Generating an Optimal Editorial Board for a Research Journal

Bobby Ullman

with guidance from David Walker

Abstract

An ideal editorial board for a research journal covers all published topics, chooses leaders in these topics, and represents these topics proportionally. While traditional editorial boards are chosen by hand, we explore using the wealth of relationship data found in published works to generate an editorial board algorithmically. We model a journal by the coauthorship and citations between contributors in the journal and then apply stochastic blockmodeling on a graph derived from these relationships in order to topically cluster the contributors. By selecting leading members is each topic cluster we generate editorial boards that are optimized over our three editorial board criteria. We compare these generated editorial boards with actual editorial boards and find that in many cases our generated lists more closely represent the community.

Introduction

Science is driven by the publication and sharing of research. To this end, journals exists to publish results in a particular area of focus thus establishing a community of researchers that write in and read the journal. These journals not only serve as a sharing mechanism between research in similar fields, but also establish the face of that field to the outside world. As such, the content of these publications are critical in defining the field and its community.

The scientific review process has evolved to ensure the quality of this published content and legitimacy of the related field of study. Each journal selects people to review submitted material and to determine appropriate articles for publication. Since the concept of appropriateness is inherently subjective, the selection of these people has significant impact on which papers are published. Thus, the selection of this group is almost as important as the content of the submitted works.

In this paper, we focus on selecting editorial boards for journals and also apply our ideas to the analogous problem of choosing program committees for the POPL conference. Yet, selecting a

representative group from a publishing body has many other varied applications. For example, selecting The United States President's Council of Advisors on Science and Technology (PCAST) or the membership of a University department are both analogous problem. Regardless, in this paper we cast the problem as selecting an editorial boards for a journal.

Journal organizers aim to choose a editorial board that as a whole:

- covers all specialized subfields published in the journal;
- is comprised of leaders in those subfields;
- represents the subfields proportionally to the prevalence of the subfields in the journal; and
- satisfies any number of other factors such as diversity, personal relationships, etc.

Currently, most editorial boards are formed via social mechanisms where a small well-connected group of people uses its personal knowledge of the community as well as suggestions from colleagues to select people for the editorial board. This social mechanism is often augmented by considering authors of papers with desirable keywords or authors who frequently contribute to related publications [8]. This social mechanism often works well, but has a few potential dangers. First, it tends to focus only on the most famous or well-known people. Although this is often appropriate, sometimes slightly less well-known people are more representative of a particular subtopic. Second, if the top suggestions within a specialty are unavailable it may be unclear who else is a desirable candidate. Third, this social mechanism carries unintended biases of the group that may or may not reflect the desirability of candidates.

An alternative approach is to suggest an editorial board by algorithmic analysis of data. Our approach is to represent the important relationship data of a scientific community in a graph and then applying graph clustering algorithms and vertex centrality measures to break the community into clusters and select representative members from each cluster. These representative members can be viewed as topically representing the community and as such would be an appropriate selection for a journal editorial board serving this community. In our selection process we focus only on the first three editorial board criteria (coverage, leadership, proportional representation) as we aim only to provide a refined list of candidates to serve as a starting point for the current editorial board

selection process.

Stochastic Blockmodel Ensemble

Finding highly-connected groups of vertices (or communities) in a graph is a widely studied problem with many different approaches. A stochastic blockmodel clustering algorithm attempts to partition the N vertex set into B blocks so as to minimize some carefully chosen objective function. In practice, this is done efficiently by picking some vertex block assignment and by stochastically changing vertex assignments keeping only those changes that result in a decrease in the objective function. This stochastic annealing is repeated until a low objective steady state is reached. The details of the stochastic blockmodeling used in this paper to cluster scientific communities, outlined below, is efficiently implementing in C++ as part of the `graph-tool` python library [4].

Minimal Description Length [5]

Take a graph with N vertices each assigned to one of B blocks. When $a,b \in \{1,2,3,\ldots,B\}$ and $a \neq b$, let e_{ab} be the number of edges with one vertex in block a and the other vertex in block b. Let e_{aa} be twice the number of edges with both ends in block a. Let e_a be the number of edges with only one vertex in block a. Let n_a be the number of vertices in block a.

For an unweighted graph we can count the number of possible graphs with a given block assignment by

$$\Omega = \prod_{a \geq b} \Omega_{ab}, \text{ where } \quad \Omega_{ab} = \binom{n_a n_b}{e_{ab}} \text{ and } \Omega_{aa} = \binom{\binom{n_a}{2}}{e_{ab}/2}$$

We define an entropy function $S' = \ln \Omega$, which simplifies via Sterling's approximation to

$$S' = \frac{1}{2}\sum_{ab} e_{ab} H\left(\frac{e_{ab}}{n_a n_b}\right) \cong \frac{1}{2}\sum_{ab} e_{ab} - e_{ab} \ln\left(\frac{e_{ab}}{n_a n_b}\right)$$

where $H(x) = -x\ln(x) - (1-x)\ln(1-x)$, the binary entropy function. The quantity S' is known as the microcanonical entropy [2] of a block arrangement. We can impose a stricter constraint and only count possible graphs that have the same vertex degrees as the original graph. This yields an

analogous entropy function

$$S = -E - \sum_i n_i \ln(i!) - \frac{1}{2}\sum_{ab} e_{ab} \ln\left(\frac{e_{ab}}{e_a e_b}\right).$$

We apply this degree-constrained entropy function in weighted graphs even though there is no longer a counting interpretation.

