Solving the maximum clique and vertex coloring problems on very large sparse networks*

Anurag Verma Austin Buchanan Sergiy Butenko

16 February 2017

Abstract

This paper explores techniques for solving the maximum clique and vertex coloring problems on very large-scale real life networks. Due to the size of such networks and the intractability of the considered problems, previously developed exact algorithms may not be directly applicable. The proposed approaches aim to reduce the network instances to a size that is tractable for existing solvers, while preserving optimality. Two clique relaxation structures are exploited for this purpose. In addition to the known k-core structure, a newly introduced clique relaxation, k-*community*, is used to further reduce the instance size. Experimental results on real life graphs (collaboration networks, P2P networks, social networks, etc.) show the proposed procedures to be effective by finding, for the first time, exact solutions for instances with over 18 million vertices.

1. Introduction

A simple undirected graph, denoted by $G = (V, E)$, is defined by a set of vertices V, and a set of edges E representing the pairs of vertices that are adjacent. Graphs can be used to represent information in a very concise manner based on pairwise relationships between entities. A clique, defined as a subset of vertices that are all pairwise adjacent, is a graph-theoretic concept often used to represent dense clusters. Cliques are key in the development of many graph-based data mining approaches for the analysis of networks arising in diverse application areas, such as social, communication, and biological systems (Bomze et al. 1999, Butenko and Wilhelm 2006).

The maximum clique problem is to find a clique of maximum cardinality in the given graph. The size of a maximum clique in G is known as the *clique*

*Published as A. Verma, A. Buchanan, S. Butenko. Solving the maximum clique and vertex coloring problems on very large sparse networks. *INFORMS Journal on Computing*, 27(1):164-177, 2015. http://dx.doi.org/10.1287/ijoc.2014.0618. Note: Table 3 was missing parentheses in the version on Austin Buchanan's personal webpage, and this was fixed on 16 February 2017. The journal version already had these parentheses.

number of G, denoted by $\omega(G)$. Vertex coloring is another classical combinatorial optimization problem. A proper vertex coloring refers to an assignment of a color to each vertex such that no two adjacent vertices have the same color. The vertex coloring problem is to find a proper coloring that uses the fewest number of colors, known as the *chromatic number* of G, denoted by $\chi(G)$. The vertex coloring problem has applications in scheduling and timetabling, register allocation, frequency assignment, and air traffic flow management (Malaguti and Toth 2010, Pardalos et al. 1998).

The maximum clique and vertex coloring problems have captured the attention of many researchers in computer science and operations research. Both problems are among Karp's original 21 problems shown to be NP-complete (Karp 1972) and are known to be hard to approximate (Håstad 1999, Feige and Kilian 1998, Zuckerman 2007). Numerous techniques for solving the maximum clique problem have been developed, including exact methods using integer programming, implicit enumeration and scale reduction approaches (Carraghan and Pardalos 1990, Corno et al. 1995, Balas and Xue 1996, Wood 1997, Östergård 2002, Tomita and Seki 2003, Butenko and Trukhanov 2007), as well as heuristics and meta-heuristics (Gendreau et al. 1993, Protasi et al. 1995, Abello et al. 1999, Katayama et al. 2005). Similarly, many exact approaches – implicit enumeration, integer and constraint programming (Campêlo et al. 2008, Hansen et al. 2009, Méndez-Díaz and Zabala 2006, Mehrotra and Trick 1995, Gualandi and Malucelli 2012, Held et al. 2012, Malaguti et al. 2011), and heuristics (Brélaz 1979, Bollobás and Thomason 1985, Culberson and Luo 1996, Morgenstern 1996) – have been proposed for the vertex coloring problem. Despite these efforts, there are still unsolved clique instances with 1024 vertices (N. J. A. Sloane Last accessed: February 2017) and unsolved coloring instances with fewer than 200 vertices (Malaguti and Toth 2010).

Recent advances in information technology have resulted in data sets that are much larger than what most exact algorithms have been designed for and tested on. Many real life networks of interest are very large, with tens of millions of vertices, and have low edge densities, with the degrees of vertices often following a power-law distribution (Newman 2003). It should be noted that the maximum clique and vertex coloring problems remain hard to approximate when restricted to power-law graphs, unless NP=ZPP (Shen et al. 2012).

There is limited existing research on solving the maximum clique problem in the very large, sparse graphs. Many existing approaches for solving the maximum clique problem rely heavily on an adjacency matrix representation of the graph. Graphs with millions of vertices will simply not fit into a computer's working memory when stored in this way (e.g., a graph with 10 million vertices would require 11.4 TB of memory). So, most existing maximum clique implementations are not directly applicable. Accordingly, heuristic algorithms are frequently used, although they do not provide any guarantee of the quality of the optimal solution. One of the key properties of power-law graphs that can be exploited in designing algorithms for the problems of interest is the presence of a large number of low-degree vertices, which can be removed without changing the clique and chromatic numbers of the graph. This observation motivated the so-

called "peeling" *scale reduction* technique proposed by Abello et al. (1999), who attempted to solve the maximum clique problem on a graph with 53,767,087 vertices and over 170 million edges representing AT&T call data. Its largest connected component had 44,989,297 vertices and contained a clique of size 30, which was found using GRASP metaheuristic. Applying a peeling procedure that recursively removed vertices of degree less than 30, they managed to bring the size of the problem instance down to 8,724 vertices and about 320,000 edges. This graph was guaranteed to contain any clique of size 31 or more if one exists; however, the largest clique Abello et al. (1999) were able to find using GRASP contained 30 vertices. As for the vertex coloring problem, we are not aware of any work targeting graphs with more than several thousand vertices.

In this paper, we present some advances in scale reduction methods for the maximum clique problem and the vertex coloring problem in very large sparse graphs. The proposed techniques allow one to extend the applicability of existing exact algorithm implementations to larger graphs, where they fail if applied directly. The algorithms have been tested on graphs with up to 18 million vertices originating from some real life applications. The organization of the rest of the paper is as follows. Section 2 presents clique relaxations used to devise scale reduction approaches. Section 3 is devoted to the maximum clique problem in very large sparse graphs. Section 4 presents scale reduction approaches for the vertex coloring problem, and finally Section 5 concludes the paper.

2. Cores and Communities

The scale reduction approach used by Abello et al. (1999) for finding large cliques in a massive telecommunication network was based on the so-called "peeling" procedure, which, given a heuristically computed clique C of size k in G, recursively removes vertices of degree less than k. Obviously, such vertices cannot belong to any clique of size $k+1$ or greater, hence their removal does not impact cliques containing more vertices than C. Thus, if C is not a maximum clique of G, the graph obtained as the result of applying the peeling procedure must contain all maximum cliques of G. The peeling procedure is effectively an algorithm for computing the largest *k-core* of G, which is formally defined next. Given a graph $G = (V, E)$ and $S \subseteq V$, $G[S]$ denotes the subgraph induced by S, and $\delta(G)$ is the minimum degree of G.

Definition 1 (k-core; degeneracy)**.** *A subset $S \subseteq V$ of vertices is called a k-core if $\delta(G[S]) \geq k$. The largest k for which G has a nonempty k-core is called the degeneracy of G.*

It should be noted that a k-core of a graph as presented in Definition 1 may not be unique. However, the *maximum* k-core is unique and can be found in linear time by iteratively removing a vertex if its degree is less than k (Matula and Beck 1983). In fact, the same algorithm computes the degeneracy of a

graph in linear time. Consult Lick and White (1970) for more information about degeneracy.

One hurdle encountered by Abello et al. (1999) was that the peeling procedure did not yield a sufficient reduction in the graph size to employ an exact algorithm. As a result, they were not able to guarantee optimality of the solutions they computed. This can be attributed to the weakness of k-cores as clique relaxations. In the following, we present a different clique relaxation structure called k-community that can be used for scale reduction purposes, and study the properties of k-community to ascertain its relative strength when compared to k-cores.

It should be noted that Cohen (2008) introduced what they called k-truss for finding cohesive subgraphs for social network analysis. A k-truss is defined as a connected subgraph with each edge being a part of at least $k-2$ triangles. This implies that any clique of size k induces a k-truss. However, to maintain consistency with the definition of k-cores, we define a k-community as follows.

Definition 2 (k-community)**.** *A subgraph $G' = (V', E')$ of G is called a k-community subgraph if for each edge $\{u, v\} \in E'$ its incident vertices u and v have at least k common neighbors in G'. A subset V' of vertices of G is called a k-community if there exists a k-community subgraph of G that has V' as its vertex set.*

Observe that a clique of size k is a $(k-2)$-community. As is the case with the k-core, the maximum k-community of a graph can be found in polynomial time. Algorithm 1 describes a simple iterative procedure for finding the k-community of a graph G. In the first iteration of the algorithm an edge is removed if its incident vertices have fewer than k common neighbors. If no edges were removed, the k-community of the graph has been found. Otherwise another iteration is performed. The algorithm can be implemented by examining an edge only if one of its neighboring edges was removed in the previous iteration. This limits the number of times the size of the common neighborhood of an edge's endpoints is calculated to $(2\Delta + 1)$ times, Δ being the highest degree of a vertex in the graph. The overall time complexity of finding the k-community can be shown to be $O(mk\Delta)$. A brief outline of the proof is as follows: suppose m_1 edges get deleted in the first iteration. Then it takes $O(m_1\Delta)$ time to delete those edges, and $O(m_1k\Delta)$ to investigate new ones. The second term arises because it takes $O(\Delta)$ time to investigate one edge, and there are at most $2m_1k$ edges to be investigated since each of the removed edges affects at most $2k$ edges. Similarly, suppose m_2 edges get deleted in the next iteration. Then it takes $O(m_2\Delta)$ time to delete those edges, and $O(m_2k\Delta)$ to investigate new ones. Similarly define m_3, m_4 and so on. Since $\sum_i m_i \leq m$, the algorithm runs in $O(mk\Delta)$ time. It should be noted that, in a subsequent paper, Buchanan et al. (2013) show that a maximum k-community (for any value of k) can be found in time $O(nm)$.

