Community Finding in Large Social Networks Through Problem Decomposition*

Anand Narasimhamurthy, Derek Greene, Neil Hurley, and Pádraig Cunningham

School of Computer Science and Informatics, University College Dublin
{anand.narasimhamurthy,derek.greene,neil.hurley,padraig.cunningham}@ucd.ie

University College Dublin
Technical Report UCD-CSI-2008-04
August 2008

Abstract. The identification of cohesive communities is a key process in social network analysis. However, the algorithms that are effective for finding communities do not scale well to very large problems, as their time complexity is worse than linear in the number of edges in the graph. This is an important issue for those interested in applying social network analysis techniques to very large networks, such as networks of mobile phone subscribers. In this respect the contributions of this report are two-fold. First we demonstrate these scaling issues using a prominent community-finding algorithm as a case study. We then show that a two-stage process, whereby the network is first decomposed into manageable subnetworks using a multilevel graph partitioning procedure, is effective in finding communities in networks with more than 10^6 nodes.

1 Introduction

After several years of academic research on social network analysis (SNA), considerable commercial interest in exploiting SNA has recently emerged. The research reported in this work is motivated by the prospect of using SNA in large-scale applications, such as exploring online communities [1], mining web usage data [2], identifying research trends in bibliographic networks [3], and analysing relationships among mobile telephone subscribers. The identification of cliques or communities in network data has received a lot of attention in SNA research, where a number of promising techniques have emerged [4–6]. These techniques can identify subgraphs such that the density of edges within the subgraph is greater than the density of edges connecting the subgraph to the rest of the network. They can also identify overlapping communities, where an individual can belong to more than one group. This is an important facility for network analysis in many real-world domains.

* The work was supported by Science Foundation Ireland Grant No. 05/IN.1/I24 and Enterprise Ireland Grant No. PC/2007/010

Another highly significant issue in this area is that of scalability. Unfortunately, it is in the nature of the analysis entailed in many community-finding algorithms that they do not scale well to very large graphs. In fact, existing techniques in this area generally cannot handle graphs with more than a few tens of thousands of nodes (*e.g.* [6]). In contrast, many real-world networks will be substantially larger. For instance, in a study reported by Abello *et al.* [7], a one-day telephone call graph at AT&T consisted of 53,767,087 vertices and more than 170 million edges.

To deal with this issue of scalability, we propose a pragmatic problem decomposition strategy: use a "top-down" graph partitioning technique to decompose the network into smaller subnetworks, on which it is then feasible to apply a more computationally intensive community-finding algorithm. To perform the initial partitioning, we use the multilevel method proposed by Dhillon *et al.* [8], referred to as Graclus. For the second stage of the process, the CFinder algorithm [5] is employed to discover small, overlapping communities. Two evaluations are presented in this report to demonstrate the effectiveness of the proposed two-stage strategy, covering two key aspects of the community-finding problem:

1. **Scalability:** The two stage strategy is evaluated on synthetic data to demonstrate that it results in a significant reduction in running time, making it possible to discover communities in graphs much larger than can be tackled by a community finding algorithm operating on its own.
2. **Solution quality:** The evaluation also demonstrates that the two stage process produces good solutions – this is assessed according to two related criteria:
   – The extent to which graph partitioning preserves communities intact when applied to a large graph.
   – The agreement between the communities discovered when the community finding algorithm is run on the entire graph, and those discovered when using the subgraphs obtained via graph partitioning.

We perform evaluations on networks with community structures similar to those occurring in large-scale, real-world networks such as those pertaining to mobile phone subscribers – *i.e.* small, dense, localised communities embedded in a very large sparse graph. We show that we can discover communities in these networks in a computationally efficient manner, without adversely affecting the "quality" of these communities.

The remainder of this report is organised as follows. The next section provides a summary of existing methods, specifically community-finding algorithms and top-down network partitioning algorithms, which are relevant in SNA. We summarise our proposed problem decomposition approach in Section 3. The datasets used in our experiments are presented in Section 4, and our experimental results are subsequently discussed in Section 5. The report finishes with some conclusions in Section 6.

2 Related Work

2.1 Community-Finding Algorithms

Before discussing recent work in the area of SNA, it is important to say something about the terms "clique" and "community" that pervade this discussion. A "clique" has a formal meaning in graph theory – it refers to a set of nodes that are fully connected (*i.e.* an edge exists between all pairs of nodes). Although the problem of finding a maximum clique in a graph is NP-hard [9], a number of clique-finding techniques employing a variety of heuristics have been proposed in the graph theory literature [10–13]. Most would agree that this definition of a clique is too strict for SNA, as "communities" of dense connections will be of interest, even if not all pairs of nodes are connected. If the objective is not to discover maximum cliques but instead to discover *dense* structure in the network, then the question will arise of which criterion to optimise when searching for dense structures. It is interesting to see how this issue has been addressed by various authors in the SNA literature. Additionally, it is important to note that techniques for optimising an appropriate *connectedness* criterion often have poor time complexity, thus limiting their usefulness when working with very large graphs. We now outline three prominent algorithms that have been employed for community-finding in SNA tasks.

Newman & Girvan's method (GN). This is essentially a top-down hierarchical clustering strategy, which employs a novel partitioning criterion. This criterion is based on the *shortest path betweenness* measure [14], which is well-established as a measure of node centrality in SNA. In the GN algorithm, the "edge betweenness" refers to the number of shortest paths in a network that pass along an edge. Edges that score highly on this criterion are likely to be inter-cluster edges (*i.e.* edges that link adjacent clusters). It may seem surprising that a criterion for identifying cluster centrality can also identify links between adjacent clusters when applied to edges rather than nodes. In fact, these edges are *weak ties* in the sense of this term introduced by Granovetter [15]. That is, while they are important links connecting individuals across the network, they represent associations between casual acquaintances rather than close friends. The GN approach works by generating a successive partitioning of a graph, cutting the edge with the highest edge betweenness score at each stage. Clearly this will be computationally expensive, given the number of shortest-path calculations that must be performed. Newman & Girvan say the time complexity of this algorithm is $O(n^3)$ on sparse graphs, making it applicable to graphs of up to 10,000 nodes at the time of publication in 2003.

CONGA. A related community-finding approach from the recent literature that can deal with overlapping communities is the CONGA algorithm, proposed by Gregory [6]. Since it is based on the GN algorithm described above [16], this approach also employs a divisive hierarchical partitioning strategy. In order to allow nodes to belong to more than one cluster, CONGA permits nodes to be split, i.e. a real node v will be split into $\{v_1, v_2\}$ with the incoming edges to v divided between v_1 and v_2, and a new *virtual* edge created to link v_1 and v_2. If this new virtual edge has a higher edge betweenness than any real edge, then the node should be split on this basis. The enhancement that CONGA brings to the GN algorithm is to add this node-splitting step as an extension to the partitioning phase of the algorithm. Gregory states that the worst case time complexity of this algorithm is $O(e^3)$, where e is the number of edges. In practice CONGA has been shown to handle networks of up to 4,000 nodes and 7,000 edges.