Since S is strictly decreasing in B, optimizing S over B would always lead to the degenerate solution where $B = N$. As such we do not use S as our objective function and instead we define a new objective D where

$$D = S + L$$

and L measures the quantity of information in e_{ab} and vertex block assignments [7]. We think of e_{ab} as an adjacency matrix of a graph with B vertices and $E = \sum_{ab} e_{ab}/2$ edges. Thus there are $P = \binom{B}{2}$ possible edges in the graph and thus $\binom{P}{E}$ possible block adjacency matrices e. In addition there are B^N possible ways to partition the vertices of the original graph into blocks. We define L as the logarithm of the product of these two values, which simplifies to

$$L \cong EH\left(\frac{B(B+1)}{2E}\right) + N \ln B$$

where H is the same binary entropy function seen earlier. Since S is decreasing in B, but L is increasing in B, minimizing D over all block configurations with B variable will now yield a non-degenerate solution. We term D the *description length* since it captures the amount of information need to describe the blockmodel. As such minimizing D aligns with a rephrasing of Occam's razor known as the *minimum description length principle*, namely that the best model is one that minimizes the data needed to describe it.

Monte Carlo Interference

For fixed B we could minimize S directly just by examining each vertex and changing its block membership if that change decreased S. This approach is simple, but converges slowly to a stable

state. Instead we use the Markov chain Monte Carlo (MCMC) interference algorithm. This takes a random sample from the distribution of possible block assignments where low entropy assignments are more likely. This approach is not always guaranteed to give the best solution but is much more efficient than direct minimization of S.

The basic MCMC interference algorithm works as follows. Given a graph G and a number of blocks B, start by assigning all vertices in G randomly to one of the B blocks. Pick some vertex i at random. Say i is in block a. Select some other block b from the B possible choices with equal probability. Randomly select vertex j from the neighbor set of i and say j has block assignment c. Then we change the block assignment of i from a to b with probability P, a function of the block assignments a, b, c, and the change in entropy. Otherwise we give i the block assignment of the other end of a randomly select edge leaving block c. Now, another vertex is selected at random and this process is repeated.

The exact choice of the probability function P determines the probability distribution of possible partitions generated by this random process. We choose

$$P = \min\left\{ e^{-\beta \Delta S} \frac{\sum_c p_c^i p(a \to b|c)}{\sum_c p_c^i p(b \to a|c)}, 1 \right\} \; with \quad p(a \to b|c) = \frac{e_{bc}+1}{e_c + B}$$

where ΔS is the change in entropy if vertex i is changed from block a to b, p_c^i is the percent of edges leaving i that enter block c, and β is some large value. This choice of P samples low entropy states with much higher probability [6, 7]. Notice that for large β we have $P = 1$ whenever $\Delta S < 0$, meaning that we will accept all block assignments that decrease entropy.

Each MCMC sweep updates one vertex block assignment and can be done in $O(E)$ time. After each sweep we increase the value of β and we terminate after a number of successive sweeps that do not change the maximum or minimum block description length achieved.

Since we can minimize D for a fixed B with the above methods, we can minimize D over variable B by finding an interval (b_1, b_2) containing the the minimum D (over all B) and then successively bisecting that interval.

Community Graph Representation

In an effort to build an editorial board for a journal, we attempt to build a weighted graph whose vertex set is comprised of all authors who have contributed to that journal and whose edge weights indicate the degree of proximity between two authors. To generate a community graph for a journal, we examine the relationships in the past papers published in the journal. We let each author correspond to a vertex in our community graph and then use the coauthorship and citation data in published papers to indicate where to place edges between authors. Clearly, coauthoring a paper or citing another in a paper indicates some sort of proximity between the two authors, but it is less clear how to weight these connections or even how to combine the somewhat unrelated coauthorship and citation data into one graph. Regardless, we aim for our graph representation to place a high weight on edges between two authors that are closely related and a low weight on edges between two that are less related.

Edge Weight

Let Z be the set of all papers in our data set. We let w_{ab}, the weight of the edge between author a and author b, be

$$w_{ab} = \sum_{z \in Z} f_{ab}(z) + g_{ab}(z)$$

where f accounts for the relationship between a and b from coauthorship in paper z, while g accounts for the relationship between a and b from citation (either from a citing b or b citing a).

Notice that although it may seem plausible to try to interpret citation data as directed edges we do not ($g_{ab} = g_{ba}$). In other words, we assume that authors are equally connected to each other by the act of one of them citing each other. This is not to say that both authors involved in a citation are equally important in the community, in fact if vertex degree were meant to indicate importance of an author, instead of just network connection, it seems almost certain that the person being cited should receive a larger bonus than the citer.

In addition, we mix coauthorship and citation, arguably unrelated relationship data, because using

both together is more robust. Including citation data identifies topically related authors who have for some reason not published together. As such, using coauthorship data alone does not provide a full picture of the network around a specialization, which results in poor topical clusters. That said, coauthorship should not be ignored entirely as it places additional weight on the most important relationships in the graph.

