Cluster Ranking with an Application to Mining Mailbox Networks

Ziv Bar-Yossef[*]
Department of Electrical Engineering, Technion
and Google Inc.
Haifa, Israel
zivby@ee.technion.ac.il

Ido Guy
Department of Computer Science, Technion
and IBM Research Lab
Haifa, Israel
ido@cs.technion.ac.il

Ronny Lempel
IBM Research Lab
Haifa 31905, Israel
rlempel@il.ibm.com

Yoëlle S. Maarek[†]
Google Inc.
Haifa Engineering Office, Israel

Vladimir Soroka
IBM Research Lab
Haifa 31905, Israel
vladi@il.ibm.com

Abstract

We initiate the study of a new clustering framework, called cluster ranking. *Rather than simply partitioning a network into clusters, a cluster ranking algorithm also orders the clusters by their* strength. *To this end, we introduce a novel strength measure for clusters—the integrated cohesion—which is applicable to arbitrary weighted networks.*

We then present C-Rank: a new cluster ranking algorithm. Given a network with arbitrary pairwise similarity weights, C-Rank creates a list of overlapping clusters and ranks them by their integrated cohesion. We provide extensive theoretical and empirical analysis of C-Rank and show that it is likely to have high precision and recall.

Our experiments focus on mining mailbox networks. *A mailbox network is an egocentric social network, consisting of contacts with whom an individual exchanges email. Ties among contacts are represented by the frequency of their co-occurrence on message headers. C-Rank is well suited to mine such networks, since they are abundant with overlapping communities of highly variable strengths. We demonstrate the effectiveness of C-Rank on the Enron data set, consisting of 130 mailbox networks.*

1. Introduction

Cluster ranking. When clustering large networks, clustering algorithms frequently produce masses of clusters. This phenomenon is magnified when employing "fuzzy" or "soft" clustering methods, which partition the network into overlapping clusters. These tend to generate numerous clusters even on small networks. The abundance of clusters may make the results hard to digest and interpret. Moreover, typically only a small portion of the clusters are interesting or meaningful, giving rise to a "needle in a haystack" problem: how to select the important clusters from the masses of results returned?

In order to address the above difficulties, we propose a new clustering framework, called *cluster ranking*. Given a *cluster strength measure*, which assigns a "strength score" to every subset of nodes, and given a *maximality criterion*, which determines which sets of nodes are "self-contained", a cluster ranking algorithm outputs the maximal clusters in the network, ordered by their strength. The ranking provides information that is usually not conveyed by traditional clustering: which clusters are more important than others. This information can be used, for instance, to quickly single out the most significant clusters. Similarly to search algorithms in information retrieval, cluster ranking algorithms are measured by *precision* and *recall*. Our new framework is described in Section 3.

Cluster strength measure. A crucial ingredient in the new framework is the choice of a suitable cluster strength measure. A proper definition of such a measure turns out to be a major challenge. Even for unweighted networks, there is no consensus on how to measure quality of a cluster or of a clustering [8, 19].

We propose a novel cluster strength measure—the *integrated cohesion*—which is applicable to arbitrary weighted networks. To define this measure, we first define the *cohesion* of unweighted clusters. Several notions of edge separators [31, 19, 15, 26, 12] have been used in the past to

[*] Supported by the European Commission Marie Curie International Re-integration Grant.

[†] Done in part when the author was at the IBM Research Lab in Haifa.

capture how "cohesive" an unweighted cluster is. We observe that these notions are unsatisfactory, especially in the presence of overlapping clusters. We then show that *vertex separators*, rather than edge separators, are more effective in measuring cohesion.

Extending cohesion to capture strength of weighted clusters is tricky, since edge weights have to be taken into account as well. A standard approach for handling edge weights is "thresholding": one determines a threshold T, and transforms the weighted network into an unweighted network, by keeping only the edges whose weight exceeds the threshold T. We show that standard thresholding is insufficient for measuring strength of weighted clusters. We then introduce *integrated cohesion* as an effective measure of strength for weighted clusters. The integrated cohesion of a cluster is the sum of the cohesion scores of all the unweighted clusters obtained by applying all possible thresholds to the given weighted cluster. Our new cluster strength measures are discussed in Section 4.

Cluster ranking algorithm. Having set up the new framework, we present C-Rank: a cluster ranking algorithm. C-Rank is designed to work for networks with arbitrary weights. The network's nodes are assumed neither to belong to a metric space nor to conform to any statistical model. C-Rank produces and ranks overlapping clusters and is thus in particular an overlapping clustering algorithm.

C-Rank works in three phases. First, it identifies a list of candidate clusters. Then, it ranks these candidates by their integrated cohesion. Finally, it eliminates redundant clusters—ones that are non-maximal.

At the core of C-Rank is a hierarchical overlapping clustering procedure, which constructs a hierarchy of overlapping clusters in unweighted networks. This procedure may be of independent interest. Given a network G, the procedure finds a *sparse vertex separator* in G, and uses the separator to split the network into a collection of overlapping clusters. The procedure then recurses on each of the clusters, until reaching cliques or singletons. Interestingly, the hierarchy produced by the procedure may be a DAG, rather than a tree. We provide rigorous theoretical analysis of this procedure and show that it is guaranteed to find *all* maximal clusters in G (note that other soft clustering algorithms may not have this guarantee and are thus less useful in our framework). The procedure may run in exponential time in the worst-case—an unavoidable artifact of the quest for overlapping clusters. Yet, we show that its running time is only polynomial in the output length. In practice, it took C-Rank several minutes to cluster networks consisting of thousands of nodes on a standard PC.

Given a weighted network, C-Rank produces candidate clusters by transforming the network into multiple unweighted networks using a gradually increasing threshold. The hierarchical overlapping clustering procedure is used to extract clusters from each of these unweighted networks. Full details of the algorithm are given in Section 5.

Mailbox networks. We demonstrate the efficacy of the novel framework and of the C-Rank algorithm in a new domain: clustering mailbox networks. A *mailbox network* is an "egocentric" social network [29, 35]—a network centered around a root individual. Unlike global "sociocentric" networks [14], it provides the subjective viewpoint of an individual on her social environment. A mailbox network is generated by mining messages in an individual's mailbox. Actors in this network are the individual's group of contacts. The weight of an edge connecting two actors is the number of messages on whose header both actors appear (either as co-recipients, or as a sender and a recipient). This weight represents the strength of the ties between the two actors from the individual's perspective. Mailbox networks are abundant with overlapping communities of variable strengths, and thus C-Rank is highly suitable for them.

Automatically discovering communities within mailbox networks could be beneficial in various applications. In email clients, the knowledge of one's favorite communities could support the automation of a variety of features such as completion of groups when entering multiple recipients, detection of missing or redundant recipients, etc. Email communities might also help in spam filtering by identifying "spam groups" [5]. In the intelligence domain, communities can evidence gangs or potential criminal groups around known criminals.