CFinder. In contrast to the two previous algorithms which operate in a top-down manner, CFinder [5] is a bottom-up technique that uses cliques as building blocks for large groups. The algorithm first extracts all maximal complete subgraphs (*i.e.* cliques) that do not form part of larger complete subgraphs, and composes these into larger structures. Although an efficient approach for finding all cliques in a general network may not be feasible since determining a maximum clique is NP-hard, Palla *et al.* nevertheless claim that their approach performs well on real networks. From these cliques, all k-cliques are enumerated (*i.e.* complete subgraph of size k). The algorithm then proceeds to find *k-clique-communities*, which are defined as the union of all k-cliques that can be reached from each other through a series of adjacent k-cliques. This approach can discover overlapping and nested communities. Palla *et al.* point out that, given the nature of the algorithm, it is difficult to analyse the time complexity of CFinder.

2.2 Graph Partitioning

It is not always easy to distinguish between the community-finding algorithms reviewed in the previous section, and the graph partitioning algorithms discussed here. However, we can make several broad distinctions. Notably, community-finding algorithms are designed to find regions of dense structure that are not well-connected with the rest of the network. These regions will often exhibit a certain degree of overlap. In contrast, the graph partitioning problem involves finding the optimal division of a graph into k disjoint parts according to a chosen criterion, such that the partition covers the entire graph. Common applications for graph partitioning algorithms include VLSI module placement, load balancing for parallel processing, and image segmentation [17].

Of the vast array of approaches that have been proposed in the literature, two of the most important classes are spectral clustering and multilevel partitioning. Spectral methods produce a partition based on the eigendecomposition of the graph. Usually the Laplacian matrix is used in this context, rather than the original adjacency matrix. The reader is referred to [18] for a detailed discussion on the Laplacian matrix, and the connection between the eigenvalues of the Laplacian with many key invariants of the graph. Spectral approximations for a variety of partitioning criteria have been formulated, including the minimum cut [19], ratio cut [20], and normalised cut [17].

Many multilevel approaches for graph partitioning have been developed over the years, the Metis algorithm [21] being the most well-known example. In these approaches a sequence of successively smaller, coarser hypergraphs is constructed. A bisection of the smallest hypergraph is computed and is successively projected to the next finest level. At each level, an iterative refinement algorithm is used to improve the bisection. Most multilevel algorithms are based on the seminal work by Kernighan & Lin [22]. Given an edge weighted graph $G = (V, E)$, and an initial partitioning of the nodes $V = A \cup B$, the algorithm proceeds by finding equal-sized subsets of nodes $X \in A$ and $Y \in B$, such that exchanging X and Y reduces the total cost of edges between the old partitions A and B. Previously it was assumed that the initial partitions would be of equal sizes. However, a number of variants of the original algorithm have been proposed which support unbalanced clusters.

Recently, Dhillon *et al.* [8] developed a fast multi-level algorithm that directly optimises various weighted graph clustering objectives. The authors show that a general weighted k-means objective is mathematically equivalent to a weighted graph clustering objective, and they exploit this equivalence. The main advantage of their method is that it approximates graph clustering objectives without requiring an eigendecomposition, which can be computationally intensive for large graphs. Another advantage of this algorithm compared to other multilevel approaches is that it does not require the partitions to be of equal sizes.

3 Problem Decomposition Strategy

While community-finding algorithms have been aimed at SNA applications, these techniques are computationally expensive and the communities they discover are small – normally comprising some tens of nodes. When dealing with very large graphs, which may potentially contain hundreds of thousands or even millions of nodes, algorithms with running time $O(e^2)$ or $O(e^3)$ will not be practical. Therefore the question arises, how can we find small communities in such large networks?

The approach we take in this work is to employ a two-stage problem decomposition strategy as outlined below:

Stage 1: Given a large graph, apply a computationally tractable graph partitioning algorithm that minimises the number of links broken in the process. The resulting partitions will themselves form large subgraphs.

Stage 2: Once the original graph has been split into manageable subgraphs, apply a more computationally intensive community-finding algorithm to each subgraph, and subsequently combine the results.

For the first stage of the process, we suggest the use of the implementation of the multilevel partitioning method proposed by Dhillon *et al.* [8], referred to as Graclus[1]. We choose this approach as it allows us to optimise the objectives for

[1] Available from `http://www.cs.utexas.edu/users/dml/Software/graclus.html`

spectral clustering, which have previously proved successful in other areas [17], but without requiring a costly eigendecomposition. Thus Graclus will provide a partitioning of a large graph with a small running time. For the second stage of the process, we employ the CFinder algorithm [5], as it supports the discovery of overlapping, localised and nested communities. It should be noted that, while we focus on the pairing of Graclus and CFinder in the remainder of this report, our proposed strategy can naturally be generalised to make use of any suitable combinations of algorithms.

4 Experimental Datasets

Two sets of experiments were performed, one using synthetic data, and the other using real data. Synthetic data was used for two main purposes. Firstly, we wished to examine how community-finding techniques scale to large, sparse graphs, approaching the size of those occurring in real-world applications, such as the analysis of mobile subscriber networks. For these scalability experiments synthetic graphs consisting of 1k, 2k, 5k, 10k, 100k, and one million nodes were used. Secondly, we wished to assess the extent to which graph partitioning techniques preserve communities intact. In addition to using synthetic data, we also performed experiments on real data. Graphs derived from the CORA bibliographic network [23] were used for this purpose.

4.1 Synthetic data

We performed experiments on two sets of synthetically generated graphs[2]. The generation and structure of these graphs is now described in detail.

Power law graphs. An important characteristic of networks is the distribution of edges. The most straightforward assumption about the distribution of edges is that all possible edges between the vertices are equally probable [24]. This defines a random graph where the degree of each node follows a Poisson distribution. It has been shown that for many real graphs the degree distribution follows a power law rather than a Poisson distribution [25]. An example of the distinction between the distributions is given in Figure 1. The important point about power law graphs is the increased probability of nodes with large numbers of edges. These nodes represent *hubs* which have a significant impact on the overall connectivity of the graph, since a connected set of hubs can provide short paths between a large proportion of the nodes in the network. Thus the average distance between vertices in a power law graph will be small. Power-law graphs also tend to have a high cluster coefficient and exhibit scale-free characteristics, so the plot of degree distribution is self-similar at different levels of resolution.

