Computing Complexity Measures of Degenerate Graphs

Pål Grønås Drange
University of Bergen, Norway

Patrick Greaves
Birkbeck, University of London, UK

Irene Muzi
Birkbeck, University of London, UK

Felix Reidl
Birkbeck, University of London, UK

Abstract

We show that the VC-dimension of a graph can be computed in time $n^{\lceil \log d+1 \rceil} d^{O(d)}$, where d is the degeneracy of the input graph. The core idea of our algorithm is a data structure to efficiently query the number of vertices that see a specific subset of vertices inside of a (small) query set. The construction of this data structure takes time $O(d2^d n)$, afterwards queries can be computed efficiently using fast Möbius inversion.

This data structure turns out to be useful for a range of tasks, especially for finding bipartite patterns in degenerate graphs, and we outline an efficient algorithm for counting the number of times specific patterns occur in a graph. The largest factor in the running time of this algorithm is $O(n^c)$, where c is a parameter of the pattern we call its *left covering number*.

Concrete applications of this algorithm include counting the number of (non-induced) bicliques in linear time, the number of co-matchings in quadratic time, as well as a constant-factor approximation of the ladder index in linear time.

Finally, we supplement our theoretical results with several implementations and run experiments on more than 200 real-world datasets – the largest of which has 8 million edges – where we obtain interesting insights into the VC-dimension of real-world networks.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases vc-dimension, datastructure, degeneracy, enumerating

Digital Object Identifier 10.4230/LIPIcs.IPEC.2023.14

Related Version *Full Version*: `https://arxiv.org/abs/2308.08868`

Supplementary Material
Software (Source code): `https://github.com/microgravitas/mantis-shrimp`
archived at `swh:1:dir:74c67c5dffc03a1c1383be7947d7def7cc0d4798`

Funding *Pål Grønås Drange*: Supported by the Research council of Norway, grant number 329745: *Machine Teaching for Explainable AI.*

1 Introduction

Our work began with the simple question: What is the Vapnik–Chervonenkis (VC) dimension of real-world networks? That is, what is the largest vertex set X such that every subset $X' \subseteq X$ is the neighbourhood (when restricted to X) of some vertex in the network?

This parameter, developed in the context of learning theory, happens to be extremely useful in the theory of sparse graphs (e.g. [11]) and is one possible method of capturing the "complexity" of an object. It is therefore a natural statistic to consider when a) trying to categorise networks and b) identifying structural properties that can be leveraged to design efficient algorithms.

As the best-known general algorithm to compute the VC-dimension of a graph takes $O(n^{\log n})$ time, and in fact the problem being LOGNP-complete [16], we investigated whether a better algorithm is possible if we assume our input graph to be sparse, more precisely, to be d-degenerate. This choice is motivated by the observation that the degeneracy for most real-world networks is small (see e.g. [7] and our results in the full version [9]).

Our first achievement is an algorithm that computes the VC-dimension of a d-degenerate graph in time $O(n^{\lceil \log d+1 \rceil} d^{O(d)})$. A core concept is a novel data structure which enables us to efficiently query the size of the intersection of several neighbourhoods for a small set of vertices, described in Section 3, which we use to quickly determine whether a given candidate set is shattered by its neighbours.

But the general idea of this algorithm can be generalised to other bipartite "patterns" like bicliques, co-matchings, and ladders (defined in Section 2.2). These objects are also closely related to notions of "complexity" of graphs. They appear, for example, in the study of graph width measures [10] and algorithm design for sparse classes [13] (see also there for connections to stability theory). Our general pattern-finding algorithm presented in Section 3 can count bicliques in linear time, co-matchings in quadratic time and find partial ladders in linear time, see Section 4 for these and further results.

Dense structures like cliques or bicliques are famously important in the analysis of networks, and we suggest that co-matchings and ladders might be of similar interest – but without a program to compute them, we cannot hope for these statistics to be trialled in practice. We therefore implemented algorithms to compute the VC-dimension, ladder index, maximum biclique[1] and maximum co-matching of a graph. To establish their practicality, we ran these four algorithms on 206 real-world networks from various sources, see Section 5. The VC-dimension algorithm in our experiments terminated within 10 minutes on networks with up to ∼33K vertices, the other three on networks up to ∼93K vertices. This is already squarely in the region of "practical" for certain types of networks and we believe that with further engineering – in particular to improve space efficiency – our implementation can be used to compute these statistics on much larger networks.

Prior work. We briefly mention a few relevant previous articles on the subject. Eppstein, Löffler, and Strash [12] gave an algorithm for enumerating maximal cliques in d-degenerate graphs in $O(dn3^{d/3})$ time, i.e., fixed-parameter tractable time when parameterized by the degeneracy. They also give experimental results showing that their algorithm works well on large real-world networks. Bera, Pashanasangi, and Seshadhri [1], extending the classic result

[1] There are probably faster programs to compute bicliques in practice, we compute this statistic here as a baseline.

by Chiba and Nishizeki [6], show that for all patterns H of size less than six, we can count the number of appearances of H in a d-degenerate graph G in time $O(m \cdot d^{k-2})$, where m is the number of edges in G and k is the number of vertices in H. Recently, Bressan, and Roth [3] gave algorithms for counting copies of a graph H in a d-degenerate graph G in time $f(d,k) \cdot n^{\mathbf{im}(H)} \log n$, for some function f, where k again is the number of vertices in H, n the number of vertices in G, and $\mathbf{im}(H)$ is the size of a largest induced matching in H.

2 Preliminaries

For an integer k, we use $[k]$ as a short-hand for the set $\{0, 1, 2, \ldots, k-1\}$. We use blackboard bold letters like $\mathbb{X}$ to denote sets X associated with a total order $<_{\mathbb{X}}$. The *index function* $\iota_{\mathbb{X}} \colon X \to \mathbb{N}$ maps elements of X to their corresponding position in $\mathbb{X}$. We extend this function to sets via $\iota_{\mathbb{X}}(S) = \{\iota_{\mathbb{X}}(s) \mid s \in S\}$. For any integer $i \in [|X|]$ we write $\mathbb{X}[i]$ to mean the ith element in the ordered set. An *index set* I for $\mathbb{X}$ is simply a subset of $[|X|]$ and we extend the index notation to sets via $\mathbb{X}[I] \coloneqq \{\mathbb{X}[i] \mid i \in I\}$. We write $\pi(H)$ for the set of all permutations of H.

For a graph G we use $V(G)$ and $E(G)$ to refer to its vertex- and edge-set, respectively. We used the short hands $|G| \coloneqq |V(G)|$ and $\|G\| \coloneqq |E(G)|$.

An *ordered graph* is a pair $\mathbb{G} = (G, <)$ where G is a graph and $<$ a total ordering of $V(G)$. We write $<_{\mathbb{G}}$ to denote the ordering for a given ordered graph and extend this notation to the derived relations $\leqslant_{\mathbb{G}}$, $>_{\mathbb{G}}$, $\geqslant_{\mathbb{G}}$.

We use the same notations for graphs and ordered graphs, additionally we write $N^-(u) \coloneqq \{v \in N(u) \mid v <_{\mathbb{G}} u\}$ for the *left neighbourhood* and $N^+(u) \coloneqq \{v \in N(u) \mid v >_{\mathbb{G}} u\}$ for the *right neighbourhood* of a vertex $u \in \mathbb{G}$. We further use $d^-_{\mathbb{G}}(u)$ and $d^+_G(u)$ for the left and right degree, as well as $\Delta^-(\mathbb{G}) \coloneqq \max_{u \in \mathbb{G}} d^-_{\mathbb{G}}(u)$ and $\Delta^+(\mathbb{G}) \coloneqq \max_{u \in \mathbb{G}} d^+_{\mathbb{G}}(u)$. We omit the graphs in the subscripts if clear from the context.

A graph G is d-degenerate if there exists an ordering $\mathbb{G}$ such that $\Delta^-(\mathbb{G}) \leqslant d$. An equivalent definition is that a graph is d-degenerate if every subgraph has a vertex of degree at most d. The number of edges in a d-degenerate graph is bounded by dn and many important sparse graph classes – bounded treewidth, planar graphs, graphs excluding a minor – have finite degeneracy. The degeneracy ordering of a graph can be computed in time $O(n+m)$ [15], and $O(dn)$ for d-degenerate graphs.

Let $\mathcal{F} \subseteq 2^U$ be a set family over U. We define the intersection of a set family with set $X \subseteq U$ as $\mathcal{F} \cap X \coloneqq \{F \cap X \mid F \in \mathcal{F}\}$. A set $X \subseteq U$ is then *shattered* by $\mathcal{F}$ if $\mathcal{F} \cap X = 2^X$. The *graph representation* of a set family $\mathcal{F}$ is the bipartite graph $G(\mathcal{F}) = (\mathcal{F}, U, E)$ where for each $F \in \mathcal{F}$ and $x \in U$ we have the edge $Fx \in E$ iff $x \in F$. In the other direction, we define for a graph G its *neighbourhood set system* $\mathcal{F}(G) \coloneqq \{N(v) \mid v \in G\}$.

The Vapnik–Chervonenkis dimension (VC-dimension) of a set family $\mathcal{F} \subseteq 2^U$ is the size of the largest set in U that is shattered by $\mathcal{F}$ and we write this quantity as $\mathbf{vc}(\mathcal{F})$. The VC-dimension of a graph G is defined as the VC-dimension of its neighbourhood set system, i.e. $\mathbf{vc}(G) \coloneqq \mathbf{vc}(\mathcal{F}(G))$.

