Scalable Large Near-Clique Detection in Large-Scale Networks via Sampling

Michael Mitzenmacher
Harvard University
michaelm@eecs.harvard.edu

Jakub Pachocki
Carnegie Mellon University
pachocki@cs.cmu.edu

Richard Peng
MIT
rpeng@mit.edu

Charalampos E. Tsourakakis
Harvard University
babis@seas.harvard.edu

Shen Chen Xu
Carnegie Mellon University
shenchex@cs.cmu.edu

ABSTRACT

Extracting dense subgraphs from large graphs is a key primitive in a variety of graph mining applications, ranging from mining social networks and the Web graph to bioinformatics [41]. In this paper we focus on a family of poly-time solvable formulations, known as the *k-clique densest subgraph problem* (k-Clique-DSP) [57]. When $k = 2$, the problem becomes the well-known *densest subgraph problem* (DSP) [22, 31, 33, 39]. Our main contribution is a sampling scheme that gives *densest subgraph sparsifier*, yielding a randomized algorithm that produces high-quality approximations while providing significant speedups and improved space complexity. We also extend this family of formulations to bipartite graphs by introducing the (p, q)-*biclique densest subgraph problem* ((p,q)-Biclique-DSP), and devise an exact algorithm that can treat both clique and biclique densities in a unified way.

As an example of performance, our sparsifying algorithm extracts the 5-clique densest subgraph –which is a large-near clique on 62 vertices– from a large collaboration network. Our algorithm achieves 100% accuracy over five runs, while achieving an average speedup factor of over 10 000. Specifically, we reduce the running time from ∼2 107 seconds to an average running time of 0.15 seconds. We also use our methods to study how the k-clique densest subgraphs change as a function of time in time-evolving networks for various small values of k. We observe significant deviations between the experimental findings on real-world networks and stochastic Kronecker graphs, a random graph model that mimics real-world networks in certain aspects.

We believe that our work is a significant advance in routines with rigorous theoretical guarantees for scalable extraction of large near-cliques from networks.

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.

DOI: http://dx.doi.org/10.1145/2783258.2783385.

Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph Algorithms

General Terms

Theory, Experimentation

Keywords

Dense subgraphs; Near-clique extraction; Graph Mining

1. INTRODUCTION

Graph sparsifiers constitute a major concept in algorithmic graph theory. A sparsifier of a graph G is a sparse graph H that is similar to G in some useful way. For instance, Benczúr and Karger introduced the notion of cut sparsification to accelerate cut algorithms whose running time depends on the number of edges [15]. Spielman and Teng strengthened the notion of a cut sparsifier by introducing the more general notion of a spectral sparsifier [52]. Other graph sparsifiers include vertex sparsifiers [48], flow sparsifiers [8, 43], distance sparsifiers [53, 54] and triangle sparsifiers [58].

In this work we are interested in dense subgraph discovery, a major research topic in graph mining with a large number of related applications. To motivate dense subgraph discovery in the context of data mining, consider the following real-world scenario. Suppose we are given an undirected "who-calls-whom" graph, where vertices correspond to people and an edge indicates a phone-call exchange. Finding large near-cliques in this graph is particularly interesting, since they tend to be unusual and may be considered as a graph feature that indicates a group of collaborators/accomplices. More generally, a wide range of graph applications involve the discovery of large near-cliques [41]. In bioinformatics, dense subgraphs are used for detecting protein complexes in protein-protein interaction networks [11] and for finding regulatory motifs in DNA [30]. They are also used for expert team formation [19, 55], detecting link spam in Web graphs [32], graph compression [21], reachability and distance query indexing [36], insightful graph decompositions [51] and mining micro-blogging streams [9].

Contributions. Among the various existing formulations for dense subgraph discovery, we focus on the *k-clique dens-*

	k=2				k=3			
	$\|S\|$	f_e	err.	sp.	$\|S\|$	f_e	err.	sp.
Web-G	117	0.46	0.3	1.9	61.6	0.85	0.5	16.0
CA-As	582	0.09	1.5	2.5	136.4	0.48	0.6	24.4
P-B	301	0.19	4.1	1.6	105	0.52	3.3	42.8

Table 1: Average output sizes $|S|$, edge densities $f_e(S) = \frac{e(S)}{\binom{|S|}{2}}$, error $\left(1 - \frac{\rho_k(S)}{\rho_k^*}\right) \times 100\%$ (err.) and speedups (sp.) obtained from the Google Web-graph (Web-G), the astrophysics collaboration graph (CA-As) and the political blogs network (P-B) for $k = 2, 3$ over 5 runs using our randomized algorithm. For details see text of Section 1.

est subgraph problem (k-Clique-DSP), which is solvable in polynomial time for any constant $k \geq 2$ [57]. When $k = 2$, the problem is also known as the *densest subgraph problem* (DSP) [22, 33, 39]. We formally define these problems in Section 3.1. Our contributions are summarized as follows.

- We extend this family of problems by introducing the (p, q)-*biclique densest subgraph problem* ((p,q)-Biclique-DSP). This novel formulation corresponds to dense subgraph discovery in bipartite graphs.
- We present two new simple exact algorithms that simplify prior work [57]. More importantly, they allow us to treat clique and biclique densities, and potentially other induced subgraph densities of interest, in a unified way.
- Our main theoretical contribution is the notion of the *densest subgraph sparsifier*. Specifically, we propose a randomized algorithm which yields high-quality approximations for both the k-Clique-DSP and the (p,q)-Biclique-DSP while providing significant speedups and improved space complexity.
- We evaluate our method on a large collection of real-world networks. We observe a strong empirical performance of our framework.
- Using our proposed method, we study how densities evolve in time evolving graphs. As an example of the utility of our method, we compare our collection of real-world networks against stochastic Kronecker graphs [44], a popular random graph model that mimics real-world networks in certain respects. We show, perhaps surprisingly, that they differ significantly.

Before continuing, we preview our results in Table 1 for three real-world networks. A detailed description of the datasets is shown in Table 2. We run a readily available highly optimized exact maximum flow implementation of the push-relabel algorithm due to Goldberg and Tarjan [34] to measure the optimal k-clique densities ρ_k^* for $k = 2, 3$ and the corresponding running times. We also run our randomized algorithm, which combines our sampling scheme with the same maximum flow implementation. Over five runs and for each output set S, we measure its size $|S|$, its edge density $f_e(S) = \frac{e(S)}{\binom{|S|}{2}}$, the error it achieves defined as $\left(1 - \frac{\rho_k(S)}{\rho_k^*}\right) \times 100\%$, and the speedups obtained compared to the exact algorithm. Here, $\rho_k(S), \rho_k^*$ are the output and the optimal k-clique density respectively. Table 1 reports the resulting averages. We observe that our algorithm achieves high-quality approximations while speeding up the computations. The speedups increase as a function of the k-clique density. We further observe that the subgraphs obtained for $k = 3$ tend to be smaller and closer to cliques compared to the ones obtained for $k = 2$.

The qualitative results of Table 1 become further pronounced on large-scale inputs. Specifically, exact algorithms may require more memory than what is available. Also, we observe that as the value of k increases the output approaches the clique structure. Similar observations hold for the case of bipartite graphs as well.

