Hierarchical clustering with relational constraints of large data sets

Vladimir Batagelj
University of Ljubljana, FMF, Department of Mathematics
vladimir.batagelj@fmf.uni-lj.si

Anuška Ferligoj, Andrej Mrvar
University of Ljubljana, Faculty of Social Sciences
{anuska.ferligoj, andrej.mrvar}@fdv.uni-lj.si

October 27, 2009

Abstract

In the paper an adaptation of the hierarchical clustering with relational constraints approach proposed by Ferligoj and Batagelj (1982, 1983) to large data sets is presented. To obtain an efficient algorithm for large networks, we: (1) compute the dissimilarities between units (vertices of network) only for endpoints of existing links of constraining relation; (2) define the dissimilarities between clusters based only on the dissimilarities of the corresponding links and derive the update relations. We also show that for selected dissimilarities between clusters, the Bruynooghe reducibility property holds. This allows us to speed-up the hierarchical clustering procedure by using the RNN (reciprocal nearest neighbors) approach.

The developed algorithms are implemented in Pajek – a program for analysis and visualization of large networks. The proposed approach is illustrated by clustering of US counties.

Keywords: agglomerative clustering, algorithm, relational constraint, large data set, network, nearest neighbors, reducibility

1 Introduction

In the paper an adaptation of the hierarchical clustering with relational constraints approach proposed by Ferligoj and Batagelj (1982, 1983) [9, 10] to large data sets is presented. In the original agglomerative method for relational constraints [9, 10] a complete dissimilarity matrix is needed. To obtain fast algorithms for large sparse relations we propose to consider only the dissimilarities between linked units.

We shall deal with the clustering problem \((\Phi, P)\):
Determine the clustering $C^* \in \Phi$ for which

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

where $U$ is a finite set of units; $C$ is a cluster, $\emptyset \subset C \subseteq U$; $C = \{C_i\}$ is a clustering; $\Phi$ is a set of feasible clusterings; and $P : \Phi \rightarrow \mathbb{R}_0^+$ is a criterion function.

Since the set of units $U$ is finite, the set of feasible clusterings is also finite. Therefore the set $\min(\Phi, P)$ of all solutions of the problem (optimal clusterings) is not empty. (In theory) the set $\min(\Phi, P)$ can be determined by the complete search. We denote the value of criterion function for an optimal clustering by $\min(\Phi, P)$.

Generally the clusters of clustering $C = \{C_1, C_2, \ldots, C_k\}$ need not to be pairwise disjoint; yet, the clustering theory and practice mainly deal with clusterings which are the partitions of $U$

$$\bigcup_{i=1}^{k} C_i = U \quad \text{and} \quad i \neq j \Rightarrow C_i \cap C_j = \emptyset$$

Each partition determines an equivalence relation in $U$, and vice versa. We denote the set of all partitions of $U$ into $k$ clusters by $\Pi_k(U)$; and by $\Pi(U)$ the set of all partitions of $U$.

Joining the individual units into a cluster $C$ we make a certain "error", we create certain "tension" among them – we denote this quantity by $p(C)$. The cluster-error $p(C)$ has usually the properties:

$$p(C) \geq 0 \quad \text{and} \quad \forall X \in U : p(\{X\}) = 0$$

In the continuation we shall assume that these properties of $p(C)$ hold.

The criterion function $P(C)$ combines these "partial/local errors" into a "global error". Usually it takes the form:

**S.** \[ P(C) = \sum_{C \in C} p(C) \quad \text{or} \quad \text{M.} \quad P(C) = \max_{C \in C} p(C) \]

For simple criterion functions usually $\min(\Pi_{k+1}(U), P) \leq \min(\Pi_k(U), P)$. In this case the clustering of singletons $C^* = \{\{X\} : X \in U\}$ (if it is feasible) is an optimal solution. To get an interesting clustering problem we fix the number of clusters $k$ and consider the problem $(\Phi_k, P)$ where $\Phi_k \subseteq \Pi_k(U)$.