2.1 Set dictionaries

In the following we will make heavy use of data structures that model functions of the form $f \colon 2^U \to \mathbb{Z}$ for some universe U. Since the arguments in our use-case are assumed to be small, we use prefix-tries [17] in our theoretical analysis (see notes on practical implementations below):

▶ **Definition 1** (Subset dictionary). *Let U be a set and let $\mathbb{U}$ be an arbitrary total order of U. A* subset dictionary *D over U associates a key $X \subseteq U$ with an integer $D[X]$ by storing the sequence $\mathbb{X}$ of X under $<_{\mathbb{U}}$ in a prefix trie.*

Accordingly, insertion/update/deletion of a value for a key X takes time $O(|X|)$ *if* we can assume the key X to be present in some canonical order. Our algorithms all work on graphs imbued with a (degeneracy) ordering and we will sort the left-neighbourhood $N^-(\bullet)$ of each vertex according to this global ordering, which we will simply call "sorting the left-neighbourhoods" for brevity. Subsets of these left-neighbourhoods are assumed to inherit this ordering, which covers all operations that we will need in our algorithms, which in conclusion means that we can assume that all sets used as keys in subset dictionaries have a canonical ordering.

Unless otherwise noted, we will use the convention that $D[X] = 0$ for all keys X that have not been inserted into D.

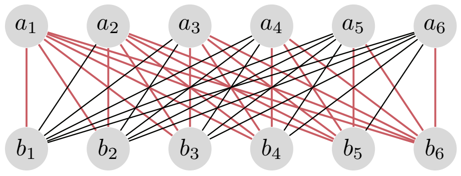
2.2 Bipartite patterns and left-covers

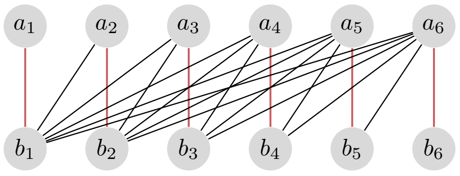
▶ **Definition 2** (Pattern). *A* pattern *H is a complete graph whose edges are partitioned into sets B, R, and W (black, red and white). We say that a graph G* contains *H (or H* appears in *G) if there exists a vertex set $S \subseteq V(G)$ and a bijection $\phi\colon V(H) \to S$ such that $uv \in B \implies \phi(u)\phi(v) \in E(G)$ and $uv \in R \implies \phi(u)\phi(v) \notin E(G)$.*

We say that a pattern H is bipartite *if the vertex set of H can be partitioned into two sets X, Y such that all edges inside of X and inside of Y are white.*

For a vertex $v \in V(H)$ we write $N(v)$ to denote its neighbours according to the black edge relation only. An *ordered* pattern $\mathbb{H}$ is a pattern whose vertex set comes with a linear order $<_{\mathbb{H}}$. Given a vertex $v \in \mathbb{H}$, we write $N^-(u) := \{v \in N(u) \mid v <_{\mathbb{H}} u\}$.

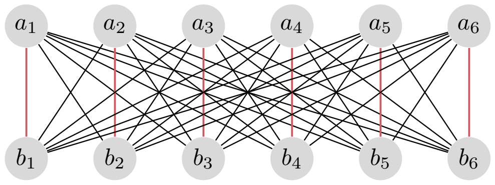
A *ladder* (sometimes called a chain graph) L_n of size n is a bipartite pattern defined on two vertex sequences $A = (a_i)_{i\in[n]}$ and $B = (b_i)_{i\in[n]}$, where $a_ib_j \in B$ if $i > j$ and $a_ib_j \in R$ otherwise. Note that for any $1 \leqslant l \leqslant r \leqslant n$ the subgraph induced by the sequences $(a_i)_{i\in[l,r]}$ and $(b_i)_{i\in[l,r]}$ induces a ladder. A *semi-ladder* $\tilde{L}_n$ has the same black edges, but only the edges a_ib_i, $i \in [n]$ are red. All the remaining edges are white:



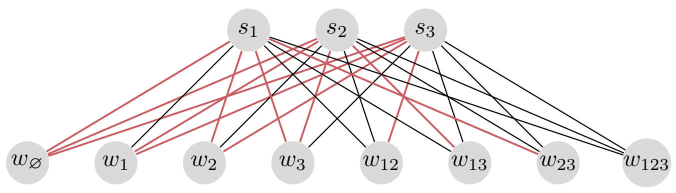


The *Ladder index* of a graph G is the largest n such that G contains the pattern L_n.

A *co-matching* $\overline{M}_n$ (also called *crown*) has black edges a_ib_j for $i \neq j$ and red edges a_ib_i for $i \in [n]$.



Finally, the *shattered* pattern U_n of size n has a side S (the *shattered* set) of size n and a side W (the *witness* set) of size 2^n. We index the vertices of W by subsets $I \subseteq S$, then the vertex w_I has black edges into I and red edges into $S \setminus I$:



▶ **Definition 3** (Left-cover, left-covering number). *Given an ordered bipartite pattern $\mathbb{H}$ with bipartition (X, Y), a* left-cover *is a set of vertices $C \subseteq V(\mathbb{H})$ such that either $X \subseteq N^-(C) \cup C$ or $Y \subseteq N^-(C) \cup C$. The* left-covering number $\operatorname{lc}(\mathbb{H})$ *is the minimum size of a left cover of $\mathbb{H}$. For an (unordered) pattern H we define its left-covering number as*

$$\operatorname{lc}(H) \coloneqq \max_{\mathbb{H} \in \pi(H)} \operatorname{lc}(\mathbb{H}).$$

Note that we include the covering set C itself in the cover, this is necessary since for a given ordering of a pattern some vertices might not have right neighbours and can therefore not be covered by left neighbourhoods.

The left-covering number of a pattern is the first important measure that will influence the running time of the main algorithm presented later. The second important measure relates to the number of non-isomorphic "half"-ordered patterns we can obtain from a bipartite pattern, that is, how many distinct objects we find by ordering one partition. A useful tool to concretise this notion is the following function:

▶ **Definition 4** (Signature). *Let H be a bipartite pattern with bipartition (X, Y) and let $\mathbb{Z}$ be an ordering of $Z \in \{X, Y\}$. Then the* signature $\sigma_{\mathbb{Z}}(H)$ *is defined as the multiset*

$$\sigma_{\mathbb{Z}}(H) \coloneqq \{\{\iota_{\mathbb{Z}}(N(u)) \mid u \in (X \cup Y) \setminus Z\}\}.$$

For orderings $\mathbb{Z}, \mathbb{Z}' \in \pi(Z)$ we define the equivalence relation

$$\mathbb{Z} \sim_H \mathbb{Z}' \iff \sigma_{\mathbb{Z}}(H) = \sigma_{\mathbb{Z}'}(H).$$

▶ **Definition 5** (Half-ordering asymmetry). *Given a bipartite pattern H with bipartition (X, Y) and a partite set $Z \in \{X, Y\}$, we define the* half-ordering asymmetry $\operatorname{hoa}(H, Z)$ *as the number of equivalence classes under the $\sim_H$ relation*

$$\operatorname{hoa}(H, Z) \coloneqq |\pi(Z)/\sim_H|.$$

We further define the half-ordering asymmetry of H as

$$\operatorname{hoa}(H) \coloneqq \max\{\operatorname{hoa}(H, X), \operatorname{hoa}(H, Y)\}.$$

Alternatively, $\operatorname{hoa}(H, Z) \coloneqq |\{\sigma_{\mathbb{Z}}(H) \mid \mathbb{Z} \in \pi(Z)\}|$.

3 A general pattern-finding algorithm

We first describe a general-purpose algorithm for finding patterns in degenerate graphs. Afterwards, we will describe more specialised algorithms using similar ideas to find specific patterns.

▶ **Theorem 6.** *Let G be a d-degenerate graph and let H be a bipartite pattern with bipartition (X, Y) where $|X| \geqslant |Y|$. Then after a preprocessing time of $O(|X|^{\operatorname{lc}(H)}|H|! + d2^d n)$, we can in time $O\big(n^{\operatorname{lc}(H)}(4d\operatorname{lc}(H))^{|X|}d|X|^3 \operatorname{hoa}(H)\big)$ count how often H appears in G.*

The main ingredient of our algorithm will be the following data structure:

▶ **Theorem 7.** *Let $\mathbb{G}$ be an ordered graph on n vertices with degeneracy d. After a preprocessing time of $O(d2^d n)$, we can, for any given $S \subseteq V(G)$, compute a subset dictionary Q_S in time $O(|S|2^{|S|} + d|S|^2)$ which for any $X \subseteq S \subseteq V(G)$ answers the query*

$$Q_S[X] \coloneqq \big|\{v \in G \mid S \cap N(v) = X\}\big|$$

in time $O(|X|)$.

▶ **Lemma 8.** *Let $\mathbb{G}$ be an ordered graph with degeneracy d. Then in time $O(d2^d n)$ we can compute a subset dictionary R over $V(G)$ which for any $X \subseteq V(G)$ answers the query*

$$R[X] \coloneqq \big|\{v \in G \mid X \subseteq N^-(v)\}\big|$$

in time $O(|X|)$.

Proof. Given $\mathbb{G}$ as input, we compute R as follows:

```
Initialize $R$ as an empty trie storing integers;
for $u \in \mathbb{G}$ do
    for $X \subseteq N^-(u)$ do
        $R[X] \leftarrow R[X] + 1$    // Non-existing keys are treated as zero
return $R$;
```

Note that every update of the data structure with key X takes time $O(|X|)$, since $|X| \leqslant d$ it follows that the total initialisation time is bounded by $O(d2^d n)$. ◀

▶ **Lemma 9.** *Let $\mathbb{G}$ be an ordered graph with degeneracy d and let $S \subseteq V(G)$. If we assume the subset dictionary R of Lemma 8 is given, we can construct in time $O(|S|2^{|S|} + d|S|^2)$ a subset dictionary Q_S over S which for $X \subseteq S$ answer the query*

$$Q_S[X] \coloneqq \big|\{v \in G \mid S \cap N(v) = X\}\big|$$

in time $O(|X|)$.

