

Discovering and Visualizing Self-Organizing Communities within Email Archives

Gene Chipman

Term Project

Information Visualization CMSC 838F

May 11, 2002

Background

With the advent of the Internet, use of electronic mail (email) for both personal communication and professional collaboration has continually increased. The use of email for communication will soon far exceed the use of the conventional mail, if it does not already. Email naturally lends itself to being archived because it is stored as a digital medium and as a result, many people have amassed and archived large collections. As with any new communication medium, its existence has changed the dynamics of social communities and led to the creation of new ones. However, the extent of the resulting changes and unique effects has yet to be assessed. The large and relatively complete archives of email will allow the possibility of creating a tool to visualize and explore these dynamics.

We assert that through properly visualizing these archives it will be possible to analyze social relationships, discovering the inherent formation of communities and understand the nature of email collaboration. Possible applications include understanding social or family interactions, studying research related correspondence to better understand the origin of scientific disciplines and innovation, and tracking the communication and flow of information in large organizations. Our goal is to develop a flexible tool that would provide the user an ability to identify and visualize communities of email users and the interactions between these communities.

Previous Work

Work has been done on identifying structures or communities within the World Wide Web. Some of this work uses the network nature of the Web in conjunction with techniques from graph theory. Abstracting the World Wide Web as a

graph, methods based on finding strong connectivity or graph flows can lead to the discovery of organization within the Web. Further work by Flake et al. used a directed graph structure extracted from World Wide Web links and a modification of the max flow graph algorithm to discover communities. ⁴ By using specific websites as seed nodes, the algorithm can find communities related to that site. Creating aggregate structures can also help in the identification of communities. ⁵

Adding more information to the procedure, like the body of the text and scores based on text correlation allow the graph structure to be assigned weights more appropriately. ⁶ This can also lend itself to use of clustering algorithms such as self-organizing maps.

Some research has been done into the benefits of understanding and managing our sets of contacts and communities. Study of social networks has

¹ Flake, G. W., Lawrence, S. and Giles, C. L. Efficient Identification of Web Communities. Proceedings of the ACM (Year?) 150 -160.

² Gibson, D., Kleinberg, J. and Raghavan, P. Inferring Web Communities from Link Topology. Proceedings of the ACM (year?) 225-234.

³ Schwartz, M. F. and Wood, D. C. M. Discovering Shared Interests using Graph Analysis. Communications of the ACM, Vol. 36, No. 8, August 1993, 78 -89.

⁴ Flake, G. W., Lawrence, S., Giles, C. L. and Coetzee, F. M. Self-Organization of the Web and Identification of Communities. IEEE Computer, 35(3) 66 -71.

⁵ Botafogo, R. A. and Shneiderman, B. Identifying Aggregates in Hypertext Structures. Hypertext Proceedings 1991, December 1991, 63-74.

⁶ Kleinberg, J. M. Hubs, Authorities, and Communities. ACM Computing Surveys, Vol. 31, No. 4es, December 1999.

been ongoing for a number of years and researchers are beginning to address the need to manage our contacts space.⁷ Political historians and social researchers have a great interest in the email archives of large and public organizations, for instance, the email archive of the White House. Email archives are becoming an important aspect of our business and personal lives.⁸

Approach

The techniques referenced above may be applicable to email archives, however, some differences exist between the World Wide Web and email archives. First, an email archive is egocentric in nature, that is, it can only show connections in which the email recipient was involved and therefore some connectivity that may exist between others is lost, resulting in a graph representation that is incomplete. This problem persists to some extent unless a complete archive is available for every person, which is unlikely. If archives exist for major contributors it should still be possible to apply these techniques. With modifications, they might work for the archive of a single person.

So the approach to this project is twofold. First, an interface for querying the data is needed. Minimum components include the ability to display the quantity of email messages being passed between individuals as well as being able to filter this data by setting periods of message dates. It is also necessary to display the relation between individuals and communities, closely coupled with the use of algorithms developed for community discovery. Second, techniques for discovering inherent communities within the archive must be developed. Ultimately this will consist of adapting and combining existing graph algorithms already used for World Wide Web analysis. At this point in the project development, only a simple algorithm is used,

primarily for testing the interface portion of the project.