Notation: Let $G(V, E)$ be an undirected graph, $|V| = n, |E| = m$. We define $f_k(v)$ to be the number of k-cliques that vertex v participates in, $v \in V$. Let $C_k(G)$ be the set of k-cliques in graph G and $c_k(G) = |C_k(G)|$, $k = 2, \ldots, n$. For instance, $f_2(v)$ is the degree of vertex v and $c_2(G) = m$. Notice that $\sum_{v \in V(G)} f_k(v) = k c_k(G)$. We omit the index G when it is obvious to which graph we refer to. Similarly, when G is bipartite –let $V = L \cup R$– we define $C_{p,q}(G)$ to be the set of bipartite cliques (bicliques) with p, q vertices on the left,right respectively, $c_{p,q}(G) = |C_{p,q}(G)|$ and $f_{p,q}(v)$ the number of (p, q)-bicliques that v participates in for all $v \in L \cup R$. For any $S \subseteq V$ we define $c_k(S)$ ($c_{p,q}(S)$) to be the number of k-cliques ((p, q)-bicliques) induced by S. As a ground-truth measure of how close a set $S \subseteq V$ ($L' \cup R', L' \subseteq L, R' \subseteq R$) lies to a clique (biclique) we use $f_2(S) = c_2(S)/\binom{|S|}{2}$ ($f_{1,1}(L', R') = \frac{c_{1,1}}{|L'||R'|}$). Notice that $f_2, f_{1,1} \in [0, 1]$. We slightly abuse the notation and use the usual notation f_e to denote either $f_{1,1}$ or f_2 depending on whether the graph is bipartite or not respectively.

Theoretical Preliminaries. In Section 3 we use the following Chernoff bounds [47].

Theorem 1 (Chernoff bounds). *Consider a set of mutually independent binary random variables $\{X_1, \ldots, X_t\}$. Let $X = \sum_{i=1}^{t} X_i$ be the sum of these random variables. Then we have*

$\Pr[X < (1 - \epsilon)\mu] \leq e^{-\epsilon^2 \mu/2}$ *when* $E[X] \geq \mu$ *(I)*
and
$\Pr[X > (1 + \epsilon)\mu] \leq e^{-\epsilon^2 \mu/3}$ *when* $E[X] \leq \mu$. *(II)*

2. RELATED WORK

Dense subgraph discovery is a major problem in both algorithmic graph theory and graph mining applications. It is not a surprise that numerous formulations and algorithms have been proposed for this problem [41]. These formulations can be categorized (roughly) into five categories: (a) **NP**-hard formulations, e.g., [6, 7, 13, 16, 28, 55, 56]. These formulations are also typically hard to approximate as well due to connections with the Maximum Clique problem [35]. (b) Heuristics with no theoretical guarantees, e.g., [23, 60]. The survey by Bomze et al. contains a wide collection of such heuristics [18]. (c) Brute force techniques may be applicable when the graph size is small [37]. (d) Enumeration techniques for finding maximal cliques [20, 26, 46, 59]. (e) Poly-time solvable formulations that may or not succeed in finding large-near cliques.

In the following we focus on the latter category and specifically on the most notable family of such formulations, the *k-clique densest subgraph problem* (k-Clique-DSP) [57]. The

2-clique densest subgraph problem is widely known as the *densest subgraph problem* (DSP) and has a long history since the early 80s [10, 22, 31, 33, 38, 39]. As Bahmani, Kumar and Vassilvitskii point out "the densest subgraph problem lies at the core of large scale data mining" [12]. In the densest subgraph problem (DSP) we wish to compute the set $S \subseteq V$ which maximizes the average degree [33]. A densest subgraph can be identified in polynomial time by solving a maximum flow problem [31, 33]. Charikar [22] proved that the greedy algorithm proposed by Asashiro et al. [10] produces a 2-approximation of the densest subgraph in linear time. The same algorithm is also used to find k-cores [14], maximal connected subgraphs in which all vertices have degree at least k. The directed version of the problem has also been studied [22, 39]. We notice that there is no size restriction of the output. When restrictions on the size of S are imposed, the problem becomes **NP**-hard [7, 39]. Finally, Bhattacharya et al. studied the densest subgraph problem when the graph is dynamic, i.e., it changes over time [17]. Using sampling and a potential function argument, they were able to achieve a fully-dynamic $(4+\epsilon)$-approximation algorithm which uses sublinear space and poly-logarithmic amortized time per update, where $\epsilon > 0$.

The main issue with the DSP is that many times it fails to extract large near-cliques, namely sets of vertices which are "close" to being cliques modulo "few" edges. The two latter notions will be quantified throughout this work by recording both the size of the output set S and the edge density $f_2(S)$. Notice that recording only $f_2(S)$ is meaningless as a single edge achieves the maximum possible edge density. Motivated by this issue, Tsourakakis proposed recently the the K-CLIQUE-DSP as a way to overcome this issue [57]. The K-CLIQUE-DSP generalized the DSP by maximizing the average number of induced k-cliques over all possible subsets of V, $k \geq 2$. Specifically, each $S \subseteq V$ induces a non-negative number of k-cliques $c_k(S)$. The K-CLIQUE-DSP aims to maximize the average number $\frac{c_k(S)}{|S|}$. It was shown in [57] that for $k = \Theta(1)$ we can solve the K-CLIQUE-DSP in polynomial time. Experimentally it was shown that for $k = 3$ the output solutions were approaching the desired type of large near-cliques for several real-world networks. On the other hand, the network construction proposed by [57] is not scalable to large-scale graphs, both in terms of space- and time-complexity.

An attempt was made in the same paper to improve the space complexity using submodularity optimization. Specifically, for $k = 3$ Tsourakakis proposed an algorithm that uses linear space and runs in $O\big((n^5 m^{1.4081} + n^6)\log(n)\big)$ time. This algorithm still does not scale.

3. PROPOSED METHOD

3.1 Problem Definitions

We start by introducing our objectives, formally defining the *k-clique densest subgraph problem* [57], and introducing a variant for bipartite graphs that we call the *(p,q)-clique densest subgraph problem* ((P,Q)-BICLIQUE-DSP).

DEFINITION 1 (k-clique density). *Let $G(V,E)$ be an undirected graph. For any $S \subseteq V$ we define its k-clique density $\rho_k(S)$, $k \geq 2$ as $\rho_k(S) = \frac{c_k(S)}{s}$, where $c_k(S)$ is the number of k-cliques induced by S and $s = |S|$.*

DEFINITION 2 ((p,q)-biclique density). *Let $G(L \cup R, E)$ be an undirected bipartite graph. For any $S \subseteq L \cup R$ we define its (p,q)-biclique density $\rho_{p,q}(S)$, $p,q \geq 1$ as $\rho_{p,q}(S) = \frac{c_{p,q}(S)}{s}$, where $c_{p,q}(S)$ is the number of (p,q)-bicliques induced by S and $s = |S|$.*

Now we introduce the decision problems for both the k-clique density and the (p,q)-biclique density.