The degree sequence of a graph is the set of node degrees $(d_1, \dots d_n)$, ordered such that $d_1 \geq \cdots \geq d_n$. To construct synthetic graphs that approximate real-world power-law graphs, we generated random graphs with a given fixed input

[2] Available for download at `http://mlg.ucd.ie/datasets/graph.html`

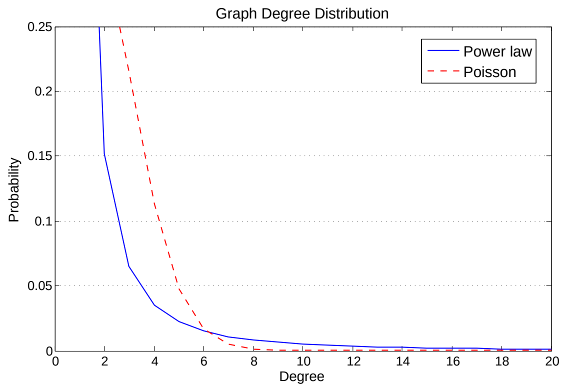


Fig. 1. A comparison of degree distribution for random and power law graphs - a common model for degree distribution in random graphs is a Poisson distribution.

degree sequence, and chose the degree sequence to follow an exact power-law. That is, for $k = 1 \ldots n$, we choose d_k such that

$$d_k = \text{round}(ck^{\alpha}) \, ,$$

for chosen constants c and α. In particular, we select a reasonable value of α based on values reported from real-world graphs. The constant of proportionality c, was set by specifying the minimum degree $d_n = d_{\min}$, so that

$$d_k = \text{round}\left(\left(\frac{k}{n}\right)^{\alpha} d_{\min}\right)$$

Note that, provided the degree sequence is graphical, it is possible to generate a graph corresponding to that degree sequence by applying the constructive method proposed in [26]. Real-world graphs do not exhibit the perfect power-law characteristics of these synthetic graphs, but rather tend to exhibit a power-law behaviour in certain ranges of the node degrees. Nevertheless, this is a useful approximation that allows us to examine algorithms on large-scale graphs which exhibit some of the characteristics that can be found in real-world graphs. In the experiments described later in Section 5, we use graphs with $d_{\min} = 8$ and $\alpha = 10$.

Graphs with embedded communities. An important aspect of our evaluations was to compare the groups discovered by the community finding algorithm to the actual structures present in the data, and to examine the extent to which the prior application of graph partitioning affects the discovery of these structures. For this purpose, we generated graphs with embedded communities, where these communities represent a “ground truth”. To achieve this, we followed a procedure similar to that outlined by Newman & Girvan [4]. The basic idea is that nodes in each of the communities acquire edges among other members of the same community, as well as nodes outside the community, with certain probabilities. These probabilities are defined in terms of two key parameters:

– Let p_{in} denote the probability with which a node in a community acquires edges with other members of the same community.
– Let p_{out} denote the probability with which a node acquires edges randomly with the rest of the nodes in the graph.

When constructing artificial graphs, values for these parameters can be reached in a number of ways. For instance, Newman & Girvan [4] fix the expected degree of each vertex, and then compute values for p_{in} and p_{out} accordingly.

While cliques and variants such as k-plexes are precisely defined, as noted previously, the term "community" is not clearly defined in the literature. In most cases, the term refers to a group of nodes which have a high degree of connectivity between themselves, relative to the rest of the network. In order to capture this notion of a community, we introduce the measure *assoc*, which quantifies the connectivity between a community C and a graph G, where $C \subset G$,

$$assoc(C,G) = \sum_{u \in C, v \in G, u \neq v} w(u,v) \tag{1}$$

and where $w(u,v)$ is the weight of the edge connecting a pair of nodes u and v. Note that in the data described here, edge weights are either 1 or 0. In this case $assoc(C,C)$ is a count of the number of edges connecting nodes within C, and we can use $p_{in}(C)$ as a quality measure where

$$p_{in}(C) = \frac{assoc(C,C)}{c \times (c-1)/2} \tag{2}$$

and $c = |C|$. Another measure of community quality is the ratio $r(C)$ of the within-community connectivity in C to the connectivity of the community's nodes to the entire graph:

$$r(C) = \frac{assoc(C,C)}{assoc(C,G)} \tag{3}$$

To construct the graphs used in our experiments, we fix the values for the two measures $p_{in}(C)$ and $r(C)$ for all communities – thus the graphs are defined by the parameters p_{in} and r. From these values, we can compute the other parameter p_{out} used by Newman & Girvan [4]. We chose to fix p_{in} as opposed to p_{out}, since it has a straightforward interpretation – it is a measure of how close the community is to forming a maximal clique. For instance, p_{in}=1 corresponds to a fully connected clique. It is worth noting that the parameter p_{in} is closely related to the *clustering coefficient* defined by Watts & Strogatz [27]. In fact, it is equivalent to the clustering coefficient of the subgraph obtained by considering only the nodes of the particular community and their connecting edges.

For evaluation purposes, we constructed "unit" graphs consisting of 1,000 nodes, containing embedded communities of different sizes, ranging from 10 to

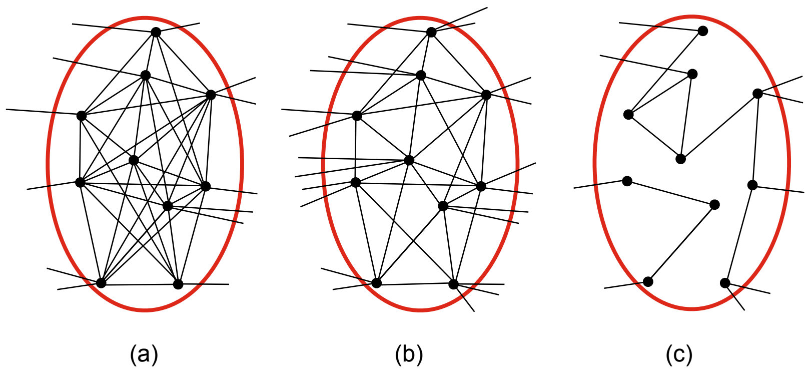


Fig. 2. This figure shows three example communities of 10 nodes in the synthetic data with different parameter values: in (a) $p_{in} = 0.7, r = 0.7$, in (b) $p_{in} = 0.5, r = 0.5$, in (c) $p_{in} = 0.2, r = 0.5$. For instance, in example (c) 20% of the possible links in the community exist, and half of all links are internal to the community.

40 nodes. These communities were non-overlapping, and all nodes were assigned to a community. Successively larger graphs were then generated such that the number of nodes and communities were scaled by the same integral ratio. For example, a graph of 2,000 nodes had exactly twice the number of communities of the same respective sizes as a graph of 1,000 nodes.

In order to better understand the nature of the synthetic data, some example communities consisting of ten nodes, but of different link densities, are shown in Figure 2. In Figure 2(a) a community of ten nodes with $p_{in} = 0.7$ and $r = 0.7$ is shown. At this density, on average 70% of all possible within-community edges exist, and 70% of all edges associated with the community are internal. In the evaluation in Section 5.2 it is clear that the community finding algorithms perform well when the community "signature" is this pronounced. In the scenario in Figure 2(b) the community signature is weaker with just 50% of possible internal links present and as many external as internal links in existence. The evaluation in Section 5.2 still shows good performance on this data. The situation in Figure 2(c) is effectively a pathological situation, as only 20% of all possible internal links are present. In this situation the community does not even form a single connected component. Perhaps unsurprisingly, most community finding algorithms will perform poorly in this context.

