Finding Connected Subgraphs of Fixed Minimum Density: Implementation and Experiments*

Christian Komusiewicz, Manuel Sorge, and Kolja Stahl

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany
{christian.komusiewicz,manuel.sorge}@tu-berlin.de
kolja.stahl@campus.tu-berlin.de

Abstract. We consider the following problem. Given a graph and a rational number μ, $0 < \mu \leq 1$, find a connected subgraph of density at least μ with the largest number of vertices. Here, the density of an n-vertex graph with m edges is $m/\binom{n}{2}$. This problem arises in many application contexts such as community detection in social networks. We implement a branch and bound algorithm and tune it for efficiency on sparse real-world graphs for the case $\mu \geq 1/2$. Central issues for the implementation are the choice of branching candidates, two new upper bounding procedures, and several data reduction and early termination rules.

1 Introduction

Identifying dense subgraphs is a problem arising in the analysis of social [4], financial [5], and biological networks [3]. In most applications, the desired dense subgraphs do not contain an edge between each vertex pair but rather adhere to a more relaxed notion of density. Many different, mathematically precise definitions of such desired subgraphs have been proposed [4, 10]. We consider the concept of μ-cliques, used for example by Abello et al. [1, 2]. It is defined as follows.

Definition 1. *The* density *of an n-vertex graph with m edges is $m/\binom{n}{2}$. A graph is a μ-*clique *if its density is at least μ.*

In general, μ-cliques need not be connected. However, this is an important property expected from a community. Hence, we impose connectivity as a further constraint on the μ-cliques we are looking for. As observed previously, demanding connectivity also allows for a simple solving algorithm [9].

Our goal in this work is to develop an implementation for finding large connected μ-cliques in a given graph for some fixed $\mu \geq 1/2$. Most input graphs in the mentioned applications are sparse with few high-degree vertices [5, 15]. We thus aim to tune the implementation to perform well on graphs with this structure. Our implementation is based on an *exact* algorithm which, given k, either

* An extended abstract of this manuscript is to appear in the *Proceedings of the 14th International Symposium on Experimental Algorithms*, LNCS, Springer, 2015.

finds a μ-clique with k vertices or determines correctly that no such subgraph exists. Exact algorithms are desirable because they yield reference points for the performance of heuristics and because surprising results can be attributed to the model (here: connected μ-cliques) rather than to deficiencies of the algorithm.

Contribution. Our implementation follows the branch and bound paradigm and is based on an algorithm proposed by a subset of the authors [9]. The input is a graph G, the density threshold μ, and the minimum required number k of vertices in the desired μ-clique. The algorithm proceeds roughly as follows. In each step, we maintain a set P of vertices which we aim to extend to a μ-clique. To do this, we maintain also an *active vertex* v whose neighbors we will consider to add to P first. That is, given P and v we branch into all possibilities of adding a neighbor of v to P, and into the possibility of making v permanently inactive and consequently choosing a new active vertex in P. We terminate this process if P has size k and report $G[P]$ if it is a connected μ-clique.

This algorithm is called for increasing values of k. If for some value of k no connected μ-clique is found, then it stops and returns the largest μ-clique computed so far. This approach is only correct if the nonexistence of a μ-clique of order k implies that there is also no μ-clique of order $k+1$. In a first step, we thus examine whether connected μ-cliques fulfill a nestedness property (which is called *quasi-heredity* [12]). We obtain that for $\mu \geq 1/2$, connected μ-cliques are quasi-hereditary, but that for $\mu < 1/2$, they cannot be assumed to be quasi-hereditary. Accordingly, we focus on the case $\mu \geq 1/2$ in our experiments.

We develop several approaches to improve the running time of the above algorithm and we detail them in Sections 3 and 4: First, we consider upper bounds on the density we can achieve when we are given P. If the upper bound is smaller than the given μ, then we can terminate branching early. We modify a known upper bound [11], obtaining two new variants. Second, we develop scoring functions to determine which vertex should be chosen as active vertex and which of its neighbors should be included into P first, so to quickly find solutions. Finally, we also employ several further "early termination" rules (either finding a connected μ-clique of the desired order or deciding that there is none), improved branching rules, as well as several heuristic tricks that speed up the computation of the upper bounds, for example.

In Section 5 we report our experimental findings. Briefly, we find that our branching approach for connected μ-cliques is competitive with the state of the art algorithm for possibly disconnected μ-cliques. The upper bound k imposed on the solution order (which is incremented until we face a no-instance) seems to be crucial to limit the search space. Using this approach we find optimal connected μ-cliques for several real-world instances for the first time. Furthermore, we find that a very simple bound performs best, since the upper bounds are often applied without avail. Due to lack of space, we defer proofs to a full version of this article.

Related Work. Finding a μ-clique of order k in a given graph is a decision version of Densest k-Subgraph, where we seek to find a k-vertex subgraph with the maximum number of edges. This problem is NP-hard even on graphs with maximum degree three [7]. Moreover, it is W[1]-hard with respect to k [6] and thus unlikely to be solvable in $f(k) \cdot n^{O(1)}$ time. Under the Unique Games Conjecture there is no polynomial-time constant-factor approximation algorithm [13]. Finding μ-cliques with k vertices remains NP-hard for every rational μ [12]. On the positive side, finding μ-cliques of maximum order is tractable on graphs with small maximum degree and on graphs with few high-degree vertices [9].

We are aware of two experimental studies for finding large μ-cliques via exact algorithms. Pattillo et al. [12] develop two mixed-integer programming (MIP) formulations for this problem, which were used to solve several real-world instances with up to 154 vertices with CPLEX. Pajouh et al. [11] instead implemented an algorithm which implicitly enumerates vertex subsets. They developed an easy-to-compute upper bound for the number of edges induced by any extension of a vertex set P to one with k vertices. Their algorithm seems to be the state of the art, improving on the MIP formulation in almost all test instances. Hence, we use it as a main reference point here. Note that, in contrast to our algorithm, both algorithms may report *disconnected* μ-cliques.

There is also a large body of work on heuristic algorithms for finding μ-cliques (see [1, 2, 16], for example) as well as heuristics and exact algorithms for other concepts of dense subgraphs (see Balasundaram and Pajouh [4] for a survey).

Preliminaries. We consider only undirected and simple graphs $G = (V, E)$ where $V = V(G)$ denotes the vertex set and $E = E(G)$ denotes the edge set. Unless stated otherwise, n denotes the number of vertices, also called *order* of the graph, and m the number of edges of G. The open neighborhood of a vertex v is denoted by $N(v)$. The degree of a vertex v is denoted by $\deg(v) := |N(v)|$. For a vertex set $S \subseteq V$, we use $N_S(v) := N(v) \cap S$ and $\deg_S(v) := |N_S(v)|$ to denote the neighborhood and degree restricted to S. Furthermore, we use $G[S] := (S, \{\{u, v\} \in E \mid \{u, v\} \subseteq S\})$ to denote the subgraph of G *induced by S*. The *degeneracy* of a graph G is the smallest integer d such that every subgraph of G has a vertex of degree at most d.

