

Spectral Clustering in Telephone Call Graphs*

Miklós Kurucz András Benczúr Károly Csalogány László Lukács
Data Mining and Web search Research Group, Informatics Laboratory
Computer and Automation Research Institute of the Hungarian Academy of Sciences
{realace, benczur, cskaresz, lacko}@ilab.sztaki.hu

ABSTRACT

We evaluate various heuristics for hierarchical spectral clustering in large telephone call graphs. Spectral clustering without additional heuristics often produces very uneven cluster sizes or low quality clusters that may consist of several disconnected components, a fact that appears to be common for several data sources but, to our knowledge, not described in the literature. Divide-and-Merge, a recently described postfiltering procedure may be used to eliminate bad quality branches in a binary tree hierarchy. We propose an alternate solution that enables k -way cuts in each step by immediately filtering unbalanced or low quality clusters before splitting them further.

Our experiments are performed on graphs with various weight and normalization built based on call detail records. We investigate a period of eight months of more than two millions of Hungarian landline telephone users. We measure clustering quality both by cluster ratio as well as by the geographic homogeneity of the clusters obtained from telephone location data. Although divide-and-merge optimizes its clusters for cluster ratio, our method produces clusters of similar ratio much faster, furthermore we give geographically much more homogeneous clusters with the size distribution of our clusters resembling to that of the settlement structure.

Categories and Subject Descriptors

J.4 [Computer Applications]: Social and Behavioral Sciences; G.2.2 [Discrete Mathematics]: Graph Theory—*Graph algorithms*; G.1.3 [Mathematics of Computing]: Numerical Analysis—*Numerical Linear Algebra*

General Terms

clustering, social networks

*Support from a Yahoo Faculty Research Grant, the Inter-University Center for Telecommunications and Informatics (ETIK) and by grant *ASTOR* NKFP 2/004/05

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Joint 9th WEBKDD and 1st SNA-KDD Workshop '07 (WebKDD/SNA-KDD'07), August 12, 2007, San Jose, California, USA, 2007
Copyright 2007 ACM 978-1-59593-848-0 ...\$5.00.

Keywords

spectral clustering, telephone call graph, social network mining, sociodemographic exploration

1. INTRODUCTION

In general, clustering covers a wide class of methods to locate relevant information and organize it in an intelligible way. The purpose of clustering telephone users includes user segmentation, selection of communities with desired or undesired properties as e.g. high ADSL penetration or high recent churn rate or for viral marketing [35]: we form groups to enhance marketing communication by also relying on the spread of information within the social network. We show, even if geographic location is available, clusters have more desirable properties such as the weight of edges across different clusters are much smaller.

The main contribution of our research is the use of telephone call graphs for testing and evaluating clustering algorithms. We believe the telephone call graphs behave similar to other social networks such as those of bloggers and our results may be used in a more general setting. As a preliminary experiment we included a measurement on the LiveJournal blogger network and identified the well-known Russian user group [24, 43] in Section 3.2.

The telephone call graph is formed from the call detail record, a log of all calls within a time period including caller and callee id, duration, cost and time stamp. The vertex set consists of all nodes that appear at least once as caller or callee; over this set calls form directed edges from caller to callee. Edges are weighted by various aggregates of call multiplicity, duration or cost; time stamps are ignored in this work. The resulting graph obeys the power law degree distribution and contains a giant connected component of almost all nodes [1].

We compare several clustering measures on call graphs. Unlike in the examples of [26], in our graphs the “right” clustering is by no means obvious but, similar to the findings of [26], the goodness measures can be fooled. The typical examples of practically useless spectral splits have uneven sizes or disconnected clusters; in certain cases the clustering procedure simply wastes computational resources for unnecessary steps, a phenomenon reported in particular for power law graphs [30]. We believe our findings are beyond “it works well on my data” and apply to a more general class of social networks or other small-world power law graphs.

Practical evaluation of spectral clustering in graphs is investigated mainly in the area of netlist partitioning [4] with the recent exception of the findings of Lang [29, 30]. He

suggests semidefinite programming techniques to avoid imbalanced cuts, however the reported running times are several hours for a single cut even for 10 million edge graphs. Techniques to scale the semidefinite programming based approaches and a comparison of the performance remains future work.

The telephone call graph appears less in the publications of the data mining community compared to the social network of bloggers [28, and references therein] or the World Wide Web [18, and many others]. Few exceptions include a theoretical analysis of connected components and eigenvalues [1, 14, 13] and several churn prediction by machine learning methods on real data [40; 5, etc.]. Closest to our results are the structural investigations of mobile telephone call graphs [33, 34] and the sketch-based approximate k -means clustering of traffic among AT&T collection stations over the United States [15]; for this latter result however the underlying graph is much smaller (20,000 nodes) and their main goal is to handle the time evolution as an additional dimension. Telephone call graphs are also used by the graph visualization community: [42] reports visualization on graphs close to the size of ours with efficient algorithms to select the neighborhood subgraph to be visualized. In addition [16] gives an example of long distance telephone call fraud application by manual investigation.

While a comprehensive comparison of clustering algorithms is beyond the scope of this paper, we justify the use of a top-down hierarchical clustering by observing that telephone call graphs and social networks in general are small world power law graphs. Small world implies very fast growth of neighborhood that strongly overlap; power law implies high degree nodes that locally connect a large number of neighboring nodes. Recent bottom-up alternatives such as clique percolation [17] suffer from these phenomena: the extreme large number of small (size 5 or 6) cliques do not only pose computational challenges but also connect most of the graph into a single cluster; the number of larger sized cliques however quickly decays and by using them we leave most of the nodes isolated or in very small clusters. The superiority of spectral clustering over density based methods is also suggested in [10] for document collections.

The applicability of spectral methods to graph partitioning is observed in the early 70's [22, 20]. The methods are then rediscovered for netlist partitioning, an area related to circuit design, in the early 90's [9, 2, 4, 3]. Since the "Spectral Clustering Golden Age" [19; 32; 44, etc] 2001 we only list a random selection of results. Spectral clustering is applied for documents [44, 10, 11] as well as image processing [38, 31, 32], see many earlier references in [26]. More recently, several approximate SVD algorithms appeared [23; 21; 36, and many others]; with the expansion of available data volumes their use in practice is likely in the near future.