Some elementary properties of the k-community that form the basis of the proposed scale reduction techniques for the maximum clique problem are provided below.

Algorithm 1 $k-$ `Community`(G): Algorithm to find a k-community subgraph of G

repeat
  for every $(i,j) \in E$ **do**
    if $|N(i) \cap N(j)| < k$ **then**
      Remove edge $\{i,j\}$ from E
    end if
  end for
until No edge is removed in the current iteration
Remove all isolated vertices to obtain $G_k \equiv (V_k, E_k)$
return G_k

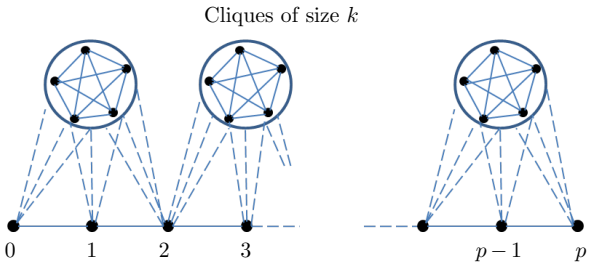

Figure 1: A k-community with $p(k+2)/2+1$ vertices and diameter p.

Property 2.1. *A clique of size k is a $(k-t)$-community for any $2 \leq t \leq k$.*

Property 2.2. *If G has no nonempty k-community, then $\omega(G) \leq k+1$.*

Note that the converse of Property 2.2 is not true. It can also be easily observed that, while a k-community of G is a $(k+1)$-core of G, the converse is not true in general. This implies that a k-community is a stronger clique relaxation when compared to a k-core. Next we establish some cohesiveness properties of k-communities.

Theorem 1 (Cohesiveness properties of k-communities)**.** *A k-community subgraph $G' = (V', E')$ with $|V'| = n$ satisfies the following conditions.*

(a) If G' is connected, the diameter of G' is at most $\left\lfloor \frac{2(n-1)}{k+2} \right\rfloor$.

(b) The minimum degree $\delta(G')$ is at least $k+1$.

(c) The density of G' is at least $\frac{k+1}{n-1}$.

Figure 1 shows that the bound on the diameter is sharp.

Proof. Let the diameter of a given k-community subgraph G' with n vertices be p. Then there exist vertices u, v in the k-community such that the length

of a shortest path between u and v in G' is p. Let $u = x_0, x_1, ..., x_p = v$ be such a shortest path. Note that x_i and x_{i+1}, $i = 0, ..., (p-1)$ have at least k common neighbors. Also, since $x_0, x_1, ..., x_p$ is the shortest path from x_0 to x_p in G', the vertices x_i, x_{i+1}, x_j, and x_{j+1} cannot have any common neighbors if $|i-j| \geq 2$. Hence, endpoints of every alternate edge in the path have at least k unique common neighbors. Thus, the number of vertices in the graph satisfies $n \geq (p+1) + k(\lceil \frac{p}{2} \rceil) \geq (p+1) + k(\frac{p}{2}) = 1 + p(\frac{k+2}{2})$. Hence, $p \leq \left\lfloor \frac{2(n-1)}{k+2} \right\rfloor$, which establishes (a). Statement (b) is trivial, and (c) directly follows from (b). □

3. Solving the Maximum Clique Problem on Very Large Sparse Graphs

In this section, we devise a new scale reduction method for the maximum clique problem based on k-communities. Property 2.2 is used to design algorithms for finding an upper bound on the clique number, while Property 2.1 is used to find a maximum clique as described in Algorithm 2. This algorithm requires $O(m+n)$ memory using the adjacency list representation. It first employs a greedy heuristic `HeuristicClique`(G) for the maximum clique problem to obtain a lower bound ω_{lower}. Next, either binary or linear search is used in `UpperBound`$(G, \omega_{\text{lower}}, \Delta)$ to obtain an upper bound ω_{upper} on the clique number. Binary search is performed in the interval $[\omega_{\text{lower}}, \Delta + 1]$ and finds the smallest integer k such that $(k+1)$-core or k-community of G is an empty set. By Property 2.2, $(k+1)$ is an upper bound on the clique number. The algorithm runs in $O(m\Delta^2 \log \Delta)$ time. Since finding the k-community modifies the edge set, the whole edge set has to be duplicated from the original graph or a previously computed $(\omega_{\text{lower}} - 2)$-community after each iteration. This can take significant amounts of memory for large graphs. Therefore, a linear search strategy is also considered that starts with $k = \omega_{\text{lower}} - 2$ and increments k till the upper bound is found. Since k is incrementing, any edge that was removed in finding k-community will also be removed in finding $(k+1)$-community, and the whole graph does not need to be duplicated in each iteration. Although the worst case time complexity of this algorithm is $O(m\Delta^3)$, it might be faster than the binary search in practice, as it operates with a single copy of the graph, whereas the binary search copies the graph $O(\log \Delta)$ times.

Subsequently, in line 4, `ScaleReduction1`$(G, \omega_{\text{upper}})$ finds the maximum $(\omega_{\text{upper}} - 1)$-core or the maximum $(\omega_{\text{upper}} - 2)$-community of G. If the number of vertices in the resulting graph G_{upper} is sufficiently small ($K = 12,000$ was used in experiments), then a lower bound ω_{lower} on the clique number is obtained using the procedure `FindMaxCliqueExact`, which can be any exact algorithm for the maximum clique problem that performs well on graphs with up to K vertices. For this paper, we use the algorithm developed by Östergård (2002). Since G_{upper} is not guaranteed to contain a maximum clique of G, we go a step further to obtain a reduced graph G_{lower} in line 10 that has the same clique number as G. Such a graph can be obtained by finding either the maximum $(\omega_{\text{lower}} - 1)$-core

or the maximum $(\omega_{\text{lower}} - 2)$-community of G in `ScaleReduction2`$(G, \omega_{\text{lower}})$. Finally, we use an exact algorithm (Östergård 2002) to find the clique number of G_{lower}. We considered six variations of Algorithm 2 that depend on the

Algorithm 2 `FindClique`(G): Algorithm to find a maximum clique of G

```
1: ω_lower ← HeuristicClique(G)
2: ω_upper ← upperBound(G, ω_lower, Δ)
3: if ω_lower < ω_upper then
4:     G_upper ≡ (V_upper, E_upper) ← ScaleReduction1(G, ω_upper)
5:     if |V_upper| ≤ K then
6:         ω_lower ← max{ω_lower, FindMaxCliqueExact(G_upper)}
7:     end if
8: end if
9: if ω_lower < ω_upper then
10:    G_lower ≡ (V_lower, E_lower) ← ScaleReduction2(G, ω_lower)
11:    for each connected component G' = (V', E') of G_lower do
12:        if |V'| ≤ K then
13:            ω_lower ← max{ω_lower, FindMaxCliqueExact(G')}
14:        end if
15:    end for
16:    if the clique number of each connected component is found then
17:        ω_upper ← ω_lower
18:    end if
19: end if
20: return [ω_lower, ω_upper]
```

choices of the upper bound search (binary or linear) and the scale reduction procedures. The core-based scale reduction finds the maximum $(\omega_{\text{upper}}-1)$-core of G in `ScaleReduction1`$(G, \omega_{\text{upper}})$ and the maximum $(\omega_{\text{lower}} - 1)$-core of G in `ScaleReduction2`$(G, \omega_{\text{lower}})$. Similarly, the community-based scale reduction finds the maximum $(\omega_{\text{upper}}-2)$-community of G in `ScaleReduction1`$(G, \omega_{\text{upper}})$ and the maximum $(\omega_{\text{lower}} - 2)$-community of G in `ScaleReduction2`$(G, \omega_{\text{lower}})$. Finally, the hybrid scale reduction finds the maximum $(\omega_{\text{upper}} - 1)$-core of G in `ScaleReduction1`$(G, \omega_{\text{upper}})$ and the maximum $(\omega_{\text{lower}} - 2)$-community of G in `ScaleReduction2`$(G, \omega_{\text{lower}})$.