Proof. We first construct an auxiliary subset dictionary $\hat{Q}$ which for $X \subseteq S$ answers the query

$$\hat{Q}_S[X] \coloneqq \big|\{v \in G \mid S \cap N^-(v) = X\}\big|$$

in time $O(|X|)$. We first prove the following claim which implies that $\hat{Q}_S$ is the (upwards) Möbius inversion of R over S and hence can be computed in time $O(|S|2^{|S|})$ using Yate's algorithm [18, 14, 2].

▷ Claim 10. $\big|\{v \in G \mid S \cap N^-(v) = X\}\big| = \sum_{X \subseteq Y \subseteq S} (-1)^{|Y \setminus X|} R[Y]$,

Proof. First consider $v \ngeqslant_{\mathbb{G}} X$. Then X cannot be contained in $N^-(v)$ and therefore v does not contribute to the left-hand side. Note that v is not counted by $R[Y]$ for any $Y \supseteq X$, therefore v does not contribute to the right-hand side.

Consider therefore $v \geqslant_{\mathbb{G}} X$. First, assume that $S \cap N^-(v) = X$ and therefore v contributes to the left-hand side. Then v is counted on the right-hand side exactly once by the term $R[X]$ which has a positive sign.

Consider now v with $S \cap N^-(v) \neq X$. If $X \not\subseteq N^-(v)$, then v does not contribute to the left-hands side and it is not counted by any term $R[Y]$, $Y \supseteq X$ on the right-hand side. We are therefore left with vertices v where $I \coloneqq S \cap N^-(v)$ satisfies $X \subset I$. Note that I is counted by every term $R[Y]$ with $X \subseteq Y \subseteq I$. Since

$$\sum_{X \subseteq Y \subseteq I} (-1)^{(Y \setminus X)} = \sum_{0 \leqslant k \leqslant |I \setminus X|} (-1)^k \binom{|I \setminus X|}{k} = 0$$

we conclude that these counts of v cancel out and contribute a sum-total of zero to the right-hand side. This covers all cases and we conclude that the claim holds. ◁

It remains to be shown how the query $Q_S[X]$ can be computed using $\hat{Q}_S[X]$. To this end, consider a vertex $v \in G$ where $S \cap N(v) \neq S \cap N^-(v)$ as these contribute to $\hat{Q}_S[X]$ but must not be counted by $Q_S[X]$. Note that any such vertex must be contained in $N^-(S)$ since v has at least one right-neighbour in S. Accordingly, we apply the following correction to $\hat{Q}_S[X]$:

```
Let $Q_S = \hat{Q}_S$
for $u \in N^-(S)$ do
    $Q_S[N^-(u) \cap S] \leftarrow Q_S[N^-(u) \cap S] - 1$
    $Q_S[N(u) \cap S] \leftarrow Q_S[N(u) \cap S] + 1$
```

This correction takes time $O(d|S|^2)$. ◀

We are now ready to describe the pattern-counting algorithm.

Proof of Theorem 6. The problem is trivial for $|X| = 1$ since then the pattern is either a single edge or anti-edge. Thus assume $|X| \geqslant 2$ in the following, in particular for the running time calculations.

We first compute the left-covering number $\mathrm{lc}(H)$ by simply brute-forcing all orderings of H in time $O(|H|! \cdot \max\{|X|, |Y|\}^{\mathrm{lc}(H)}) = O(|H|!|X|^{\mathrm{lc}(H)})$. At the same time, whenever we find that a specific ordering $\mathbb{H}$ has a minimal left-covering of X, then we add the signature $\sigma_{\mathbb{X}}(H)$ with $\mathbb{X} \coloneqq \mathbb{H}[X]$ to a collection $\mathcal{X}$. Similarly, if we find that a minimal left-covering in $\mathbb{H}$ covers Y we add the signature $\sigma_{\mathbb{Y}}(H)$ with $\mathbb{Y} \coloneqq \mathbb{H}[Y]$ to a collection $\mathcal{Y}$. We will later use that $|\mathcal{X}| \leqslant \mathrm{hoa}(H, X)$ and $|\mathcal{Y}| \leqslant \mathrm{hoa}(H, Y)$.

We now compute an ordering $\mathbb{G}$ for G of degeneracy d in time $O(dn)$, sort the left-neighbourhoods in time $O(d \log d \cdot n)$ time and compute the data structure R as per Lemma 8 in time $O(d2^d n)$. If we want to compute the number of times H appears in G, we further need to initialise a subset dictionary K.

We now iterate through all subsets $C \subseteq V(G)$ of size $\mathrm{lc}(H)$ and for each such set we iterate through all subsets $Z \subseteq N^-_{\mathbb{G}}(C) \cup C$ of size $|X|$ or $|Y|$, in total this takes time $O(n^{\mathrm{lc}(H)}((d+1)\,\mathrm{lc}(H))^{|X|})$. We describe the remainder of the algorithm for a set $X = Z$ of size $|X|$, the procedure for a set Y works analogously. Let $\mathbb{X}$ be the ordering of X in $\mathbb{G}$.

To verify that X can be completed into a pattern H in G, we compute the data structure Q_X in time $O(|X|2^{|X|} + d|X|^2)$ as per Theorem 7. To check whether H exists in G, we iterate through all signatures $\sigma \in \mathcal{X}$ and test whether $Q_X[\mathbb{X}[A]] > 0$ for all index sets $A \in \sigma$, this takes time $O(|\mathcal{X}||X||Y|)$, in total the verification step for X takes time

$$O\big((|X|2^{|X|} + d|X|^2) \cdot |\mathcal{X}||X||Y|\big) = O\big(d|X|^3 2^{|X|}\,\mathrm{hoa}(H)\big)$$

where we used that $|X| \geqslant |Y|$ and $|X| \geqslant 2$. This bound also holds for checking Y since $|\mathcal{X}| + |\mathcal{Y}| \leqslant 2\,\mathrm{hoa}(H)$. Finally, if we exhaust all orderings of H without finding the pattern, we report that it does not exist in G.

To *count* in how many ways X can be extended into the pattern H in G, we compute

$$c_{H,X} \coloneqq \sum_{\sigma \in \mathcal{X}} \prod_{A^{(k)} \in \sigma} \binom{Q_X[\mathbb{X}[A]]}{k}$$

where k denotes the multiplicity of A in the multiset σ. Note, however, that we have to take care not to double-count the contribution of X to the overall count as we might encounter the set X multiple times. To that end, we record the intermediate result by setting $K[X] \coloneqq c_{H,X}$ and we forgo the above computation if X exists already as a key in K. The computation of $c_{H,X}$ and this additional book keeping takes time $O(|X| + |\mathcal{X}||X||Y|)$, in total we arrive at the same running time $O\big(d|X|^3 2^{|X|} \operatorname{hoa}(H)\big)$ like for the decision variant. After exhausting all orderings of H we report back the number of times H appears in G as the sum of all entries of K.

The total running time of either variant of the algorithm is, as claimed,

$$O\Big(|X|^{\operatorname{lc}(H)}|H|! + d2^d n + dn + n^{\operatorname{lc}(H)}\big((d+1)\operatorname{lc}(H)\big)^{|X|} \cdot d|X|^3 2^{|X|} \operatorname{hoa}(H)\Big)$$
$$= O\Big(|X|^{\operatorname{lc}(H)}|H|! + d2^d n + n^{\operatorname{lc}(H)}(4d\operatorname{lc}(H))^{|X|} d|X|^3 \operatorname{hoa}(H)\Big). \qquad ◀$$

4 Concrete applications

4.1 Finding bicliques and co-matchings

We note that $\operatorname{lc}(K_{t,t}) = 1$ and $\operatorname{hoa}(K_{t,t}) = 1$, therefore the application of Theorem 6 gives the following:

▶ **Corollary 11.** *Let G be a d-degenerate graph. Then we can compute the number of biclique patterns $K_{s,t}$ ($s \geqslant t$) in time $O\big(s \cdot (2s)! + d2^d n + n(4d)^s ds^3\big)$.*

Let $\overline{M}_t$ be a co-matching on $2t$ vertices. We will assume in the following that the partite sets of $\overline{M}_t$ are $X \coloneqq (x_1, \ldots, x_t)$ and $Y \coloneqq (y_1, \ldots, y_t)$ so that the edges $x_i y_i$ for $i \in [t]$ are forbidden.

▶ **Lemma 12.** $\operatorname{lc}(\overline{M}_t) = 2$ *and* $\operatorname{hoa}(\overline{M}_t) = 1$.

Proof. Let $\overline{\mathbb{M}}_t$ be an ordering of $\overline{M}_t$ and let z be the last vertex in that order. Then $N^-(z)$ covers all vertices of one partite set except one vertex z'. Thus $\{z, z'\}$ is a left-cover of $\overline{\mathbb{M}}_t$.

To determine the half-ordering asymmetry, note that for *every* ordering $\mathbb{Z}$ of $Z \in \{X, Y\}$ the signature $\sigma_{\mathbb{Z}}(\overline{M}_t)$ is simply the set $\binom{[t]}{t-1}$, so the total number of signatures is one. ◀

▶ **Corollary 13.** *Let G be a d-degenerate graph. Then we can compute the number of co-matching patterns $\overline{M}_t$ in time $O\big(t^2(2t)! + d2^d n + n^2(8d)^t dt^3\big)$.*

4.2 Finding shattered sets

A direct application of Theorem 6 to locate a shattered pattern U_t is unsatisfactory as the running time will include a factor of n^t since $\operatorname{lc}(U_t) = t$. By the following observation, we can bound t by the degeneracy of the graph, but we can greatly improve the running time by further adjusting the algorithm.