Our experiments are performed on the call graph of more than two millions of Hungarian landline telephone users [7], a unique data of long time range with sufficiently rich sociodemographic information on the users. We differ from prior work on spectral clustering in two aspects:

- We evaluate clustering algorithms by measuring external sociodemographic parameters such as geographic location in addition to graph properties such as cluster ratio.
- Our problems are larger than previously reported: the

recent Divide-and-Merge algorithm [11] runs experiments over 18,000 nodes and 1.2 millions of nonzeros compared to near 50,000,000 edges in our graph. Improved hardware capabilities hence require new algorithms and lead to new empirical findings in the paper.

We summarize our key findings on implementing spectral clustering in the telephone call graph that may be applied for other graphs as well.

- We give a k -way hierarchical clustering algorithm variant that outperforms the recently described Divide-and-Merge algorithm of Cheng et al. [11] both for speed and accuracy.
- Compared to the Laplacian $D - A$ typically used for graph partitioning, we show superior performance of the normalized Laplacian $D^{-1/2}AD^{-1/2}$ introduced for spectral bisection in [38] and [19] as the relaxation of the so-called normalized cut and min-max cut problems, respectively. We are aware of no earlier systematic experimental comparison. While in [38, 18] both described, their performance is not compared in practice; Weiss [41] reports "unless the matrix is normalized [...] it is nearly impossible to extract segmentation information" but no performance measures are given; finally [39] give theoretic evidence for the superiority of normalization.
- We compare various edge weighting schemes, in particular introduce a neighborhood Jaccard similarity weight. This weight outperforms the best logarithmic weighting in certain cases, justifying the discussion of [26, Section 2] that the input matrix should reflect the similarity of the nodes instead of their distance.
- We introduce size balancing heuristics that improve both the geographic homogeneity and the size distribution of the clusters formed by the algorithm. These methods outperform and completely replace Lin-Kernighan type heuristics proposed by [19].
- We partially justify previous suggestions to use several eigenvectors [4, 31]; however we observe no need for too many of them.

2. SPECTRAL CLUSTERING ALGORITHMS FOR THE CALL GRAPH

Spectral clustering refers to a set of heuristic algorithms, all based on the overall idea of computing the first few singular vectors and then clustering in a low (in certain cases simply one) dimensional subspace. Variants dating back to the 1970's described in [4] fall into two main branches. The first branch is initiated by the seminal work of Fiedler [22] who separates data points into the positive and negative parts along the principal axes of the projection. His original idea uses the second singular vector, the so-called Fiedler vector; later variants [6, 2] use more vectors. Hagen and Kahng [25] is perhaps the first to use the second smallest eigenvalue for graph partitioning of difficult real world graphs.

The second branch of hierarchical spectral clustering algorithm divides the graph into more than two parts in one step. While the idea of viewing nodes as d -dimensional vectors after projecting the adjacency matrix into the space of

the top k singular vectors is described already by Chan et al. [9], much later Zha et al. [44] introduce the use of k -means over the projection.

The formation of the input matrix to SVD computation from the detailed call list strongly affects the outcome of clustering. In addition to various ways of using cost and duration including a neighborhood Jaccard similarity weight, in Section 2.5 we also compare the use of the Laplacian and weighted Laplacian. The *Laplacian* is $D - A$ such that D is the diagonal matrix where the i -th entry is the total edge weight at node i . The *weighted Laplacian* $D^{-1/2}AD^{-1/2}$ is first used for spectral bisection in [38, 19]. The Laplacian arises as the relaxation of the minimum ratio cut [25]; weighted Laplacian appears in the relaxation of normalized cut [38] and min-max cut [19].

While implementation issues of SVD computation are beyond the scope of the paper, we compare the performance of the Lanczos and block Lanczos code of `svdpack` [8] and our implementation of a power iteration algorithm. Hagen et al. [25] suggest fast Lanczos-type methods as robust basis for computing heuristic ratio cuts; others [26, 11] use power iteration. Since the SVD algorithm itself has no effect on the surrounding clustering procedure, we only compare performances later in Section 3.5.

2.1 Algorithm overview

In the bulk of this section we describe our two main algorithms, one belonging to each branch of hierarchical spectral clustering. In both cases good cluster qualities are obtained by heuristics for rejecting uneven splits and small clusters described in general in Section 2.2. The first algorithm in Section 2.3 is based on k -way hierarchical clustering as described among others by Alpert et al. [4]; the second one in Section 2.4 on the more recent Divide-and-Merge algorithm [11].

When clustering the telephone call graph, the main practical problem arises when the graph or a remaining component contains a densely connected large subset. In this case spectral clustering often collects tentacles loosely connected to the center [12] into one cluster and keeps the dense component in one [30]. While even the optimum cluster ratio cut might have this structure, the disconnected cluster consists of small graph pieces that each belong strongly to certain different areas within the dense component. In addition a disconnected graph has multiple top eigenvalue, meaning that we must compute eigenvectors separate for each connected component. However if we treat each connected component as a separate cluster, we obtain an undesired very uneven distribution of cluster sizes.

Both of the algorithms we describe target at balancing the output clusters. The original Divide-and-Merge algorithm of [11] achieves this simply by producing more clusters than requested and merging them in a second phase. We observed this algorithm itself is insufficient for clustering power law graphs since for our data it chops off small pieces in one divide step. In a recursive use for hierarchical clustering the number of SVD calls hence becomes quadratic in the input size even if only a relative small number of clusters is requested.

The key in using spectral clustering for power law graphs is our small cluster redistribution heuristics described in the next subsection. After computing a 2-way or k -way split we test the resulting partition for small clusters. First we try to

redistribute nodes to make each component connected. This procedure may reduce the number of clusters; when we are left with a single cluster, the output is rejected. The main difference in our two algorithms is the way rejected cuts are handled as described in Sections 2.2 and 2.3.

