Lecture Notes in Artificial Intelligence 5439

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Haizheng Zhang Myra Spiliopoulou
Bamshad Mobasher C. Lee Giles
Andrew McCallum Olfa Nasraoui
Jaideep Srivastava John Yen (Eds.)

Advances in Web Mining and Web Usage Analysis

9th International Workshop
on Knowledge Discovery on the Web, WebKDD 2007
and 1st International Workshop
on Social Networks Analysis, SNA-KDD 2007
San Jose, CA, USA, August 12-15, 2007
Revised Papers

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Haizheng Zhang
One Microsoft Way, Redmond, WA, USA - E-mail: hazhan@microsoft.com

Myra Spiliopoulou
Otto von Guericke University, Magdeburg, Germany
E-mail: myra@iti.cs.uni-magdeburg.de

Bamshad Mobasher
DePaul University, Chicago, IL, USA - E-mail: mobasher@cti.depaul.edu

C. Lee Giles
Pennsylvania State University, University Park, PA, USA - E-mail: giles@ist.psu.edu

Andrew McCallum
University of Massachusetts, Amherst, MA, USA - E-mail: mccallum@cs.umass.edu

Olfa Nasraoui
University of Louisville, Louisville, KY, USA - E-mail: olfa.nasraoui@louisville.edu

Jaideep Srivastava
University of Minnesota, Minneapolis, MN, USA - E-mail: srivasta@cs.umn.edu

John Yen
Pennsylvania State University, Unversity Park, PA, USA
E-mail: jyen@ist.psu.edu

Library of Congress Control Number: 2009921448

CR Subject Classification (1998): I.2, H.2.8, H.3-5, K.4, C.2

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-642-00527-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-00527-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12624300 06/3180 5 4 3 2 1 0

Preface

This year's volume of Advances in Web Mining and Web Usage Analysis contains the postworkshop proceedings of a joint event, the 9th International Workshop on Knowledge Discovery from the Web (WEBKDD 2007) and the First SNA-KDD Workshop on Social Network Analysis (SNA-KDD 2007). The joint workshop on Web Mining and Social Network Analysis took place at the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). It attracted 23 submissions, of which 14 were accepted for presentation at the workshop. Eight of them have been extended for inclusion in this volume.

WEBKDD is one of the most traditional workshops of the ACM SIGKDD international conference, under the auspices of which it has been organized since 1999. The strong interest for knowledge discovery in the Web, fostered not least by WEBKDD itself, has led to solutions for many problems in the Web's premature era. In the meanwhile, the Web has stepped into a new era, where it is experienced as a *social medium*, fostering interaction among people, enabling and promoting the sharing of knowledge, experiences and applications, characterized by group activities, community formation, and evolution. The design of Web 2.0 reflects the social character of the Web, bringing new potential and new challenges. The 9th WEBKDD was devoted to the challenges and opportunities of mining for the social Web and promptly gave rise to the joint event with the First Workshop on Social Network Analysis (SNA-KDD).

Social network research has advanced significantly in the last few years, strongly motivated by the prevalence of online social websites and a variety of large-scale offline social network systems. These social network systems are usually characterized by complex network structures and by rich contextual information. Researchers are interested in identifying common static topological properties of these networks, as well as the dynamics pertaining to formation and evolution. Social network analysis becomes necessary in an increasing number of application domains, including community discovery, recommendation systems, and information retrieval.

The objective of the joint WEBKDD/SNA-KDD 2007 workshop was to foster the study and interchange of ideas for the analysis and understanding of the social Web as the largest example of a social network.

Social networking on the Web is a phenomenon of scientific interest per se; there is demand for flexible and robust community discovery technologies, but also for interdisciplinary research on the rules and behavioral patterns that emerge and characterize community formation and evolution. The social flair of the Web poses new challenges and brings new opportunities for the individual. Among other things, the need for information now encompasses more than the traditional plain document search, as people started getting informed in blogs, as well as contributing with content, ratings, and recommendations to the

satisfaction of the information needs of others. Data miners are expected to deliver solutions for searching, personalizing, understanding, and protecting these social structures, bearing in mind their diversity and their scale.