3.1. Quality of k-community-based upper bound in power-law random graphs

While in general the community-based upper bound on the clique number can be arbitrarily larger than the clique number, a rather tight asymptotic bound can be established for power-law random graphs. A graph G is called a power-law graph if the number of vertices with degree q is proportional to $q^{-\alpha}$, where $\alpha \in (1, 3)$ is a constant. Power-law graphs are ubiquitous in nature, and many graphs studied in literature have been empirically found to follow this struc-

ture (Newman 2003). We characterize the quality of the upper bound ω_{upper} on the clique number obtained in the procedure upperBound$(G, \omega_{\text{lower}}, \Delta)$ of Algorithm 2 for power-law random graphs. We use the hidden variable ensemble model for generating power-law random graphs (Bianconi and Marsili 2006):

1. A hidden continuous variable q_i is assigned to each vertex i according to the power-law distribution.

2. Each pair of vertices with hidden variables q and q' are linked with probability
$$r(q, q') = \frac{qq'}{\bar{q}n}, \tag{1}$$
where $\bar{q}$ is the expectation of q, equivalently the average degree.

To ensure that the linking probabilities $r(q, q')$ are less than 1, we introduce a cutoff $Q = \sqrt{\bar{q}n}$ on the power-law distribution. Hence, the hidden variable distribution is as follows, where ρ_0 is a scaling factor used to make the probabilities sum to one.

$$\rho(q) = \begin{cases} \rho_0 q^{-\alpha}, & q \in [1, Q]; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Furthermore, the cutoff Q can be estimated as $Q \sim n^{1/\alpha}, \alpha \in (1, 2]$ and $Q \sim n^{1/2}, \alpha \in (2, 3)$. It has been shown by Bianconi and Marsili (2006) that

$$\omega = \Omega(n^{1/3\alpha}) \quad \text{and } \omega = O(n^{1/2\alpha}) \qquad \text{if } \alpha \in [1, 2); \tag{3}$$
$$\omega = \Omega(n^{(3-\alpha)/4}) \text{ and } \omega = O(n^{(3-\alpha)/6}) \text{ if } \alpha \in (2, 3). \tag{4}$$

With this model in mind, in the rest of this section we establish some asymptotic results that hold true with high probability – that is, the probability tends to 1 as the number n of vertices in the random graph goes to infinity.

Theorem 2. *Given a power-law random graph generated from the hidden variable ensemble model with coefficient $\alpha \in (1, 2) \cup (2, 3)$, the community-based upper bound ω_{upper} on the clique number is $O(\omega^3)$ with high probability.*

Proof. The theorem follows directly from Lemmata 1 and 2 below. □

Lemma 1. *For a power-law random graph with exponent $\alpha \in [2, 3)$ described using the hidden variable ensemble, the community-based upper bound ω_{upper} on the clique number is $O(n^{(3-\alpha)/2})$ with high probability. Furthermore, $\omega_{\text{upper}} = O(\omega^3)$ for $\alpha \in (2, 3)$.*

Proof. Let E_k denote the set of edges in the k-community of G, and E_k^t denote the set of edges remaining after t iterations of removing edges with endpoints having less than k common neighbors. Further, let G_k^t denote the subgraph $G[E_k^t]$ induced by the set of edges E_k^t. We proceed by showing that $|E_k| \longrightarrow 0$ with high probability for $k > \Theta(n^{(3-\alpha)/2})$. Consider the probability that vertices

i and j with hidden variables q_i and q_j are both connected to vertex s with hidden variable q_s in G:

$$w_n(i,j,q_s) = P(\{i,s\} \in E,\ \{j,s\} \in E) = r(q_i,q_s)r(q_j,q_s) = \frac{q_i q_s}{\bar{q}n}\frac{q_j q_s}{\bar{q}n} = \frac{q_i q_j}{\bar{q}^2 n^2} q_s^2. \tag{5}$$

The probability that i and j are both adjacent to a randomly chosen vertex in G is

$$w_n(i,j) = \int_1^Q \rho(q_s) w(i,j,q_s) dq_s \simeq \int_1^{\sqrt{n}} \rho_0 q_s^{-\alpha} \frac{q_i q_j}{\bar{q}^2 n^2} q_s^2 dq_s \simeq \frac{\rho_0 q_i q_j n^{-(\alpha+1)/2}}{3-\alpha}. \tag{6}$$

Let $\eta(i,j)$ denote the number of common neighbors that i and j have in G. We can show that

$$\mathbb{E}[\eta(i,j)] = \mu_{\eta(i,j)} = \frac{\rho_0 q_i q_j n^{(1-\alpha)/2}}{3-\alpha}, \tag{7}$$

$$\mathbb{V}ar[\eta(i,j)] = \sigma^2_{\eta(i,j)} \simeq \frac{\rho_0 q_i q_j n^{(1-\alpha)/2}}{3-\alpha}. \tag{8}$$

Now, we can observe the following using the one-sided Chebyshev's inequality,

$$P(\eta(i,j) \geq k) \leq \begin{cases} \frac{\sigma^2_{\eta(i,j)}}{(k-\mu)^2}, & k > \mu_{\eta(i,j)} \\ 1, & k < \mu_{\eta(i,j)}. \end{cases} \tag{9}$$

Note that since $q_i, q_j \in [1, \sqrt{n}]$, when $k > \Theta(n^{(3-\alpha)/2})$, we have $k \gg \mu_{\eta(i,j)}$. Thus,

$$P(\eta(i,j) \geq k) \leq \frac{\sigma^2_{\eta(i,j)}}{k^2}. \tag{10}$$

Now consider the expected number of neighbors vertex i will have in the graph G_k^1 as $n \longrightarrow \infty$.

$$\mathbb{E}[|N_{G_k^1}(i)|] = \int_1^{\sqrt{n}} n\rho(q_j) r(q_i,q_j) P(\eta(i,j) \geq k) dq_j \tag{11}$$

$$\leq \int_1^{\sqrt{n}} n\rho_0 q_j^{-\alpha} \frac{q_i q_j}{\bar{q}n} \frac{\sigma^2_\eta}{k^2} dq_j \leq \frac{n^{(1-\alpha)/2}\rho_0 q_i^2}{(3-\alpha)k^2} n^{(3-\alpha)/2} \leq c\frac{q_i^2 n^{2-\alpha}}{k^2} = O(1) \tag{12}$$

since $k > \Theta(n^{(3-\alpha)/2})$ and $q_i^2 \leq n,\ \forall i$. Similarly, the variance of the number of neighbors can be found to be

$$\mathbb{V}ar[|N_{G_k^1}(i)|] \leq c\frac{q_i^2 n^{2-\alpha}}{k^2}\left(1 - \frac{q_i^2 n^{1-\alpha}}{k^2}\right) = O(1). \tag{13}$$

Thus, using the one-sided Chebyshev's inequality, we can claim that as $n \longrightarrow \infty$,

$$P(|N_{G_k^1}(i)| \geq k) \longrightarrow 0 \tag{14}$$

and further that

$$P(|N_{G_k^2}(i)| = 0) \longrightarrow 1. \tag{15}$$

Thus, with high probability all the edges in G_k will be deleted, leaving the k-community empty for any $k > \Theta(n^{(3-\alpha)/2})$. Hence, the upper bound ω_{upper} is $O(n^{(3-\alpha)/2})$ with probability tending to 1 as $n \longrightarrow \infty$. From equation (4), we can deduce that $\omega_{\text{upper}} = O(\omega^3)$ with high probability. □

Lemma 2. *For a power-law random graph with exponent $\alpha \in (1,2)$ obtained using the hidden variable ensemble, the community-based upper bound ω_{upper} on the clique number is $O(n^{1/\alpha})$ with high probability. Furthermore, $\omega_{\text{upper}} = O(\omega^3)$ for $\alpha \in (1,2)$.*

Proof. We proceed in a similar manner as Lemma 1. Using the same notation, we can show that

$$w_n(i,j) \simeq \frac{\rho_0 q_i q_j n^{3/\alpha - 1}}{(3-\alpha)(\bar{q}n)^2} \simeq \frac{\rho_0 q_i q_j n^{-1/\alpha - 1}}{(3-\alpha)} \tag{16}$$

and

$$\mathbb{E}[\eta(i,j)] = \mu_{\eta(i,j)} = \frac{\rho_0 q_i q_j n^{-1/\alpha}}{3-\alpha}, \tag{17}$$

$$\mathbb{V}ar[\eta(i,j)] = \sigma^2_{\eta(i,j)} = \frac{\rho_0 q_i q_j n^{-1/\alpha}}{3-\alpha}. \tag{18}$$

Note since $q_i, q_j \in [1, n^{1/\alpha}]$, when $k > \Theta(n^{1/\alpha})$, $k \gg \mu_{\eta(i,j)}$, and we have

$$P(\eta(i,j) \geq k) \leq \frac{\sigma^2_{\eta(i,j)}}{k^2}. \tag{19}$$

Now consider the expected number of neighbors vertex i will have in the graph G_k^1 as $n \longrightarrow \infty$.

$$\mathbb{E}[|N_{G_k^1}(i)|] = \int_1^{n^{1/\alpha}} n\rho(q_j) r(q_i, q_j) P(\eta(i,j) > k) dq_j \tag{20}$$

$$\leq \int_1^{n^{1/\alpha}} \frac{n^{1-3/\alpha} \rho_0^2 q_i^2}{(3-\alpha)k^2} q_j^{2-\alpha} dq_j \leq \frac{n^{1-3/\alpha} \rho_0^2 q_i^2}{(3-\alpha)^2 k^2} n^{3/\alpha - 1} \leq c \frac{q_i^2}{k^2} = O(1) \tag{21}$$

since $k > \Theta(n^{1/\alpha})$ and $q_i < n^{1/\alpha}$. Finding $\mathbb{E}[|N_{G_k^1}(i)|]$ and $\mathbb{V}ar[|N_{G_k^1}(i)|]$ and using the one-sided Chebyshev's inequality as in the proof of Lemma 1, we can deduce that all the edges in G_k will be deleted, leaving the k-community of G empty for any $k > \Theta(n^{1/\alpha})$. Thus, the upper bound ω_{upper} is $O(n^{1/\alpha})$. We obtain $\omega_{\text{upper}} = O(\omega^3)$ using equation (3). □

3.2. Computational Results

All computational experiments reported in this paper (except for Table 4) were conducted on a *Dell Precision WorkStation T7500*® computer running Windows 7, with two Intel® Xeon® E5620 2.40 GHz quad-core processors and 12 GB RAM. The algorithms were implemented in the C++ programming language using Microsoft Visual Studio 2008 environment (except for Table 4).