To express the cluster-error $p(C)$ we define on the space of units $U$, $U \subset U$ a dissimilarity $d : U \times U \rightarrow \mathbb{R}_0^+$ for which we require the properties D1 and D2:

**D1.** $\forall X \in U : d(X, X) = 0$

**D2.** symmetric: $\forall X, Y \in U : d(X, Y) = d(Y, X)$

Now we can define several cluster-error functions:

**S.** \[ p(C) = \sum_{X,Y \in C, X < Y} w(X) \cdot w(Y) \cdot d(X, Y) \]

**S.** \[ p(C) = \frac{1}{w(C)} \sum_{X,Y \in C, X < Y} w(X) \cdot w(Y) \cdot d(X, Y) \]

where $w : U \rightarrow \mathbb{R}^+$ is a weight of units, which is extended to clusters by $w(\{X\}) = w(X)$, $X \in U$ and $w(C_1 \cup C_2) = w(C_1) + w(C_2)$ for $C_1 \cap C_2 = \emptyset$. Often $w(X) = 1$ holds for each $X \in U$. Then $w(C) = \text{card}(C)$. Additional examples of cluster-error functions are:
2 Conditions for hierarchical methods

In this section we present the conditions on the structure of the set of feasible clusterings that allow us to use the hierarchical clustering methods for solving the clustering problem.

The set of feasible clusterings $\Phi$ determines the feasibility predicate $\Phi(C) \equiv C \in \Phi$ defined on $\mathcal{P}(\mathcal{P}(U) \setminus \{\emptyset\})$; and conversely $\Phi \equiv \{C \in \mathcal{P}(\mathcal{P}(U) \setminus \{\emptyset\}) : \Phi(C)\}$. $\mathcal{P}(A)$ denotes the power set of set $A$.

In the set $\Phi$ the relation of clustering inclusion $\sqsubseteq$ can be introduced by

$$C_1 \sqsubseteq C_2 \equiv \forall C_1 \in C_1, C_2 \in C_2 : C_1 \cap C_2 \in \{\emptyset, C_1\}$$

we say also that the clustering $C_1$ is a refinement of the clustering $C_2$.

It is well known that the set of all partitions $(\Pi(U), \sqsubseteq)$ is a partially ordered set; even more, a semimodular lattice. Because each subset of partially ordered set is also partially ordered, we have: Let $\Phi \subseteq \Pi(U)$ then $(\Phi, \sqsubseteq)$ is a partially ordered set.

The clustering inclusion determines two related relations (on $\Phi$):

the strict inclusion $\sqsubset$: $C_1 \sqsubset C_2 \equiv C_1 \sqsubseteq C_2 \land C_1 \neq C_2$

and the predecessor $\sqsubset$: $C_1 \sqsubset C_2 \equiv C_1 \sqsubseteq C_2 \land \exists C \in \Phi : (C_1 \sqsubseteq C \land C \sqsubset C_2)$

2.1 Conditions on the structure of the set of feasible clusterings

We shall assume that the set of feasible clusterings $\Phi \subseteq \Pi(U)$ satisfies the following conditions:

**F1.** $O \equiv \{\{X\} : X \in U\} \in \Phi$

**F2.** The feasibility predicate $\Phi$ is local – it has the form $\Phi(C) = \bigwedge_{C \in C} \varphi(C)$ where $\varphi(C)$ is a predicate defined on clusters from $\mathcal{P}(U) \setminus \{\emptyset\}$.

The intuitive meaning of $\varphi(C)$ is: $\varphi(C) \equiv$ the cluster $C$ is ’good’. Therefore the locality condition can be read: a ’good’ clustering $C \in \Phi$ consists of ’good’ clusters.

**F3.** The predicate $\Phi$ has the property of binary heredity with respect to the fusibility predicate $\psi(C_1, C_2)$, i.e.,

$$C_1 \cap C_2 = \emptyset \land \varphi(C_1) \land \varphi(C_2) \land \psi(C_1, C_2) \Rightarrow \varphi(C_1 \cup C_2)$$

This condition means: in a ’good’ clustering, a fusion of two ’ fusible’ clusters produces a ’good’ clustering.

