Finding Planted Partitions in Nearly Linear Time using Arrested Spectral Clustering

Nader H. Bshouty BSHOUTY@CS.TECHNION.AC.IL
Department of Computer Science, Technion, 32000 Haifa, Israel

Philip M. Long PLONG@GOOGLE.COM
Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043 USA

Abstract

We describe an algorithm for clustering using a similarity graph. The algorithm (a) runs in $O(n \log^3 n + m \log n)$ time on graphs with n vertices and m edges, and (b) with high probability, finds all "large enough" clusters in a random graph generated according to the planted partition model. We provide lower bounds that imply that our "large enough" constraint cannot be improved much, even using a computationally unbounded algorithm. We describe some experiments running the algorithm and a few related algorithms on random graphs with partitions generated using a Chinese Restaurant Processes, and some results of applying the algorithm to cluster DBLP titles.

1. Introduction

This work is aimed at using theoretical analysis to guide the design of large-scale clustering algorithms. We are especially interested in problems in which the purpose of clustering is to identify near-duplicates, for example to eliminate them from the index of a search engine.

For problems like these, it is possible to use a technique such as minhashing (Cohen, 1997; Broder, 1997) to build a graph in which most pairs of items that should be clustered together are represented by edges, and there are not too many spurious edges. There can be billions, or more, of vertices. There are many clusters, and many vertices are not clustered. The largest clusters are the most important, since they do the most harm. Because the input is large, and such duplicate-detection must be done often, efficient algorithms are important. Even if massive parallelism is available to speed up algorithms, an algorithm that expends limited resources overall is still valuable as a means to reduce costs and make limited computing resources available for other purposes.

We analyze algorithms using the *planted partition* model (Condon & Karp, 2001; McSherry, 2001) in which an adversary partitions the vertices of a graph into clusters $C_1, ..., C_t$, and then edges are included between members of the same cluster independently with one probability p, and edges are included between members of different clusters with another, smaller, probability q. We show that, if $p \geq 1/2$, a $O(n \log^3 n + m \log n)$ time algorithm can, with probability $1 - 1/\text{poly}(n)$, output a partition $U_1, U_2, ...U_s$ that, for each input cell C_i of size $\Omega(\max\{qn, \log n\})$, contains a cell U_j such that $U_j = C_i$. We also provide lower bounds that imply that this bound on the size of C_i cannot be improved much, even with unlimited computation.

Condon and Karp (2001) analyzed a nearly linear-time algorithm for recovering a planted partition in the case that the cells of the partition have the same size, and this size, and the number of cells, is known to the algorithm a priori. Their algorithm applies local perturbation to partition the vertices into two groups, and then applies the algorithm recursively to each of the two groups. They used the knowledge of the size of the cells to determine whether to make another recursive call or not.

Recent research has provided approximation guarantees for polynomial-time clustering algorithms using weaker assumptions than the planted partition model that we study here (Bansal et al., 2004; Kannan et al., 2004; Balcan et al., 2008), but using algorithms that require substantially more than linear time.

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

McSherry (2001) analyzed a spectral clustering algorithm using a model much like the embedded partition model studied here. The aim of this work was to obtain similar guarantees with a faster algorithm. A number of authors have proposed to approximate the spectrum of similarity matrices for applications like this by different sampling schemes (Williams & Seeger, 2000; Frieze et al., 2004; Drineas & Mahoney, 2005; Kumar et al., 2009). Achlioptas and McSherry (2007) provided a theoretical analysis of the approximation capabilities of one such scheme. However the approximation is performed, however, when there are k clusters, the resulting spectral clustering algorithms appear to take $\Omega(k^2)$ time. In contrast, the Subsquare algorithm takes time nearly linear in the number of edges, no matter how many clusters there are, which appears to make it more suitable for applications like near-duplicate detection. This is supported by some experiments – we tried out an algorithm obtained by using a sampling scheme like the one analyzed by Achlioptas and McSherry in concert with the spectral clustering algorithm proposed by Ng, Jordan and Weiss (2001).

Bansal, et al (2004), in Section 6 of their paper, described an algorithm based on neighborhood overlap, together with a sketch of an analysis in the case in which edges between members of the same cluster are included with probability 1. Gibson, et al (2005) described a somewhat complicated algorithm that starts by applying minhashing to vertex neighborhoods in an effort to quickly find vertices with similar neighborhoods.