The test cases were obtained from the Stanford Large Network Dataset Collection (SNAP Last accessed: February 2017), referred to as the SNAP, and the 10th DIMACS implementation challenge (DIMACS10 Last accessed: February 2017). These databases have a collection of large networks of sizes ranging from tens of thousands of vertices and edges to tens of millions of vertices and edges. They include social networks, web graphs, road networks, internet networks, citation networks, collaboration networks, random geometric graphs, and communication networks. The multitude of domains these networks originate from, along with the very large sizes of the networks, make the two datasets suitable candidates for performing computational studies for our algorithm. For conciseness, we consider all the networks that have at least 30,000 vertices and a few cases with fewer vertices. The networks in the database that were directed graphs were converted to an undirected graph by replacing each directed edge with an undirected edge.

Table 1 describes the networks from the two datasets that were used for this study. The networks in the DIMACS dataset were further classified into two categories based on whether they follow a heavy tail degree distribution or not. Table 2 compares the upper bounds based on k-cores and k-communities. The number of vertices remaining in the corresponding $(\omega_{\text{upper}} - 1)$-core and $(\omega_{\text{upper}} - 2)$-community are also reported. In addition, the table provides the lower bounds ω_{lower} found in the course of Algorithm 2, along with the number of vertices remaining in the corresponding $(\omega_{\text{lower}} - 1)$-core and $(\omega_{\text{lower}} - 2)$-community.

The largest graph considered, *uk-2002*, consumed 4.1 GB of hard drive space as a text file. The implementation read the graph into memory, at which point the process was using 3.8 GB of memory. Reading the graph from the input file took 789 seconds, which is surprisingly longer than the time to solve maximum clique in five out of the six approaches. The implementation's peak memory usage when computing a maximum clique of *uk-2002* was 5.4 GB.

It can be seen that compared to the k-core upper bound, the k-community upper bounds are significantly lower, almost by a factor of 2. This is because the k-community is a much tighter relaxation of a clique. Furthermore, the number of vertices remaining in the corresponding $(\omega_{\text{upper}} - 1)$-core and $(\omega_{\text{upper}} - 2)$-community provide further evidence of the tightness of k-communities. The lower bounds ω_{lower} were obtained by running an exact algorithm on the corresponding $(\omega_{\text{upper}} - 1)$-cores and $(\omega_{\text{upper}} - 2)$-communities of the graphs. In many cases the $(\omega_{\text{upper}} - 1)$-cores have a large number of vertices, and the lower bounds cannot be found this way. In such cases, the lower bound obtained from the greedy algorithm is reported (marked by an asterisk).

Table 1: Description of the various networks used for computations. More information about the graphs can be obtained from (Snap Last accessed: February 2017) and (Dimacs10 Last accessed: February 2017).

Network Type	Example/Description
SNAP	
Social Networks	Epinions.com: Who-trusts-whom network of Epinions.com.
	Slashdot Slashdot social network for a month.
	Wikipedia who-votes-on-whom network.
Communication	Email network from a EU research institution.
	Wikipedia talk (communication) network.
Citation Networks	Citation network among US Patents.
	Arxiv High Energy Physics paper citation network.
Web graphs	Web graph of Stanford.edu.
	Web graph from Google.
Product Co-purchasing	Amazon product co-purchasing network for a day.
Internet P2P	Gnutella peer to peer network for a day.
DIMACS10-HeavyTail	
Clustering	Used as benchmarks in the graph clustering.
Coauthors	Social networks are created from co-authorships and citations.
Random Geometric	Generated from random points in the unit square. Edges connect vertices whose Euclidean distance is below $0.55\log(n)/n$.
Kronecker	Synthetic graphs created with the Kronecker generator.
DIMACS10-QuasiRegular	
Matrix	Florida Sparse Matrix Collection.
Walshaw	Benchmarks for graph partitioning algorithms.

Table 3 provides the maximum clique sizes found by Algorithm 2, with the time taken by six variants of the algorithm. These variants differ in the clique relaxation used for scale reduction (k-core, k-community, and hybrid) and the search procedure used to find the upper bounds (linear and binary search).

It can be observed that for almost all the graphs, the maximum clique was found within a few minutes. The relative tightness of k-communities when compared to k-cores is apparent not just from the upper bounds found in Table 2, but also from the fact that many instances that could not be solved using a k-core reduction were solved by the k-community reduction. The linear upper bound search, which was introduced as a less memory-intensive algorithm targeting large graphs, does prove to be effective in reducing the time taken by the algorithm for the larger instances. A glance at Table 2 also suggests that the upper bounds are fairly tight, and that the lower bounds obtained are very close to the clique number. Amongst the three variants, the hybrid scale reduction method with linear search seems to perform the best overall.

The results presented in this table also highlight the main contribution of this paper in that we are able to obtain the maximum cliques for very large scale graphs with a proof of optimality for all the graphs tested. The pool of test instances taken is diverse, with both power-law (most of the SNAP and DIMACS-HeavyTail instances) and fairly regularly-structured (DIMACS10-QuasiRegular instances) graphs present. Also note that although the residual

Table 2: Comparison of the upper bounds (ω_{upper}) and lower bounds (ω_{lower}) obtained by the k-core scheme vs the k-community scheme. Comparison of the number of vertices in the corresponding $(\omega_{\text{upper}} - 1)$-core, $(\omega_{\text{upper}} - 2)$-community, $(\omega_{\text{lower}} - 1)$-core, and $(\omega_{\text{lower}} - 2)$-community, ($n_{\omega_{\text{upper}}}$ and $n_{\omega_{\text{lower}}}$) is also included. The $n_{\omega_{\text{upper}}}$ and $n_{\omega_{\text{lower}}}$ values are in bold when they are larger than $12,000$. An asterisk marks the ω_{lower} value when the best lower bound was obtained from the greedy algorithm.