PROBLEM 1 (DECISION). *We distinguish two decision problems, depending on whether the input graph G is bipartite or not.*
(a) Non-bipartite: Given a graph $G(V,E)$, does there exist $S \subseteq V$ such that $\rho_k(S) \geq D$?
(b) Bipartite: Given a bipartite graph $G(L \cup R, E)$, does there exist $L' \subseteq L, R' \subseteq R$ such that $\rho_k(L', R') \geq D$?

Similarly, we introduce the optimization problems for both types of density.

PROBLEM 2 (OPTIMIZATION). *We distinguish two optimization problems, depending on whether the input graph G is bipartite or not.*
(a) Non-bipartite: Given a non-bipartite $G(V,E)$, find a subset of vertices S^* such that $\rho_k(S^*) \geq \rho_k(U)$ for all $U \subseteq V$. Let $\rho_k^* \doteq \rho_k(S^*)$ be the optimal k-clique density. We refer to this problem as the k-clique densest subgraph problem (K-CLIQUE-DSP).
(b) Bipartite: Given a bipartite graph $G(L \cup R, E)$, find a subset of vertices $L^* \cup R^*$ such that $\rho_{p,q}(L^*, R^*) \geq \rho_k(L', R')$ for all $L' \cup R' \subseteq V$. Let $\rho_{p,q}^* \doteq \rho_k(L^*, R^*)$ be the optimal (p,q)-biclique density. We refer to this problem as the (p,q)-biclique densest subgraph problem ((P,Q)-BICLIQUE-DSP).

A basic but important observation we use in this work is the following: given an efficient algorithm for PROBLEM 1, we can solve PROBLEM 2 using $O(\log n)$ iterations for any $k = \Theta(1)$. These iterations correspond to binary searching on D for the optimal density ρ_k^*, $\rho_{p,q}^*$ [39, 33, 57].

3.2 Exact Algorithms

The main result of this Section are two exact algorithms for PROBLEM 1 which both simplify prior work [57] for the K-CLIQUE-DSP and more importantly can be easily extended to maximize biclique densities. We present our first network construction in the context of PROBLEM 1 (a), which implies an efficient algorithm for the K-CLIQUE-DSP. Then, we discuss how it is modified to solve PROBLEM 1 (b). We briefly present a second construction for completeness; our time- and space-complexity bounds are asymptotically the same for both algorithms.

Construction A. We construct a bipartite network $\mathcal{H}_D$ for a given threshold parameter D with two special vertices, the source s and the sink t. The left side of the vertices A corresponds to the vertex set V and the right side of the vertices B corresponds to the set of $(k-1)$-cliques C_{k-1} (in contrast to the set of k-cliques C_k as in [57]). We add four types of capacitated arcs.

Type I : We add an arc for each $v \in A$ from the source s to v with capacity equal to $f_k(v)$.
Type II: We add an arc of capacity 1 from each $v \in A$ to each $(k-1)$-clique $c \in B$ iff they form a k-clique.
Type III : We add an arc of infinite capacity from each $(k-1)$-clique to each its vertices.
Type IV: We add an arc from each vertex $v \in A$ to the sink t of capacity kD.

The infinite capacity arcs of type III may appear "unnatural" but they are crucial. The intuition behind them is that we wish to enforce that if a vertex in B corresponding to a $(k-1)$-clique lies on the source side of the cut then so do its component vertices in A. The next theorem proves that the above network construction can reveal the existence or lack thereof of a subgraph whose k-clique density is greater or equal than D.

THEOREM 2. *There exists a polynomial time algorithm which solves the* DECISION-K-CLIQUE-DSP *for any* $k = \Theta(1)$ *using the above network construction.*

PROOF. We first construct the network $\mathcal{H}_D$ as described above. We then compute the minimum st-cut (S,T), $s \in S, t \in T$. Both steps require polynomial time for $k = \Theta(1)$. Notice that the value of the cut is always upper bounded by kc_k since $S = \{s\}, T = V(\mathcal{H}_D) \backslash S$ is a valid st-cut. Let $A_1 = S \cap A, B_1 = S \cap B$, and $A_2 = T \cap A, B_2 = T \cap B$. We first note that the set of $(k-1)$-cliques B_1 corresponds to the set of induced $(k-1)$-cliques by A_1. This is because edges of type III only allow the inclusion of cliques with all vertices in A_1, and including such cliques can only lower the weight of the cut. We consider three type of terms that contribute the value of the min st-cut. Clearly due to arcs of type I we pay $\sum_{v \in A_2} f_k(v)$. Due to arcs of type IV we pay $kD|A_1|$ in total. Now we consider the arcs that cross the cut due to the existence of k-cliques. We define $c_k^{(j)}$ for $j = 1, \ldots, k$ to be the number of k-cliques in G that have exactly j vertices in A_1. Notice that $c_k^{(k)} = c_k(A_1)$. We observe that for each k-clique ∇ with j vertices in A_1, $j = 1, \ldots, k-1$, we pay in total j units in the min st-cut due to j arcs of type II from each v in A_1 towards the unique $(k-1)$-clique with which it forms ∇. Therefore, the value of the min st-cut is

$$val = \sum_{v \in A_2} f_k(v) + kD|A_1| + \sum_{j=1}^{k-1} j c_k^{(j)}.$$

Notice that $\sum_{j=1}^{k} j c_k^{(j)} = \sum_{v \in A_1} f_k(v) = kc_k - \sum_{v \in A_2} f_k(v)$ and therefore we can rewrite the value of the min st-cut as $val = kc_k + kD|A_1| - kc_k(A_1)$. Hence, there exists $U \subseteq V$ such that $\rho_k(U) > D$ iff there exists a minimum st-cut with value less than kc_k. Furthermore, when a cut is found, A_1 suffices as this subset of vertices. □

Time- and space-complexity analysis. To bound the time and space required of our algorithm, we first notice that the only $(k-1)$-cliques that matter are the ones that participate in a k-clique. Therefore, $|B| = O(kc_k)$ as each k-clique gives rise to k distinct $(k-1)$-cliques. Therefore, $|V(\mathcal{H}_D)| = O(n + kc_k), |E(\mathcal{H}_D)| = O(n + \sum_{v \in V(G)} f_k(v) + (k-1)kc_k) = O(n + k^2 c_k)$. This implies that the total space complexity is $O(n + k^2 c_k)$. The time complexity depends on the k-clique listing algorithm and the maximum flow routine we use. For the former we use the Chiba-Nishizeki algorithm [25]. This algorithm lists C_k in $O(km\alpha(G)^{k-2})$, where $\alpha(G)$ is the arboricity of the graph. The state-of-the-art strongly polynomial time algorithm is due to Orlin [49] and runs in $O(nm)$ time for networks with n vertices and m edges. The best known weakly polynomial algorithm is due to Lee and Sidford [42] and runs in $\tilde{O}\left(m\sqrt{n}\log^2 U\right)$ where the $\tilde{O}()$ notation hides logarithmic factors and U is the maximum capacity. While the latter is faster as a subroutine for our max-flow instance, we express the run time in terms of Orlin's algorithm to keep expressions simple. The running time for any k constant is $O(m\alpha(G)^{k-2} + (n + c_k)^2)$.

