure 1: A complete bipartite graph  $K_{3,3}$  where the three internally vertex-disjoint paths from  $u_1$  to  $v_2$  (left side) and  $u_1$  to  $u_2$  (right side) are highlighted. Note that two of the paths on the left side are of length three. Thus, a  $K_{3,3}$  is a 1-robust 2-club but not a 2-robust 2-club. It is also a 2-hereditary 2-club as deleting two vertices leaves either a  $K_{1,3}$  or a  $K_{2,2}$ , both of which are 2-clubs. Finally, it is a 3-connected 2-club (the pictures above cover all cases up to symmetry).

**Three 2-Club Models** In the classic 2-club model, one relaxes the clique demand by allowing that not all vertices of the community are adjacent but that they may also have distance two or, equivalently, that every vertex pair is either adjacent or has a common neighbor. The problem with this simple relaxation is that there may be only one vertex which is the common neighbor of all nonadjacent vertices. Hence, removing this vertex may destroy connectivity of the 2-club. Next, we describe three “well-connected” variants of 2-clubs that have been proposed in the literature [31, 34]; these three variants will be the central subjects of our studies.

In the first variant of well-connected 2-clubs, one demands that every pair of adjacent vertices has  $t - 1$  common neighbors and every pair of nonadjacent vertices has  $t$  common neighbors.

**Definition 1** ([34]). *A vertex set  $S \subseteq V$  is a  $t$ -robust 2-club in a graph  $G = (V, E)$  if any pair of vertices in  $S$  is connected in  $G[S]$  by  $t$  internally vertex-disjoint paths of length at most two.*

In the second variant, one demands that the deletion of few vertices does not destroy the 2-club property.

**Definition 2** ([31]). *A vertex set  $S \subseteq V$  is a  $t$ -hereditary 2-club in a graph  $G = (V, E)$  if  $G[S \setminus U]$  is a 2-club for all  $U \subset S$  where  $|U| \leq t$ .*

In the third variant, one only demands that the deletion of few vertices does not destroy connectivity.

**Definition 3** ([31]). *A vertex set  $S \subseteq V$  is a  $t$ -connected 2-club in a graph  $G = (V, E)$  if it is a 2-club and  $G[S]$  is  $t$ -connected, that is,  $|S| > t$  and  $G[S \setminus U]$  is connected for all  $U \subseteq S$  where  $|U| < t$ .*

Observe the difference of one in the sizes of the sets  $U$  in **Definitions 2** and **3**. Refer to **Figure 1** for a comparison of the three variants of well-connected 2-clubs. Assuming 2-club sizes  $|S| \geq t + 2$ , it is easy to see that a  $t$ -robust 2-club is a  $(t - 1)$ -hereditary 2-club which again is a  $t$ -connected 2-club. Clearly, for all three models, fulfilling the connectivity demands for some  $t$  implies that they are also fulfilled for all  $t' \leq t$ . Hence, the size of maximum-cardinality 2-clubs is nonincreasing with increasing  $t$ .

Our three central computational problems, formulated as decision problems (which is more suitable for statements on computational complexity), are defined as follows.

$t$ -ROBUST /  $t$ -HEREDITARY /  $t$ -CONNECTED 2-CLUB

**Input:** An undirected graph  $G = (V, E)$  and nonnegative integers  $t$  and  $k$ .

**Question:** Does  $G$  contain a  $t$ -robust /  $t$ -hereditary /  $t$ -connected 2-club of size at least  $k$ ?

For brevity, when making general statements holding for all three variants, then we refer to them as  $t$ -well-connected 2-clubs.

**Related Work** Dense subgraph discovery or mining cohesive subgraphs is an active field in graph mining research [26, 31, 24]. Algorithms, complexity studies, and experiments for finding  $s$ -clubs and in particular 2-clubs play a significant role in this context [3, 7, 8, 9, 11, 20, 21, 22, 30, 33, 34, 36]. Veremyev and Boginski [34] introduced and motivated the concept of  $t$ -robustness and performed experiments for  $t = 2$  and  $t = 3$ , showing that the predicted communities for these settings are much larger than maximum cliques. Pattillo et al. [31] discussed several further variants of  $s$ -clubs.

For  $t$ -connected 2-clubs, the special case  $t = 2$  demands that the 2-club is biconnected. This case was studied by Yezereska et al. [36] who presented a branch-and-bound and a branch-and-cut algorithm for the problem of finding biconnected 2-clubs. Moreover, Yezereska et al. [36] showed that the problem of finding a largest biconnected 2-club is NP-hard.

Finally, Carvalho and Almeida [10] studied a variant of  $s$ -clubs where each vertex is demanded to be part of at least one triangle.

There is a large body of previous work that considers the standard 2-club model. In particular, several algorithmic approaches (heuristics, exact algorithms) have been proposed and examined for the NP-hard computational task [7] to find maximum-cardinality 2-clubs [7, 6, 8, 11, 21, 30, 33].

Of particular importance to our work is an exact algorithm for 2-CLUB that uses the following branching: If the graph contains two vertices  $u$  and  $v$  that have distance at least three, then branch into the two cases to remove  $u$  or  $v$  from  $G$ . This algorithm was first proposed by Bourjolly et al. [7]. Later, it was shown that this branching gives a fixed-parameter algorithm for the parameter  $|V| - k$  with running time  $O(2^{|V|-k}nm)$  [33] and an algorithm with running time  $O(\alpha^n)$  where  $\alpha < 1.62$  is the golden ratio [11]. A combination of this branching algorithm with data reduction and pruning rules achieves the so far best performance on random networks and on sparse real-world networks [21].

**Our Contributions** On a conceptual level, we provide an alternative characterization of  $t$ -hereditary 2-clubs (see Lemma 1). Moreover, we present a “unifying view” on all three considered models based on “compatible vertices” (Definition 4).

On the level of algorithm theory, to the best of our knowledge we provide the first (formal) NP-hardness proofs for  $t$ -ROBUST 2-CLUB and  $t$ -HEREDITARY 2-CLUB for all  $t \geq 1$ . The corresponding reduction also yields an exponential running time lower bound based on the Strong Exponential Time Hypothesis (SETH). Moreover, we generalize the NP-hardness result for  $t$ -CONNECTED 2-CLUB due to Yezereska et al. [36] by showing NP-hardness for all  $t \geq 1$  even when restricted to split graphs. On the positive side, we generalize the above-mentioned fixed-parameter algorithm for 2-CLUB parameterized by  $|V| - k$  [7, 11, 21, 33], to obtain a fixed-parameter algorithm for all three problem variants.

On an algorithm engineering and empirical level, under heavy use of efficient and effective data reduction rules and using the above-indicated “unified view” on the three problem variants, we develop an implementation of our fixed-parameter algorithm that outperforms existing implementations on almost all (large and sparse) real-world graphs we experimented with. Only for random graphs we are clearly beaten by previous implementations.

## 2 Preliminaries

We only consider simple undirected graphs  $G = (V, E)$  where  $V$  is the vertex set and  $E \subseteq \{\{u, v\} \mid u, v \in V\}$  is the edge set. Throughout this work, we use  $n := |V|$  and  $m := |E|$  to denote the number of vertices and the number of edges in the input graph  $G$ . A *subgraph*  $(V', E')$  of a graph  $G = (V, E)$  is a graph with  $V' \subseteq V$  and  $E' \subseteq E$  such that all edges in  $E'$  are between vertices in  $V'$ , i.e. a graph derived from  $G$  by deleting vertices and edges. For a graph  $G = (V, E)$  and a

subset  $S \subseteq V$  of vertices,  $G[S] := (S, \{\{u, v\} \in E \mid u, v \in S\})$  denotes the *subgraph induced by  $S$* . For  $v \in V$  we set  $G - v := G[V \setminus \{v\}]$ . For  $F \subseteq E$  we set  $G - F := (V, E \setminus F)$ .

A *path* is a sequence of vertices such that no vertex occurs twice and any two successive vertices in the path are adjacent. The *length of a path* is the number of edges along it. The *distance*  $d(u, v)$  between two vertices  $u$  and  $v$  is the length of a shortest path between these two vertices. If there is a path between two vertices, then these vertices are said to be *connected*, and *disconnected* otherwise. A graph is *connected* if every pair of its vertices is connected. A *connected component* of a graph is a maximal set of vertices which are pairwise connected. Unless stated otherwise, we assume that the input graphs are connected, as otherwise we can process each connected component separately. The *diameter* of a graph is the length of the longest shortest path in the graph, i.e. the maximum distance between any two vertices in the graph. The *open  $d$ -neighborhood*  $N_d(v)$  of a vertex  $v$  is the set of all vertices within distance  $d$  of  $v$  except  $v$  itself. The *closed  $d$ -neighborhood*  $N_d[v]$  is defined as  $N_d[v] := N_d(v) \cup \{v\}$ . We set  $N(v) := N_1(v)$ .

The *vertex connectivity* of a graph is the minimum number of vertices that has to be removed to disconnect the graph or make it trivial; *edge connectivity* is defined the same way over edges. The *edge density* of a graph is  $m/\binom{n}{2}$ , that is, the fraction of present edges compared to the maximum possible number of edges in a graph of  $n$  vertices. In the CLIQUE problem we are given an undirected graph  $G = (V, E)$  and an integer  $k$ , and we ask whether  $G$  contains a clique of size  $k$ , that is, a set  $S \subseteq V$  of size  $k$  such that every pair of vertices in  $S$  is adjacent.

Somewhat abusing notation we will use the terms  *$t$ -robust*,  *$t$ -hereditary*, and  *$t$ -connected 2-club* to refer to both the vertex sets and the subgraphs induced by them.

### 3 Structural Properties of Well-Connected 2-Clubs

Before studying the computational complexity of finding  *$t$ -well-connected 2-clubs*, we first derive some structural properties of the respective  *$t$ -well-connected 2-club models*. We begin with a simple characterization of  *$t$ -hereditary 2-clubs*.

**Lemma 1.** *A vertex set  $S \subseteq V$  is a  $t$ -hereditary 2-club in a graph  $G = (V, E)$  if and only if each nonadjacent pair of vertices of  $S$  has at least  $t + 1$  common neighbors in  $G[S]$ .*

*Proof.* ( $\Rightarrow$ ) We show the contraposition of this direction. Thus, assume that there are two nonadjacent vertices  $u$  and  $v$  in  $S$  that have at most  $t$  common neighbors. Now let  $U := N(u) \cap N(v)$  and observe that  $|U| \leq t$  by our assumption. The set  $S$  is not a 2-club in  $G[V \setminus U]$  because  $u$  and  $v$  are nonadjacent and have no common neighbors. Hence,  $S$  is not a  $t$ -hereditary 2-club.

( $\Leftarrow$ ) Assume that every pair of nonadjacent vertices in  $S$  has at least  $t + 1$  common neighbors. Now, let  $U \subseteq V$  be any vertex set of size at most  $t$ . Consider the graph  $G[V \setminus U]$ . We show that  $S$  is a 2-club in  $G[V \setminus U]$ . Let  $u$  and  $v$  be two vertices of  $S \setminus U$ . If  $u$  and  $v$  are adjacent, then the distance between them is one. Otherwise, the distance between them is two:  $u$  and  $v$  have at least  $t + 1$  common neighbors in  $G$  and one of them is not contained in  $U$ .  $\square$

With this lemma at hand, we can now define the three variants of well-connected 2-clubs based on a predicate over vertex pairs; these definitions are of central importance for specifying our algorithms.

**Definition 4.** *Two vertices  $v$  and  $w$  in a graph are called compatible*

- *for  $t$ -robust 2-clubs if they are adjacent and have at least  $t - 1$  common neighbors, or if they have at least  $t$  common neighbors,*
- *for  $t$ -hereditary 2-clubs if they are adjacent or if they have at least  $t + 1$  common neighbors,*
- *for  $t$ -connected 2-clubs if they are at distance at most two and are connected by at least  $t$  internally vertex-disjoint paths.*

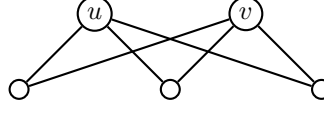


Figure 2: The two vertices  $u$  and  $v$  are compatible with respect to the 3-robust 2-club model but the three other vertices are pairwise not compatible. Thus, there is no 3-robust 2-club in the displayed graph.

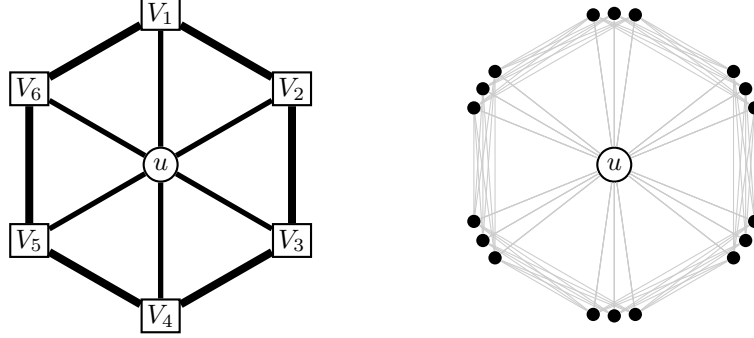


Figure 3: Left: A schematic graph where each of  $V_1, V_2, \dots, V_6$  represents a set of  $t$  vertices, for some arbitrary  $t \geq 1$ . Each edge  $\{v, w\}$  in the left graph represents a complete bipartite graph with the partite sets represented by  $v$  and  $w$ . Right: The actual graph for  $t = 3$ . The left graph is a 2-club since  $u$  is adjacent to all vertices. Furthermore, the graph is  $t$ -connected, that is, each pair of vertices is connected via  $t$  internally vertex-disjoint paths. Thus the graph is a  $t$ -connected 2-club. Deleting  $u$  results in a graph of diameter three. Hence, the graph is not a 1-hereditary 2-club and not a 2-robust 2-club.

