Partitioning Approach to Visualization of Large Graphs

Vladimir Batagelj[1], Andrej Mrvar[2], and Matjaž Zaveršnik[1]

[1] FMF, Department of Mathematics, and IMFM,
Department of Theoretical Computer Science
University of Ljubljana
Jadranska 19, 1000 Ljubljana, Slovenia
vladimir.batagelj@uni-lj.si,
matjaz.zaversnik@fmf.uni-lj.si
[2] Faculty of Social Sciences
University of Ljubljana
Kardeljeva ploščad 5, 1000 Ljubljana, Slovenia
andrej.mrvar@uni-lj.si

Abstract. The structure of large graphs can be revealed by partitioning graphs to smaller parts, which are easier to handle. In the paper we propose the use of core decomposition as an efficient approach for partitioning large graphs. On the selected subgraphs, computationally more intensive, clustering and blockmodeling can be used to analyze their internal structure. The approach is illustrated by an analysis of Snyder & Kick's world trade graph.

1 Approaches to Clustering in Graphs

In the analysis of a large graph we often decompose / reduce it to several smaller manageable parts. This can be done even hierarchically. In the paper we discuss subgraphs induced by classes (clusters) of some partition of the graph vertex set.

There exist several approaches for partitioning graphs. They can be divided in the following groups:

- connectivity based partitions:
  - standard concepts from Graph Theory: components, cliques, k-cores, distance partition from selected subset, ...
  - neighborhoods of "central" vertices
- neighborhood based partitions: cluster is a set of units with *similar* neighborhoods (degree partition, regular partition, colorings, ...)
- other approaches:
  - eigen-vector methods
  - hierarchy of similar graphs

Only approaches with subquadratic time complexities (such as $\mathcal{O}(n)$, $\mathcal{O}(n \log n)$ and $\mathcal{O}(n\sqrt{n})$, n is the number of vertices), are fast enough for large graphs.

J. Kratochvíl (Ed.): GD'99, LNCS 1731, pp. 90–97, 1999.

2 Cores

In this paper we propose the use of cores as an efficient approach to decomposition of large graphs. The notion of core was introduced by Seidman in 1983 [8].

Let $G = (V, L)$ be a graph. V is the set of *vertices* and $L = E \cup A$ is the set of *lines* (*edges* or *arcs*). A subgraph $H = (W, L|W)$ induced by the set W is a *k-core* or a *core of order* k iff $\forall v \in W : \deg_H(v) \geq k$ and H is a maximum subgraph with this property. The core of maximum order is also called the *main* core.

The degree $\deg(v)$ can be: in-degree, out-degree, in-degree + out-degree, min(in-degree, out-degree), ... determining different types of cores. The notion of cores can be generalized to take values of lines into account. Similarly the *p-core* can be defined, where $p \in (0, 1)$, with the requirement that every vertex in p-core has the proportion p of all neighbors in the core.

The cores have the following properties:

- The cores are nested: $i < j \implies H_j \subseteq H_i$ (see Figure 1).
- There exists an efficient algorithm to determine the core hierarchy.
- Cores are not necessarily connected subgraphs.

An algorithm for determining the cores hierarchy is based on the following property:

> If from a given graph $G = (V, L)$ we recursively delete all vertices, and lines incident with them, of degree less than k the remaining graph is the k-core.

INPUT: graph $G = (V, L)$ represented by lists of neighbors
OUTPUT: table *core* with core number for each vertex

```
1.1     compute the degrees of vertices;
1.2     order the set of vertices V in increasing order of their degrees;
2       for each v ∈ V in the order do begin
2.1         core[v] := degree[v];
2.2         for each u ∈ Neighbors(v) do
2.2.1           if degree[u] > degree[v] then begin
2.2.1.1             degree[u] := degree[u] − 1;
2.2.1.2             update the ordering of V
                end
        end;
```

Let us denote $n = |V|$ and $m = |L|$. We shall show that the described algorithm can be implemented to run in time $O(m)$.

To (1.1) compute the degrees of all vertices we need time $O(m)$. Using *bin sort* – collecting all vertices of the same degree in a separate list, combined into the ordered list by the table of their starts, we can (1.2) order the set V in time $O(n)$. The statement (2.1) requires a constant time and therefore contributes $O(n)$ to the algorithm.