It seems reasonable to assume that the authors on a highly collaborative paper, a paper with many coauthors, are less connected to each other. As such we let

$$f_{ab}(z) = \begin{cases} 0 & : n = 1 \\ \frac{\lambda}{n} & : n \neq 1 \end{cases}$$

where λ is a constant and n is the number of authors on paper z. We define $g_{ab}(z) = 0$ when a and b do not cite each other in z and otherwise $g_{ab}(z) = \delta$ some constant.

Average Representative Distance

Given a community graph G with vertex set V and a set of vertices $S \subseteq V$ selected to represent the community we may compute the average representation distance

$$\alpha_G(S) = \frac{1}{|V'|} \sum_{v \in V'} d_S(v)$$

where V' is the vertex set of the dominant component of G and $d_S(v)$ is the minimum distance from v to any vertex in S. Here we compute distance on the unweighted community graph G where each edge contributes 1 unit to the distance. If there is no path from v to S we remove v from V' in the calculation of α. We use the average representation distance to evaluate the quality of a chosen representative set. This is a reasonable evaluation heuristic for a representative set since it captures the first two editorial board criteria.

We choose to use clustering instead of minimizing this quantity directly for two main reasons. First, blockmodel clustering is much more efficient than minimizing this quantity directly. Mini-

mizing α directly is $O(V^n)$ where V is the number of vertices in the community graph and n is the number of representatives to select, since you may need to compute α for every set of $n > 2$ vertices. Our implementation of stochastic block modeling instead runs in $O(E \ln E)$. Even for dense graphs $E \leq V^2$ so this clustering approach has faster expected runtime for large graphs.

Second, clustering provides more information than a simple selection set that minimizes α, namely the clusters themselves. As a result these topical clusters allow us to select a representative group that proportionally represents the community as a whole (our third editorial board criterion).

Implementation

Data Mining

We implemented a simple data-miner `miner.py` that downloads clerical data for papers submitted to a conference tracked by the DBLP computer science bibliography [1] and produces a JSON file of the conference data. In addition the data-miner searchers the CiteSeerX citation database[3] for citations in the DBLP papers and adds the citations to the entry for each paper. For code and more detailed documentation see `www.github.com/rjullman/conf-miner`.

Note that a single author may be published under multiple similar names (e.g. Daniel H. Ullman, D. H. Ullman, D. Ullman, D. Howard Ullman). Since the data-miner and the clustering script are intended to be used together, there is some cursory effort by `cluster.py` to find and merge name discrepancies when building the community graph.

Clustering

The stochastic blockmodel clustering implementation `cluster.py` analyzes the paper JSON data from `miner.py` and suggests a editorial board. The entire program is built on top of `graph-tool`[4], a python library with core computational algorithms written in C++.

The program does four main things: (1) creates a community graph from compatible JSON data, (2) partitions the graph into clusters, (3) selects representative members from each cluster to represent the entire community, and (4) computes statistics on the quality of the chosen representative

set. The first is done as detailed earlier with $\lambda = 60$ in f_{ab}. This was chosen to ensure that edge weights stay above 1, a convenience for using `graph-tool`, and so that most edge weights are integral, a general convenience for debugging and inspection of the graph by hand. By default $\delta = 1$ in g_{ab}, but δ can be modified and even set to 0 from the command line. It seems reasonable to let only one of these constants be easily varied because the ratio between γ and δ, the influence of coauthorship versus citations in the graph, is the critical quantity.

For the clustering step we use the `minimize_blockmodel_dl` routine in `graph-tool`, an implementation of stochastic blockmodeling using MCMC interference to minimize graph description length over the number of clusters and cluster assignments. Since conference communities tend to be highly connected a single pass of the clustering algorithm tends to be too coarse, only breaking apart the most unrelated members of the communities instead of selecting those most related. As such we instead perform multiple passes on the graph, in each pass reapplying the clustering algorithm on the subgraphs induced by the clusters in the previous pass. The recursive clustering ends at either a fixed depth (selected by the user), a minimum cluster size (selected by the user), or upon reaching a stable state.

Given clusters we select representatives in each cluster based on their activity in the community. This is judged solely by the number of papers they have published. Originally there was some attempt to use traditional graph centrality measures on the cluster's induced subgraph, but this had a variety of critical pitfalls. First, influential members in the community at large were unfairly penalized as their contributions outside of their most active subfield were discounted because edges leaving the cluster are removed in the induced subgraph. Second, this graph, designed to capture relationship data, is badly tuned for detecting influence. For example, the inclusion of citation data would have to be greatly changed to promote the cited member without promoting the citer. In fact, although not in the current implementation, it seems reasonable to rank authors in each cluster by some combination of number of citations received and number of papers published instead of just the number of papers published alone. Regardless, improving cluster centrality measures would be a possible extension to our results.

The total number of representatives to select is fixed and the top representatives are chosen from each cluster proportional to the cluster's size in the overall community. We handle this apportionment problem by Hamilton's apportionment method, namely giving every cluster its fair share of the representatives rounded down and then giving the extra representatives to clusters with the highest fractional part of a deserved representative. This apportionment method was selected for its simplicity. In addition, slight changes in apportionment will only slightly affect our results since our quality heuristic, average representative distance, is robust to small variations in the representative, because our community graphs tend to have a large number of vertices.