In the light of our experimental findings shown in Table 2, we observe that for small values of k, which is the range of values we are interested in this work, $c_k \gg n$ and $c_k \gg m$. Furthermore, the arboricities of real-world networks are small, which allows us to list small k-cliques efficiently [27]. Therefore, "in practice" the space complexity is $O(c_k)$ and the running time $O(c_k^2)$.

Bipartite graphs. The algorithmic scheme described above adapts to the (p,q)-*biclique densest subgraph problem* ((P,Q)-BICLIQUE-DSP) as well. We list $C_{p,q}$ the set of (p,q)-bicliques using the $O(\alpha(G)^3 4^{\alpha(G)} n)$ algorithm due to Eppstein [26]. The arcs of type I have capacities equal to $f_{p,q}(v)$. For arcs of type II, we add from each vertex v depending on whether $v \in L$ or $v \in R$ an arc of capacity 1 to each $(p-1,q)$-biclique or $(p,q-1)$-biclique with which it forms a (p,q)-biclique. Arcs of type III, IV are added in a similar way as in the K-CLIQUE-DSP. Similarly, as in our analysis above, under the realistic assumption that for small values of p, q it is true that $c_{p,q} \gg n$ and $c_{p,q} \gg m = c_{1,1}$ the space complexity is $O(c_{p,q})$ and the time complexity is $O(c_{p,q}^2)$.

Construction B. For our alternative algorithm, we again construct a bipartite network $\mathcal{H}_D$ for a given threshold parameter D with two special vertices, the source s and the sink t. The left side of the vertices A corresponds to the vertex set V whereas the right side of the vertices B corresponds to the k-clique ((p,q)-biclique) set C_k ($C_{p,q}$). Now we add three types of capacitated arcs.

Type I: We add an arc from the source s to each vertex $v \in A$ with capacity D.
Type II: We add k (or $p+q$) arcs of capacity $+\infty$ from each $v \in A$ to each $e \in B$ iff v is a vertex of e.
Type III: We add an arc from each k-clique ($(p,q)-biclique$) $e \in A$ to the sink t with capacity 1.

Using a similar type of argument as above, we obtain the following theorem (details omitted). As already mentioned, our time- and space-complexity bounds remain the same for this construction.

THEOREM 3. *Let (S,T) be the min st-cut in $\mathcal{H}_D$. If there exists a set $Q \subseteq V$ such that $\rho_k(Q) > D$ ($\rho_{p,q}(Q) > D$), then $S \backslash \{s\} \neq \emptyset$. If for all sets $Q \subseteq V$ the inequality $\rho_k(Q) \leq D$ ($\rho_{p,q}(Q) \leq D$) holds, then $S = \{s\}$.*

3.3 Densest Subgraph Sparsifiers

The cost of the exact algorithm scales with c_k, which can be much larger than the size of the graph for moderate values of k. We address this issue by sampling the graph to a smaller one. We show that sampling each k-clique or (p,q)-biclique independently with appropriate probabilities

preserves the maximum densities. In order to present our results for both density types in a unified way we consider an r-uniform hypergraph $\mathcal{H}(V_{\mathcal{H}}, E_{\mathcal{H}})$ whose vertex set $V_{\mathcal{H}}$ is the same as the vertex set V of the input graph G. We abstract the k-Clique-DSP and (p,q)-Biclique-DSP using as the set of hyperedges $E_{\mathcal{H}}$ either the set C_k of k-cliques ($r = k$) or the set $C_{p,q}$ of (p, q)-bipartite cliques ($r = p + q$) respectively.

The next theorem is the main theoretical result of this Section. Without loss of generality, we assume that $\rho_k^*, \rho_{p,q}^* = \Omega(\log n)$ as otherwise the hypergraph is already sparse. We abuse slightly the notation by using $\rho(U)$ to denote the average hyper-degree of $U \subseteq V(\mathcal{H})$, i.e., the number of hyperedges $e(U)$ induced by U divided by $|U|$.

Theorem 4. *Let $\epsilon > 0$ be an accuracy parameter. Suppose we sample each hyperedge $e \in E_{\mathcal{H}}$ independently with probability $p_D = C\frac{\log n}{D}$ where $D \geq \log n$ is the density threshold parameter and $C = \frac{6}{\epsilon^2}$ is a constant depending on ϵ. Then, the following statements hold simultaneously with high probability: (i) For all $U \subseteq V$ such that $\rho(U) \geq D$, $\tilde{\rho}(U) \geq (1-\epsilon)C \log n$ for any $\epsilon > 0$. (ii) For all $U \subseteq V$ such that $\rho(U) < (1-2\epsilon)D$, $\tilde{\rho}(U) < (1-\epsilon)C \log n$ for any $\epsilon > 0$.*

The high-level structure of our proof is analogous to that of cut-sparsifiers [15]: we bound the failure probability over each subset U, and combine them via the union bound. Let $\tilde{\rho}(U)$ be the random variable corresponding to average hyper-degree of $U \subseteq V(\mathcal{H})$. We need to prove that *for all* subsets $U \subseteq V(\mathcal{H})$ with average hyper-degree density $\rho(U) < (1-2\epsilon)D$ the inequality $\tilde{\rho}(U) < (1-\epsilon)C \log n$ holds with high probability. The Chernoff bound as stated in Theorem 1 gives a failure probability $1/poly(n)$. However, the number subsets U is 2^n, which means a straight-forward invocation requires a much lower failure probability. We remedy this by showing that the failure probability decreases exponentially as a function of the size of the set U, so it is indeed much smaller for most subsets U.

Proof. (i) Let $U \subseteq V$ such that $\rho(U) \geq D$. Let $e(U)$ be the number of induced hyperedges by U. We define X_U to be the random variable which equals the number of hyperedges induced by U after sampling each hyperedge independently with probability $p_D = C\frac{\log n}{D}$. By the linearity of expectation, we obtain that $\mathbb{E}[X_U] = pe(U) \geq C|U| \log n$. Hence, by applying the Chernoff bound (I), we obtain

$$\mathbf{Pr}\left[X_U < (1-\epsilon)C|U| \log n\right] \leq e^{-\epsilon^2 C|U| \log n/2} = n^{-3|U|}.$$

We define the following bad events. Let $\mathcal{B}$ be the event "$\exists U \subseteq V : \rho(U) \geq D, \tilde{\rho}(U) < (1-\epsilon)C \log n$" and $\mathcal{B}_x$ "$\exists U \subseteq V : |U| = x, \rho(U) \geq D, \tilde{\rho}(U) < (1-\epsilon)C \log n$" for $x \in \{2, \ldots, n\}$. By conditioning on the size of U and applying the union bound, we obtain

$$\mathbf{Pr}[\mathcal{B}] \leq \sum_{x=2}^{n} \binom{n}{x} \mathbf{Pr}[\mathcal{B}_x] \leq \sum_{x=2}^{n} \left(\frac{en}{x}\right)^x n^{-3x} = o(1).$$

(ii) This statement is proved in a similar way, namely by using Chernoff bound (II) and substituting $(1+\epsilon)$ in the above proof with $(1-\epsilon)$. Finally, it is easy to verify that both statements hold simultaneously with high probability. □