Experimental results. We evaluated C-Rank on our own mailboxes as well as on 130 mailboxes from the Enron data set [21]. To evaluate the quality of C-Rank, we adapted the popular edge betweenness clustering algorithm of Girvan and Newman [15] to the cluster ranking framework, and compared the two algorithms. We found that C-Rank dominates the edge betweenness algorithm under almost any metric. We also evaluated the robustness of C-Rank under random removal of data, and found it to be quite resilient. These results are presented in Section 6.

2. Related work

The literature on clustering and community detection consists of numerous measures of quality for communities and clustering. These vary from distance-based metrics (such as minimum diameter, sum-of-squares, k-means, and k-medians, cf. [18]), to graph-theoretic measures (such as normalized cuts [31], conductance [19], degree-based methods [12, 13, 17], performance [34], edge betweenness [15], modularity [26], bipartite cores [22], and k-cliques [27]), to statistical methods (e.g., [3]). Unfortunately, there is no single, widely acceptable, definition, and many of the above notions are known to work badly in some situations (cf. [8, 19, 20]). Furthermore, many of the above measures are suited for restricted scenarios, such as hard partitional clustering, model-based clustering, or clustering of metric space data points.

Fuzzy cluster analysis (cf. [16]) is a branch of data clustering, in which each data point can be associated with multiple clusters with different confidence probabilities. Fuzzy clustering can be used in particular to generate overlapping clusters. Nevertheless, most of the classical work in the area (e.g., Fuzzy c-means) assumes the data points lie in a metric space, which respects the triangle inequality. We consider arbitrary weighted networks whose induced distance measure does not necessarily satisfy the triangle inequality. More recent studies (e.g., [32, 30, 2, 7, 27, 4]) address the general scenario of networks with arbitrary pairwise similarity weights. These algorithms substantially differ from ours, because they do not rank clusters and are not guaranteed to output all maximal clusters.

Pereira, Tishby, and Lee [28] present a hierarchical soft clustering algorithm for weighted networks that lie in a metric space, using a technique called *deterministic annealing*. This technique bares some similarity to the increasing threshold used by C-Rank to find candidate clusters.

Several works studied communities in email networks. Tyler *et al.* [33] mined communities in *sociocentric* email networks, i.e., ones extracted from the viewpoint of an organization's mail server. Fisher and Dourish [11, 10] study egocentric mailbox networks as we do, yet they detect communities by manual inspection and not by an automatic algorithm. Boykin and Roychowdhury [5] mined communities in mailbox networks in order to detect "spam communities". Their clustering algorithm, however, is too coarse to reveal the overall community structure of the network. McCallum *el al.* [24] cluster email messages in an individual's mailbox, based on their text content, rather than on the message headers.

3. Cluster ranking framework

Throughout, $G = (V_G, E_G)$ is an undirected network and $n = |V_G|$ is the number of nodes in G. G has no parallel edges, yet self loop edges are allowed. Every edge $e \in E_G$ is associated with a non-negative weight $W(e)$. The weight represents the strength of the tie between the two connected nodes. The self loop weight represents the intrinsic "importance" of the corresponding node. If u and v are not connected by an edge, we implicitly assume $W(u, v) = 0$. In the special case all edge weights are 1, G is called an *unweighted* network. Note that edge weights can be arbitrary, and in particular need not correspond to a metric.

The first basic ingredient of the cluster ranking framework is the following:

Definition 1 (Cluster strength measure). A *cluster strength measure* is a function μ, mapping networks to non-negative real values. $\mu(C)$ is the *cluster strength* of a network C.

Intuitively, $\mu(C)$ represents how "strong" C is as a cluster. There could be many possible realizations of this definition, depending on the properties of a cluster viewed as making it "strong". One simple example is the *clique strength measure* for unweighted networks. This measure takes on only Boolean values: a network C is of strength 1 if it is a clique, and is of strength 0 otherwise.

Cluster strength is an intrinsic property of the network C. Typically, C is a subset of a larger network G and thus cluster strength by itself is insufficient to represent the "desired" clusters in a network. For example, a small clique A, which is embedded in a larger clique B, is strong under the clique strength measure, but is evidently not very interesting, because it is simply an integral part of the larger clique. In order to capture these redundant clusters, we introduce the second basic ingredient of the framework:

Definition 2 (Maximality criterion). Let $G = (V_G, E_G)$ be a network. A *maximality criterion* is a Boolean function, mapping subsets of V_G to $\{0, 1\}$. All the subsets that are mapped to 1 are called *maximal* and all the subsets that are mapped to 0 are called *non-maximal*.

A natural maximality criterion in the cliques example maps a set C to 1 if and only if it is a clique and not contained in any other clique. The maximal clusters in this case are the maximal cliques in G.

We can now state the cluster ranking problem:

The cluster ranking problem

Input: *A network G.*
Output: *The maximal clusters in G ordered by strength.*

The cluster ranking problem, as stated, could be a hard optimization problem. One immediate difficulty is that the number of maximal clusters may be very large, so just outputting them may take a long time. We thus measure the performance of ranking algorithms not only relative to the input length but also relative to the output length. A more serious problem is that typically the computational problem itself (even when the output is short) is hard. It follows that in reality we cannot expect a ranking algorithm to provide an exact solution to the ranking problem. A typical ranking algorithm may include on its list non-maximal clusters and/or may miss some maximal clusters. We thus adapt information retrieval metrics to evaluate the quality of cluster ranking algorithms.

For a network G and for a ranking algorithm A, let $A(G)$ be the list of clusters returned by A when given G as input.

Let $I(G)$ denote the list of all maximal clusters in G. The *recall* of A is: $\text{recall}(A, G) = \frac{|A(G) \cap I(G)|}{|I(G)|}$. The *precision* of A is: $\text{precision}(A, G) = \frac{|A(G) \cap I(G)|}{|A(G)|}$.

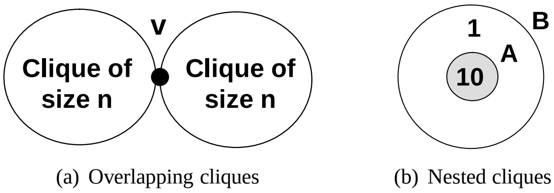
4. New cluster strength measure and maximality criterion

In this section we develop a new measure of cluster strength and a corresponding maximality criterion. Our measure is quite general, and in particular is suited for finding overlapping clusters in networks with arbitrary weights.

4.1. Unweighted networks

In unweighted networks a strong cluster is one which is "cohesive" in the sense that it does not "easily" break up into smaller pieces. This intuition has been formalized via various notions of graph partitioning, such as normalized cuts [31], conductance [19], edge betweenness [15], modularity [26], and relative neighborhoods [12]. The underlying principle in all these approaches is the same: a network is cohesive if and only if it does not have any "weak" edge separator (a.k.a. edge cut). An *edge separator* is a subset of the network's edges whose removal from the network makes the network disconnected. The above approaches differ in the way they measure the "weakness" of the edge separator.

We observe that regardless of the weakness measure used, edge separators sometimes fail to capture the cohesion of networks, especially in the presence of overlapping clusters. While the existence of a weak edge separator in a network is sufficient to make the network noncohesive, it is not a necessary condition. A simple example for this is illustrated in Figure 1(a). Here, we have two cliques of size n that overlap in a single node. It is easy to check that any edge separator of this network has $\Omega(n)$ edges and thus will be considered relatively strong almost under any measure. However, this network is clearly noncohesive because it naturally decomposes into the two overlapping cliques.