**F4.** The fusibility predicate $\psi$ is compatible with clustering inclusion $\sqsubseteq$, i.e.,

$$\forall C_1, C_2 \in \Phi : (C_1 \sqsubseteq C_2 \land C_1 \setminus C_2 = \{C_1, C_2\} \Rightarrow \psi(C_1, C_2) \lor \psi(C_2, C_1))$$

**F5.** The interpolation property holds in $\Phi$, i.e., $\forall C_1, C_2 \in \Phi :$

$$(C_1 \sqsubseteq C_2 \land \text{card}(C_1) > \text{card}(C_2) + 1 \Rightarrow \exists C \in \Phi : (C_1 \sqsubseteq C \land C \sqsubset C_2))$$
These conditions provide a framework in which the agglomerative hierarchical methods can be applied also for solving constrained clustering problems with $\Phi_k(U) \subseteq \Pi_k(U)$.

In the ordinary clustering problem both predicates $\varphi(C)$ and $\psi(C_p, C_q)$ are always true – all conditions F1-F5 are satisfied.

3 Clustering with relational constraint

Suppose that the units are described by attribute data $a: U \to [U]$ and related by a binary relation $R \subseteq U \times U$ that determine the relational data $(U, R, a)$.

We want to cluster the units according to the similarity of their descriptions, but also considering the relation $R$ – it imposes constraints on the set of feasible clusterings, usually in the following form:

$$\Phi(R) = \{ C \in \mathcal{P}(U) : \text{each cluster } C \in \mathcal{C} \text{ induces a subgraph } (C, R \cap C \times C) \text{ in the graph } (U, R) \text{ of the required type of connectedness} \}$$

For example in the regionalization problem we have to group given territorial units into regions (larger territorial units) such that units inside the region will be as similar as possible and form a contiguous part of the territory.

3.1 Some types of relational constraints

We can define different types of sets of feasible clusterings for the same relation $R$. For the definitions of the notions from graph theory see Wilson [12]. Some examples of types of relational constraint $\Phi^i(R)$, see Figure 1, are

<table>
<thead>
<tr>
<th>type of clusterings $\Phi^i(R)$</th>
<th>type of connectedness</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Phi^1(R)$</td>
<td>weakly connected units</td>
</tr>
<tr>
<td>$\Phi^2(R)$</td>
<td>weakly connected units that contain at most one center</td>
</tr>
<tr>
<td>$\Phi^3(R)$</td>
<td>strongly connected units</td>
</tr>
<tr>
<td>$\Phi^4(R)$</td>
<td>clique</td>
</tr>
<tr>
<td>$\Phi^5(R)$</td>
<td>a trail exists containing all the units of the cluster</td>
</tr>
</tbody>
</table>

A trail is a walk in a graph in which all arcs are distinct. A set of units $L \subseteq C$ is a center of cluster $C$ in the clustering of type $\Phi^2(R)$ iff the subgraph induced by $L$ is strongly connected and $R(L) \cap (C \setminus L) = \emptyset$ – there is no arc leading from the center to the rest of the cluster.

3.2 Properties of relational constraints

The sets of feasible clusterings $\Phi^i(R)$ are linked as follows:

$\Phi^4(R) \subseteq \Phi^3(R) \subseteq \Phi^2(R) \subseteq \Phi^1(R)$ and $\Phi^4(R) \subseteq \Phi^5(R) \subseteq \Phi^2(R)$.

If the relation $R$ is symmetric, then $\Phi^3(R) = \Phi^1(R)$.

If the relation $R$ is an equivalence relation, then $\Phi^4(R) = \Phi^1(R)$.

