CL-SNA: Social Network Analysis with Lisp

[Draft Paper]

Mehmet Gencer
Computer Science Department
Istanbul Bilgi University
Kurtulus Deresi Cad. No:47
34440, Dolapdere, Istanbul, Turkey
mgencer@cs.bilgi.edu.tr

Coskun Gunduz
Computer Science Department
Istanbul Bilgi University
Kurtulus Deresi Cad. No:47
34440, Dolapdere, Istanbul, Turkey
cgunduz@cs.bilgi.edu.tr

Vehbi Sinan Tunalioglu
Computer Science Department
Istanbul Bilgi University
Kurtulus Deresi Cad. No:47
34440, Dolapdere, Istanbul, Turkey
vst@cs.bilgi.edu.tr

Keywords

Common Lisp, Social Network Analysis, Graph Theory, Graph Visualization

1. INTRODUCTION

Social relations have been a research interest in many areas of social sciences. In sociology, for example, individuals up in the power hierarchy in a group can be determined quantitatively by looking at the number of direct relations to others in the group[1]. Similarly, in management and economics, alliance relationships between firms are examined carefully to explain phenomenon such as innovation and market leadership[6]. Similar problems were encountered in Anthropology, Zoology, and many other fields. In most studies regarding social relations, one must look into the content of social relations in addition to their structure, an endeavor which demands ethnographic methods. Nevertheless, analysis of the structure alone has been a powerful tool which not only enables quantitative assessment of social structures, but also prompts elaboration of interpretive process.

Methods used in analysis of social relations was originally coined as *sociometrics* by Psychotherapist Jacob Moreno in 1930s, while he developed systematic ways to represent and assess like-dislike relations between school children[8]. Today, *social network analysis* (SNA) is the generally preferred term for this growing corpus of methods. However, use of these methods, majority of which originated in social sciences, are very common in other fields. A well known example is popularity ranking of websites by looking at how many times they were linked from other sites[7], as in Google search engine. Although various metrics defined in SNA are substantially void of any meaning which is social *per se* but simply definitions over graphs, the term *social network analysis* is widely regarded for the very fact that it gives such meanings to them (For a good review see Scott[10]).

As SNA methods became widespread, many software tools and libraries were developed to compute metrics and visualize social networks. UCINET[11] was among the first to offer a comprehensive set of features, and is a commercial software. A more recent package, Pajek[9], is freeware. There also exist a few libraries for R-Project[4]. However, most of the open source packages which are available seem to be incomplete when one needs a toolkit which can provide a wide range of SNA metrics for different types of graphs (e.g. including directed graphs), means for partitioning and clustering, tools for visualization, and at the same time one that is easily re-programmable for specific needs suitable to exchange data with other software tools.

CL-SNA package aims to bridge this gap by being feature rich and easily extensible by the virtue of being written in Lisp. Following framework sets the design principles of CL-SNA:

- A social graph, or a set of graphs, can be constructed by means of reading in a concise format, or in a programmatic way.
- Graph nodes and edges can be attributed with meta-information, in form of a mapping, to facilitate selective processing of sub-graphs later.
- A variety of SNA metrics including, but not limited to, degree centrality, betweenness, closeness, eigenvalue centrality, are provided. New ones can be easily added using graph search or matrix based methods.
- SNA measures of graph nodes can be exported for statistical processing in other software such as R.
- Graphs can be visualized in graphics interface, exported in suitable image formats, or exported in formats suitable for visualization in other packages such as Pajek.

Although all of the aspects above are important for user convenience, an important part of CL-SNA's competence lies in variety and accessibility of SNA metrics implemented. For the large part we have drawn from comprehensive work of

Wasserman et.al. [12] in the implementation, and referenced others whenever necessary.

CL-SNA package is distributed with General Public License[5] and it comes with a variety of examples and comprehensive documentation for both novice users and more experienced users who wish to extend it for their particular problem.

2. KEY FEATURES

CL-SNA's operation consists of three phases:

1. data input,
2. processing, and
3. visualization and data export.

While trying to satisfy the requirements of a social network analysis methodology, we attempt to propose open standards in each phase. Data import and export is of great importance among these. Another important issue is a well defined API which enables further development and porting of the application suite to other programming languages and environments, such as R, SciLab, Python or Matlab.

International Network for Social Network Analysis[3] provides a compilation of available software for social network analysis[2]. Most of them are special purpose software. Besides covering most methodologies, CL-SNA attempts to fill the gap to provide a well defined input format.