4.2 Real data

The CORA bibliographic dataset [23] contains information and annotations such as authors, cited papers and topic for over 50,000 research papers[3]. For our evaluation, the paper citation graph derived from the CORA dataset was used. Only

[3] See `http://www.cs.umass.edu/~mccallum/code-data.html`

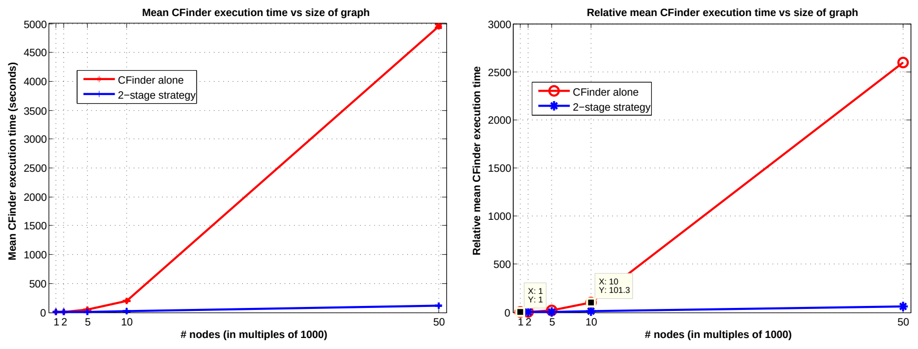
the papers with available authors were selected, where the total number of these was 28,400. We suggest that this graph is a reasonable proxy for real-world data, such as mobile subscriber networks, since the communities are small despite the scale of the graph – typically containing 10 to 20 members. The edge directions and weights were ignored, and the largest weakly-connected component was used for the experiments described in the next section. This resulted in a graph consisting of 24,542 nodes.

5 Experimental Evaluation

The evaluation presented in this section has two objectives: to highlight the scalability advantages of the two stage process, and to examine how the prior application of graph partitioning affects our ability to locate communities in a large graph. Additionally we compare the communities obtained from the entire graph with those discovered from the individual subgraphs obtained via graph partitioning. These evaluations on synthetic data are presented in Section 5.1 and Section 5.2 respectively, where we use both the power-law and embedded community graphs described in Section 4. Since the results were generally similar across a range of synthetic graph parameters, we discuss only representative results. The results of evaluations on real data are then presented in Section 5.3.

5.1 Scalability: Synthetic data

Our scalability analysis entails comparing the running times of the two-stage process and CFinder alone on graphs generated as described in Section 4. In this analysis the graph partitioning stage is set to yield sub-graphs of about 1,000 nodes, *i.e.* a graph of 10,000 nodes is divided into 10 sub-graphs. The results for the power law graphs are shown in Figure 3, and those for graphs with embedded



(a) (b)

Fig. 3. CFinder execution times (in seconds) versus graph size for synthetically generated power law graphs: (a) actual execution time, (b) execution time relative to 1000 node graphs.

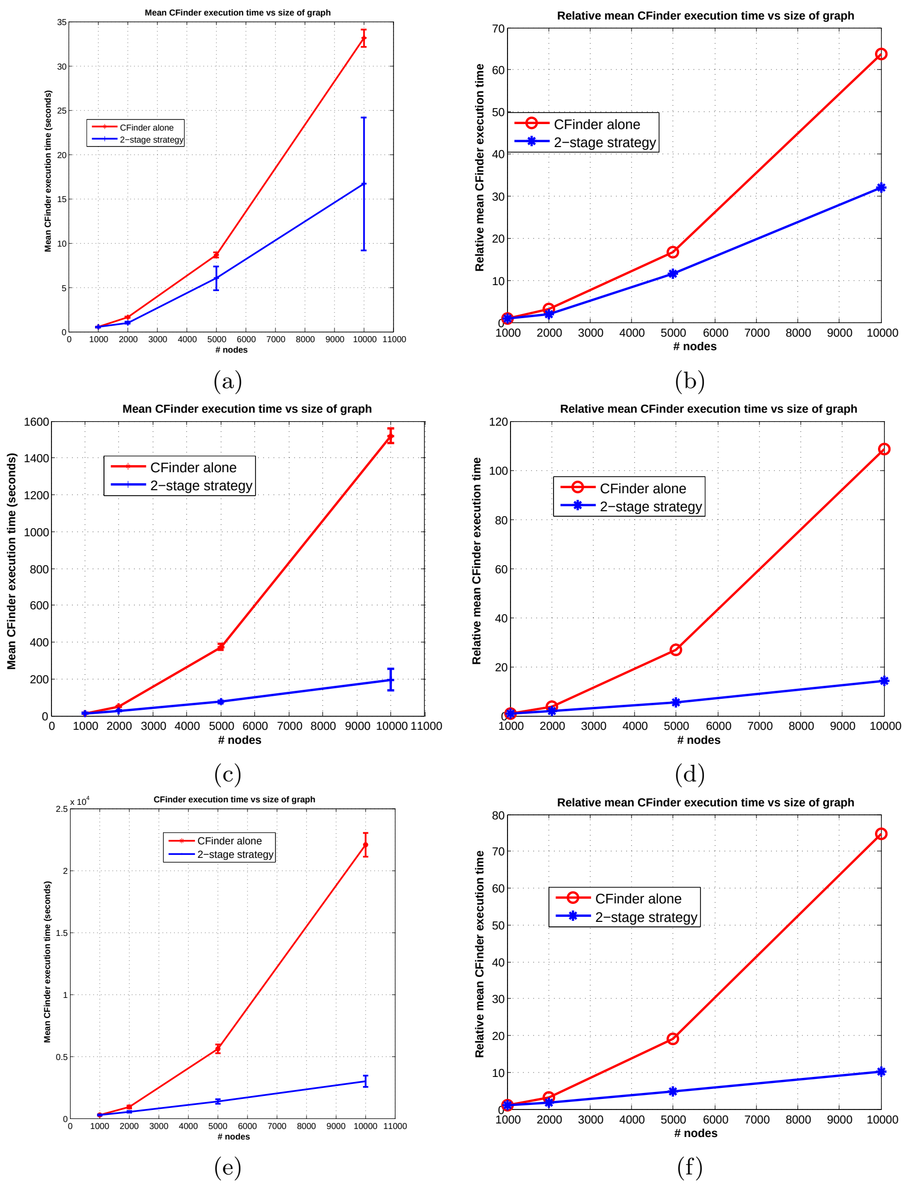


Fig. 4. CFinder execution times versus graph size for graphs with embedded communities. Actual execution times are shown in (a,c,e) while relative execution times relative to a 1,000 node graph are shown in (b,d,f). The parameters for the graphs are $(p_{in}, r) = (0.2, 0.5)$ in (a,b), $(p_{in}, r) = (0.5, 0.5)$ in (c,d), $(p_{in}, r) = (0.7, 0.7)$ in (e,f).

communities are shown in Figure 4. The results shown in Figure 4 correspond to parameter sets $(p_{in}, r) = (0.2, 0.5)$, $(p_{in}, r) = (0.5, 0.5)$ and $(p_{in}, r) = (0.7, 0.7)$.