The Subsquare algorithm proposed in this paper can be viewed as an approximation to more computational intensive spectral algorithms. Roughly, the spectral algorithms like to cluster together a pair (v, w) of vertices if a random walk from v is likely to land at w (von Luxburg, 2007). We show that, to obtain guarantees for the embedded partition model, it is sufficient to examine walks of length 2, or, in other words, to consider the extent to which neighborhoods of pairs of vertices overlap. However, simply squaring the adjacency matrix, even in combination with random sparsification as in (Achlioptas & McSherry, 2007), appears not to work – if we sparsify sufficiently to achieve nearly linear time, then the neighborhoods of vertices in large clusters will not overlap enough to be detected. When deciding whether to cluster together two vertices, instead of considering the overlap between random subsamples of *both* of their neighborhoods, Subsquare checks how many members of a subsample of *one* vertex's neighbors are in the list of *all* of the other vertex's neighbors.

2. The Problem and Results

2.1. Planted partition model

We will analyze algorithms using a slight variant of the *planted partition model* (Jerrum & Sorkin, 1998) (see also (Condon & Karp, 2001; Bui et al., 1987; McSherry, 2001)).

Let $G = G(V, E)$ be a random graph with the set of vertices $V = \{v_1, \ldots, v_n\}$ and set of edges E that is built as follows: (1) Choose an integer t and arbitrary disjoint subsets $C_1, \ldots, C_t \subset V$ where $C_1 \cup C_2 \cup \cdots \cup C_t = V$; (2) For every $i \geq 1$ and every $v, w \in C_i$ add the edge (v, w) to E with probability p; (3) For every $1 \leq i < j \leq t$, $v \in C_i$ and $w \in C_j$ add the edge (v, w) to E with probability q.

The choices of whether to add the edges are mutually independent. We call a random graph generated this way a (p, q)-*clustered* graph. The sets $C_1, \ldots, C_t$ are called *clusters*. We can think of clusters of size 1 as unclustered vertices.

For any vertex v, let $C(v)$ be the cluster containing v, let $N(G, v)$ be the neighbors of v in G. We also will assume that $p \geq 1/2$ (roughly, that a majority of good edges are present). Let m be the number of edges in the input graph, and n be the number of vertices.

2.2. The Algorithm

The algorithm Subsquare makes its clustering decisions based on sampling estimates of some edges of the square of the adjacency matrix of G. The algorithm has three parameters, c_0, c_1 and δ, which will be determined in the analysis. Details are as follows:

2.3. Main results

Theorem 1 *There are constants c, δ_0, m_0 and n_0 such that, for all $p \in [1/2, 1], q \in [0, 1]$, $m \geq m_0$, $n \geq n_0$, and $0 < \delta \leq \delta_0$, if G is a (p, q) clustered graph with clusters $C_1, ..., C_t$, then, with probability $1 - \delta$, Algorithm* Subsquare*, when run with $c_0 = c/2$ and $c_1 = c/2^5$, outputs a partition $U_1, ..., U_s$ of the vertices of G such that, for all cluster indices i such that*

$$|C_i| \geq c \max\{qn, \log(n/\delta)\} \tag{1}$$

there is a output cluster index j such that $C_i = U_j$.

We will show that a constraint like (1) is necessary.

Note that if all C_i satisfy (1), then Subsquare is guaranteed to output the correct clustering with high probability.

- Choose a random bijection $\pi : V \to \{1, ..., n\}$ to order the vertices.
- Make two passes over V in the order given by π, performing the following steps for each vertex v:
  1. If v has fewer than $c_0 \log(n/\delta)$ neighbors, assign v to its own cluster and go to the next v.
  2. Let R_{temp} consist of the neighbors of v that have already been assigned to clusters.
  3. Form R by independently including each element of R_{temp} with probability $\min\left\{\frac{c_0 \log(n/\delta)}{|R_{temp}|}, 1\right\}$.
  4. Choose S by independently including each member of $N(G, v)$ with probability $\frac{c_0 \log(n/\delta)}{|N(G,v)|}$.
  5. Initialize the set $\mathcal{D}$ of candidate clusters for v to $\emptyset$.
  6. For each $w \in R$, if
     (a) $|S \cap N(G, w)| \geq c_1 \log(n/\delta)$,
     (b) $|N(G, w)| \geq c_0 \log(n/\delta)$, and S_w obtained by sampling each $u \in N(G, w)$ with probability $\frac{c_0 \log(n/\delta)}{|N(G,w)|}$ satisfies $|S_w \cap N(G, v)| \geq c_1 \log(n/\delta)$,
     add $\hat{C}(w)$ (i.e., w's cluster) to $\mathcal{D}$.
  7. If $\mathcal{D}$ is not empty, set $\hat{C}(v)$ to $\hat{C}(\text{argmin}_{w' \in \cup_{C \in \mathcal{D}}} \pi(w'))$ and otherwise, start a new cluster consisting only of v.