The program also reports basic graph properties, the average representation distance of the chosen representative set, and the number of vertices disconnected from the representative set. Community graphs tend to have one large connected component with many tiny other components. For example, much of the data used for our analysis had around 15% of the vertices in small components containing 9 or fewer vertices. The remaining 85% of the vertices are in the single dominant component. These vertices tend to almost entirely constitute the unrepresented vertices and tend to have little contribution to the community as a whole. Yet, instead of restricting our community graph only to the largest connected component, we instead include all vertices initially, but exclude disconnected vertices from the average representation distance calculation and note their quantity.

The script also supports data visualization, query sets that allow filtering of the community graph before choosing representatives, and filter sets that allow restrictions on the chosen representative. For code and more detailed documentation see `www.github.com/rjullman/cluster`.

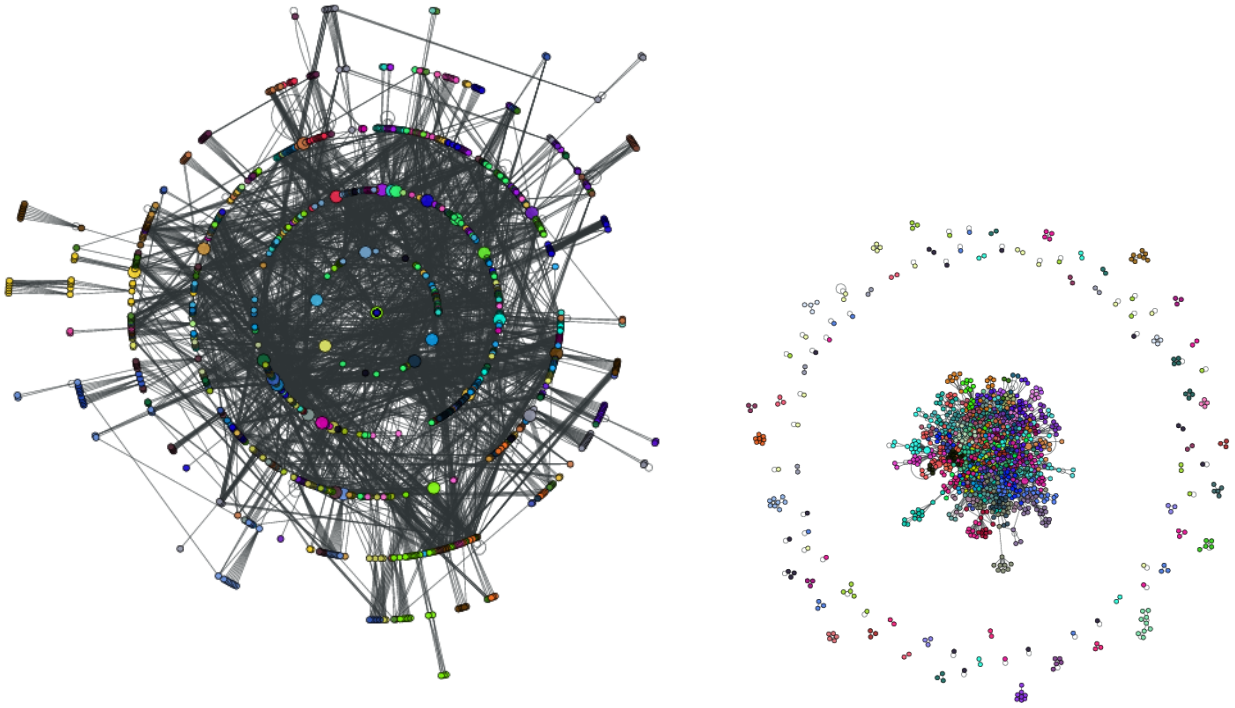
Evaluation

We examine the Principles of Programming Languages (POPL) conference community graph including papers published from 2000 to 2013 (see Appendix POPL) to evaluate our community graph construction and selected representative set. As a well-established conference, POPL serves as a standard set for reflecting on our clustering method as well as generalizing about the structure

of other similarly established conferences.

POPL's Structure

In the past 13 years, POPL has published close to 600 papers referencing 1400 contributors. On average a paper in POPL has 1.55 authors per paper and 2.33 citations. Of the 1400 contributors 970 have published at least 1 paper in POPL while the remaining 430 authors are only cited in POPL. This reaffirms that POPL, like any well-established scientific community, draws heavily (nearly 30%) from ideas central in other communities.



(a) Radial Layout (b) Component Layout

Figure 1: POPL community graph displayed radially with (a) the highest degree vertex in the center and (b) display in component clusters. Vertices in the same color are clustered together and larger vertices are selected in the representative set. These visuals are less useful for closely inspecting the graph (for that see Appendix POPL), but are informative about the general topology of the community. In particular, (a) captures general distance measures since vertices are layed out so that a majority of edges connect vertices in adjacent rings, while (b) captures overall connectivity and component density.