Note that if there are two compatible vertices  $u$  and  $v$ , then this does not necessarily mean that in a graph  $G$  there is some  $t$ -well-connected 2-club  $S$  containing both  $u$  and  $v$ , see [Figure 2](#) for an example. [Definitions 1 to 4](#) and [Lemma 1](#) imply the following.

**Observation 1.** *A graph is a  $t$ -robust,  $t$ -hereditary, or  $t$ -connected 2-club if and only if every pair of vertices in  $G$  is compatible.*

[Definition 4](#) and [Observation 1](#) immediately give the following relation between the three concepts. Observe that there is an offset of 1 in the correspondence between  $t$ -robust or  $t$ -connected 2-clubs and  $t$ -hereditary 2-clubs.

**Observation 2.** *A  $t$ -robust 2-club is a  $(t - 1)$ -hereditary 2-club and a  $t$ -connected 2-club. A  $t$ -hereditary 2-club of size at least  $t + 2$  is a  $(t + 1)$ -connected 2-club.*

In [Figure 1](#) in [Section 1](#), we can already see that a complete bipartite  $K_{t,t}$  is a  $(t - 1)$ -hereditary 2-club but not even a 2-robust 2-club. Moreover, [Figure 3](#) depicts a  $t$ -connected 2-club that is neither a 2-robust nor a 1-hereditary 2-club. Thus, [Observation 2](#) lists all relations between the three  $t$ -well-connected 2-club models we study.

In our implementation, we use data reduction rules that remove vertices of low degree. The rationale behind these rules is provided by the following observations.

**Observation 3.** *1. Neither a  $t$ -robust 2-club nor a  $t$ -connected 2-club can contain a vertex of degree less than  $t$ .*

*2. A  $t$ -hereditary 2-club  $S$  containing a vertex of degree less than  $t + 1$  is a clique.*

*Proof.* First, consider  $t$ -robust and  $t$ -connected 2-clubs. Let  $S$  be a vertex set containing a vertex  $u$  of degree less than  $t$ . The number of vertex-disjoint paths (of any length) between  $u$  and any other

vertex in  $S$  is less than  $t$  since  $u$  has less than  $t$  neighbors. Hence,  $S$  is neither a  $t$ -robust 2-club nor a  $t$ -connected 2-club.

We now show the claim for  $t$ -hereditary 2-clubs. Let  $S$  be a  $t$ -hereditary 2-club containing a vertex  $u$  of degree less than  $t+1$ . It follows from [Definition 4](#) and [Observation 1](#) that  $u$  is adjacent to all other vertices in  $S$ . Hence,  $|S| \leq t+1$  and all vertices in  $S$  have also degree less than  $t+1$ . It follows from [Definition 4](#) and [Observation 1](#) that  $S$  is a clique.  $\square$

Thus, for  $t$ -connected and  $t$ -robust 2-clubs one may remove all vertices of degree less than  $t$ . Furthermore, to find  $t$ -hereditary 2-clubs containing vertices of degree less than  $t+1$  it is sufficient to solve CLIQUE:

**Observation 4.** *Every clique is a  $t$ -hereditary 2-club for all values of  $t$ .*

*Proof.* Clearly, a clique  $K$  is a 2-club. Since  $K$  does not contain any nonadjacent vertices, [Lemma 1](#) implies that  $K$  is a  $t$ -hereditary 2-club.  $\square$

[Observation 4](#) implies that the largest  $t$ -hereditary 2-club is at least as large as a maximum-cardinality clique. In contrast, for every graph there are values of  $t$  such that the graph neither contains a  $t$ -robust 2-club nor a  $t$ -connected 2-club. For example a clique of  $n$  vertices is not  $n$ -robust and also not  $n$ -connected.

For  $t = n - 1$ , where  $n$  is the number of graph vertices, we always obtain the clique model as the following observation implies.

**Observation 5.** *A  $t$ -hereditary 2-club  $S$  with at most  $t+2$  vertices is a clique. A  $t$ -robust/ $t$ -connected 2-club  $S$  with at most  $t+1$  vertices is a clique of size  $t+1$ .*

*Proof.* First consider a  $t$ -hereditary 2-club  $S$  with at most  $t+2$  vertices. For any pair of vertices of  $S$ , there are only  $t$  further vertices in  $S$  which potentially can be common neighbors for both. Therefore, by [Definition 4](#), any two vertices in  $S$  can only be compatible if they are adjacent. Thus,  $S$  is a clique.

For  $t$ -robust 2-club and  $t$ -connected 2-clubs the statement follows from [Observation 3](#) and the fact that the maximum degree in a graph on  $t+1$  vertices is at most  $t$ .  $\square$

## 4 Complexity Classification and Exact Algorithms

In this section, we show that all three  $t$ -well-connected 2-club detection problems are NP-complete for all fixed  $t$ . In the case of  $t$ -robust and  $t$ -hereditary 2-clubs, we observe that the NP-completeness also holds for graphs that share properties with scale-free networks. The NP-hardness of the problems motivates the development of fixed-parameter algorithms for them. Fixed-parameter algorithms are exact algorithms whose running time is polynomial in the input size but usually exponential in a problem-specific parameter [[12](#), [14](#), [29](#), [15](#)]. We will describe such an algorithm for all three problems and show that, in the case of  $t$ -ROBUST 2-CLUB and  $t$ -HEREDITARY 2-CLUB, it is likely to be optimal with respect to worst-case running time. Subsequently, we will show how to solve all three problems by solving  $O(n)$  smaller instances of the problem. This, together with the fixed-parameter algorithm, will serve as the basis for our implementation, providing an exact combinatorial algorithm for finding well-connected 2-clubs.

### 4.1 Hardness Results

By a polynomial-time reduction from 2-CLUB, we show, for all values of  $t$ , the NP-hardness of finding  $t$ -hereditary and  $t$ -robust 2-clubs.

**Theorem 1.**  *$t$ -ROBUST 2-CLUB is NP-complete for every  $t \geq 1$ .  $t$ -HEREDITARY 2-CLUB is NP-complete for every  $t \geq 0$ .*

*Proof.* Containment in NP is obvious. The NP-hardness of 1-ROBUST 2-CLUB and 0-HEREDITARY 2-CLUB is already known as these problems are exactly the 2-CLUB problem [7]. The NP-hardness of both problems for the remaining values of  $t$  can be shown via simple polynomial-time reductions from the 2-CLUB problem, which was shown to be NP-complete by Bourjolly et al. [7].

For any 2-CLUB instance  $(G, k)$  we construct an equivalent  $t$ -HEREDITARY 2-CLUB instance  $(G', k+t)$ ,  $t > 0$ , where  $G' = (V', E')$  is obtained from  $G$  by adding a set  $V^* = \{v_1^*, \dots, v_t^*\}$  of  $t$  further vertices and making them adjacent to all vertices of  $G$  and to each other. This can obviously be done in polynomial time. It remains to be shown that  $G$  has a 2-club of size at least  $k$  if and only if  $G'$  has a  $t$ -hereditary 2-club of size at least  $k+t$ .

First, consider a 2-club  $S$  in  $G$ . Then, the set  $S \cup V^*$  is a  $t$ -hereditary 2-club in  $G'$ : Only vertices in  $S$  can be pairwise nonadjacent. Moreover, each pair of nonadjacent vertices in  $S$  has at least one common neighbor in  $S$  and  $t$  common neighbors in  $V^*$ . Thus, by Definition 4 and Observation 1,  $S \cup V^*$  is a  $t$ -hereditary 2-club.

Conversely, consider a  $t$ -hereditary 2-club  $S'$  of size at least  $k+t$  in  $G'$ . Note that  $|V^*| = t$  and thus, by Definition 2,  $S' \setminus V^*$  is a 2-club of size at least  $k$  in  $G$ .

For  $t$ -ROBUST 2-CLUB,  $t > 1$ , we adapt the construction by adding to  $G$  a set  $V^* = \{v_1^*, \dots, v_{t-1}^*\}$  of  $t-1$  further vertices and making them adjacent to all vertices of  $V \cup V^*$ .

If  $G$  has a 2-club  $S$  of size at least  $k$ , then  $S \cup V^*$  is a  $t$ -robust 2-club of size at least  $k+t-1$  in  $G'$ : Every pair of vertices from  $S$  is connected by at least  $t$  internally vertex-disjoint paths of length two in  $G'[S \cup V^*]$  since they have  $t-1$  such paths via the vertices in  $V^*$  and are either adjacent or have a common neighbor in  $S$ .

Conversely, let  $S'$  be a  $t$ -robust 2-club of size at least  $k+t-1$  in  $G'$  and let  $S := S' \cap V$ . Then,  $S$  is a 2-club of size at least  $k$  in  $G$ : Every pair of nonadjacent vertices in  $S$  is connected by  $t$  vertex-disjoint paths of length two in  $G'[S']$ . At least one of these paths contains only vertices from  $S$  since  $|S' \setminus S| \leq t-1$ . Hence, these nonadjacent vertices have a common neighbor in  $S$ .

Altogether we have polynomial-time reductions from 2-CLUB to  $t$ -HEREDITARY 2-CLUB for every  $t \geq 0$  and from 2-CLUB to  $t$ -ROBUST 2-CLUB for every  $t \geq 1$ .  $\square$

The above reduction leaves the graph almost unchanged and thus many NP-hardness results that were obtained for 2-CLUB transfer almost directly to  $t$ -HEREDITARY 2-CLUB and  $t$ -ROBUST 2-CLUB. We list here the hardness results for graphs with properties that are typical for certain real-world networks. For instance, the class of split graphs finds applications in social networks in the context of “core-periphery models” [5].

The graph classes mentioned in the subsequent corollary are defined as follows. A graph  $G = (V, E)$  is a *split graph* if  $V$  can be partitioned into  $V_1$  and  $V_2$  such that  $G[V_1]$  is a clique and  $G[V_2]$  is an independent set. A graph has *domination number* 1 if there is a vertex that is adjacent to all other vertices. A graph  $G$  has *degeneracy*  $d$  if every subgraph of  $G$  contains a vertex of degree at most  $d$ .

**Corollary 1.**  *$t$ -HEREDITARY 2-CLUB and  $t$ -ROBUST 2-CLUB are NP-complete for  $t \geq 1$  on*

- (1) *graphs with both diameter two and domination number one,*
- (2) *connected graphs with average vertex degree at most  $\alpha$ , for all  $\alpha > 2$ ,*
- (3) *graphs with degeneracy  $6+t$ , and*
- (4) *split graphs.*

*Proof.* Regarding (1), note that the graph  $G'$  constructed by the reduction in the proof of Theorem 1 has at least one vertex adjacent to all other vertices and thus diameter two and domination number one.

Regarding (2), note that the statement is known for 1-robust 2-clubs, that is, standard 2-clubs [22]. Thus, we discuss subsequently  $t$ -hereditary 2-clubs for  $t \geq 1$  and  $t$ -robust 2-clubs for  $t \geq 2$ . We add a long path to the graph obtained by the construction in the proof of Theorem 1 and add an edge from one endpoint of the path to an arbitrary vertex in the original graph. This



decreases the average degree to any constant arbitrarily close to 2. No vertex of the path can be contained in any  $t$ -hereditary ( $t \geq 1$ ) or  $t$ -robust ( $t \geq 2$ ) 2-club of size at least three. Since we can assume that the solution size  $k$  is at least three, we obtain an equivalent instance with the desired average degree. This adjusted reduction shows NP-hardness for any average degree greater than two.

Regarding (3), note that 2-CLUB is NP-hard on graphs with degeneracy six [22]. Adding  $t$  vertices that are adjacent to all other vertices increases the degeneracy by at most  $t$ , therefore  $t$ -HEREDITARY 2-CLUB and  $t$ -ROBUST 2-CLUB are NP-hard on graphs with degeneracy  $6 + t$ .

Regarding (4), observe that if the graph  $G$  in the 2-CLUB instance is a split graph, then also the graph  $G'$  constructed by the reduction in the proof of Theorem 1 is a split graph. This implies the NP-hardness of  $t$ -HEREDITARY 2-CLUB and  $t$ -ROBUST 2-CLUB on split graphs since 2-CLUB remains NP-hard on split graphs [2].  $\square$

$t$ -CONNECTED 2-CLUB is known to be NP-hard for  $t \leq 2$  [4, 36]; for the sake of completeness, we show hardness for all constant  $t$ . Observe in this context that the reduction in the proof of Theorem 1 does not work for  $t$ -CONNECTED 2-CLUB: Adding  $t - 1$  common neighbors for all vertices of  $G$  results in a  $t$ -connected graph of diameter two.

**Theorem 2.**  *$t$ -CONNECTED 2-CLUB is NP-complete, even on split graphs, for all  $t \geq 1$ .*

*Proof.* Containment in NP is obvious. To show NP-hardness, we use a known reduction from CLIQUE to 2-CLUB [4]. We assume, however, that in the CLIQUE instance every vertex has at least  $t$  neighbors; obviously, CLIQUE remains NP-hard with this restriction. Given such an instance  $(G = (V, E), k)$  of CLIQUE, construct an instance  $G' = (V', E')$  of  $t$ -CONNECTED 2-CLUB as follows. The vertex set  $V'$  consists of  $V$  and one vertex for each edge of  $G$ . More formally,  $V' := V \cup V_E$  where  $V_E := \{v_e \mid e \in E\}$ . Now construct the edge set  $E'$ : First, make  $V_E$  a clique. Second, for each  $v \in V$  and each edge  $e$  that is incident with  $v$  in  $G$ , add an edge between  $v$  and  $v_e$ . The graph  $G'$  is a split graph and can obviously be constructed in polynomial time. We conclude the proof by showing that  $G$  has a clique of size at least  $k$  if and only if  $G'$  has a  $t$ -connected 2-club of size at least  $k + |E|$ .

Let  $S$  be a clique in  $G$ . Then  $S \cup V_E$  is a  $t$ -connected 2-club of size at least  $k + |E|$  in  $G'$ :  $t$ -connectivity follows from the fact that, by the assumption on the minimum degree in  $G$ , each vertex of  $S$  has  $t$  neighbors in the large clique  $V_E$ . The 2-club property follows from the fact that each pair of vertices  $u, v \in S$  has a common neighbor in  $V_E$  since  $u$  and  $v$  are adjacent in  $G$ .