Interactive Visual Interface

The major portion of screen space in the application is used for display of an adjacency matrix. An adjacency matrix was chosen because it should provide desirable aspects; the ability to view overall organization and some clustering effects while retaining the ability to see details of specific email senders/recipients, and fast display updates, necessary for making the tool interactive. The alternative considered was to use a spring model based graph drawing algorithm such as Neato.⁹ This would provide a graphical, more intuitive display of the communities and their connections. However, it suffers in that it would provide less information on community members and refresh times would hamper interactivity.

The adjacency matrix can display data in different forms. One form can display email senders in rows and recipients in columns (see Figure 1). In this mode, cells in the matrix are showing number of email sent from one person (row) to another (column). The intensity of cells are relative in that the greatest intensity of display is given to the single largest entry (most messages from any single person to any single person).

Moving the mouse over any cell will show a label detailing the sender, recipient and number of messages out of the maximum possible. Moving the mouse over a row name will show the total numbers of messages sent by this person (which is the sum of all cells along that whole row). Moving the mouse over any column name has a similar affect, showing the total number of messages received (the sum of all cells along that column).

Cell selection will also result in a horizontal bar highlighting the sender's row and a vertical bar highlighting the recipient's column, much like a crosshair. Default highlight color is yellow. There is also a second set of highlight bars that appear in another color (default of green). A horizontal highlight bar of this color will appear upon the row in which the selected recipient appears (as a sender of emails). A vertical one

⁷Nardi, B. A., Whittaker, S., Isaacs, E., Creech, M., Johnson, J. and Hainsworth, J. Integrating Communication and Information Through Contact Map. *Communications of the ACM*, Vol. 45, No. 4, April 2002, 89 -95.

⁸Adams, J. S. and Beise, C. M. Career Implications of Electronic Mail Use and Misuse: Research in progress. *Proceedings of the ACM* 1997, 246 -248.

⁹AT&T Labs -Research. www.graphviz.org

appear on the column in which the selected sender appears (as a recipient of emails). In effect this second highlight shows message traffic in the reversed direction of the selected cell.

For example, if the cell for email from Ralph to Fred is selected, a pair of yellow lines will cross at that cell. The horizontal yellow line will highlight all of Ralph's sent messages and the vertical yellow line will highlight all of Fred's received messages. There will also be a pair of green lines crossing on the cell where Fred sent email to Ralph, the horizontal green line highlighting all of Fred's sent messages and the vertical green line all of Ralph's received messages. This feature allows quick determination of whether a message relationship is bidirectional, in addition to showing where elsewhere are messages that the set two are involved in.

Below the matrix display is a range slider, a widget that allows the user to specify a minimum and maximum, similar to standard slider widgets except that two values instead of one are controlled. The two endpoints can be controlled separately, or the entire interval can be adjusted as one, changing the minimum and maximum together. This range slider provides a dynamic query method for filtering displayed email based on date by specifying a range of dates for which messages are displayed.¹⁰ Changes of the date filter automatically update the matrix display so that cell intensity corresponds to the number of messages within that date range. This intensity remains relative to the maximum possible over all dates.

A graph data structure underlies the adjacency matrix and maintains the complete set of individuals and message entries. From the menu a community finding algorithm can be selected. It will be applied to the underlying graph data structure to discover the inherent communities. Once determined, a list of communities appears in a list box to the right of the matrix. Communities are named for the individual in the community who sends the most email to others in the community.

When selecting a community from the list, members of that community are moved into the positions in the upper left corner of the matrix, swapping places with the individuals currently in those rows and columns. A blue box highlights this community group (see Figure 2). Their position in this area is in an order corresponding to number of messages sent or received by them, within the community. Rows are sorted on order based on messages sent and columns based on messages received. This allows the user to immediately see the members of the community and how prolific they are within the community. Note that the person whom the community is named after will appear in the first row entry since this person is the most prolific sender in that community.