Here are also the corresponding fusibility predicates:

$\psi^1(C_1, C_2) \equiv \exists X \in C_1 \exists Y \in C_2 : (XRY \lor YRX)$
A unit $X$ is initial/terminal in cluster $C$ iff a trail on $C$ exists containing all units of $C$ and starting/terminating in $X$. $I$ is the set of all initial units in $C$, and $T$ is the set of all terminal units in $C$.

It is easy to verify that for all of them the properties F1-F5 hold; except for the predicate $\psi^3$ for which the property F5 fails.

### 3.3 Agglomerative method for relational constraints

We can use both hierarchical and local optimization methods for solving some types of problems with relational constraint (Ferligoj, Batagelj 1983). In this paper we shall elaborate on the agglomerative method:

1. $k := n$; $C(k) := \{\{X\} : X \in U\}$; $h_D(\{X\}) = 0$, $X \in U$;
2. while $\exists C_i, C_j \in C(k) : (i \neq j \wedge \psi(C_i, C_j))$ repeat
   2.1. $(C_p, C_q) := \text{argmin}\{D(C_i, C_j) : i \neq j \wedge \psi(C_i, C_j)\}$;
   2.2. $C := C_p \cup C_q$; $k := k - 1$; $h_D(C) = D(C_p, C_q)$;
   2.3. $C(k) := C(k + 1) \setminus \{C_p, C_q\} \cup \{C\}$;
   2.4. determine $D(C, C_s)$ for all $C_s \in C(k)$
   2.4. adjust the relation $R$ as required by the clustering type
3. $m := k$

The rules for adjusting the relation after joining of clusters are presented in Figure 2.

The condition $\psi(C_i, C_j)$ is equivalent to $C_iRC_j$ for tolerant, leader and strict method; and to $C_iRC_j \wedge C_jRC_i$ for two-way method.

![Figure 1: Some graphs of different types](image-url)
4 Dissimilarities between clusters

In the original agglomerative method for relational constraints [9, 10] a complete dissimilarity matrix is needed. To obtain fast algorithms for large sparse relations we propose to consider only the dissimilarities between linked units.

Let \((U, R)\), \(R \subseteq U \times U\) be a graph, and \(S\) and \(T\) clusters such that \(\emptyset \subset S, T \subset U\) and \(S \cap T = \emptyset\). We call a block of relation \(R\) for \(S\) and \(T\) its part \(R(S, T) = R \cap S \times T\).

The symmetric closure of relation \(R\) we denote with \(\hat{R} = R \cup R^{-1}\). It holds \(\hat{R}(S, T) = \hat{R}(T, S)\).

For all dissimilarities \(D(S, T)\) between clusters \(S\) and \(T\) we set:

\[
D(\{s\}, \{t\}) = \begin{cases} 
  d(s, t) & \text{if } s \hat{R} t \\
  \infty & \text{otherwise}
\end{cases}
\]

where \(d\) is a selected dissimilarity between units.

It turns out that for three of the frequently used dissimilarities between clusters – minimum, maximum and average – the standard update formulas hold also in the case when we consider only dissimilarities between linked units. In the following \(T_1 \cap T_2 = \emptyset\).

4.1 Minimum

The minimum dissimilarity between clusters \(S\) and \(T\) is defined as

\[
D_{\min}(S, T) = \min_{(s, t) \in R(S, T)} d(s, t)
\]
If $\hat{R}(S, T) = \emptyset$ then $D_{\text{min}}(S, T) = \infty$.

It holds

$$D_{\text{min}}(S, T_1 \cup T_2) = \min_{(s, t) \in \hat{R}(S, T_1 \cup T_2)} d(s, t) = \min(\min_{(s, t) \in \hat{R}(S, T_1)} d(s, t), \min_{(s, t) \in \hat{R}(S, T_2)} d(s, t)) = \min(D_{\text{min}}(S, T_1), D_{\text{min}}(S, T_2))$$

4.2 Maximum

The maximum dissimilarity between clusters $S$ and $T$ is defined as

$$D_{\text{max}}(S, T) = \max_{(s, t) \in \hat{R}(S, T)} d(s, t)$$

If $\hat{R}(S, T) = \emptyset$ then $D_{\text{max}}(S, T) = 0$.