Conversely, consider a  $t$ -connected 2-club  $S'$  of size at least  $k + |E|$  in  $G'$ . Consider  $S := S' \cap V$ . Clearly,  $|S| \geq k$ . Moreover,  $S$  is a clique in  $G$ : every pair of vertices  $u$  and  $v$  in  $S$  has a common neighbor in  $G'$ . This common neighbor represents an edge of  $G$  that is incident with  $u$  and  $v$ .  $\square$

## 4.2 Fixed-Parameter Algorithm with Respect to the Dual Parameter $n-k$

We now describe a fixed-parameter algorithm for all three problems which is the basis for our implementation. The general idea—branching on vertices that are incompatible—was introduced by Bourjolly et al. [7] for 2-CLUB. The analysis of the algorithm exploits the parameter  $\ell := n - k$ , that is, the number of vertices not contained in a largest  $t$ -well-connected 2-club.

**Theorem 3.** *For  $\ell := n - k$ ,  $t$ -HEREDITARY 2-CLUB and  $t$ -ROBUST 2-CLUB can be solved in  $O(2^\ell \cdot nm)$  time and  $t$ -CONNECTED 2-CLUB can be solved in  $O(2^\ell \cdot t^2 nm)$  time.*

*Proof.* The algorithm is a simple branching algorithm that, as long as two vertices are incompatible, branches into deleting either of them; see Algorithm 1 for a pseudocode. This procedure leads to a search tree with at most  $2^\ell$  leaves. To determine in Line 3 whether the current graph already fulfills the desired property, we check whether there are any incompatible vertices in the graph. Thus, the only difference between the three algorithms is the definition of compatibility, that is, in Lines 3 and 5 we need different algorithms to find incompatible vertices.



---

**Algorithm 1:** A fixed-parameter algorithm computing a maximum-size  $t$ -well-connected 2-club parameterized by  $\ell := n - k$ .

---

**Input:** A graph  $G = (V, E)$  and positive integers  $t$  and  $\ell$ .

**Output:** True if  $G$  contains  $t$ -well-connected 2-club of size at least  $n - \ell$ , false otherwise.

```

1 branch( $G, \ell$ )
2 Function branch( $G, \ell$ )
3   if  $G$  is a  $t$ -well-connected 2-club then return true
4   if  $\ell = 0$  then return false
5    $u, v \leftarrow$  two incompatible vertices
6   return branch( $G - u, \ell - 1$ )  $\vee$  branch( $G - v, \ell - 1$ )

```

---

For  $t$ -ROBUST 2-CLUB and for  $t$ -HEREDITARY 2-CLUB, we first count for each vertex pair the number of common neighbors. This can be done in  $O(nm)$  time by considering for each vertex in  $G$  each pair of neighbors. An incompatible pair can now be found in  $O(n^2)$  time by considering each vertex pair and applying Definition 4.

For  $t$ -CONNECTED 2-CLUB, we first determine in  $O(nm)$  time whether there is a vertex pair that has distance at least three in  $G$ . If yes, then the two vertices are nonadjacent. Otherwise, we apply the algorithm of Galil [18] that determines in  $O(t^2nm)$  time whether the graph is  $t$ -connected or to find a pair of vertices  $u$  and  $v$  that are in different  $t$ -connected components if this is not the case.  $\square$

We next prove that Algorithm 1 is likely to be asymptotically optimal for  $t$ -ROBUST 2-CLUB and  $t$ -HEREDITARY 2-CLUB by showing that there can be no algorithm with running time  $(2 - \epsilon)^\ell \cdot n^{O(1)}$  for any  $\epsilon > 0$  for  $(2, t)$ -CLUB, unless the so-called Strong Exponential Time Hypothesis (SETH) [23] fails.<sup>1</sup> The SETH postulates that CNF-SAT, the satisfiability problem for boolean formulas in conjunctive normal form, cannot be solved in time  $(2 - \epsilon)^N \cdot |F|^{O(1)}$  for any  $\epsilon > 0$ , where  $N$  is the number of variables in the given formula  $F$  and  $|F|$  is the formula size. The proof is based on the reduction described in the proof of Theorem 1 and a similar known result for 2-CLUB [21].

**Theorem 4.** *If the Strong Exponential Time Hypothesis (SETH) holds, then  $t$ -ROBUST 2-CLUB and  $t$ -HEREDITARY 2-CLUB do not admit a  $(2 - \epsilon)^\ell \cdot n^{O(1)}$ -time algorithm for any  $\epsilon > 0$ .*

*Proof.* The reduction presented in the proof of Theorem 1 transforms a 2-CLUB instance  $(G = (V, E), k)$  into a  $t$ -ROBUST 2-CLUB or a  $t$ -HEREDITARY 2-CLUB instance  $(G' = (V', E'), k')$  where  $|V'| = |V| + t - 1$  and  $k' = k + t - 1$  or  $|V'| = |V| + t$  and  $k' = k + t$ . Hence, the dual parameters  $\ell = |V| - k$  and  $\ell' = |V'| - k'$  are not changed by either reduction. Consequently, any algorithm solving  $t$ -ROBUST 2-CLUB or  $t$ -HEREDITARY 2-CLUB in  $(2 - \epsilon)^\ell \cdot n^{O(1)}$  time for some  $\epsilon > 0$  implies an algorithm solving 2-CLUB in this time. This is impossible if the SETH is true [21].  $\square$

As a final note, as shown by Chang et al. [11] for 2-CLUB, Algorithm 1 has a worst-case running time of  $\alpha^n n^{O(1)}$ , where  $\alpha < 1.62$  is the golden ratio.

**Corollary 2.**  *$t$ -ROBUST 2-CLUB,  $t$ -HEREDITARY 2-CLUB, and  $t$ -CONNECTED 2-CLUB can be solved in  $O(1.62^n)$  time.*

*Proof.* Consider the search tree algorithm of Theorem 3. Whenever we branch over a pair of incompatible vertices  $v$  and  $w$ , we can delete  $v$  in one branch and fix  $v$  (as contained in the sought solution) in the other branch. The justification for fixing  $v$  is that the first branch already explores all possible solutions that do not contain  $v$ , therefore  $v$  can be assumed to be in the solution in the second branch. For any fixed vertex, we delete all vertices that are incompatible with  $v$ . Thus, in

---

<sup>1</sup> The SETH is a well-accepted hypothesis in computational complexity theory and is used in several other lower-bound results [27].

the branch fixing  $v$ , we can delete  $w$  and, possibly, some further vertices. Branching continues until either (a) the graph is  $t$ -well-connected 2-club or (b) all remaining vertices are fixed. In case (a) we found a solution and in case (b) we fixed two incompatible vertices, leading to a contradiction. Letting  $j$  denote the number of nonfixed vertices in  $G$ , the recursion for the number of leaves in the search tree is  $T(j) = T(j - 1) + T(j - 2)$  and initially  $j = n$ . This is the recurrence for Fibonacci numbers implying a search tree size of  $O(\alpha^n)$  where  $\alpha < 1.62$ .  $\square$

### 4.3 Turing Kernelization

In this section, we describe parameterized preprocessing rules for our three  $t$ -well-connected 2-club detection problems. Herein, the parameter is the maximum degree  $\Delta$  of the input graph  $G$ . The idea of a “Turing kernelization” is to solve the original problem instance by solving polynomially many problem instances whose size is upper-bounded in the parameter value (cf. Kratsch [25]). These problem instances are small if the parameter value is small. In our case, we need to solve  $n$  problem instances, each consisting of  $O(\Delta^2)$  vertices. An analogous result for 2-club was shown by Schäfer et al. [33].

**Theorem 5.**  *$t$ -ROBUST 2-CLUB,  $t$ -HEREDITARY 2-CLUB,  $t$ -CONNECTED 2-CLUB can be solved by solving  $O(n)$  instances of  $t$ -ROBUST 2-CLUB,  $t$ -HEREDITARY 2-CLUB, and  $t$ -CONNECTED 2-CLUB, respectively, where each instance contains at most  $\Delta^2$  vertices.*

*Proof.* By definition, every 2-club has diameter at most two and is therefore fully contained in the closed 2-neighborhood of each of its vertices in the original input graph. To find the largest  $t$ -well-connected 2-club in a graph  $G$  it is now sufficient to search in each of the  $n$  closed 2-neighborhoods of  $G$  for a largest such 2-club and take the largest among all of them. In a graph with maximum degree  $\Delta \geq 2$  the size of the closed 2-neighborhood of any vertex  $v$  is upper-bounded by  $\Delta^2$  as  $v$  can have at most  $\Delta$  neighbors which each can have at most  $\Delta - 1$  other neighbors besides  $v$ .  $\square$

While **Theorem 5** is simple to show, it has important practical implications. It allows us to decompose larger (and relatively sparse) graphs into a linear number of smaller graphs where the problem can be solved efficiently. The smaller individual graphs are called *Turing kernels*. This result has been used for 2-CLUB-implementations [21] and we use it as well in our implementation as described in **Section 5**. As a corollary, we obtain the following running time bound employing the parameter  $\Delta$ . To this end, note that one can compute all Turing kernels in  $O(n\Delta^2)$  time.

**Corollary 3.**  *$t$ -ROBUST 2-CLUB,  $t$ -HEREDITARY 2-CLUB, and  $t$ -CONNECTED 2-CLUB can be solved in  $O(1.62^{\Delta^2} \cdot n)$  time on graphs with maximum degree  $\Delta$ .*

## 5 Implemented Algorithm

In this section we describe our implementation for solving all three extensions of the 2-CLUB model. Recall that we use the term  $t$ -well-connected 2-clubs to make statements that hold for all three models. The pseudocode given in **Algorithm 2** shows the general setup of the algorithm which includes the Turing kernelization (**Theorem 5**) and the search tree (**Theorem 3**) combined with data reduction rules inspired by Hartung et al. [21]. In addition, our algorithm uses data structures for maintaining compatibility information (see **Definition 4**) for all vertices.

While the criterion for compatibility depends on the selected model, our algorithm mostly works with the compatibility information. Thus, the major difference between the three models within our algorithm is the test whether two vertices are compatible. For the  $t$ -robust 2-club and the  $t$ -hereditary 2-club model, the compatibility of two vertices depends on whether they are adjacent and how many common neighbors they have. It is thus relatively easy to maintain the compatibility information for these two models as deleting a vertex can only affect the compatibilities of its neighbors. For the  $t$ -connected 2-club model one needs to compute the number of (internally) vertex-disjoint paths to determine whether two vertices at distance at most two are compatible. To this end, we use a standard linear-time reduction to a maximum flow problem with unit capacities. To compute the maximum flow, we use the classic algorithm of Ford and Fulkerson [16].

---

**Algorithm 2:** Our algorithm for finding  $t$ -well-connected 2-clubs.

---

**Input:** A graph  $G = (V, E)$  and a positive integer  $t$ .  
**Output:** A maximum-size  $t$ -well-connected 2-club.

```

1  $\ell \leftarrow 0$ ;  $S \leftarrow \emptyset$  //  $\ell$ : lower bound,  $S$ : best solution so far
2 if  $t = 1$  then  $\ell \leftarrow \Delta + 1$  // initialize lower bound
3  $\mathcal{T} \leftarrow$  Turing kernels sorted by size in nondecreasing order
4 foreach Turing kernel  $T \in \mathcal{T}$  do
5   Apply data reduction rules on  $T$  // see Section 5.2
6   if size of  $T$  is larger than  $\ell$  then
7     solution  $\leftarrow$  branch on  $T$  // using Algorithm 1; see Section 5.1
8     if solution size  $> \ell$  then
9        $\ell \leftarrow$  solution size;  $S \leftarrow$  solution
10 return  $S$ 

```

---

## 5.1 The Search Tree Method and Turing Kernelization

The first step of the algorithm is to utilize Turing kernelization based on our observations from Section 4.3. For each vertex  $v$  of the original graph, we construct a Turing kernel consisting of the closed 2-neighborhood  $N_2[v]$  of that vertex. We then run the search tree method described in Section 4.2 on each of the generated Turing kernels. For each Turing kernel we initially mark the vertex  $v$  from which we derived the kernel (marking means that we assume that  $v$  is part of the  $t$ -well-connected 2-club). The marking is correct since every vertex is marked exactly once. Additionally, after solving the Turing kernel for  $v$  we remove  $v$  from the graph and from all subsequent Turing kernels because if  $v$  is part of an optimal solution, then we have found this solution in the kernel for  $v$ . Also, we discard any Turing kernel not larger (in terms of number of vertices) than the current lower bound. The largest  $t$ -well-connected 2-club of the input graph then is simply the largest among the  $t$ -well-connected 2-clubs of each individual kernel.

As an initial lower bound, we use maximum degree plus one for  $t = 1$  ( $t = 0$  in the case of  $t$ -hereditary 2-clubs). For  $t \geq 2$ , we set the initial lower bound to zero. This is because we know of no lower bounds which are easy to compute and sufficiently strong. Due to the absence of an initial lower bound, we check the individual kernels of the graph in order of nondecreasing size. The idea is to keep for all considered kernels the gap between the current solution size lower bound and the size of the currently considered kernel as small as possible, as this gap constitutes an upper bound on the depth of the search tree for any given kernel. To reduce overhead from re-evaluating the size of each kernel after every vertex deletion, we sort the kernels of the graph once at the beginning of the algorithm by the size of the 2-neighborhood of the respective vertices and stick to this order throughout the run of the algorithm, even though some kernels might become smaller than some of their predecessors due to vertex deletions. In our experiments we observed that this worked quite well in the sense that on instances where the algorithm needs more than a minute, less than 10% of the running time of our algorithm is spent on the branching.