The WEBKDD/SNA-KDD workshop invited research results on the emerging trends and industry needs associated with the traditional Web, the social Web, and other forms of social networking systems. This included data mining advances on the discovery and analysis of communities, on personalization for solitary activities (like search) and social activities (like discovery of potential friends), and on the analysis of user behavior in social structures (like blogs).

In the first paper *Spectral Clustering in Social Networks*, Miklós Kurucz, András A. Benczúr, Károly Csalogány, and László Lukács study large graphs of interconnected entities like phonecall networks and graphs of linked Web pages. They study the potential of spectral clustering for the identification of modular and homogeneous clusters and propose heuristics that alleviate shortcomings of the basis method and yield better results with respect to homogeneity and to the distribution of cluster sizes.

In the second paper *Looking for Great Ideas: Analyzing the Innovation Jam*, Wojciech Gryc, Mary Helander, Rick Lawrence, Yan Liu, Claudia Perlich, Chandan Reddy, and Saharon Rosset of IBM T.J. Watson Research Center report on methods for the analysis of the *Innovation Jam*. IBM introduced this online discussion forum in 2006, with the objective of providing a platform where new ideas were fostered and discussed among IBM employees and some external participants. The authors report on their findings about the activities and the social formations within this forum, and about their methods for analyzing the graph structure and the contributed content.

The third paper *Segmentation and Automated Social Hierarchy Detection through Email Network Analysis* by Germán Creamer, Ryan Rowe, Shlomo Hershkop, and Salvatore J. Stolfo studies the potential of data mining in corporate householding. The task is the identification of patterns of communication and the ranking of relationships among persons that communicate electronically, in particular through email. The authors have analyzed the Enron mailserver log and compared their findings with the human-crafted knowledge about the relationships of major players in that corporation.

The fourth paper *Mining Research Communities in Bibliographical Data* by Osmar R. Zaïane, Jiyang Chen, and Randy Goebel studies the implicit relationships among entities in a bibliographic database. Bibliographic data are of paramount importance for a research community, but the understanding of the underlying social structure is not straightforward. The authors have studied the DBLP database and designed the *DBConnect* tool. *DBConnect* uses random walks to identify interconnected nodes, derive relationships among the individuals/authors that correspond to these nodes, and even formulate recommendations about research cooperations among network members.

In the fifth paper *Dynamics of a Collaborative Rating System*, Kristina Lerman studies the Web as a participatory medium, in which users contribute, distribute, and evaluate information and she investigates collaborative decision

taking in the news aggregator platform Digg. Decision taking refers to the selection of the front-page stories featured regularly by Digg. This selection is based on the preferences of individual users, so the author investigates how a user influences other users and how this influence changes when a user contributes new content and obtains new friends.

In the sixth paper *Applying Link-Based Classification to Label Blogs*, Smitri Bhagat, Graham Cormode, and Irina Rozenbaum study the challenge of object labeling in blogs, thereby exploiting the links used by bloggers to connect related contents. They model this task as a graph labeling problem, for which they propose generic solutions. They then apply these solutions to the issue of blog labeling, whereby they are not only considering content but also the profiles of the bloggers themselves, attempting to assess the similarity of bloggers with respect to specific properties, such as age and gender, by studying the graph structure in which they participate.

In the seventh paper *Why We Twitter: An Analysis of a Microblogging Community*, Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng study the microblogging platform Twitter to understand the motives of users who choose microblogging for communication and information sharing. They identify four categories of microblogger intention, as well as different user roles within Twitter. They stress the differences between blogging and microblogging and compare the statistics of traffic in Twitter with those of blogs and other social networks.