2.2 Small cluster redistribution heuristics

Algorithm 1 `redistribute`(C_1, \dots, C_k): Small cluster redistribution

```

 $C_0 = C'_1 \cup \dots \cup C'_{k'}$   $\leftarrow$  the connected components of
 $C_1, \dots, C_k$ 
repeat
   $p \leftarrow -1$ 
  for all  $C'_i$  do
    if  $|C'_i| < \text{limit} \cdot |C_0|$  then
      for all  $v \in C'_i$  do
        Find  $C'_j$  with largest total edge weight  $d(v, C'_j)$ 
        among  $i \neq j$ 
         $p[v] \leftarrow j$ 
      for all  $v \in C_0$  do
        if  $p[v] \neq -1$  then
          Move  $v$  to  $C'_{p[v]}$ 
    until there were no changes
return all nonempty  $C'_i$ 

```

We give a subroutine to rejects very uneven splits that is used in both our Divide-and-Merge implementation (Section 2.2) and in k -way clustering (Section 2.3). Given a split of cluster C_0 (that may be the entire graph) into at least two clusters $C_0 = C_1 \cup \dots \cup C_k$, we first form the connected components of each C_i . This results in a split into possibly more parts, $C_0 = C'_1 \cup \dots \cup C'_{k'}$ with $k' \geq k$. Now we impose a relative threshold `limit` and redistribute the small clusters to others vertex by vertex, in one step scheduling a vertex v to component C'_j with $d(v, C'_j)$ maximum where $d(A, B)$ denotes the number of edges with one end in A and another in B . Scheduled vertices are moved into their clusters at the end so that the output is independent of the order vertices v are processed. By this procedure we may be left with more or less than k components; more than k never occurs in practice. We will have to reject clustering C_0 if we are left with the single cluster $C'_i = C_0$; in this case we either try splitting it with modified parameters or completely give up forming subclusters of C_0 .

2.3 K -way hierarchical clustering

Algorithm 2 k -way hierarchical clustering

```

while we have less than cnum clusters do
   $A \leftarrow$  adjacency matrix of largest cluster  $C_0$ 
  Project  $D^{-1/2}AD^{-1/2}$  into first  $d$  eigenvectors
  For each node  $i$  form vector  $v'_i \in R^d$  of the projection
   $v_i \leftarrow v'_i / \|v'_i\|$ 
   $(C_1, \dots, C_k) \leftarrow$  output of  $k$ -means( $v_1, \dots, v_{|C_0|}$ )
  Call redistribute( $C_1, \dots, C_k$ )
  Discard  $C_0$  if  $C_0$  remains a single cluster

```

In our benchmark implementation we give k , the number of subclusters formed in each step, d , the dimension of the SVD projection and `cnum`, the required number of clusters as input. Algorithm 2 then always attempts to split the largest

available cluster into $k' \leq k$ pieces by k -means after a projection onto d dimensions. Note that k -means may produce less than the prescribed number of clusters k ; this scenario typically implies the hardness of clustering the graph. If, after calling small cluster redistribution (Algorithm 1), we are left with a single cluster, we discard C_0 and does not attempt to split it further.

In our real life application we start out with low values of d and increase it for another try with C_0 whenever splitting a cluster C_0 fails. We may in this case also decrease the balance constraint.

Notice the row normalization step $v_i \leftarrow v'_i / \|v'_i\|$; this step improves clustering qualities for our problem. We also implemented column normalization, its effect is however negligible.

2.4 Divide-and-Merge Baseline

Algorithm 3 Divide and Merge: Divide Phase

```

while we have less than cnum0 clusters do
   $A \leftarrow$  adjacency matrix of largest cluster  $C_0$ 
  Compute the second largest eigenvector  $v'$  of  $D^{-1/2}AD^{-1/2}$ 
  Let  $v = D^{-1/2}v'$  and sort  $v$ 
   $i \leftarrow$  ratio_init
  while  $C_0$  is not discarded do
    Find  $1/i \leq t \leq 1 - 1/i$  such that the cut
       $(S, T) = (\{1, \dots, t \cdot n\}, \{t \cdot n + 1, \dots, n\})$ 
    minimizes the cluster ratio
     $(C_1, \dots, C_\ell) \leftarrow$  redistribute( $S, T$ )
    if  $\ell > 1$  then
      Discard  $C_0$  and add clusters  $C_1, \dots, C_\ell$ 
    else
      if  $i = 3$  then
        Discard cluster  $C_0$ 
      else
         $i \leftarrow i - 1$ 

```

The Divide-and-Merge algorithm of Cheng et al. [11] is a two phase algorithm. In the first phase we recursively bisect the graph: we perform a linear scan in the second eigenvector of the Laplacian sorted by value to find the optimal bisection. The algorithm produces `cnum0` clusters that are in the second phase merged to a required smaller number `cnum` of clusters by optimizing cut measures via dynamic programming.

In order to adapt the Divide-and-Merge algorithm originally designed for document clustering [11], we modify both phases. First we describe a cluster balancing heuristic based on Algorithm 1 for the divide phase. Then for the merge phase we give an algorithm that produces low cluster ratio cuts, a measure defined below in this section. In [11] the merge phase of the divide-and-merge algorithm is not implemented for cluster ratio. Since this measure is not monotonic over subclusters, we give a new heuristic dynamic programming procedure below.

We observed tiny clusters appear very frequent in the Divide phase (Algorithm 3) as described in Section 2.2. Splits along the second eigenvector are apparently prone to find a disconnected small side consisting of outliers. In this case the small component heuristics of Algorithm 1 are insufficient themselves since we are starting out with two clusters;

if we completely redistribute one, then we are left with the component unsplit. We hence introduce an additional balancing step with the intent to find connected balanced splits along the second eigenvector. We could restrict linear scan to an 1/3-2/3 split; in many cases this however leads to a low quality cut. Hence first we weaken the restriction to find an $1/\text{ratio_init} - (1 - 1/\text{ratio_init})$ cut and gradually decrease the denominator down to 3. We stop with the first cut not rejected by Algorithm 1. If no such exists, we keep the cluster in one and proceed with the remaining largest one.