These parameter values correspond to community signatures of the type shown in Figure 2. The results show average execution times computed over 50 runs.

In these figures, both the actual and relative execution times are shown. The graphs showing the actual execution times are included to illustrate the dependence of the CFinder execution time on the link density. It is clear that execution time increases steeply as more links are added to the graph. For both the power law and embedded community graphs, we observe a steep increase in the execution times as the graphs grow in size even though the average "community size" is the same. For example, we can see from Figure 3(b) that the average execution time for a 10,000 node graph is more than 100 times that for a graph of 1000 nodes. The execution time is approximately $O(n^2)$ in the number of nodes. The running times for the two-stage strategy are also shown in the graphs. It is clear that this strategy results in a dramatic reduction in overall running time. The combined running times are computed as follows. Let the graph be partitioned into k parts. The total running time on a graph may be expressed as

$$T = G_k + \sum_{i=1}^{k} C_i$$

where G_k denotes the execution time for partitioning the graph into k parts using Graclus, and C_i denotes the execution time of CFinder on the i^{th} subgraph resulting from the graph partitioning. For our datasets, the running time for Graclus was usually a fraction of a second for graphs consisting of up to a few tens of thousands of nodes. Thus the total running time was dominated by the CFinder execution times on the individual subgraphs.

In Figures 3 and 4 we have shown the combined running times for graphs containing up to a few tens of thousands of nodes. However, the problem decomposition strategy permits discovery of communities in graphs of sizes significantly larger than possible using the community finding algorithm on its own. For instance Graclus can easily handle sparse graphs up to a few million nodes. Sample execution times of two stage strategy for larger graphs are shown in Table 1. In

Graph type	# Nodes	# Partitions	CFinder execution time	
			CFinder only	2-stage
Embedded communities $(p_{in}, Nconn_{avg}) = (0.5, 0.5)$	50k	50	40604.6	1227.94
Embedded communities $(p_{in}, Nconn_{avg}) = (0.5, 0.5)$	100k	10	189022.89	29967.89
Power law $(d_{min}, \alpha) = (8, 10)$	1,000k	1000	N/A	365.73

Table 1. Comparison of sample CFinder execution times (in seconds) for large synthetic graphs. Note that N/A indicates that CFinder did not terminate within a "reasonable" period of time.

cases where the subgraphs were larger than could be handled by CFinder, we further partitioned the subgraphs. The actual running time of CFinder on the entire graph is shown where available, although in many cases CFinder either took an unreasonably long time or did not terminate at all. For our datasets we found that graphs of 10,000 nodes and $> 80,000$ edges were close to the limit of what can be handled by CFinder on a 2GHz dual core machine with 4GB RAM, running 64-bit Linux.

5.2 Solution Quality: Synthetic data

Having established that the two-stage strategy could scale for graphs of up to 1,000,000 nodes, the next objective of our evaluation was to examine the quality of the results produced by the two-stage process. The first issue was to assess the extent to which the graph partitioning stage preserves communities when applied to a large graph. The second issue was to compare the sets of communities discovered by CFinder alone to those discovered by the two-stage process.

A few sample communities discovered by the CFinder software are shown in Figure 5. As mentioned before, CFinder enumerates all small cliques (specifically all k-cliques), and uses these as "building blocks" to enumerate communities. A k-clique-community is defined as the union of all k-cliques that can be reached from each other through a series of adjacent k-cliques, making this strategy

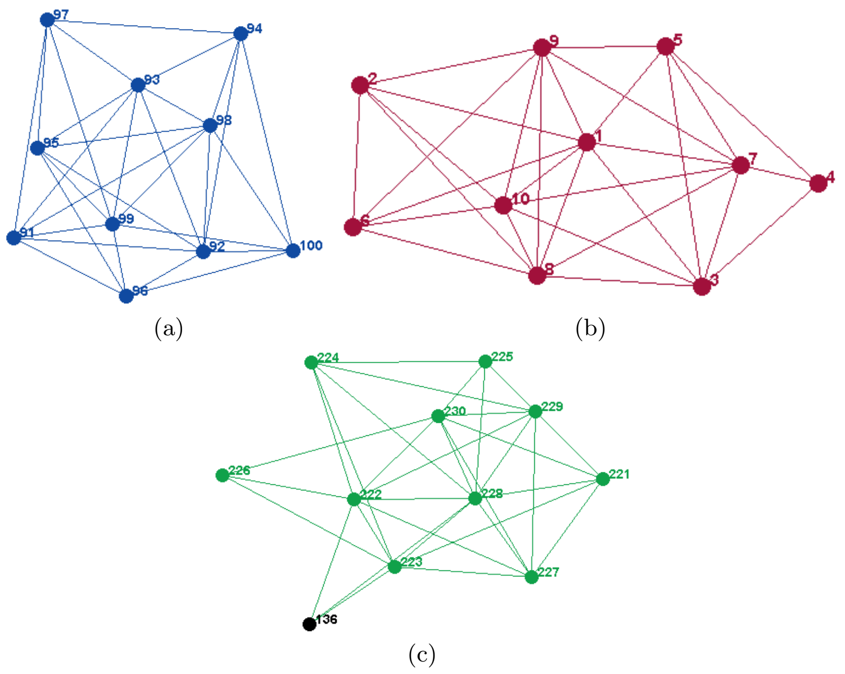
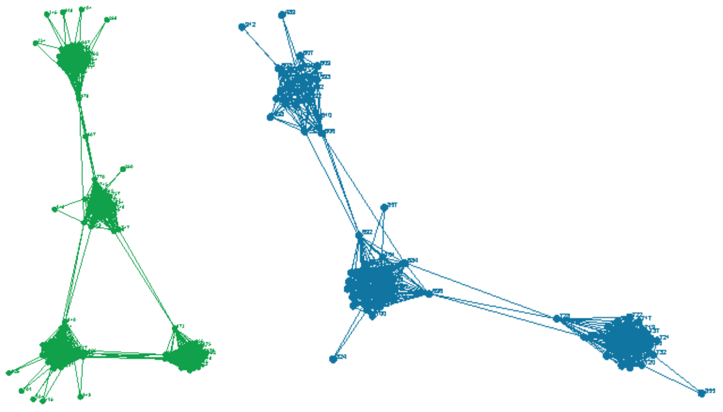


Fig. 5. Illustrative examples of smaller communities discovered by CFinder on synthetic data.



(a) (b)

Fig. 6. Illustrative examples of larger communities discovered by CFinder on synthetic data, which are comprised of smaller, interconnected communities.