It holds

$$D_{\text{max}}(S, T_1 \cup T_2) = \max_{(s, t) \in \hat{R}(S, T_1 \cup T_2)} d(s, t) = \max(\max_{(s, t) \in \hat{R}(S, T_1)} d(s, t), \max_{(s, t) \in \hat{R}(S, T_2)} d(s, t)) = \max(D_{\text{max}}(S, T_1), D_{\text{max}}(S, T_2))$$

4.3 Average

Let $w : U \rightarrow \mathbb{R}$ be a weight on units. For example $w(v) = 1$, for all $v \in U$. Then we define the average dissimilarity between clusters $S$ and $T$ as:

$$D_a(S, T) = \frac{1}{w(\hat{R}(S, T))} \sum_{(s, t) \in \hat{R}(S, T)} d(s, t)$$

Because for $T_1 \cap T_2 = \emptyset$ hold also $\hat{R}(S, T_1) \cap \hat{R}(S, T_2) = \emptyset$ and $\hat{R}(S, T_1 \cup T_2) = \hat{R}(S, T_1) \cup \hat{R}(S, T_2)$, we get

$$w(\hat{R}(S, T_1 \cup T_2)) = w(\hat{R}(S, T_1)) + w(\hat{R}(S, T_2))$$

and

$$w(\hat{R}(S, T_1 \cup T_2)) D_a(S, T_1 \cup T_2) = \sum_{(s, t) \in \hat{R}(S, T_1 \cup T_2)} d(s, t) = \sum_{(s, t) \in \hat{R}(S, T_1)} d(s, t) + \sum_{(s, t) \in \hat{R}(S, T_2)} d(s, t) = w(\hat{R}(S, T_1)) \cdot D_a(S, T_1) + w(\hat{R}(S, T_2)) \cdot D_a(S, T_2)$$

Therefore, for $\hat{R}(S, T_1) \neq \emptyset$ and $\hat{R}(S, T_2) \neq \emptyset$, it holds

$$D_a(S, T_1 \cup T_2) = \frac{w(\hat{R}(S, T_1))}{w(\hat{R}(S, T_1 \cup T_2))} D_a(S, T_1) + \frac{w(\hat{R}(S, T_2))}{w(\hat{R}(S, T_1 \cup T_2))} D_a(S, T_2)$$
5 Hierarchies, level functions and reducibility

5.1 Hierarchies

The agglomerative clustering procedure produces a series of feasible clusterings \( C(n), C(n - 1), \ldots, C(m) \) with \( C(m) \in \text{Max } \Phi \) (maximal elements for \( \sqsubseteq \)). Their union \( T = \bigcup_{k=n}^{m} C(k) \) is a hierarchy and has the property

\[
\forall C_p, C_q \in T : C_p \cap C_q \in \{\emptyset, C_p, C_q\}
\]

The set inclusion \( \subseteq \) is a tree or hierarchical order on \( T \). The hierarchy \( T \) is complete iff \( U \in T \).

For \( W \subseteq U \) we define the smallest cluster \( C_T(W) \) from \( T \) containing \( W \) as:

\[
\begin{align*}
&c_1. \quad W \subseteq C_T(W) \\
&c_2. \quad \forall C \in T : (W \subseteq C \Rightarrow C_T(W) \subseteq C)
\end{align*}
\]

\( C_T \) is a closure on \( T \) with a special property

\[
Z \notin C_T(\{X, Y\}) \Rightarrow C_T(\{X, Y\}) \subset C_T(\{X, Y, Z\}) = C_T(\{X, Z\}) = C_T(\{Y, Z\})
\]

5.2 Level functions

A mapping \( h : T \rightarrow \mathbb{R}^+_0 \) is a level function on \( T \) iff

\[
\begin{align*}
l_1. & \quad \forall X \in U : h(\{X\}) = 0 \\
l_2. & \quad C_p \subseteq C_q \Rightarrow h(C_p) \leq h(C_q)
\end{align*}
\]

A simple example of level function is \( h(C) = \text{card}(C) - 1 \).