Algorithm 4 Merge Phase

```

for all clusters  $C_0$  from leaves up to the root do
  if  $C_0$  is leaf then
     $\text{OPT}_n(C_0, 1) = 0, \text{OPT}_d(C_0, 1) = |C_0|$ 
  else
    Let  $C_1, \dots, C_\ell$  be the children of  $C_0$ 
    for  $i$  between 1 and total below  $C_0$  do
       $\text{numer}(i_1, \dots, i_\ell) \leftarrow 0; \text{denom}(i_1, \dots, i_\ell) \leftarrow 1$ 
      for all  $i_1 + \dots + i_\ell = i$  do
         $\text{numer}(i_1, \dots, i_\ell) \leftarrow$ 
           $\sum_{j \neq j'} d(C_j, C_{j'}) + \sum_{j=1 \dots \ell} \text{OPT}_n(C_j, i_j)$ 
         $\text{denom}(i_1, \dots, i_\ell) \leftarrow$ 
           $\sum_{j \neq j'} |C_j| \cdot |C_{j'}| + \sum_{j=1 \dots \ell} \text{OPT}_d(C_j, i_j)$ 
        if  $\frac{\text{OPT}_n(C_0, i)}{\text{OPT}_d(C_0, i)} > \frac{\text{numer}(i_1, \dots, i_\ell)}{\text{denom}(i_1, \dots, i_\ell)}$  then
           $\text{OPT}_n(C_0, i) = \text{numer}(i_1, \dots, i_\ell)$ 
           $\text{OPT}_d(C_0, i) = \text{denom}(i_1, \dots, i_\ell)$ 

```

Now we turn to the the Merge phase (Algorithm 4). Our goal is to optimize the final output for cluster ratio defined below. Let there be N users with N_k of them in cluster k for $k = 1, \dots, m$. The *cluster ratio* is the number of calls between different clusters divided by $\sum_{i \neq j} N_i \cdot N_j$. The *weighted cluster ratio* is obtained by dividing the total weight of edges between different clusters by $\sum_{i \neq j} w_{ij} N_i \cdot N_j$ where w_{ij} is the total weight of edges between cluster i and j .

In order to compute the optimal merging upwards from leaves by dynamic programming (Algorithm 4) we aim to use an idea similar to computing cluster ratio when linearly scanning in the Divide step as described in [10]. Unfortunately however cluster ratio is not monotonic in the cluster ratio within a subcomponent; instead we have to add the numerator and denominator expressions separately within the subcomponents. We can only give a heuristic solution below to solve this problem.

In order to find a good cluster ratio split into i subsets of a given cluster C_0 , we try all possible $i_1 + \dots + i_\ell = i$ split sizes within subclusters C_1, \dots, C_ℓ . By the dynamic programming principle we assume good splits into i_j pieces are known for each subcluster C_j ; as we will see, these may not be optimal though. For these splits we require the numerator and denominator values $\text{OPT}_n(C_j, i_j)$ and $\text{OPT}_d(C_j, i_j)$. If we use the corresponding splits for all j , we obtain a split

of cluster ratio

$$\frac{\sum_{j \neq j'} d(C_j, C_{j'}) + \sum_{j=1 \dots \ell} \text{OPT}_n(C_j, i_j)}{\sum_{j \neq j'} |C_j| \cdot |C_{j'}| + \sum_{j=1 \dots \ell} \text{OPT}_d(C_j, i_j)}$$

for the union of the subcomponents. Note however that this expression is not monotonic in the cluster ratio of subcomponent j , $\text{OPT}_n(C_j, i_j)/\text{OPT}_d(C_j, i_j)$, and the minimization of the above expression cannot be done by dynamic programming. As a heuristic solution, in Algorithm 4 we always use the optimal splits from children. Even in this setting the algorithm is inefficient for branching factor more than two; while in theory Merge could be used after k -way partitioning as well, the running time is exponential in k since we have to try all (or at least most) splits of i into $i_1 + \dots + i_\ell$.

2.5 Weighting schemes

Spectral clustering algorithm may take any input matrix A and partition the rows based on the geometry of their projection into the subspace of the top k singular vectors [26]. Kannan et al. [26] suggest modeling the input as a similarity graph rather than as a distance graph, raising the question of interpreting the call information including the number, total duration and price between a pair of callers.

Earlier results for graph partitioning either use the unweighted or weighted Laplacian $D - A$ vs. $D^{-1/2}AD^{-1/2}$, the first appearing in the relaxation of the ratio cut [25], the second the normalized [38] and min-max [19] cut problems. Weighting strategies in more detail are discussed for netlist partitioning are only [4, and references therein]; in particular Alpert and Kahng [2] empirically compared some of them. Since netlists are hypergraphs, we may not directly use their findings, however they indicate the importance of comparing different strategies to weight the graph.

We have several choices to extract the social network based on telephone calls between users: we may or may not ignore the direction of the edges and weight edges by number of calls, duration or price, the latter emphasizing long range contacts.

First of all we may try to directly use the total cost or duration as weight in the adjacency matrix. However then the Lanczos algorithm converges extremely slow; while it converges within a maximum of 120 iterations in all other cases, 900 iterations did not suffice for a single singular vector computation with raw values. We hence use $1 + \log w_{ij}$ where w_{ij} is either the total cost or duration between a pair of users i and j .

We also investigate a Jaccard similarity based weight of user pairs that characterize the strength of their connection well, based on the remark of [26] for modeling the input as a similarity graph. Since filling a quadratic size matrix is infeasible, we calculate the ratio of their total call duration made to common neighbors and of their total duration for all existing edges. This method results in weights between 0 and 1; the reweighted graph yields clusters of quality similar to the logarithm of call cost or duration. In our algorithms we use $1 + \text{Jac}_{ij}$ to distinguish non-edges from low weight edges.

We build a graph from the detailed call record so that a vertex is assigned to each customer and an edge is drawn between two vertices if they have called each other during the specified time period. The edges are weighted by the total time of calls between the two vertices. However, this

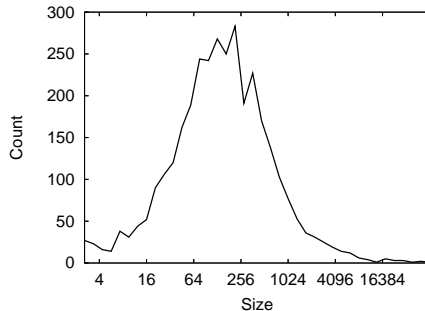


Figure 1: Distribution of settlement sizes in the data.

weighting requires quadratic space and is hence infeasible for the scale of our problem; we only compute the weight for existing edges.