▶ **Observation 14.** *Let G be a d-degenerate graph. Then* $\mathbf{vc}(G) \leqslant d + 1$.

Proof. Assume $S \subseteq V(G)$ is shattered by $W \subseteq V(G)$, with $S = |\,\mathbf{vc}(G)|$. Let $W' \subseteq W$ be those witnesses that have $|S|-1$ neighbours in S. Then $G[W' \cup S]$ induces a graph of minimum degree $|S|-1$ and we must have that $|S|-1 \leqslant d$ and accordingly $\mathbf{vc}(G) = |S| \leqslant d+1$. ◀

The core observation that allows further improvements is that many orderings of U_{d+1} have degeneracy *larger* than d and can therefore not appear in a d-degenerate graph. In particular, the ordering in which all witnesses of U_{d+1} appear before the shattered set has degeneracy 2^{d+1} and can therefore be ruled out. We refine this idea further in the following lemma.

▶ **Lemma 15.** *Let $\mathbb{G}$ be a d-degenerate ordering of a graph G. Let G contain the shattered pattern U_t and let $\mathbb{U}_t := \mathbb{G}[U_t]$ be its ordering. Then $\mathrm{lc}(\mathbb{U}_t) \leqslant \lceil \log d + 1 \rceil$. Specifically, we either have that $t \leqslant \lceil \log d + 1 \rceil$ or that $\mathbb{U}_t$ can be covered by $\lceil \log d + 1 \rceil$ witness vertices.*

Proof. Let $S = (s_1, \ldots, s_t)$ and $W = (w_1, \ldots, w_{2^t})$ be the vertices of U_t in G and let the indices of the variables reflect the ordering of the corresponding vertices in $\mathbb{U}_t$.

Partition the set S into $p := \lceil \log d + 1 \rceil$ sets $S_1, \ldots, S_p$ such that each set has size at least $\lfloor t/p \rfloor$ and at most $\lceil t/p \rceil$. For each set S_i define the set of "apex"-witnesses $A_i := \{w \in W \mid N(w) \supset S_i\}$. Note that, for all $i \in [p]$,

$$|A_i| = 2^{|S \setminus S_i|} \geqslant 2^{t - \lceil t/p \rceil} = 2^{\lceil t \frac{p-1}{p} \rceil}.$$

We call a set A_i *good* if $\max_{\mathbb{G}} A_i > \max_{\mathbb{G}} S_i$, that is, at least one apex vertex from A_i can be found to the right of S_i. We now distinguish two cases:

Case 1. All A_i, $i \in [p]$, are good.
It follows that $\mathbb{U}_t$ can be left-covered by taking one vertex from each A_i, $i \in [p]$. We conclude that $\mathrm{lc}(\mathbb{U}_t) \leqslant p = \lceil \log d + 1 \rceil$.

Case 2. Some A_i, $i \in [p]$, is not good.
Let $u = \max_{\mathbb{G}} S_i$ be the last vertex in S_i, note that $A_i \leqslant_{\mathbb{G}} u$ and accordingly $A_i \subseteq N^-(u)$. But then we must have that $|A_i| \leqslant d$ and accordingly that

$$2^{\lceil t \frac{p-1}{p} \rceil} \leqslant d \iff \lceil t \frac{p-1}{p} \rceil \leqslant \log d \implies t \frac{p-1}{p} \leqslant \log d \iff t \leqslant \frac{p}{p-1} \log d$$
$$\iff t \leqslant \frac{\lceil \log d + 1 \rceil}{\lceil \log d + 1 \rceil - 1} \log d = \frac{\log d}{\lceil \log d \rceil} \lceil \log d + 1 \rceil \leqslant \lceil \log d + 1 \rceil.$$

We therefore find that $\mathrm{lc}(\mathbb{U}_t) \leqslant |S| \leqslant \lceil \log d + 1 \rceil$. ◀

▶ **Theorem 16.** *Let G be a d-degenerate graph on n vertices. Then we can determine the VC-dimension of its neighbourhood set system $\mathcal{F}(G)$ in time $O(n^{\lceil \log d + 1 \rceil} d^{d+2} (2d \log d)^{d+1})$.*

Proof. We first compute an ordering $\mathbb{G}$ of G with degeneracy d in time $O(dn)$ and sort all left-neighbourhoods in time $O(d \log d \cdot n)$. Let $p := \lceil \log d + 1 \rceil$ in the following.

Let $\mathbb{U}_t = (S, W)$ be a shattered set of size $t \leqslant d+1$ in $\mathbb{G}$. By Lemma 15 we then have that $\mathrm{lc}(\mathbb{U}_t) \leqslant p$. Therefore to locate the set S we first guess up to p vertices and then exhaustively search through their (closed) left-neighbourhoods in time

$$\binom{n}{p}\binom{dp}{t} \leqslant \left(\frac{en}{p}\right)^p \left(\frac{edp}{t}\right)^t = O\left(n^{\lceil \log d + 1 \rceil} (d \log d)^{d+1}\right).$$

Now that we can locate S we apply Theorem 7 in order to verify that S is indeed shattered: For each candidate set S from the previous step, we compute a subset dictionary Q_S in time $O(|S|2^{|S|} + d|S|^2) = O(d2^d)$ and then check whether $Q_S[X] > 0$ for each $X \subseteq S$. This latter step takes time $O(|S|2^{|S|})$ and is therefore subsumed by the construction time of Q_S. We conclude that the algorithm runs in total time

$$O(d \log d \cdot n) + O(d2^d n) + O\left(n^{\lceil \log d+1 \rceil}(d \log d)^{d+1} \cdot d2^d\right) = O\left(n^{\lceil \log d+1 \rceil} d^{d+2} (2d \log d)^{d+1}\right)$$

as claimed. ◀

We note that the exponent of $\lceil \log d + 1 \rceil$ in the running time is almost tight:

▶ **Theorem 17.** *Graph VC-dimension parameterized by the degeneracy d of the input graph cannot be solved in time $f(d) \cdot n^{o(\log d)}$ unless all problems in* SNP *can be solved in subexponential time.*

Proof. We adapt the W[1]-hardness reduction from k-Clique to VC-dimension by Downey, Evans, and Fellows [8] and combine it with the result by Chen *et al.* [5, 4] which states that k-Clique cannot be solved in time $f(k)n^{o(k)}$ unless all problems in SNP admit subexponential-time algorithms.

Given an instance (H, k) for k-Clique, we construct a graph G as follows. We first create k copies $V_1, \ldots, V_k$ of $V(H)$. For $v \in H$, let us denote its copies by $v^{(1)}, \ldots, v^{(k)}$ with $v^{(i)} \in V_i$ for $i \in [k]$. We now add the following vertices and edges:

- A single isolated vertex w_0,
- a vertex set W_1 which contains one pendant vertex for each $v^{(i)}$, $v \in H$ and $i \in [k]$,
- a vertex set W_2 which for each edge $uv \in H$ contains $\binom{k}{2}$ vertices w^{ij}_{uv}, $i, j \in [k]$, each of which $u^{(i)}$ and $v^{(j)}$ as its only neighbours, and
- a vertex set A which for each index set $I \subseteq [k]$ contains a vertex a_I which is connected to all vertices in V_i for each $i \in I$.

Note that the graph is bipartite with partite sets $\mathcal{V} := V_1 \cup \cdots \cup V_k$ and $\mathcal{W} := W_1 \cup W_2 \cup A$.

Let us first show that if H contains a clique of size k then G contains a shattered set of size k. Let $u_1, \ldots, u_k$ be distinct vertices that form a complete graph in H. We claim that then the set $S := \{u_1^{(1)}, \ldots, u_k^{(k)}\}$ is shattered in G. First, note that for every subset $X \subset S$, $|X| \geqslant 3$, there exists a witness vertex $a \in A$ such that $N(a) \cap S = X$. For the empty set we have the witness w_0, for every singleton subset $\{u\} \subseteq S$ we have that the pendant vertex $p \in N(u) \cap W_1$ witnesses $\{u\}$. Therefore, only subsets of size exactly two need to be witnesses to shatter S. Consider $\{u_i^{(i)}, u_j^{(j)}\} \subseteq S$ for $i \neq j$. Since $u_i u_j \in H$, the vertex $w^{ij}_{u_i u_j}$ exists in W_2 and its neighbourhood in S is exactly $\{u_i^{(i)}, u_j^{(j)}\}$. We conclude that all subsets of size two in S are witnessed as well and therefore S is shattered.

In the other direction, assume that G contains a shattered set (S, W) of size k. Without loss of generality, assume that $k \geqslant 3$.

▷ Claim 18. $S \subseteq \mathcal{V}$ and $W \subseteq \mathcal{W}$.

Proof. Since G is bipartite we either have that $S \subseteq \mathcal{V}$ and $W \subseteq \mathcal{W}$ or that $S \subseteq \mathcal{W}$ and $W \subseteq \mathcal{V}$. Let us now show that the latter is impossible.

Since $k \geqslant 3$ we have that every vertex in S has degree at least four. Accordingly, W cannot contain vertices from W_1 or W_2, which leaves us with $W \subseteq A$. However, all vertices in V_i, $i \in [k]$, have the exact same neighbours in A. Therefore only k subsets of A are witnessed by vertices in $\mathcal{V}$ and therefore the largest shattered set in A has size at most $\log k$. We conclude that S cannot be contained in A and the claim holds. ◁

We now claim that $|S \cap V_i| = 1$ for all $i \in [k]$. Assume otherwise, so let $u^{(i)}, v^{(i)} \in S$ for some $i \in [k]$. But then the set $\{u^i, v^i\}$ cannot be witnessed: not by a vertex from W_1, since it only contains vertices with one neighbour, not by a vertex from W_2, since these vertices each have at most one neighbour in each set V_i, and not by a vertex from A since we need all $2^k - \binom{k}{2} - k - 1$ vertices of A to witness subsets of S of size at least three.