In the last paper *A Recommender System Based on Local Random Walks and Spectral Methods*, Zeinab Abbassi and Vahab S. Mirrokni study interlinked blogs and propose a recommendation system for blogs that exploits this link structure. They observe the blogs as nodes of a social network, design a metric of similarity among them and devise also a *personalized* rank vector that expresses the relevance among nodes in the social network. They analyze the blog network, identify connected and strongly connected components and propose two algorithms that use this structure to formulate recommendations to a user.

August 2007

Haizheng Zhang
Myra Spiliopoulou
Bamshad Mobasher
C. Lee Giles
Andrew McCallum
Olfa Nasraoui
Jaideep Srivastava
John Yen

Organization

Workshop Chairs

Haizheng Zhang	Microsoft, USA
Myra Spiliopoulou	Otto von Guericke University Magdeburg, Germany
Lee Giles	Pennsylvania State University, USA
Andrew McCallum	University of Massachusetts, Amherst, USA
Bamshad Mobasher	DePaul University, USA
Olfa Nasraoui	University of Louisville, USA
Jaideep Srivastava	University of Minnesota, USA
John Yen	Pennsylvania State University, USA

Program Committee

Lada Adamic	University of Michigan
Sarabjot S. Anand	University of Warwick
Ricardo Baeza-Yates	Yahoo Research & Univ. Pompeu Fabra-Barcelona
Arindam Banerjee	University of Minnesota
Bettina Berendt	HU Berlin
Ed Chi	Xerox PARC
Tina Eliassi-Rad	Lawrence Livermore National Laboratory
Lise Getoor	University of Maryland
Joydeep Ghosh	University of Texas
Mark K. Goldberg	Rensselaer Polytechnic Institute
Andreas Hotho	University of Kassel
David Jensen	University of Massachusetts, Amherst
Ke Ke	Central Washington University
Ravi Kumar	Yahoo Research
Mark Last	Ben-Gurion University
Victor Lesser	University of Massachusetts, Amherst
Jure Leskovec	Carnegie Mellon University
Mark Levene	Birkbeck University of London
Ee-Peng Lim	Nanyang Tech. University, Singapore
Huan Liu	Arizona State University
Sanjay Kumar Madria	University of Missouri-Rolla
Ernestina Menasalvas	University Polytecnica Madrid, Spain
Dunja Mladenic	J. Stefan Institute, Slovenia
Alex Nanopoulos	Aristotle University, Greece
Seung-Taek Park	Yahoo! Research

Srinivasan Parthasarathy	Ohio State University
Jian Pei	Simon Fraser University, Canada
Xiaodan Song	NEC Labs America
Chris Volinsky	AT&T Labs-Research
Stefan Wrobel	Fraunhofer IAIS
Xifeng Yan	IBM Research
Mohammed Zaki	Rensselaer Polytechnic Institute
Alice Zheng	Carnegie Mellon University

Table of Contents

Spectral Clustering in Social Networks*

Miklós Kurucz, András A. Benczúr, Károly Csalogány, and László Lukács

Data Mining and Web search Research Group, Informatics Laboratory
Computer and Automation Research Institute of the Hungarian Academy of Sciences
{mkurucz,benczur,cskaresz,lacko}@ilab.sztaki.hu

Abstract. We evaluate various heuristics for hierarchical spectral clustering in large telephone call and Web 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, no general solution provided so far. 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 and Web crawls. We measure clustering quality both by modularity as well as by the geographic and topical homogeneity of the clusters. Compared to divide-and-merge, we give more homogeneous clusters with a more desirable distribution of the cluster sizes.

Keywords: spectral clustering, telephone call graph, social network mining, Web graph.

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 [38]: 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

* Support from a Yahoo Faculty Research Grant and by grant *ASTOR* NKFP 2/004/05. This work is based on an earlier work: Spectral Clustering in Telephone Call Graphs, in Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, Pages 82–91 (2007) (C) ACM, 2007. http://doi.acm.org/10.1145/1348549.1348559

H. Zhang et al. (Eds.): WebKDD/SNA-KDD 2007, LNCS 5439, pp. 1–20, 2009.

behave similar to other social networks such as those of bloggers and our results may be used in a more general setting. As additional experiments we included a measurement on the LiveJournal blogger network where we showed the hardness of the spectral clustering task as well as identified the well-known Russian user group [25,47] in Section 3.4. We also partitioned the UK2007-WEBSPAM host graph where our algorithm was able to identify meaningful clusters while baseline algorithms completely failed.