This similarity coefficient is also useful for finding important connections and ignoring "accidental" unimportant connections. We sort all pairs of vertices descending by the above similarity coefficient and compare the resulting order with the actual edges of the original graph by counting the ratio of actual edges and toplist size in different sized toplists of the order. The resulting scheme downweights unimportant edges and adds "missing" calls to the network.

3. EXPERIMENTS

The experiments were carried out on a cluster of 64-bit 3GHz P-D processors with 4GB RAM each. Depending on algorithms and parameter settings, the running time for the construction of 3000 clusters is in the order of magnitude of several hours or a day.

3.1 Data set

For a time range of 8 months, after aggregating calls between the same pairs of callers we obtained a graph with $n = 2,100,000$ nodes and $m = 48,400,000$ directed edges that include 10,800,000 bidirectional pairs.

Settlement sizes (Fig. 1) follow a distribution very close to lognormal with the exception of a very heavy tail of Hungary's capital Budapest of near 600,000 users. In a rare number of cases the data consists of subpart names of settlements resulting in a relatively large number of settlements with one or two telephone numbers; since the total number of such nodes is negligible in the graph, we omit cleaning the data in this respect.

We discard approximately 30,000 users (1.5%) that become isolated from the giant component; except for those 130 users initially in small components all nodes can be added to the cluster with most edges in common but we ignore them for simplicity.

The graph has strong topdown regional structure with large cities appearing as single clusters. These small world power law graphs are centered around very large degree nodes and very hard to split. In most parameter settings we are left with a large cluster of size near that of the Budapest telephone users. For this reason we re-run some experiments with Budapest users removed from the graph.

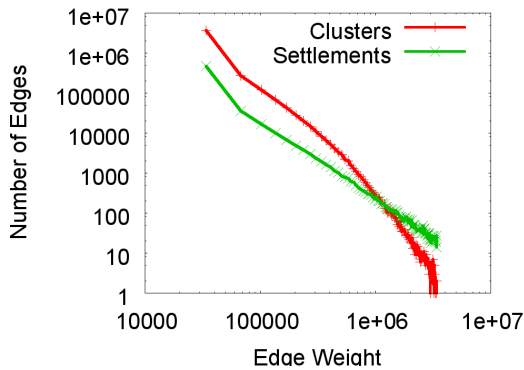


Figure 2: Distribution of the edge weights across different clusters, for a spectral clustering and a trivial clustering obtained by considering one settlement as one cluster. The horizontal axis contains the total edge weight in seconds and the vertical axis shows the number of cluster pairs with the given weight between them.

One may argue whether clustering reveals additional information compared to the settlements themselves as “ground truth” clusters. We give positive answer to this question by showing that the distribution of the total call duration across different clusters is superior for those obtained by spectral clustering. In Fig. 2 we form two graphs, one with a node for each settlement and another with a node for each (spectral) cluster. The weight of an edge between two such nodes is the total call duration between the corresponding clusters. We observe both graphs have power law distribution. The graph obtained by spectral clustering has a much smaller exponent and the edges across clusters have much smaller weight. In fact we use settlement information as an external validation tool for our experiments and not as ground truth.

3.2 The LiveJournal Blogger network

We crawled $n = 2,412,854$ LiveJournal bloggers and formed a graph with $m = 40,313,378$ edges, out of which 18,727,775 is bidirectional. By using $d = 30$ we obtained two clusters, one consisting of the well-known Russian user group [24, 43] of 85,759 users with location given in Russia, 9,407 in Ukraine and 29,697 outside (with large number from US, Israel, Belarus and Estonia); the large cluster contains 1,849 users who gave Russia as location. When trying to split the large cluster further, we obtained an eigenvalue sequence very close to one with $\sigma_{30} = 0.986781$. For even a 100-dimensional embedding the k -way clustering algorithm managed only to chop off tiny clusters, as observed in general in [30]. Improving the performance of our algorithm for this type of data remains future work.

3.3 Evaluation measures

3.3.1 Graph based properties

In the next two subsections we define the quality measures we use for evaluating the output of a clustering algorithm besides cluster ratio defined in Section 2.4. While several measures other than (weighted) cluster ratio exist for measuring the quality of a graph partition, cluster ratio reflect best the balance constraints and applies best to large num-

ber of parts. We remark that we always compute cluster ratio with the original edge weights regardless of the matrix used for SVD computation.

3.3.2 Sociodemographic properties

Telephone users as nodes have rich attributes beyond graph theory. We may measure clustering quality by the entropy and purity of geographic location or other external property within the cluster. By using the notation of the previous subsection let $N_{i,k}$ denote the cluster confusion matrix, the number of elements in cluster k from settlement i and let $p_{i,k} = N_{i,k}/N_k$ denote the ratio within the cluster. Then the *entropy* E and *purity* P [27] (the latter also called *accuracy* in [10]) are defined as

$$E = (-1/\log m) \sum_k (N_k/N) \sum_i p_{i,k} \log p_{i,k} \quad \text{and}$$

$$P = \frac{1}{N} \sum_k \max_i N_{i,k},$$

where the former is the average entropy of the distribution of settlements within the cluster while the latter measures the ratio of the “best fit” within each cluster.

3.4 Divide-and-Merge vs. k -way hierarchical algorithm with different input matrices

The comparison of various input matrices to both divide-and-merge and k -way hierarchical clustering is shown in Fig. 3. Most importantly we notice the weighted Laplacian $D^{-1/2}AD^{-1/2}$ significantly outperforms the unweighted $D - A$ in all respects. Call length and call cost behaves similar; as expected, the former yields geographically more homogeneous clusters by underemphasizing long distance calls, while the latter performing better for the cluster ratio measure. The logarithm of the price or duration performs very close to Jaccard reweighting with no clear winner.

When comparing Divide-and-Merge and k -way partitioning (Fig. 3) we observe the superiority of the latter for larger k . For $k = 2$ we basically perform Divide without Merge; the poor performance is hence no surprise. For $k = 4$ however the small cluster redistribution heuristic already reaches and even outperforms the flexibility of the Merge phase in rearranging bad splits.