Figure 1. Algorithm Subsquare.

Theorem 2 *The expected running time of Subsquare is* $O(n(\log^2(n/\delta))(\log n) + m \log n)$.

3. The analysis

Our proof of Theorem 1 is broken up into a series of lemmas.

Throughout this analysis, we will follow the convention of using "w.h.p." as a shorthand for "with probability $1 - \delta/\text{poly}(n)$". As we will see, polynomially many events will be bounded with this probability, so that, with probability at least $1 - \delta$, all of them hold. We will show that, for any polynomial in this confidence bound, some values of the constants in our analysis will work.

Definition 3 *Refer to the comparisons performed for a particular value of w in Step 6 of Subsquare as an* edge test. *Say that the test* succeeds *if $C(v) = C(w)$ and v is put into $\mathcal{D}$, or if $C(v) \neq C(w)$ and v is not put into $\mathcal{D}$.*

Note that, to put $\hat{C}(w)$ into $\mathcal{D}$, Subsquare requires both that $|S \cap N(G, w)| \geq c_1 \log(n/\delta)$ and that $|S_w \cap N(G, v)| \geq c_1 \log(n/\delta)$.

Lemma 4 *W.h.p., for any vertex v whose cluster $C(v)$ satisfies (1), at least half of v's neighbors are in $C(v)$.*

Proof: Since edges between vertices from distinct clusters are included with probability q, we have $E(|N(G, v) - C(v)|) \leq qn \leq |C(v)|/16$, by (1), as long as $c \geq 16$. But, since $|C(v)| \geq c \log(n/\delta)$, the standard Chernoff bound (see Theorem 4.3 of (Motwani & Raghavan, 1995)) implies that, w.h.p., $|N(G, v) - C(v)| \leq |C(v)|/8$. On the other hand, since $p \geq 1/2$, we have $E(|N(G, v) \cap C(v)|) \geq (|C(v)| - 1)/2$, and, since $|C(v)| \geq c \log n$, this implies that, w.h.p., $|N(G, v) \cap C(v)| \geq (|C(v)| - 1)/4 \geq |C(v)|/8$. ■

Lemma 5 *For a pair $\{v, w\}$ of vertices, if $C(v) = C(w)$, and the cluster satisfies (1), and an edge test is conducted for v and w, then, w.h.p., this edge test succeeds.*

Proof: First, we have that, w.h.p.,

$$|S \cap C(v)| \geq (c_0/4) \log(n/\delta) \tag{2}$$

since Lemma 4 implies that w.h.p., $|N(G, v) \cap C(v)| \geq |N(G, v)|/2$, which in turn implies $E(|S \cap C(v)|) \geq (\frac{c_0 \log(n/\delta)}{|N(G,v)|})(|N(G, v)|/2)$, from which (2) follows by a Chernoff bound.

Now, conditioned on a fixed value of S, which depends only on the random neighbors of v and the randomization of the algorithm, the events that the members of $S \cap C(v)$ are neighbors of w are independent. Given (2), we have

$$E(|S \cap N(G, w)|) \geq E(|S \cap C(v) \cap N(G, w)|) \geq |S \cap C(v)|/2$$

since $p \geq 1/2$. Thus, a Chernoff bound implies that, w.h.p. $|S \cap N(G, w)| \geq c_1 \log(n/\delta)$.

Arguing analogously, we get that, w.h.p. $|S_w \cap N(G, v)| \geq c_1 \log(n/\delta)$, and therefore the lemma holds. ■

Lemma 6 *For any pair $\{v, w\}$ of vertices, if $C(v)$ satisfies (1), and an edge test is conducted between v and*

w, and $C(v) \neq C(w)$, then w.h.p. the edge test is successful.

Proof: Because the edge test is symmetrical, we may assume without loss of generality that it takes place during v's turn. Thus, it suffices to show that, w.h.p., $S \cap N(G,w) < c_1 \log(n/\delta)$.

For each u, let X_u be the random variable that indicates whether $u \in N(G,v) \cap N(G,w)$. Suppose that $\{Y_u : u \in V\}$ are $\{0,1\}$-valued random variables such that $\Pr(Y_u = 1) = \min\left\{\frac{c_0 \log(n/\delta)}{|N(G,v)|}, 1\right\}$ for all u, and that are mutually independent. We can think of the variables Y_u as an extension of all of V of the random variables that were used to decide whether to include each member of $N(G,v)$ in S.