			k-core				k-comm			
Graph	n	m	ω_{lower}	ω_{upper}	$n_{\omega_{\text{lower}}}$	$n_{\omega_{\text{upper}}}$	ω_{lower}	ω_{upper}	$n_{\omega_{\text{lower}}}$	$n_{\omega_{\text{upper}}}$
SNAP										
Wiki-Vote	7,115	100,762	17	54	2316	336	17	23	458	50
p2p-Gnutella04	10,876	39,994	3	8	8,379	365	4	4	12	12
p2p-Gnutella25	22,687	54,705	4	6	9,764	6,091	4	4	25	25
p2p-Gnutella24	26,518	65,369	4	6	11,478	7,480	4	4	41	41
Cit-HepTh	27,770	352,285	22	38	7,278	52	21	30	366	48
Cit-HepPh	34,546	420,877	18	31	11,284	40	18	25	193	36
p2p-Gnutella30	36,682	88,328	3 *	8	**20,194**	14	4	4	42	42
p2p-Gnutella31	62,586	147,892	4 *	7	**24,222**	1,004	4	4	57	57
soc-Epinions1	75,879	405,740	23	68	5,004	486	23	33	402	61
Slashdot0811	77,360	469,180	26	55	5,050	129	26	35	164	87
Slashdot0902	82,168	504,230	27	56	5,043	134	27	36	165	96
Amazon0302	262,111	899,792	7	7	286	286	7	7	105	105
Email-EuAll	265,214	364,481	16	38	1,691	292	16	20	157	62
web-Stanford	281,903	1,992,636	18 *	72	34,325	387	61	62	128	64
web-NotreDame	325,729	1,090,108	155	156	1,367	1,367	155	155	155	155
Amazon0312	400,727	2,349,869	9 *	11	**244,256**	**27,046**	11	11	4,534	4,534
Amazon0601	403,394	2,443,408	11 *	11	**32,886**	**32,886**	11	11	5,361	5,361
Amazon0505	410,236	2,439,437	8 *	11	**295,845**	**32,632**	11	11	4,878	4,878
web-BerkStan	685,230	6,649,470	201	202	392	392	201	201	392	392
web-Google	875,713	4,322,051	44	45	103	48	44	44	48	48
WikiTalk	2,394,385	4,659,565	26	132	**15,807**	700	26	53	1,559	237
cit-Patents	3,774,768	16,518,947	10	65	**354,843**	106	10	36	3,131	83
DIMACS10-HeavyTail										
as-22july06	22,963	48,436	17	26	144	71	17	17	45	45
cond-mat-2005	40,421	175,691	30	30	30	30	30	30	30	30
kron_g500-simple-logn16	65,536	2,456,071	136	433	6,885	694	136	285	2,513	676
caidaRouterLevel	192,244	609,066	16 *	33	4,021	92	17	19	58	36
coAuthorsCiteseer	227,320	814,134	87	87	87	87	87	87	87	87
citationCiteseer	268,495	1,156,647	10	16	**35,093**	67	13	13	13	13
coAuthorsDBLP	299,067	977,676	115	115	115	115	115	115	115	115
cnr-2000	325,557	2,738,969	84	84	86	86	84	84	86	86
coPapersCiteseer	434,102	16,036,720	845	845	845	845	845	845	845	845
coPapersDBLP	540,486	15,245,729	337	337	337	337	337	337	337	337
eu-2005	862,664	16,138,468	387	389	405	405	387	387	391	391
in-2004	1,382,908	13,591,473	489	489	491	491	489	489	490	490
rgg_n_2_21_s0	2,097,152	14,487,995	19	19	19	19	19	19	19	19
rgg_n_2_22_s0	4,194,304	30,359,198	20	20	20	20	20	20	20	20
rgg_n_2_23_s0	8,388,608	63,501,393	21	21	22	22	21	21	22	22
rgg_n_2_24_s0	16,777,216	132,557,200	21	21	82	82	21	21	44	44
uk-2002	18,520,486	261,787,258	944	944	944	944	944	944	944	944
DIMACS10-QuasiRegular										
G_n_pin_pout	100,000	501,198	3 *	8	**99,942**	**74,227**	4	4	4	4
preferentialAttachment	100,000	499,985	6 *	6	**100,000**	**100,000**	6	6	7	7
smallworld	100,000	499,998	5 *	8	**100,000**	**99,737**	6 *	6	**14,749**	**14,749**
luxembourg.osm	114,599	119,666	2 *	3	**114,599**	**93,000**	3	3	204	204
wave	156,317	1,059,331	5 *	9	**156,311**	**119,747**	6	7	389	9
audikw1	943,695	38,354,076	36 *	48	**937,779**	**687,633**	36	39	**185,805**	135
ldoor	952,203	22,785,136	21 *	35	**952,203**	**900,844**	21 *	21	**952,203**	**952,203**
ecology1	1,000,000	1,998,000	2 *	3	**1,000,000**	**1,000,000**	2 *	2	**1,000,000**	**1,000,000**
belgium.osm	1,441,295	1,549,970	3	4	**1,238,894**	5	3	3	7,113	7,113
333SP	3,712,815	11,108,633	3 *	5	**3,712,815**	**2,261,408**	4	4	28	28
cage15	5,154,859	47,022,346	6 *	26	**5,135,355**	**27,712**	6 *	6	**520,172**	**520,172**

Table 3: Maximum clique sizes found by Algorithm 2, and the time (in CPU seconds) taken by k-community, k-core, and hybrid scale reductions when using binary and linear search for finding the upper bound. The best time of computing an optimal solution for each instance is shown in bold. The cases in which optimality of the clique found could not be validated are shown in parentheses.

				Binary			**Linear**		
Graph	n	m	ω	**k-Core**	**k-Comm**	**Hybrid**	**k-Core**	**k-Comm**	**Hybrid**
SNAP									
Wiki-Vote	7,115	100,762	17	0.20	**0.19**	**0.19**	**0.19**	0.20	**0.19**
p2p-Gnutella04	10,876	39,994	4	1.42	**0.06**	0.66	0.97	**0.06**	0.97
p2p-Gnutella25	22,687	54,705	4	2.56	0.06	1.34	2.54	**0.05**	1.31
p2p-Gnutella24	26,518	65,369	4	2.95	**0.08**	1.28	2.87	0.09	1.19
Cit-HepTh	27,770	352,285	23	1.40	0.92	**0.42**	1.42	0.92	1.42
Cit-HepPh	34,546	420,877	19	2.79	0.23	0.81	2.75	**0.22**	1.33
p2p-Gnutella30	36,682	88,328	4	(0.16)	0.20	0.14	(0.08)	0.19	**0.09**
p2p-Gnutella31	62,586	147,892	4	(0.20)	**0.03**	0.20	(0.08)	**0.03**	0.09
soc-Epinions1	75,879	405,740	23	1.01	0.97	**0.67**	1.39	0.95	1.39
Slashdot0811	77,360	469,180	26	0.97	**0.22**	0.53	1.50	**0.22**	1.51
Slashdot0902	82,168	504,230	27	1.00	**0.23**	0.52	1.50	**0.23**	1.50
Amazon0302	262,111	899,792	7	1.76	0.75	1.76	**0.28**	0.34	**0.28**
Email-EuAll	265,214	364,481	16	0.22	**0.20**	**0.20**	0.22	0.22	0.22
web-Stanford	281,903	1,992,636	61	5.18	4.07	5.16	2.43	4.07	**2.42**
web-NotreDame	325,729	1,090,108	155	**0.27**	0.59	0.28	0.28	0.59	0.28
Amazon0312	400,727	2,349,869	11	5.32	2.85	5.24	**2.01**	2.06	54.82
Amazon0601	403,394	2,443,408	11	4.88	1.79	5.45	**1.31**	1.78	40.12
Amazon0505	410,236	2,439,437	11	4.91	2.82	5.05	1.83	**1.70**	25.48
web-BerkStan	685,230	6,649,470	201	13.26	26.54	13.65	10.75	26.61	**10.51**
web-Google	875,713	4,322,051	44	4.15	3.74	4.13	1.87	3.31	**1.83**
WikiTalk	2,394,385	4,659,565	26	(10.56)	**9.91**	13.39	(8.33)	**9.91**	11.23
cit-Patents	3,774,768	16,518,947	11	(31.75)	20.64	32.51	(18.22)	**16.63**	18.81
DIMACS10-HeavyTail									
as-22july06	22,963	48,436	17	0.02	0.03	0.03	**0.01**	0.02	0.02
cond-mat-2005	40,421	175,691	30	**0.22**	0.25	0.33	0.28	0.27	0.27
kron_g500-simple-logn16	65,536	2,456,071	136	656.34	769.25	657.44	656.99	767.90	**656.01**
caidaRouterLevel	192,244	609,066	17	0.44	**0.13**	0.45	0.44	**0.13**	0.45
coAuthorsCiteseer	227,320	814,134	87	0.70	0.48	0.47	0.70	**0.44**	0.76
citationCiteseer	268,495	1,156,647	13	(2.07)	**0.89**	2.11	1.44	0.90	1.44
coAuthorsDBLP	299,067	977,676	115	**0.37**	0.42	0.45	0.39	0.41	0.41
cnr-2000	325,557	2,738,969	84	19.66	14.51	19.63	**4.15**	15.04	**4.15**
coPapersCiteseer	434,102	16,036,720	845	**3.42**	4.98	3.43	3.54	4.98	**3.42**
coPapersDBLP	540,486	15,245,729	337	1.97	3.42	**1.93**	**1.93**	3.46	1.95
eu-2005	862,664	16,138,468	387	83.74	279.31	84.04	**15.54**	1,644.54	15.59
in-2004	1,382,908	13,591,473	489	**15.43**	54.88	15.59	15.49	54.87	15.52
rgg_n_2_21_s0	2,097,152	14,487,995	19	1.29	1.73	1.29	**1.28**	1.70	**1.28**
rgg_n_2_22_s0	4,194,304	30,359,198	20	2.65	3.25	2.65	2.67	3.21	**2.64**
rgg_n_2_23_s0	8,388,608	63,501,393	21	**4.99**	5.07	5.01	**4.99**	5.07	**4.99**
rgg_n_2_24_s0	16,777,216	132,557,200	21	25.99	**17.00**	26.40	20.14	17.15	20.23
uk-2002	18,520,486	261,787,258	944	182.16	377.79	182.49	146.52	4,013.36	**145.30**
DIMACS10-QuasiRegular									
G_n_pin_pout	100,000	501,198	4	(0.39)	0.45	0.70	(0.19)	**0.39**	0.59
preferentialAttachment	100,000	499,985	6	0.94	0.33	0.92	**0.14**	0.31	0.23
smallworld	100,000	499,998	6	(0.28)	0.31	0.52	(0.11)	**0.22**	2.04
luxembourg.osm	114,599	119,666	3	(0.28)	0.17	0.30	(0.22)	**0.09**	0.30
wave	156,317	1,059,331	6	(0.80)	1.15	1.92	(0.34)	**0.80**	39.19
audikw1	943,695	38,354,076	36	(27.39)	19.11	48.48	(7.38)	**17.41**	97.20
ldoor	952,203	22,785,136	21	(9.08)	39.81	45.24	(2.62)	**21.12**	223.96
ecology1	1,000,000	1,998,000	2	(1.58)	1.25	1.81	(0.76)	**0.81**	1.70
belgium.osm	1,441,295	1,549,970	3	2.20	3.81	2.31	1.54	2.18	**1.54**
333SP	3,712,815	11,108,633	4	(70.03)	18.10	136.85	(116.43)	**9.39**	85.36
cage15	5,154,859	47,022,346	6	(16.61)	17.66	44.62	(14.13)	**12.12**	27.08

graphs ($(\omega_{\text{upper}}-2)$-communities and $(\omega_{\text{upper}}-1)$-cores) are quite large for the DIMACS10-QuasiRegular graphs, the greedy clique is the same size as the upper bound found, not requiring an exact algorithm at all. To the best of our knowledge, the current paper represents the first published attempt to solve the maximum clique problem to optimality in very large scale networks in a systematic fashion. In other related publications the focus was on computing all maximal cliques (Modani et al. 2010, Lu et al. 2010, Cheng et al. 2011, Eppstein and Strash 2011). In particular, Modani et al. (2010) used scale-reduction techniques similar to those proposed in this paper on two graph instances representing telecommunication data in order to enumerate all maximal cliques of size exceeding a given threshold. Lu et al. (2010) propose a distributed algorithm and applied it to 11 SNAP instances on an 80-node computer cluster. Cheng et al. (2011) developed an external memory algorithm and tested it on 4 instances. Finally, Eppstein and Strash (2011) compare the performance of several algorithms for enumerating all maximal cliques on large sparse graphs, including 13 SNAP instances. This implementation, which will be called ES, is used for comparison purposes.