Wasserman et.al.[12] refers to three different notations which are consistent with each other: Graph theoretic, sociometric and algebraic notations. Graph theoretic, among these, can be generalized easily to multirelational networks and this notation is adopted in CL-SNA. On the other hand, this is a well known and easily interpretable language for software developers. A well defined data input format has been defined to represent social network data with an important addition: metadata. Metadata is necessity for describing the data set, required for classification, processing and outputting.

We pay much attention to a particular issue: Software development methodology. While agile methodology has been employed during development, we simultaneously aimed for a robust API specification which enables us to extend both the API and the implementation incrementally, i.e. brand new functionality can be added easily to the software. Thus, the documentation of the API is a serious task among others, so that, at the end, it introduces the possibility of similar SNA implementations in different programming languages and environments.

Finally, CL-SNA offers a rich set of visualization and data oriented output facilities. Generic interfaces allow us, users and future developers to visualize the output both in CL-SNA framework and external viewers including Graphviz, ImageMagick, gnuplot etc., while providing the advantage of exporting results and data with and without manipulating them.

3. IMPLEMENTATION APPROACH

There are various types of networks which may be the subject of SNA methods. In graph theoretic terms, these networks can be: (1) directed or undirected, (2) one or two mode, (3) weighted or not weighted, and (4) signed or not signed. Furthermore, the corresponding graphs can be trees, bipartite graphs etc. Although CL-SNA package does not claim to handle all combinations, it does attempt to generalize analysis implementation and provide convenience methods for converting and dissecting network data for further processing.

Social networks may come in various sizes and different organizational features. Naturally, one should not expect a single recipe leading to enlightenment about all networks one may encounter. In this respect, we attempt to provide a package which has sensible defaults for most analysis procedures, but remain open to recombination of procedures according to social researcher's interpretation of the specific problem in hand. In addition the family of research problems in consideration necessitates descriptive methods through better visualization and capability to combine power of CL-SNA package with other specialized analysis tools.

In order to convene our approach to accommodate these various constraints, the following three sections summarize the implementation approaches for analysis, visualization and data exchange.

3.1 Organization of Network Analysis Procedures

Different levels of analysis leads to a variety of network metrics. There are several types of measures for assessment of properties for a particular node, a group of nodes, or the whole network:

Network Level Measures: A common measure is graph density, which is defined as a ratio of the number of existing links to the number of all possible links. However, most measures are more challenging to compute for large number of nodes and/or links. For example, one would need the reachability matrix, so that we can check partitions in the graph, etc.

Actor(Node) level measures: An actor's location in a social network is defined through its relations with others. Several measures have been employed for quantification of structural and locational properties of actors in different contexts. In addition to simpler measures such as degree centrality, which is basically the number of links to the actor, other measures such as betweenness centrality, closeness centrality, or information centrality have been defined. In betweenness centrality, for example, one must check how often an actor stands on the way linking two actors, which requires substantial time to compute for larger networks.

Group level measures: Discovery and comparison of structural subgroups in a social group has been very interesting for social scientists. Different methods for discovery of subgroups have been proposed, for example, based on reach diameters or nodal degrees. In addition

relations of different subgroups can be assessed similar to the case of actors.

SNA literature reveals various approaches for establishing such measures. The key split in this corpus of approaches is the use of graph search methods and matrix methods. For large networks, matrix methods impose an overwhelming computational demand in obtaining some measures. Imagine, for example, using matrix methods for a social network having number of nodes, n, equal to 20,000 and represented as a matrix, S, which is an $n \times n$ matrix . Even a simple check such as connectivity will require computing S^n, which imposes a computational complexity at the level of $O(n^3 \log(n))$. For these reasons, our approach in CL-SNA is to represent social networks as S-Expressions and only when matrix methods are more convenient switch to using them.

In order to speak of group level measures, one needs means to determine subgroups. Identification of subgroups is perhaps one of the most interesting tasks in social network analysis. For this purpose, the package also provides several methods based on established group cohesion tests in SNA literature such as LS sets or Lambda sets, and efficient generation of candidate subgroups.