We claim that w.h.p.,

$$\sum_{u \in N(G,v)} Y_u \geq (c_0/2)\log(n/\delta). \tag{3}$$

If $|N(G,v)| \leq c_0 \log(n/\delta)$, this is because in this case $\Pr(Y_u = 1) = 1$ for all u, and therefore $\sum_{u \in N(G,v)} Y_u = |N(G,v)|$. Otherwise, it follows from a Chernoff bound, together with the fact that

$$\mathbf{E}\left(\sum_{u \in N(G,v)} Y_u\right) \geq c_0 \log(n/\delta).$$

Since S is a subset of the vertices for which $Y_u = 1$, we have that $|S \cap N(G,w)| \leq \sum_{u \in V - \{v,w\}} X_u Y_u$.

Note that, though $\{X_u Y_u : u \in V - \{v,w\}\}$ might not be independent, after we condition the values of $\{X_u : u \in V - \{v,w\}\}$, then they are. We will first obtain a high probability bound on $\sum_u X_u$, and then use it to get a bound on $\sum_u X_u Y_u$ that holds with high probability.

For each $u \in V - \{v,w\}$, either $u \notin C(v)$ or $u \notin C(w)$, so $\Pr(u \in N(G,v) \cap N(G,w)) \leq pq$. Thus $\mathbf{E}(\sum_u X_u) \leq pqn \leq qn \leq |C(v)|/2^9$ by (1), so long as $c \geq 2^9$. Note that each X_u is determined solely by the random generation of the graph, thus the various X_u variables are mutually independent. Applying a Chernoff bound, w.h.p., $\sum_u X_u \leq |C(v)|/2^8$. Since $p \leq 1/2$, w.h.p., $|N(G,v)| \geq |C(v)|/4$, so

$$\sum_u X_u \leq |N(G,v)|/2^6. \tag{4}$$

Since for each u, $\Pr(Y_u = 1) \leq \frac{c_0 \log(n/\delta)}{|N(G,v)|}$, we have that, conditioned on the event that (4) holds, $\mathbf{E}(\sum_u X_u Y_u) \leq (c_0/2^5)\log(n/\delta) \leq (c_1/2)\log(n/\delta)$. Applying another Chernoff bound together with the fact that the Y_u's are independent completes the proof. ■

Lemma 7 *For any C that satisfies (1) for all $v \in C$ and all vertices w, with high probability, if $\hat{C}(v) = \hat{C}(w)$ then $C(v) = C(w)$.*

Proof: This follows from induction using Lemma 6. ■

Definition 8 *The first member of a cluster C encountered by Subsquare is called the* head *of C.*

Lemma 9 *During the first pass through the vertices, for any C that satisfies (1), for any vertex v with an edge to the head v_C of C, w.h.p., $\hat{C}(v) = \hat{C}(v_C)$.*

Proof: The proof is by induction on $\pi(v)$.

If $|R_{temp}| \leq c_0 \log(n/\delta)$, then $v_C \in R$ and the lemma follows from Lemma 5.

Suppose $|R_{temp}| > c_0 \log(n/\delta)$. Since $p \geq 1/2$, w.h.p., $|R_{temp} \cap N(G,v_C)| > |R_{temp}|/4$. Thus, $E(|R \cap N(G,v_C)|) \geq (c_0/4)\log(n/\delta)$, and, therefore, w.h.p. $|R \cap N(G,v_C)| \neq \emptyset$, and the lemma follows from Lemma 5 together with the inductive hypothesis, together with the fact that Subsquare assigns v to the cluster in $\mathcal{D}$ with the least head node. ■

Lemma 10 *During the second pass through the vertices, for any C that satisfies (1), for any $v \in C$, $\hat{C}(v) = \hat{C}(v_C)$.*

Proof: Since $p \geq 1/2$, $\mathbf{E}(|N(G,v) \cap N(G,v_C)|) \geq |N(G,v)|/2$. During the second pass, $R_{temp} = N(G,v)$, thus, w.h.p., $|R \cap N(G,v_C)| \neq \emptyset$. By Lemma 9, though, for any $w \in N(G,v_C)$, after the first pass, w.h.p., $\hat{C}(w) = \hat{C}(v_C)$. So, w.h.p., some w with $\hat{C}(w) = \hat{C}(v_C)$ will be in R. The lemma then follows from Lemma 5. ■

Now that we have finished the proof of Theorem 1, we now turn to Theorem 2, the analysis of the running time.