## 5.2 Data Reduction Rules

To further improve the search tree method, we exhaustively apply various data reduction rules to shrink the original graph as well as every Turing kernel. The data reduction rules we use are mostly generalizations of known data reduction rules for 2-CLUB [21]. They are as follows:

**Reduction Rule 1** (Marked Incompatible Rule). *If at any time two vertices are both marked and incompatible, then abort the branch.*

**Reduction Rule 2** (Incompatible Resolution). *Remove all vertices which are incompatible with any marked vertices.*

The correctness of these two rules is obvious as a  $t$ -well-connected 2-club by definition cannot contain incompatible vertices.

**Reduction Rule 3** (Low Degree Rule). *Remove vertices of degree less than  $t$  (less than  $t + 1$  for the  $t$ -hereditary 2-club model). For  $t = 1$  ( $t = 0$  for the  $t$ -hereditary 2-club model) delete all vertices of degree one. If a marked vertex was deleted, then abort the branch.*

This data reduction rule is not universally correct, so we need to make two exceptions.

First, deleting degree-one vertices for  $t = 1$  ( $t = 0$  for the  $t$ -hereditary 2-club model) is only correct when initializing the algorithm with the closed 1-neighborhood of the maximum degree vertex as initial solution. A 2-club containing a degree-one vertex can only consist of the closed 1-neighborhood of its neighbor. Hence, no solution better than the one obtained from the closed 1-neighborhood of the maximum degree vertex contains degree-one vertices.

Second, [Observation 3](#) shows that for the  $t$ -hereditary 2-club model, deleting vertices of degree less than  $t + 1$  is only correct if the solution is not a clique. However, cliques of size at most  $t$  are by definition  $t$ -hereditary 2-clubs. Thus, we first assume that the  $t$ -hereditary 2-club has size at least  $t + 1$  which allows us to apply the Low Degree Rule. If we do not find a  $t$ -hereditary 2-club under this assumption, then we use a standard CLIQUE algorithm to find the largest clique and return it. The running time of the CLIQUE algorithm is, if it is invoked at all, much lower than the running time for finding the  $t$ -hereditary 2-club of size at least  $t + 1$ . We use the Low Degree Rule extensively on the whole graph as well as on every kernel individually, both initially and whenever the degree of a vertex falls below  $t$  due to vertex deletions.

**Reduction Rule 4** (Low Compatibility Rule). *Remove vertices whose number of compatible vertices is lower than the current lower bound. If a marked vertex was deleted, then abort the branch.*

The correctness of this data reduction rule follows from the fact that a  $t$ -well-connected 2-club containing some vertex  $v$  can only contain vertices which are compatible with  $v$ . Thus, if  $v$  has less compatibilities than the current lower bound, then no  $t$ -well-connected 2-club containing  $v$  contains more vertices than the current lower bound.

Our next data reduction rule uses the notion of vertex cover. A *vertex cover* in a graph  $G = (V, E)$  is a vertex subset  $V' \subseteq V$  such that each edge in  $E$  is incident with at least one vertex in  $V'$ .

**Reduction Rule 5** (Vertex Cover Rule). *Let  $G = (V, E)$  be the current instance and let  $G_C = (V, E')$  be the corresponding incompatibility graph, where  $\{v, w\} \in E'$  if and only if  $v$  and  $w$  are incompatible. Compute a lower bound  $b$  on the vertex cover size of  $G_C$ . If  $|V| - b$  does not exceed our current lower bound for the solution size, then abort the branch.*

The correctness of the Vertex Cover Rule follows from the observation that the size of a minimum vertex cover of  $G_C$  is a lower bound on the number of vertices in  $G$  that still must be deleted in order to obtain a  $t$ -well-connected 2-club. As a quickly-to-compute lower bound for the vertex cover size we use the size of a maximal matching.

The Vertex Cover Rule (VCR) reduces the search space substantially, but it is costly in terms of running time. We therefore use the following observations to cut down the number of necessary applications of the VCR: Whenever we have created no new incompatibilities since the last application of the VCR, the last result of the VCR is still a valid lower bound for the required number of vertex deletions. Thus, we reuse the last lower bound  $b$  on the vertex cover when applying the VCR rule. To further decrease the number of VCR applications, we remember the last vertex cover of the incompatibility graph and only count as new those incompatibilities which are not covered by this last vertex cover. If we have created  $x$  new conflicts since the last application of the VCR, then a new application of the VCR will report at most  $x$  further required deletions over the last result of the VCR. So if the last upper bound minus  $x$  is not enough to trigger the VCR, then there is no need to check the VCR again.

**Reduction Rule 6** (No Choice Rule). *If at any time two nonadjacent marked vertices have exactly  $x$  common neighbors, then mark all of their neighbors which are not marked yet. Here,  $x$  is:*

- $t$  for the  $t$ -robust 2-club model,
- $t + 1$  for the  $t$ -hereditary 2-club model, and
- 1 for the  $t$ -connected 2-club model.

The reason for having a much weaker bound for  $t$ -connected 2-clubs is that the  $t$  paths connecting two compatible vertices can have length more than two: For example, consider a cycle  $C_5$  on five vertices with exactly two nonadjacent vertices  $u$  and  $v$  being marked. In the cycle,  $u$  and  $v$  have exactly one common neighbor. However, the  $C_5$  is a 2-connected 2-club. The graph shown in Figure 3 in Section 3 “generalizes” the  $C_5$  example: The graph is a  $t$ -connected 2-club is displayed where vertex pairs (e.g. a marked vertex in the set  $V_1$  and a marked vertex in the set  $V_4$ ) have only one common neighbor. Thus, if two nonadjacent marked vertices have at least two common neighbors, then one might still be able to delete one of the two vertices without destroying  $t$ -connectedness.

We exhaustively apply all reduction rules except the Vertex Cover Rule on each constructed Turing kernel. We use the Vertex Cover Rule only during the branching to prune the search tree.

### 5.3 Data Structures

To perform the data reduction rules quickly at all times, the following information about the current Turing kernel is held up-to-date. To this end, recall that the current Turing kernel is a vertex  $v \in V$  together with its 2-neighborhood and that the Turing kernels are processed one after the other, see Lines 4 to 9 in Algorithm 2.

**Common Neighbor Matrix** A matrix storing for each pair of vertices in the current Turing kernel their number of common neighbors. This data structure is very important for the  $t$ -hereditary 2-club and  $t$ -robust 2-club model. However, also in the  $t$ -connected 2-club model it is useful to check in constant time whether two nonadjacent vertices have a common neighbor, that is, whether they are at distance at most two.

**Incompatibility Graph** A graph over the same vertex set as in the Turing kernel where two vertices are adjacent if and only if the two vertices are incompatible. Hence, for each edge  $\{v, w\}$  in this graph at least one of the two vertices  $v$  or  $w$  must be deleted in order to obtain a  $t$ -well-connected 2-club.

**Compatibility Vector** A vector containing for each vertex  $v$  of the current Turing kernel the number of vertices that are compatible with  $v$ . Among other things this vector allows a fast (linear in the number of vertices of the current Turing kernel) look-up of the vertex that is compatible with the fewest other vertices; this vertex is used as the next branching candidate. When looking for 2-clubs, the entries in the compatibility vector correspond to the size of the 2-neighborhood of the respective vertex.

The initialization of these data structures requires  $O(m\Delta + n^2)$  time for the  $t$ -hereditary 2-club model and  $t$ -robust 2-club model, where  $n$  is the number of vertices in the kernel,  $m$  is the number of edges in the kernel, and  $\Delta$  is the maximum vertex degree in the kernel. For the  $t$ -connected 2-club model we implemented the classical maximum flow algorithm of Ford and Fulkerson [16] to determine whether two vertices are compatible. Thus, for  $t$ -connected 2-clubs there is an additional  $O(tm)$  running time factor giving an overall initialization time of  $O(tn^2m)$ .

We first initialize the common neighbor matrix with zero values and then for each vertex  $v$  of the kernel increase the entry of each pair of neighbors of  $v$  by one. Afterwards, for the  $t$ -hereditary 2-club model and the  $t$ -robust 2-club model, a single pass over the common neighbor matrix suffices to count the number of compatible vertices for each vertex and identifying every initial incompatibility. For the  $t$ -connected 2-club model, we run the maximum flow algorithm for every pair of vertices that are at distance at most two from each other. The initialization cost can

be quite large for dense kernels with many vertices (more than 50% of the overall running time on large social networks), so we apply all data reduction rules we can use without this information first. If after this quick preprocessing the kernel has less vertices than the current lower bound, then we discard the kernel before the initialization of the data structures. To keep the information continuously up-to-date, for every vertex  $v$  that is deleted, we do the following.

1. Decrease the number of compatible vertices of all vertices not compatible with  $v$  by one; this can be done in  $O(n)$  time, where  $n$  is the number of vertices in the current kernel.
2. Decrease the common neighbor counter for all pairs of neighbors of  $v$  by one; this can be done in  $O(\deg(v)^2)$  time, where  $\deg(v)$  is the degree of  $v$ .
3. Update the incompatibility graph:
  - For  $t$ -robust 2-clubs and  $t$ -hereditary 2-clubs, we can quickly update the incompatibility graph while updating the common neighbor counter (see Definition 4).
  - For  $t$ -connected 2-clubs, we run the maximum flow algorithm between every pair of vertices that were compatible before deleting  $v$ . If the two vertices have distance more than two, then we set them incompatible without running the maximum flow algorithm.

For  $t$ -robust 2-clubs and  $t$ -hereditary 2-clubs, this leads to an  $O(n + \deg(v)^2)$ -time overhead for every deletion of a vertex  $v$  to keep the information up-to-date. For  $t$ -connected 2-clubs, the overhead is in the worst case  $O(tn^2m)$ . Overall, the overhead is quite significant but enables us to perform our data reduction rules quickly and extensively whenever they apply and as soon as they apply.

## 5.4 Integer Linear Programming Formulation for the $t$ -Hereditary 2-Club Problem

We want to utilize a state-of-the-art ILP solver (Gurobi) as benchmark for our algorithm on instances with  $t \geq 2$ . To this end, we now describe the ILP formulation for  $t$ -HEREDITARY 2-CLUB which we will use in our performance tests in Section 6. Given a graph  $G = (V, E)$  and an integer  $t$ , the  $t$ -HEREDITARY 2-CLUB problem can be solved with the following ILP:

Introduce a 0/1-variable  $x_v$  for each vertex  $v$  of the graph  $G$ . Setting  $x_v = 1$  indicates  $v \in S$ , where  $S$  is the solution set of vertices, that is, a set of vertices constituting a maximum  $t$ -hereditary 2-club. The ILP is

$$\begin{aligned}
& \max \sum_{v \in V} x_v \\
& \text{s. t. } (t+1) \cdot x_u + (t+1) \cdot x_w - \sum_{v \in N(u) \cap N(w)} x_v \leq (t+1) & \forall \{u, w\} \notin E, \\
& x_v \in \{0, 1\} & \forall v \in V.
\end{aligned}$$

For each nonadjacent pair of vertices  $u$  and  $w$  we have a constraint enforcing to either take at most one of the two vertices  $u$  and  $w$  into the solution or add at least  $t+1$  of their common neighbors, too.

This formulation generalizes the ILP formulation for the 2-CLUB problem as first described by Bourjolly et al. [7] and used several times afterwards [4, 9, 8]. Furthermore, we remark that the ILP formulation of Veremyev and Boginski [34] for finding  $t$ -robust 2-clubs is very similar.

## 6 Computational Experiments

In this section, we provide an experimental evaluation of our implementation<sup>2</sup> (see Section 5) on random graphs as well as on real-world graphs from the 10th DIMACS challenge [13]. Our

<sup>2</sup>Available from <http://ftp.akt.tu-berlin.de/software/well-connected-2-club/wc2club.jar>.



Table 1: Results on random graphs of density 0.15. All time measurements are the average over 100 instances with the described parameters. All running times are given in seconds. If a running time is bold, then some of these 100 instances (the number is given in brackets) did not finish within the time limit of one hour and the respective instance is accounted with one hour in the average. Bold values are therefore just lower bounds on the real running times.

$a$	$b$	$n$	WCC	HKN	CHLS	ILP
0.0	0.3	140	3.50	1.27	1.00	3.61
		150	5.71	1.71	1.82	4.48
		160	6.84	1.99	2.64	5.03
0.05	0.25	140	19.34	6.11	7.37	38.67
		150	39.54	11.98	18.00	63.23
		160	72.35	18.33	30.54	95.04
0.1	0.2	140	70.09	24.34	29.99	246.20
		150	183.04	59.41	85.94	<b>605.76 [2]</b>
		160	557.92	178.16	306.51	<b>1,115.38 [9]</b>
0.15	0.15	140	105.68	39.85	48.62	612.50
		150	322.70	118.77	168.91	<b>1,508.00 [8]</b>
		160	<b>1,132.05 [1]</b>	384.38	<b>611.65 [1]</b>	<b>2,594.98 [46]</b>

implementation solves the optimization versions of the problems and finds  $t$ -well-connected 2-clubs of maximum size. We refer to our implementation in the following as WCC (well-connected 2-club). For the special case of solving the basic 2-CLUB problem (see Section 6.1), we compare WCC against the programs of Hartung et al. [21]<sup>3</sup> (referred to as HKN) and Chang et al. [11]<sup>4</sup> (referred to as CHLS). Furthermore, in Section 6.2 we also compare against two programs of Yezerka et al. [36]<sup>5</sup> (one branch-and-bound algorithm, one ILP) for finding 2-connected 2-clubs. In Section 6.3 we analyze the performance of WCC for finding  $t$ -well-connected 2-clubs in social networks for  $t \geq 1$ . In all our experiments, we set the time limit to one hour. All time measurements include the time to read the input graph.

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. Our implementation is in Java and runs under the OpenJDK runtime environment in version 1.7.0\_65. The C++ implementation of Yezerka et al. [36] was compiled with g++ version 5.4.0.