Table 4 compares the runtimes of the hybrid approach with the implementation of Eppstein and Strash (2011) which is publicly available at `http://www.ics.uci.edu/~dstrash/quick-cliques.tar.gz`. It should be noted that ES solves a different problem—the problem of listing all maximal cliques, whereas our implementation finds a single maximum clique. Second, our initial implementation was written for a Windows machine, whereas their code is for Unix. In order to compare the implementations more fairly, we first ported our code to Unix. We then modified our code to rely on the ES implentation to compute degeneracy. Then both codes were executed on the same machine running CentOS with one Intel® Xeon® W3520 2.67 GHz quad-core processor and 12GB RAM. (The other computational experiments reported in this paper were gathered from a different machine that runs Windows 7.) The ES implementation performs well in most cases. However, on some instances the ES implementation did not finish with 10 hours. This is due, in part, to large memory use; all 12 GB was being used for those instances.

4. Solving the Vertex Coloring Problem on Very Large Sparse Graphs

Although vertex coloring is a celebrated problem, most literature devoted to solving it focuses on small instances with up to a thousand vertices, with numerous benchmark instances still unsolved (Malaguti and Toth 2010). In this section, we look at the vertex coloring problem on large sparse instances. To the best of our knowledge, there are no published results for vertex coloring on graphs of the scales being considered in this paper. The largest graph for which the chromatic number is reported in a recent survey by Malaguti and Toth (2010) has 3,600 vertices. The primary pretext of the scale reduction method for

Table 4: A comparison of the hybrid approach with ES (Eppstein and Strash 2011). If the computation was not completed after 36,000 seconds, it was aborted and denoted in the table by "Time Limit."

Graph	n	m	ω	ES Time	Hybrid Time
SNAP					
Wiki-Vote	7,115	100,762	17	0.88	0.14
p2p-Gnutella04	10,876	39,994	4	0.04	0.01
p2p-Gnutella25	22,687	54,705	4	0.08	0.59
p2p-Gnutella24	26,518	65,369	4	0.08	0.88
Cit-HepTh	27,770	352,285	23	1.33	0.14
Cit-HepPh	34,546	420,877	19	1.13	0.14
p2p-Gnutella30	36,682	88,328	4	0.12	0.02
p2p-Gnutella31	62,586	147,892	4	0.23	0.07
soc-Epinions1	75,879	405,740	23	3.56	0.31
Slashdot0811	77,360	469,180	26	1.61	0.17
Slashdot0902	82,168	504,230	27	1.80	0.20
Amazon0302	262,111	899,792	7	1.43	0.28
Email-EuAll	265,214	364,481	16	0.89	0.18
web-Stanford	281,903	1,992,636	27	3.81	4.48
web-NotreDame	325,729	1,090,108	155	1.57	0.20
Amazon0312	400,727	2,349,869	11	3.92	2.15
Amazon0601	403,394	2,443,408	11	4.04	2.12
Amazon0505	410,236	2,439,437	11	4.04	1.90
web-BerkStan	685,230	6,649,470	201	14.98	8.70
web-Google	875,713	4,322,051	44	6.75	1.22
WikiTalk	2,394,385	4,659,565	26	165.94	7.53
cit-Patents	3,774,768	16,518,947	11	43.24	11.38
DIMACS10-HeavyTail					
as-22july06	22,963	48,436	17	0.08	0.01
cond-mat-2005	40,421	175,691	30	0.25	0.01
kron_g500-simple-logn16	65,536	2,456,071	136	Time Limit	524.49
caidaRouterLevel	192,244	609,066	17	1.26	0.34
coAuthorsCiteseer	227,320	814,134	87	1.16	0.08
citationCiteseer	268,495	1,156,647	13	2.52	0.54
coAuthorsDBLP	299,067	977,676	115	1.60	0.08
cnr-2000	325,557	2,738,969	84	5.89	3.03
coPapersCiteseer	434,102	16,036,720	845	37.66	2.87
coPapersDBLP	540,486	15,245,729	337	27.26	1.47
eu-2005	862,664	16,138,468	387	56.69	31.51
in-2004	1,382,908	13,591,473	489	77.10	11.53
rgg_n_2_21_s0	2,097,152	14,487,995	19	19.22	0.75
rgg_n_2_22_s0	4,194,304	30,359,198	20	41.37	1.57
rgg_n_2_23_s0	8,388,608	63,501,393	21	98.92	3.18
rgg_n_2_24_s0	16,777,216	132,557,200	21	Time Limit	14.06
uk-2002	18,520,486	261,787,258	944	Time Limit	87.60
DIMACS10-Quasi-Regular					
G_n_pin_pout	100,000	501,198	4	0.69	0.31
preferentialAttachment	100,000	499,985	6	0.72	0.22
smallworld	100,000	499,998	6	0.59	0.34
luxembourg.osm	114,599	119,666	3	0.20	0.19
wave	156,317	1,059,331	6	1.43	0.68
audikw1	943,695	38,354,076	36	79.18	14.75
ldoor	952,203	22,785,136	21	24.97	10.54
ecology1	1,000,000	1,998,000	2	2.19	0.66
belgium.osm	1,441,295	1,549,970	3	2.87	0.65
333SP	3,712,815	11,108,633	4	17.71	115.94
cage15	5,154,859	47,022,346	6	65.02	22.19

vertex coloring presented in this section can be established using the following observations.

Lemma 3. *Suppose the k-core of G has been properly colored. Then the remaining vertices in G can be colored using at most k colors.*

Proof. Let C_k be the k-core of G, and let $v_1, v_2, .., v_{n_k}$ be the order in which the vertices in G were removed to obtain C_k. This means that when the vertex v_i was removed, its degree in $G[C_k \cup \{v_{i+1}, .., v_{n_k}\}]$ was less than k. Now suppose we have a valid coloring for $G' = G[C_k]$ which uses colors $\{1, 2, .., c\}$. Then, since v_{n_k} is adjacent to less than k vertices in G', it can be assigned a color from $\{1, 2, .., k\}$ and added to G' without violating the properness of the coloring. This process can be repeated by coloring and adding $v_{n_k-1}, v_{n_k-2}, .., v_1$ to G' in the order provided to ensure that the new vertices added use only colors in $\{1, 2, .., k\}$. □

Theorem 3. *Let $C_k \subseteq V$ denote the k-core of a graph $G = (V, E)$. The following inequalities hold.*

$$\chi(G[C_k]) \leq \chi(G) \leq \max\{\chi(G[C_k]), k\} \tag{22}$$

Consequently, G is k-colorable if and only if $G[C_k]$ is k-colorable.

Proof. Let $G' = G[C_k]$. The first inequality holds since $\chi(G') \leq \chi(G)$ for any subgraph G' of G. The second inequality follows directly from Lemma 3. □

4.1. Scale Reduction Algorithm

With the above properties in mind, Algorithm 3 aims to solve the vertex coloring problem on large graphs if an effective algorithm `ExactColoring` is available for solving the vertex coloring problem on smaller graphs. The algorithm makes use of Theorem 3 to establish upper and lower bounds for the chromatic number. We start with k set to the largest integer k' such that the k'-core (denoted by C_k henceforth) is non-empty (i.e., k' is the degeneracy of G). Lower bounds are obtained using `ExactColoring` on $G' = G[C_k]$, and upper bounds by coloring the remaining vertices in G as done in the proof of Lemma 3. If the bounds are not tight, then we decrease k by 1, increasing the size of G'. `ExactColoring` on G' will provide a lower bound that can be no worse, and potentially better (since $\chi(G[C_{k-1}]) \geq \chi(G[C_k])$). Reducing k further increases the possibility of reaching one of the stopping criteria of Algorithm 3, $c_1 \geq k$. A better upper bound might also be obtained by coloring the remaining vertices of G starting with the colors obtained by optimally coloring G'. A major issue with the effectiveness of the algorithm is that as we reduce k, the size of the k-core(G) becomes larger, and can eventually result in `ExactColoring` being unable to obtain an optimal coloring within a reasonable time. If that is the case, then this algorithm fails to obtain the optimal coloring to the whole graph and we report the best upper and lower bounds found.