One example to define subgroups is LS sets. Let N be the set of all nodes in the graph, which has n nodes, and let us consider a subset, N_s. We will define a subset of nodes taken from N_s as L so that $L \subset N_s$. The set of nodes, N_s is an LS set if any proper subset $L \subset N_s$, has more lines to nodes in $N_s - L$ (other nodes in the subset) than to $N - N_s$ (nodes outside the subset). A search without using any heuristics this search will mean testing relative connectivity of $\sum_{i=2} n \frac{n!}{i!(n-i)!}$ candidate subsets. Thus especially for large networks, tractability of such probing requires a sensible selection of subgroup candidates. There are various matrix or search based approaches in SNA literature regarding such subgroup investigations (see Scott[10] for an excellent review of such methods), having names as k-cores, n-cliques, LS sets, Lambda sets, etc. Providing a rich library of these methods is a key long-term target for CL-SNA package.

3.2 Visualization

One of the research branches in graph theory is the efficient visualization of graphs. There are algorithms developed to achieve this aim. We've leaved to implement and developed such algorithms out of the scope of the application. This job is delegated to the Graphviz suit, which is the main visualization output of the application.

However, even Graphviz cannot always automatically generate well-balanced, compact visualization of social network data. For this, we approached to define subgraphs (in our terminology subnets), and use them as a proxy to compact and balanced visualizations.

Also, we add further information to visualization which are derived from social network metrics. For example, actors with higher degree of centrality are marked with darker colours. Further information is put on the graph visualization as text labels on demand, such as graph density or graph centrality.

```
(SOCIAL-NETWORK
 (METADATA
  (VERSION 1.0)
  (NAME "Florentine Families")
  (DESCRIPTION "Marriage ties of Florentine
                families.")
  (AUTHOR "Isthisa Record")
  (DATE "2002-03-07")
  (DIRECTED No)
  (@ (KEY-1 VAL-1)
     (KEY-2 VAL-2)
     (KEY-3 VAL-3)))
 (DATA
  (NODE Medici (@ (KEY-1 VAL-1)
                  (KEY-2 VAL-2)
                  (KEY-3 VAL-3)
                  (KEY-4 VAL-4)))
  (NODE Pucci)
  (EDGE Medici Pucci (@ (WEIGHT 10)
                        (KEY-1 VAL-1)))))
```

Figure 1: Custom social graph data format

3.3 Data Input and Export

Currently there are two methods for data input. One is reading matrix files, and the other is reading a custom graph data file. The custom graph format uses a markup based on S-Expressions which can be read (parsed) and written easily by Lisp programs. These S-Expressions is also the way the social network is stored and passed around in CL-SNA package. The markup follows XML tagging conventions. It is also possible to add necessary meta attributes for the whole network, or node and edges. Such an example input data file is shown in Figure 1. The $< key, value >$ pairs in various elements are free-form attributes. These attributes can be employed in further merging or dissection of graphs for analysis and CL-SNA package provides essential methods for this purpose.

Unlike matrix notation, it is required in this format for nodes to have distinct names, which in turn is used in describing edges.

In addition to S-Expression format the package can read sociomatrix files, an example of which is shown in the next section. Furthermore, it is possible to export social networks in 'comma separated values' format which is a standard for statistical analysis programs such as SPSS and R or other network analysis packages such as UCINET.

4. EXAMPLE DIRECT USAGE

For demonstration purposes, let us take one of the examples used by Wasserman and Faust[12]: marriage ties between Florentine families of 16th century. There are only 16 entities in this data set, but it is sufficient to demonstrate typical types of analysis. For example, we may look into which families are more central in the set or what groups emerge through marriage relations.

```
"Acciaiuoli" "Albizzi" ... "Tornabuoni"
   0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
   0 0 0 0 0 1 1 0 1 0 0 0 0 0 0
   0 0 0 0 1 0 0 0 1 0 0 0 0 0 0
   0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0
   0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0
   0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1
   0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
   1 1 1 0 0 0 0 0 0 0 0 1 1 0 1
   0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
   0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1
   0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
   0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0
   0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0
```

Figure 2: The matrix representing data for marriage ties between Florentine families

To start the analysis, we first need a way to load the data of the graph corresponding to our social network. In this example, we have a matrix representation, so called 'adjacency matrix', as shown in Figure 2. Note that there is no specification in matrix notation whether the corresponding graph is directed or weighed. The relation in consideration, namely the marriage relation, is not directed. Thus, the matrix is symmetrical. And all weights are 1s.