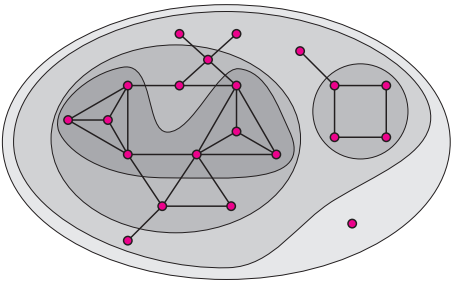

Fig. 1. 0, 1, 2 and 3 core.

The conditional statement (2.2.1) can be implemented to run in constant time by the use of the table determining the position of each vertex data in the order list. Since it is executed for each line at most twice the contribution of (2.2) in all repetitions of (2) is $O(m)$.

Therefore the total time complexity of the algorithm is $O(m)$.

2.1 Core Decomposition

We propose the following approach to decomposition of large graphs based on k-cores: The main core is the most interesting, so we analyze it separately. After that we determine the residual graph, which can be obtained in different ways: by shrinking the main core; by deleting lines in the main core; or by deleting the main core completely (deleting vertices and lines). Again we analyze the main core of the residual graph, and so on ... . Note, as a 'main' core we can consider also the union of some top-level cores selected on the basis of 'core spectrum'.

3 Clusterings in Graphs

Let E be a finite *set of units*. Its nonempty subset $C \subseteq E$ is called a *cluster*. A set of clusters $\mathbf{C} = \{C_i\}$ forms a *clustering*. The clustering $\mathbf{C}$ is a *complete* clustering if it is a partition of the set of units E.

The *clustering problem* $(\Phi, P, \min)$ can be expressed as: Determine the clustering $\mathbf{C}^* \in \Phi$, for which

$$P(\mathbf{C}^*) = \min_{\mathbf{C} \in \Phi} P(\mathbf{C})$$

where Φ is a *set of feasible clusterings* and $P : \Phi \to \mathbb{R}_0^+$ is a *clustering criterion function*. We denote the *set of minimal solutions* by $\mathrm{Min}(\Phi, P)$.

Let $(\mathbb{R}_0^+, \boxplus, 0, \leq)$ be an ordered abelian monoid – usually $\boxplus$ stands for $+$ or max. A *simple* criterion function P has the form:

$$P(\mathbf{C}) = \boxplus_{C \in \mathbf{C}} p(C), \quad p(C) \geq 0 \qquad \text{and} \qquad \forall X \in E : p(\{X\}) = 0$$

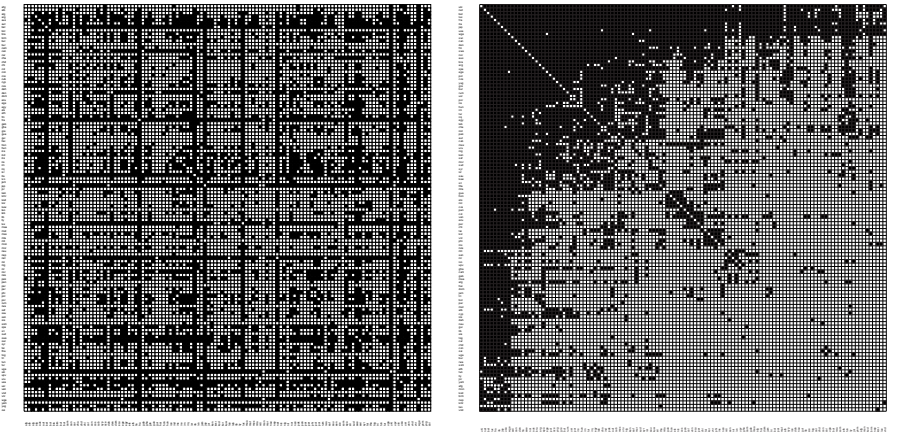

Fig. 2. World trade graph – original adjacency matrix and adjacency matrix reordered according to core decomposition.