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 [28], in our graphs the "right" clustering is by no means obvious but, similar to the findings of [28], 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 [32]. 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 [31,32]. 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 [30, and references therein] or the World Wide Web [19, and many others]. Few exceptions include a theoretical analysis of connected components and eigenvalues [1,15,14] and several churn prediction by machine learning methods on real data [44,5, etc.]. Closest to our results are the structural investigations of mobile telephone call graphs [35,36] and the sketch-based approximate k-means clustering of traffic among AT&T collection stations over the United States [16]; 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: [46] reports visualization on graphs close to the size of ours with efficient algorithms to select the neighborhood subgraph to be visualized. In addition [17] 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 [18] 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 [11] for document collections.

The applicability of spectral methods to graph partitioning is observed in the early 70's [23,21]. The methods are then rediscovered for netlist partitioning, an area related to circuit design, in the early 90's [10,2,4,3]. Since the "Spectral Clustering Golden Age" [20,34,48, etc] 2001 we only list a random selection of results. Spectral clustering is applied for documents [48,11,12] as well as image processing [41,33,34], see many earlier references in [28]. More recently, several approximate SVD algorithms appeared [24,22,39, 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 [12] runs experiments over 18,000 nodes and 1.2 millions of nonzeroes 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. [12] 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 [41] and [20] 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 [41,19] both described, their performance is not compared in practice; Weiss [45] reports "unless the matrix is normalized [. . .] it is nearly impossible to extract segmentation information" but no performance measures are given; finally [43] 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 [28, 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 [20].
- We partially justify previous suggestions to use several eigenvectors [4,33]; 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 a 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 [23] 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 [27] 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. [10], much later Zha et al. [48] 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 [41,20]. The Laplacian arises as the relaxation of the minimum ratio cut [27]; weighted Laplacian appears in the relaxation of normalized cut [41] and min-max cut [20].

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. [27] suggest fast Lanczos-type methods as robust basis for computing heuristic ratio cuts; others [28,12] use power iteration. Since the SVD algorithm itself has no effect on the surrounding clustering procedure, we only compare performances later in Section 3.6.

2.1 Overview of the Algorithm

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 [12].

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 [13] into one cluster and keeps the dense component in one [32]. 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 [12] 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

We give a subroutine to reject 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 a cluster (that may be the entire graph) into at least two clusters $C_1 \cup \ldots \cup C_k$, we first form the connected components of each C_i and select the largest C_i'. We consider vertices in $C_i - C_i'$ outliers. In addition we impose a relative threshold `limit` and consider the entire C_i outlier if C_i' is below limit.

Next we redistribute outliers and check if the resulting clustering is sensible. In one step we schedule a single 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

Algorithm 1. `redistribute`$(C_1, \ldots, C_k)$: Small cluster redistribution

for all C_i **do**
  $C_i' \leftarrow$ largest connected component of C_i
  if $|C_i'| <$ `limit` $\cdot\, |C_1 \cup \ldots \cup C_k|$ **then**
    $C_i' \leftarrow \emptyset$
Outlier $= (C_1 - C_1') \cup \ldots \cup (C_k - C_k')$
for all $v \in$ Outlier **do**
  $p(v) \leftarrow j$ with largest total edge weight $d(v, C_j')$
for all $v \in$ Outlier **do**
  Move v to new cluster $C_{p(v)}$
return all nonempty C_i

is independent of the order vertices v are processed. By this procedure we may be left with less than k components; we will have to reject clustering if we are left with the entire input as a single cluster. In this case we either try splitting it with modified parameters or completely give up forming subclusters.

2.3 *K*-Way Hierarchical Clustering

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 do 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.