Therefore S intersects each V_i in exactly one vertex. Since S is shattered, every subset $\{u^{(i)}, v^{(j)}\}$, $i \neq j$, is shattered. By the same logic as above, this can only be due to a witness $w^{ij}_{uv} \in W_2$ and therefore $uv \in H$. We conclude that indeed $u_1, \ldots, u_k$ induce a complete graph in H, as claimed.

Finally, we need to determine the degeneracy of G. Consider the following elimination sequence: We first delete all of $\{w_0\} \cup W_1 \cup W_2$, all of which have degree at most two. Note now that all vertices in $\mathcal{V}$ have at most $|A| < 2^k$ neighbours in A, so we delete $\mathcal{V}$ and then A. In total, the maximum degree we encountered in this deletion sequence is $< 2^k$.

Assume we could solve Graph VC-Dimension in time $f(d)n^{o(\log d)}$. In the above reduction the degeneracy of the constructed graph is $d < 2^k$, thus this running time for Graph VC-Dimension would imply a running time of

$$f(d) \cdot n^{o(\log d)} = f(2^k) \cdot n^{o(\log 2^k)} = f(2^k) \cdot n^{o(k)}$$

for k-Clique. We conclude that VC-dimension parameterized by the degeneracy of the input graph cannot be solved in time $f(d)n^{o(\log d)}$ unless all problems in SNP can be solved in subexponential time. ◀

We note that Lemma 15 allows us to approximate the VC-dimension of degenerate graphs.

▶ **Theorem 19.** *Let G be a d-degenerate graph on n vertices. Then for any $0 < \varepsilon \leqslant 1$ we can approximate the VC-dimension of G in time $O(d2^d(2n)^{\lceil \varepsilon(1+\log d)\rceil})$ within a factor of ε.*

Proof. We first compute a d-degenerate ordering $\mathbb{G}$ of G in time $O(dn)$ and sort its left-neighbourhoods in time $O(d \log d \cdot n)$. Let $U_t = (S, W)$ be the largest shattered set in G and let $\mathbb{U}_t$ be its ordering in $\mathbb{G}$. We further prepare the use of Theorem 7 by computing the necessary data structure in time $O(d2^d n)$.

Let $c := \lceil \varepsilon(1 + \log d)\rceil$. The algorithm now iterates over all $C \subseteq V(G)$ of size c and searches the left-neighbourhood $L := N^-[C]$ for a shattered set by first computing a subset dictionary Q_L in time $O(d2^d)$ and then finding the largest shattered subset $S \subseteq L$ by brute-force in time $O(|L|2^{|L|}) = O(cd2^{cd})$.

We claim that this simple algorithm computes the claimed approximation of the VC-dimension. By Lemma 15 we either have that $t \leqslant \log d + 1$ or that $\mathbb{U}_t$ can be left-covered by $\log d + 1$ witness vertices. In the first case, our algorithm will trivally locate an ε-fraction of a maximal solution since it tests every set of size c. In the second case, the shattered set S of $\mathbb{U}_t$ is covered by the left-neighbourhood of witness vertices $w_1, \ldots, w_p \in W$ for $p := \log d+1$. Then by simple averaging, there exist c witnesses W' such that $|N^-[W'] \cap S| \geqslant c|S|/p = ct/(\log d + 1)$. Since the above algorithm will find the shattered set $N^-[W] \cap S$ when inspecting the left-neighbourhood of W, we conclude that it will output at least a value of $ct/(\log d + 1)$. In either case the approximation factor is $\frac{c}{1+\log d} \geqslant \varepsilon$, as claimed. ◀

We would like to highlight the special case of $c = 1$ of the above theorem as it provides us with a linear-time approximation of the VC-dimension, which is probably a good starting point for practical applications:

▶ **Corollary 20.** *Let G be a d-degenerate graph on n vertices. Then we can approximate the VC-dimension of G in time $O(d2^d n)$ within a factor of $\frac{1}{1+\log d}$.*

4.3 Approximating the ladder and semi-ladder index

Before we proceed, we note that degenerate graphs cannot contain arbitrarily long ladders:

▶ **Observation 21.** *If G is d-degenerate then G cannot contain a ladder of length $2d+2$.*

Proof. Note that a ladder of length t contains a complete bipartite graph $K_{\lfloor t/2 \rfloor, \lfloor t/2 \rfloor}$, i.e. a subgraph of minimum degree $\lfloor t/2 \rfloor$. Therefore $t < 2d+2$. ◀

Again we find that a direct application of Theorem 6 to ladder patterns does not yield a satisfying running time since $\mathrm{lc}(L_t) \approx t/2$. However, we can always left-cover a large portion of a ladder with only one vertex:

▶ **Observation 22.** *Let (A, B) induce a ladder of length t in G. For every ordering $\mathbb{G}$ of G there exists a vertex $u \in A \cup B$ such that $|N^-(u) \cap (A \cup B)| \geqslant \lfloor t/2 \rfloor$.*

Proof. Let $A' := (a_i)_{i \geqslant t/2}$ and $B' := (b_i)_{i \leqslant t/2}$, then $G[A' \cup B']$ contains a biclique with partite sets A', B'. Let $u \in A' \cup B'$ be the largest vertex according to $<_{\mathbb{G}}$, then $N^-(u) \cap (A', B')$ is either all of A' or all of B'. In either case the claim holds. ◀

▶ **Theorem 23.** *Let G be a d-degenerate graph on n vertices and let t be its ladder-index. Then we can in time $O(d^2 8^d \cdot n)$ decide whether G contains a ladder of size at least $\lfloor t/2 \rfloor$.*

Proof. We compute a degeneracy ordering $\mathbb{G}$ of G and initialize the data structure R as per Lemma 8 in time $O(2^d n)$.

Let (A, B) induce a ladder of maximum size t in G, by Observation 21 we have that $t \leqslant 2d+1$. By Observation 22, there exists a vertex $u \in A \cup B$ such that $N^-(u)$ contains either $A' := (a_i)_{i \geqslant t/2}$ or $B' := (b_i)_{i \geqslant t/2}$. Wlog assume $A' \subseteq N^-(u)$ and let $k := |A'|$. We guess u in $O(n)$ time and $A' \subseteq N^-(u)$ in time $O(2^d)$. To verify that A' can be completed into a ladder, we compute the data structure $Q_{A'}$ in time $O(k2^k + dk)$ using Lemma 9.

Finally, we verify that there exists a sequence of subsets $A'_1 \subset A'_2 \subset \ldots \subset A'_k = A'$ where $Q_{A'}[A'_k] > 0$ for all $i \in [k]$; as each lookup in $Q_{A'}$ has cost equal to the size of the query set this will take time proportional to $\sum_{i=0}^{k} i\binom{k}{i} = k2^{k-1}$ in the worst case (where we have to query all subsets of A' before finding the sequence). Since $k := \lfloor t/2 \rfloor$, the total running time of this algorithm is

$$O(2^d n) + O\Big(2^d n \cdot (k2^k + dk) \cdot k2^{k-1}\Big) = O\Big(2^d k 2^k (k2^k + dk) \cdot n\Big).$$

We can simplify this expression further by using that $k \leqslant d$ which leads us to the claimed running time of $O(d^2 8^d \cdot n)$ ◀

5 Implementation and experiments

Based on the above theoretical ideas, we implemented algorithms[2] to compute the VC-dimension, find the largest biclique, co-matchings (within an additive error of 1) and ladder (within a factor 2). The last three algorithms all simply check the left-neighbourhood for the respective structure. Aside from optimisations of the involved data structures we will not describe these algorithms in further detail.

We observe that for practical purposes the data structure R can be computed progressively: if we know that our algorithm currently only needs to compute Q_S from R (as per Lemma 9) with $|S| = k$ ($k \leqslant d$), then it is enough to only count sets of size $\leqslant k$ in R. We can achieve this

[2] Source code available under `https://github.com/microgravitas/mantis-shrimp/`

in time $O(\binom{d}{k}n)$, which is far preferable to using $O(2^d n)$ time to insert all left-neighbourhood subsets into R. If k remains much smaller than d, this improves our running time and space consumption substantially.

The second important optimisation regards subset dictionaries. While tries are useful in our theoretical analysis, in practice we opted to use bitsets for the data structures Q_S, as their universe S can assumed to be small. Bitsets also allow for a very concise and fast implementation of the fast Möbius inversion, which needs to happen very frequently inside the hot loop of the search algorithms.

The algorithm to compute the VC-dimension includes a few simple optimisations that vastly improved its performance. Note that if we are currently searching for a shattered set of size k, then a candidate vertex for a shattered set of size k must have at least $\binom{k-1}{i-1}$ neighbours of degree at least i, for $1 \leqslant i \leqslant k-1$. Our algorithm recomputes the set of remaining candidates each time it finds a larger shattered set. The (progressive) computation of the data structure R can then also be restricted to only those left-neighbourhoods subsets which only contain candidate vertices.

Accordingly, the algorithm performs well if it finds large shattered sets fast. To that end, it first only looks at k-subsets of left-neigbhourhoods of single vertices. Once that search is exhausted, it considers left-neighbourhoods of pairs, then triplets, *etc.* up to $\lceil \log d + 1 \rceil$ vertices (as per Lemma 15). As this search is very expensive once we need to consider the joint left-neighbourhood of several vertices, the algorithm estimates the work needed and compares it against simply brute-forcing all k-subsets of the remaining candidates. Since the number of candidates shrinks quite quickly in practice, the algorithm usually concludes with such a final exhaustive search.