Each hierarchy / level function determines an ultrametric dissimilarity \( \delta \) on \( U \)

\[
\delta(X, Y) = h(C_T(\{X, Y\}))
\]

The converse is also true [7]: Let \( d \) be an ultrametric on \( U \). Denote \( \overline{B}(X, r) = \{Y \in U : d(X, Y) \leq r\} \). Then for any given set \( A \subseteq \mathbb{R}^+ \) the set

\[
C(A) = \overline{B}(X, r) : X \in U, r \in A \} \cup \{\{U\} \cup \{\{X\} : X \in U\}
\]

is a complete hierarchy, and \( h(C) = \text{diam}(C) \) is a level function.

The pair \((T, h)\) is called a dendrogram or a clustering tree because it can be visualized as a tree.

5.3 Reducibility

The dissimilarity \( D \) has the reducibility property [3] iff

\[
D(C_p, C_q) \leq \min(D(C_p, C_s), D(C_q, C_s)) \Rightarrow \\
\min(D(C_p, C_s), D(C_q, C_s)) \leq D(C_p \cup C_q, C_s)
\]

or equivalently

\[
D(C_p, C_q) \leq t, \quad D(C_p, C_s) \geq t, \quad D(C_q, C_s) \geq t \Rightarrow \quad D(C_p \cup C_q, C_s) \geq t
\]

A very important consequence of reducibility is:
If a dissimilarity $D$ has the reducibility property then $h_D$ from the agglomerative method is a level function.

All our three dissimilarities (minimum, maximum and average) have the reducibility property.

6 Nearest neighbors networks

For a given dissimilarity $d$ on the set of units $U$ and relational constraint $R$ we define [11] the $k$ nearest neighbors graph $G_{NN}^k = (U, A)$

$$(X, Y) \in A \iff X\leqslant R Y$$ and $Y$ is selected among the $k$ nearest neighbors of $X$

By setting for $(X, Y) \in A$ its value to $w((X, Y)) = d(X, Y)$ we obtain a network $N_{NN} = (U, A, w)$.

In the case of equidistant pairs of units we have to decide – or to include them all in the network, or to specify an additional selection rule. We shall denote by $G_{NN}^\ast$ the network with included all nearest equidistant pairs, and by $G_{NN}$ a network where a single nearest neighbor is always selected.

6.1 Structure and properties of the nearest neighbor networks

Let $N_{NN} = (U, A, w)$ be a single nearest neighbor network. A pair of units $X, Y \in U$ are reciprocal nearest neighbors or RNNs iff $(X, Y) \in A$ and $(Y, X) \in A$.

Suppose $\text{card}(U) > 1$ and $R$ has no isolated units. Then in $N_{NN}$

- every unit/vertex $X \in U$ has the $	ext{deg}(X) \geq 1$ – there is no isolated unit;
- along every walk the values of $w$ are not increasing.

using these two observations we can show that in $N_{NN}^\ast$:

- all the values of $w$ on a closed walk are the same and all its arcs are reciprocal – all arcs between units in a nontrivial (at least 2 units) strong component are reciprocal;
- every maximal (can not be extended) elementary (no arc is repeated) walk ends in a RNNs pair;
- there exists at least one RNNs pair corresponding to $\text{min}_{X, Y \in U, X \neq Y} d(X, Y)$.

6.2 Fast agglomerative clustering algorithms

Any network $N_{NN}$ is a subnetwork of $N_{NN}^\ast$. Its connected components are directed trees with a single RNNs pair in the root.