Proof of Theorem 2: In $O(m \log n)$ time, Subsquare can create a data structure that will enable it to test membership in $N(G,v)$ for any vertex v in $O(\log n)$ time. This can be done by creating a balanced binary tree for each vertex v with the neighbors of v on the leaves. These "neighbor trees" can be found if there is a higher-level balanced binary tree that has a pointer to a neighbor tree for each vertex v at each leaf.

We can then see that, for each vertex, at most $O(\log(n/\delta))$ candidate clusters are examined, and each

candidate requires neighbor lookups for $O(\log(n/\delta))$ neighbors, each requiring $O(\log n)$ time. ■

4. Lower bounds

Theorem 11 *For any function ψ such that $\psi(n) = o(\log(n))$, there is no algorithm A with the following property:*

> *There are constants c, m_0 and n_0 such that, for all $p \in [1/2, 1], q \in [0, 1]$, $m \geq m_0$, and $n \geq n_0$, if G is a (p, q) clustered graph with clusters $C_1, ..., C_t$, then, with probability $1/8$, Algorithm A, outputs a partition $U_1, ..., U_s$ of the vertices of G such that, for all cluster indices i such that $|C_i| \geq c \max\{qn, \psi(n)\}$ there is a output cluster index j such that $C_i = U_j$.*

Proof Sketch: Suppose $p = 1/2$ and $q = 0$. Let $t = n/(c\psi(n))$.

We will show that, for a randomly chosen partition $C_1, ..., C_t$, Algorithm A fails with probability at least $1/8$, which will imply the existence of a partition $C_1, ..., C_t$ with this property. Suppose that the random partition over the vertices $v_1, ..., v_n$ is chosen by assigning v_i to C_i for $i \leq t$, and, for $i > t$, assigning each vertex independently to a random cluster. Let $\mathcal{C}$ be this clustering.

Suppose further that, if any vertex $v \notin \{v_1, ..., v_t\}$ (i.e., whose cluster assignment was chosen randomly) is not incident on any edges, then Algorithm A is given the cluster assignments of all vertices other than v. A lower bound given this assumption implies a lower bound for the original clustering problem, since A has the option to ignore the additional information.

If $\mathcal{C}' = (C'_1, ..., C'_t)$ consists of the cluster assignments to vertices other than v, we have $\Pr(C(v) = j|G) = \Pr(C(v) = j|N(G, v), \mathcal{C}')$.

Further, applying Bayes' rule, it is possible to show that $\Pr(C(v) = j|N(G, v) = \emptyset, \mathcal{C}') = \frac{2^{-|C'_j|}}{t \Pr(N(G,v)=\emptyset|\mathcal{C}')}$.

Thus, given that $N(G, v) = \emptyset$, together with the knowledge of the cluster assignments other than v, a Bayes optimal algorithm chooses arbitrarily from among the clusters of smallest size, and, if $n_i = |C'_i|$, then the probability that the Bayes optimal algorithm makes an error on v is $1 - \frac{\max_j 2^{-n_j}}{\sum_{j=1}^{t} 2^{-n_j}}$.

We claim that, for large enough n, with probability $1/2$, the cluster assignments to vertices other than v leave at least two clusters with one member each. Let each $B_1, ..., B_t$ be an indicator variable for whether the corresponding cluster in $C_1, ..., C_t$ has more than one member. $B_1, ..., B_t$ are negatively associated in the sense of (Joag-Dev & Proschan, 1983) (see (Dubhashi & Ranjan, 1998)). For each i, $\mathbf{E}(B_i) = (1 - 1/t)^{n-t-1}$. Since negatively associated random variables obey Chernoff bounds (see (Dubhashi & Ranjan, 1998)), we have

$$\begin{aligned}
&\Pr(\sum_i B_i \leq 2) \leq \exp(-(t/8)(1 - 1/t)^{n-t-1}) \\
&= \exp\left(-n\frac{(1 - 1/t)^{n-t-1}}{c\psi(n)}\right) \\
&= \exp(-n \exp(-\Theta(\psi(n)))) = \exp(-n^{1-o(1)}).
\end{aligned}$$

Thus, for large enough n, with probability at least $1/2$, we have that, for distinct i and j, $n_i = n_j = 1$, and thus the conditional probability that A makes a mistake, given that $N(G, v) = \emptyset$, is at least $1 - \frac{1}{2+\sum_{i=3}^{t} 2^{-n_i}} \geq 1/2$.