2 Connected μ-cliques and Quasi-Heredity

We now study some properties of connected μ-cliques. The property of being a μ-clique, without the connectivity constraint, is not hereditary [10, 12]. That is, there are μ-cliques G such that some induced subgraph of G is not a μ-clique. Being a μ-clique is, however, quasi-hereditary, that is, every μ-clique G or order n has an induced subgraph of order $n-1$ which is a μ-clique. This is implied by the following which slightly extends [10, Proposition 6.3.2] and [12, Proposition 2].

Lemma 1. *Let $G = (V, E)$ be a graph with density exactly μ and let v be a vertex in G. Then, $G[V \setminus \{v\}]$ has density at least μ if and only if* $\deg(v) \leq 2m/n$.

Thus, removing a vertex of minimum degree in a μ-clique yields a μ-clique, implying the quasi-heredity of μ-cliques.

The argument for μ-cliques does not extend easily to *connected* μ-cliques: it could be the case that all vertices v with $\deg(v) \leq 2m/n$ are cut-vertices. Moreover, it is not hard to check that additionally demanding connectedness does not yield a hereditary graph property (consider a clique with a degree-one vertex attached to it). Thus, it is interesting to know whether connected μ-cliques are at least quasi-hereditary. Somewhat surprisingly, this depends on μ: for large μ we observe quasi-heredity whereas for small μ this is impossible.

Theorem 1. *If $\mu \geq 1/2$, then "being a connected μ-clique" is quasi-hereditary.*

In contrast, for $\mu < 1/2$, we obtain a family of counterexamples, showing that we can use quasi-heredity safely only when $\mu \geq 1/2$.

Theorem 2. *For any fixed rational $\mu = a/b$ such that $0 < \mu < 1/2$ and b is odd, "being a connected μ-clique" is not quasi-hereditary.*

3 Upper Bounds

In this section we detail several upper bounds that are used in the algorithm. We start with a previously known upper bound on the order of the μ-clique that depends on the number of edges m and number of vertices n in the graph G.

Proposition 1 (Edge bound [12]). *If $G[S]$ is a μ-clique in a connected graph G, then $|S| \leq \left(\mu + 2\sqrt{(\mu+2)^2 + 8(m-n)\mu}\right)/2\mu$.*

This upper bound obviously also applies to connected μ-cliques. In the course of the algorithm, some vertices of the input graph G are discarded in some recursive branches. Thus m and n decrease in these branches and the bound may then show that no μ-clique of order k exists. While the bound is easy to compute, it rarely leads to early termination.

The following bounds are based on the strategy to gradually extend the "pivot" set P. The aim is to decide whether it is still possible to extend P to a μ-clique of order k. In the following, let $\ell := k - |P|$ denote the number of vertices that we still need to add. Moreover, for a vertex set $S \subseteq V$ let $\mathrm{m}(S)$ denote the number of edges in $G[S]$. Pajouh et al. [11] proved the following.

Proposition 2 (Inner *P*-bound [11]). *Let $G = (V, E)$ be a graph and $P \subseteq V$ a vertex subset. Then, for any $S \supseteq P$ with $|S| - |P| = \ell$, we have*

$$\mathrm{m}(S) \leq \mathrm{m}(P) + \frac{1}{2}\sum_{v \in P} \min\{\deg_{V\setminus P}(v), \ell\} + \frac{1}{2}\sum_{i=1}^{\ell}\Big(\deg_P(v_i) + \min\{\deg_{V\setminus P}(v_i), \ell - 1\}\Big),$$

where $v_1, \ldots, v_\ell \in V \setminus P$ exhibit the largest values of

$$(\deg_P(v_i) + \min\{\deg_{V\setminus P}(v), \ell - 1\})/2.$$

Note that the degree of the vertices in P can be large, and hence, the sum over all $v \in P$ does not make a good estimate on the number of edges between P and $S \setminus P$ in this case. We now aim to make this estimate from "outside" of P instead. This often yields a better bound because $|P|$ is usually relatively small in the course of the algorithm.

Proposition 3 (Outer *P*-bound). *Let $P \subseteq V$ be a vertex set in $G = (V, E)$. Then for any $S \supseteq P$ with $|S| - |P| = \ell$, we have*

$$\mathrm{m}(S) \leq \mathrm{m}(P) + \sum_{i=1}^{\ell} \Big(\deg_P(v_i) + \min\{\deg_{V \setminus P}(v_i), \ell - 1\}/2 \Big),$$

where $v_1, \ldots, v_\ell \in V \setminus P$ exhibit the largest values of

$$\deg_P(v_i) + \min\{\deg_{V \setminus P}(v_i), \ell - 1\}/2.$$

By replacing the estimate of the edges contained in $G[S \setminus P]$ by the trivial upper bound $\binom{\ell}{2}$, we get the following.

Proposition 4 (Simple *P*-bound). *Let $P \subseteq V$ be a vertex set in $G = (V, E)$. Then for any $S \supseteq P$ with $|S| - |P| = \ell$, we have*

$$\mathrm{m}(S) \leq \mathrm{m}(P) + \binom{\ell}{2} + \sum_{i=1}^{\ell} \deg_P(v_i),$$

where $v_1, \ldots, v_\ell \in V \setminus P$ exhibit the largest values of $\deg_P(v_i)$.

While the simple P-bound is the least tight of these three P-bound variants, it is also the one with the least computational overhead. It thus is a crucial feature of our algorithm (see Section 5).

4 Algorithm and Heuristic Improvements

We now describe our algorithm in detail, including several heuristic speed-ups; a pseudocode is shown in Algorithm 1. As outlined in the introduction, we maintain a partial solution P throughout the execution of the algorithm as well as an active vertex v. Initially, P contains a single vertex (we try all possibilities). We successively either add a neighbor of v to P (trying all possibilities) or make v inactive, meaning that no further neighbors of v should be added to P. Inactive vertices are maintained in a set $I \subseteq P$. The procedure is terminated if P reaches size k or all vertices are inactive. As previously shown, this strategy finds a connected μ-clique with k vertices if there is one [9]. After each step of either adding a vertex to P or making a vertex inactive, we check whether the bounds from Section 3 imply that no k-vertex μ-clique containing P exists.

Our general strategy to find the largest μ-clique is to apply Algorithm 1 with successively increasing k. Due to quasi-heredity, once the algorithm asserts that there is no k-vertex μ-clique subgraph, then there is also none with more than k vertices. Next, we describe several speed-up tricks.

Algorithm 1: Find μ-clique

```
   Input: A graph $G$, $k \in \mathbb{N}$, $1/2 \leq \mu \leq 1$
   Output: A connected $\mu$-clique in $G$ of order $k$ if there is one, otherwise $\bot$.
1  foreach $v \in V(G)$ do
2      Recurse($G$, $\{v\}$, $\emptyset$, $v$)
3      Remove $v$ from $G$
4  return $\bot$
5  Procedure Recurse($G$, $P$, $I$, $a$)
6      if $|P| = k$ and $G[P]$ is a $\mu$-clique then return $P$
7      if $|P| = k$ and $G[P]$ is not a $\mu$-clique then break
8      if edge bound, simple $P$-bound, or outer $P$-bound are violated then break
9      foreach $u \in N(a) \setminus P$ do
10         Recurse($G$, $P \cup \{u\}$, $I$, $a$)
11         Remove $u$ from $G$
12     $I \leftarrow I \cup \{a\}$
13     if $P = I$ then break
14     Remove all vertices in $N(a) \setminus P$ from $G$, choose $w \in P \setminus I$, and set $a \leftarrow w$
15     Recurse($G$, $P$, $I \cup \{u\}$, $a$)
```

4.1 Simple Early Termination Rules and Improved Branching

The goal of the following modifications is to avoid branching (Line 9, Algorithm 1) if a solution can be obtained greedily or if some branches are symmetric to others that have been already explored.