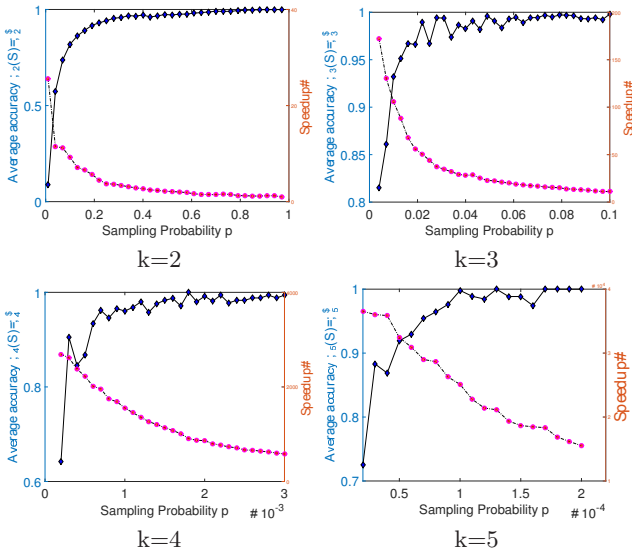

Figure 1: Accuracy $\rho_k(S)/\rho_k^*$ and speedup as functions of the sampling probability p for the CA-Astro collaboration network.

Consequences. Combining Theorem 4 with the exact algorithm from Section 3.2 gives a faster routine. If the graph contains a subgraph of density D, then Theorem 4 shows that we never get a subgraph whose density is below $1-2\epsilon$ for any $\epsilon > 0$, which implies an $(1-2\epsilon)$-approximation algorithm. In expectation the speedup -assuming Orlin's algorithm as the max-flow subroutine and the realistic assumptions in the previous section- is $O(\frac{1}{p_D^2})$. The space reduction is $O(\frac{1}{p_D})$.

Sampling with negligible overhead. In our implementation we generate our sample "on the fly". Specifically, while we list the set of cliques or bicliques we decide whether we keep each one independently. We can avoid tossing a coin for each k-clique or (p, q)-biclique using a method from [40], which we describe here for completeness. The number of unselected samples F between any two samples follows a geometric distribution, $\mathbf{Pr}[F = f] = (1-p)^{f-1}p$. A value F from this distribution can be generated by generating a random variable $U \sim [0, 1]$ and setting $F \leftarrow \lceil \frac{\ln U}{1-p} \rceil$. This allows us to generate the indices of the samples directly, and reduce the overhead to sublinear.

4. EXPERIMENTAL RESULTS

4.1 Experimental Setup

The experiments were performed on a single machine, with an Intel Xeon CPU at 2.83 GHz, 6144KB cache size, and 50GB of main memory. We find densest subgraphs on the samples using binary search and maximum flow computations as described in [57]. The flow computations were done using HIPR-3.7 [1], a C++ implementation of the push-relabel algorithm [34]. It is worth mentioning that the HIPR maximum flow package is a high-performance implementation of the Goldberg-Tarjan algorithm [34]. Previous work [24] as well as our experiments strongly suggest that HIPR in practice is very efficient.

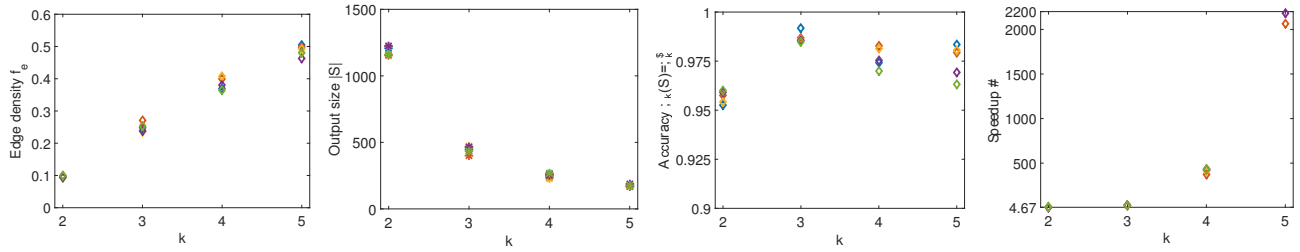

Figure 2: Figure plots (a) the edge density f_e (b) the output size $|S|$, (c) the accuracy $\rho_k(S)/\rho_k^*$ and (d) the time speedup versus $k = 2, 3, 4, 5$ for five instances of our randomized algorithm on the Epinions graph. It is worth mentioning that the observed accuracy 95.2% for $k = 2$ is the smallest among all datasets and k values we experimented with. Notice how the output set gets denser and smaller as the k value grows.

In order to compare performance in a consistent fashion, when we give performance results on the the density of vertex subsets obtained by the algorithm on sampled graphs, we measure the density of this vertex subset in the original graph.

While parametric maximum flow routines lead to more streamlined and likely more efficient algorithms, the interaction between sampling and single commodity flow routines was more direct. Furthermore, the high performance of this optimized routine was sufficient for our experiments. We anticipate generalizations based on extending to multicore settings and incorporating parametric flow routines will lead to further improvements. However, it is worth emphasizing that the key step towards making our density optimization framework scalable is the reduction of the network size on which maximum flows are computed.

We use publicly available datasets from SNAP [2] and [3]. Additionally, we use two bipartite IMDB author-to-movie graphs from US IMDB-B and Germany IMDB-G-B respectively. We use both bipartite and non-bipartite graphs, as shown in Table 2. All graphs were made simple by removing any self-loops and multiple edges; for each graph we record the number of vertices n and the number of edges m. We record for non-bipartite graphs the counts of k-cliques for $k = 3, 4, 5$ as well as the required time to list them using C++ code which is publicly available [4]. We notice that the our run times are better than existing scalable MapReduce implementations [29]. For bipartite graphs we record $c_{2,2}, c_{3,3}$ and the respective run times using our own C++ implementation. We note that both the exact and the randomized algorithm require the subgraph listing procedures.

We measure for each graph the run times of the maximum flow computation with and without sampling on construction B from Section 3.2. Their ratio yields the speedup for our randomized algorithm. We note that running the maximum flow subroutine can become a significant challenge. This is because despite the relatively small size of the input graph, the number of k-cliques c_k is typically very large. A typical file listing the set of all k-cliques for $k = 3, 4, 5$ may span several gigabytes, even if the input graph is on the order of kilo- or mega-bytes. This creates a two-fold challenge for the exact algorithm, both in terms of its space usage and its run time. For each approximate solution $S \subseteq V$ we measure the accuracy $\frac{\rho_k(S)}{\rho_k^*}$. When the exact algorithm cannot run due to lack of available memory, we report the density and the run time for the output set S, rather than the accuracy and the speedup.

In Section 4.4 we study time-evolving networks. We use the Patents citation graph and the Autonomous Systems (AS) datasets. The former spans 37 years, specifically from January 1, 1963 to December 30, 1999, The latter contains 733 daily instances which span an interval of 785 days from November 8 1997 to January 2 2000. Finally, we generate synthetic graphs using the stochastic Kronecker model [44] using as our seed the 2×2 matrix [0.9 0.5; 0.5 0.2]. Finally, the code is publicly available [5].