5.1 Results

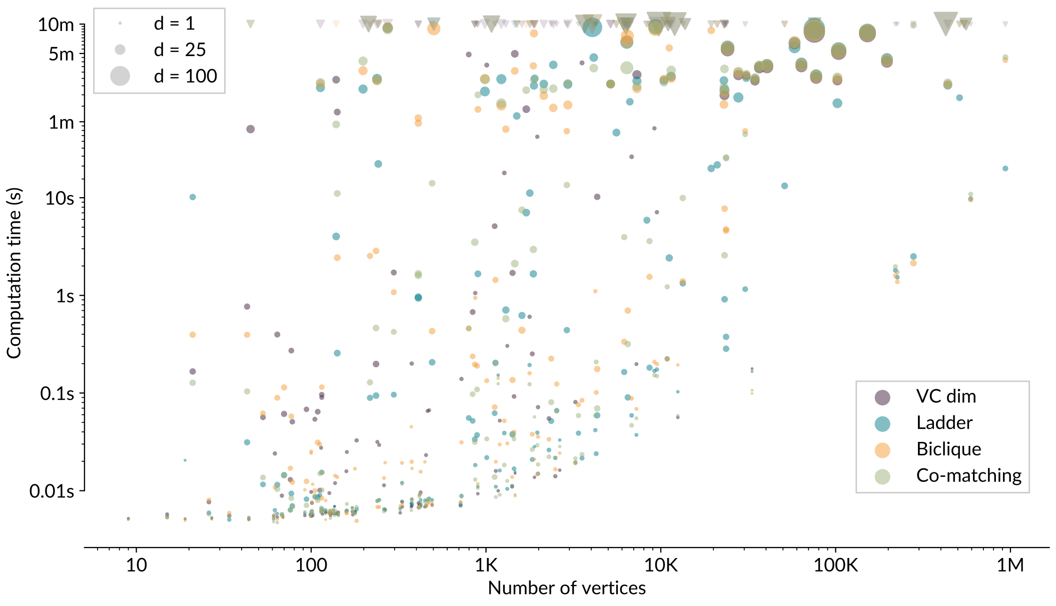


Figure 1 Running times of all four algorithms on a collection of 206 networks. The size of the circles indicates the degeneracy of the networks, triangles indicate that program timed out on the network after 10 minutes.

We implemented all four algorithms in Rust and tested them on a diverse collection of 206 networks[3], using a PC with a AMD Ryzen 3 2200G CPU and 24 GB RAM. The primary goal of our experiments was to verify that the data structures and algorithms in this paper could be of practical use, therefore we ran each algorithm only once per network[4] and timed out after 10 minutes.

Of all the four measures, computing the VC-dimension is, unsuprisingly, the most computationally challenging and the program timed out or ran out of memory for networks larger than a few ten-thousand nodes or of degeneracy higher than 24. The broad summary of the results looks as follows (Figure 1 visualises these results in more detail):

Statistics	Completed	Max size (n)	Max degeneracy
VC-dimension	126	33266 (`BioGrid-Chemicals`)	24 (`wafa-eies`)
Biclique	176	935591 (`teams`)	191 (`BioGrid-All`)
Co-matching	179	935591 (`teams`)	255 (`dogster_friendships`)
Ladder index	187	935591 (`teams`)	191 (`BioGrid-All`)

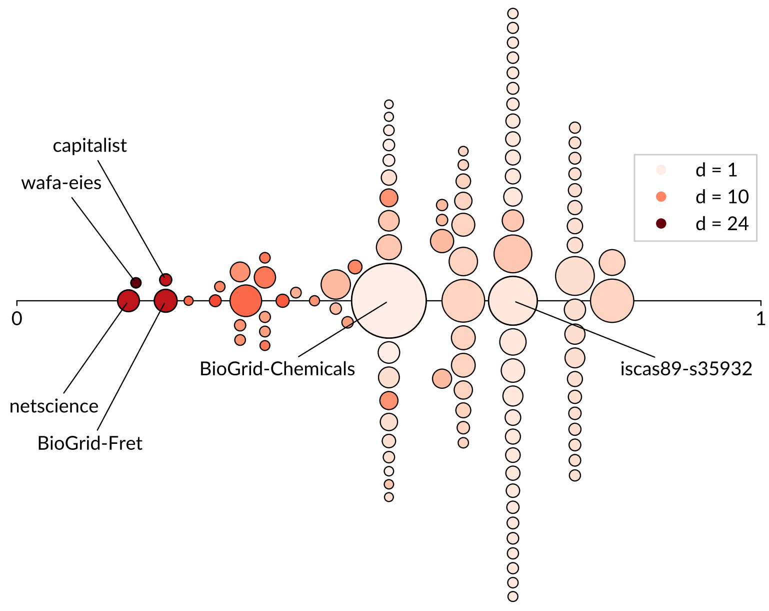


Figure 2 VC-dimension of networks normalized by their degeneracy + 1. Networks with large degeneracy tend towards the left, meaning that the VC-dimension does not increase proportionally to the degeneracy.

We are also interested in typical values of the VC-dimension of networks and how it compares to the degeneracy. This topic deserves a deeper investigation, but we can report some preliminary results here for those networks where our program terminated before the timeout. In Figure 2 we normalised the VC-dimension by the degeneracy-plus-one, so values close to one indicate that the VC-dimension is on the order of the degeneracy while values close to zero indicate that it is much smaller than the degeneracy. We see a clear tendency that networks with larger degeneracy tend towards zero, which we interpret as the VC-dimension "growing slower" than the degeneracy in typical networks.

[3] `https://github.com/microgravitas/network-corpus`
[4] The variance in running times was on the orders of seconds.

6 Conclusion

On the theoretical side, we outlined a general bipartite pattern-finding and -counting algorithm in degenerate graphs. Its running time crucially depends on two complexity measures of patterns, namely the left-covering number and the half-ordering asymmetry. These general algorithms can be further improved for specific patterns, which we exemplify for shattered set, ladder, co-matching and biclique patterns. Our results also include improved running times when the input graphs are of bounded degeneracy.

On the experimental side, we demonstrate that this style of algorithm is feasible and practical for computation on real-world networks, which often exhibit low degeneracy. The experiments also suggest that the VC-dimension of networks tends to be a very small parameter, which makes it an interesting target for the development of fast algorithms that exploit low VC-dimension.

References

1 Suman K. Bera, Noujan Pashanasangi, and C. Seshadhri. Linear time subgraph counting, graph degeneracy, and the chasm at size six. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151, pages 38:1–38:20. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.ITCS.2020.38`.

2 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set Partitioning via Inclusion-Exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009. `doi:10.1137/070683933`.

3 Marco Bressan and Marc Roth. Exact and Approximate Pattern Counting in Degenerate Graphs: New Algorithms, Hardness Results, and Complexity Dichotomies. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 276–285, 2022. `doi:10.1109/FOCS52979.2021.00036`.

4 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. On the computational hardness based on linear FPT-reductions. *Journal of Combinatorial Optimization*, 11(2):231–247, 2006. `doi:10.1007/s10878-006-7137-6`.

5 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006. `doi:10.1016/j.jcss.2006.04.007`.

6 Norishige Chiba and Takao Nishizeki. Arboricity and Subgraph Listing Algorithms. *SIAM Journal on Computing*, 14(1):210–223, 1985. `doi:10.1137/0214017`.

7 Erik D. Demaine, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, Somnath Sikdar, and Blair D. Sullivan. Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs. *Journal of Computer and System Sciences*, 105:199–241, 2019.

8 Rodney G. Downey, Patricia A. Evans, and Michael R. Fellows. Parameterized learning complexity. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 51–57, 1993.

9 Pål Grønås Drange, Patrick Greaves, Irene Muzi, and Felix Reidl. Computing complexity measures of degenerate graphs. *CoRR*, arXiv:2308.08868 [cs.DS], 2023. `doi:10.48550/arXiv.2308.08868`.

10 Eduard Eiben, Robert Ganian, Thekla Hamm, Lars Jaffke, and O-joung Kwon. A Unifying Framework for Characterizing and Computing Width Measures. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215, pages 63:1–63:23. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.ITCS.2022.63`.

11 Kord Eickmeyer, Archontia C. Giannopoulou, Stephan Kreutzer, O-joung Kwon, Michał Pilipczuk, Roman Rabinovich, and Sebastian Siebertz. Neighborhood Complexity and Kernelization for Nowhere Dense Classes of Graphs. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80, pages 63:1–63:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017. `doi:10.4230/LIPIcs.ICALP.2017.63`.

12 David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. *ACM Journal of Experimental Algorithmics*, 18:3.1:3.1–3.1:3.21, 2013. doi:10.1145/2543629.

13 Grzegorz Fabiański, Michał Pilipczuk, Sebastian Siebertz, and Szymon Toruńczyk. Progressive algorithms for domination and independence. In *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126, pages 27:1–27:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.STACS.2019.27.

14 Donald E. Knuth. *Art of computer programming, volume 2: Seminumerical algorithms.* Addison–Wesley Professional, 2014.

15 David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM*, 30(3):417–427, 1983. doi:10.1145/2402.322385.

16 Christos H. Papadimitriou and Mihalis Yannakakis. On limited nondeterminism and the complexity of the VC dimension. *Journal of Computer and System Sciences*, 53(2):161–170, 1996.

17 Robert Sedgewick and Kevin Wayne. *Algorithms, 4th edition.* Addison–Wesley, 2011.

18 Frank Yates. The design and analysis of factorial experiments. *Imperial Bureau of Soil Science*, 1937.

A Complete results

For space reasons, some of the network names are abbreviated below. Abbreviated names are marked in gray. Some of the rows have been deferred to the full version of the paper [9].