Because non-members of the community are not moved except as displaced by community members, some of the previous ordering of the matrix is maintained. Remnants of the previously viewed community will still remain, allowing the user to see some of the interdependence of the two communities (current and previously sorted).

When a community is selected, members of this community also appear in a second list box below the community list box. This allows the user to see who is in the community. Selecting a person within this list will highlight the person position within the matrix, showing if and where they exist in both row and column.

Another menu selection will allow the matrix to display individuals in rows and communities in columns (see Figure 3). A pair of checkboxes allow the user to select whether incoming and/or outgoing messages are displayed. The date range slider still operates the same. Selecting a community from the community list will place members at the top of the matrix in order based on the number of messages sent and/or received with the community (as determined by the selection of the checkboxes). It also outlines the rows of individuals belonging to the community. Selecting a person from the members list will highlight the row corresponding to that person.

Data Structures and Algorithms

There are four basic data structures comprising this tool.

¹⁰Shneiderman, B. Dynamic Queries: for visual information seeking. IEEE Software, vol. 11, 6 (Nov 1994).

1. A mapping between addresses and names.
2. A directed graph structure where each vertex represents a name and directed edges are the set of message date/times from one name to the another and a similar graph for the relation between individuals and communities.
3. An adjacency matrix consisting of rows and columns with ability to vary the order.
4. A set of communities.

The mapping between addresses and names help store reconcile the various addresses and names an individual may use for their email. When reading data in, address/name pairs are first read into this hash table. Addresses are used as key entries with name as the resulting value. When a new address/name pair is encountered, the name is looked up in the hash to determine its mapped value. Then the new address is mapped to this value, consolidating the new address/name pair to a common name if it exists. If it does not exist, it is added to the hash as is. This map can be saved as a file and then edited to double check, cleanup and consolidate the results. The edited file can then be loaded for use when reading in the message data from an archive to insure that people will not be displayed in the tool with several different names, which is common in an email archive.

Vertices in the graph data structure represent individuals and the edge from one vertex to another represent the set of messages between those persons. An array of all messages between two persons is saved in edge between their respective vertices, sorted in order based on date/time. Keeping the message information in this format allows for quickly determining the number of messages sent from one person to another within a certain date/time range. The number of messages is the difference between the index of the first entry with the date range's start date/time and the index of the first entry after the date range's stop date/time. Using a binary search this allows the number of messages within a date range to be calculated in $\log(n)$ time. However, the current version used a linear search for expediency (i.e., to get the project finished on time).

The adjacency matrix displays the data stored in the underlying graph data structure. It is very important to pay attention to the times necessary to manipulate this data structure since these

times affect the graphical repaint times and hence the very interactive nature of the tool. We would also like this tool to scale to large data sets. The adjacency matrix has a set of rows and a set of columns that are some subset of the vertices in the graph. Using hash tables to maintain rows and columns, inserting and swapping rows and columns into the matrix can be done in constant time. This will allow sorting and displaying a community to occur in $n \log(n)$ time instead of n^2 time.

When repainting the display, the matrix takes advantage of the underlying graph structure. It takes linear time to paint the row and column labels. However, a value is not painted for every cell in the matrix, only for the non-empty cells based on the set of edges from the underlying graph corresponding to the vertex represented by that column. Therefore, repaint time is based on:

$$(\# \text{columns}) * (\# \text{vertices in graph}) * (\# \text{edges per column}) * (\# \text{messages in edge})$$

With some optimization, ideally, the matrix repaint time will be:

$$(\# \text{columns}) * (\# \text{edges per column}) * (\log \# \text{messages in edges})$$

A Community data structure consists of a hash set of all members, a hash set of the members who sent messages to the community and a hash set of members who received messages from the community. Though it uses potentially three times as much memory as an alternative method that would encode membership information into a single hash table, it has the advantage of being able to obtain any of the three sets in constant time. For the adjacency matrix of a person to community display, an underlying graph is created with vertices for all communities and members, and directed edges in the graph that are the collection of all messages from that person to persons in the community, or from all persons in the community to that person. Rows in the adjacency matrix are persons and columns are communities.