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
  $v_i \leftarrow v_i'/||v_i'||$
  $(C_1, \ldots, C_k) \leftarrow$output of k-means$(v_1, \ldots, v_{|C_0|})$
  Call `redistribute`$(C_1, \ldots, C_k)$
  Discard C_0 if C_0 remains a single cluster

2.4 Divide-and-Merge Baseline

The Divide-and-Merge algorithm of Cheng et al. [12] 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 $\texttt{cnum}_0$ 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 [12], 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 [12] 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 $1/\texttt{ratio\_init}$–$(1 - 1/\texttt{ratio\_init})$ cut and gradually decrease the

Algorithm 3. Divide and Merge: Divide Phase

while we have less than $\texttt{cnum}_0$ 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
  $i \leftarrow \texttt{ratio\_init}$
  while C_0 is not discarded **do**
    Find $1/i \leq t \leq 1 - 1/i$ such that the cut

$$(S, T) = (\{1, \ldots, t \cdot n\}, \{t \cdot n + 1, \ldots, n\})$$

    minimizes the cluster ratio
    $(C_1, \ldots, C_\ell) \leftarrow \texttt{redistribute}(S, T)$
    if $\ell > 1$ **then**
      Discard C_0 and add clusters $C_1, \ldots, C_\ell$
    else
      if $i = 3$ **then**
        Discard cluster C_0
      else
        $i \leftarrow i - 1$

Algorithm 4. Merge Phase

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

$$\mathrm{OPT}_n(C_0, i) = \mathrm{numer}(i_1, \ldots, i_\ell); \quad \mathrm{OPT}_d(C_0, i) = \mathrm{denum}(i_1, \ldots, i_\ell)$$

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.

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 $k = 1, \ldots, m$. 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 [11]. 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 $i_1 + \ldots + i_\ell = i$ split sizes within subclusters $C_1, \ldots, C_\ell$. 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 $\mathrm{OPT}_n(C_j, i_j)$ and $\mathrm{OPT}_d(C_j, i_j)$. If we use the corresponding splits for all j, we obtain a split of cluster ratio

$$\frac{\sum_{j \neq j'} d(C_j, C_{j'}) + \sum_{j=1\ldots\ell} \mathrm{OPTn}(C_j, i_j)}{\sum_{j \neq j'} |C_j| \cdot |C_{j'}| + \sum_{j=1\ldots\ell} \mathrm{OPT_d}(C_j, i_j)}$$

for the union of the subcomponents. Note however that this expression is not monotonic in the cluster ratio of subcomponent j, $\mathrm{OPT}_n(C_j, i_j)/\mathrm{OPT}_d(C_j, i_j)$, 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 $i_1 + \ldots + i_\ell$.

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 [28]. Kannan et al. [28] 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 [27], the second the normalized [41] and min-max [20] 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 my 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 [28] 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 $1 + \mathrm{Jac}_{ij}$ 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 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 down-weights unimportant edges and adds "missing" calls to the network.

3 Experiments

In this section we describe our experiments performed mainly on the Hungarian Telecom call detail record and, in order to extend the scope of applicability, on the UK2007-WEBSPAM crawl and the LiveJournal blogger friends network.

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 for the Hungarian Telecom data, the largest of the graphs used in our experiments.

3.1 Evaluation Measures

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

In addition we tested *normalized network modularity* [42]

$$Q_{norm} = \sum_{\text{clusters } s} \frac{N}{N_k} \left[\left(\frac{d(C_s, \overline{C_s})}{2M} \right)^2 - \frac{d(C_s, C_s)}{M} \right]$$

where M is the total weight of the edges and $d(X, Y)$ is the weight of edges with tail in X and head in Y; we also use the notation of Section 2.4 for the sizes of the clusters. As we will see in Section 3.8, normalized modularity turned out instable and we suspect it may not be the appropriate measure for cluster quality.

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 Section 2.4 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 [29] (the latter also called *accuracy* in [11]) 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.

3.2 Detailed Call Record Data

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, left) 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.

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. 1, right, 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.