4.2 Effect of Sampling

As can be seen in Table 3, for the standard (non-bipartite) graphs, our algorithm's performance is quite strong. Speedups are over a factor of 10 for most of our examples when $k = 3$, and increase further for larger k. Similarly, the error is small for $k = 2$ and 3, and continues to decrease for larger k. Further, performance is generally better on larger graphs. Overall these results are consistent with our theoretical analysis, and demonstrate the scalability that can be achieved via sparsification.

Figure 1 shows the effect of different sampling probabilities on the accuracy and speedup of our algorithm. For the CA-Astro graph and $k = 2, 3, 4, 5$ we plot the average accuracy and speedup obtained by sampling with different probabilities p. The x-axis shows the smallest possible range of p around which we obtain concentration. For each value of p, we run our randomized algorithm five times and report averages. The results are well concentrated around the respective averages. We observe that this range decreases quickly as k increases. We also notice the significant speedup achieved while enjoying high accuracies: for $k \geq 3$, we obtain at least 50-fold speedups with an accuracy more than 95%, and we even see a 25000 fold speedup for $k = 5$.

We observed some additional interesting findings worth reporting. For $k = 3$ in the CA-Astro graph we found that while the densities are well-concentrated around the mean (standard deviation equals 1.96), the output solutions can look different. For instance we find that the output for four out of five runs consists of around 140 vertices with $f_e = 0.4$, while for one the output is a subgraph on 76 vertices with $f_e = 0.8$. Similarly, when $k = 4$ we obtain for four out of five runs the same output, namely a subgraph on 62 vertices with $f_e = 0.96$. For one run we obtain a subgraph on 139 vertices with $f_e = 0.43$. In addition to the beneficial effects of our randomized algorithm that we have already

Name	n	m	c_3	T	c_4	T	c_5	T	$c_{2,2}$	T	$c_{3,3}$	T
■ Web-Google	875 713	3 852 985	11 385 529	8.5	32 473 410	16.5	81 928 127	36.4	-	-	-	-
⋆ Epinions	75 877	405 739	1 624 481	1.6	5 803 397	4.8	17 417 432	13.4	-	-	-	-
⊙ CA-Astro	18 772	198 050	1 351 441	0.6	9 580 415	3.94	64 997 961	27.2	-	-	-	-
■ Pol-blogs	1 222	16 714	101 043	0.05	422 327	0.2	1 377 655	0.7	-	-	-	-
⊙ Email-all	234 352	383 111	383 406	0.4	1 057 470	0.9	2 672 050	1.9	-	-	-	-
■ LastFm-B	17 644	92 366	-	-	-	-	-	-	18 266 703	27.8	-	-
⋆ IMDB-B	241 360	530 494	-	-	-	-	-	-	691 594	3.6	261 330	3.3
⋆ IMDB-G-B	21 258	42 197	-	-	-	-	-	-	14 919	0.1	2 288	0.1
⊙ Bookmarks-B	71 090	437 593	-	-	-	-	-	-	431 996	0.82	14 901	0.53

Table 2: Datasets used in our experiments. The number of vertices n and edges m is recorded for each graph. For each non-biparite graph we show the counts c_k and the respective runtimes to list all the k-cliques, $k = 3, 4, 5$. For each bipartite graph (name-B) we show the counts $c_{k,k}$ and the respective runtimes for $k = 2, 3$. Respective run times are reported in seconds (T).

G	$k=2$				$k=3$				$k=4$				$k=5$			
	No Sampl.		Sampl.		No Sampl.		Sampl.		No Sampl.		Sampl.		No Sampl.		Sampl.	
	T	ρ_2^*	sp	err	T	ρ_3^*	sp	err	T	ρ_4^*	sp	err	T	ρ_5^*	sp.	err.
■	33.9	26.8	1.85	0.3	132.8	394.0	15.97	0.5	460.3	4 136.6	111.3	0.3	-	-	-	-
⋆	2.37	60.2	4.67	4.3	15.76	860.0	28.67	1.3	85.7	6 920.1	414.7	0.2	37 939.6	410.3	2 190	2.5
⊙	1.19	32.1	2.25	1.5	12.34	546.9	24.35	0.6	155.1	7 351.8	610.1	0.6	2 107.2	77 288.0	14 604	0
■	0.05	27.9	2.60	4.1	0.64	328.8	42.79	3.3	4.39	208 497	345.7	2.3	24.04	10 352.4	2 201	1.3
⊙	2.49	36.3	1.87	1.6	3.82	359.1	6.7	2.6	13.5	2 268.5	27.9	1.7	52.4	9 432.9	108.4	0.8

Table 3: Results for non-bipartite graphs. For each $k = 2, 3, 4, 5$ we report (i) for the exact algorithm the run time T (seconds) and the optimal ρ_k^* density, (ii) for our randomized algorithm the speedup (sp) and the error $\left(1 - \frac{\rho_k(S)}{\rho_k^*}\right) \times 100\%$ (err). For $k = 5$ the exact algorithm cannot run on the Web-Google graph. Our randomized algorithm achieves an average 5-clique density of 32 640 and an average runtime of 2.86 seconds.

G	$(p,q) = (1,1)$				$(p,q) = (2,2)$				$(p,q) = (3,3)$			
	No Sp.		Sp.		No Sp.		Sp.		No Sp.		Sp.	
	T	$\rho_{1,1}^*$	sp	err	T	$\rho_{2,2}^*$	sp	err	T	$\rho_{3,3}^*$	sp	err
■	0.49	31.8	2.74	2.6	536.8	17 217	3 404	1.3	-	-	-	-
⋆	4.82	6.49	1.45	0.3	11.24	171.5	8.33	1.7	5.20	494.7	8.93	0.9
⋆	0.26	3.38	1.24	9	0.13	29.2	3.51	2.8	0.03	28.9	1.39	0
⊙	0.71	4.16	1.49	1.9	4.34	50.3	3.23	1.3	0.28	180.8	3.04	0

Table 4: Results for bipartite graphs reported in a similar way as in Table 3.

discussed in Section 3.3, an additional effect may be the ability to sample from the set of subgraphs with near-optimal density. Understanding how our algorithm might be useful in obtaining diverse sets such subgraphs is an interesting future research direction.

Table 4 shows our results for bipartite graphs. We witness again high quality approximations and speedups which become larger as the respective count grows. Notice that for the LastFm graph there are no results for $p = q = 3$. This is because the listing algorithm takes a long time to list $K_{3,3}$s. This is the only instance for which we faced this issue. A possible way to tackle such issues is to first sample the original graph [50]. This is an interesting research extension.