3.5 Evaluation of Singular Value Decomposition algorithms

In our implementation we used the Lanczos code of `svdpack` [8] and compared it with block Lanczos and a power iteration developed from scratch. While block Lanczos runs much slower, it produces the exact same output as Lanczos; in contrast power iteration used by several results [26, 11] is slightly faster for computing the Fiedler vector but much less accurate; computing more than two dimensions turned out useless due to the numerical instability of the orthogonal projection step. Improving numerical stability is beyond the scope of this paper and we aware of no standard power iteration implementation. Running times for the first split are shown in Table 1; in comparison the semidefinite programming bisection code of Lang [30] ran 120 minutes for a much smaller subgraph ($n = 65,000$, $m = 1,360,000$) while for the entire graph it did not terminate in two days.

We remark that modifications of `svdpack` are necessary to handle the size of our input. After removing the obso-

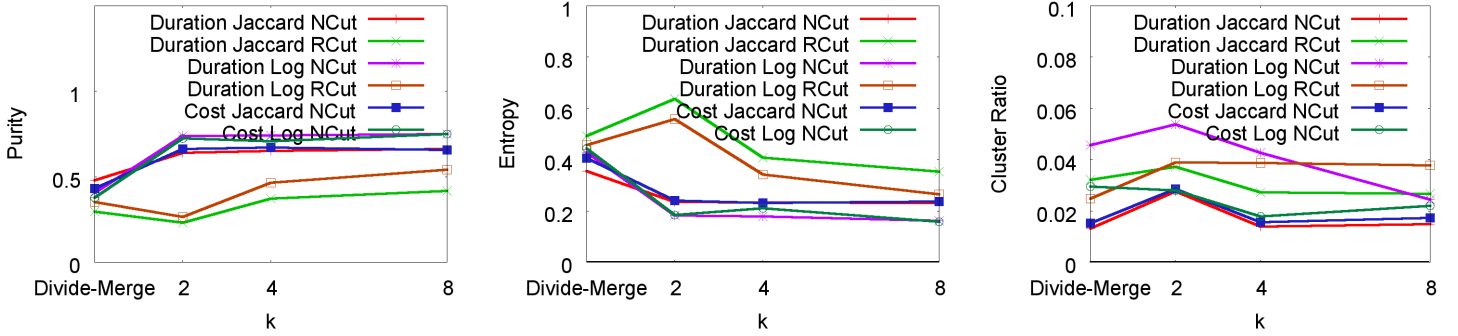


Figure 3: Evaluation of various reweighting techniques over the adjacency matrix for purity (left), entropy (center) and cluster ratio (right) of the arising clusters on the vertical axis. Curves correspond to combinations of unweighted vs. weighted Laplacian (NCut for normalized cut relaxation, as opposed to RCut, ratio cut relaxation), length vs. cost based, and Jaccard vs. logarithmic weight input matrices. Four different algorithms, Divide-and-Merge bipartition as well as k -way partition with $d = 30$ for three values $k = 2, 4$ and 8 are on the horizontal axis.

Algorithm	$d = 2$	$d = 5$	$d = 10$	$d = 15$	$d = 20$	$d = 25$
Lanczos	17	24	44	47	55	96
Block Lanczos	19	34	66	105	146	195
Power	15	40	95	144	191	240

Table 1: Running times for the d dimensional SVD computation by various algorithms, in minutes.

lete condition on the maximum size of an input matrix, we abstracted data access within the implementation to computing the product of a vector with either the input matrix or its transpose.

The entire running time for producing 3000 clusters depend more on the parameter settings than the choice of Divide-and-Merge vs. k -way partitioning. All runs took several hours up to a day; only the slowest Divide-and-Merge with `limit = 100` running over a day. Since the number of possible parameters is very large, we omitted running time graphs.

3.6 Size limits and implications on the size distribution of clusters

In Fig. 4 we see the effect of changing `limit` for the two algorithms. Recall (Algorithm 1) that `limit` bounds the ratio of the smallest cut from the average. If this is very large (100 in the Figure), we are left with no constraint. If however it is close to one, we enforce very strict balancing that deteriorates clustering quality. The optimal values lie around 4...6; these values are also optimal for running time. Very large values, in particular for Divide-and-Merge, slow algorithms down by producing low size gain in one costly SVD computation. Notice the unexpected increase of cluster ratio for large values; this is due to the fact that the densely connected near 600,000 Budapest users could only be split with liberal balancing conditions. While splitting Budapest has no effect on purity or entropy, it adds a large number of edges cut in cluster ratio. For this reason we repeated the experiment by removing Budapest users to see no negative effect of liberal balance on the clustering quality measures.

Notice the superiority of the k -way algorithm over Divide-and-Merge is also clear for their best parameter settings of Fig. 4.

We also remark here that we implemented a Lin-Kernighan

type point redistribution at cut borders proposed by [19] but it had negligible effect on the quality.

Besides clustering quality, we also look at how “natural” are the cluster sizes produced by the algorithms in Fig. 5. We observe strong maximum cluster size thresholds for Divide-and-Merge: that algorithm forces splitting hard regions for the price of producing a negatively skewed distribution of a large number of small clusters that are of little practical use. With the exception of Divide-and-Merge with no limits we never split Budapest users as seen from the top list (Fig. 5, right). When repeating the experiment by discarding Budapest users, the huge clusters disappear.

3.7 The effect of more dimensions

As suggested by [4, 31] more eigenvalues produce better quality cuts. However the price for using more eigenvalues is slowdown hence a good balance between the number d of eigenvalues and the branching k must be chosen. In Fig. 6 we observe we should not choose k too large (somewhere between 5 and 10 for this graph) but compute somewhat more eigenvectors.

Conclusion

We gave a k -way hierarchical spectral clustering algorithm with heuristics to balance cluster sizes. We also implemented the heuristics in the recent Divide-and-Merge algorithm [11]. Our algorithm outperformed Divide-and-Merge for clustering the telephone call graph. We also measured the effect of several choices for the input to SVD: we found the weighted Laplacian performing much better than the unweighted counterpart and introduced a neighborhood Jaccard weighting scheme that performs very good for SVD input.