The highest-degree vertex in the dominant component of the community graph is a distance of 6

or less to all other vertices in that component (figure 1a). Assuming a roughly even distribution of vertices we can achieve an average representation distance of below 3 with a representation set containing only one member, which serves as an expected maximum for any reasonably chosen representative sets with more members. Furthermore, around a third of vertices lie within a much denser area of the graph with a distance of 3 from this high degree vertex (viewed visually as the inner two rings in figure 1a).

The graph has roughly 100 connected components and all but one have fewer than 10 vertices (figure 1b). The smaller disconnected components account for roughly 200 vertices and so we would expect even a well chosen representative set for POPL to not represent roughly 15% of the graph. Given that 30% of the graph is not even publishing in the POPL network we would not expect a well-chosen representative board to represent these tangential fields.

The Citation Coauthor Constant Ratio

The edge weight functions for coauthor edges and citations edges are determined by constants λ and δ (respectively). The ratio λ/δ indicates the relative contribution of citation data and coauthor data in the block description length computed when finding clusters. We find that a ratio between 10 and 60 produces a significantly better representative set where as relying too heavily on either coauthorship or citation data alone produced a lower quality representation (figure 2). This supports the theory that a single coauthorship contains more important relationship information in the community than a single citation. Yet, since there are many more citations than authors in a paper, the overall important of each is comparable. In our graph the ratio of number of citation and coauthor relationships is only about 2.2, but this number is much lower than it should be given that the citation information on many papers could not be found by `miner.py`.

Comparison to Actual Editorial Boards

We computed programmatically a suggested program committee for POPL each of the last 4 years and compared that to the actual chosen committees. We find that the generated committee has a significantly lower average representation distance than the actual committees. Yet, this is not quite

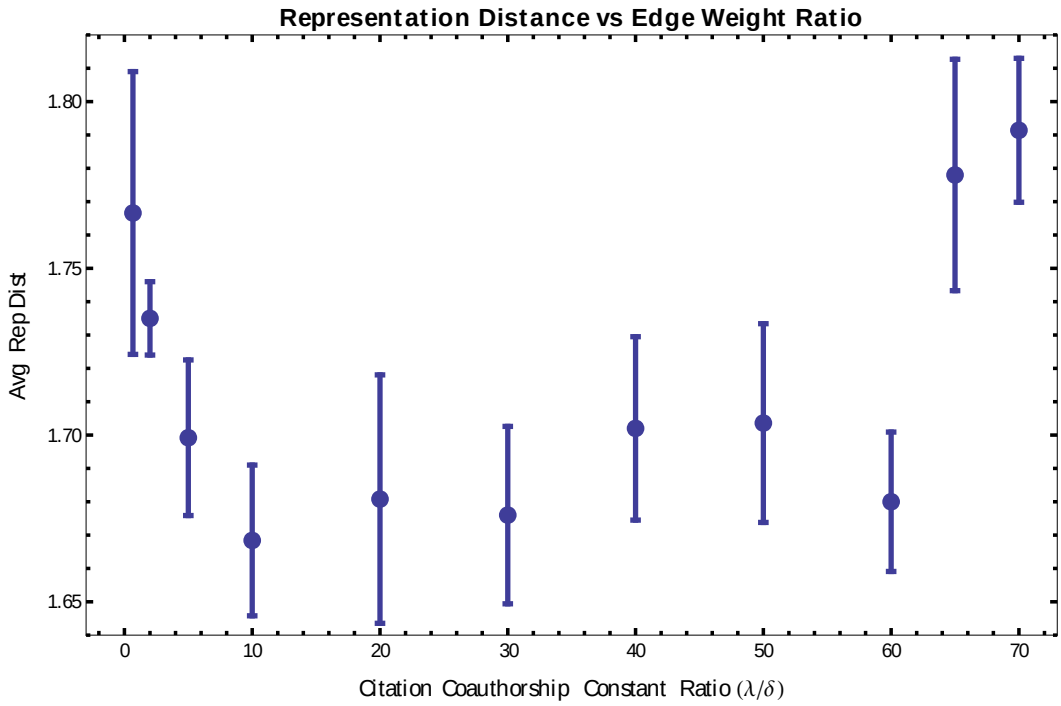
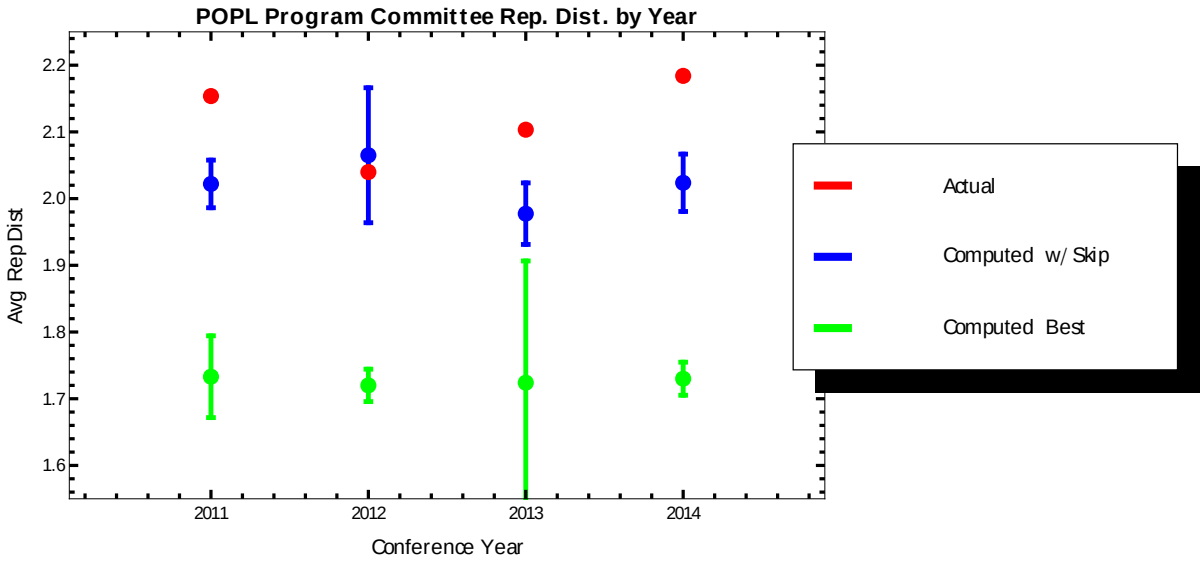


