Pajek:
A Program for Large Network Analysis

Vladimir Batagelj
Department of Mathematics, University of Ljubljana
Jadranska 19, 1000 Ljubljana
Slovenia

Andrej Mrvar
Faculty of Social Sciences, University of Ljubljana
Kardeljeva pl. 5, 1000 Ljubljana
Slovenia

Social Capital $

Semantic Networks

Pajek Software

Official Journal of the International Network for Social Network Analysis
Pajek: A Program for Large Network Analysis

Vladimir Batagelj and Andrej Mrvar
University of Ljubljana

Large networks, having thousands of vertices and lines, can be found in many different areas, e.g.: genealogies, flow graphs of programs, molecule, computer networks, transportation networks, social networks, intra/inter organisational networks... Many standard network algorithms are very time and space consuming and therefore unsuitable for analysis of such networks. In the article we present some approaches to analysis and visualisation of large networks implemented in program Pajek. Some typical examples are also given.

INTRODUCTION

Pajek (Slovene word for Spider; the “j” is pronounced like an English “y”) is a program, for Windows (32 bit), for analysis of large networks. It is freely available, for noncommercial use, at its homepage:

http://vlado.fmf.uni-lj.si/pub/networks/pajek/

Large networks can be found in many different areas. Usually they are produced automatically, using computers, from different data sources that are already available in computer readable form. For example:

- large genealogies (genealogies having some 10,000 people [21]), e.g., Theoretical Computer Science Genealogy (1,882 persons [30]);

- networks derived from dictionaries and other texts (character mutation/insertion/deletion network on 52,652 English words [24]);

- transportation networks (American airlines with 332 airports [31]);

- large molecule (molecule having thousands of atoms, e.g. DNA [28]);

- communication networks: links among pages or servers on Internet, usage of Usenet [29], phone calls [20];

- flow graphs of programs [15];

- bibliographies, citation networks [9, 7], Erdős graph (network with 5,822 co-authors [18]),...

1Contact the authors at vladimir.batagelj@uni-lj.si and andrej.mrv@uni-lj.si.
Such networks cannot be treated efficiently using standard network analysis tools which are mostly based on matrix representation and are therefore limited to networks of moderate size.

The main goals in the design of Pajek are (see Figure 1):

- to support abstraction by (recursive) factorization of a large network into several smaller networks that can be treated further using more sophisticated methods;
- to provide the user with some powerful visualisation tools;
- to implement a selection of efficient algorithms for analysis of large networks.

One of the approaches to support abstraction is: find clusters (components, neighbourhoods of 'central' vertices, cores,...) in a network, extract and show vertices that belong to the same clusters separately, possibly with the parts of the context (detailed local view), shrink vertices in clusters and show relations among clusters (global view).
Figure 2. Genealogy of US presidents.
EFFICIENT ALGORITHMS FOR LARGE NETWORKS

Time $T(n)$ and space $S(n)$ complexities of an algorithm are estimates of the time and memory space needed to run it on instances of size $n$ (in our case - number of vertices or lines). In most large networks the number of lines $m$ is of the same order as the number of vertices — $O(n)$ or at most $O(n \log n)$ (such networks are considered sparse networks). In the following we assume that we have to analyse large but sparse networks.

According to capabilities of nowadays computers, space complexity for storing sparse networks is not crucial any more. The problem can be solved using appropriate data structures for internal representation of data (doubly linked lists representation of networks is used in Pajek).

Table 1: Time complexities of algorithms (Pentium 64MB RAM 90mHz).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$T(n)$</th>
<th>1.000</th>
<th>10.000</th>
<th>100.000</th>
<th>1.000.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shuffle</td>
<td>$O(n)$</td>
<td>0,00 s</td>
<td>0,015 s</td>
<td>0,17 s</td>
<td>2,22 s</td>
</tr>
<tr>
<td>Quick Sort</td>
<td>$O(n \log n)$</td>
<td>0,00 s</td>
<td>0,00 s</td>
<td>0,40 s</td>
<td>5,14 s</td>
</tr>
<tr>
<td>Heap Sort</td>
<td>$O(n \log n)$</td>
<td>0,00 s</td>
<td>0,06 s</td>
<td>0,98 s</td>
<td>14,35 s</td>
</tr>
<tr>
<td>Insertion Sort</td>
<td>$O(n^2)$</td>
<td>0,07 s</td>
<td>7,50 s</td>
<td>12,50 m</td>
<td>20,83 h</td>
</tr>
<tr>
<td>XY</td>
<td>$O(n^2)$</td>
<td>0,10 s</td>
<td>1,67 m</td>
<td>1,16 d</td>
<td>3,17 y</td>
</tr>
</tbody>
</table>

Key: $s =$ seconds; $m =$ minutes; $h =$ hours; $d =$ days; $y =$ years.
Having much faster computers does not help a lot in the case of high order time complexities. In the theory problems solvable with algorithms of polynomial complexity are considered easy. But, in the case of large $n$, in practice already algorithms of time complexity of order $O(n^{-2})$ can be too slow (for the interactive use), what can be seen in Table 1.