*Simple Rules.*We use two greedy termination rules. First, if at some time in Algorithm 1 the graph G is a connected μ-clique, then we can obtain a k-vertex μ-clique using Theorem 1 by greedily deleting a non-cut vertex of minimum degree. Second, if adding $k-|P|$ edges to $G[P]$ would yield a μ-clique, then it suffices to simply check whether the connected component containing P is large enough.

Pending Trees. The latter observation can be extended to any *pending tree* on P, that is, an induced tree T in G containing exactly one vertex v of P such that deleting v cuts T from the rest of the graph. We avoid branching on vertices in pending trees as follows. Adding ℓ vertices from such a tree to P adds exactly ℓ edges. Hence, any solution containing pending tree vertices is found by first branching on the vertices that are not in pending trees and then applying the simple check described above. Hence, after computing the set of all pending trees, we can restrict the branching step in Line 9 of Algorithm 1 to vertices *not* contained in any pending tree.

Twins. We call two vertices u and v *twins* if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. While we cannot assume that, if a vertex is in a solution, then also all its twins are, we do have the following property. Given $P \subseteq V(G)$, if there is no k-vertex μ-clique that contains P and a vertex $v \in V(G) \setminus P$, then there is no μ-clique containing P and any of the twins of v in $V(G) \setminus P$. Note that after Line 10 in Algorithm 1 we know that no k-vertex μ-clique containing u exists. Hence, we may not only remove u in Line 11, but also all its twins in $V(G) \setminus P$. In order to do this, we compute the set of twins for each vertex in the beginning. (Note that two twins in a graph remain twins after deleting any subset of vertices.)

Pre-evaluation of the Modifications. Since the simple rules above can be computed very quickly, we enabled them in all variants of Algorithm 1 we tested. Regarding pending trees and twins, we found that their benefits overlapped strongly in our benchmark instances of Section 5. That is, enabling both at the same time did not yield meaningful speed-up over the variants in which only one of them was enabled. Hence, we enabled only the twin modification, which showed a slightly greater reduction in calls to `Recurse`.

4.2 Order of adding vertices to P

We now consider the order in which vertices are added to the partial solution P in Lines 1 and 9 of Algorithm 1. Intuitively, for a yes-instance, we would like to order the vertices in such a way that a solution is discovered within only few branches. This approach is followed in our *optimistic* ordering. The optimistic ordering also serves as a greedy heuristic by determining which vertices to add in the first descent in the recursion. For a no-instance, however, it is better to add vertices to P that lead to sparse partial solutions, so that it can be easily determined that these vertices are not in a solution. Subsequently, these vertices will be removed in Lines 3 and 11, truncating the search space. This approach is followed in our *pessimistic* ordering.

Basic Optimistic Ordering. The optimistic ordering is based on two simple heuristics. The first heuristic, `MaxDegKeep`, starts with a highest-degree vertex and then recursively selects neighbors of already selected vertices with highest degree, until k vertices are selected. The second one, `MinDegDel`, instead removes vertices of minimum degree—omitting cut vertices—until only k vertices remain. In preliminary experiments we observed that the inequality $d \geq \Delta/10$ seems to be a good predictor on which of the two heuristics performed better. Here, Δ is the maximum degree of the input graph, and d is its degeneracy. If $d \geq \Delta/10$, then `MaxDegKeep` worked better and `MinDegDel` otherwise.

Based on the above observation we define score(v) for each vertex v and we first add vertices with the higher scores to P in Lines 1 and 9. If $d \geq \Delta/10$, then score(v) is simply the degree of v in the input graph. If $d < \Delta/10$, then score(v) is the largest degree encountered when deleting vertices of minimum degree from the input graph until v is deleted.

Breaking Ties. Most of our instances, and most of the instances we expect to be encountered in practice, fall into the "$d < \Delta/10$" category. Since often these graphs have thousands of vertices and small maximum score, many vertices receive the same score. Thus, we try to break ties by modifying the score. We tested two alternatives for tie-breaking: a) the number of neighbors with larger score, and b) the number of edges in the neighborhood of the vertex. Interestingly, pre-evaluation showed that a) performed worse than without tie breaking, increasing running times and calls to `Recurse`. Tie breaker b) showed improvements on some instances, so we opted to test only b) in Section 5.

Neighborhood-based Scoring. As the set P grows, it intuitively becomes more important to add many edges to $G[P]$ when adding vertices. Thus, in a variant of the vertex scoring, for each vertex $v \in V \setminus P$, we add $|N_P(v)|$ to score(v).

Pessimistic Ordering. The pessimistic ordering is obtained by essentially reversing the ordering given by the score of the vertices. That is, we first consider vertices, which we expect to not be in a μ-clique of order k. We break ties among them by considering first vertices with the fewest number of edges in the neighborhood. In the neighborhood-based scoring for the pessimistic variant, we score vertices with the fewest neighbors in P highest.

4.3 Application of the Upper Bounds

We now list several optimizations we employed for Line 8 of Algorithm 1.

– Since the edge bound and simple P-bound can be computed quickly, we determined in preliminary experiments that it is always better to enable both bounds. In particular, the simple P-bound has to be enabled in any good configuration of the algorithm. Thus, both bounds are always enabled in Section 5.

– The simple P-bound and outer P-bound rely on knowing the number of neighbors in P for each vertex outside of P. To amortize the corresponding computation cost, this information is kept and updated in each call to `Recurse`.

– The outer P-bound is based on certain values for each vertex. Then, from the ℓ largest of these values, it derives an upper bound on the density achievable in a k-vertex subgraph containing P. Compared to the trivial approach of computing all values, a considerable speed-up can be achieved by computing the values one-by-one, and only as long as the upper bound derived from the ℓ largest values computed so far still is below μ.

5 Implementation and Experiments

The algorithm described in Section 4 was implemented in Haskell and compiled using ghc version 7.4.1; the source code and test data is freely available, see `http://fpt.akt.tu-berlin.de/connected-mu-clique`. All experiments were run on an Intel Xeon E5-1620 computer with 4 cores at 3.6 GHz and 64 GB RAM. The operating system was Debian GNU/Linux 6.0. Our implementation does not use multiprocessing capabilities, however, up to four experiments were run on the machine at once (one on each core). Unless stated otherwise, the time limit was one hour.