The 'cluster error' function $p(C)$ is usually expressed using some dissimilarity measure d over E. For example:

$$p(C) = \max_{u,v \in C} d(u,v) \qquad \text{or} \qquad p(C) = \frac{1}{|C|} \sum_{u,v \in C} d(u,v)$$

An example of dissimilarity between vertices ($\oplus$ denotes the symmetric difference) is:

$$d(u,v) = \frac{|N^+(u) \oplus N^+(v)|}{|N^+(u) \cup N^+(v)|}$$

where $N^+(v) = \{u \in V : (v:u) \in L\} \cup \{v\}$ is the *rooted neighborhood* of vertex $v \in V$.

Assume $E = V$. A pair of clusters (C_1, C_2) determines a *block* – a subgraph

$$B(C_1, C_2) = (C_1 \cup C_2, \{(u,v) \in L : u \in C_1, v \in C_2\})$$

Given a clustering $\mathbf{C}$ we obtain a reduced graph or *blockmodel* of G by shrinking each cluster to a vertex and deciding for each induced block whether it produces a line in the reduced graph, and of what type.

To evaluate the quality of blockmodel a criterion function of *general* form is needed:

$$P(\mathbf{C}; \mathcal{T}) = \sum_{(C_1,C_2) \in \mathbf{C} \times \mathbf{C}} \min_{T \in \mathcal{T}} w(T) \delta(C_1, C_2; T)$$

where $\mathcal{T}$ is a set of feasible types, and δ measures the deviation of blocks, induced by a clustering, from the ideal block structure. For details see [1,3].

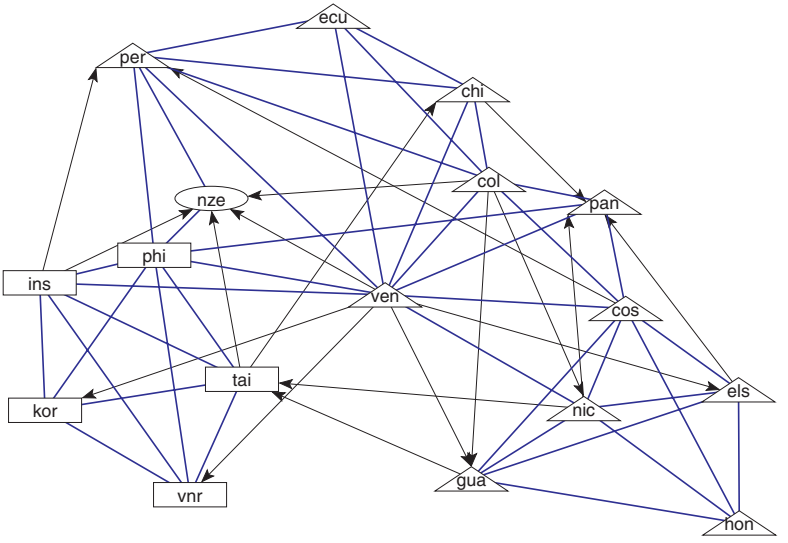

Fig. 3. Secondary core.

4 Example – World Trade Graph

We selected *Snyder & Kick's world trade graph* [9] to illustrate our partitioning approach. World trade among 118 countries is described by 2116 edges and 514 arcs.

Because of limited space for this paper or because of their format some picture were omitted. They are available at:

`http://vlado.fmf.uni-lj.si/pub/networks/doc/part/gd99.htm`

First we tried to obtain a layout of World trade graph using standard spring embedders. The resulting layouts were not satisfactory – the inner structure of the graph cannot be noticed in them.

There are also too many lines. For such graphs a presentation by adjacency matrix is an option. The adjacency matrix of the World trade graph is displayed on the left side of Figure 2. To see some structure in it we have to reorder it.

The first approach to reorder the vertices was by the core decomposition. The main core contains 54 countries (from 118). Its layout was obtained using a 3D spring embedder. The lines were omitted in the picture. By examining countries we found out that such layout could be expected – similar countries were put close to each other.