(a) Overlapping cliques (b) Nested cliques

Figure 1. Clique examples

We propose using *vertex separators*, rather than edge separators, to measure the cohesion of a network. A vertex separator is a subset of the network's nodes whose removal leaves the network disconnected. In the example network above, the single node in which the two cliques overlap is a vertex separator. Intuitively, a network is cohesive if and only if it does not have a small vertex separator that separates it into large pieces.

Formally, a *vertex separator* of an undirected and unweighted network $C = (V_C, E_C)$ is a subset S of V_C s.t. the network induced on $V_C \setminus S$ (i.e., the network obtained from C by removing S and all its incident edges) is disconnected. A *partition* induced by a vertex separator S is a partition of $V_C \setminus S$ into two disjoint sets A and B s.t. no edge in E_C connects A and B. Note that the same separator may induce multiple different partitions. We define the cohesion of a network via the notion of "vertex separator sparsity" (cf. [1, 9]):

Definition 3 (Network cohesion). Let $C = (V_C, E_C)$ be an unweighted network. The *cohesion* of C is:

$$\text{cohesion}(C) = \min_{(S,A,B)} \frac{|S|}{\min\{|A|, |B|\} + |S|},$$

where the minimum is over all vertex separators S of C and over all partitions of C induced by S. The cohesion of a singleton (a cluster of size 1) is 1, if it has a self loop, and 0 otherwise.

The ratio $\frac{|S|}{\min\{|A|,|B|\}+|S|}$ is called the *sparsity* of the partition. It is minimized when S is small and A and B are both large. That is, under the above definition, a network C is cohesive if and only if it cannot be broken into large pieces by removing a small number of nodes from the network. The fact that the two pieces are large is important, because it may be easy to cut off a small part from a network, even if the network is cohesive, e.g., by isolating a single leaf node.

The cohesion of a network takes on values between 0 (for disconnected networks) and 1 (for cliques). Note that sparse vertex separators subsume weak edge separators: if the network has a weak edge separator, then it must also have a sparse vertex separator. However, as the example network above demonstrates, the converse is not true.

Computing the cohesion of a network is an NP-hard optimization problem [6]. Yet, it can be approximated in polynomial time [23, 9]. In this paper we use a faster flow-based approximation of network cohesion, which is described in the full version of this paper [1].

4.2. Weighted networks

In weighted networks cohesion is no longer the sole factor determining cluster strength. Edge weights should be taken into account as well. For example, a clique of size n all of whose edges are of weight 1 and a clique of size n all of whose edges are of weight 100 are equally cohesive. Yet, clearly the latter clique is "stronger" than the former. How do we then combine cohesion and edge weights into a single strength measure?

[1] Manuscript of full draft available at http://www.ee.technion.ac.il/people/zivby.

One of the popular methods for dealing with weighted networks is "thresholding" (see, e.g., [27]): given a weighted network C, one selects a *weight threshold* $T \geq 0$, and transforms C into an unweighted network C^T by changing all the weights that are greater than T to 1 and all the weights that are at most T to 0. C is then clustered by simply clustering C^T. This approach, though, is too coarse, especially in the presence of overlapping clusters. To illustrate the problem, consider the example network depicted in Figure 1(b). In this example, we have two nested cliques. A smaller clique A all of whose edges are of weight 10 is nested within a larger clique B, whose other edges are of weight 1. Clearly, both A and B are clusters of interest, yet any choice of a single threshold results in the loss of at least one of them. If the threshold is set to be less than 1, then A is lost, while if the threshold is set to be at least 1, then B is lost.

Our crucial observation is that in order to determine the strength of a weighted network, we should not fix a single weight threshold, but rather consider all possible weight thresholds *simultaneously*. A strong cluster is one that has high cohesion under many different thresholds. Formally, this is captured by the following measure:

Definition 4 (Integrated network cohesion). Let C be a weighted network. The *integrated cohesion* of C is:

$$\mathrm{intcohesion}(C) = \int_0^\infty \mathrm{cohesion}(C^T)dT.$$

For example, the integrated cohesion of a clique all of whose edges are of weight k is k. Similarly, the integrated cohesion of a singleton whose self loop weight is k is also k. Although integrated cohesion is defined as a continuous infinite sum, in practice: (1) It is always finite, as for all thresholds T that are greater than the maximum edge weight, C^T is an empty graph, and thus $\mathrm{cohesion}(C^T) = 0$. (2) It can be computed by summing up a finite number of cohesion values. The only weight thresholds in which the induced unweighted network can change are the distinct edge weights of C. Therefore, by summing up at most $|E_C|$ cohesion scores, one can compute the integrated cohesion.

4.3. Maximality criteria

We now define maximality criteria for weighted and unweighted networks.

Unweighted networks. In order to define maximality in unweighted networks, we first discuss the notion of *cluster subsumption*. Our maximal clusters will be the ones that are not subsumed by any other cluster.

Let us begin with a motivating example. Consider the two clusters depicted in Figure 2. The larger cluster D is the union of two overlapping cliques D_1, D_2 of size n each, whose overlap is of size k. The smaller cluster C is a union of two overlapping cliques $C_1 \subseteq D_1$, $C_2 \subseteq D_2$ of size $n/2$ each, whose overlap coincides with $D_1 \cap D_2$. It can be checked that C is more cohesive than D, yet clearly C is "uninteresting", since it is an integral part of D. We would like then to say that D *subsumes* C, and thus C cannot be maximal. In fact, in this example C is not unique. Any union of a subset of D_1 with a subset of D_2 whose overlap coincides with $D_1 \cap D_2$ will give a cluster, which is more cohesive than D, but is subsumed by D.

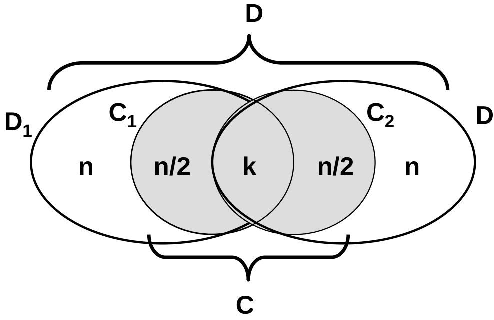


Figure 2. Example of cluster subsumption.

What really makes D subsume C in the above example? If we break up D into its natural clusters D_1 and D_2, then also C breaks up into different pieces (C_1 and C_2). That is, the partition of D *induces* a partition of C.

To formally define subsumption, we introduce some terminology:

Definition 5 (Covers). Let V be a set. A *cover* of V is a collection of subsets $V_1, \ldots, V_k \subseteq V$ whose union is V: $\bigcup_{i=1}^k V_i = V$. Note that sets participating in a cover, unlike a partition, can overlap. The cover is called *trivial*, if at least one of $V_1, \ldots, V_k$ equals V. Given a subset $V' \subseteq V$, the cover of V' *induced* by $V_1, \ldots, V_k$ is $V'_1 = V_1 \cap V', \ldots, V'_k = V_k \cap V'$.