For further work we propose the implementation and com-

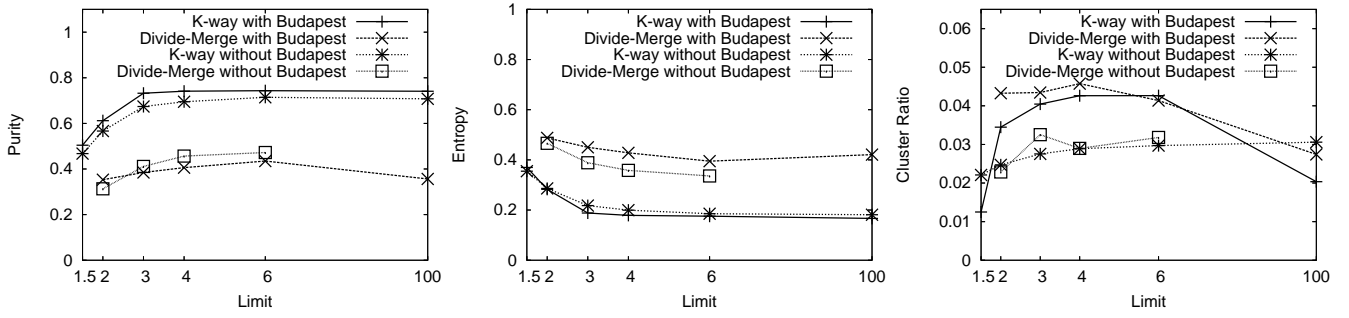
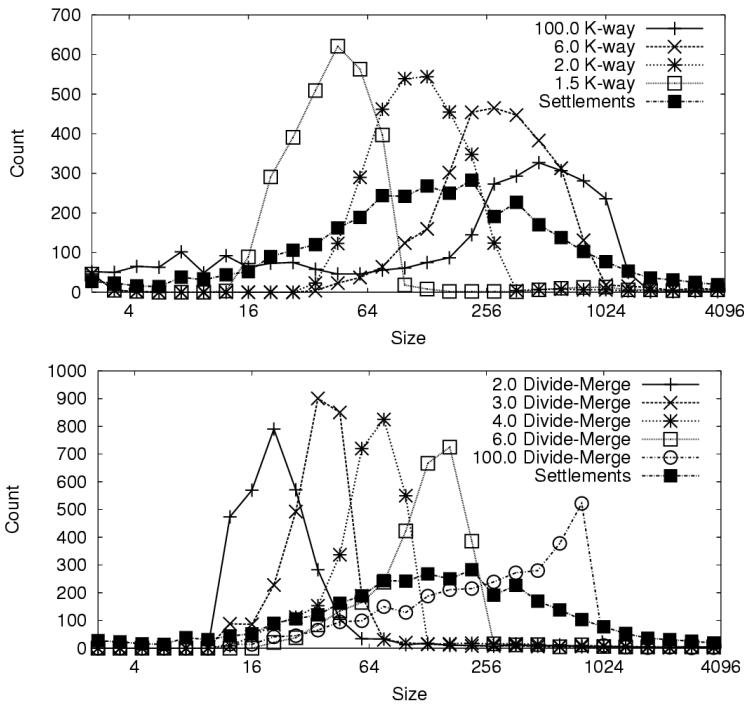


Figure 4: Effect of size limits on clustering quality, $k = 4$ and $d = 30$.



Algorithm & Limit	1.	2.	1. w/o Budapest
K-way 1.5	947571	134450	161515
K-way 2.0	828543	104421	86904
K-way 3.0	746869	63433	73683
K-way 4.0	746869	49240	61922
K-way 6.0	746869	55287	61922
K-way 100.0	712557	49577	60667
D-M 2.0	956415	643520	195418
D-M 3.0	817094	120746	163169
D-M 4.0	817094	120746	163169
D-M 6.0	725058	190964	150666
D-M 100.0	357618	297407	

Figure 5: Distribution of cluster sizes for k -way hierarchical partitioning (k -way) and Divide-and-Merge (Divide-Merge) for various parameters of

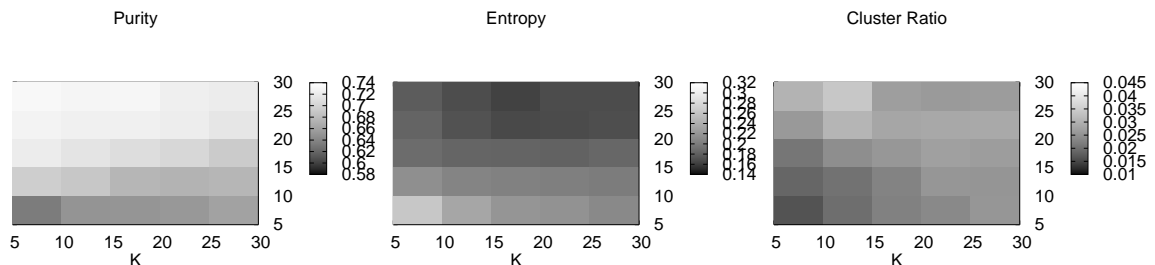


Figure 6: Relation between dimensions d (vertical), branching k (horizontal) and quality (darkness) for purity (left), entropy (center) and cluster ratio (right). The darker the region, the better the clustering quality except for purity where large values denote good output quality.

parison of fast SVD approximations and experiments with graphs of even larger scale and in particular of the LiveJournal blogger social network. In its current form our Jaccard weighting scheme requires quadratic space; we believe a fingerprint based approximation such as [37] that can give weight to nonexisting edges will improve the clustering quality. Comparison with other clustering algorithms and in particular with a possible scalable implementation of the semidefinite programming based approaches of Lang [29, 30] also remain future work.