## 6.1 Comparison with other 2-Club Algorithms

We may use our implementation (WCC) in three ways to find 2-clubs: by looking for 1-robust 2-clubs, 0-hereditary 2-clubs, or 1-connected 2-clubs. We report the running times for the setting where we look for 0-hereditary 2-clubs since this is faster than the other two settings. The setting where we look for 1-connected 2-clubs is particularly slow since our implementation uses a maximum flow computation to verify that the graph is connected.

**Random Graphs** As in previous experimental evaluations [21, 11], we use the random graph generator due to Gendreau et al. [19] where the density of the resulting graphs is controlled by two parameters,  $0 \leq a \leq b \leq 1$ , and the expected density is  $(a + b)/2$ .

The results on random graphs give a clear picture: As our algorithm is not designed for random graphs but large sparse real-world networks, the implementations of Hartung et al. [21] (HKN) and Chang et al. [11] (CHLS) are both significantly faster than ours. However, WCC still outperforms our ILP. Table 1 summarizes the results for random graphs with expected density 0.15, which produces the hardest instances as already observed earlier [7, 21, 11].

<sup>3</sup>Available from [http://ftp.akt.tu-berlin.de/two\\_club/](http://ftp.akt.tu-berlin.de/two_club/).

<sup>4</sup>Available from <https://sites.google.com/site/kdynamicneighborhoodproject/>.

<sup>5</sup>Yezerka et al. [36] sent us the code of their two programs.



Table 2: Our data set sorted by the number of edges. Here,  $n^*$  is the number of vertices (excluding isolated vertices that are discarded when reading the graph),  $m$  is the number of edges, density is defined as  $m/\binom{n^*}{2}$ ,  $\Delta$  is the maximum degree, and size is the number of vertices in a largest 2-club.

Graph	$n^*$	$m$	density	avg. deg.	$\Delta$	size
karate	34	78	0.14	4.59	17	18
dolphins	62	159	$8.41 \cdot 10^{-2}$	5.13	12	13
adjnoun	112	425	$6.84 \cdot 10^{-2}$	7.59	49	50
polbooks	105	441	$8.08 \cdot 10^{-2}$	8.4	25	28
football	115	613	$9.35 \cdot 10^{-2}$	10.66	12	16
celegans-metabolic	453	2,025	$1.98 \cdot 10^{-2}$	8.94	237	238
jazz	198	2,742	0.14	27.7	100	103
netscience	1,461	2,742	$2.57 \cdot 10^{-3}$	3.75	34	35
email	1,133	5,451	$8.5 \cdot 10^{-3}$	9.62	71	72
power	4,941	6,594	$5.4 \cdot 10^{-4}$	2.67	19	20
uk	4,824	6,837	$5.88 \cdot 10^{-4}$	2.83	3	5
add20	2,395	7,462	$2.6 \cdot 10^{-3}$	6.23	123	124
add32	4,960	9,462	$7.69 \cdot 10^{-4}$	3.82	31	32
data	2,851	15,093	$3.71 \cdot 10^{-3}$	10.59	17	18
hep-th	7,610	15,751	$5.44 \cdot 10^{-4}$	4.14	50	51
polblogs	1,224	16,715	$2.23 \cdot 10^{-2}$	27.31	351	352
PGPgiantcompo	10,680	24,316	$4.26 \cdot 10^{-4}$	4.55	205	206
whitaker3	9,800	28,989	$6.04 \cdot 10^{-4}$	5.92	8	9
crack	10,240	30,380	$5.8 \cdot 10^{-4}$	5.93	9	10
cs4	22,499	43,858	$1.73 \cdot 10^{-4}$	3.9	4	6
coAuthorsCiteseer	$2.3 \cdot 10^5$	$8.1 \cdot 10^5$	$3.15 \cdot 10^{-5}$	7.16	1,372	1,373
coAuthorsDBLP	$3 \cdot 10^5$	$9.8 \cdot 10^5$	$2.19 \cdot 10^{-5}$	6.54	336	337
citationCiteseer	$2.7 \cdot 10^5$	$1.2 \cdot 10^6$	$3.21 \cdot 10^{-5}$	8.62	1,318	1,319
graph-thres-01	$7.2 \cdot 10^5$	$2.5 \cdot 10^6$	$9.81 \cdot 10^{-6}$	7.02	804	805
coPapersDBLP	$5.4 \cdot 10^5$	$1.5 \cdot 10^7$	$1.04 \cdot 10^{-4}$	56.42	3,299	3,300
coPapersCiteseer	$4.3 \cdot 10^5$	$1.6 \cdot 10^7$	$1.7 \cdot 10^{-4}$	73.89	1,188	1,189

The algorithm HKN could solve all random instances within the time limit of one hour. Our algorithm WCC as well as the algorithm CHLS could solve all but one instance within this time limit. Both WCC and CHLS fail to solve the same instance, which HKN could solve in 2,571 seconds. The ILP, however, failed on 65 of the overall 1,200 instances.

On average, WCC is 3.1 times slower than HKN and 2.4 times slower than CHLS. If one considers the accumulated running times over all instances, then WCC is 3 times slower than HKN and 1.9 times slower than CHLS. Compared to the ILP on instances where it found a solution within the time limit, WCC is on average 2.4 times faster and considering the accumulated running times (with unsolved instances counted with one hour), WCC is at least 2.7 times faster.

The reason for this relatively weak performance of WCC is the extensive usage of data reduction rules. The algorithm tries to apply all data reduction rules, but on random data they seldom apply. To improve the performance on random graphs, improving the efficiency of the data reduction rules as well as investigating heuristic selection strategies concerning when to apply which rule could be a promising approach for improvements.

**Real-World Graphs** We considered real-world graphs from the 10th DIMACS challenge [13]. We ran our algorithm on instances from the categories *Clustering* and *Walshaw's Graph Partitioning Archive* [13]; this is a standard benchmark for graph clustering and community detection algorithms also used by Yezereska et al. [36] to evaluate their implementation for finding biconnected 2-clubs. To test our algorithm on large scale social network graphs we ran it also on graphs from the *co-author and citation* category [13] and a graph (graph-thres-01) mined from DBLP<sup>6</sup> in 2012. Table 2 shows our test set of graphs and Table 3 overviews the results for the real-world graphs.

<sup>6</sup><http://dblp.uni-trier.de/faq/Am+I+allowed+to+crawl+the+dblp+website.html>

Table 3: Results for 2-CLUB on real-world graphs with a time limit of one hour per instance. Here,  $n^*$  is the number of vertices with degree at least one,  $m$  is the number of edges, and size is the number of vertices in a largest 2-club. Empty cells represent timeouts (for WCC or HKN) or insufficient memory (for CHLS). All times are in seconds.

Graph	$n^*$	$m$	size	WCC	HKN	CHLS
karate	34	78	18	0.14	0.16	0
dolphins	62	159	13	0.15	0.18	0
adjnoun	112	425	50	0.15	0.19	0
polbooks	105	441	28	0.19	0.24	0.01
football	115	613	16	0.21	0.33	0.08
celegans-metabolic	453	2,025	238	0.22	0.37	0.02
jazz	198	2,742	103	0.37	0.4	0.15
netscience	1,461	2,742	35	0.18	0.24	0.11
email	1,133	5,451	72	0.59	43.65	2.57
power	4,941	6,594	20	0.25	0.54	0.7
uk	4,824	6,837	5	0.23	113.2	17.6
add20	2,395	7,462	124	0.38	0.88	0.42
add32	4,960	9,462	32	0.24	0.79	0.73
data	2,851	15,093	18	0.46	2,664.71	7.88
hep-th	7,610	15,751	51	0.3	1.19	3.8
polblogs	1,224	16,715	352	3.81	7.55	16.14
PGPgiantcompo	10,680	24,316	206	0.45	1.57	3.64
whitaker3	9,800	28,989	9	0.62	93.1	
crack	10,240	30,380	10	0.71	157.09	
cs4	22,499	43,858	6	0.83	281.86	
coAuthorsCiteseer	$2.3 \cdot 10^5$	$8.1 \cdot 10^5$	1,373	11.46	50.35	
coAuthorsDBLP	$3 \cdot 10^5$	$9.8 \cdot 10^5$	337	15.76	78.51	
citationCiteseer	$2.7 \cdot 10^5$	$1.2 \cdot 10^6$	1,319	21.46	83.9	
graph-thres-01	$7.2 \cdot 10^5$	$2.5 \cdot 10^6$	805	75.14	279.29	
coPapersDBLP	$5.4 \cdot 10^5$	$1.5 \cdot 10^7$	3,300	250.83	2,288.49	
coPapersCiteseer	$4.3 \cdot 10^5$	$1.6 \cdot 10^7$				

Similarly to the experiments on random graphs, CHLS was the fastest solver on small instances. This is probably due to the fact that CHLS is written in C (WCC and HKN are written in Java) and it uses an adjacency matrix as graph representation (WCC and HKN use adjacency lists). Due to the usage of an adjacency matrix to store the graph, CHLS could not store and thus not solve any large graph in the *Co-author* category.

WCC could solve all but one of the instances of the smaller graphs (less than 50,000 edges) within one second; the exception, namely *polblogs*, needed less than four seconds. On medium-size and large instances, WCC clearly outperformed the other algorithms. The only instance that WCC could not solve within one hour is the largest graph *coPapersCiteseer*. A second run without time limit shows a running time of 1.8 hours for this instance; the second-best solver HKN needs 8.6 *days*. This good performance is mainly due to the extensive usage of data reduction rules—a feature that slows down our algorithm on random graphs.

## 6.2 Comparison with other 2-Connected 2-Club Algorithms

We considered the same real-world instances as in Section 6.1 to compare against the two implementations of Yezerska et al. [36]. Their first implementation, which we refer to as BB, is a combinatorial branch-and-bound algorithm with some lower bound heuristics. Their second implementation, which we refer to as BC, is a branch-and-cut algorithm that is based on an ILP formulation. The results of the three algorithms are shown in Table 4.

WCC solved 21 out of the 26 instances within the time limit of one hour and all but one of the smaller graphs with less than 50,000 edges within less than one minute. Only one instance, namely *polblogs*, could not be solved within one hour. Moreover, WCC solved two larger graphs with more than 800,000 edges within less than 10 minutes. As already discussed in Section 5, WCC is not as efficient for the  $t$ -connected 2-club model as for the other two models. In fact, a closer inspection

Table 4: Results for 2-CONNECTED 2-CLUB on real-world graphs with a time limit of one hour per instance. Here,  $n^*$  is the number of vertices with degree at least one,  $m$  is the number of edges, and size is the number of vertices in a largest 2-club. Empty cells represent timeouts. All times are in seconds.

Graph	$n^*$	$m$	size	WCC	BB	BC
karate	34	78	17	0.17	0.02	0.03
dolphins	62	159	12	0.26	0.07	0.13
adjnoun	112	425	48	1.39	0.58	0.26
polbooks	105	441	28	0.53	0.09	0.15
football	115	613	16	3.68	0.86	1.9
celegans-metabolic	453	2,025	222	28.6	95.03	16.11
jazz	198	2,742	103	22.3	9.39	0.43
netscience	1,461	2,742	25	0.27	198.47	76.67
email	1,133	5,451	69	39.76	290.84	35.24
power	4,941	6,594	14	0.28		2,577.7
uk	4,824	6,837	5	0.32		
add20	2,395	7,462	124	8.01	128.84	106.98
add32	4,960	9,462	30	0.51		
data	2,851	15,093	17	2	3,338.59	2,639.83
hep-th	7,610	15,751	45	1.12		
polblogs	1,224	16,715	346			482.11
PGPgiantcompo	10,680	24,316	196	23.1		
whitaker3	9,800	28,989	9	3.6		
crack	10,240	30,380	10	7.03		
cs4	22,499	43,858	6	1.61		
coAuthorsCiteseer	$2.3 \cdot 10^5$	$8.1 \cdot 10^5$	484	359		
coAuthorsDBLP	$3 \cdot 10^5$	$9.8 \cdot 10^5$	325	198.83		
citationCiteseer	$2.7 \cdot 10^5$	$1.2 \cdot 10^6$				
graph-thres-01	$7.2 \cdot 10^5$	$2.5 \cdot 10^6$				
coPapersDBLP	$5.4 \cdot 10^5$	$1.5 \cdot 10^7$				
coPapersCiteseer	$4.3 \cdot 10^5$	$1.6 \cdot 10^7$				

of our implementation on the graph **polblogs** showed that more than 95% of the running time went into the maximum flow-based connectivity check. Herein, the reduction to the maximum flow instance, without the computation of the maximum flow itself, took more than half of the time. Although this reduction can be performed in linear time, it is applied very often after branching or after applying the data reduction rules.

BB solved 11 and BC solved 13 out of the 26 instances. Both of these solvers were faster on the smallest instances and on a few medium size instances. In fact, the smaller graph **polblogs** that WCC could not solve is solved by BC in around eight minutes. However, WCC solved three smaller graphs (**power**, **uk**, **add32**) with less than 10,000 edges in less than a second each and both BB and BC could not solve them in less than half an hour. Neither BB nor BC could solve any of the larger graphs with more than 800,000 edges. In fact, the ILP-based solver BC did not even finish writing all constraints within one hour for these large graphs.

Summarizing, WCC solved substantially more instances than BB and BC. Again, the good performance is mainly due to the extensive usage of data reduction rules. Furthermore, as one can see in the results for **polblogs**, if the data reductions are not effective enough, then WCC needs much more time to solve the instance.

We did not perform tests on random graphs since we expect that the outcome is similar as in [Section 6.1](#): WCC will be outperformed by BB and BC.