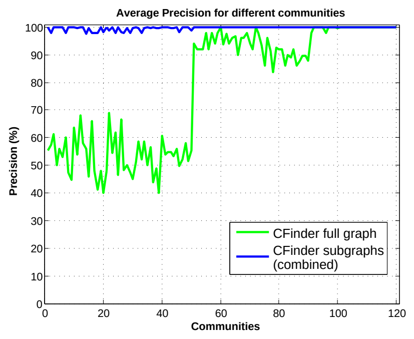
useful in discovering overlapping as well as nested communities. Thus the communities discovered by CFinder can vary in size from a few nodes (Figure 5) to tens or even hundreds of nodes (Figure 6). Hence it is necessary to compare the communities obtained by CFinder to the true communities, specifically to what extent CFinder is able to faithfully discover the true communities. As an example, consider again Figures 5(a)-(c). In Figures 5(a) and 5(b), CFinder faithfully uncovers the communities (*i.e.* only community nodes with no other extraneous ones). Figure 5(c) is a case where an extraneous node (the node labelled 136, highlighted in a different colour) is chosen along with the other correct community nodes. A different scenario is shown in Figure 6, where the large communities presumably subsume a number of smaller communities. Again it is important here to assess whether or not the true communities are faithfully recovered.

Validation measures. Given the issues raised by Figures 5 and 6, we use *precision* and *recall* as measures of cluster quality in our evaluation process. Precision can be seen as a measure of relevance – *i.e.* the fraction of results are *true positives* (although other relevant results may be missed out). Recall is a measure of completeness – *i.e.* the fraction of all relevant results that were retrieved (possibly including a number of irrelevant results). Formally these measures are defined as follows:

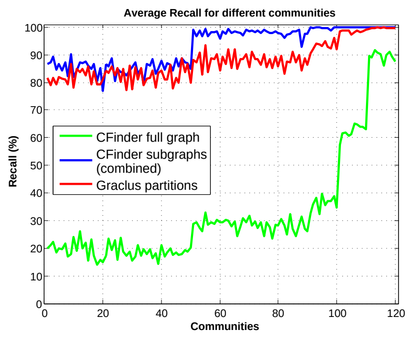
$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

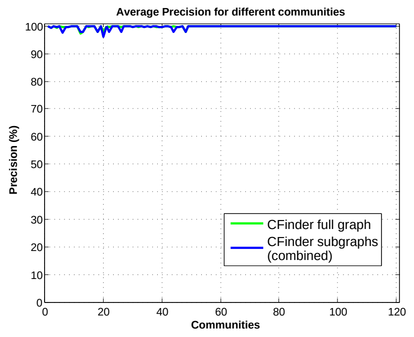
where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives. To calculate the actual validation scores for a given clustering, we compared each reference community from the "ground



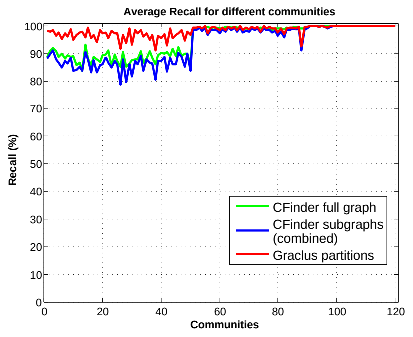
(a)



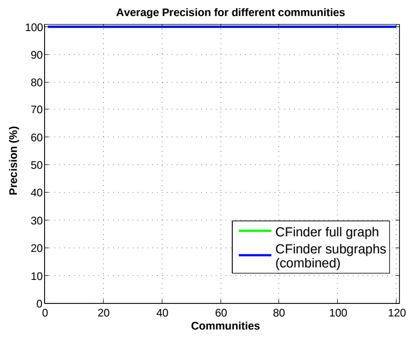
(b)



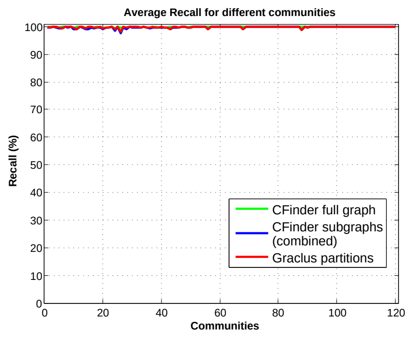
(c)



(d)



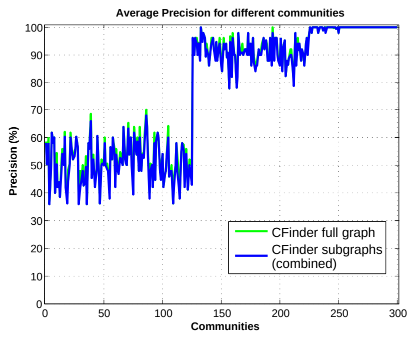
(e)



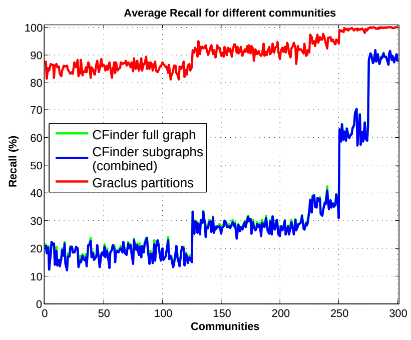
(f)

Fig. 7. Mean precision and recall scores for synthetic graphs of 2,000 nodes, generated with different parameter values: in (a,b) $(p_{in}, r) = (0.2, 0.5)$, in (c,d) $(p_{in}, r) = (0.5, 0.5)$, in (e,f) $(p_{in}, r) = (0.7, 0.7)$.

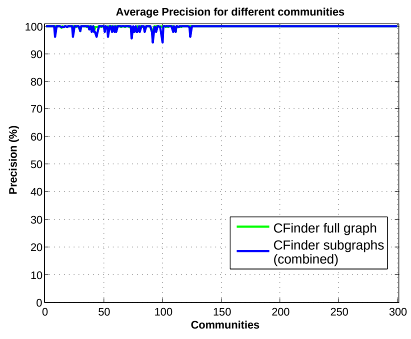
truth" with every one of the clusters discovered by the community finding algorithm. Then, for each of the reference communities, we select the combination yielding the highest values for Eqn. 4 and Eqn. 5, and return these values. We repeated this procedure across multiple experimental runs, and subsequently calculated the mean precision and recall scores.