We performed the following experiments. First, for $\mu = 0.7$, we compared all configuration variants of our algorithm in order to identify the best ones. The comparison is done on 25 real-world and benchmark instances. Then, we compare our algorithm to the one of Pajouh et al. [11] on a representative subset of the real-world instances for several values of μ. Finally, we perform experiments on random graphs to determine more precisely the limits of our algorithm and of the algorithm of Pajouh et al. [11].

Table 1. Reported μ-clique orders and running times (s) of the algorithm configurations across the test data set. The "# solved" column denotes the number of instances solved to optimality. Optimality is also indicated by a star on the μ-clique order. For any variant, the "# max k" column denotes the number of graphs where the largest μ-clique order was achieved among all variants. This is also indicated by a bold μ-clique order. A bold time means that this variant was the fastest among all variants that solved this instance.

	#max k	#solved	*ERDOS-99-2*	*Human-all*	*GEOM-0*	*email-Enron*	*Acker-all*	*Acker-pc*
(O)-(↑)-(B)	24	6	**15*** (2850.14)	**25** (3600.0)	**28** (3600.0)	**58** (3600.0)	**25** (3600.0)	**17*** (199.03)
(O)-(↑)-(B,N)	24	6	**15*** (2850.27)	**25** (3600.0)	**28** (3600.0)	**58** (3600.0)	**25** (3600.0)	**17*** (201.31)
(↑)-(B,N)	24	6	**15*** **(2809.09)**	**25** (3600.0)	**28** (3600.0)	**58** (3600.0)	**25** (3600.0)	**17*** (190.91)
(↓)-(B)	6	3	**15*** (3319.91)	16 (3600.0)	**28** (3600.0)	20 (3600.0)	20 (3600.0)	**17*** **(64.36)**

5.1 Finding the best Algorithm Variants

Our test bed consists of 25 networks overall. Of these networks, 12 are from the Second DIMACS Implementation Challenge, chosen to represent hard instances for dense subgraph problems, and 13 are real-world social and biological networks, chosen from several applications to represent instances one might face in practice. Table 1 shows the performance of four algorithm variants (including the three best) on a subset of these instances. Each variant is represented by a string in which O denotes that the outer P-bound is enabled, B denotes that tie-breaking is enabled, N denotes that neighborhood-based scoring is enabled, $\uparrow$ denotes the optimistic ordering and $\downarrow$ denotes the pessimistic ordering.

Our observations are roughly as follows: For instances with larger maximum k, the optimistic ordering outperforms the pessimistic one. Those with small maximum k are solved slightly faster with pessimistic ordering. The outer P-bound usually does not reduce search tree size significantly but it runs fast enough to have only a small negative effect on running times. Tie-breaking allows to discover several μ-cliques in instances of medium difficulty which otherwise seem to be hard to find. The effect of neighborhood-based scoring is negligible.

5.2 Comparison with a Previous Approach

We compared our algorithms with an exact branch and bound algorithm for finding μ-cliques by Pajouh et al. [11]. In the following, we denote their algorithm by BB. (Recall that BB may report disconnected μ-cliques.) For the

Table 2. Largest μ-cliques found by the branch and bound algorithm (BB) by Pajouh et al. [11], and by our algorithm (O)-(↑)-(B,N), indicated by A1, and (O)-(↑)-(N), indicated by A2. Bold values represent *maximum* connected μ-clique orders as reported by the corresponding algorithm.

	$\mu = 0.55$			$\mu = 0.7$			$\mu = 0.9$		
	BB	A1	A2	BB	A1	A2	BB	A1	A2
Acker-all	32	32	32	25	25	25	15	**15**	**15**
Human-all	41	37	39	31	26	27	20	20	18
email-Enron	86	81	68	55	58	44	29	29	21
ERDOS-99-2	20	19	20	14	**14**	14	9	**9**	**9**
GEOM-0	39	32	32	30	28	28	23	**23**	**23**
wiki-Vote	104	84	103	65	62	61	31	28	26

comparison, we chose several real-world instances from the test bed above and the three values of $\mu = 0.55, 0.7, 0.9$. The results are shown in Table 2. In terms of quickly finding large solutions, BB performs better than our algorithm but the favor shifts towards ours for larger μ. Our algorithm could verify optimality for several instances with larger values of μ, whereas BB was never able to verify optimality within the time limit. While Table 2 shows results with the outer P-bound enabled; In some instances, enabling the outer P-bound reduces the number of calls to `Recurse`, but this is rare.

5.3 Evaluation on Random Instances

Erdős-Rényi Random Graphs. For each combination of $n = 10, 20, \ldots, 1200$ vertices and edge probability $p = 0.05, 0.1, 0.2$, we generated 15 Erdős-Rényi random graphs. The average running times of our algorithm variant (↑)-(B,N) and algorithm BB are shown in Fig. 1 for those n, where all 15 instances were solved to optimality within 20 minutes. For $p = 0.1$ and $p = 0.2$, the reported maximum μ-cliques of our algorithm were around ten at the cut-off points due to the time limit. Our algorithm clearly outperforms BB in terms of verifying optimality on these instances. Furthermore, the differences get more pronounced as p gets smaller, that is, the graphs get sparser.

Random Small-World Graphs with Planted μ-cliques. In order to assess the order of the retrieved μ-cliques, we generated random networks with a planted μ-clique of order 10, 20, and 30. For each order, we created six networks, two networks with 500 vertices, two with 1000 vertices, and two with 2000 vertices. First, we sample a μ-clique of the appropriate order using the Erdős-Rényi model with edge probability $p = \mu$ and ensuring density at least μ. Then, we add vertices according to the Barabási-Albert model, making a new vertex adjacent to $\lfloor k/i \rfloor$ previous ones with probability proportional to their degrees. Herein, k is the μ-clique order and $i = 2$ for the first graph and $i = 4$ for the second one. Table 3 shows our results. If the planted μ-clique has order 10, our algorithm outperforms BB as it can exactly solve these instances. For planted μ-cliques of order

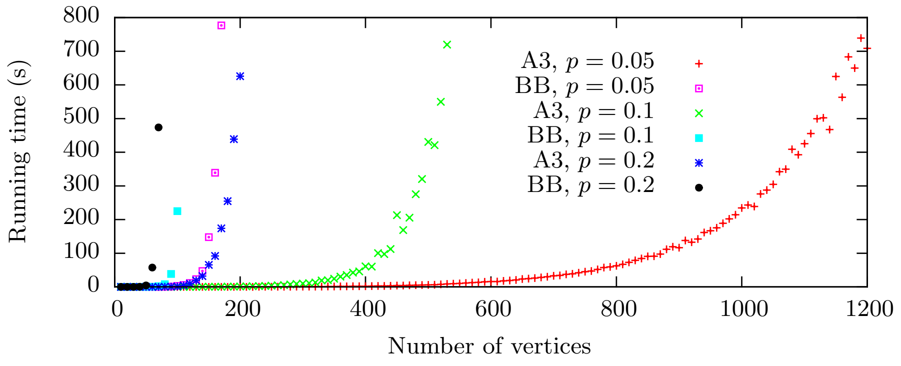

Fig. 1. Running times for varying order and edge probability p of Erdős-Rényi graphs. A3 denotes our algorithm in variant (↑)-(B,N).