Based on the nearest neighbor network very efficient algorithms for agglomerative clustering for methods with the reducibility property can be built.

chain := []; W := U;
while card(W) > 1 do begin
  if chain = [] then select an arbitrary unit $X \in W$ else $X := \text{last}(chain)$;
  ...
grow a NN-chain from X until a pair (Y, Z) of RNNs are obtained; agglomerate Y and Z:

\[ T := Y \cup Z; \quad W := W \setminus \{Y, Z\} \cup \{T\}; \quad \text{compute } D(T, W), W \in W \]

end;

It can be shown that if the clustering method has the reducibility property then the NN-chain remains a NN-chain also after the agglomeration of the RNNs pair.

This algorithm was implemented in Pajek – a program for analysis and visualization of large networks [2]. The detailed algorithm is presented in the Appendix.

6.3 Example: US counties \( t = 200 \)

For an example we present the clustering of 3111 mainland US counties into contiguous clusters ("regions") considering the counties neighboring relation [6] and as dissimilarity the euclidean distance between standardized values of the following three variables (US Census 2000) [5]: V47 – Percent of White, V125 – Educational attainment 1990, V126 – Household income. The neighboring relation is of the \textit{queen} type – two counties are considered neighbors if their borders have at least one point in common [4]. In this case, to determine the contiguous clusters, the tolerant strategy was used and the average method for updating the dissimilarity. From the obtained dendrogram we determined the clustering corresponding to the cut at level 200. All clusters containing a single county were collected in cluster 0 (cyan).

In Figure 3 the obtained clustering is presented. The clusters are represented by vertex (county) color and dissimilarities by the edge thickness and darkness. In Figure 4 a selected section from Figure 3 is displayed. The original color picture in pdf format is available on the web page

http://pajek.imfm.si/doku.php?id=examples

7 Hierarchical clustering of two-mode networks

At the Bertinoro workshop on graph drawing (9-14. March 2008) Katharina Zweig (Lehmann) asked for a method for clustering bipartite graphs. It turned out that our clustering method can be easily adapted to solve also this problem.

Let \(((U, V), L, d)\), \(d : L \to \mathbb{R}_0^+\) be a weighted two-mode (bipartite) network. \(d\) is a dissimilarity measure between linked units (vertices). As an example of such dissimilarity we can take \(d(u, v) = d(v, u) = w^* - w_4(u, v)\) where \(w_4(u, v)\) is the number of different 4-cycles containing the line \((u, v)\) and \(w^* = \max_{(u, v) \in L} w_4(u, v)\).

We call a \textit{two-mode partition} a set

\[ C = \{(C_1, D_1), (C_2, D_2), \ldots, (C_k, D_k)\} \]

such that the sets \(\{C_i\} \setminus \emptyset\) and \(\{D_j\} \setminus \emptyset\) are partitions of sets \(U\) and \(V\); and \((\emptyset, \emptyset) \notin C\).

We start the hierarchical clustering with the singeltons partition

\[ C_N = U^{(1)} \times \{\emptyset\} \cup \{\emptyset\} \times V^{(1)} = \{\{u_1\}, \emptyset\), \{u_2\}, \emptyset\), \ldots, \{\emptyset, \{v_n\}\} \}

where \(U^{(1)} = \{\{u\} : u \in U\}\) and \(N = |U \cup V|\).
Figure 3: Example: US counties $t = 200$
The fusibility condition is now \( \psi((C_i, D_i), (C_j, D_j)) = C_i RD_j \lor C_j RD_i \) where \( C_i RD_j \equiv \exists u \in C_i \exists v \in D_j : (u, v) \in L \).

The relation \( R \) update rules are

\[
C_{pq}RC_s = C_pRC_s \lor C_qRC_s \quad \text{and} \quad C_qRC_{pq} = C_qRC_p \lor C_qRC_q
\]

For a pair of two-mode clusters \((C_p, D_p)\) and \((C_q, D_q)\), such that \( C_p RD_q \), let \( \delta(p, q) \) denote the difference of cluster \( p \) from cluster \( q \). \( \delta \) is determined as follows

\[
\delta(\{u\}, \{v\}) = d(u, v)
\]

and the update formulae for \( \delta \) after merging (see Figure 5) are: if \( C_p RD_q \land C_q RD_p \) then

\[
\delta((pq), s) = \oplus(\delta(p, s), \delta(q, s)) \quad \text{and} \quad \delta(s, (pq)) = \oplus(\delta(s, p), \delta(s, q))
\]

where \( \oplus \in \{\min, \max, \text{ave}\} \); otherwise only values of existing links are considered.