Currently a simple algorithm attempts to form communities somewhat based on the actual data. It is detailed in Appendix A. Though communities formed with this algorithm are somewhat random, persons within them tend to have some relationship.

Future Work

The community algorithm suffers that a person can be assigned to one and only one community, so there is no overlap between community members. The original idea for this project was partly based on methods for finding communities within the World Wide Web. These methods allow overlap between community members and are adaptable for finding email communities. Another method for finding clusters in directed graphs is MCL¹¹. Though intended for finding non-overlapping clusters, this algorithm has intermediate steps that produce overlapped clusters.

One challenge for applying these algorithms is the egocentric nature of an email archive collected by one user. One solution is to combine archives of several persons. But without that, some modifications of the underlying graph data may be necessary to get acceptable performance from clustering algorithms, such as reducing or eliminating links to and from the primary individual of the archive.

It is also going to be necessary to change the tool in order to handle much larger datasets. Presently it works well for a few hundred senders and recipients, but beyond that, the display gets hard to read, especially since the matrix tends to be sparsely populated. One simple improvement would be to display a logarithmic intensity instead of one linearly based on number of messages. Exact parameters for doing so may need to be tinkered with, or better yet, made user configurable. Filtering methods for limiting the number of members in the display may be the best approach to this problem. Dynamic query filters provide a straightforward and interactive method for a user to reduce the displayed information to a manageable set. In addition to the data range filter already provided, filter parameters could include address domain, community memberships, number of messages, even alphabetical.

Another approach might be to implement some form of overview and detail for larger datasets. Being able to zoom into or out of the matrix would help alleviate the overload of information while still allowing the user to control what information is displayed. Implementing sub-communities may aid in understanding the relations within a community while also giving a hierarchical form to the data which will more easily support display techniques, such as treemaps.

It would be useful to be able to manipulate and create communities interactively by having methods for adding and removing community members. This could be done within the matrix by dragging a row or column to within the community border (blue box), or it would also be possible to do this by dragging a person into the list of persons making up the community.

It would be very useful to tie this tool to the actual data ultimately contained within the email archive. Once a user has discovered who is talking to who, it might be desired to easily access those messages. Once a person starts to understand what their own email patterns are, they might want to use or export this information in a form that helps them to manage their contact set and establish methods for organizing and archiving existing and future email.

In order to verify the benefit of these approaches to understanding email communities, it will be important to identify some likely tasks that users of this tool, such as social historians, might do. The next study can be conducted comparing this approach to currently used methods.

Ultimately it is planned to use header data from Dr. Ben Scheiderman's email archive collected over a period of more than 15 years as a source in developing our tool. His involvement in many of the developments of Human Computer Interaction (HCI) should provide what may be the best single source for studying the organization of communities and ongoing collaboration in this field.

Conclusions

We have presented a unique approach to viewing email relationships and communities. Most tools previously have been oriented towards managing individual email messages and message threads

¹¹ van Dongen, S. Graph Clustering by Flow Simulation. PhD Thesis, University of Utrecht, May 2000.

and until now, there has been no approach to visualizing the overall nature of an archive and the social network it may reveal. Some work in finding inherent communities using graph algorithms has been applied to the World Wide Web and this is an intuitive extension of that work to email archives.

In addition to applying methods of self-identification of communities, this approach provides the ability to apply dynamic queries to the underlying archive in order to interactively and intuitively understand those communities. Being able to visualize the effects that sorting or filtering one community has on another may provide a very powerful tool for understanding the relationship between communities as well.

While providing these tools for investigating the greater relationships of communities, it also provides a simple method for seeing the history of email between individuals. This allows researchers to not only determine what communities exist and who is a member but also a simple approach for locating and tracing the interaction between individuals, including the effects of different date ranges in this history. The adjacency matrix format allows a reasonably large number of individual's message histories to be viewed and correlated at once and in a straightforward manner.

Appendix A

Random Community Finding Algorithm

Set P = all persons, set C = empty set of communities

While P has persons

 elect a person at random from P

 Let WITHOUT = sum of all messages to and from persons in P

 Let WITHIN = sum of all messages to and from persons assigned to any community in C

 If (WITHIN >= WITHOUT)

 Remove person from P and assign to the community in C which they have the most messages with

 Else

 Remove person from P. Create new community, assign person to the community and add the community to C.