Table 3. Comparison of the retrieved μ-clique orders in random small-world networks. Here, k denotes the order of the planted μ-clique, n denotes the order of the input graph, and each * denotes an instance that was solved within the time limit.

k	n	(↑)-(B,N)	(↑)-(N)	(O)-(↑)-(N)	(O)-(↑)-(B,N)	BB
	500	11.0 (**)	11.0 (**)	11.0 (**)	11.0 (**)	10.0
10	1000	12.0 (**)	12.0 (**)	12.0 (**)	12.0 (**)	11.0
	2000	12.0 (**)	12.0 (**)	12.0 (**)	12.0 (**)	11.0
	500	21.5 (*)	21.5 (*)	21.0	21.0	21.5
20	1000	21.0	21.0	21.0	21.0	21.0
	2000	20.5	20.5	20.5	20.5	20.5
	500	26.0	26.0	25.5	25.5	30.5
30	1000	29.0	29.0	29.0	29.0	31.0
	2000	30.0	30.0	30.0	30.0	30.0

30, BB outperforms our algorithm. For order 20, they behave roughly the same. The algorithm variants without outer P-bound perform slightly better than the ones with the outer P-bound, tie-breaking has no effect in these instances.

6 Conclusion and Outlook

We proposed a new algorithm for finding connected μ-cliques which is based on searching for successively larger solutions. As known upper bounds are apparently not tight enough, this strategy seems to be imperative for bounding the search space in each iteration. Using this approach, we could verify optimality for several real-world instances.

In ongoing work, we developed two tighter upper bounds. They showed promising reductions of the search space for some instances. However, they require more computational overhead which increases the overall computation time. It is thus interesting to improve the corresponding implementations and to find easily checkable conditions on when the bounds might apply.

Acknowledgment. Manuel Sorge and Kolja Stahl gratefully acknowledge support by Deutsche Forschungsgemeinschaft (DFG), project DAPA, NI 369/12.

References

[1] J. Abello, P. M. Pardalos, and M. G. C. Resende. On maximum clique problems in very large graphs. In *External Memory Algorithms and Visualization*, volume 50 of *DIMACS*, pages 119–130. AMS, 1999.

[2] J. Abello, M. G. C. Resende, and S. Sudarsky. Massive quasi-clique detection. In *Proc. 5th LATIN*, volume 2286 of *LNCS*, pages 598–612. Springer, 2002.

[3] G. D. Bader and C. W. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(1): 2, 2003.

[4] B. Balasundaram and F. M. Pajouh. Graph theoretic clique relaxations and applications. In *Handbook of Combinatorial Optimization*, pages 1559–1598. Springer, 2013.

[5] V. Boginski, S. Butenko, and P. M. Pardalos. On structural properties of the market graph. In *Innovations in Financial and Economic Networks*, New Dimensions in Networks, pages 29–45. Edward Elgar Publishing, Cheltenham, England, 2003.

[6] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141 (1&2):109–131, 1995.

[7] U. Feige and M. Seltser. On the densest k-subgraph problem. Technical report, The Weizmann Institute, Department of Applied Math and Computer Science, 1997.

[8] F. Harary. The maximum connectivity of a graph. *Proceedings of the National Academy of Science of the United States of America*, 48(7):1142–1146, 1962.

[9] C. Komusiewicz and M. Sorge. Finding dense subgraphs of sparse graphs. In *Proc. 7th IPEC*, volume 7535 of *LNCS*, pages 242–251. Springer, 2012. Long version to appear under the title "An Algorithmic Framework for Fixed-Cardinality Optimization in Sparse Graphs Applied to Dense Subgraph Problems" in *Discrete Applied Mathematics*, Elsevier.

[10] S. Kosub. Local density. In *Network Analysis*, volume 3418 of *LNCS*, pages 112–142. Springer, 2004.

[11] F. M. Pajouh, Z. Miao, and B. Balasundaram. A branch-and-bound approach for maximum quasi-cliques. *Annals of Operations Research*, 216(1): 145–161, 2014.

[12] J. Pattillo, A. Veremyev, S. Butenko, and V. Boginski. On the maximum quasi-clique problem. *Discrete Applied Mathematics*, 161(1-2):244–257, 2013.

[13] P. Raghavendra and D. Steurer. Graph expansion and the unique games conjecture. In *Proc. 42nd STOC*, pages 755–764. ACM, 2010.

[14] P. Turán. On an extremal problem in graph theory. *Matematikai és Fizikai Lapok*, 48(436-452):137, 1941.

[15] A. Wagner and D. A. Fell. The small world inside large metabolic networks. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1478):1803–1810, 2001.

[16] J. Zhang and Y. Chen. Monte Carlo algorithms for identifying densely connected subgraphs. *Journal of Computational and Graphical Statistics*, 2014. Available Online.

7 Appendix

7.1 Omitted Proofs from Section 2

Proof (Lemma 1). The statement is obviously true for $n \leq 2$, thus assume that $n > 2$ in the following. Let μ' denote the density of $G[V \setminus \{v\}]$. Then, $\mu \leq \mu'$ if and only if

$$\begin{aligned}
& \frac{2m}{n(n-1)} \leq \frac{2(m-\deg(v))}{(n-1)(n-2)} \\
\Leftrightarrow \quad & \frac{2m}{n} \leq \frac{2(m-\deg(v))}{(n-2)} \\
\Leftrightarrow \quad & 2mn - 4m \leq 2mn - 2\deg(v)n \\
\Leftrightarrow \quad & -4m \leq -2\deg(v)n \\
\Leftrightarrow \quad & \frac{2m}{n} \geq \deg(v).
\end{aligned}$$

□

In the following, we call vertices with degree less or equal than $2m/n$ *density-deletable* which indicates that deleting them results in a graph which fulfills the density condition (Lemma 1).

Proof (Theorem 1). The claim is trivially true for $\mu = 1$. Thus, let $G = (V, E)$ be a connected μ-clique, $1/2 \leq \mu < 1$. We show that G contains a vertex v such that v is not a cut-vertex and v is density-deletable, that is, $\deg(v) \leq 2m/n$. We distinguish two cases.

Case 1: G does not contain a cut-vertex. Let v be the vertex with minimum degree in G. Obviously, $\deg(v) \leq 2m/n$. Then, by Lemma 1, $G[V \setminus \{v\}]$ is a μ-clique. By the case assumption, it is also connected.

Case 2: G has a cut-vertex u. Let $G_1, \ldots, G_i$, $i \geq 2$ be the connected components of $G[V \setminus \{u\}]$. Let G_1 be the connected component with the minimum number of vertices and observe that G_1 has at most $(n-1)/2$ vertices. Therefore, the degree of every vertex v in G_1 is at most $(n-1)/2$: the vertex v has at most $(n-1)/2 - 1$ neighbors in G and at most one further neighbor, namely u. Since G is a μ-clique and $\mu \geq 1/2$ we have $2m/n \geq (n-1)/2$. By the observation above, each vertex v in G_1 has degree at most $(n-1)/2$ and $(n-1)/2 \leq 2m/n$ by the discussion above. Consequently, every vertex v in G_1 is density-deletable.