Now let us compute a lower bound on the probability that any vertex is disconnected. At least $t/2$ of the clusters have at most $2n/t$ elements. Let us call these clusters *light*. Let us lower bound the probability of an isolated vertex by the probability of an isolated vertex in a light cluster. For some such cluster, let ℓ be the number of elements in the cluster. Any individual vertex in the cluster is disconnected from the rest with probability $2^{-\ell}$, so the probability that the cluster is not connected is at least this large. Since the edges in different clusters are chosen independently, the probability that any light cluster is disconnected is therefore at least $1 - (1 - 2^{-2n/t})^{t/2}$ which is at least $1 - \exp(-(t/2)2^{-2n/t})$. Now,

$$\begin{aligned}
&\lim_{n\to\infty} \frac{t}{2} 2^{-2n/t} = \lim_{n\to\infty} \frac{n}{2\psi(n)} 2^{-2c\psi(n)} \\
&= \lim_{n\to\infty} \exp(\ln n - o(\ln n)) = \infty.
\end{aligned}$$

Thus, for large enough n, the probability that some cluster is disconnected is at least $1/2$, so that the probability of error is at least $1/8$. ■

Theorem 12 *For any function ψ such that $\psi(n) = o(n)$, there is no algorithm A with the following property:*

> *There are constants c, m_0 and n_0 such that, for all $p \in [1/2, 1], q \in [0, 1]$, $m \geq m_0$, and $n \geq n_0$, if G is a (p, q) clustered graph with clusters $C_1, ..., C_t$, then, with probability $3/4$, Algorithm A, outputs a partition $U_1, ..., U_s$ of the vertices of G such that, for all cluster indices i such that $|C_i| \geq c \max\{q\psi(n), \log n\}$ there is a output cluster index j such that $C_i = U_j$.*

Proof: Let $t = \lfloor \frac{n}{\psi(n)} \rfloor$ and $p = q = 1/2$. Note that, since $\psi(n) = o(n)$, if n is large enough, $t \geq 2$.

Suppose $\mathcal{C} = (C_1, C_2, ..., C_t)$ is obtained by picking an arbitrary vertex v, randomly partitioning the remaining vertices into equal-sized clusters, and then randomly assigning v to one of those clusters.

Since the edge probabilities are unaffected by the cluster assignments, and algorithm A that makes random cluster assignments is Bayes optimal, and therefore its probability of error is at least $1 - 1/t \geq 1/2$. Since this is true on average for a randomly chosen clustering $\mathcal{C}$, there exists a clustering $\mathcal{C}$ such that A has probability $1/2$ of making an error for $\mathcal{C}$. ■

5. Experiments

5.1. Modifications to Subsquare

We performed our experiments with a slightly modified version of Subsquare.

First, instead of independently randomly putting each vertex in R with probability $\frac{c_0 \log(n/\delta)}{|R_{temp}|}$, the modified algorithm sets R to be a random subset of size at most $c_0 \log(N/\delta)$, and it chooses S similarly.

Second, in Step 6 of Figure 1, instead of requiring that w has $c_0 \log(N/\delta)$ neighbors, the revised algorithm estimates the probability that a random neighbor of w is also a neighbor of v dividing the number of neighbors of w found in v's adjacency list by one more than the number of neighbors tried.

Third, when computing the above statistics, the counts for neighbors w of v that have been assigned to the same cluster $C(w)$ are consolidated, so that the algorithm computes an overall estimate $\hat{p}(C, v)$ of the probability that a random neighbor of v is in an adjacency list of a random vertex that has been assigned to cluster C. Vertex v is then assigned to the cluster, from among those for which $\hat{p}(C, v) \geq \theta$, which contained the most of the neighbors w that were examined during v's turn.

5.2. Other Algorithms

The first, inspired by the analysis of Achlioptas and McSherry (2007), randomly samples edges with probability p, and then applies the spectral clustering algorithm of Ng, Jordan and Weiss (2001). The NJW algorithm requires the user to specify the number k of clusters. We computed the top eigenvectors using the iterative block power method (Strang, 1986) which has the number of iterations as an adjustable parameter. The NJW algorithm also calls for the application of k-means, which also has a variable number of iterations. We experimented with various settings of these parameters, as described below.

The second is an algorithm that performs minhashing (Cohen, 1997; Broder, 1997), and forms a cluster from each bucket. This can be equivalently described as an algorithm that processes the vertices in random order, and, if a vertex v has not been assigned a cluster when it was encountered, a new cluster is formed with v and its unclustered neighbors.