Algorithm 3 `Color`(G): Algorithm to find a vertex coloring of G

1: $\chi_{\text{lower}} \leftarrow \omega(G)$
2: $k' \leftarrow$ the degeneracy of G
3: $\chi_{\text{upper}} \leftarrow k' + 1$
4: $k \leftarrow k'$
5: **while** $\chi_{\text{lower}} < \chi_{\text{upper}}$ **do**
6: $\quad c_1 \leftarrow$ `ExactColoring`$(k-\text{core}(G))$
7: $\quad$ **if** $c_1 \geq k$ **then**
8: $\quad\quad$ **return** $[c_1, c_1]$
9: $\quad$ **else**
10: $\quad\quad$ Heuristically color the remaining vertices in G using $c_2 \leq k$ colors.
11: $\quad\quad \chi_{\text{lower}} \leftarrow \max\{\chi_{\text{lower}}, c_1\}$
12: $\quad\quad \chi_{\text{upper}} \leftarrow \min\{\chi_{\text{upper}}, \max\{c_1, c_2\}\}$
13: $\quad$ **end if**
14: $\quad k \leftarrow k - 1$
15: **end while**
16: **return** $[\chi_{\text{lower}}, \chi_{\text{upper}}]$

4.2. Heuristics for Improving Lower and Upper Bounds

Before employing the while loop in Algorithm 3, various steps can be taken to improve the lower and upper bounds on the chromatic number. It is possible to obtain improved upper bounds by using heuristic algorithms developed in the literature (Matula and Beck 1983, Brélaz 1979, Malaguti and Toth 2010). Furthermore, improved lower bounds can be obtained by selecting subgraphs G' of G that could potentially require many colors in an optimal coloring computable using `ExactColoring`. For improving the upper bound, we used heuristics to color the graph G. There are plenty of greedy algorithms that use different vertex orderings and color the vertices using one of the available colors. These heuristics are variations of the sequential greedy algorithm (SEQ) (Malaguti and Toth 2010) and often exhibit poor performance, but their speed is valuable when dealing with large graphs. For example, Matula and Beck (1983) consider an ordering based on core decomposition, which is to iteratively remove a least degree vertex from the graph, and to color vertices in the reverse order of their removal. This ordering has been originally proposed by Szekeres and Wilf (1968). Unlike other SEQ-based algorithms, this ordering ensures that the number of colors used does not exceed the degeneracy number of the graph by more than one (Szekeres and Wilf 1968). A more sophisticated technique is DSATUR, which orders vertices dynamically, coloring the vertex with most forbidden colors first (Brélaz 1979). For the computational results reported in this section, the following heuristics were used.

- **Degeneracy:** Coloring with $k'+1$ colors, where k' is the degeneracy of G.
- **SEQ:** Color vertices according to a given ordering,

  - **SEQ-Inorder:** Original vertex ordering;
  - **SEQ-Degree:** Descending order of degree;
  - **SEQ-Core:** Core decomposition-based ordering.

- **DSATUR:** Color the vertex with most forbidden colors first. If all colors are forbidden, add a new color.

In order to improve the lower bound on the chromatic number, we find the chromatic number of a subgraph G'. Again, there is a tradeoff involved in choosing how big a subgraph we select – a large subgraph will provide a better bound, but might not be tractable itself, and a small subgraph might not provide a good bound at all. In this paper, we use the following subgraphs:

- **Clique+Neighborhood:** Use a maximum clique as the initial subgraph, and if the size of the subgraph is less than the solvability threshold add neighboring vertices till the threshold is reached.
- **Core decomposition:** Keep removing a least-degree vertex till the total number of vertices left does not exceed the solvability threshold.
- **k-Community:** Use the k-community for the largest k such that k-community is non-empty.

The (solvability) threshold for the number of vertices to be used in all of the above sub-graphs is determined by the performance of the `ExactColoring` algorithm, and was set at 100 vertices for the results presented in this section. The `ExactColoring` algorithm used was 'Backtrack DSATUR'; its implementation is available from (Culberson Last accessed: February 2017).

4.3. Computational Results

For testing the effectiveness of our scale reduction approach, we used the same test instances that were used for the maximum clique problem. Table 5 reports the results, including the method that provided the best lower and upper bound for each instance. As can be seen from the table, we could find provably optimal coloring for 33 of 53 instances tested. Furthermore, for 30 instances out of the 33 that were solved (and for 45 out of all 53 instances), the best lower bound was found to be the same as the clique number. Note that the k-community approach has never yielded the best lower bound. For 20 of the 33 instances solved (27 of 53 total) the best computed upper bound was found using degeneracy. On 11 occasions the degeneracy bound was further improved in the while-loop of Algorithm 3. In 7 of these cases the improvement was sufficient for proving optimality.

5. Conclusion

This paper introduces scale reduction algorithms based on clique relaxations such as k-community and k-core to find the maximum cliques and vertex col-

Table 5: lower and upper bounds on the chromatic number as obtained by Algorithm 3. The instances where the optimal coloring was found are highlighted in bold.

Graph	n	m	ω	χ_{lower}	χ_{upper}	$k'+1$	%Gap	Best LB	Best UB	Time(s)
SNAP										
Wiki-Vote	7115	100762	17	19	24	54	20.83	CliqueNhood	DSATUR	604.67
p2p-Gnutella04	10876	39994	4	4	6	8	33.33	Clique	DSATUR	3.68
p2p-Gnutella25	22687	54705	4	4	6	6	33.33	Clique	Degeneracy	14.27
p2p-Gnutella24	26518	65369	4	4	6	6	33.33	Clique	Degeneracy	19.29
Cit-HepTh	27770	352285	23	23	25	38	8.00	Clique	while-loop	29.90
Cit-HepPh	34546	420877	19	19	21	31	9.52	Clique	while-loop	42.89
p2p-Gnutella30	36682	88328	4	4	6	8	33.33	Clique	while-loop	33.33
p2p-Gnutella31	62586	147892	4	4	6	7	33.33	Clique	DSATUR	5.26
soc-Epinions1	75879	405740	23	25	30	68	16.67	CliqueNhood	DSATUR	608.15
Slashdot0811	77360	469180	26	29	29	55	0.00	k-core	while-loop	2.70
Slashdot0902	82168	504230	27	29	29	56	0.00	k-core	while-loop	4.00
Amazon0302	262111	899792	7	7	7	7	0.00	Clique	Degeneracy	1.30
Email-EuAll	265214	364481	16	18	20	38	10.00	k-core	while-loop	642.18
web-Stanford	281903	1992636	61	61	61	72	0.00	Clique	while-loop	13.62
web-NotreDame	325729	1090108	155	155	155	156	0.00	Clique	SEQ-Inorder	0.80
Amazon0312	400727	2349869	11	11	11	11	0.00	Clique	Degeneracy	86.73
Amazon0601	403394	2443408	11	11	11	11	0.00	Clique	Degeneracy	68.32
Amazon0505	410236	2439437	11	11	11	11	0.00	Clique	Degeneracy	44.64
web-BerkStan	685230	6649470	201	201	201	202	0.00	Clique	SEQ-Inorder	27.32
web-Google	875713	4322051	44	44	44	45	0.00	Clique	SEQ-Inorder	4.47
WikiTalk	2394385	4659565	26	31	51	132	39.22	CliqueNhood	DSATUR	1221.04
cit-Patents	3774768	16518947	11	11	12	65	8.33	Clique	DSATUR	761.86
DIMACS10-HeavyTail										
as-22july06	22963	48436	17	17	17	26	0.00	Clique	while-loop	0.14
cond-mat-2005	40421	175691	30	30	30	30	0.00	Clique	Degeneracy	0.06
kron_g500-simple-logn16	65536	2456071	136	136	155	433	12.26	Clique	DSATUR	2842.85
rgg_n_2_17_s0	131072	728753	15	15	15	15	0.00	Clique	Degeneracy	0.17
caidaRouterLevel	192244	609066	17	17	17	33	0.00	Clique	while-loop	1.44
coAuthorsCiteseer	227320	814134	87	87	87	87	0.00	Clique	Degeneracy	0.30
citationCiteseer	268495	1156647	13	13	13	16	0.00	Clique	while-loop	1.55
coAuthorsDBLP	299067	977676	115	115	115	115	0.00	Clique	Degeneracy	0.36
cnr-2000	325557	2738969	84	84	84	84	0.00	Clique	Degeneracy	9.89
coPapersCiteseer	434102	16036720	845	845	845	845	0.00	Clique	Degeneracy	5.17
rgg_n_2_19_s0	524288	3269766	18	18	18	18	0.00	Clique	Degeneracy	1.43
coPapersDBLP	540486	15245729	337	337	337	337	0.00	Clique	Degeneracy	3.79
eu-2005	862664	16138468	387	387	387	389	0.00	Clique	SEQ-Inorder	68.64
rgg_n_2_20_s0	1048576	6891620	17	17	17	18	0.00	Clique	SEQ-Core	1.81
in-2004	1382908	13591473	489	489	489	489	0.00	Clique	Degeneracy	22.51
rgg_n_2_21_s0	2097152	14487995	19	19	19	19	0.00	Clique	Degeneracy	3.70
rgg_n_2_22_s0	4194304	30359198	20	20	20	20	0.00	Clique	Degeneracy	7.38
rgg_n_2_23_s0	8388608	63501393	21	21	21	21	0.00	Clique	Degeneracy	14.99
rgg_n_2_24_s0	16777216	1.33E+08	21	21	21	21	0.00	Clique	Degeneracy	50.21
uk-2002	18520486	2.62E+08	944	944	944	944	0.00	Clique	Degeneracy	330.59
DIMACS10-QuasiRegular										
G_n_pin_pout	100000	501198	4	4	8	8	50.00	Clique	Degeneracy	22.17
preferentialAttachment	100000	499985	6	6	6	6	0.00	Clique	Degeneracy	0.39
smallworld	100000	499998	6	6	8	8	25.00	Clique	Degeneracy	35.81
luxembourg.osm	114599	119666	2	3	3	3	0.00	CliqueNhood	Degeneracy	12.99
wave	156317	1059331	6	6	9	9	33.33	Clique	Degeneracy	98.59
audikw1	943695	38354076	36	36	44	48	18.18	Clique	DSATUR	4391.04
ldoor	952203	22785136	21	21	35	35	40.00	Clique	Degeneracy	4474.09
ecology1	1000000	1998000	2	2	2	3	0.00	Clique	SEQ-Inorder	2.65
belgium.osm	1441295	1549970	3	3	3	4	0.00	Clique	while-loop	6.33
333SP	3712815	11108633	3	4	5	5	20.00	CliqueNhood	Degeneracy	11734.50
cage15	5154859	47022346	6	6	13	26	53.85	Clique	DSATUR	36141.60