Vertex separators not only provide us with a robust notion of network cohesion, but they also enable us to break up networks into their "natural" top-level clusters:

Definition 6 (Vertex separator cover). Let $G = (V_G, E_G)$ be an unweighted network and let S be a vertex separator of G. Let $A_1, \ldots, A_k$ be the k connected components of $G \backslash S$. The *S-cover* of G is $S \cup A_1, S \cup A_2, \ldots, S \cup A_k$.

Note that the clusters participating in a vertex separator cover overlap, because all of them contain the separator. In the example depicted in Figure 2, the intersection $D_1 \cap D_2$ is a (sparsest) vertex separator of both D and C. The corresponding covers of D and C are D_1, D_2 and C_1, C_2, respectively.

Definition 7 (Subsumption). Let $C \subsetneq D$ be two clusters in an unweighted network G. D is said to *subsume* C, if there exists a sparsest vertex separator S of D, whose corresponding cover induces a non-trivial cover of C.

In the example above D subsumes C, because the cover corresponding to the sparsest vertex separator of D is D_1, D_2, and this cover induces the non-trivial cover C_1, C_2 of C.

The notion of subsumption does not properly handle cliques, because the vertex separator of any clique is already trivial, and thus the covers it induces on all its subsets are trivial too. In particular, non-maximal cliques are not subsumed by any of their supersets under this definition. To fix this anomaly, we explicitly postulate that if D is a clique, then it subsumes all its proper subsets.

We can now define maximality in unweighted networks:

Definition 8 (Maximality in unweighted networks). Let $G = (V_G, E_G)$ be an unweighted network. A subset $C \subseteq V_G$ is called *maximal*, if it is not subsumed by any other subset of V_G.

In the example network depicted in Figure 2, the cluster C is non-maximal, because it is subsumed by the cluster D.

The following lemma shows that the above maximality criterion captures natural types of clusters:

Lemma 9. *Let G be an unweighted network. Then, the connected components of G and the maximal cliques in G are maximal.*

For lack of space, the proof of this lemma, as all other proofs in this paper, appears in the full version of the paper.

Weighted networks. Having defined maximality in unweighted networks, it is quite straightforward to extend the definition to weighted networks:

Definition 10 (Maximality in weighted networks). Let $G = (V_G, E_G)$ be a weighted network. A subset $C \subseteq V_G$ is called *maximal*, if there exists at least one threshold $T \geq 0$, for which C is maximal in the unweighted network G^T.

In the example network depicted in Figure 2, if the edges of the cluster C are all of weight 10 and the rest of the edges in the cluster D are of weight 1, then C is now maximal, because it is maximal in the unweighted network G^T, for all $T \in [1, 10)$.

Remark. A common pattern in social networks is the "onion pattern" [11]: a sequence of nested clusters, each of which is only slightly stronger than the cluster it is contained in. This pattern characterizes, for instance, the collaboration within projects: most of the interaction occurs within a core team of project members, while larger circles of consultants are only peripherally involved. The different layers of an "onion" give rise to clusters that are all maximal. Nevertheless, it is clear that not all of them are of interest. This motivates us to search for clusters that are not just maximal but are rather maximal *by a margin*.

We say that a cluster C is *maximal by a margin* ϵ, if there exists an interval $[T_1, T_2]$, where $T_2 \geq (1 + \epsilon)T_1$, s.t. C is maximal in G^T, for all $T \in [T_1, T_2]$. For instance, if in the network depicted in Figure 2, the weight of edges in C is 1.1 rather than 10, then C is maximal by a margin of 0.1.

5. The C-Rank algorithm

In this section we describe C-Rank: an algorithm for detecting and ranking clusters in weighted networks. C-Rank consists of three major phases: (1) identification of candidate clusters; (2) ranking the candidates by integrated cohesion; and (3) elimination of non-maximal clusters.

5.1. Candidate identification in unweighted networks

Our candidate identification procedure (see Figure 3) finds the sparsest vertex separator of the given network, uses its induced cover to split the network into overlapping clusters, and then recurses on the clusters. The recursion stops when reaching cliques or singletons, since they cannot be further partitioned. If more than one vertex separator exists, one of them is chosen arbitrarily.

```
Procedure unweightedCRank($G$, $\mathcal{L}$)
add $G$ to $\mathcal{L}$
if $G$ is a clique or a singleton return
$S$ := sparsest vertex separator of $G$
$A_1, \ldots, A_k$ := connected components of $G \setminus S$
for i = 1 to k do
  $G_i$ := sub-network of $G$ induced on $S \cup A_i$
  if $G_i$ not already in $\mathcal{L}$ then
    unweightedCRank($G_i$, $\mathcal{L}$)
```

Figure 3. Identifying candidate clusters in unweighted networks.

As the procedure detects overlapping clusters, it may encounter the same cluster more than once. Thus, in order to avoid duplications, the procedure checks that a cluster is not already on the list, before recursively processing it.

The procedure not only produces a list of maximal clusters from the given network G, but also implicitly organizes them in a *hierarchy*, similarly to hierarchical clustering. The difference is that here, due to the overlapping clusters, the hierarchy is not necessarily a tree, but is rather a DAG (Directed Acyclic Graph). The root of the hierarchy is the whole network G and its leaves are either singletons or cliques. Each cluster in the hierarchy is covered by its child clusters. We call such a hierarchy a *hierarchical overlapping clustering*.

Example run. Figure 4 shows an example run of the above procedure on a simple 5-node network. The procedure first detects $S = \{c, d\}$ as the sparsest vertex separator of the network and removes it from the network. The resulting connected components are $A_1 = \{a, b\}$ and $A_2 = \{e\}$. The procedure adds S to each of the connected components, obtaining the two overlapping clusters $\{a, b, c, d\}$ and $\{c, d, e\}$. No recursive calls need to be made in this example, because both of these clusters are cliques.

Analysis. We next analyze the quality and the performance of the algorithm. We start by showing that the algorithm is guaranteed to have an ultimate recall of 1:

Lemma 11. *Given an unweighted network G, C-Rank outputs all the maximal clusters in G.*

The lemma establishes that C-Rank has ultimate recall. But what about precision? How likely is C-Rank to output clusters that are non-maximal? When C-Rank splits a cluster C into sub-clusters $C_1, \ldots, C_k$ using a vertex separator, $C_1, \ldots, C_k$ are not subsumed by C. If C is maximal, then $C_1, \ldots, C_k$ are likely to be maximal too. However, this intuition does not always work, as $C_1, \ldots, C_k$ may be subsumed by subsets of C. This situation, though, rarely happens. We do not have theoretical guarantees about the precision of C-Rank, but we provide empirical evidence in Section 6 that its precision is good.