							Running time				Statistics			
Network	n	m	δ	$\bar{d}$	deg	Δ	VC-dim	biclique	crown	ladder	VC-dim	biclique	crown	ladder
AS-oregon-1	11174	23409	2389	4.2	17	2389	600.09	342.08	175.43	2.42	[5,18]	12	[13,14]	[17,35]
AS-oregon-2	11461	32730	2432	5.7	31	2432	600.40	167.57	201.43	174.27	[6,32]	[7,31]	[7,32]	[7,63]
BG-AC-Luminescence	1840	2312	376	2.5	6	376	0.25	0.04	0.03	0.02	4	5	7	[6,13]
BG-AC-Rna	13765	42815	3572	6.2	54	3572	600.21	601.05	601.38	601.54	[4,55]	[5,54]	[5,55]	[5,109]
BG-AC-Western	21028	64046	535	6.1	17	535	600.08	600.30	600.29	21.70	[5,18]	[7,17]	[9,18]	[17,35]
BG-Biochemical-Activity	8620	17746	427	4.1	11	427	600.11	1.55	3.59	0.18	[5,12]	8	[7,8]	[11,23]
BG-Bos-Taurus	454	424	27	1.9	3	27	0.02	0.01	0.01	0.01	2	3	[3,4]	[3,7]
BG-C.-Elegans	6394	23646	522	7.4	64	522	600.21	399.05	214.18	390.57	[4,65]	[6,64]	[5,65]	[6,129]
BG-Canis-Familiaris	143	125	90	1.7	2	90	0.01	0.01	0.01	0.01	2	2	3	[2,5]
BG-Chemicals	33266	28093	413	1.7	1	413	0.18	0.10	0.11	0.17	1	[0,1]	2	[0,3]
BG-Co-Localization	3543	4452	63	2.5	6	63	240.56	0.08	0.02	0.02	3	5	7	[6,13]
BG-Co-Purification	4326	5970	1972	2.8	12	1972	10.21	0.18	0.07	0.06	4	6	13	[12,25]
BG-Cricetulus-Griseus	69	57	30	1.7	1	30	0.01	0.01	0.01	0.01	1	[0,1]	2	[0,3]
BG-D.-Discoideum-Ax4	27	20	4	1.5	1	4	0.01	0.01	0.01	0.01	1	[0,1]	2	[0,3]
BG-D.-Growth-Defect	1447	2193	213	3.0	5	213	0.09	0.04	0.03	0.02	4	4	6	[5,11]
BG-D.-Melanogaster	9330	60556	303	13.0	83	303	600.32	538.71	553.50	566.57	[4,84]	[5,83]	[5,84]	[5,167]
BG-E.-Nidulans-Fgsc-A4	64	62	44	1.9	2	44	0.01	0.01	0.00	0.01	2	2	3	[2,5]
BG-E.-Coli-K12-Mg1655	1273	1889	58	3.0	5	58	17.95	0.05	0.02	0.04	3	4	6	[5,11]
BG-Far-Western	1199	1089	60	1.8	3	60	0.05	0.03	0.01	0.02	3	3	4	[3,7]
BG-Fret	1700	2395	51	2.8	19	51	80.71	600.09	126.45	7.04	4	[11,19]	[17,18]	[19,39]
BG-Hepatitus-C-Virus	136	134	133	2.0	1	133	0.01	0.01	0.01	0.01	1	[0,1]	2	[0,3]
BG-Homo-Sapiens	24093	369767	2882	30.7	71	2882	329.05	338.13	343.83	346.77	[3,72]	[3,71]	[3,72]	[3,143]
BG-HSV-1	178	208	40	2.3	3	40	0.01	0.02	0.01	0.01	3	3	4	[3,7]

Continued on next page

							Running time				Statistics			
Network	n	m	δ	$\bar{d}$	deg	Δ	VC-dim	biclique	crown	ladder	VC-dim	biclique	crown	ladder
BG-HSV-5	121	107	27	1.8	1	27	0.01	0.01	0.01	0.01	1	[0,1]	2	[0,3]
BG-HSV-8	716	691	119	1.9	3	119	0.01	0.01	0.01	0.01	2	3	[3,4]	[3,7]
BG-HPV-16	173	186	93	2.2	2	93	0.01	0.01	0.01	0.01	2	2	[2,3]	[2,5]
Cannes2013	438089	835892	15169	3.8	27	15169	600.79	144.70	149.74	142.22	[5,28]	[6,27]	[6,28]	[6,55]
CoW-interstate	182	319	25	3.5	4	25	0.03	0.00	0.01	0.01	3	4	[3,4]	[4,9]
EU-email-core	986	16064	345	32.6	34	345	600.86	164.82	163.41	122.48	[5,35]	[6,34]	[6,35]	[6,69]
JDK_dependency	6434	53658	5923	16.7	65	5923	600.41	453.02	600.99	601.73	[4,66]	[5,65]	[5,66]	[5,131]
NZ_legal	2141	15739	429	14.7	25	429	600.36	110.96	128.55	146.23	[6,26]	[7,25]	[7,26]	[7,51]
Noordin-terror-loc	127	190	18	3.0	3	18	0.01	0.01	0.01	0.01	3	3	[3,4]	[3,7]
Noordin-terror-orgas	129	181	21	2.8	3	21	0.01	0.01	0.01	0.01	3	3	[3,4]	[3,7]
ODLIS	2900	16377	592	11.3	12	592	600.03	48.04	13.49	0.44	[5,13]	6	[9,10]	[12,25]
Opsahl-forum	899	7036	128	15.7	14	128	600.02	80.31	113.04	1.66	[5,15]	5	[6,7]	[14,29]
Y2H_union	1966	2705	89	2.8	4	89	42.11	0.01	0.03	0.04	3	4	5	[4,9]
Yeast	2361	7182	66	6.1	10	66	600.04	0.23	0.08	0.05	[4,11]	7	[9,10]	[10,21]
actor_movies	511463	1470404	646	5.7	14	646	600.36	600.51	600.46	105.76	[5,15]	[7,14]	[6,15]	[14,29]
airlines	235	1297	130	11.0	13	130	0.20	2.85	0.46	0.09	5	8	[11,12]	[13,27]
american_revolution	141	160	59	2.3	3	59	0.01	0.01	0.01	0.01	3	3	[2,3]	[3,7]
autobahn	374	478	5	2.6	2	5	0.01	0.05	0.01	0.01	2	2	3	[2,5]
bahamas	219856	246291	14902	2.2	6	14902	600.12	1.59	1.97	1.80	[3,7]	6	[3,4]	[6,13]
bergen	53	272	32	10.3	9	32	0.06	0.06	0.01	0.01	4	6	[8,9]	[9,19]
bitcoin-otc-positive	5573	18591	788	6.7	20	788	600.14	600.45	600.24	46.50	[6,21]	[8,20]	[10,21]	[20,41]
bn-fly-d._medulla_1	1781	8911	927	10.0	18	927	600.04	600.17	600.15	11.17	[5,19]	[8,18]	[8,19]	[18,37]
boards_gender_2m	4220	5598	45	2.7	4	45	600.10	1.10	0.06	0.04	[3,5]	4	[3,4]	[4,9]
ca-CondMat	23133	93439	279	8.1	25	279	600.05	600.65	208.73	600.65	[4,26]	[14,25]	26	[18,51]
ca-HepPh	12006	118489	491	19.7	238	491	600.21	600.33	600.13	600.17	[3,239]	[3,238]	[3,239]	[3,477]
celegans	297	2148	134	14.5	10	134	1.72	1.08	0.42	0.10	5	7	[8,9]	[10,21]
chess	7301	55899	181	15.3	29	181	182.00	130.97	138.03	157.14	[6,30]	[6,29]	[6,30]	[6,59]
cit-HepTh	27769	352285	2468	25.4	37	2468	180.07	189.90	194.29	106.40	[4,38]	[4,37]	[4,38]	[3,75]
codeminer	724	1015	55	2.8	4	55	0.14	0.03	0.01	0.01	3	4	[4,5]	[4,9]
columbia-mobility	863	4147	228	9.6	9	228	600.11	0.20	0.03	0.03	[4,10]	6	10	[9,19]