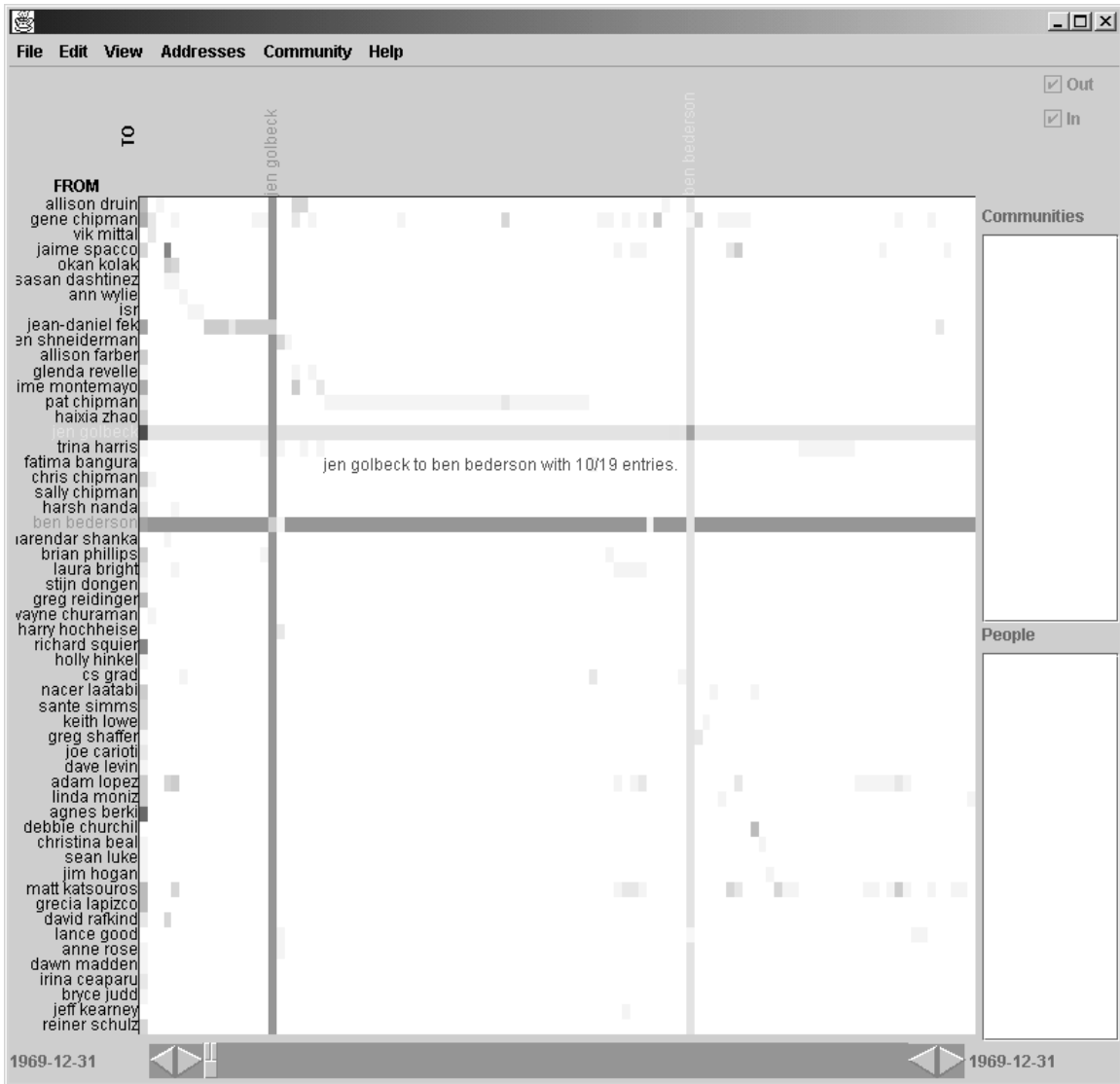


Figure1.