Figure 2: Shows average representation distance to representation set computed at different ratios λ/δ. Since the clustering algorithm is stochastic in nature the exact result changes when run on the same data. As such the above numbers are found by repeating the algorithm 6 times at each ratio. As seen in the figure, we find significantly lower representation distances when the ratio is somewhere between 10 and 60.

a fair comparison since the generated list is free to include people regardless of whether or not they would actual accept an invitation to join the committee. In particular, it seems unlikely that all the most influential people in each subfield would be willing to participate. To that end we also generated a review board skipping the top 2 choices for the representative set in each cluster. These sets filter out the most well-known contributors in each field making the suggested committee a feasible real-world choice. The average representation distance of this committee still tend to be slightly better than that of the committees that were actually chosen (figure 3).

Comparing the program committee for POPL 2014, a list of 26 people, with the suggested representative set found programmatically (see Appendix POPL) we find that 2 people, namely Andrew Appel and Stephanie Weirich, were selected on both lists. Additionally, 8 people on the actual 2014 committee were top alternative choices in the representative set. We place little significance on these exact numbers since it is unclear even using our average representation



	Average Rep. Distance ($\pm\sigma$)			
Year	2011	2012	2013	2014
Actual	2.15	2.04	2.10	2.18
Computed w/ Skip	2.02	2.07	1.98	2.02
Computed Best	1.73	1.72	1.72	1.73

Figure 3: The average representation distance for the actual POPL program committees and the generated committees for the past 4 years. The generated committees with skip (blue) were found by ignoring the 2 top ranked members in each cluster when generating the committee.

distance which list is actually a better choice. Yet even if people do not want to switch entirely to this automated system, they could still use this generated list as a starting point for selecting candidates by hand. The data in table 3 suggests that choosing candidates in this way yields a lower average representation distance. In addition, the generated list of proposed candidates is grouped by cluster making it easier to ensure proportional representation even when refining the generated list by hand (see Appendix POPL).

Final Thoughts

Topical clustering of the community network to produce a representative set yields results quite similar to a typical review board. The best way to evaluate the quality of this selection, much better than our roughly quality heuristic (average representation distance), is to see if the generated lists are useful to conference organizers in practice. We hope that, if used, these generated selections will

help identify both subfield leaders unknown by the conference organizer and appropriate alternates when a first choice selection is unavailable.

There are many directions one can go from here. The POPL data set serves as a great reference for small to medium-size communities, but the next logical step is to apply these clustering techniques to much larger graphs. This could be done by selecting a more frequently published journal, for instance Science Magazine, or by combining a group of related conferences. In addition, it is worth exploring other edge weighting functions besides our chosen g and f. A particularly interesting case would be to ignore coauthorship data in favor of generalizing the notion of a citation. This could involve making citation weight scale with the number of people in the cited paper much as coauthorships did in this paper. In addition, this construction could also be made to encapsulate coauthorship data by having authors effectively "cite" their own papers. This approach balances coauthorship and citation data into one generalization that simplifies (and perhaps improves) the current model. Another area for expansion is in how we find the most important members in each cluster. It seems natural to use the conference data again, although with a different weighting scheme, to determine important vertices. For example, this new graph could use directional edges to capture the important directionality of citations information. Finally, we could examine other representation quality heuristics other than the average representation distance.

References

[1] "Dblp computer science bibliography," http://www.dblp.org/db/, [Online].

[2] G. Bianconi, "Entropy of network ensembles," *Phys. Rev. E*, vol. 79, p. 036114, Mar 2009. Available: http://link.aps.org/doi/10.1103/PhysRevE.79.036114

[3] S. Lawrence and K. Bollacker, "Citeseerx digital library engine," http://citeseer.ist.psu.edu/, 2007-2013, [Online].

[4] T. P. Peixoto, "Graph tool," http://graph-tool.skewed.de/, 2009, [Online].

[5] T. P. Peixoto, "Entropy of stochastic blockmodel ensembles," *Physical Review E*, vol. 85, no. 5, p. 056122, 2012.

[6] T. P. Peixoto, "Efficient monte carlo and greedy heuristic for the inference of stochastic block models," 2013.