Because the main core is quite large we divided it further into three clusters using pre-specified blockmodeling [3] with the center-periphery pattern as a goal. The first of the three obtained clusters is a clique on 8 vertices formed by

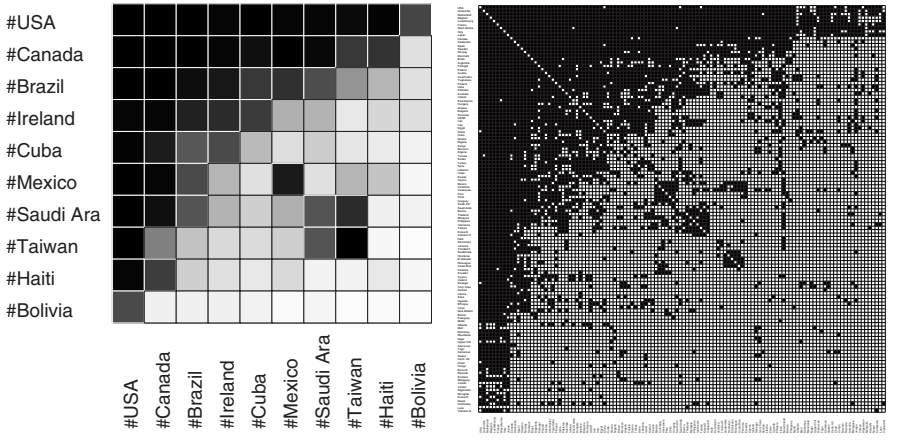

Fig. 4. Blockmodel and reordered adjacency matrix.

Luxembourg, France, Belgium, Denmark, The Netherlands, Great Britain, Italy and Japan. The other two clusters are far away from cliques.

Vertices and lines of the main core were removed from the graph and the secondary core was computed afterwards. It is shown in Figure 3. Three additional steps were needed to complete the decomposition.

The right side of Figure 2 shows the reordered matrix according to the cores – 1's are mostly in the upper left corner. The partition into clusters can be seen in the matrix as well.

Next we applied a blockmodeling (structural equivalence, 10 clusters) and obtained the following clustering $\mathbf{C}_{10}$:

1. USA, UK, Netherlands, Belgium, Luxembourg, France, West Germany, Italy, Japan.
2. Canada, Switzerland, Spain, Sweden, Norway, Denmark.
3. Brazil, Argentina, Portugal, Poland, Austria, Czechoslovakia, Yugoslavia, Finland, India, Pakistan, Australia.
4. Ireland, East Germany, Hungary, Greece, Bulgaria, Rumania, USSR, Iran, Iraq, Egypt, China.
5. Cuba, Ghana, Nigeria, Kenya, Morocco, Algeria, Tunisia, Sudan, Turkey, Syria, Lebanon, Israel, Kuwait, Ceylon.
6. Mexico, Colombia, Venezuela, Peru, Chile, Uruguay, South Africa.
7. Saudi Arabia, Burma, Thailand, Malaysia, Philippines, Indonesia.
8. Taiwan, Korea-S, Vietnam-S.
9. Haiti, Dominican Rep., Jamaica, Trinidad-Tobago, Guatemala, Honduras, El Salvador, Nicaragua, Costa Rica, Panama, Ecuador, Cyprus, Iceland, Senegal, Ivory Coast, Guinea, Liberia, Zaire, Uganda, Ethiopia, Libya, New Zealand.
10. Bolivia, Paraguay, Malta, Albania, Mali, Dahomey, Mauritania, Niger, Upper Volta, Sierra Leone, Togo, Cameroun, Gabon, Centr. Afr. Rep., Chad, Congo, Burundi, Rwanda, Somalia, Malagasy, Jordan, Yemen, Afghanistan, Mongolia, Korea-N, Nepal, Cambodia, Laos, Vietnam-N.