The third is version 9.308 of MCL (van Dongen, 2000; Enright et al., 2002). The key parameter for MCL, which trades between precision and recall, is called "expansion". We found that, on this data, the default value of 2 optimized excessively for precision, so we tried 1.5, 2, and 3 which roughly covered the range of values suggested on the MCL website. To save space, we report only on the best result, for 1.5, below (the other results are significantly worse). While Subsquare and the minhash algorithm were coded in python in the most straightforward way, the MCL implementation is optimized C code.

5.3. Chinese Restaurant Process Data

To get large-scale data with known cluster assignments and some variety in cluster sizes, we generated data by first partitioning the vertices using a Chinese restaurant process (Pitman, 1995; Teh, 2010) with $\alpha = 1$ and $\theta = -20/n$. This tends to generate clusters of sizes roughly on the order of 20 vertices, which is similar to the most challenging applications targeted in this work. We generated graphs with $n = 10000, 20000, 50000, 100000$ vertices. We included edges between vertices in the same cell with probability $1/2$, and then added random edges until the number of such "noisy" edges was equal to the number of "clean" edges. For each such dataset we ran Subsquare and the minhash algorithm 10 times, and averaged the precision and recall obtained. (When computing precision and recall, we interpret a clustering as a series of predictions of whether each pair of vertices are in the same cluster.) We also recorded the time taken by each algorithm. We stopped the MCL algorithm after it ran for 1000 minutes on the two larger datasets.

We ran Subsquare with δ and c_1 chosen so that $c_1 \log(n/\delta) = 100$, and with the threshold θ described in Section 5.1 chosen to be 0.05.

Results are as follows (time is in minutes):

n	Minhash		Subsquare		MCL	
	time	F	time	F	time	F
10000	1	0.24	3	0.97	47	0.96
20000	1	0.24	6	0.98	175	0.97
50000	1	0.24	13	0.99	>1000	
100000	3	0.24	28	0.99	>1000	

The time required by Subsquare is seen to scale linearly with the number of vertices. Subsquare achieves better accuracy than MCL while requiring an order of magnitude less time. The time required by MCL appears to be scaling superlinearly.

For the sampling spectral algorithm, we tried (a) sampling each edge with various probabilities p: 0.05, 0.1, 0.2; (b) setting the number of clusters to various values s: 100, 200, 500 (since the expected size of each cluster is 20 the true number is approximately 500); (c) running the block power method for various numbers of iterations I_{pow}: 10, 25, and 100; (d) running k-means for various numbers of iterations I_k: 2, 5, and 10. We tried all 81 combinations of these parameters on the $n = 10000$ dataset. After eliminating the combinations that required more than 1000 minutes to run, the five combinations of parameters with the best values of the F-score are shown, along with the Subsquare result.

p	s	I_{pow}	I_k	time	F-score
0.2	200	25	10	503	0.59
0.2	200	10	10	231	0.55
0.2	200	10	5	212	0.53
0.2	200	25	5	466	0.52
0.2	100	100	10	920	0.40
Subsquare				3	0.97

5.4. Bi-holdout experiments

To evaluate algorithms on the DBLP data, we used a single training-test split variant of bi-cross-validation (Gabriel, 2002; Owen & Perry, 2009), which might be called a bi-holdout experiment. For graph-based clustering, the most natural application of bi-cross-validation appears to be to repeatedly apply the following steps: (a) Randomly split the vertices into training vertices U and test vertices W; (b) Run each algorithm on the subgraph induced by U, called the "training graph"; (c) For each test vertex $w \in W$, determine the cluster assignment of w by using the assignment made to a random neighbor of w in U. (The edges of the graph consisting only of edges with one vertex in W and one vertex in U, which might be called the "bridge graph", are used for this.); (d) For each pair of vertices in W, predict that there is an edge between them if and only if they are assigned to the same cluster in the preceding step; (e) Compare the predictions with the actual presence of absence of edges between vertices in W (these edges constitute the "test graph"). We only used a single training-test split because our graphs are large. A fraction 1/10 of the vertices were placed in the test graph.

We first used the bi-holdout estimate to compare the algorithms on the synthetic data described above. We applied both Subsquare and the minhash algorithm to the graph in the n=100000 case, and compared the results using this protocol. The Subsquare algorithm achieved an F-score of 0.136, where the minhash algorithm obtained 0.046. While Subsquare still performs better, the edge-prediction accuracy obtained is much smaller than the cluster-membership accuracy that was measured earlier. This is not surprising, in light of the fact that there are three chances for a correct determination that two vertices u and v in the test graph should be clustered together to fail to be recognized by this protocol: (1) the edge between u and v may be missing from the graph, (2) a spurious edge from u to the training graph may be chosen, and (3) a spurious edge from v to the training graph may be chosen. Nevertheless, since these should be expected to effect all algorithms equally, the statistics obtained through this protocol should still give an idea of the relative merit of clustering algorithms.