Showingmessageentriesbetweenselectedindividuals

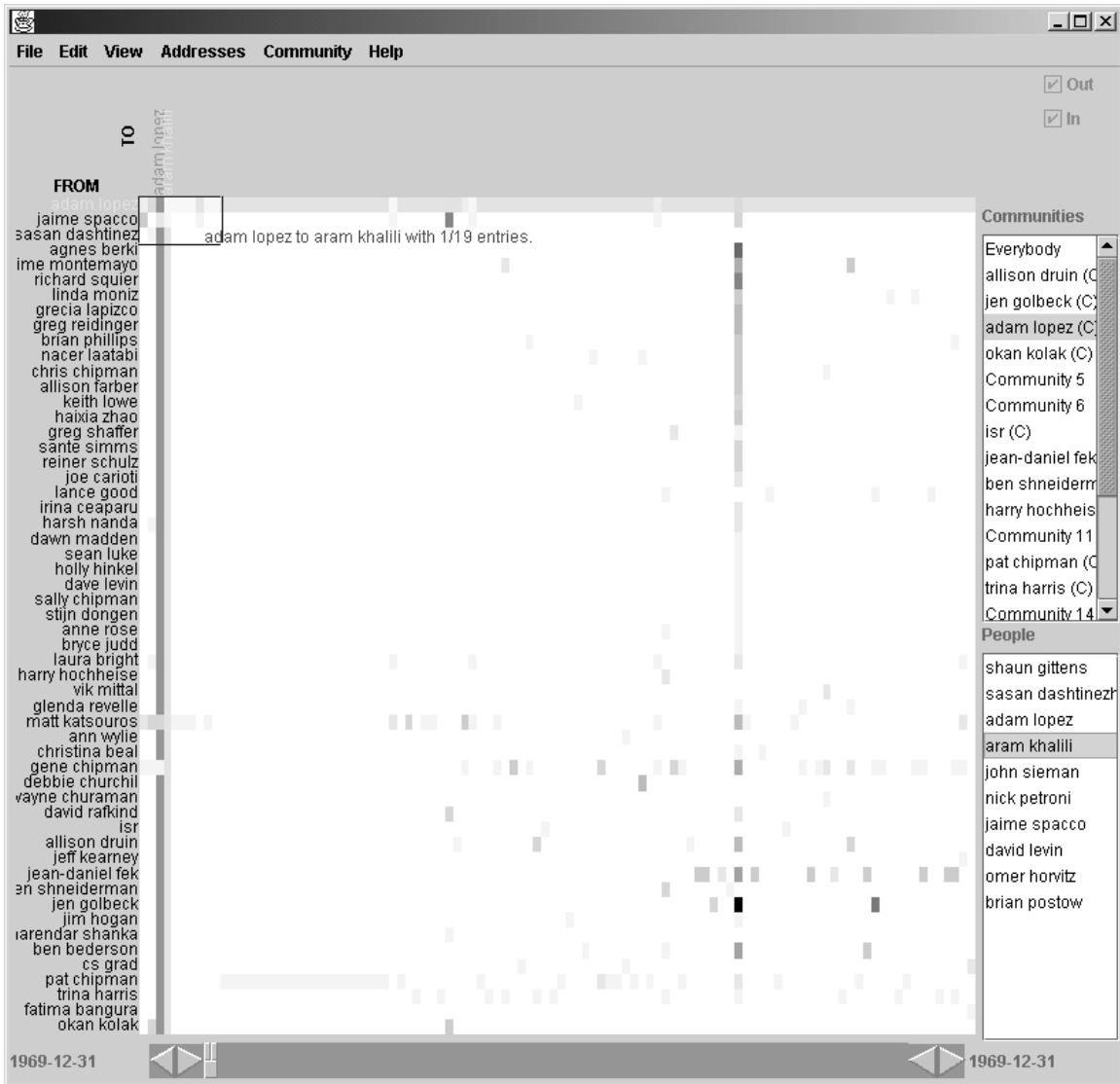


Figure2.