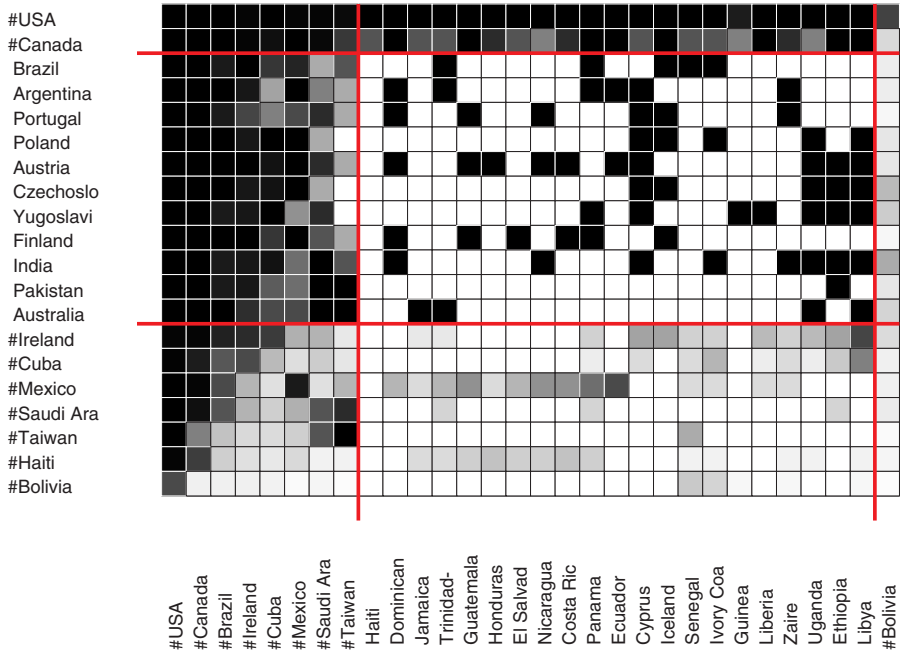

Fig. 5. Contextual matrix (C_3, C_9).

with $P(\mathbf{C}_{10}) = 1452$ – the CONCOR clustering reported in [9] has $P = 2673$. The blockmodel induced by $\mathbf{C}_{10}$ is displayed on the left side of Figure 4, and the corresponding adjacency matrix on its right side.

Adjacency matrix presentation can be used for some hundreds of vertices. For larger graphs we can save the idea by introducing a *contextual matrix presentation* of selected block – adjacency matrix for a selected block surrounded by a blockmodel (see Figure 5).

Another, indirect [2], approach to analyze the graph is to introduce a dissimilarity d into set V, or its subset, and apply some of multivariate techniques to it. We applied Ward's clustering method to the dissimilarity d from Section 3. The results are comparable to that obtained by core decomposition and blockmodeling.

5 Conclusion

The cores, because they can be efficiently determined, are one among few concepts that provide us with meaningful decompositions of large graphs. We expect that different approaches to the analysis of large graphs can be built on this basis. (Most) current clustering and blockmodeling methods can be applied only

to (sub)graphs of moderate size – the development of subquadratic algorithms for these problems is a challenge for a near future.

All the computations in this paper were done with programs `Pajek` (Slovene word for Spider) and `Model2` for Windows (32 bit). They are freely available, for noncommercial use, at their homepage [10].

References

1. BATAGELJ, V. (1997): Notes on Blockmodeling. *Social Networks*, **19**, 143-155.
2. BATAGELJ, V., FERLIGOJ, A., and DOREIAN, P. (1992). Direct and Indirect Methods for Structural Equivalence. *Social Networks* **14**, 63-90.
3. BATAGELJ, V., FERLIGOJ, A., and DOREIAN, P. (1998): Fitting Pre-Specified Blockmodels, in *Data Science, Classification, and Related Methods*, Eds., C. Hayashi, N. Ohsumi, K. Yajima, Y. Tanaka, H. H. Bock, and Y. Baba, Springer-Verlag, Tokyo, p.p. 199-206.
4. BATAGELJ, V., FERLIGOJ, A. (1998): Constrained Clustering Problems, in *Advances in Data Science and Classification* Eds., A. Rizzi, M. Vichi, H.-H. Bock, Proceedings of IFCS'98, Rome, 21-24. July 1998. Springer, Berlin, p. 137-144.
5. BATAGELJ, V., MRVAR, A. (1998): Pajek – A Program for Large Network Analysis. *Connections* **21 (2)**, 47-57.
6. FRUCHTERMAN, T. M. J., REINGOLD, E. M. (1991): Graph Drawing by Force-Directed Placement. *Software, Practice and Experience* **21**, 1129-1164.
7. KAMADA, T., KAWAI, S. (1989): An Algorithm for Drawing General Undirected graphs. *Inf. Proc. Letters* **31**, 7-15.
8. SEIDMAN, S. B. (1983): Network structure and minimum degree. *Social Networks* **5**, 269-287.
9. SNYDER, D., KICK, E. (1979): Structural position in the world system and economic growth 1955-70: A multiple network analysis of transnational interactions. *American Journal of Sociology* **84**, 1096-1126.
10. Program Pajek
    http://vlado.fmf.uni-lj.si/pub/networks/pajek/