At least one vertex in G_1 is not a cut-vertex in G. Deleting this vertex results in a connected μ-clique. □

Proof (Theorem 2). Let $\mu = a/b$ be as described above and assume without loss of generality that $a > 3$. We construct a connected μ-clique G on n vertices and then show that every $(n-1)$-vertex subgraph of G has either density less than μ or two connected components. The graph G will consist of two disjoint subgraphs G' that are connected by a path on three vertices. These vertices will be the only density-deletable vertices, and hence, any proper density-μ subgraph will be disconnected by Lemma 1.

Let $\epsilon := 1/2 - a/b$ and observe that $\epsilon > 0$ by the assumption of the statement. To construct G, first choose n to be an odd integer such that $n > 1/\epsilon$, n is a multiple of b, and $m := (a/b) \cdot \binom{n}{2}$ is an even integer. Eventually, G will have exactly m edges and thus density exactly μ. Let us show that an integer n as above exists. Note that, if n fulfills the second condition, to fulfill the third condition it suffices that $(n-1)/2$ is even.

There is clearly an odd integer x that fulfills the first two conditions. Since x is odd, we have $x = 2y + 1$ for some integer y. Thus, if the third condition is not fulfilled by setting $n := x$, then $(x-1)/2 = y$ is odd. Then, however, setting $n := 3x = 6y + 3$ gives $(n-1)/2 = (3x-1)/2 = (6y+2)/2 = 3y + 1$. Since y is odd, this number is even. Thus, there exists an n which fulfills all three conditions.

The next step is to construct a graph G^* on $\lfloor n/2 \rfloor = (n-1)/2$ vertices and $m/2$ edges. Such a graph G^* exists if the desired number of edges does not exceed the number of all possible edges. That is,

$$\begin{aligned} & & \mu \cdot \binom{n}{2} \cdot \frac{1}{2} &< \binom{n-1}{2} \\ \Leftrightarrow & & \frac{\mu \cdot n \cdot (n-1)}{4} &< \frac{(n-1)(n-2)}{8} \\ \Leftrightarrow & & \mu \cdot n &< \frac{(n-2)}{2} \\ \Leftrightarrow & & \frac{n}{2} - \epsilon \cdot n &< \frac{n}{2} - 1. \end{aligned}$$

The last inequality is fulfilled since $n > 1/\epsilon$. Hence, such a graph G^* exists. Consider now the average degree δ over all vertices of G^*. We have

$$\delta = 2 \cdot \frac{m}{2} \cdot \frac{1}{(n-1)/2} = \mu \cdot \binom{n}{2} \cdot \frac{2}{n-1} = \mu \cdot n.$$

Since n is a multiple of b, the average degree is thus an integer and since $a > 3$ we have $\delta \geq 3$. For an integer average degree $\delta < p - 1$, there exists a δ-connected δ-regular graph on p vertices [8], meaning that removing any set of less than δ edges leaves a connected graph and each vertex has degree exactly δ. Thus, we let G^* be a δ-connected δ-regular graph on $(n-1)/2$ vertices.

Now, we perform a final modification to G^* in order to obtain a new graph: Pick a vertex u and two nonadjacent neighbors v and w of u. Since G^* is connected and not a clique, such a vertex u exists. Now remove the edges $\{u, v\}$ and $\{u, w\}$ and add the edge $\{v, w\}$. Call the resulting graph G', and observe that G' has $m/2 - 1$ edges and all vertices in G' have degree δ, except one vertex, which has degree $\delta - 1$. Moreover, G' is connected since G^* is δ-connected, $\delta \geq 3$, and thus removing two edges does not make it disconnected.

We now obtain G as follows: Take the disjoint union of two copies of G', add one further vertex v', and make this vertex adjacent to the uniquely defined vertex of minimum degree in each copy of G'. The graph G has $2(m/2-1)+2) = m$ edges and thus it has density exactly μ. There are three vertices in G whose degree is at most the average degree: v^* and its two neighbors. These three vertices are cut-vertices. Consequently, every density-deletable vertex is a cut-vertex which implies that G has no $(n-1)$-vertex subgraph that is a connected μ-clique. □

7.2 Omitted Proofs from Section 3

Proof (Proposition 3). Clearly, we have

$$\mathrm{m}(S) = \mathrm{m}(P) + \sum_{v\in P} \deg_{S\setminus P}(v)/2 + \sum_{v\in S\setminus P} (\deg_{S\setminus P}(v) + \deg_P(v))/2.$$

Grouping the terms counting edges between $S \setminus P$ and P, we get

$$\begin{aligned}\mathrm{m}(S) &= \mathrm{m}(P) + \sum_{v\in S\setminus P} \Big(\deg_P(v) + \deg_{S\setminus P}(v)/2\Big) \\ &= \mathrm{m}(P) + \sum_{v\in S\setminus P} \Big(\deg_P(v) + \min\{\deg_{S\setminus P}(v), \ell-1\}/2\Big) \\ &\le \mathrm{m}(P) + \sum_{i=1}^{\ell} \Big(\deg_P(v_i) + \min\{\deg_{V\setminus P}(v_i), \ell-1\}/2\Big),\end{aligned}$$

where $v_1, \dots, v_\ell \in V \setminus P$ exhibit the largest values of

$$\deg_P(v_i) + \min\{\deg_{V\setminus P}(v_i), \ell-1\}/2.$$

□

7.3 Further Upper bounds

Here, we describe the two additional upper bounds mentioned in Section 6.

A worst-case example for the three P-bounds described Section 3 occurs if there are ℓ vertices V' adjacent to all vertices of P such that each vertex of V' has $\ell-1$ further degree-one neighbors. Then all P-bound variants above evaluate to the trivial upper bound of $\mathrm{m}(S) \le \mathrm{m}(P) + |P|\cdot\ell + \binom{\ell}{2}$. To avoid this behavior we now look at pairs of vertices outside of P instead of just singletons. The bound we obtain in this manner is given below. Let us first define the following function describing a value that each vertex pair may contribute to $m(S)$. Here, $A \triangle B := (A\setminus B)\cup(B\setminus A)$ denotes the symmetric difference of two sets A and B.

Definition 2. *For $u, v \in V \setminus P, u \ne v$, define*

$$\mathrm{val}(u, v) := \deg_P(u) + \deg_P(v) + I(u, v) + D(u, v)/2 + |N(v) \cap \{u\}|,$$

where

$$I(u,v) := \min\{|N_{V\setminus P}(u) \cap N_{V\setminus P}(v)|, \ell - 2\}, \text{ and}$$

$$D(u,v) := \min \begin{cases} |(N_{V\setminus P}(u) \triangle N_{V\setminus P}(v)) \setminus \{u,v\}|, \\ \max\{0, \ell - 2 - |N_{V\setminus P}(u) \cap N_{V\setminus P}(v)|\}. \end{cases}$$

Proposition 5 (Pairwise P-bound). *Let $P \subseteq V$ be a vertex set in $G = (V,E)$. Then for any $S \supseteq P$ with $|S| - |P| = \ell \geq 2$, we have*

$$\mathrm{m}(S) \leq \mathrm{m}(P) + \frac{1}{\ell - 1} \sum_{i=1}^{\binom{\ell}{2}} \mathrm{val}(u_i, v_i),$$

where $\{u_1, v_1\}, \ldots, \{u_{\binom{\ell}{2}}, v_{\binom{\ell}{2}}\}$ are the $\binom{\ell}{2}$ vertex pairs in $V \setminus P$ that exhibit the largest values of $\mathrm{val}(u_i, v_i)$.