orings in large, low-density graphs. While we were able to solve the maximum clique problems on all the instances tested, the vertex coloring problem considered on the same instances appears to pose a more formidable challenge. Any advancements in exact algorithms for the maximum clique problem and the vertex coloring problem on smaller graphs will directly impact the performance of this methodology in a positive manner.

Acknowledgements

We would like to thank an Associate Editor and two reviewers for their suggestions that helped to significantly improve the paper. Partial support by the US Department of Energy Grant DE-SC0002051 and the US Air Force Office of Scientific Research Grant FA9550-12-1-0103 is gratefully acknowledged.

References

Abello, J., P. M. Pardalos, M. G. C. Resende. 1999. External memory algorithms 119–130.

Balas, E., J. Xue. 1996. Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. *Algorithmica* **15** 397–412.

Bianconi, G., M. Marsili. 2006. Emergence of large cliques in random scale-free networks. *Europhysics Letters* **74** 740–746.

Bollobás, B., A. Thomason. 1985. Random graphs of small order. *Annals of Discrete Mathematics* **28** 47–97.

Bomze, I. M., M. Budinich, P. M. Pardalos, M. Pelillo. 1999. The maximum clique problem. D.-Z. Du, P. M. Pardalos, eds., *Handbook of Combinatorial Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1–74.

Brélaz, D. 1979. New methods to color the vertices of a graph. *Commun. ACM* **22** 251–256.

Buchanan, A., J.L. Walteros, S. Butenko, P. M. Pardalos. 2013. Solving maximum clique in sparse graphs: an $O(nm + n2^{d/4})$ algorithm for d-degenerate graphs. *Optimization Letters* To appear.

Butenko, S., S. Trukhanov. 2007. Using critical sets to solve the maximum independent set problem. *Operations Research Letters* **35** 519–524.

Butenko, S., W. E. Wilhelm. 2006. Clique-detection models in computational biochemistry and genomics. *European Journal of Operational Research* **173** 1–17.

Campêlo, M., V. A. Campos, R. C. Corrêa. 2008. On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics* **156** 1097–1111.

Carraghan, R., P. M. Pardalos. 1990. An exact algorithm for the maximum clique problem. *Operations Research Letters* **9** 375–382.

Cheng, J., Y. Ke, A. W.-C. Fu, J. X. Yu, L. Zhu. 2011. Finding maximal cliques in massive networks. *ACM Trans. Database Syst.* **36** 21:1–21:34.

Cohen, J. D. 2008. Trusses: Cohesive subgraphs for social network analysis. Tech. rep., National Security Agency, Fort Meade, MD.

Corno, F., P. Prinetto, M. Sonza Reorda. 1995. Using symbolic techniques to find the maximum clique in very large sparse graphs. *Proceedings of the 1995 European conference on Design and Test*. EDTC '95, IEEE Computer Society, Washington, DC, USA, 320–324.

Culberson, J. C. Last accessed: February 2017. Culberson's Graph Coloring Programs.
http://webdocs.cs.ualberta.ca/ joe/Coloring/Colorsrc/index.html.

Culberson, J. C., F. Luo. 1996. Exploring the k-colorable landscape with iterated greedy. D.S. Johnson, M.A. Trick, eds., *Cliques, Coloring, and Satisfiability*, *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, vol. 26. American Mathematical Society, 245–284.

Dimacs10. Last accessed: February 2017. 10th DIMACS Implementation Challenge: Graph Partitioning and Graph Clustering. http://www.cc.gatech.edu/dimacs10/.

Eppstein, D., D. Strash. 2011. Listing all maximal cliques in large sparse real-world graphs. P. Pardalos, S. Rebennack, eds., *Experimental Algorithms*, *Lecture Notes in Computer Science*, vol. 6630. Springer Berlin / Heidelberg, 364–375.

Feige, U., J. Kilian. 1998. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences* **57** 187–199.

Gendreau, M., P. Soriano, L. Salvail. 1993. Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research* **41** 385–403.

Gualandi, S., F. Malucelli. 2012. Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS Journal on Computing* **24** 81–100.

Hansen, P., M. Labbé, D. Schindl. 2009. Set covering and packing formulations of graph coloring: Algorithms and first polyhedral results. *Discrete Optimization* **6** 135–147.

Håstad, J. 1999. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica* **182** 105–142.

Held, S., W. Cook, E. Sewell. 2012. Maximum-weight stable sets and safe lower bounds for graph coloring. *Mathematical Programming Computation* **4** 363–381.

Karp, R. M. 1972. Reducibility among combinatorial problems. R. E. Miller, J. W. Thatcher, eds., *Complexity of Computer Computations*. Plenum, New York, 85–103.

Katayama, K., A. Hamamoto, H. Narihisa. 2005. An effective local search for the maximum clique problem. *Information Processing Letters* **95** 503–511.

Lick, D.R., A.T. White. 1970. k-degenerate graphs. *Canad. J. Math* **22** 1082–1096.

Lu, L., Y. Gu, R. Grossman. 2010. dMaximalCliques: A distributed algorithm for enumerating all maximal cliques and maximal clique distribution. *IEEE International Conference on Data Mining Workshops (ICDMW)*. 1320–1327.

Malaguti, E., M. Monaci, P. Toth. 2011. An exact approach for the vertex coloring problem. *Discrete Optimization* **8** 174–190.

Malaguti, E., P. Toth. 2010. A survey on vertex coloring problems. *International Transactions in Operational Research* **17** 1–34.

Matula, D.W., L.L. Beck. 1983. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM* **30** 417–427.

Mehrotra, A., M. A. Trick. 1995. A column generation approach for graph coloring. *INFORMS Journal on Computing* **8** 344–354.

Méndez-Díaz, I., P. Zabala. 2006. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics* **154** 826–847.

Modani, N., K. Dey, S. Mukherjea, A. A. Nanavati. 2010. Discovery and analysis of tightly knit communities in telecom social networks. *IBM Journal of Research and Development* **54** 7:1–7:13.

Morgenstern, C. 1996. Distributed coloration neighborhood search. D.S. Johnson, M.A. Trick, eds., *Cliques, Coloring, and Satisfiability*, *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, vol. 26. American Mathematical Society, 335–358.

N. J. A. Sloane. Last accessed: February 2017. Challenge Problems: Independent Sets in Graphs.
http://www2.research.att.com/ njas/doc/graphs.html.

Newman, M. 2003. The structure and function of complex networks. *SIAM Review* **45** 167–256.

Östergård, P. R. J. 2002. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics* **120** 197–207.

Pardalos, P. M., T. Mavridou, J. Xue. 1998. The graph coloring problem: A bibliographic survey. D.-Z. Du, P. M. Pardalos, eds., *Handbook of Combinatorial Optimization*, vol. 2. Kluwer Academic Publishers, 331–395.

Protasi, B., R. Battiti, M. Protasi. 1995. Reactive local search for the maximum clique problem. Tech. rep., Algorithmica.

Shen, Y., D.T. Nguyen, Y. Xuan, M.T. Thai. 2012. New techniques for approximating optimal substructure problems in power-law graphs. *Theoretical Computer Science* **447** 107–119.

Snap. Last accessed: February 2017. Stanford Large Network Dataset Collection. http://snap.stanford.edu/data/.

Szekeres, G., H. S. Wilf. 1968. An inequality for the chromatic number of a graph. *Journal of Combinatorial Theory* **4** 1–3.

Tomita, E., T. Seki. 2003. An efficient branch-and-bound algorithm for finding a maximum clique. C. Calude, M. Dinneen, V. Vajnovszki, eds., *Discrete Mathematics and Theoretical Computer Science*, *Lecture Notes in Computer Science*, vol. 2731. Springer Berlin / Heidelberg, 278–289.

Wood, D. R. 1997. An algorithm for finding a maximum clique in a graph. *Operations Research Letters* **21** 211–217.

Zuckerman, D. 2007. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing* **3** 103–128.