Therefore, most of the algorithms implemented in Pajek have subquadratic time complexities: $O(n)$, $O(n \log n)$, $O(n\sqrt{n})$, or are restricted to small sets of selected vertices.

---

Figure 4. Reordered Snyder and Kick world trade matrix.
DATA STRUCTURES

Currently Pajek uses six data structures to implement the algorithms:

- network - main object (vertices and lines);
- permutation - reordering of vertices;
- vector - values of vertices;
- cluster - subset of vertices (e.g. one class from partition);
- partition - tells for each vertex to which cluster the vertex belongs;
- hierarchy - hierarchically ordered clusters and vertices.

The power of Pajek is based on several transformations which support different transitions among these data structures.

Besides its own input formats, Pajek supports several other formats: UCINET DL [33]; GED [19], genealogies can be read either as Ore-graph or p-graph [21, 16, 17]; and some molecular formats: BS (Ball and Stick), MAC (Mac Molecule) and MOL (MDL MOLfile).

ALGORITHMS

Using the above data structures the basic set of efficient algorithms was implemented [1, 24, 13, 14], for example:

- *partitions*: degree, depth, core, p-cliques, centers;
- *binary operations*: union, intersection, difference;
- *components*: strong, weak, biconnected, symmetric [1];
- *decompositions*: symmetric-acyclic;
- *paths*: shortest path(s), all paths between two vertices [11];
- *flows*: maximum flow between two vertices [14];
- *citation weights*: Paths Count Method and SPLC Method [2];
- *neighbourhood*: k-neighbours;
- *CPM* (Critical Path Method);
- *extracting* subnetwork;
- shrinking clusters in network (generalized blockmodeling) [4];
- reordering: topological ordering, Richards’s numbering, depth/breadth first search;
- reduction: hierarchy, subdivision, degree;
- simplifications and transformations: deleting loops, multiple lines, transforming arcs to edges ...

Figure 5. Clique decomposition of 9-core of Erdős graph.

We can define an often used sequence of elementary operations as a macro and run it as a single command. Using systems of macros we can adapt Pajek to special groups of users (analysis of genealogies, chemical applications,...).

Some special algorithms for solving problems from different areas of network analysis were also included in Pajek: e.g., algorithms for checking whether a program is written structurally [15]; simulation of Petri nets [12]; searching for given fragments/patterns in molecule or genealogies.

Special emphasis was given to automatic generation of network layouts. Several standard algorithms for automatic graph drawing were implemented: spring embedders based on minimi-
sation of the total energy of the system (Kamada-Kawai [10] and Fruchterman-Reingold [6]),
layouts determined by eigenvectors (Lanczos algorithm [3, 5]), drawing in layers (genealogies
and other acyclic structures). These algorithms were modified and extended to enable additional
options: drawing with constraints (optimisation of the selected part of the network, fixing some
vertices to predefined positions, using values of lines as similarities or dissimilarities), drawing
in space. Pajek also provides tools for manual graph editing.

Pajek supports several output graphic formats which can be examined by special 2D and 3D
viewers (Encapsulated PostScript - GSView [25]; VRML - CosmoPlayer [23]; MDLMOL -
Rasmol [28], Chime [22], Kinemages - Mage [26]).

EXAMPLES

In Figure 2 the largest connected p-graph component of the genealogy of the USA presidents
[32] is presented. The shortest kinship path between Franklin Delano Roosevelt and George
Herbert Walker Bush, determined using macro Path, is displayed in Figure 3.

Figure 4 represents a reordered matrix of Snyder and Kick’s world trade relation (iterative core
decomposition with additional analysis of internal structure of cores).

The cliques decomposition (obtained by MODEL2 [27]) of the main core of Erdős graph [18]
is displayed in Figure 5.

The last picture (Figure 6) presents a snapshot of the 3D display by Mage [26] of the Prison
network (from the UCINET [33] dataset).

Figure 6. Gagnon and MacRae Prison Data.
FUTURE PLANS

Pajek is in a constant development. The latest version is available from its homepage. In the near future we are planning to implement the following additional options: different clustering and decomposition procedures, some statistics (triad counts), animation and presentation of the sequence of networks, output formatting, control structures in macros, planarity testing and layout...

REFERENCES


[23] Plug-in Cosmo Player: http://cosmosoftware.com/