[7] T. P. Peixoto, "Parsimonious module inference in large networks," *Physical Review Letters*, vol. 110, no. 14, p. 148701, 2013.

[8] F. Tip, "PC-Miner," https://github.com/franktip/pcminer, 2010, [Online].

Appendix - POPL Conference

The Principles of Programming Languages (POPL) conference serves as a good test for `miner.py` and `cluster.py`. The data and graph (figure 4) below were produced with the following queries:

```
» python miner.py popl -f popl.dat -l 21
```

```
» python cluster.py popl.dat -n 30 -o popl30reps.pdf -di 1000 -a 5 > popl30reps
```

popl30reps

The marker '–>' indicates that that person was found on the actual POPL 2014 program committee.

```
--------------------
Cedric Fournet
A: Andrew D. Gordon
A: Martin Abadi
A: Karthikeyan Bhargavan
A: Georges Gonthier
A: Gordon D. Plotkin
--------------------
Frank Pfenning
A: Sungwoo Park
A: Didier Remy
A: John C. Mitchell
A: Davide Ancona
--> A: Elena Zucca
--------------------
Amir M. Ben-Amram
A: Amr Sabry
A: Kevin Hammond
A: Neil D. Jones
A: Andrea Asperti
A: Samir Genaim
--------------------
Naoki Kobayashi 0001
A: Hiroshi Unno
A: Tachio Terauchi
--> A: Atsushi Igarashi
A: Jeffrey Mark Siskind
A: Barak A. Pearlmutter
--------------------
Andrew Kennedy
--> A: Nick Benton
A: Don Syme
A: Patricia Johann
A: Anindya Banerjee
--> A: Neelakantan R.
       Krishnaswami
--------------------
Giuseppe Castagna
A: Philip Wadler
A: Jerome Simeon
--> A: Jeremy G. Siek
A: Janis Voigtlander
A: Michele Bugliesi
--------------------
Nobuko Yoshida
A: Davide Sangiorgi
A: Marco Carbone
A: Robin Milner
A: Xinyu Feng
A: Kohei Honda
--------------------
Cormac Flanagan
A: Andrew M. Pitts
A: Yannis Smaragdakis
A: Ondrej Lhotak
A: Steffen Losch
A: Thomas H. Austin
--------------------
Thomas A. Henzinger
A: Ranjit Jhala
A: Rupak Majumdar
--> A: Andrey Rybalchenko
A: Patrick Maxim Rondon
A: Ashutosh Gupta
--------------------
Francois Pottier
A: Sam Staton
A: Paul Blain Levy
A: Benjamin Delaware
A: Bruno C. d. S. Oliveira
A: Tom Schrijvers
--------------------
Derek Dreyer
A: Umut A. Acar
A: Amal Ahmed
A: Chung-Kil Hur
A: Georg Neis
A: Andreas Rossberg
--------------------
Suresh Jagannathan
--> A: Viktor Vafeiadis
A: Matthew Fluet
A: Jan Vitek
A: Ruy Ley-Wild
A: Aleksandar Nanevski
--------------------
Peter Sewell
A: Susmit Sarkar
A: Francesco Zappa Nardelli
A: Mark Batty
A: Scott Owens
A: Magnus O. Myreen
--------------------
Hans-Juergen Boehm
A: Sarita V. Adve
A: Tatiana Shpeisman
A: Leaf Petersen
A: Jeremy Manson
A: William Pugh
--------------------
P. Madhusudan
A: Rajeev Alur
```

A: Pavol Cerny
A: Swarat Chaudhuri
A: Xiaokang Qiu
A: Gennaro Parlato

Simon J. Gay
A: Nils Gesbert
A: Luis Caires
A: Joao Costa Seco
A: Chandrasekhar Boyapati
A: Karl Naden

Benjamin C. Pierce
A: Martin Hofmann 0001
A: Alan Schmitt
A: Marco Gaboardi
A: Daniel Wagner
--> A: J. Nathan Foster

Simon L. Peyton Jones
A: Manuel M. T. Chakravarty
A: Gabriele Keller
A: Koen Claessen
A: Simon Marlow
A: Dimitrios Vytiniotis

Helmut Seidl
A: Markus Muller-Olm
A: Chen Ding
A: Noah D. Goodman
A: Jeanne Ferrante
A: Martin D. Schwarz

Rastislav Bodik
A: Viktor Kuncak
A: Martin C. Rinard
A: Trishul M. Chilimbi
A: Kapil Vaswani
A: Ali Sinan Koksal

Sumit Gulwani
A: George C. Necula
A: David Monniaux
A: Bill McCloskey
A: Westley Weimer
A: Feng Zhou

Martin Odersky
A: Kunle Olukotun
A: Tiark Rompf
A: Arvind K. Sujeeth
A: Nada Amin
A: Kevin J. Brown

--> Andrew W. Appel
A: Jerome Vouillon
A: Zhong Shao
A: Aquinas Hobor
A: Paul-Andre Mellies
A: Hayo Thielecke

Shaz Qadeer
A: Shuvendu K. Lahiri
A: Sriram Sankaranarayanan
A: Aarti Gupta
A: Brian Hackett
A: Radu Rugina