We will use the low level access to CL-SNA in the following example. Let us read the social network data with CL-SNA and put into a variable for future processing. This data comes from a matrix file so it needs some conversion first by using `matrix2sexp`:

```
* (setf graph
    (matrix2sexp
      (read-matrix-from-file
         "florantine-families.txt"
         :labels t )))
```

`:labels t` statement refers to that there exist labels in the matrix file to identify the nodes in the graph. Now, let us look at the graph-wide property of density:

```
*  (graph-density graph)
1/6
```

Now we can compute degree prestige of individual nodes:

```
* (setf nodenames (get-node-names graph))
* (setf edges (get-edges graph))
* (print-degrees
    (degree-centrality-of-all-nodes
       nodenames edges))
in-degree of Acciaiuoli          : 1
in-degree of Albizzi             : 3
in-degree of Barbadori           : 2
```

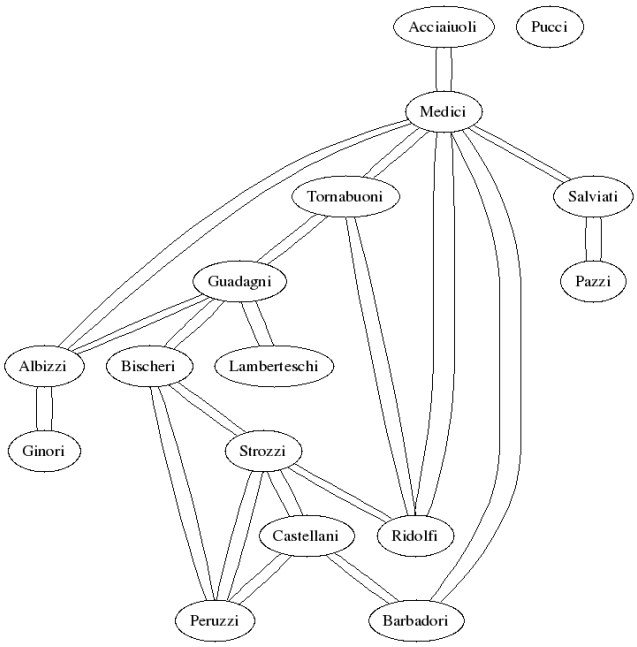

Figure 3: Visualization of marriage ties between Florentine families

```
in-degree of Bischeri            : 3
in-degree of Castellani          : 3
in-degree of Ginori              : 1
in-degree of Guadagni            : 4
in-degree of Lamberteschi        : 1
in-degree of Medici              : 6
in-degree of Pazzi               : 1
in-degree of Peruzzi             : 3
in-degree of Pucci               : 0
in-degree of Ridolfi             : 3
in-degree of Salviati            : 2
in-degree of Strozzi             : 4
in-degree of Tornabuoni          : 3
```

Before we proceed to node and subgroup level properties of this data set, let us have a look at it. To do this we will first need to export it into Graphviz's dot format:

```
* (graph2dotfile "florantine.dot" graph)
NIL
```

The above command will save the graph in a format suitable for display in Graphviz program, into file "./florantine.dot". Which in turn can be visualized with various Graphviz command-line utilities. The result is shown in Figure 3.

This version of visualization is quite rudimentary. As no guidance in grouping the nodes for display is provided to the external tool (Graphviz) it is hard to visually spot distinctive nodes, such as the *Medici* family which has more relations than any other family.

A suggested visualization of groups in our example using the

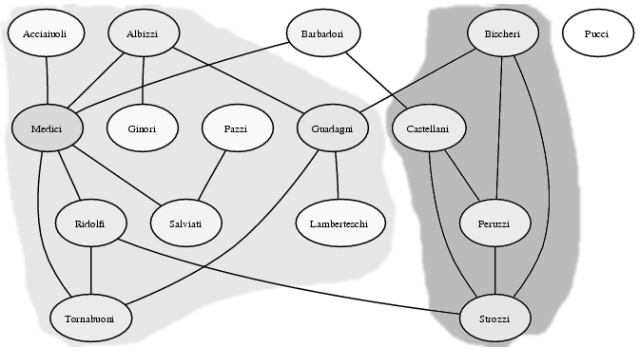

Figure 4: Better visualization emphasizing subgraphs

LS set approach is shown in Figure 4. The groups are better visible in this version, however, further work is necessary to achieve better integration with Graphviz.

5. COMPUTATIONAL ASPECTS

The package includes the implementations of social network metrics on the basis of graph theoretic notation. This notation brings some important keypoints of graph theory to this approach. For many functions, runtime analysis should be covered during implementation and efficiency was a concern as much as functionality.

Adjacency matrices are used for feeding the input to the software, too. If there are n vertices in the graph i.e. nodes or elements in the social network, then there are n^2 numbers in the adjacency matrix by its definition. Thus, even for simpler functions, the runtime of the code goes up to $O(n^2)$, which is bigger than many functional and useful procedures like sorting.