4. REFERENCES

- [1] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *Proceedings of the 32th ACM Symposium on Theory of Computing (STOC)*, pages 171–180, 2000.
- [2] C. J. Alpert and A. B. Kahng. Multiway partitioning via geometric embeddings, orderings, and dynamic programming. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 14(11):1342–1358, 1995.
- [3] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: a survey. *Integr. VLSI J.*, 19(1-2):1–81, 1995.
- [4] C. J. Alpert and S.-Z. Yao. Spectral partitioning: the more eigenvectors, the better. In *DAC '95: Proceedings of the 32nd ACM/IEEE conference on Design automation*, pages 195–200, New York, NY, USA, 1995. ACM Press.
- [5] W.-H. Au, K. C. C. Chan, and X. Yao. A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE Trans. Evolutionary Computation*, 7(6):532–545, 2003.
- [6] E. R. Barnes. An algorithm for partitioning the nodes of a graph. *SIAM Journal on Algebraic and Discrete Methods*, 3(4):541–550, Dec. 1982.
- [7] A. A. Benczúr, K. Csalogány, M. Kurucz, A. Lukács, and L. Lukács. Sociodemographic exploration of telecom communities. In *NSF US-Hungarian Workshop on Large Scale Random Graphs Methods for Modeling Mesoscopic Behavior in Biological and Physical Systems*, 2006.
- [8] M. W. Berry. SVDPACK: A Fortran-77 software library for the sparse singular value decomposition. Technical report, University of Tennessee, Knoxville, TN, USA, 1992.
- [9] P. K. Chan, M. D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. In *DAC '93: Proceedings of the 30th international conference on Design automation*, pages 749–754, New York, NY, USA, 1993. ACM Press.
- [10] D. Cheng, R. Kannan, S. Vempala, and G. Wang. On a recursive spectral algorithm for clustering from pairwise similarities. Technical report, MIT LCS Technical Report MIT-LCS-TR-906, 2003.
- [11] D. Cheng, S. Vempala, R. Kannan, and G. Wang. A divide-and-merge methodology for clustering. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 196–205, New York, NY, USA, 2005. ACM Press.
- [12] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences of the United States of America*, 99(25):15879–15882, 2002.
- [13] F. Chung, L. Lu, and V. Vu. Eigenvalues of random power law graphs. *Annals of Combinatorics*, 2003.
- [14] F. Chung, L. Lu, and V. Vu. Spectra of random graphs with given expected degrees. *Proceedings of National Academy of Sciences*, 100:6313–6318, 2003.
- [15] G. Cormode, P. Indyk, N. Koudas, and S. Muthukrishnan. Fast mining of massive tabular data via approximate distance computations. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, page 605, Washington, DC, USA, 2002. IEEE Computer Society.
- [16] K. C. Cox, S. G. Eick, G. J. Wills, and R. J. Brachman. Brief application description; visual data mining: Recognizing telephone calling fraud. *Data Min. Knowl. Discov.*, 1(2):225–231, 1997.
- [17] I. Derényi, G. Palla, and T. Vicsek. Clique percolation in random networks. *Physical Review Letters*, 94:49–60, 2005.
- [18] C. H. Q. Ding, X. He, and H. Zha. A spectral method to separate disconnected and nearly-disconnected web graph components. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 275–280, New York, NY, USA, 2001. ACM Press.
- [19] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 107–114, Washington, DC, USA, 2001. IEEE Computer Society.
- [20] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, Sept. 1973.
- [21] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, pages 9–33, 2004.
- [22] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98), 1973.
- [23] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low rank approximations. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 370–378, 1998.
- [24] E. Gorny. Russian livejournal: National specifics in the development of a virtual community. pdf online, May 2004.
- [25] L. W. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.
- [26] R. Kannan, S. Vempala, and A. Vetta. On clusterings — good, bad and spectral. In *IEEE:2000:ASF*, pages 367–377, 2000.
- [27] G. Karypis. CLUTO: A clustering toolkit, release 2.1. Technical Report 02-017, University of Minnesota, Department of Computer Science, 2002.
- [28] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. Structure and evolution of blogspace. *Commun. ACM*,

- 47(12):35–39, 2004.
- [29] K. Lang. Finding good nearly balanced cuts in power law graphs. 2004.
- [30] K. Lang. Fixing two weaknesses of the spectral method. In *NIPS '05: Advances in Neural Information Processing Systems*, volume 18, Vancouver Canada, 2005.
- [31] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *Int. J. Comput. Vision*, 43(1):7–27, 2001.
- [32] M. Meila and J. Shi. A random walks view of spectral segmentation. In *AISTATS*, 2001.
- [33] A. A. Nanavati, S. Gurumurthy, G. Das, D. Chakraborty, K. Dasgupta, S. Mukherjea, and A. Joshi. On the structural properties of massive telecom graphs: Findings and implications. In *CIKM*, 2006.
- [34] J. P. Onnela, J. Saramaki, J. Hyvonen, G. Szabo, D. Lazer, K. Kaski, J. Kertesz, and A. L. Barabasi. Structure and tie strengths in mobile communication networks, Oct 2006.
- [35] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70, New York, NY, USA, 2002. ACM Press.
- [36] T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.
- [37] T. Sarlós, A. A. Benczúr, K. Csalogány, D. Fogaras, and B. Rácz. To randomize or not to randomize: Space optimal summaries for hyperlink analysis. In *Proceedings of the 15th International World Wide Web Conference (WWW)*, pages 297–306, 2006. Full version available at <http://www.ilab.sztaki.hu/websearch/Publications/>.
- [38] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.
- [39] U. von Luxburg, O. Bousquet, and M. Belkin. Limits of spectral clustering. pages 857–864, Cambridge, MA, 2005. MIT Press.
- [40] C.-P. Wei and I.-T. Chiu. Turning telecommunications call details to churn prediction: a data mining approach. *Expert Syst. Appl.*, 23(2):103–112, 2002.
- [41] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *ICCV (2)*, pages 975–982, 1999.
- [42] G. J. Wills. NicheWorks — interactive visualization of very large graphs. *Journal of Computational and Graphical Statistics*, 8(2):190–212, 1999.
- [43] P. Zakharov. Structure of livejournal social network. In *Proceedings of SPIE Volume 6601, Noise and Stochastics in Complex Systems and Finance*, 2007.
- [44] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Spectral relaxation for k-means clustering. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *NIPS*, pages 1057–1064. MIT Press, 2001.