Proof. First, we clearly have

$$\mathrm{m}(S) = \mathrm{m}(P) + \mathrm{m}(S \setminus P, P) + \mathrm{m}(S \setminus P), \tag{1}$$

where, for two vertex sets $A, B \subseteq V$, $\mathrm{m}(A,B)$ denotes the number of edges between A and B in G. To get a new upper bound of the left hand side we now consider the contribution of each pair of vertices $u, v \in S \setminus P$ to the last two terms. We first focus on the last term. If there is at least one vertex pair in $S \setminus P$ (that is, $\ell \geq 2$), we have

$$\begin{aligned} \mathrm{m}(S \setminus P) &= \frac{1}{2(\ell-1)} \sum_{\{u,v\} \subseteq S \setminus P} \deg_{S\setminus P}(u) &(2)\\ &= \frac{1}{4(\ell-1)} \left(\sum_{\{u,v\} \subseteq S \setminus P} \deg_{S\setminus P}(u) + \sum_{\{u,v\} \subseteq S \setminus P} \deg_{S\setminus P}(v) \right) &(3)\\ &= \frac{1}{2(\ell-1)} \sum_{\{u,v\} \subseteq S \setminus P} \deg_{S\setminus P}(u) + \deg_{S\setminus P}(v) &(4) \end{aligned}$$

Note that Eq. (4) is equivalent to

$$\mathrm{m}(S \setminus P) = \frac{1}{2(\ell-1)} \sum_{\{u,v\} \subseteq S \setminus P} 2|N_{S\setminus P}(u) \cap N_{S\setminus P}(v)| + |N_{S\setminus P}(u) \triangle N_{S\setminus P}(v)|. \tag{5}$$

Now we replace parts of the right hand side of this equation by minima of two terms. This replacement does not affect Eq. (5) but it will have an effect later, when we replace the (unknown) vertex pairs that the sum ranges over by those vertex pairs that maximize this sum. Note that $|N_{S\setminus P}(u) \cap N_{S\setminus P}(v)|$ is at most

$\ell - 2$; hence we may replace this summand with the minimum of the two, which is exactly $I(u, v)$. Next, observe that

$$|N_{S\setminus P}(u) \triangle N_{S\setminus P}(v)| = |(N_{S\setminus P}(u) \triangle N_{S\setminus P}(v)) \setminus \{u, v\}| + 2|N(v) \cap \{u\}|.$$

Furthermore,

$$\begin{aligned} |(N_{S\setminus P}(u) \triangle N_{S\setminus P}(v)) \setminus \{u, v\}| &\leq \ell - 2 - |N_{S\setminus P}(u) \cap N_{S\setminus P}(v)| \\ &\leq \max\{0, \ell - 2 - |N_{S\setminus P}(u) \cap N_{S\setminus P}(v)|\}, \end{aligned}$$

and this means that we can replace the left hand side by the minimum of the two sides above which is exactly $D(u, v)$. Plugging both minima into Eq. (5), we arrive at

$$\mathrm{m}(S \setminus P) = \frac{1}{2(\ell - 1)} \sum_{\{u,v\}\subseteq S\setminus P} 2I(u, v) + D(u, v) + 2|N(v) \cap \{u\}|.$$

Considering the second to last term in the sum in Eq. (1), notice that, similarly to Eq. (4), we have

$$\mathrm{m}(S \setminus P, P) = \frac{1}{\ell - 1} \sum_{\{u,v\}\subseteq S\setminus P} \deg_P(u) + \deg_P(v).$$

Hence, we can write Eq. (1) as

$$\begin{aligned} &\mathrm{m}(S) = \mathrm{m}(P) + \\ &\frac{1}{\ell - 1} \sum_{\{u,v\}\subseteq S\setminus P} \deg_P(u) + \deg_P(v) + I(u, v) + D(u, v)/2 + |N(v) \cap \{u\}|. \end{aligned} \quad (6)$$

When applying the bound we do not know S, so we replace the pairs u, v by the $\binom{\ell}{2}$ vertex pairs from V that exhibit the largest value of the term in the sum. The term in the sum is calculated by replacing each occurrence of S with V. This only increases the right hand side of Eq. (6). $\square$

The overhead for computing the pairwise P-bound is higher than for the other P-bounds. Hence, it is crucial to find an efficient implementation. A straightforward implementation of the bound iterates over all pairs and computes val for each pair, yielding a worst-case running time of $\Omega(n^3)$. Apart from heuristic improvements in Section 4, one can also improve on the straightforward running time if the input graph is sparse. The improvement is achieved by considering only the present edges and their contribution to the val-values as follows.

Lemma 2. *The pairwise P-bound can be computed in $O(n^2 + m \cdot n)$ time.*

Proof (Sketch). First, we construct three arrays A, B, C with $\binom{|V\setminus P|}{2}$ entries each, one for each pair in $\binom{V\setminus P}{2}$. Eventually, entry $A[\{u, v\}]$ will hold $\mathrm{val}(u, v)$, $B[\{u, v\}]$ will hold $|N_{V\setminus P}(u) \cap N_{V\setminus P}(v)|$ and $C[\{u, v\}]$ will hold $|(N_{V\setminus P}(u) \triangle$

$N_{V \setminus P}(v)) \setminus \{u, v\}|$. First, for each vertex $v \in V \setminus P$ we add $\deg_P(v)$ to the entry $A[\{u, v\}]$ for each $u \in V(G) \setminus P$. This takes $O(n^2)$ time. After that, it remains to take care of the remaining three terms in val. The last one is trivial, so we omit it. Note that the entry $A[\{u, v\}]$ can be easily computed in $O(1)$ time once we have $B[\{u, v\}]$ and $C[\{u, v\}]$. To compute the arrays B and C, we consider each edge $\{x, y\}$ and do the following. For each $w \in N(y) \setminus (P \cup \{x\})$ we increment $B[\{w, x\}]$, and for each $w \in (V(G) \setminus (P \cup N(y) \cup \{x\})$ we increment $C[\{w, x\}]$; this takes $O(n)$ time. Symmetrically, for each $w \in N(x) \setminus (P \cup \{y\})$ we increment $B[\{w, y\}]$, and for each $w \in (V(G) \setminus (P \cup N(x) \cup \{y\})$ we increment $C[\{w, y\}]$. Note that, after we have done that for all edges in $G[V \setminus P]$, B and C hold the required values. Finally, it remains to find the $\binom{\ell}{2}$ pairs with the largest values in A, which can be done by sorting the array in $O(n^2)$ time using bucket sort. □

Our final bound is based on Turán graphs.