As an example, to calculate the maximum degree of centrality in the graph, the whole matrix should be traversed and degree of centrality for each vertex should be calculated. Then, the maximum of these values is found as the output of the related function. That makes $n^2 + n$ arithmetic operations that corresponds to $O(n^2)$. For increasing the efficiency of, the traversing step can be faster if the graph is undirected. This property makes the adjacency matrix symmetric so that only checking the lower half (or the upper half without loss of generality) may reduce the runtime of traversing process by half. That results in $(n^2/2) + n$ arithmetic operations which is still $O(n^2)$.

Degree of centrality is an easy function to implement. So that it is easy to reduce the number of arithmetic operations by trivial methods. On the other hand, there are some functions which are much more complex in terms of algorithm and runtime analysis.

Subgroup identification is an important problem in social network analysis. In this paper, subgroups are defined as LS sets as explained in part 3.1. When this definition is taken into consideration, a function is needed to produce all subsets of a set of nodes. Also for other social network analysis tasks, such a function will be useful. In this specific subgrouping problem, generating all subsets of a set of nodes, then for each set, counting the edges to that set and to the complement of that set outcomes an $O(n^3)$ runtime. So, some pruning needed by defining some heuristics such that the star node and the adjacent nodes are taken in the same set to start with. Another heuristic is putting the nodes with only one edge to the same set with its adjacent node. Since these heuristics speed up the subgrouping procedure, it is hard to calculate a new runtime result. Because, one can not make a decision named as an average case such that if there are n nodes, k of them are adjacent to star node etc. That's why, $O(n^3)$ runtime declaration is not updated.

6. LARGER EXAMPLE SIMPLER INTERFACE

The examples above can be cumbersome for simple analysis. For this reason, CL-SNA package provides several convenience functions to facilitate easier and faster experimentation with social network graphs. In order to avoid repetition of commands and computation of some essential measures, two convenience functions, namely `savews` and `loadws`, are provided to store and retrieve user's workspace between separate runs.

As another example, we will look at the relations between software packages in Debian GNU/Linux Distribution[1]. There are 20,610 graph nodes representing software packages in this project and 92,534 graph edges representing dependency links. Although the nodes in this graph are not people, since each package is maintained by a developer the graph also represents dependency between developers. The example sessions below shows usage of this toolkit functions for Debian packages data.

```
 * (new-project "dpkgs.sexp")
Graph loaded successfully.
Graph Name : Debian packages
Number of nodes: 20610, Number of edges:92534

 * (summary)
GRAPH SUMMARY:
Graph Name : Debian packages
Number of nodes: 20610, Number of edges:92534
Graph Density: 0.000218
Finding node degrees. This will take
                    about 5.721377 minutes.
Most central nodes with degree>1057:
LIBSTDC++6 : 1864
PYTHON     : 1127
LIBGCC1    : 2392
LIBICE6    : 1065
LIBSM6     : 1057
ZLIB1G     : 2080
PERL       : 1721
LIBC6      : 9387
XLIBS      : 2665
LIBXEXT6   : 1821
 * (savews)
```

When a work session is saved as above, certain properties of

[1]see http://www.debian.org

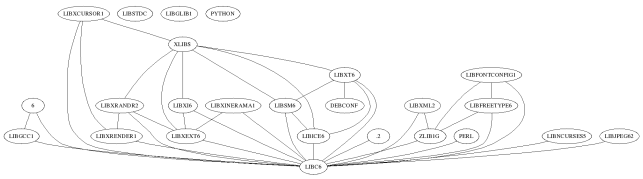

Figure 5: Subset of Debian packages network with nodes having centrality larger than 500

graph nodes will be saved along. So the following commands in the next session will display top ten nodes most faster than the first run and one will not see 'Finding degrees ...' warning:

```
 * (loadws)
 * (summary)
-------------------------
GRAPH SUMMARY:
Graph Name : DEBIAN
Number of nodes: 20610, Number of edges:92534
Graph Density: 0.000218
Most central nodes with degree>1057:
LIBSTDC++6 : 1864
⋮
```

One can manage multiple workspaces by using `loadwsfrom` and `savewsto` functions which accept file path parameter for location to save or load the workspace.