The performance of C-Rank is directly related to the number of clusters it produces. Clearly, since the number of maximal clusters can be exponential in the size of the input network G, then C-Rank may run for an exponential amount of time. However, when the list of maximal clusters is short, then C-Rank will also run more quickly:

Lemma 12. *Suppose that on a given network G C-Rank outputs a list of m candidate clusters $C_1, \ldots, C_m$. Then, the running time of C-Rank is $O(\sum_{i=1}^{m} (f(|C_i|) + |C_i|^2))$, where $f(n)$ is the amount of time needed to compute the sparsest vertex separator of a network of size n.*

Recall that finding the sparsest vertex separator of a network is NP-hard. Hence, in a naive implementation of C-Rank, $f(n)$ will be exponential in n, which is of course unacceptable. Therefore, C-Rank does not compute exact sparsest separators, but rather approximate sparsest separators. These separators are computable in quadratic time. The approximation procedure is described in the full version of the paper.

5.2. Candidate identification in weighted networks

The simplest way to extract all maximal clusters from a weighted network $G = (V_G, E_G)$ is the following. We enumerate all possible thresholds T (there are at most $|E_G|$ such thresholds), compute G^T, and output all the maximal clusters in G^T using the procedure unweightedCRank. This guarantees that we output all maximal clusters of G, and hence obtain ultimate recall.

The above brute force enumeration could be very time-consuming, since we need to make up to $|E_G|$ calls to unweightedCRank, and each call is made over the entire network G. Furthermore, this approach tends to be wasteful, as we may identify the same clusters again and again under different thresholds. For example, a maximal clique all of whose edges are of weight T will be discovered at all thresholds $T' < T$. A natural question is then whether we can trade the ultimate recall guarantee for better performance?

To this end, we make the following observation. What is the reason for a cluster to be maximal at G^T, for some threshold T, while not being maximal at $G^{T'}$, for all $T' < T$? This can happen only if for every $T' < T$, there exists a cluster $D \supsetneq C$ that subsumes C at $G^{T'}$, but does not subsume it anymore at G^T. If D itself was maximal at $G^{T'}$, then the algorithm should have identified D at that time. This gives us an opportunity for large savings in running time. For every threshold T', after having identified the maximal clusters at $G^{T'}$, we do not need to search the entire network for new maximal clusters at the subsequent threshold, but rather only *within* the maximal clusters of $G^{T'}$. This limits our search space and also enables faster advancement of thresholds.

In practice, our algorithm does not even search within all the maximal clusters, but rather only within the most cohesive ones. Note that the efficiency gains of this approach may come at the price of compromising the ultimate recall guarantee of the algorithm, because we may miss clusters that are subsumed by non-maximal clusters or by noncohesive clusters.

The procedure for identifying candidate clusters in weighted networks is depicted in Figure 5. Given a network G, the procedure sets a threshold T to be the minimum edge weight in G and computes the unweighted network G^T. Note that G^T has the same edges as G, except for the minimum weight edges that are eliminated. The procedure then finds the maximal clusters in G^T and adds them to the list of candidate clusters. Next, the procedure recursively searches for more clusters within the clusters of G^T whose cohesion exceeds the *cohesion threshold* β.

The first call to the procedure (i.e., with the original network G) slightly differs from subsequent recursive calls: the threshold T is set to be 0 and not the minimum edge weight. This guarantees that the first unweighted network processed is G^0, which has exactly the same edges as G.

The recursion stops when reaching a cluster C and a threshold T s.t. C^T cannot be further partitioned into sub-clusters by the procedure unweightedCRank. This means that C^T must be either a clique or a singleton, and thus C

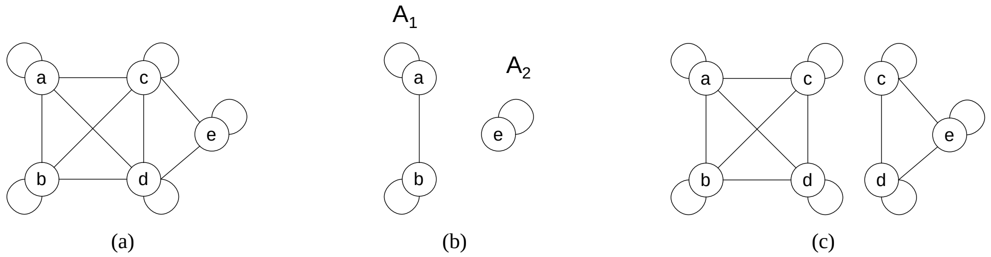


Figure 4. Identifying unweighted clusters: Example run.

```
Procedure weightedCRank($G$, $\beta$, $\mathcal{L}$)
  $T$ := minimum edge weight in $G$
  $G^T$ := unweighted network obtained from $G$ using threshold $T$
  $\mathcal{L}^T$ := an empty list of clusters
  unweightedCRank($G^T$, $\mathcal{L}^T$)
  append $\mathcal{L}^T$ to $\mathcal{L}$
  for all clusters $C \in \mathcal{L}^T$ for which $\mathrm{cohesion}(C) \geq \beta$ do
    weightedCRank($C$, $\beta$, $\mathcal{L}$)
```

Figure 5. Identifying candidate clusters in weighted networks.

is either a homogeneous clique (i.e., a clique all of whose edges are of the same weight) or a singleton.

Example run. Figure 6 shows an example run of the above procedure on a 5-node network G. The procedure applies a threshold of $T = 0$ and obtains the unweighted network G^0 depicted in Figure 6(b). The procedure then finds unweighted clusters in G^0, resulting in the clusters $\{a, b, c, d\}$ and $\{c, d, e\}$ depicted in Figure 6(c). A recursive call is made on each of these two clusters. We focus, for example, on the cluster $\{a, b, c, d\}$ (Figure 6(d)). The minimum edge weight in this cluster is 2, and thus the procedure applies a threshold $T = 2$, resulting in the unweighted network depicted in Figure 6(e). This network breaks into the two clusters $\{a, b, c\}$ and $\{d\}$. More recursive calls are made on these clusters, and we focus on the one made on $\{a, b, c\}$ (Figure 6(f)). The minimum edge weight this time is $T = 5$ and thus the resulting unweighted network is the one depicted in Figure 6(g). Note that the network now consists of singletons only, and therefore the recursion stops. The final list of clusters that will be returned is: $\{a, b, c, d, e\}$, $\{a, b, c, d\}$, $\{c, d, e\}$, $\{a, b, c\}$, $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, and $\{e\}$. Some of these clusters (namely, $\{a\}$,$\{b\}$, and $\{e\}$) will be eliminated at the third phase of C-Rank, because they are not maximal.

5.3. Candidate ranking

At its second phase, C-Rank computes the integrated cohesion of each one of the candidate clusters and ranks them accordingly. The main thing to note is that calculating the integrated cohesion of a cluster C requires computing the cohesion of C^T for k values of the threshold T, where k is the number of distinct edge weights in C. Thus, each such calculation requires at most $|E_C|$ sparsest separator calculations, giving a total of $O(|E_C| \cdot f(|C|))$ running time.

5.4. Candidate elimination

The third and last phase of C-Rank consists of eliminating non-maximal clusters from the ranked list of clusters. Testing maximality directly is hard, since to check whether a cluster C is maximal or not, we would need to compare C against all its supersets $D \supsetneq C$. Each comparison entails testing whether D subsumes C under each one of the possible thresholds T. This process requires exponential enumeration, and moreover every single subsumption test may be prohibitive, since D may have many different sparsest vertex separators.