Using \( \delta \) the clustering dissimilarity \( D(p, q) \) is defined as

\[
D(p, q) = \oplus(\delta(p, q), \delta(q, p))
\]

Again the link needs not to exist in both 'directions'.
8 Conclusions

This paper is an extended and detailed version of the talks presented at: 6th Slovenian International Conference on Graph Theory, Bled, Slovenia, 24 – 30 June 2007; International workshop on Detection and visualization of communities in large complex networks, UCL, Université catholique de Louvain, Louvain-la-Neuve, Belgium, March 13-14, 2008; and IFCS 2009, Dresden, March 13-18, 2009.

Appendix: Fast agglomerative algorithm for clustering with relational constraints

In this appendix we describe in pascal-like form the main part of an implementation of the fast agglomerative algorithm for clustering with relational constraints.

The variables used in the algorithm have the following meaning:
- \( n \) – number of units/vertices;
- \( w \) – boolean array \([1..2n-1]\), vertices already inspected;
- \( stack \) – current chain of nearest neighbors;
- \( nt \) – number of units + number of tree (dendrogram) vertices;
- \( np \) – number of inspected vertices;
- \( nw \) – index of the first not yet inspected vertex in \( w \);
- \( dmax \) – some very large number (infinity);
- \( father \) – integer array \([1..2n-1]\), tree represented by ‘links to fathers’;
- \( h \) – real array \([n + 1..2n-1]\), level of internal dendrogram vertices;
- \( strategy \) – relation adjustment: tolerant, leaders, strict, two-way;
- \( method \) – dissimilarity update: minimum, maximum, average;

The procedures
- \( \text{AddGraph} \) adds a vertex \( T \) to the network using selected strategy. The vertices \( X \) and \( Y \) are not deleted yet because they are needed for computing the corrected dissimilarities \( D \).
- \( \text{top}(stack) \) returns the top value from \( stack \); the value is not removed from the stack. To remove it \( \text{pop}(stack) \) is used.
- In the statement \( \text{dmin} := \text{weight}(Y,X) \); we have to consider both directions – \((Y,X)\) and \((X,Y)\).
stack := []; for i := 1 to 2*n-1 do w[i] := false; np := 0; nt := n; nw := 1; while np < nt do begin if empty(stack) then begin while w[nw] do nw := nw + 1; X := nw; end else begin X := top(stack); pop(stack); end; if empty(AllNeighbors(X)) then begin w[X] := true; np := np + 1; end else begin {determine the RNN pair \( (Y, Z) \)} if empty(stack) then begin Y := 0; dmin := dmax; end else begin Y := top(stack); dmin := weight(Y,X); end; U := X; push(stack,X); found := false; repeat if empty(AllNeighbors(U)) then begin Z := U; found := true; end else begin for V in AllNeighbors(U) do if weight(U,V) < dmin then begin dmin := weight(U,V); T := V; end; if T=U then T := Y; if T=Y then begin Z := U; found := true; end else begin Y := U; U := T; push(stack,T); T := Y; end; end; until found; {join the RNN pair \( T=(Y, Z) \)} if not (Z in OutNeighbors(Y)) then begin U := Z; Z := Y; Y := U; end; nt := nt + 1; T := nt; father[Y] := T; father[Z] := T; h[T] := weight(Y,Z); w[Y] := true; w[Z] := true; np := np + 2; AddtoGraph(Y,Z,T,strategy); for V in AllNeighbors(T) do weight(T,V) := D(Y,Z,T,V,method); RemovefromGraph(Y); RemovefromGraph(Z); pop(stack); pop(stack); push(stack,T); end; end;

Table 1: Detailed clustering algorithm
References