With such large graphs, visualization of the whole population is useless as structural outline would not be visually detectable. One possible mediation is to display a limited subset of nodes, for example those that have higher degree centrality than a certain limit. In the following example, we will switch into a context where we examine a subgraph, export the graph to Graphviz format, then switch back to our older context which is the whole graph:

```
 * (intograph (makesubgraph-degreelimit 500))
 * (dotto "dpkg500.dot")
 * (numcontexts)
Number of contexts stacked:1
 * (exitgraph)
 * (numcontexts)
Number of contexts stacked:0
```

Graph visualization using nodes with centrality higher than 500 is shown in Figure 5. In this version two nodes are visually detectable as highly central.

6.1 Benchmarks

Since determining a node's centrality is a search which takes up time proportional to the number of edges, even this simple measure takes some time to compute.

In our tests it took 7 minutes[2] to finish the summary for Debian packages network in the section above, core of which involves computing degree centrality of all nodes in the graph.

[2]on a Pentium 4 desktop computer with 512MB of Ram

7. DISCUSSION: FUTURE PROSPECTS FOR CL-SNA

Current CL-SNA implementation has no features supporting analysis of network dynamics, i.e. changes in social network. This type of analysis would require comparing two or more snapshots of a social network taken at different moments. In order to do this, one should be able to identify same subgroups in two snapshots, determine which nodes have undergone significant changes in their properties such as centrality, etc. Furthermore, clustering of social network with respect to user defined attributes should be incorporated into the longitudinal analysis. This is a challenging task. However, dynamic network analysis in an important emerging field and would be a valuable addition to any network analysis software.

Other promising prospects for future development of CL-SNA package would be better visualization and statistical analysis capabilities. Graph visualization should be seen as a separate field of work, thus it is best to stick to the approach to simply support exporting graphs for visualization by external tools. However, for complex and crowded social networks two dimensional visualization may deem insufficient. Support for three dimensional and more flexible graph visualization software is worth considering.

Statistical analysis can prove valuable in identifying structural properties of social networks and subsequent driving of network analysis. There are various statistical software packages available. Some such as R-Project are very powerful, have suitable licenses and is command driven which eases integration. Not only extension of CL-SNA with integration of such package to provide statistical capabilities is useful, but also CL-SNA can be integrated to R-Project to enable statistical processing of social graphs with full power of R.

Many of the methods used in SNA are computationally demanding for crowded networks. Parallelization of analysis procedures can greatly improve performance of the package. Sharing and management of multiple workspaces is another engineering issue which is not yet addressed in the implementation of CL-SNA.

8. SUMMARY

CL-SNA is a package for social network analysis written in Lisp. It provides

- a well defined data format for input and output of social network data supporting use of metadata,
- implementations of many metrics developed in the literature,
- a powerful and rich set of data export facilities for visualization and suspending the analysis session, and
- a robust and extensible API for future development.

Being organized towards a social researcher's perspective, CL-SNA provides an easily workable and comprehensive environment for those engaged in quantitative analysis of social network structures. But even more importantly, it aims

to accommodate the very needs of researchers in its future evolution through its design principles, and to be amenable to problem specific modification.

9. REFERENCES

[1] Ronald S. Burt. *Structural Holes: The Social Structure of Competition*. Harvard University Press, Cambridge, 1992.

[2] International Network for Social Network Analysis. Computer programs for social network analysis. http://www.insna.org/INSNA/soft_inf.html (as of 2006-12-14).

[3] International Network for Social Network Analysis. Website. http://www.insna.org (as of 2006-12-14).

[4] The R Foundation for Statistical Computing. R-project. http://www.r-project.org (as of 2006-12-13).

[5] Free Software Foundation. Gnu general public license. http://www.gnu.org/copyleft/gpl.html (as of 2006-12-13).

[6] Ranjay Gulati and Martin Gargiulo. Where do interorganizational network come from? *American Journal of Sociology*, 104:1439–1493, 1999.

[7] Jon. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46:604–632, 1999.

[8] Jacob Levy Moreno. *Who Shall Survive?: Foundations of Sociometry, Group Psychotherapy, and Sociodrama*. Beacon House, 1953.

[9] University of Ljubljana. Pajek - program for large network analysis. http://vlado.fmf.uni-lj.si/pub/networks/pajek/default.htm (as of 2006-12-13).

[10] John Scott. *Social network analysis: a handbook*. Sage Publications, London, 2000.

[11] Analytic Technologies. Ucinet. http://www.analytictech.com/ucinet/ucinet.htm (as of 2006-12-13).

[12] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*. Cambridge University Press, Cambridge, 1994.