Our candidate elimination procedure, therefore, makes two relaxations. First, each cluster C is compared not against all its possible supersets, but rather only against supersets that also belong to the list of candidates. This significantly reduces the search space and makes the enumeration only polynomial in the number of candidates.

Given a candidate cluster D that strictly contains a candidate cluster C, we do not test directly whether D subsumes C under at least one threshold T. We rather declare D as subsuming C if $\mathrm{intcohesion}(D)(1+\epsilon) \geq \mathrm{intcohesion}(C)$ (where ϵ is the maximality margin). The idea is that if D subsumes C at G^T, then D is at least (and usually more) cohesive than C in G^T. Since cohesion is monotone, D is also expected to be more cohesive than C at $G^{T'}$ for all $T' < T$. This is likely to make the integrated cohesion of D higher (or at least not much lower) than the integrated cohesion of C.

6. Experiments

Experimental setup. We tested C-Rank on our own mailboxes as well as on the Enron email data set[2], which consists of 150 mailboxes of Enron employees. The data set

[2] http://www.cs.cmu.edu/∼enron.

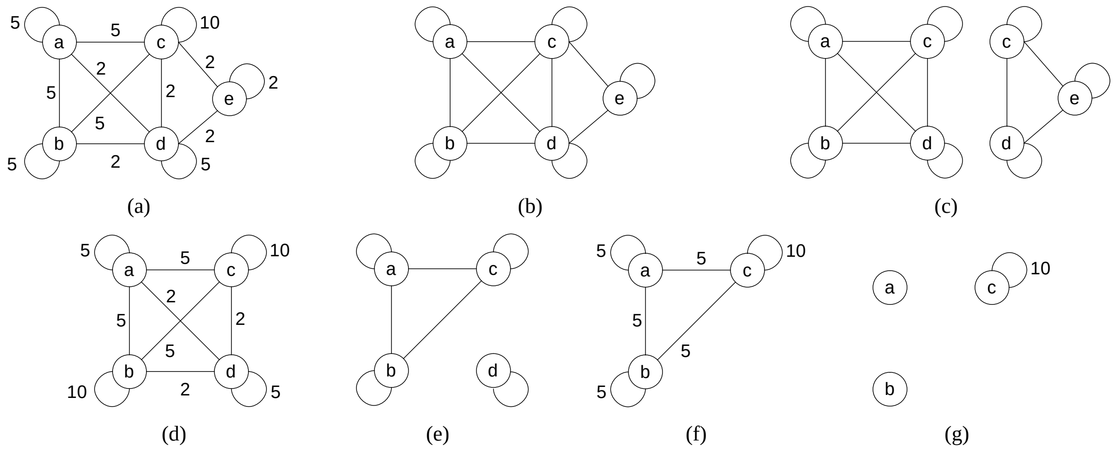


Figure 6. Identifying weighted clusters: Example run.

contains more than 500,000 messages, mostly sent along the years 2000-2002.

Given a mailbox, we constructed two corresponding networks—an *inbox network* and an *outbox network*—as follows. First, we cleaned the data, by removing duplicate messages, merging alias addresses, and ignoring messages that did not include the mailbox's owner as an explicit sender or recipient. We then split the messages into "outgoing" and "incoming". All the incoming messages were used to construct the inbox network and all the outgoing messages were used to construct the outbox network. The inbox (resp., outbox) network consists of all contacts that appear on headers of incoming (resp., outgoing) messages, excluding the mailbox's owner. Two contacts are connected by an edge if and only if they appear on at least one message header together. The weight of the edge is the number of message headers on which they co-occur. The self loop weight of a contact is the number of message headers on which it appears.

We ran C-Rank with the following parameters: (1) maximality margin $\epsilon = 0.75$; (2) cohesion threshold $\beta = 1$. In most of the experiments, we ignored the self loop weights altogether, in order to focus on the non-singleton communities, which are less trivial to find and rank.

We enforced a hard time limit of 3,600 seconds on the execution of C-Rank on each mailbox. C-Rank was unable to finish its execution on 19 of the 150 mailboxes by this time limit, and thus these mailboxes were excluded from the data set. We ran the experiments on Intel Pentium 4 2.8GHz processor workstations with 2GB of RAM.

Evaluating clustering results automatically is a difficult task. Our situation is even more complicated, because there is no benchmark cluster ranking algorithm to which we could compare C-Rank. We thus created such a benchmark from the widely used edge betweenness hierarchical clustering algorithm of Girvan and Newman [15]. (In fact, we used Newman's variant of the algorithm [25], which is adapted to weighted networks). The benchmark algorithm, which we call EB-Rank, is identical to C-Rank, except that it generates its candidate clusters using the edge betweenness algorithm. The ranking and candidate elimination phases of EB-Rank are identical to those of C-Rank.

Anecdotal results. In order to give a feel of the communities produced by C-Rank, we start with some anecdotal results from two of our mailboxes. Figure 7 shows the top 10 non-singleton communities in the inbox of Ziv Bar-Yossef. The example demonstrates that the strong communities output by the algorithm are indeed meaningful, as the owner could easily attach a title to each one of them. This list consists of few overlapping communities, since Ziv's research projects tend to be separated and have very few common participants.

Rank	Weight	Size	Member IDs	Description
1	163	2	1,2	grad student + co-advisor
2	41	17	3-19	FOCS program committee
3	39.2	5	20,21,22,23,24	old car pool
4	28.5	6	20,21,22,23,24,25	new car pool
5	28	2	26,27	colleagues
6	28	2	28,29	colleagues
7	25	3	26,30,31	colleagues
8	19	3	32,33,34	department committee
9	15.9	19	35-53	jokes forwarding group
10	15	14	54-67	reading group

Figure 7. Ziv Bar-Yossef's top 10 communities.

Figure 8 shows the top 10 communities output for the inbox of Ido Guy, including singleton communities. This example demonstrates that singleton communities can blend well with non-singleton communities and that they do not necessarily dominate the list of strong communities. In fact, Ido's list is quite diverse in terms of community sizes, ranging from singletons to groups of over 10 participants. The workplace-related communities are highly overlapping, corresponding to different projects with overlapping teams or to different sub-groups within the same project.

Rank	Weight	Size	Member IDs	Description
1	184	2	1,2	project1 core team
2	87	1	3	spouse
3	75	1	4	advisor
4	70.3	4	1,5,6,7	project2 core team
5	62	1	8	former advisor
6	48.2	6	1,2,9,10,11,12	project1 new team
7	46.9	13	13-25	academic course staff
8	46.7	9	1,5,6,7,26-30	project2 extended team (IBM)
9	42.3	5	1,2,9,10,31	project1 old team
10	41.3	13	1,5,6,7,26-30,32-35	project2 extended team (IBM+Lucent)

Figure 8. Ido Guy's top 10 communities (with singletons).