Theorem 3 ([14]). *The n-vertex graph without a clique of order k with the maximum number of edges is $K_{n_1,n_2,\ldots,n_{k-1}}$, that is, the complete $(k-1)$-partite graph with part sizes $n_1, n_2, \ldots, n_k$, such that $|n_i - n_j| \leq 1$.*

Let $T(n, k)$ denote the number of edges in this graph. In our algorithm, we use the degeneracy of the graph to upper-bound the maximum clique order in G.

Proposition 6 (Turán bound). *Let $P \subseteq V$ be a vertex set in $G = (V, E)$ such that $G[V \setminus P]$ has degeneracy d. Then for any $S \supseteq P$ with $|S| - |P| = \ell$, we have*

$$\mathrm{m}(S) \leq \mathrm{m}(P) + T(\ell, d+2) + \sum_{i=1}^{\ell} \deg_P(v_i),$$

where $v_1, \ldots, v_\ell \in V \setminus P$ exhibit the largest values of $\deg_P(v_i)$.

Proof. Clearly, $\mathrm{m}(S) \leq \mathrm{m}(P) + \mathrm{m}(S \setminus P) + \mathrm{m}(S \setminus P, P)$. By assumption $G[V \setminus P]$ has degeneracy d. Hence, $G[S \setminus P]$ has degeneracy d and does not have a clique of order $d+2$. By Proposition 6, we thus have $\mathrm{m}(S \setminus P) \leq T(\ell, d+2)$. The term $\mathrm{m}(S \setminus P, P)$ is upper-bounded as in the previous bounds. □

7.4 Additional Tables

Table 4. Graph parameters for the graphs in the test bed. Herein, δ denotes the minimum degree, Δ the maximum degree, ρ the density, h the h-index, d the degeneracy and c the number of connected components.

Graph	n	m	δ	Δ	ρ	h	d	c
acker-schmalwand-all	5704	12627	1	438	7.76e-4	43	12	128
acker-schmalwand-pc	1907	2870	1	437	1.57e-3	22	9	84
acker-schmalwand-p	1872	2828	1	437	1.61e-3	22	9	82
Human-all	14771	67297	1	8649	6.16e-4	106	19	51
Human-pc	12385	45159	1	8560	5.88e-4	81	16	42
Human-p	12342	44739	1	8560	5.87e-4	81	16	39
Worm-all	3613	6828	1	524	1.04e-3	35	10	73
brock200_1	200	14834	130	165	0.74	145	134	1
brock200_2	200	9876	78	114	0.49	99	84	1
brock200_4	200	13089	112	147	0.65	128	117	1
brock800_2	800	208166	472	566	0.65	516	486	1
brock800_4	800	207643	481	565	0.64	514	485	1
hamming8-4	256	20864	163	163	0.63	163	163	1
keller4	171	9435	102	124	0.64	106	102	1
keller5	776	225990	560	638	0.75	565	560	1
p_hat1500-1	1500	284923	157	614	0.25	456	252	1
p_hat1500-2	1500	568960	335	1153	0.50	759	504	1
p_hat300-2	300	21928	59	229	0.48	148	98	1
p_hat700-1	700	60999	75	286	0.24	207	117	1
ERDOS-97-2	5482	8972	1	257	5.97e-4	48	9	11
ERDOS-98-2	5816	9505	1	273	5.62e-4	49	9	12
ERDOS-99-2	6094	9939	1	276	5.35e-4	50	9	11
email-Enron	36692	183831	1	1383	2.73e-4	195	43	1065
wiki-Vote	7115	100762	1	1065	3.98e-3	186	53	24

r

Table 5. Comparison of different upper-bounding configurations with respect to running times and calls to `Recurse` for the largest value of k where Algorithm 1 finished in all configurations.

	(O.)-(↑)-(B,N)	(O.P.)-(↑)-(B,N)	(O.T.)-(↑)-(B,N)	(↑)-(B,N)
acker-schmalwand-all.txt	**139** (0.055602)	**139** (2.821461)	**139** (1.148493)	**139** **(0.039045)**
acker-schmalwand-pc.txt	**19** (0.002461)	**19** (0.015258)	**19** (0.028803)	**19** **(0.001861)**
acker-schmalwand-p.txt	**19** (0.003124)	**19** (0.014814)	**19** (0.027822)	**19** **(0.001765)**
Human-all.txt	**27957** (24.87847)	**27957** (604.414709)	**27957** (1051.471294)	**27957** **(22.956882)**
Human-pc.txt	**201** (0.237403)	**201** (3.680115)	**201** (5.486204)	**201** **(0.132176)**
Human-p.txt	**168** (0.197254)	**168** (3.078197)	**168** (4.558326)	**168** **(0.109572)**
Worm-all.txt	**39** (0.011335)	**39** (0.019999)	**39** (0.164047)	**39** **(0.01096)**
GEOM-0.gra	798 (0.180793)	798 (2.014961)	**674** (3.853945)	798 **(0.171525)**
wiki-Vote.txt	**6166** (4.407739)	**6166** (123.791093)	**6166** (166.930533)	**6166** **(3.415636)**
email-Enron.txt	**9658** (27.29542)	**9658** (1009.216216)	**9658** (908.459545)	**9658** **(25.183931)**
keller5	**586** (0.364413)	**586** (25.365735)	**586** (5.590281)	**586** **(0.360889)**
brock200_2	**2139** (0.084695)	**2139** (0.108529)	**2139** (0.975137)	**2139** **(0.081451)**
brock200_4	15529736 (767.480289)	15529736 (1706.657423)	**15102171** (2611.785801)	15529736 **(715.640794)**
brock800_2	**133369** (32.954492)	**133369** (45.014115)	**133369** (807.83723)	**133369** **(32.028619)**
brock800_4	**68253** (16.418768)	**68253** (23.470246)	**68253** (376.504885)	**68253** **(15.947536)**
hamming8-4	**478428** (22.677156)	**478428** (37.239283)	**478428** (101.640664)	**478428** **(21.585677)**
keller4	32842014 (1054.628497)	32842014 (2172.704948)	**30593684** (3009.737481)	32842014 **(981.066648)**
p_hat1500-1	**9524** **(2.756867)**	**9524** (3.287472)	**9524** (242.492648)	**9524** (2.901863)
p_hat1500-2	**384** (0.600911)	**384** (12.716766)	**384** (16.818033)	**384** **(0.545022)**
p_hat300-2	309153 (19.665408)	309153 (62.831661)	**203813** (55.571791)	309153 **(18.045269)**
p_hat700-1	**2278** (0.257848)	**2278** (0.303483)	**2278** (10.455954)	**2278** **(0.250244)**
p_hat700-2	67532 (17.410172)	67532 (94.852461)	**60602** (149.97654)	67532 **(16.379092)**
ERDOS-97-2.gra	**112** (0.037607)	**112** (0.049824)	**112** (0.66515)	**112** **(0.022862)**
ERDOS-98-2.gra	**84** (0.023426)	**84** (0.031959)	**84** (0.550267)	**84** **(0.018176)**
ERDOS-99-2.gra	**250** (0.080471)	**250** (0.122949)	**250** (1.7124)	**250** **(0.072934)**