### 6.3 Evaluation for $t \geq 1$

Finally, we performed experiments for computing  $t$ -well-connected 2-clubs with  $t \geq 1$ . For these experiments, we compare the running times of WCC and ILP. Moreover, we compare our three well-connected 2-club models with each other. Herein, we compare  $t$ -robust 2-clubs and  $t$ -connected 2-clubs with  $(t - 1)$ -hereditary 2-clubs. The reason is that the base case—2-clubs—is reached for

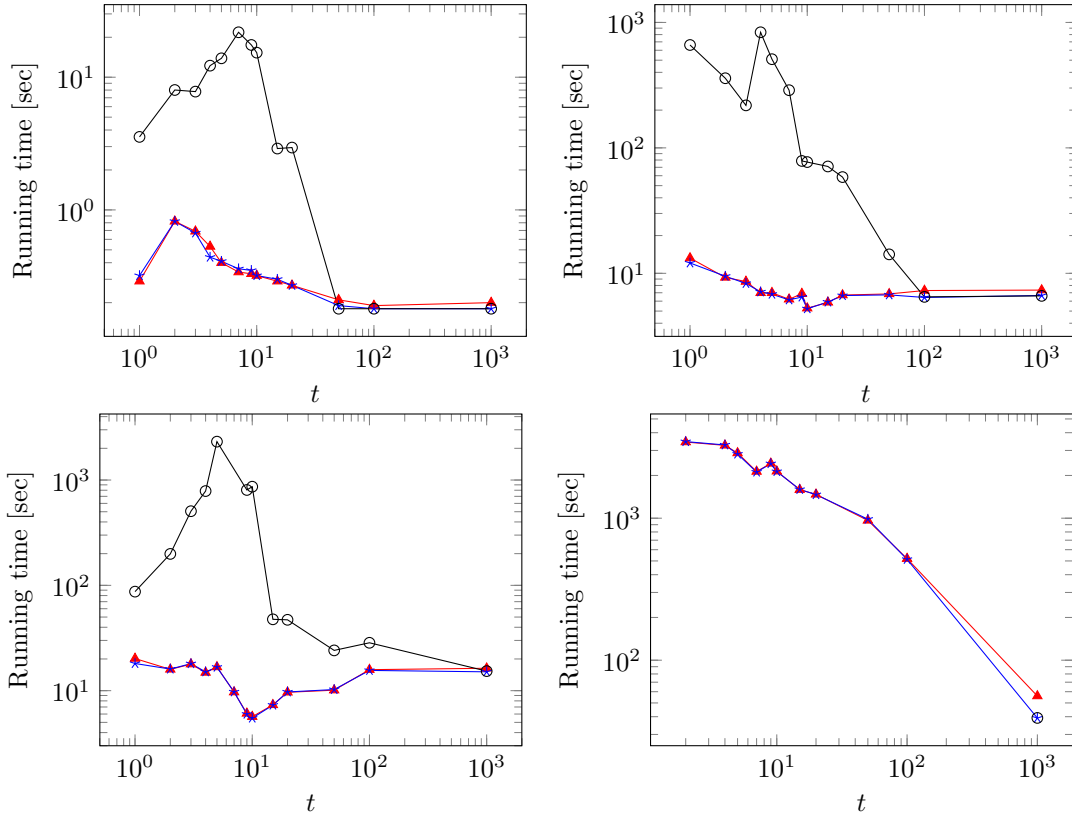


Figure 4: Running time plots for finding  $t$ -well-connected 2-clubs for different values of  $t$  on add20 (top left), coAuthorsCiteseer (top right), coAuthorsDBLP (bottom left), and coPapersCiteseer (bottom right). All running times are in seconds. In all plots, black circles correspond to  $t$ -connected 2-clubs, red triangles correspond to  $(t-1)$ -hereditary 2-clubs, and blue stars correspond to  $t$ -robust 2-clubs. In the two plots on top, the instances could be solved for all  $t$  and all three models. In the bottom-left plot, one black marker (for  $t = 7$ ) is missing due to a time out. In the bottom-right plot several markers are missing due to time outs. Overall, smaller values of  $t$  seem harder. However, there is no clear trend showing which concrete  $t$ -values lead to the computationally hardest instances.

different  $t$ -values: A 2-club is a 1-robust 2-club, a 1-connected 2-club, and a 0-hereditary 2-club. Furthermore, one can see many similar solutions when comparing  $t$ -robust 2-clubs against  $(t-1)$ -hereditary 2-clubs. We ran our implementation with values for  $t$  being 1, 2, 3, 4, 5, 7, 9, 10, 15, 20, 50, 100, and 1,000. The performance of our implementation varies for different values of  $t$  but there seems to be no clear correlation between  $t$  and the performance.

In our tests WCC clearly outperformed ILP. Furthermore, ILP could solve only small real-world graphs. Due to this one-sided result, we only discuss the results of WCC. We refer to [A](#) for a complete list of the experiments on real-world graphs (including ILP). Comparing WCC on the three models, unsurprisingly, the results for  $(t-1)$ -hereditary 2-clubs and  $t$ -robust 2-clubs are very similar (see [Figure 4](#)). In both variants, WCC solved the same 329 of the overall 338 instances within one hour. On average, the  $t$ -robust 2-club could be computed 5% faster. Probably, this is due to the fact that for computing  $(t-1)$ -hereditary 2-clubs we run a clique-algorithm if the  $(t-1)$ -hereditary 2-club has size at most  $t$  (see [Observation 5](#)). Only in 20 of the 175 instances that contain a  $t$ -robust 2-club the maximum  $(t-1)$ -hereditary 2-club is larger than the maximum  $t$ -robust 2-club (see also [Figure 5](#)). The largest relative difference in sizes (in cases where a  $t$ -robust 2-club exists) was observed in the graph `adjnoun` where the largest 4-robust 2-club has size six and

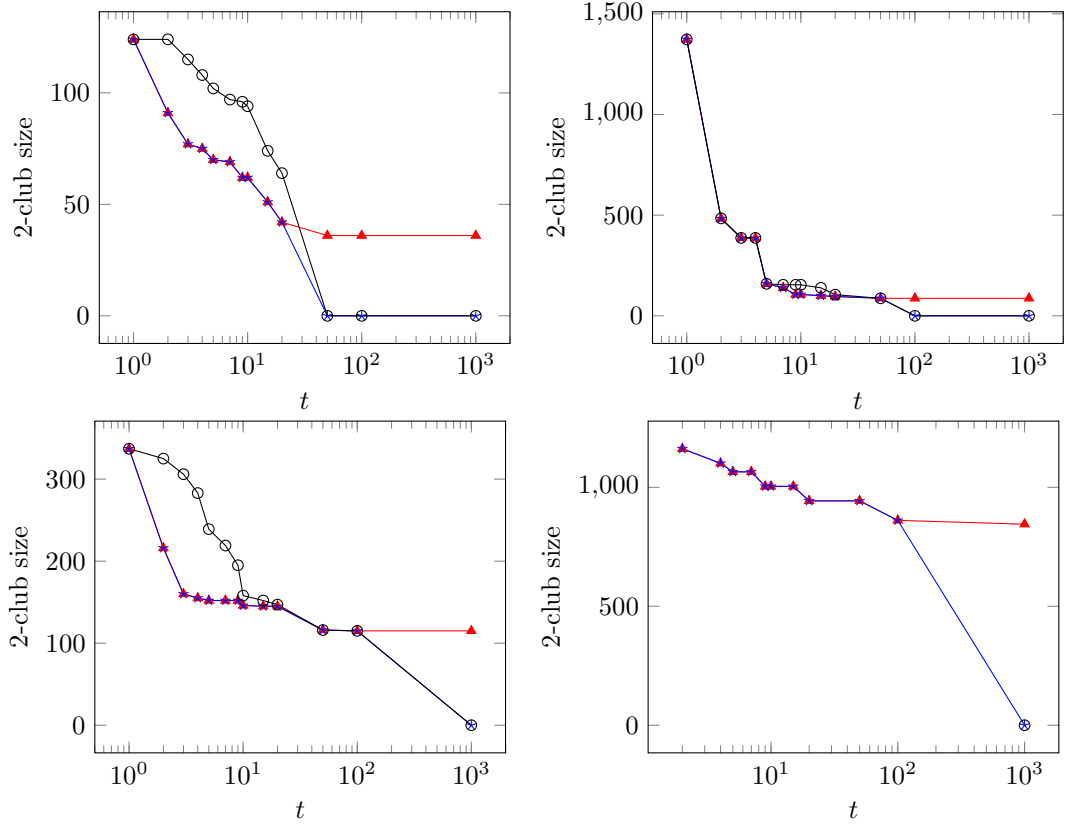


Figure 5: The ordering of the respective maximum  $t$ -well-connected 2-clubs for different values of  $t$  on the same graphs: `add20` (top left), `coAuthorsCiteseer` (top right), `coAuthorsDBLP` (bottom left), and `coPapersCiteseer` (bottom right). In all plots, black circles correspond to  $t$ -connected 2-clubs, red triangles correspond to  $(t - 1)$ -hereditary 2-clubs, and blue stars correspond to  $t$ -robust 2-clubs. The instances are the same as in Figure 4 and the same markers are missing due to time-outs. Again, the differences between  $(t - 1)$ -hereditary 2-clubs and  $t$ -robust 2-clubs are minimal. Only if there is no  $t$ -robust 2-club and the largest  $(t - 1)$ -hereditary 2-club is a clique, then there is a visible difference. The  $t$ -connected 2-club can be significantly larger than the other 2-clubs (see the two plots on the left side).

the largest 3-hereditary 2-club has size nine.

For `citationCiteseer`, `coAuthorsCiteseer`, `coAuthorsDBLP`, and `graph-thres-01`, WCC finds the largest  $(t - 1)$ -hereditary 2-club and the largest  $t$ -robust 2-club relatively quickly (always within three minutes). This is not the case for the two largest graphs `coPapersCiteseer` and `coPapersDBLP`. While `coPapersCiteseer` could be solved within one hour for all values of  $t$  except 1 and 3, `coPapersDBLP` could not be solved for  $t$  ranging from 2 to 10. However, experiments without time limit and with further statistics collected during all stages of the implementation solved all instances within two hours. Since computing and outputting these additional data affected the running time quite a lot, we did not include the measured running times in the appendix.

WCC found the largest  $t$ -connected 2-club on many instances (290 of the 338 instances were solved). While only one of the smaller graphs (`polblogs`) caused trouble for WCC, WCC struggled on large graphs with more than 800,000 edges; only 39 of the 78 instances corresponding to these large graphs could be solved within the time limit of one hour. On the instances WCC could deal with it needed much more time than for finding the largest  $t$ -robust or  $(t - 1)$ -hereditary 2-club (see also Figure 4). A good example here is the graph `coAuthorsDBLP` where finding the largest 9-connected 2-club was more than 100 times slower than finding the largest 8-hereditary 2-club.

### Detailed Running Time Analysis for $(t - 1)$ -Hereditary and $t$ -Robust 2-Club Model

A closer inspection of the coPapersDBLP graph shows that, for all  $t > 1$ , our implementation spent around 75% of the time for constructing the Turing kernels and initializing the data structures (common neighbor matrix, incompatibility graph, and compatibility vector; see Section 5). The rest of the time is spent mostly on applying the data reduction rules; for each value of  $t$ , less than 30 seconds were spent on the actual branching. For  $t = 1$ , the maximum degree of 3,299 gives a lower bound of 3,300 for the solution, which in this case is the solution size. Thus, all but six Turing kernels can be dismissed due to too small size before initializing the data structures (see discussion in Section 5.3). This explains why the implementation was that much faster for  $t = 1$ .

The coPapersCiteseer graph is quite different. This can be seen for example in the solution size: For increasing  $t$  the size of the solution decreases only slightly (see Figure 5 bottom-right plot); this is in stark contrast to the coPapersDBLP graph. Furthermore, for  $t = 1$  on the coPapersCiteseer graph, there were 3,175 Turing kernels that were all larger than the lower bound. The average number of vertices in these Turing kernels was 1,389; the maximum was 2,022, the minimum 1,189. Recall that our data structures involve square matrices (the common neighbor matrix) whose number of rows and columns is equal to the number of vertices in the respective Turing kernel. This explains why the initialization of these data structures was the bottleneck in our implementation.

## 6.4 Overall Conclusions from Experiments

We demonstrated the effectiveness of our general search-tree approach to find  $t$ -well-connected 2-clubs in large sparse real-world networks. In particular, to the best of our knowledge, we provide the first results for large sparse social networks with more than a million edges. The key ingredients for our implementation are the extensive usage of Turing kernels and data reduction rules.

For finding  $t$ -connected 2-clubs there is room for improving our implementation. To this end, the most obvious approach is to look for an efficient data structure that can quickly delete vertices and answer connectivity queries. Finding good lower bounds is another approach to speed up our implementation as we only use a lower bound for the base case of finding 2-clubs.

Our experiments also indicated that the common ILP formulations are not able to cope with the large graphs we consider. The reason is simply that creating all constraints is too time consuming.<sup>7</sup> Combining Turing-kernelization and data reduction rules with ILPs using row generation and other sophisticated tricks might lead to a competitive program.

## 7 Outlook

We studied three established models for cohesive subgraphs, namely  $t$ -robust,  $t$ -hereditary, and  $t$ -connected 2-clubs. These are considered to overcome the typical hub-and-spoke structure in the solutions of classical 2-CLUB problem. We presented theoretical findings for these models which provided the basis for our implementation of an exact algorithm for the problems. Our experiments demonstrate the efficiency of the presented algorithm on large sparse real-world graphs.

We conclude with some challenges for future research:

- Our algorithmic result with respect to maximum degree (see Theorem 5) does not fully explain the success of our implementation on our data set as the parameter value is too large (see Table 2). Moreover, we have NP-hardness even if the smaller parameters average degree and degeneracy are constant (see Corollary 1). Thus, finding the “right” structure that explains the practically observed performance also in theory remains a task for future research.

---

<sup>7</sup>To the best of our knowledge, ILP formulations for 2-CLUB and its variants have at least  $n^2 - m$  constraints [8, 36, 34]. Even for the smallest graph from the *co-author and citation* category [13], this amounts to more than  $5.2 \cdot 10^{10}$  constraints which, on the hardware we used, could not be generated within one hour.