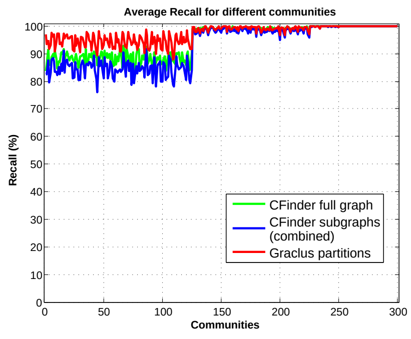
(a)



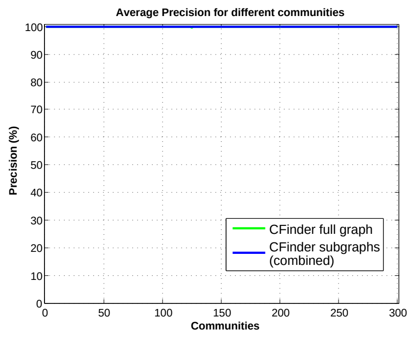
(b)



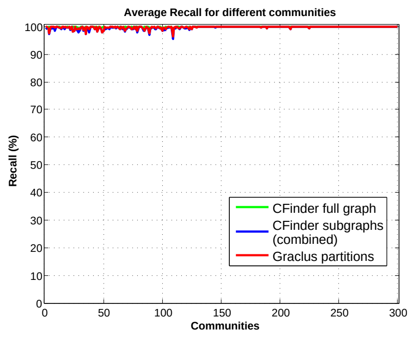
(c)



(d)



(e)



(f)

Fig. 8. Mean precision and recall scores for synthetic graphs of 5,000 nodes, generated with different parameter values: in (a,b) $(p_{in}, r) = (0.2, 0.5)$, in (c,d) $(p_{in}, r) = (0.5, 0.5)$, in (e,f) $(p_{in}, r) = (0.7, 0.7)$.

Discussion of validation results. The mean validation scores for synthetic graphs with embedded communities, containing 2,000 and 5,000 nodes, are shown in Figure 7 and Figure 8 respectively. We provide results for three sets of synthetic graph generation parameters, ranging from weak signatures ($p_{in} = 0.2, r = 0.5$) to strong signatures ($p_{in} = 0.7, r = 0.7$). The communities are arranged in increasing order of size along the x-axis. The plots on the left hand side (a,c,e) correspond to precision scores, while those on the right hand side (b,d,f) correspond to recall scores. Note that, for the graph partitioning phase, we are only concerned with recall scores, as precision values are not meaningful in this context since the number of chosen partitions is usually much smaller than the number of communities. As long as existing communities are not broken up by graph partitioning, there remains the potential for them to be discovered by the community finding algorithm in the second stage of the two-stage strategy.

From Figure 7 and 8 we observe that the recall scores for the partitions obtained from Graclus are consistently high, especially for larger communities. This is a good indication that the first stage of the two-stage strategy is relatively successful in preserving communities intact. In fact, these scores are close to 100% for communities with reasonably well-defined signatures, as shown in Figure 7(c,e) and Figure 8(c,e). It is apparent that, when errors are made, they are most likely to be made on the small communities consisting of only ten nodes. The only situation where CFinder alone outperforms the two-stage strategy in terms of solution quality is in Figure 8(d). Here the recall of the two stage-strategy is slightly worse on the small communities of ten nodes. Note that the graphs in Figure 7(a,b) and Figure 8(a,b), with $(p_{in}, r) = (0.2, 0.5)$, represent a pathological case, where neither the two-stage strategy nor CFinder alone can effectively discover the communities embedded in the data, since the community signatures are not sufficiently well-defined.

5.3 Real-world data

Scalability. In the experiments performed on the CORA bibliographic dataset, we wished to determine the degree to which communities discovered by running CFinder on the entire graph are uncovered when CFinder is run on the individual subgraphs obtained from graph partitioning. For the latter case, we pooled together the communities obtained from the different subgraphs, removing any duplicates. We then compared the total number of communities obtained from the entire graph with those obtained from partitioning the graph into $k = \{2, 3, \ldots, 10\}$ subgraphs. Figure 9 shows that increasing the number of subgraphs had little effect on the total number of communities recovered, with a small decrease from 3601 on the original graph to 3448 for $k = 10$. In contrast, we clearly observe that the overall running time of the two-stage community finding process decreases substantially as the number of subgraphs k increases.

Next we compared the actual communities discovered by applying CFinder on the CORA dataset, with those recovered by the two-stage strategy. Our experiments showed a very considerable overlap. For example, the total number of communities obtained by running CFinder on the full graph was 3,601, and the total number after partitioning the graph into two subgraphs was 3,572. Of these, the number of communities in the two sets that were identical was 3,454. Of the remaining 147 communities obtained from the full graph, there were close matches for 118 communities. When the graph was partitioned into ten subgraphs, the total number of communities was 3,448, of which 2,967 had

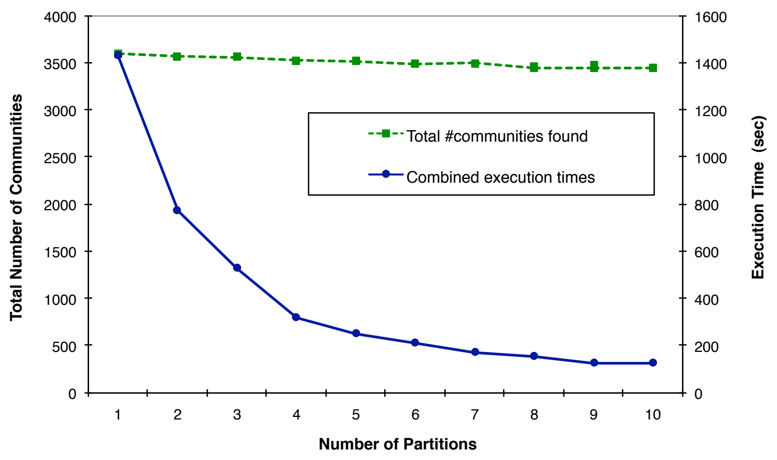


Fig. 9. The execution time for the two-stage community finding process on the CORA dataset. The first data point corresponds to the single stage process (*i.e.* CFinder alone), while the remaining points correspond to the combined times for the two-stage decomposition strategy. The total number of communities discovered is also shown.

exact matches from among the communities of the full graph. Thus, while we achieve a significant reduction in execution time when the two-stage procedure is employed, these results indicate that there is also minimal loss of information overall.

Solution quality. We now consider the issue of solution quality on the CORA dataset, focusing on the ability of the graph partitioning phase of the two-stage strategy to preserve the underlying structures in the data. Specifically, we examined the topic distribution of the clusters obtained by Graclus, where the quality or accuracy of the resulting partitions was determined by using the topic annotations in the CORA dataset as a "ground truth".

Clustering in bibliometrics is based on the assumption that a paper will tend to cite other papers in related areas of research. Therefore, we expect the communities occurring in the partitions generated by graph partitioning to represent relatively homogeneous topics. In our experiments, when Graclus was applied to the CORA dataset, we observed a high level of enrichment with respect to topics in the resulting partitions. Figure 10 shows the overall topic distribution in the CORA dataset, where we see that the top two topics are "artificial intelligence" and "programming", pertaining to 34% and 21% of the papers repectively. Sample results for a two-way partitioning produced by Graclus are shown in Figure 10 and Figure 11, indicating the proportion of each annotated topic present in both clusters. We see that the clusters are enriched with respect to the two dominant topics. The topic "artificial intelligence" accounts for 61% of the papers in one of the clusters, compared to 34% overall, while "programming" accounts for 21% of the nodes in the other cluster, compared with 12% overall.