4.3 Finding large-near (bi)cliques

In the previous section we saw that our randomized algorithm enjoys at the same time significant speedups and high accuracy. Furthermore, we observed that the speedups are an increasing function of k. In previous works, it was shown that moving from $k = 2$ to $k = 3$ yields significant benefits [57]. The ability to approximate this density for even higher values of k allows us to study the effect of increasing k in more detail. In Table 5, we list the size and edge-density of the set returned when we increase k to 5. Specifically, we observe that for all datasets the edge density f_e and the size of the output set S follow a different trend: the former increases and approaches 1 whereas the latter decreases. Table 6 shows the same output properties for the bipartite graphs. We observe a similar trend here as well, with the exception of the Bookmarks bipartite graph

G	$k=2$		$k=3$		$k=4$		$k=5$	
	f_e	|S|	f_e	|S|	f_e	|S|	f_e	|S|
■	0.46	117	0.83	63	0.89	58	0.93	53
⋆	0.12	1 012	0.26	432	0.40	235	0.50	172
⊙	0.11	18 686	0.80	76	0.96	62	0.96	62
■	0.19	16 714	0.54	102	0.59	92	0.63	84
⊙	0.13	553	0.38	167	0.48	122	0.53	104

Table 5: Optimal solutions obtained for varying values of k-clique density $k = 2, 3, 4, 5$ for non-bipartite graphs. As k increases, we consistently observe higher edge densities and smaller sizes in the optimum subgraphs.

G	$(p,q)=(1,1)$		$(p,q)=(2,2)$		$(p,q)=(3,3)$	
	f_e	|S|	f_e	|S|	f_e	|S|
■	0.05	1 256	0.12	493	-	-
⋆	0.001	9 177	0.06	181	0.30	40
⋆	0.001	6 437	0.41	18	0.43	17
⊙	0.41	20	0.41	20	0.41	20

Table 6: Optimal solutions obtained for varying values of (p, q)-biclique density $p = q = 1, 2, 3$ for bipartite graphs. As the order of the (p, q)-biclique increases, we observe higher edge densities and smaller sizes in the optimum subgraphs.

where increasing the order of the biclique does not affect the output.

Finally, Figure 2 shows for the Epinions graph the edge densities f_e, the output sizes $|S|$, the accuracies and the speedups over five runs for $k = 2, 3, 4, 5$. These data shows that sampling provides accurate, concentrated estimates. Furthermore, it highlights the significant speedups obtained from sampling and the fact that the output gets closer to a clique as k grows.

4.4 Density in Time Evolving Graphs

As Leskovec, Kleinberg and Faloutsos showed in their influential paper [45], the average degree grows as a function of time in many real-world networks. But what about the optimal 2-clique and 3-clique densities? In the following we study how ρ_2^*, ρ_3^* evolve over time for the Patents and Autonomous Systems (AS) datasets.

We also generate stochastic Kronecker graphs on 2^i vertices for $i = 8$ up to $i = 21$. We assume that the first snapshot index in plots Figure 3(e),(f) corresponds to $i = 8$, i.e., the number of vertices increases over time. For each index we generate 10 instances and report averages. We used a core-periphery type seed matrix as discussed in Section 4.1.

Patents. We observe in Figure 3(a) that both ρ_2^* and ρ_3^* exhibit an increasing trend. This increasing trend becomes is mild for ρ_3^* up to 1995, but then it takes off. Looking in Figure 3(b) makes this finding even more interesting as the number of edges grows faster than the number of triangles. This suggests that the number of triangles are localized in some region of the graph. Indeed, by careful inspection we find why this is happening. We are seeing an outlier - the company Allergan, Inc. This company tends to cite all their previous patents with each new patent and creates a dense subregion in the graph. After removing Allergan from the graph, ρ_3^* grows in a more regular fashion.

Autonomous Systems (AS) . Again, we observe in Figure 3(c) that both ρ_2^* and ρ_3^* exhibit an increasing trend, although it is slight to negligible for ρ_3^*. This stability is despite the significant increase in the number of triangles over time (see Figure 3(d)). After careful inspection, we find that the 3-densest subset consists consistently of around 35 vertices, starting from 35 vertices on 11 Aug 1997 and ending with 34 vertices on 2 Jan 2000, with minor fluctuations in between. We do not currently have a complete explanation for the observed sudden dropdowns, but we believe they are probably due to some failure in the system which reports the network.

Stochastic Kronecker graphs. In these synthetic graphs we observe that while ρ_2^* grows, when examining ρ_3^* we find a triangle densest subgraph contains on average about 2 triangles per vertex. The latter contrasts quantitatively what we observe in both real datasets, although we do note that for the Patents dataset, ρ_2^* also grows. Understanding this gap in behaviors is left as an open question.

5. CONCLUSION

Summary. Finding dense subgraphs in terms of their k-clique density is a key primitive for network graph analysis. Our primary contribution is showing that sampling provides an effective methodology for finding subgraphs with approximately maximal k-clique density. As with many other problems, sampling leads to substantial computational savings, allowing us to perform computations on larger graphs, as well as for higher values of k. We also developed two efficient exact algorithms via different reductions to maximum flow, and defined and examined bipartite variations of the problem.

Open Problems. Our work leaves several interesting open questions. One issue is that often we would like to find not just a single dense subgraph, but a collection of them; generally, we might want this collection to be *diverse*, in that

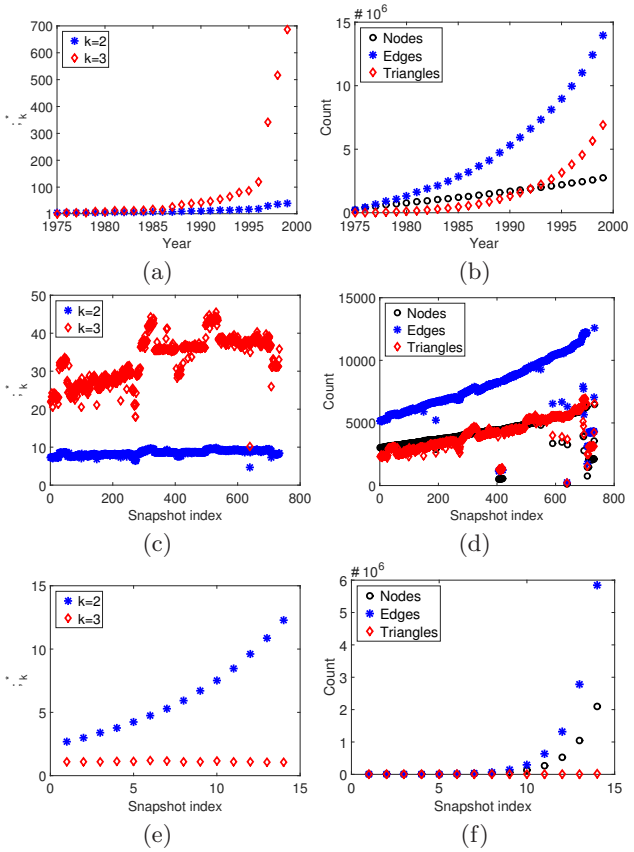

Figure 3: Optimal k-clique densities ρ_k^* for $k = 2, 3$ and number of nodes, edges and triangles versus time on the (i) Patents dataset (a),(b) (ii) Autonomous systems dataset (c),(d) and (iii) stochastic Kronecker graphs (e),(f).

they are entirely or mostly disjoint [13]. Our sampling approach appears useful for finding diverse collections of dense subgraphs; can we formalize this somehow?

We have so far only investigated the performance of sampling with exact densest subgraph algorithms. In principle, it can also be used to speed up heuristic approaches, such as the *peeling* algorithm [17, 22]. We plan to study empirically their interaction in future work. Our analysis of this sampling framework is also conducted for general graphs; it can likely be improved for specific families of graphs.