- It is open to study the polynomial-time approximability of the different  $t$ -well connected 2-club variants. Notably, 2-CLUB has a factor- $O(n^{1/2})$  approximation algorithm and is NP-hard to approximate within a factor of  $O(n^{1/2-\varepsilon})$  [2].
- An extension of the algorithmic approach to 3-clubs and beyond is another challenge. Almeida and Carvalho [1] presented an ILP formulation for  $t$ -robust 3-clubs and Veremyev and Boginski [34] presented an ILP formulation for a relaxation of  $t$ -robust  $s$ -clubs with  $s > 2$ . Solving the relaxed problems turned out to be computationally very hard; we are not aware of experimental evaluations of the exact formulation for  $t$ -robust 3-clubs.
- Veremyev et al. [35] studied a variation of  $t$ -robust 2-clubs (called  $\gamma$ -relative-robust 2-clubs) where essentially the constant  $t$  is replaced by a function in the size of the club. Our algorithms do not work for this variant since we always assume that at most one of two incompatible vertices (see Definition 4 in Section 3) can be in a  $t$ -well-connected 2-club. This assumption does not hold for  $\gamma$ -relative-robust 2-clubs. Extending our work to this model remains a task for future work.
- The combination of efficient data reduction rules with mathematical programming techniques may lead to further accelerations in finding  $t$ -well-connected 2-clubs.

**Acknowledgment** We are grateful to four anonymous reviewers of *EJOR* for their careful and constructive feedback. We thank Oleksandra Yezeraska, Foad Mahdavi Pajouh and Sergiy Butenko [36] for providing us with their source code of their programs.

## References

- [1] M. T. Almeida and F. D. Carvalho. An analytical comparison of the LP relaxations of integer models for the  $k$ -club problem. *European Journal of Operational Research*, 232(3):489–498, 2014. 22
- [2] Y. Asahiro, Y. Doi, E. Miyano, K. Samizo, and H. Shimizu. Optimal approximation algorithms for maximum distance-bounded subgraph problems. *Algorithmica*, 80(6):1834–1856, 2018. 8, 22
- [3] C. Balasubramaniam and S. Butenko. On robust clusters of minimum cardinality in networks. *Annals of Operations Research*, 249(1-2):17–37, 2017. 3
- [4] B. Balasundaram, S. Butenko, and S. Trukhanov. Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization*, 10(1):23–39, 2005. 8, 14
- [5] S. P. Borgatti and M. G. Everett. Models of core/periphery structures. *Social Networks*, 21(4):375–395, 2000. 7
- [6] 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. 3
- [7] 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. 3, 7, 8, 14, 15
- [8] A. Buchanan and H. Salemi. Parsimonious formulations of low-diameter clusters. *Optimization Online Eprints*, 2017. 3, 14, 21
- [9] 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. 3, 14
- [10] F. D. Carvalho and M. T. Almeida. The triangle  $k$ -club problem. *Journal of Combinatorial Optimization*, 33(3):814–846, 2017. 3



- [11] 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. 3, 9, 15
- [12] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015. 6
- [13] DIMACS’12. Graph partitioning and graph clustering. 10th DIMACS implementation challenge, 2012. URL <http://www.cc.gatech.edu/dimacs10/>. Accessed April 2012. 14, 16, 21
- [14] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. 6
- [15] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006. 6
- [16] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956. 10, 13
- [17] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010. 1
- [18] Z. Galil. Finding the vertex connectivity of graphs. *SIAM Journal on Computing*, 9(1):197–199, 1980. 9
- [19] 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. 15
- [20] P. A. Golovach, P. Heggernes, D. Kratsch, and A. Rafiey. Finding clubs in graph classes. *Discrete Applied Mathematics*, 174:57–65, 2014. 3
- [21] S. Hartung, C. Komusiewicz, and A. Nichterlein. Parameterized algorithmics and computational experiments for finding 2-clubs. *Journal of Graph Algorithms and Applications*, 19(1):155–190, 2015. 1, 3, 9, 10, 11, 15
- [22] S. Hartung, C. Komusiewicz, A. Nichterlein, and O. Suchý. On structural parameterizations for the 2-club problem. *Discrete Applied Mathematics*, 185:79–92, 2015. 3, 7, 8
- [23] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. 9
- [24] C. Komusiewicz. Multivariate algorithmics for finding cohesive subnetworks. *Algorithms*, 9(1), 2016. 1, 3
- [25] S. Kratsch. Recent developments in kernelization: A survey. *Bulletin of the EATCS*, 113, 2014. 10
- [26] V. E. Lee, N. Ruan, R. Jin, and C. C. Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 303–336. Springer, 2010. 3
- [27] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. 9
- [28] R. J. Mokken. Cliques, clubs and clans. *Quality & Quantity*, 13(2):161–173, 1979. 1
- [29] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. 6
- [30] F. M. Pajouh and B. Balasundaram. On inclusionwise maximal and maximum cardinality  $k$ -clubs in graphs. *Discrete Optimization*, 9:84–97, 2012. 3
- [31] J. Pattillo, N. Youssef, and S. Butenko. On clique relaxation models in network analysis. *European Journal of Operational Research*, 226(1):9–18, 2013. 1, 2, 3

- [32] M. Picker. Algorithms and experiments for finding robust 2-clubs. Master's thesis, TU Berlin, 2015. URL <http://ftp.akt.tu-berlin.de/publications/theses/MA-marten-picker.pdf>. 1
- [33] A. Schäfer, C. Komusiewicz, H. Moser, and R. Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, 6(5):883–891, 2012. 3, 10
- [34] A. Veremyev and V. Boginski. Identifying large robust network clusters via new compact formulations of maximum  $k$ -club problems. *European Journal of Operational Research*, 218(2):316–326, 2012. 2, 3, 14, 21, 22
- [35] A. Veremyev, O. A. Prokopyev, V. Boginski, and E. L. Pasiliao. Finding maximum subgraphs with relatively large vertex connectivity. *European Journal of Operational Research*, 239(2):349–362, 2014. 22
- [36] O. Yezerka, F. M. Pajouh, and S. Butenko. On biconnected and fragile subgraphs of low diameter. *European Journal of Operational Research*, 263(2):390–400, 2017. 3, 8, 15, 16, 17, 21, 22

## A Full Experimental Results for $t \geq 1$

Table 5: Results for the real-world graphs. The columns time and size denote the running time and the size for the respective model:  $t$ -robust (r),  $(t - 1)$ -hereditary (h & ILP; for WCC and our ILP), and  $t$ -connected (c) 2-clubs.

graph	$t$	size (r)	time (r)	size (h)	time (h)	time (ILP)	size (c)	time (c)
karate	1	18	0.13	18	0.17	0.14	18	0.16
	2	12	0.16	12	0.15	0.14	17	0.17
	3	6	0.15	6	0.14	0.14	12	0.16
	4	6	0.14	6	0.13	0.14	9	0.15
	5	0	0.14	5	0.13	0.14	0	0.12
	7	0	0.13	5	0.14	0.14	0	0.13
	9	0	0.13	5	0.13	0.13	0	0.13
	10	0	0.14	5	0.13	0.17	0	0.13
	15	0	0.13	5	0.13	0.12	0	0.13
	20	0	0.13	5	0.14	0.13	0	0.12
	50	0	0.13	5	0.14	0.14	0	0.13
	100	0	0.14	5	0.14	0.14	0	0.12
	1,000	0	0.16	5	0.14	0.14	0	0.13
dolphins	1	13	0.16	13	0.15	0.25	13	0.17
	2	9	0.16	9	0.15	0.2	12	0.26
	3	7	0.15	7	0.17	0.18	12	0.37
	4	6	0.16	6	0.14	0.19	11	0.18
	5	0	0.14	5	0.15	0.2	0	0.13
	7	0	0.14	5	0.14	0.2	0	0.12
	9	0	0.13	5	0.13	0.19	0	0.13
	10	0	0.13	5	0.15	0.19	0	0.13
	15	0	0.15	5	0.14	0.19	0	0.13
	20	0	0.13	5	0.18	0.2	0	0.13
	50	0	0.14	5	0.14	0.19	0	0.13
	100	0	0.13	5	0.14	0.19	0	0.13
	1,000	0	0.14	5	0.13	0.19	0	0.12
adjnoun	1	50	0.17	50	0.16	0.34	50	0.39
	2	23	0.19	23	0.18	0.38	48	1.39
	3	12	0.18	12	0.17	0.42	44	1.95
	4	6	0.17	9	0.18	0.44	39	2.47
	5	0	0.17	5	0.17	1.1	33	1.77
	7	0	0.16	5	0.14	0.5	0	0.14
	9	0	0.15	5	0.14	0.43	0	0.15
	10	0	0.15	5	0.14	0.43	0	0.13
	15	0	0.14	5	0.15	0.42	0	0.14
	20	0	0.15	5	0.15	0.45	0	0.13
	50	0	0.15	5	0.16	0.42	0	0.15
	100	0	0.15	5	0.15	0.42	0	0.13
	1,000	0	0.18	5	0.15	0.43	0	0.14
polbooks	1	28	0.18	28	0.17	0.25	28	0.22
	2	20	0.19	20	0.17	0.3	28	0.53
	3	15	0.18	15	0.17	0.37	28	0.61
	4	12	0.17	13	0.17	0.33	27	0.5
	5	10	0.17	11	0.16	0.45	26	0.43
	7	0	0.15	6	0.15	0.65	0	0.14
	9	0	0.14	6	0.15	0.49	0	0.14
	10	0	0.15	6	0.15	0.37	0	0.14
	15	0	0.13	6	0.15	0.37	0	0.14
	20	0	0.15	6	0.15	0.37	0	0.13
	50	0	0.15	6	0.15	0.36	0	0.14
	100	0	0.15	6	0.15	0.36	0	0.14
	1,000	0	0.14	6	0.16	0.36	0	0.14

Table 5: Results for the real-world graphs. The columns time and size denote the running time and the size for the respective model:  $t$ -robust (r),  $(t - 1)$ -hereditary (h & ILP; for WCC and our ILP), and  $t$ -connected (c) 2-clubs.

graph	$t$	size (r)	time (r)	size (h)	time (h)	time (ILP)	size (c)	time (c)
football	1	16	0.21	16	0.23	2	16	0.77
	2	14	0.18	14	0.18	0.37	16	3.68
	3	13	0.17	13	0.16	0.28	15	4.42
	4	12	0.18	13	0.17	0.29	15	2.27
	5	12	0.18	12	0.18	0.29	15	1.09
	7	10	0.17	11	0.17	0.29	13	0.57
	9	0	0.15	9	0.15	0.36	0	0.15
	10	0	0.15	9	0.16	0.32	0	0.13
	15	0	0.15	9	0.18	0.33	0	0.14
	20	0	0.16	9	0.15	0.33	0	0.14
	50	0	0.15	9	0.16	0.35	0	0.14
	100	0	0.15	9	0.16	0.33	0	0.14
	1,000	0	0.17	9	0.16	0.33	0	0.15
celegans-metabolic	1	238	0.25	238	0.23	4.52	238	10.69
	2	104	0.38	104	0.39	4.52	222	28.6
	3	54	0.35	54	0.38	12.46	195	57.06
	4	30	0.31	30	0.4	9.28	164	76.41
	5	20	0.28	22	0.27	22.03	112	88.26
	7	12	0.2	13	0.19	42.45	54	20.76
	9	0	0.18	9	0.16	36.8	31	0.35
	10	0	0.17	9	0.17	35.61	22	0.27
	15	0	0.15	9	0.16	33.14	0	0.15
	20	0	0.16	9	0.17	44.5	0	0.14
	50	0	0.16	9	0.16	37.28	0	0.16
	100	0	0.17	9	0.17	37.17	0	0.15
	1,000	0	0.16	9	0.17	36.96	0	0.15
jazz	1	103	0.4	103	0.37	1.38	103	5.52
	2	79	0.48	79	0.49	0.82	103	22.3
	3	73	0.41	73	0.39	1.2	103	45.05
	4	65	0.43	65	0.38	1.22	103	81.51
	5	60	0.38	60	0.36	1.22	103	105.18
	7	50	0.36	50	0.4	1.26	103	145.88
	9	44	0.32	44	0.34	1.33	103	191.64
	10	41	0.31	41	0.35	1.26	101	162.45
	15	32	0.27	32	0.26	1.4	96	15.46
	20	30	0.22	30	0.21	1.28	87	4.36
	50	0	0.15	30	0.18	1.21	0	0.15
	100	0	0.16	30	0.17	1.21	0	0.15
	1,000	0	0.16	30	0.17	1.21	0	0.14
netscience	1	35	0.19	35	0.19	22.29	35	0.27
	2	22	0.2	22	0.22	35.14	25	0.27
	3	21	0.18	21	0.18	35	21	0.27
	4	20	0.19	20	0.18	35.13	20	0.24
	5	20	0.18	20	0.18	31.25	20	0.23
	7	20	0.17	20	0.18	31.52	20	0.23
	9	20	0.19	20	0.17	31.42	20	0.24
	10	20	0.18	20	0.17	31.62	20	0.22
	15	20	0.18	20	0.18	31.49	20	0.22
	20	0	0.15	20	0.17	31.49	0	0.16
	50	0	0.16	20	0.17	32.19	0	0.16
	100	0	0.17	20	0.18	31.57	0	0.16
	1,000	0	0.17	20	0.18	31.43	0	0.16

Table 5: Results for the real-world graphs. The columns time and size denote the running time and the size for the respective model:  $t$ -robust (r),  $(t - 1)$ -hereditary (h & ILP; for WCC and our ILP), and  $t$ -connected (c) 2-clubs.