Continued on next page

							Running time				Statistics			
Network	n	m	δ	$\bar{d}$	deg	Δ	VC-dim	biclique	crown	ladder	VC-dim	biclique	crown	ladder
cora_citation	23166	89157	377	7.7	13	377	600.11	7.72	2.57	0.91	[5,14]	9	[10,11]	[13,27]
countries	592414	624402	110602	2.1	6	110602	600.14	9.49	10.88	9.75	[4,7]	5	[4,5]	[6,13]
diseasome	1419	2738	84	3.9	11	84	1.70	0.14	0.04	0.04	4	6	12	[11,23]
dogster_friendships	426820	8546581	46505	40.0	255	46505	600.38	600.57	600.00	600.44	[1,256]	[0,255]	[1,256]	[0,511]
dolphins	62	159	12	5.1	4	12	0.02	0.01	0.01	0.01	3	3	5	[4,9]
edinburgh_assoc._thes.	23132	297094	1062	25.7	34	1062	112.09	119.67	123.49	128.27	[3,35]	[3,34]	[3,35]	[3,69]
email-Enron	36692	183831	1383	10.0	43	1383	213.38	219.83	223.53	219.47	[4,44]	[4,43]	[4,44]	[4,87]
exnet-water	1893	2416	10	2.6	2	10	0.01	0.02	0.06	0.03	2	2	3	[2,5]
facebook-links	63731	817090	1098	25.6	52	1098	221.29	229.98	233.75	237.25	[3,53]	[3,52]	[3,53]	[3,105]
foldoc	13356	91471	728	13.7	12	728	600.10	1.39	9.93	1.33	[5,13]	12	[9,10]	[12,25]
foodweb-otago	141	832	45	11.8	14	45	75.41	2.43	11.06	0.26	4	12	[7,8]	[14,29]
football	115	613	12	10.7	8	12	0.10	0.12	0.01	0.02	4	4	9	[8,17]
haggle	274	2124	101	15.5	39	101	600.11	551.34	533.96	550.92	[4,40]	[8,39]	[8,40]	[8,79]
hex	331	930	6	5.6	3	6	0.01	0.02	0.01	0.01	3	2	[3,4]	[3,7]
hypertext_2009	113	2196	98	38.9	28	98	600.15	146.97	150.88	134.22	[5,29]	[9,28]	[9,29]	[9,57]
ia-infect-dublin	410	2765	50	13.5	17	50	600.10	65.59	1.60	0.94	[4,18]	9	[16,17]	[17,35]
ia-reality	6809	7680	261	2.3	5	261	26.27	0.09	0.05	0.06	4	4	[5,6]	[5,11]
iscas89-s1238	416	625	18	3.0	2	18	0.01	0.01	0.01	0.01	2	2	[2,3]	[2,5]
iscas89-s13207	2492	3406	37	2.7	4	37	0.01	0.02	0.02	0.02	4	4	[4,5]	[4,9]
iscas89-s1423	423	554	17	2.6	2	17	0.01	0.01	0.01	0.01	2	2	[2,3]	[2,5]
iscas89-s1494	473	796	56	3.4	3	56	0.07	0.01	0.01	0.01	3	3	[3,4]	[3,7]
iscas89-s15850	3247	4004	25	2.5	4	25	0.08	0.02	0.02	0.02	3	4	[3,4]	[4,9]
iscas89-s298	92	131	11	2.8	2	11	0.01	0.01	0.01	0.01	2	2	[2,3]	[2,5]
iscas89-s344	100	122	9	2.4	2	9	0.01	0.02	0.01	0.01	2	2	[2,3]	[2,5]
iscas89-s349	102	127	9	2.5	2	9	0.01	0.01	0.01	0.01	2	2	[2,3]	[2,5]
iscas89-s382	116	168	18	2.9	2	18	0.01	0.02	0.01	0.01	2	2	[2,3]	[2,5]
iscas89-s38417	9500	10635	39	2.2	4	39	7.11	0.20	0.15	0.17	4	2	[4,5]	[4,9]
iscas89-s400	121	182	19	3.0	2	19	0.01	0.01	0.01	0.01	2	2	[2,3]	[2,5]
iscas89-s420	129	145	9	2.2	2	9	0.01	0.01	0.01	0.01	2	2	[2,3]	[2,5]
iscas89-s444	134	206	19	3.1	2	19	0.01	0.01	0.01	0.01	2	2	3	[2,5]

Continued on next page

							Running time				Statistics			
Network	n	m	δ	$\bar{d}$	deg	Δ	VC-dim	biclique	crown	ladder	VC-dim	biclique	crown	ladder
iscas89-s526	160	270	12	3.4	3	12	0.02	0.01	0.01	0.01	3	3	[2,3]	[3,7]
iscas89-s526n	159	268	12	3.4	3	12	0.02	0.01	0.01	0.01	3	3	[3,4]	[3,7]
iscas89-s713	137	180	12	2.6	3	12	0.01	0.01	0.01	0.01	3	2	[2,3]	[3,7]
iscas89-s820	239	480	48	4.0	3	48	0.03	0.01	0.01	0.01	3	3	[3,4]	[3,7]
iscas89-s832	245	498	49	4.1	3	49	0.03	0.01	0.01	0.01	3	3	[3,4]	[3,7]
iscas89-s9234	1985	2370	18	2.4	4	18	0.07	0.04	0.02	0.01	3	4	[3,4]	[4,9]
iscas89-s953	332	454	12	2.7	2	12	0.01	0.01	0.01	0.01	2	2	[2,3]	[2,5]
lederberg	8324	41532	1103	10.0	15	1103	600.11	600.11	600.08	5.88	[5,16]	[7,15]	[9,16]	[15,31]
lesmiserables	77	254	36	6.6	9	36	0.27	0.06	0.01	0.01	3	6	10	[9,19]
link-pedigree	898	1125	14	2.5	2	14	0.01	0.01	0.01	0.01	2	2	[2,3]	[2,5]
livemocha	104103	2193083	2980	42.1	92	2980	308.64	318.47	325.14	326.65	[2,93]	[2,92]	[2,93]	[2,185]
loc-brightkite_edges	58228	214078	1134	7.4	52	1134	381.48	385.52	393.61	346.98	[5,53]	[5,52]	[5,53]	[5,105]
mg_casino	109	326	94	6.0	9	94	0.06	0.03	0.01	0.01	3	5	10	[9,19]
mg_forrestgump	94	271	89	5.8	8	89	0.07	0.01	0.01	0.01	3	4	9	[8,17]
mg_godfatherII	78	219	34	5.6	8	34	0.05	0.01	0.01	0.01	3	5	9	[8,17]
minnesota	2642	3303	5	2.5	2	5	0.02	0.02	0.03	0.03	2	2	3	[2,5]
moreno_health	2539	10455	27	8.2	7	27	600.10	0.12	0.07	0.07	[4,8]	5	[7,8]	[7,15]
movies	101	192	19	3.8	3	19	0.01	0.01	0.01	0.01	3	2	[3,4]	[3,7]
muenchen-bahn	447	578	13	2.6	2	13	0.01	0.02	0.01	0.01	2	[0,1]	[2,3]	[2,5]
munin	1324	1397	66	2.1	3	66	0.30	0.03	0.01	0.01	2	3	[2,3]	[3,7]
offshore	278877	505965	37336	3.6	13	37336	600.11	2.14	583.20	2.50	[5,14]	13	[9,10]	[13,27]
openflights	2939	15677	242	10.7	28	242	600.11	89.14	147.54	144.73	[5,29]	[7,28]	[7,29]	[7,57]
paradise	542102	794545	35359	2.9	23	35359	600.21	601.52	600.31	601.57	[5,24]	[22,23]	[6,24]	[23,47]
photoviz_dynamic	376	610	29	3.2	4	29	0.20	0.02	0.01	0.01	3	3	[3,4]	[4,9]
pigs	492	592	39	2.4	2	39	0.01	0.01	0.01	0.01	2	2	[2,3]	[2,5]
polbooks	105	441	25	8.4	6	25	0.05	0.01	0.01	0.01	4	5	[6,7]	[6,13]
pollination-carlinville	1500	15255	157	20.3	18	157	600.04	600.37	600.30	68.80	[5,19]	[6,18]	[6,19]	[18,37]
ratbrain	503	23030	497	91.6	67	497	600.50	538.67	601.38	600.83	[4,68]	[5,67]	[5,68]	[5,135]
reactome	6327	147547	855	46.6	191	855	600.17	600.17	600.21	600.23	[3,192]	[3,191]	[3,192]	[3,383]
residence_hall	217	1839	56	16.9	11	56	600.10	2.54	0.13	0.09	[4,12]	6	[10,11]	[11,23]

Continued on next page

							Running time				Statistics			
Network	n	m	δ	$\bar{d}$	deg	Δ	VC-dim	biclique	crown	ladder	VC-dim	biclique	crown	ladder
roget-thesaurus	1010	3648	28	7.2	6	28	228.14	0.13	0.03	0.02	4	3	[6,7]	[6,13]
slashdot_threads	51083	117378	2915	4.6	14	2915	600.13	600.17	600.14	13.23	[5,15]	[5,14]	[6,15]	[14,29]
soc-advogato	5167	39432	807	15.3	25	807	600.47	145.63	146.80	145.69	[5,26]	[6,25]	[6,26]	[6,51]
soc-gplus	23628	39194	2761	3.3	12	2761	600.04	4.74	25.84	0.28	[6,13]	9	[7,8]	[12,25]
soc-hamsterster	2426	16630	273	13.7	24	273	600.10	83.26	131.04	229.61	[5,25]	[11,24]	[23,25]	[24,49]
sp_data_school_day_2	238	5539	88	46.5	33	88	600.31	159.31	159.79	165.56	[5,34]	[6,33]	[7,34]	[7,67]
teams	935591	1366466	2671	2.9	9	2671	600.24	256.73	275.42	19.94	[6,10]	6	[6,7]	[9,19]
twittercrawl	3656	154824	1084	84.7	143	1084	600.46	600.68	600.37	600.47	[3,144]	[3,143]	[3,144]	[3,287]
unicode_languages	868	1255	141	2.9	4	141	1.06	0.94	0.01	0.02	3	4	[3,4]	[4,9]
wafa-ceos	26	93	22	7.2	5	22	0.01	0.01	0.01	0.01	3	3	[5,6]	[5,11]
wafa-hightech	21	159	20	15.1	12	20	0.17	0.40	0.13	10.16	3	7	[10,11]	[8,17]
wafa-padgett	15	27	8	3.6	3	8	0.01	0.01	0.01	0.01	2	2	[3,4]	[3,7]
web-google	1299	2773	59	4.3	17	59	600.02	50.40	0.58	0.71	[3,18]	9	18	[17,35]
wikipedia-norm	1881	15372	455	16.3	22	455	600.14	482.15	161.62	140.63	[6,23]	[10,22]	[11,23]	[11,45]
win95pts	99	112	9	2.3	2	9	0.01	0.01	0.01	0.01	2	2	3	[2,5]
word_adjacencies	112	425	49	7.6	6	49	0.01	0.03	0.01	0.01	4	4	[5,6]	[6,13]
zewail	6651	54182	331	16.3	18	331	600.09	601.22	601.20	96.19	[5,19]	[9,18]	[10,19]	[18,37]