Finally, with our new techniques we have initiated a study of how k-clique densest subgraphs evolve over time, and observed differences between real-world data and a well-known stochastic model. We plan to test larger collections of real-word data and stochastic models to understand better both the behavior of the evolution of real-world dense subgraphs over time and possible gaps in what the stochastic models are capturing.

Acknowledgments

Michael Mitzenmacher was supported in part by NSF grants IIS-0964473, CNS-1228598, and CCF-1320231. Jakub Pachocki and Shen Chen Xu were supported in part by NSF grant CCF-1065106.

6. REFERENCES

[1] http://www.avglab.com/soft/hipr.tar.
[2] http://snap.stanford.edu/data/index.html.
[3] http://grouplens.org/datasets.
[4] http://research.nii.ac.jp/~uno/codes.htm.
[5] Large Near-Clique Detection. http://tinyurl.com/o6y33g9.
[6] J. Abello, M. G. C. Resende, and S. Sudarsky. Massive quasi-clique detection. In *LATIN*, 2002.
[7] R. Andersen and K. Chellapilla. Finding dense subgraphs with size bounds. In *WAW*, 2009.
[8] A. Andoni, A. Gupta, and R. Krauthgamer. Towards $(1+\varepsilon)$-approximate flow sparsifiers. In *SODA*, pages 279–293. SIAM, 2014.
[9] A. Angel, N. Sarkas, N. Koudas, and D. Srivastava. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. In *VLDB*, 5(6), pages 574–585, Feb. 2012.
[10] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. In *Journal of Algorithms*, 34(2), 2000.
[11] G. D. Bader and C. W. Hogue. An automated method for finding molecular complexes in large protein interaction networks. In *BMC bioinformatics*, 2003.
[12] B. Bahmani, R. Kumar, and S. Vassilvitskii. Densest subgraph in streaming and mapreduce. In *VLDB* , 5(5):454–465, 2012.
[13] O. D. Balalau, F. Bonchi, T. Chan, F. Gullo, and M. Sozio. Finding subgraphs with maximum total density and limited overlap. In *WSDM*, pages 379–388. ACM, 2015.
[14] V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. In *Arxiv*, arXiv.cs/0310049, 2003.
[15] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *STOC*, pages 47–55, 1996.
[16] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: an $o(n^{1/4})$ approximation for densest k-subgraph. In *STOC*, pages 201–210, 2010.
[17] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis. Space-and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *STOC*, 2015 (to appear).
[18] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74, 1999.
[19] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich. Core decomposition of uncertain graphs. In *KDD*, pages 1316–1325, 2014.
[20] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. In *Communications of the ACM*, 16(9):575–577, 1973.
[21] G. Buehrer and K. Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *WSDM*, pages 95–106, 2008.
[22] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, pages 84–95, 2000.

[23] J. Chen, Y. Saad. Dense Subgraph Extraction with Application to Community Detection. In *TKDE*, vol. 24, pages 1216–1230, 2012.

[24] B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. In *Algorithmica*, 19(4), pages 390–410, 1997.

[25] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. In *SIAM Journal on Computing*, 14(1), pages 210–223, 1985.

[26] D. Eppstein. Arboricity and bipartite subgraph listing algorithms. In *Information Processing Letters*, 51(4), pages 207–211, 1994.

[27] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *ISAAC*, 2010.

[28] U. Feige, G. Kortsarz, and D. Peleg. The dense k-subgraph problem. In *Algorithmica*, 29(3), 2001.

[29] I. Finocchi, M. Finocchi, and E. G. Fusco. Counting small cliques in mapreduce. In *ArXiv* arXiv:1403.0734, 2014.

[30] E. Fratkin, B. T. Naughton, D. L. Brutlag, and S. Batzoglou. Motifcut: regulatory motifs finding with maximum density subgraphs. In *Bioinformatics*, vol. 22(14), pages 150–157, 2006.

[31] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. In *Journal of Computing*, 18(1), 1989.

[32] D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *VLDB*, pages 721–732, 2005.

[33] A. V. Goldberg. Finding a maximum density subgraph. Tech. report, UC Berkeley, 1984.

[34] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. In *Journal of the ACM (JACM)*, 35(4), pages 921–940, 1988.

[35] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Acta Mathematica*, 182(1), 1999.

[36] R. Jin, Y. Xiang, N. Ruan, and D. Fuhry. 3-hop: a high-compression indexing scheme for reachability query. In *SIGMOD*, 2009.

[37] D. S. Johnson and M. A. Trick. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge* American Mathematical Soc., 1996.

[38] R. Kannan and V. Vinay. Analyzing the structure of large graphs, *manuscript*, 1999.

[39] S. Khuller and B. Saha. On finding dense subgraphs. In *ICALP*, 2009.

[40] D. E. Knuth. Seminumerical algorithms. 2007.

[41] V. E. Lee, N. Ruan, R. Jin, and C. C. Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, pages 303–336, Springer, 2010.

[42] Y. T. Lee and A. Sidford. Path finding methods for linear programming: Solving linear programs in õ (vrank) iterations and faster algorithms for maximum flow. In *FOCS*, pages 424–433, 2014.

[43] F. T. Leighton and A. Moitra. Extensions and limits to vertex sparsification. In *STOC*, pages 47–56, 2010.

[44] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. In *The Journal of Machine Learning Research*, vol. 11, pages 985–1042, 2010.

[45] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD* , pages 177–187, 2005.

[46] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *SWAT* pages 260–272, 2004.

[47] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[48] A. Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *FOCS*, pages 3–12, 2009.

[49] J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. In *Mathematical Programming*, vol. 118(2), pages 237–251, 2009.

[50] R. Pagh and C. E. Tsourakakis. Colorful triangle counting and a mapreduce implementation. In *Information Processing Letters*, vol. 112(7), pages 277–281, 2012.

[51] A. E. Sariyuce, C. Seshadhri, A. Pinar, and U. V. Catalyurek. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *WWW*, 2015.

[52] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, pages 81–90, 2004.

[53] M. Thorup and U. Zwick. Approximate distance oracles. In *Journal of the ACM (JACM)*, vol. 52(1), pages 1–24, 2005.

[54] M. Thorup and U. Zwick. Spanners and emulators with sublinear distance errors. In *SODA*, pages 802–809, 2006.

[55] C.E. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *KDD*, pages 104–112, 2013.

[56] C. E. Tsourakakis. *Mathematical and Algorithmic Analysis of Network and Biological Data*. PhD thesis, Carnegie Mellon University, 2013.

[57] C. E. Tsourakakis. The k-clique densest subgraph problem. In *WWW*, pages 1122–1132, 2015.

[58] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller. Triangle sparsifiers. In *J. Graph Algorithms Appl.*, 15(6), pages 703–726, 2011.

[59] T. Uno. An efficient algorithm for solving pseudo clique enumeration problem. In *Algorithmica*, 56(1), 2010.

[60] N. Wang, J. Zhang, K.-L. Tan, and A. K. Tung. On triangulation-based dense neighborhood graph discovery. In *VLDB*, 4(2), pages 58–68, 2010.