Next, we applied performed a similar experiment using the titles in the DBLP database. We used a variant of minhashing (Cohen, 1997; Cohen et al., 2001; Li et al., 2008) to compute the similarity graph. Each title was summarized using a bag of its length-4 substrings (called "k-mers", with $k = 4$). Next, we removed all k-mers that appeared in at least 1000 titles from all bags. We then fixed a random permutation of the k-mers, and further summarized each title by a list of the first 25 k-mers, in sorted order according to this permutation, appearing in its bags. Titles whose resulting "sketches" shared at least 5 k-mers were deemed similar and given an edge in the graph. The resulting graph had roughly 1.5 million edges, and was clustered by Subsquare on a four-year-old desktop workstation in less than 25 minutes. The bi-holdout F-score for Subsquare (here with δ and c_1 chosen so that $c_1 \log(n/\delta) = 100$ and $\theta = 0.9$)) was 0.504, where the minhash algorithm obtained 0.471.

References

Achlioptas, D. and McSherry, F. Fast computation of low-rank matrix approximations. *JACM*, 54(2), 2007.

Balcan, M., Blum, A., and Vempala, S. A discriminative framework for clustering via similarity functions. *STOC*, 2008.

Bansal, N., Blum, A., and Chawla, S. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

Broder, Andrei Z. On the resemblance and containment of documents. *SEQUENCES*, pp. 21–29, 1997.

Bui, T. N., Leighton, F. T., Chaudhuri, S., and Sipser, M. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987.

Cohen, E. Size-estimation framework with applications to transitive closure and reachability. *JCSS*, 55(3):441–453, 1997.

Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J. D., and Yang, C. Finding interesting associations without support pruning. *IEEE Trans. Knowl. Data Eng.*, 13(1):64–78, 2001.

Condon, A. E. and Karp, R. M. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):221–232, 2001.

Drineas, P. and Mahoney, M. W. Approximating a gram matrix for improved kernel-based learning. *COLT*, 2005.

Dubhashi, D. and Ranjan, D. Balls and bins: A study in negative dependence. *Random Structures & Algorithms*, 13(2):99–124, 1998.

Enright, A. J., Van Dongen, S., and Ouzounis, C. A. An efficient algorithm for large-scale detection of protein families. *Nucl. Acids Res.*, 30(7):1575–1584, 2002.

Frieze, A., Kannan, R., and Vempala, S. Fast monte-carlo algorithms for finding low-rank approximations. *JACM*, 51(6):1025–1041, 2004.

Gabriel, K. Le biplot-outil d' exploration de donn'ees multidimensionelles. *Journal de la Societe Francoise de Statistique*, 143:5–55, 2002.

Gibson, D., Kumar, R., and Tomkins, A. Discovering large dense subgraphs in massive graphs. *VLDB*, 2005.

Jerrum, M. and Sorkin, G. B. The metropolis algorithm for graph bisection. *Discrete Applied Mathematics*, 82(1-3):155–175, 1998.

Joag-Dev, K. and Proschan, F. Negative association of random variables, with applications. *The Annals of Statistics*, 11(1):286–295, 1983.

Kannan, R., Vempala, S., and Vetta, A. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, 2004.

Kumar, S., Mohri, M., and Talwalkar, A. On sampling-based approximate spectral decomposition. *ICML*, 2009.

Li, P., Church, K. W., and Hastie, T. L. One sketch for all: theory and application of conditional random sampling. *NIPS*, 2008.

McSherry, F. Spectral partitioning of random graphs. *FOCS*, 2001.

Motwani, R. and Raghavan, P. *Randomized Algorithms*. Cambridge University Press, 1995.

Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. *NIPS*, 2001.

Owen, A. B. and Perry, P. O. Bi-cross-validation of the SVD and nonnegative matrix factorization. *The Annals of Applied Statistics*, 3(2):564–594, 2009.

Pitman, J. Exchangeable and partially exchangeable random partitions. *Probab. Theory Relat. Fields*, 102:145–158, 1995.

Strang, G. *Linear Algebra and its applications – third edition*. Saunders College, 1986.

Teh, Y. W. Dirichlet processes. In *Encyclopedia of Machine Learning*. 2010.

van Dongen, S. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, 2000.

von Luxburg, U. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

Williams, C. K. I. and Seeger, M. Using the Nyström method to speed up kernel machines. *NIPS*, 2000.