Showingmembersofaselectedcommunity.

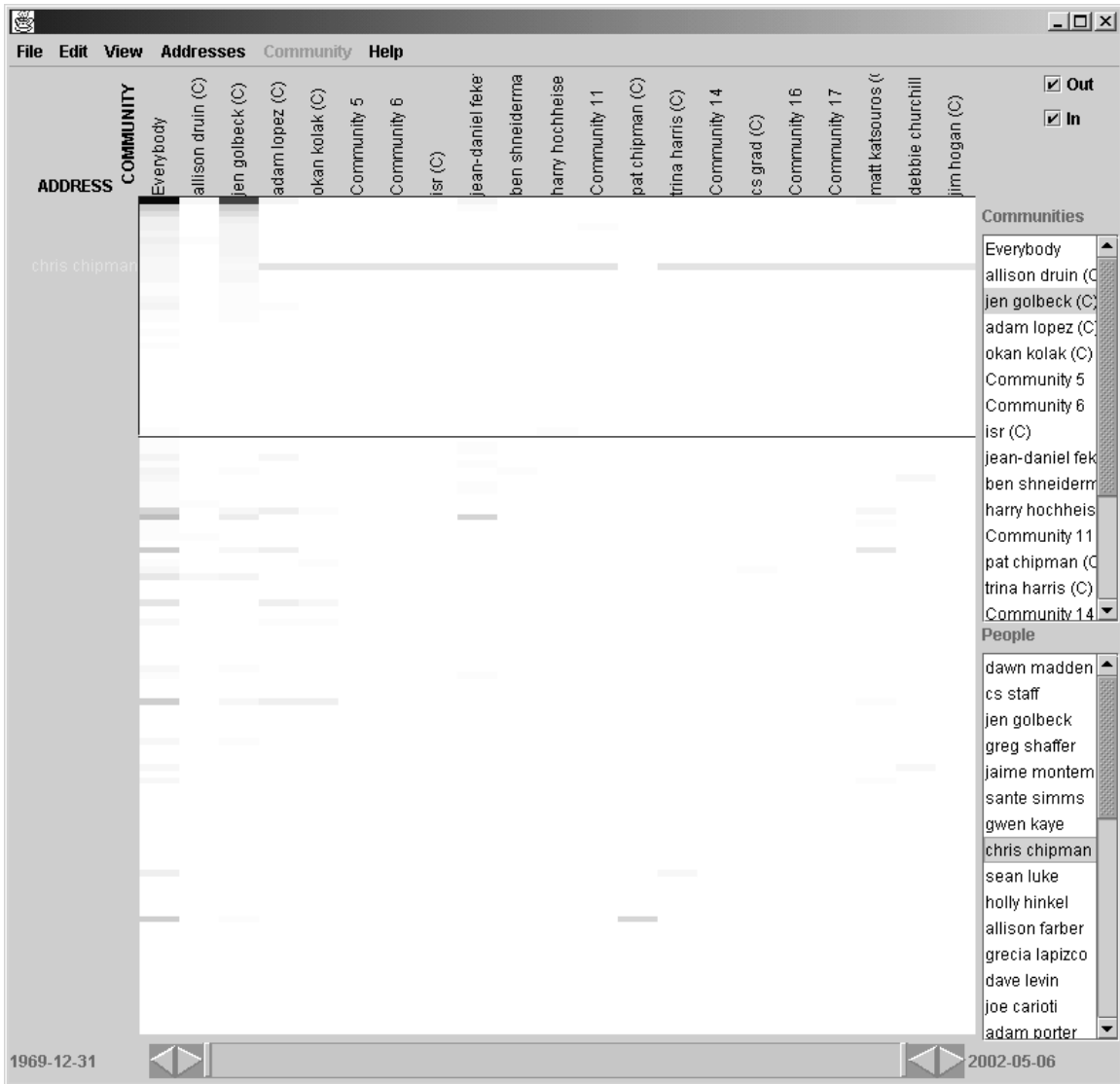


Figure3.

Adjacencymatrixshowingrelationbetweenindividualandcommunity.