Joseph Gil
A: Marcelo P. Fiore
A: Yoav Zibin
A: Paul Hudak
A: Jaswinder Pal Singh
A: Zhijing G. Mou

--> Stephanie Weirich
A: Steve Zdancewic
A: Jianzhou Zhao

A: Santosh Nagarakatte
A: Milo M. K. Martin
A: Adam Chlipala

Jeffrey S. Foster
A: Michael Hicks
A: Avik Chaudhuri
A: Iulian Neamtiu
A: Michael Norrish
A: Aseem Rastogi

Xavier Leroy
A: Norman Ramsey
A: Gabriel Dos Reis
A: Tahina Ramananandro
A: Joao Dias
A: Jean-Baptiste Tristan

Mayur Naik
A: Mooly Sagiv
A: Alex Aiken
A: Ghila Castelnuovo
A: Percy Liang
A: Omer Tripp

Dan R. Ghica
A: Shriram Krishnamurthi
A: Michael D. Adams
A: Tony Hoare
A: Rui Vieira
A: Ricardo Rocha

Vertex Count: 1400
Edge Count: 5162
Number of Reps: 180
Avg Vertex Rep Distance:
    1.29990069513
Unrepresented Vertices: 213

popl30reps.pdf

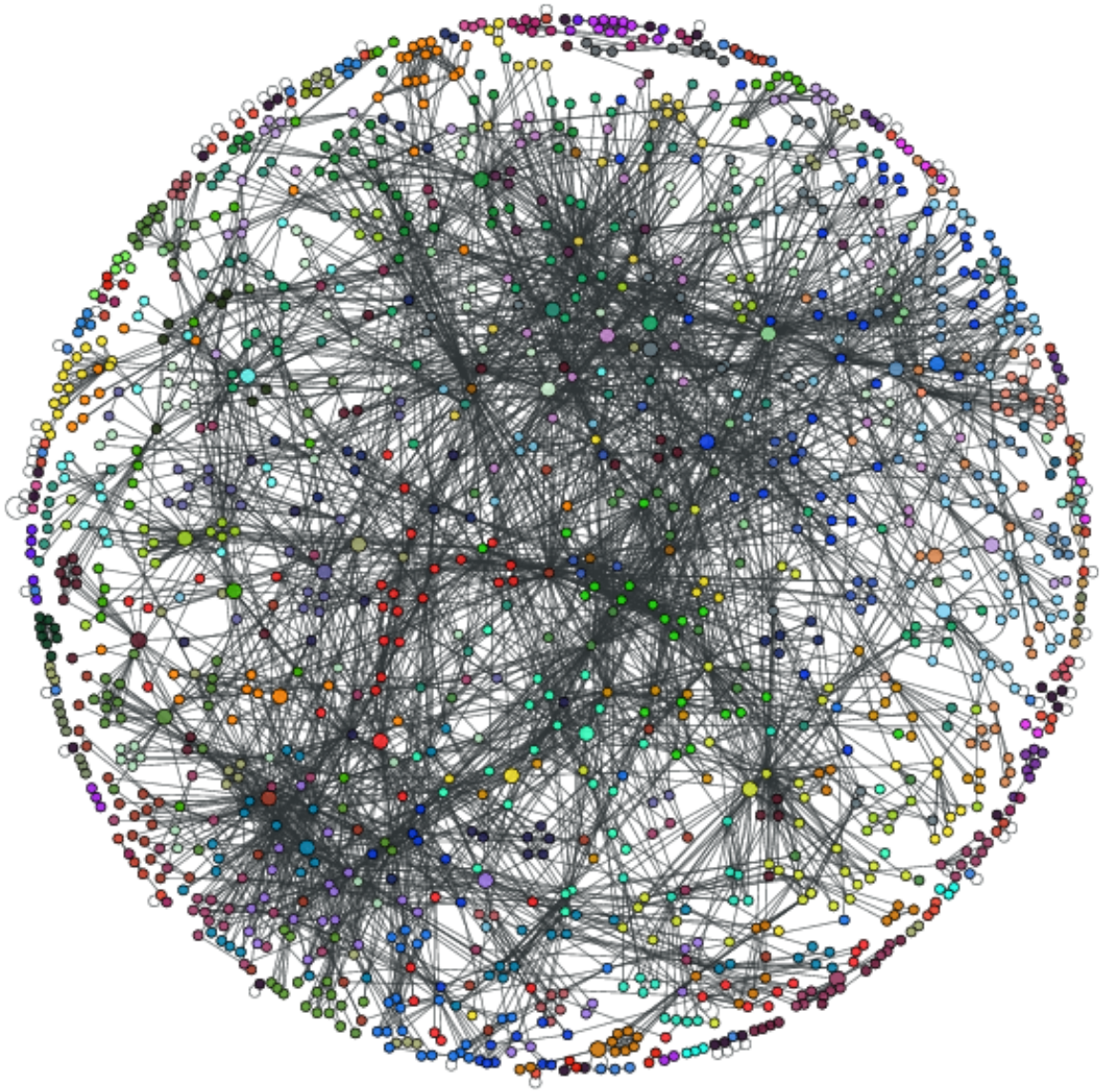


Figure 4: Graph of the Principles of Programming Languages (POPL) conference community. The selected representatives are shown as larger vertices and clusters are distinguished by color.