Enron data set statistics. Next, we present some statistical data about the results of C-Rank on the 131 mailboxes of the Enron data set. We first addressed the issue of the prominence of "singletons" in the data set. A "singleton message" is one that has only one sender or one recipient, apart from the mailbox's owner. Such a message contributes only to the self loop weight of the corresponding sender/recipient. In the examined Enron data set, about 80% of the outgoing messages and 50% of the incoming messages, regardless of the mailbox size, were singletons. This huge density of singleton messages necessarily affected also the results of C-Rank. Indeed, when taking into account self loops, 70% to 90% of the outbox communities and 20% to 65% of the inbox communities detected by C-Rank were singleton communities. We conclude that the high density of singleton communities should be attributed to the nature of the data set, rather than to biases of C-Rank. Since singletons are easy to handle separately, in the rest of our experiments, we eliminated the self loops from the network, and thus focused only on the analysis of non-singleton communities.

Figure 9 depicts the distribution of community sizes output by C-Rank. For each mailbox, we ordered all the output communities by their size, split them into 10 deciles, and calculated the median community size in each decile. We then plotted for each decile the median of these median values, over all mailboxes. The results demonstrate that C-Rank is not biased towards small communities, as one may suspect initially. The median community size at the top decile, for example, was about 20 contacts!

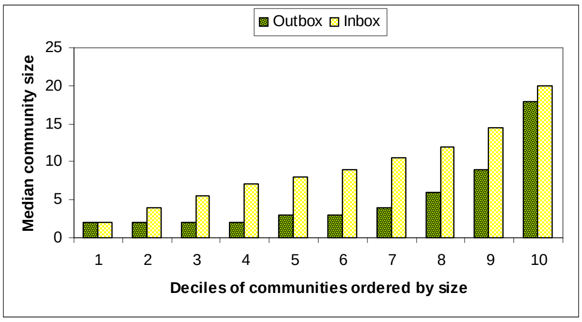


Figure 9. Distribution of community sizes.

Comparison with EB-Rank. We now describe a set of experiments that compare the results of C-Rank with the results of EB-Rank (the edge betweenness based algorithm) on the Enron data set. Figure 10 compares the relative recall of C-Rank and EB-Rank. For each mailbox, we calculated the recall of algorithm A relative to algorithm B as follows. We compared the two lists $\mathcal{L}_A$ and $\mathcal{L}_B$ of communities output by A and by B, respectively, when running on this mailbox. Intuitively, the recall of A relative to B should be the fraction of the communities in $\mathcal{L}_B$ that also appear in $\mathcal{L}_A$. However, even when A and B detect the same community, they may have slightly different "versions" of that community, differing in a few nodes. Therefore, when searching the list $\mathcal{L}_A$ for a community C that shows up on the list $\mathcal{L}_B$, we did not look for an exact copy of C, but rather for a community C' that is "comparable" to C. Formally, we say that C' is *comparable* to C, if $C' \supseteq C$ and $\mathrm{intcohesion}(C')(1+\epsilon) \geq \mathrm{intcohesion}(C)$, where ϵ is the maximality margin. The recall of A relative to B on the specific mailbox was then calculated as the fraction of the communities in $\mathcal{L}_B$, for which we found a comparable community in $\mathcal{L}_A$.

After calculating the recall for each mailbox, we ordered the networks by their size, split into 10 deciles, and plotted the median recall at each decile. The results prove that the recall of C-Rank relative to EB-Rank is significantly higher than the recall of EB-Rank relative to C-Rank. The difference even becomes higher for larger networks. This experiment underscores the advantage of overlapping clustering over partitional clustering, at least in this application domain.

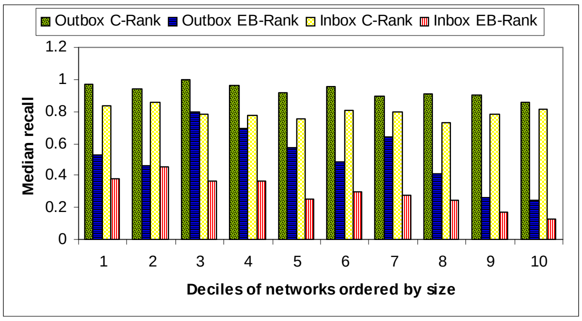


Figure 10. Relative recall of C-Rank and EB-Rank.

The previous experiment showed that C-Rank is much more successful than EB-Rank in detecting many maximal communities. However, is it possible that these extra communities are all weak, and if we focus only on the strong communities then the two algorithms are comparable? In order to explore this possibility, we compared the strength scores of the communities output by the two algorithms. For each mailbox and for each $k = 5, 10, 15, \ldots, m$, where m is the minimum number of communities output by the two algorithms on this mailbox, we calculated the median integrated cohesion of the top k communities on each of the two output lists. For each k and for each algorithm, we then plotted the median score over all networks for which $m \geq k$. These results indicate that C-Rank not only finds more maximal communities overall, but also finds *better* communities. This phenomenon is consistent across inbox and outbox and across different values of k.

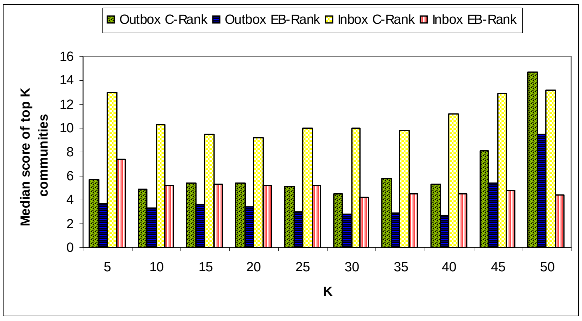


Figure 11. Distribution of community scores.

In Figure 12 we compare the precisions of C-Rank and EB-Rank. Precision was calculated as follows. For each mailbox, we compared the number of communities eventually output by the algorithm (after elimination of non-maximal communities) to the number of communities identified at the candidate identification phase. This ratio was assumed to represent the precision of the algorithm on this mailbox. We then ordered the networks by size, and split them into 10 deciles. We plotted the median precision of the algorithm in each decile. The results shown in the graph demonstrate that precision goes down with network size. The explanation is quite simple: large networks tend to be richer in complex community patterns, and "onions" (see Section 4.3) in particular. Such patterns give rise to a large number of non-maximal communities, some of which are selected in the first phase of the algorithm. Most of these communities are filtered at the elimination phase of the algorithm. Surprisingly, although C-Rank has higher recall than EB-Rank, its precision is comparable and even better than that of EB-Rank.

Robustness experiments. One indication of a good clustering algorithm is that it is robust to small changes in the data. In order to test the robustness of C-Rank, we compared the communities it output when running over on the entire Enron data set to the communities it output when running over a sample of the data. For this experiment, we focused only on sufficiently large mailboxes: ones in which the number of messages was at least 500. Overall, we used 36 outboxes and 41 inboxes in this experiment. For each such mailbox, we constructed 3 networks: one that was constructed using all the messages in the mailbox, one that was constructed using 80% randomly chosen messages from the mailbox, and one that was constructed using 20% randomly chosen messages from the mailbox. (The latter two networks were constructed 5 times each, and the results presented here are the medians over these 5 trials.) For each of the two latter networks, and for each $p = 10\%, 20\%, \ldots, 100\%$, we calculated the recall of the top $k = p \cdot m$ communities output by C-Rank on this network (where m is the total number of communities output on this network) relative to the top k communities output by C-Rank when running on the first, complete, network. For each p, we then calculated the median recall over all networks. This value, which we call "recall@p", captures how well C-Rank was able to detect the strong communities of the mailbox, when running over only a portion of the data in the mailbox.