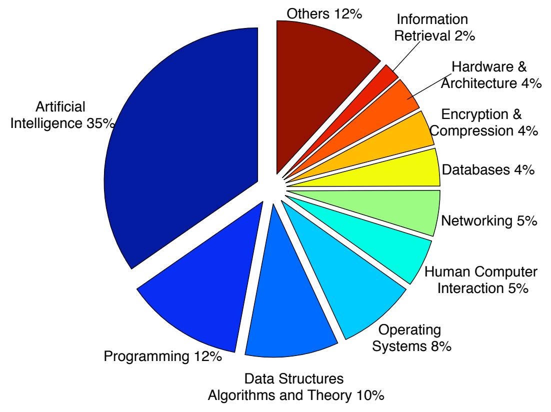
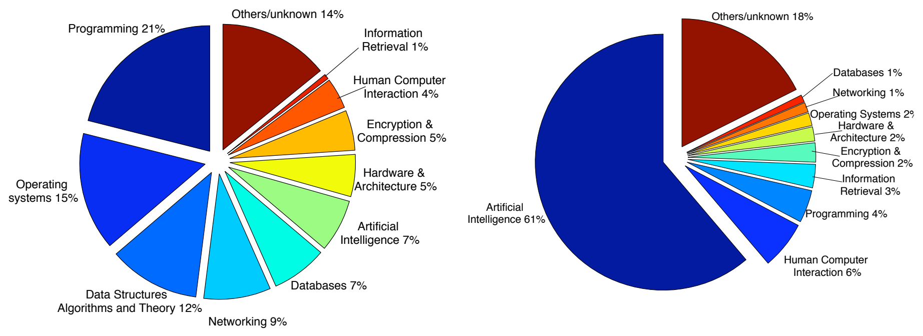


Fig. 10. Original topic distribution of papers in the CORA dataset.



(a) (b)

Fig. 11. Topic distribution of two clusters obtained from Graclus, when applied to the CORA dataset.

5.4 Summary

Since graph partitioning techniques are more scalable than community-finding algorithms, we have shown that the latter represents the "weak link" in the analysis process. Thus, our proposed strategy is feasible within some of the limitations of the community-finding approach. For instance, the authors of CFinder suggest that, if the most densely connected region of the network contains a large number of highly overlapping cliques, the computational performance of CFinder can suffer significantly. In our particular case (*i.e.* where CFinder is used as the community algorithm), the decomposition strategy can be somewhat limited in situations where communities grow in size over time. However, in many real networks, such as phone subscriber data, it will often be the case that more communites of similar size distribution emerge, rather than a "growth" in the size of communities. As illustrated by our evaluations on real and synthetic data, the two-stage strategy would be useful in this context, since it would permit discovery of communities in graphs larger than those which can be handled by CFinder alone.

6 Conclusion

In this report, we have proposed a two-stage problem decomposition approach strategy that facilitates the application of computationally expensive clique or community-finding algorithms to much larger networks than would be possible if these algorithms were applied on their own. The two-stage approach involves using a top-down graph partitioning technique to divide the network into smaller subnetworks, on which it is then feasible to apply a more computationally intensive community-finding algorithm. It is important to note that, while we have used the combination of Graclus (the graph partitioning technique) and CFinder (community-finding algorithm) in this report, the two-stage strategy could be used with alternative combinations of algorithms. We have demonstrated the usefulness of this strategy in empirical evaluations on both artificial and real-world datasets, where the computational cost of the network analysis process was significantly reduced without adversely affecting the quality of the communities that were discovered.

References

1. Paliouras, G., Papatheodorou, C., Karkaletsis, V., Spyropoulos, C.D.: Clustering the users of large web sites into communities. In: Proc. 7th International Conference on Machine Learning (ICML'00). (2000) 719–726
2. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.: Web usage mining: discovery and applications of usage patterns from Web data. ACM SIGKDD Explorations Newsletter **1** (2000) 12–23
3. He, Y., Cheung Hui, S.: Mining a Web Citation Database for author co-citation analysis. Information Processing and Management **38** (2002) 491–508
4. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E **69** (2004)
5. Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435** (2005) 814–8
6. Gregory, S.: An algorithm to find overlapping community structure in networks. In: Proc. 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'07). (2007) 91–102
7. Abello, J., Pardalos, P.M., Resende, M.G.C.: On maximum clique problems in very large graphs. In: External Memory Algorithms. DIMACS Series in Discrete Mathematics and Theoretical Computer Science (1999) 119–130
8. Dhillon, I., Guan, Y., Kulis, B.: Weighted graph cuts without eigenvectors a multilevel approach. IEEE Transactions on Pattern Analysis and Machine Intelligence **29** (2007) 1944–1957

9. Karp, R.: Reducibility among combinatorial problems. Complexity of Computer Computations **43** (1972) 85–103
10. Balas, E., Yu, C.S.: Finding a maximum clique in an arbitrary graph. SIAM J. Comput. **15** (1986) 1054–1068
11. Wood, D.R.: An algorithm for finding a maximum clique in a graph. Operations Research Letters **21** (1997) 211–217
12. Alon, N., Krivelevich, M., Sudakov, B.: Finding a large hidden clique in a random graph. In: 9th Symposium on Discrete Algorithms (SODA). (1998) 91–102
13. Östergård, P.: A fast algorithm for the maximum clique problem. Discrete Applied Mathematics **120** (2002) 197–207
14. Freeman, L.: Centrality in social networks: Conceptual clarification. Social Networks **1** (1979) 215–239
15. Granovetter, M.: The Strength of Weak Ties: A Network Theory Revisited. Sociological Theory **1** (1983) 201–233
16. Girvan, M., Newman, M.: Community structure in social and biological networks. PNAS **99** (2002) 7821–7826
17. Shi, J., Malik, J.: Normalized cuts and image segmentation. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97). (1997) 731–737
18. Chung, F.R.K.: Spectral graph theory. In: CBMS Conference on Recent Advances in Spectral Graph Theory. Number 92 in Regional Conference Series in Mathematics, California State University, Fresno (1994)
19. Pothen, A., Simon, H.D., Liou, K.P.: Partitioning sparse matrices with eigenvectors of graphs. SIAM Journal of Mathematical Analysis and Applications **11** (1990) 430–452
20. Chan, P., Schlag, M., Zien, J.: Spectral k-way ratio cut partitioning. IEEE Transactions CAD-Integrated Circuits and Systems **13** (1994) 1088–1096
21. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. **20** (1999) 359–392
22. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal **49** (1970) 291–307
23. McCallum, A., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Information Retrieval Journal **3** (2000) 127–163
24. Erdős, P., Rényi, A.: On the evolution of random graphs. Bulletin of the Institute of International Statistics **38** (1961) 343–347
25. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks. Science **286** (1999) 509–512
26. Hakimi, S.: On the Realizability of a Set of Integers as Degrees of the Vertices of a Graph,. SIAM J. Appl. Math. **10** (1962) 496–506
27. Watts, D.J., Strogatz, S.: Collective dynamics of "small-world" networks. Nature **393** (1998) 440–442