graph	$t$	size (r)	time (r)	size (h)	time (h)	time (ILP)	size (c)	time (c)
email	1	72	0.72	72	0.7	21.8	72	2.15
	2	27	0.55	27	0.69	435.28	69	39.76
	3	23	0.53	23	0.46	159	65	81.57
	4	19	0.44	20	0.42	148.12	60	181.97
	5	16	0.38	16	0.44	130.6	48	427.81
	7	13	0.3	13	0.3	129.18	35	183.52
	9	12	0.24	12	0.22	162.75	28	1.63
	10	12	0.18	12	0.18	127.74	12	0.19
	15	0	0.17	12	0.18	122.98	0	0.16
	20	0	0.18	12	0.18	62.57	0	0.17
	50	0	0.17	12	0.22	62.5	0	0.17
	100	0	0.17	12	0.2	62.62	0	0.17
	1,000	0	0.17	12	0.19	62.55	0	0.16
power	1	20	0.22	20	0.22	695.82	20	0.23
	2	9	0.24	9	0.23	3,448.41	14	0.28
	3	7	0.22	7	0.2		12	0.24
	4	6	0.2	6	0.19	3,253.65	11	0.21
	5	6	0.2	6	0.19	3,300.83	6	0.2
	7	0	0.2	6	0.21	3,301.94	0	0.19
	9	0	0.19	6	0.2	3,297.7	0	0.19
	10	0	0.2	6	0.21	3,299.93	0	0.17
	15	0	0.2	6	0.21	3,302.38	0	0.19
	20	0	0.2	6	0.21	3,298.42	0	0.19
	50	0	0.2	6	0.22	3,303	0	0.19
	100	0	0.18	6	0.22	3,297.65	0	0.19
	1,000	0	0.19	6	0.21	3,302.62	0	0.19
uk	1	5	0.3	5	0.24	2,352.47	5	0.27
	2	3	0.27	4	0.26	2,554.53	5	0.32
	3	0	0.2	3	0.24	1,269.42	0	0.19
	4	0	0.19	3	0.23	1,273.95	0	0.19
	5	0	0.2	3	0.22	1,266.36	0	0.19
	7	0	0.19	3	0.22	1,266.87	0	0.19
	9	0	0.19	3	0.22	1,271.76	0	0.19
	10	0	0.2	3	0.22	1,271.91	0	0.19
	15	0	0.2	3	0.23	1,267.47	0	0.19
	20	0	0.19	3	0.22	1,266.99	0	0.18
	50	0	0.2	3	0.22	1,272.23	0	0.19
	100	0	0.2	3	0.21	1,270.01	0	0.19
	1,000	0	0.19	3	0.22	1,265.45	0	0.19
add20	1	124	0.32	124	0.29	108.5	124	3.55
	2	91	0.82	91	0.82	114.1	124	8.01
	3	77	0.67	77	0.69	228.17	115	7.79
	4	75	0.44	75	0.53	116.47	108	12.22
	5	70	0.41	70	0.4	171.83	102	13.88
	7	69	0.36	69	0.34	154.72	97	21.81
	9	62	0.35	62	0.33	199.12	96	17.5
	10	62	0.32	62	0.32	187.66	94	15.32
	15	51	0.3	51	0.29	172.76	74	2.9
	20	42	0.27	42	0.27	181.7	64	2.95
	50	0	0.19	36	0.21	164.53	0	0.18
	100	0	0.18	36	0.19	163.65	0	0.18
	1,000	0	0.18	36	0.2	163.32	0	0.18

Table 5: Results for the real-world graphs. The columns time and size denote the running time and the size for the respective model:  $t$ -robust (r),  $(t - 1)$ -hereditary (h & ILP; for WCC and our ILP), and  $t$ -connected (c) 2-clubs.

graph	$t$	size (r)	time (r)	size (h)	time (h)	time (ILP)	size (c)	time (c)
add32	1	32	0.26	32	0.25	2,028.52	32	0.3
	2	12	0.43	12	0.39	2,375.38	30	0.51
	3	5	0.24	5	0.24		11	0.39
	4	0	0.2	4	0.23		0	0.2
	5	0	0.19	4	0.24		0	0.2
	7	0	0.2	4	0.22		0	0.19
	9	0	0.2	4	0.22		0	0.19
	10	0	0.23	4	0.22		0	0.18
	15	0	0.21	4	0.23		0	0.19
	20	0	0.21	4	0.22		0	0.19
	50	0	0.2	4	0.22		0	0.19
	100	0	0.2	4	0.22		0	0.2
	1,000	0	0.2	4	0.24		0	0.2
data	1	18	0.51	18	0.49	1,838.74	18	0.57
	2	14	0.65	14	0.56	2,382.05	17	2
	3	12	0.44	12	0.42	1,909.82	17	6.27
	4	8	0.56	9	0.47	1,338.3	17	7.73
	5	8	0.43	8	0.41	1,667.46	17	6.32
	7	0	0.33	6	0.37	1,741.07	0	2.3
	9	0	0.21	6	0.24	1,753.34	0	0.19
	10	0	0.21	6	0.25	1,746.32	0	0.2
	15	0	0.2	6	0.25	1,755.95	0	0.23
	20	0	0.19	6	0.25	1,753.3	0	0.2
	50	0	0.2	6	0.25	1,750.11	0	0.19
	100	0	0.2	6	0.25	1,794.5	0	0.19
	1,000	0	0.2	6	0.25	1,751.18	0	0.19
hep-th	1	51	0.35	51	0.31	804.54	51	0.5
	2	33	0.43	33	0.39	3,343.04	45	1.12
	3	24	0.33	24	0.37	3,010.08	40	1.92
	4	24	0.33	24	0.33	2,706.02	31	3.9
	5	24	0.29	24	0.3	3,203.32	28	1.47
	7	24	0.31	24	0.28	2,961.67	24	0.34
	9	24	0.26	24	0.25	3,426.85	24	0.32
	10	24	0.25	24	0.24	3,407.74	24	0.33
	15	24	0.25	24	0.24	2,947.59	24	0.32
	20	24	0.27	24	0.24	2,961.09	24	0.33
	50	0	0.25	24	0.26	3,407.03	0	0.23
	100	0	0.22	24	0.25	3,420.78	0	0.23
	1,000	0	0.24	24	0.26	3,027.49	0	0.22
polblogs	1	352	4.47	352	4.69	88.81	352	164.57
	2	232	5.96	232	6.5	97.18		
	3	182	6.94	182	7.44	1,552.44		
	4	158	6.11	159	5.79	1,091.47		
	5	146	3.96	147	3.95	1,168.24		
	7	124	3.12	124	3.91	1,167.79		
	9	108	2.69	108	3.04	1,173.24		
	10	104	2.48	104	2.46	1,029.84		
	15	79	1.77	81	1.82	897.59		
	20	62	1.38	64	1.36	2,371.7		
	50	0	0.2	20	0.41	3,053.38	0	0.19
	100	0	0.2	20	0.44	2,933.52	0	0.19
	1,000	0	0.2	20	0.4	2,854.28	0	0.19

Table 5: Results for the real-world graphs. The columns time and size denote the running time and the size for the respective model:  $t$ -robust (r),  $(t - 1)$ -hereditary (h & ILP; for WCC and our ILP), and  $t$ -connected (c) 2-clubs.

graph	$t$	size (r)	time (r)	size (h)	time (h)	time (ILP)	size (c)	time (c)
PGPgiantcompo	1	206	0.51	206	0.46	3,319.96	206	14.12
	2	96	1.18	96	0.94		196	23.1
	3	71	0.86	71	0.86		188	36.16
	4	64	0.79	64	0.78		177	57.6
	5	57	0.75	57	0.74		159	133.66
	7	47	0.58	47	0.59		146	136.57
	9	46	0.5	46	0.49		96	51.12
	10	45	0.44	45	0.44		85	14.26
	15	45	0.37	45	0.37		46	2.74
	20	43	0.38	43	0.38		45	1.52
	50	0	0.28	25	0.35		0	0.27
	100	0	0.28	25	0.33		0	0.28
	1,000	0	0.27	25	0.35		0	0.26
whitaker3	1	9	0.64	9	0.62		9	0.75
	2	6	0.69	6	0.61		9	3.6
	3	0	0.66	5	0.62		9	3.43
	4	0	0.29	3	0.38		0	0.3
	5	0	0.26	3	0.34		0	0.25
	7	0	0.27	3	0.35		0	0.26
	9	0	0.27	3	0.35		0	0.26
	10	0	0.26	3	0.35		0	0.26
	15	0	0.28	3	0.36		0	0.26
	20	0	0.27	3	0.36		0	0.26
	50	0	0.26	3	0.36		0	0.25
	100	0	0.26	3	0.36		0	0.26
	1,000	0	0.27	3	0.36		0	0.25
crack	1	10	0.8	10	0.75		10	1.05
	2	6	0.74	6	0.73		10	7.03
	3	0	0.85	5	0.61		9	14.58
	4	0	0.44	3	0.52		0	1.08
	5	0	0.28	3	0.37		0	0.26
	7	0	0.28	3	0.38		0	0.26
	9	0	0.27	3	0.37		0	0.27
	10	0	0.28	3	0.38		0	0.28
	15	0	0.29	3	0.38		0	0.28
	20	0	0.28	3	0.38		0	0.26
	50	0	0.28	3	0.38		0	0.27
	100	0	0.28	3	0.38		0	0.27
	1,000	0	0.28	3	0.38		0	0.27
cs4	1	6	0.85	6	0.86		6	1.35
	2	3	0.81	4	0.91		6	1.61
	3	0	0.58	3	0.63		0	0.83
	4	0	0.47	3	0.6		0	0.47
	5	0	0.48	3	0.65		0	0.46
	7	0	0.48	3	0.61		0	0.46
	9	0	0.49	3	0.62		0	0.46
	10	0	0.47	3	0.6		0	0.46
	15	0	0.47	3	0.61		0	0.46
	20	0	0.46	3	0.62		0	0.47
	50	0	0.46	3	0.61		0	0.45
	100	0	0.48	3	0.64		0	0.46
	1,000	0	0.47	3	0.62		0	0.47



Table 5: Results for the real-world graphs. The columns time and size denote the running time and the size for the respective model:  $t$ -robust (r),  $(t - 1)$ -hereditary (h & ILP; for WCC and our ILP), and  $t$ -connected (c) 2-clubs.

graph	$t$	size (r)	time (r)	size (h)	time (h)	time (ILP)	size (c)	time (c)
coAuthorsCiteSeer	1	1,373	12.09	1,373	13.23		1,373	660.97
	2	483	9.48	483	9.27		484	359
	3	387	8.3	387	8.6		387	218.18
	4	386	7.11	386	6.99		387	834.57
	5	159	6.81	159	6.99		159	509.23
	7	139	6.14	139	6.21		154	288.28
	9	106	6.47	106	6.88		154	78.67
	10	106	5.22	106	5.27		154	77.08
	15	100	5.9	100	5.87		139	71.09
	20	96	6.65	96	6.71		105	58.5
	50	87	6.73	87	6.85		87	14.16
	100	0	6.42	87	7.29		0	6.49
	1,000	0	6.66	87	7.35		0	6.61
coAuthorsDBLP	1	337	18.11	337	20.19		337	87.17
	2	216	16.04	216	15.93		325	198.83
	3	160	18.05	160	17.86		306	505.8
	4	155	15.09	155	14.88		283	785.56
	5	152	16.7	152	16.78		239	2,314.15
	7	152	9.73	152	9.69		219	
	9	152	5.98	152	6.08		195	804.74
	10	146	5.49	146	5.66		158	864.34
	15	145	7.29	145	7.33		152	47.58
	20	145	9.72	145	9.65		147	47.08
	50	116	10.25	116	10.11		116	24.12
	100	115	15.57	115	15.83		115	28.49
	1,000	0	15.1	115	16.32		0	15.34
citationCiteSeer	1	1,319	24.99	1,319	25.61		1,319	987.46
	2	268	86.8	268	71.34			
	3	94	86.5	120	86.03			
	4	55	72.79	57	71.14			
	5	45	57.11	45	58.49			
	7	31	33.41	37	31.92			
	9	27	16.87	31	17.51			
	10	27	12.12	27	12.15			
	15	0	10	13	13.74		50	12.95
	20	0	10.43	13	13.8		0	10.27
	50	0	15.5	13	19.48		0	15.65
	100	0	16.03	13	19.65		0	15.85
	1,000	0	16.11	13	19.79		0	16.07
graph-thres-01	1	805	82.48	805	82.37		805	327.16
	2	210	127.16	210	125.7			
	3	164	137.32	164	138.04			
	4	145	122.57	145	120.99			
	5	143	98.98	143	98.26			
	7	131	64.61	131	63.3			
	9	129	36.63	129	37.33			
	10	129	27.97	129	28.03			
	15	121	21.88	121	21.82		137	104.82
	20	116	33.19	116	33.33		120	55.66
	50	115	55.34	115	55.33		115	67.77
	100	114	64.23	114	64.41		114	76.73
	1,000	0	64.88	114	67.7		0	64.82

Table 5: Results for the real-world graphs. The columns time and size denote the running time and the size for the respective model:  $t$ -robust (r),  $(t - 1)$ -hereditary (h & ILP; for WCC and our ILP), and  $t$ -connected (c) 2-clubs.

graph	$t$	size (r)	time (r)	size (h)	time (h)	time (ILP)	size (c)	time (c)
coPapersDBLP	1	3,300	293.77	3,300	293.87			
	2							
	3							
	4							
	5							
	7							
	9							
	10							
	15	763	3,521.23	763	3,401.8			
	20	763	3,218.27	763	3,228.87			
	50	459	1,703.99	459	1,698.03			
coPapersCiteseer	100	350	504.11	351	503.8			
	1,000	0	75.46	337	86.38		0	74.94
	1							
	2	1,162	3,462.18	1,162	3,438.11			
	3							
	4	1,101	3,279.47	1,101	3,257.03			
	5	1,065	2,827.55	1,065	2,877.37			
	7	1,065	2,112.45	1,065	2,132.17			
	9	1,004	2,423.17	1,004	2,429.75			
	10	1,004	2,131.84	1,004	2,140.5			
	15	1,004	1,593.92	1,004	1,588.68			
	20	943	1,465.43	943	1,471.35			
	50	943	984.1	943	964.82			
	100	861	512.88	861	518.99			
	1,000	0	39.14	845	55.93		0	39.22