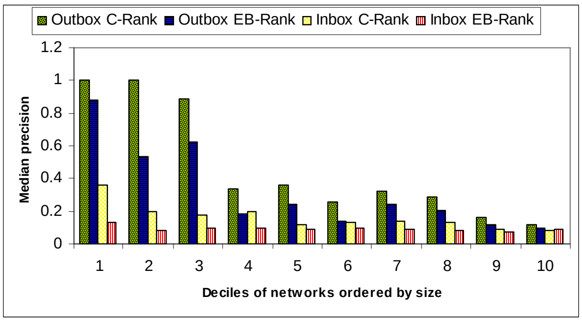


Figure 12. Precision of C-Rank and EB-Rank.

The results indicate that C-Rank is rather resilient to random removal of data. On the networks built over 80% of the data, C-Rank was able to maintain a recall of about 90% across all values of p. When running on a mere 20% of the data, C-Rank was still able to maintain reasonable recall of 45% at the top decile and 20% at the bottom decile.

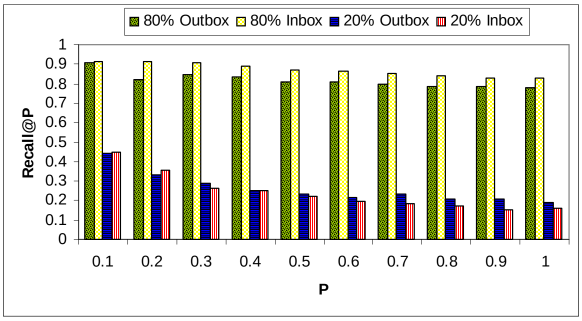


Figure 13. Recall of C-Rank on sampled data.

To sum up, we believe that the above experiments provide convincing evidence that C-Rank is able to achieve high recall values (i.e., covering many of the maximal clusters in the network), while maintaining a relatively high precision. C-Rank is completely superior to EB-Rank, which is based on the very popular edge betweenness clustering algorithm. C-Rank is also robust to random removal of data, attesting to its quality.

7. Conclusions

We presented the cluster ranking problem as a novel framework for clustering. We then proposed integrated cohesion as a new strength measure for clusters. We designed C-Rank: a cluster ranking algorithm that detects and ranks overlapping clusters in arbitrary weighted networks. We demonstrated the effectiveness of C-Rank by ranking clusters in egocentric mailbox networks. Future work will aim at applying the new framework in other domains.

Acknowledgments. We thank Gail Gilboa-Freedman, Nili Ifergan, Idit Keidar, Natalia Marmasse, Elad Shahar, Uzi Shvadron, Eyal Sonsino, and Gala Yadgar for sharing with us their mailboxes. We thank Ran El-Yaniv and Ran Wolff for very helpful suggestions.

References

[1] E. Amir, R. Krauthgamer, and S. Rao. Constant factor approximation of vertex-cuts in planar graphs. In *Proc. 35th STOC*, pages 90–99, 2003.

[2] A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R. J. Mooney. Model-based overlapping clustering. In *Proc. 11th KDD*, pages 532–537, 2005.

[3] J. D. Banfield and A. E. Raftery. Model-based guassian and non-gaussian clustering. *Biometrics*, 49:803–821, 1993.

[4] J. Baumes, M. K. Goldberg, and M. Magdon-Ismail. Efficient identification of overlapping communities. In *Proc. ISI*, pages 27–36, 2005.

[5] P. O. Boykin and V. Roychowdhury. Personal email networks: An effective anti-spam tool. *IEEE Comp.*, 38:61–68, 2005.

[6] T. N. Bui and C. Jones. Finding good approximate vertex and edge partitions is NP-hard. *Information Processing Letters*, 42:153–159, 1992.

[7] G. Cleuziou, L. Martin, and C. Vrain. PoBOC: An overlapping clustering algorithm, application to rule-based classification and textual data. In *16th ECAI*, pages 440–444, 2004.

[8] D. Fasulo. An analysis of recent work on clustering algorithms. Technical Report 01-03-02, Department of Computer Science and Engineering, University of Washington, 1999.

[9] U. Feige, M. Hajiaghayi, and J. R. Lee. Improved approximation algorithms for minimum-weight vertex separators. In *Proc. 37th STOC*, pages 563–572, 2005.

[10] D. Fisher. Using egocentric networks to understand communication. *IEEE Internet Computing*, 9(5):20–28, 2005.

[11] D. Fisher and P. Dourish. Social and temporal structures in everyday collaboration. In *Proc. CHI*, pages 551–558, 2004.

[12] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of Web communities. In *Proc. 6th SIGKDD*, pages 150–160, 2000.

[13] G. W. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35(3):66–71, 2002.

[14] L. C. Freeman. *The Development of Social Network Analysis: A study in the Sociology of Science*. Empirical Press, 2004.

[15] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99:7821–7826, 2002.

[16] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*. Wiley & Sons, 1999.

[17] H. Ino, M. Kudo, and A. Nakamura. Partitioning of Web graphs by community topology. In *Proc. 14th WWW*, pages 661–669, 2005.

[18] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

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

[20] J. M. Kleinberg. An impossibility theorem for clustering. In *Proc. 15th NIPS*, pages 446–453, 2002.

[21] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In *Proc. ECML*, pages 217–226, 2004.

[22] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the web for emerging cyber-communities. In *Proc. 8th WWW*, pages 1481–1493, 1999.

[23] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *JACM*, 46(6):787–832, 1999.

[24] A. McCallum, A. Corrada-Emmanuel, and X. Wang. Topic and role discovery in social networks. In *Proc. 19th IJCAI*, pages 786–791, 2005.

[25] M. E. J. Newman. Analysis of weighted networks. *Phyical Review E*, 70(056131), 2004.

[26] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phyical Review E*, 69(026113), 2004.

[27] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.

[28] F. C. N. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. In *31st ACL*, pages 183–190, 1993.

[29] J. Scott. *Social Network Analysis: A Handbook*. Sage, 1991.

[30] E. Segal, A. Battle, and D. Koller. Decomposing gene expression into cellular processes. In *Proc. 8th PSB*, pages 89–100, 2003.

[31] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[32] N. Slonim. *The Information Bottleneck: Theory and Applications*. PhD thesis, Hebrew University, 2002.

[33] J. Tyler, D. Wilkinson, and B. A. Huberman. Email as spectroscopy: Automated discovery of community structure within organizations. In *Proc. 1st C&T*, pages 81–96, 2003.

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

[35] B. Wellman. An egocentric network tale. *Social Networks*, 15:423–436, 1993.