Community retrieval and visualization in large graphs

Nathalie Villa\textsuperscript{1,2}, Taoufiq Dkaki\textsuperscript{3}, Sébastien Gadat\textsuperscript{1}, Jean-Michel Inglebert\textsuperscript{3}, Quoc-Dinh Truong\textsuperscript{4}

\textsuperscript{1} Institut de Mathématiques de Toulouse, UMR CNRS 5219, Université de Toulouse (Toulouse III), 118 route de Narbonne, F-31062 Toulouse cedex 9, France.
\textsuperscript{2} Toulouse School of Economics, Manufacture des tabacs, 21 allées de Brienne, F-31000 Toulouse, France.
\textsuperscript{3} Institut de Recherche en Informatique de Toulouse, Université Toulouse III, 118 Route de Narbonne, F-31062 Toulouse cedex 4, France.
\textsuperscript{4} Université de Cantho, 1 Rue Ly Tu Trong, Cantho, Vietnam.

Corresponding author: Nathalie Villa
E-mails: nathalie.villa@math.univ-toulouse.fr, dkaki@irit.fr, sebastien.gadat@math.univ-toulouse.fr, inglebert@iut-blagnac.fr, truong@univ-tlse2.fr

Key words
Graphs, Graph drawing, Clustering, Social networks, Communities, Data mining

Abstract

This paper deals with the analysis and the visualization of large graphs. Our interest in such a subject-matter is related to the fact that graphs are convenient widespread data structures. Indeed, this type of data can be encountered in a growing number of concrete problems: Web, information retrieval, social networks, biological interaction networks… Furthermore, the size of these graphs becomes increasingly large as the progression of the means for data gathering and storage steadily strengthens. This calls for new methods in graph analysis and visualization which are now important and dynamic research fields at the interface of many disciplines such as mathematics, statistics, computer science and sociology. In this paper, we propose a method for graphs representation and visualization based on a priori clustering of the vertices. Newman and Girvan (2004) points out that “reducing [the] level of complexity [of a network] to one that can be interpreted readily by the human eye, will be invaluable in helping us to understand the large-scale structure of these new network data”: we rely on this assumption to use a priori a clustering of the vertices as a preliminary step for simplifying the representation of the graphs – as a whole. The clustering phase consists in optimizing a quality measure specifically suitable for the research of dense groups in graphs. This quality measure is the modularity and expresses the “distance” to a null model in which the graph edges do not depend on the clustering. The modularity has shown its relevance in solving the problem of uncovering dense groups in a graph. Optimization of the modularity is done through a stochastic simulated annealing algorithm. The visualization/representation phase, as such, is based on a force-directed algorithm described in Truong et al. (2007). After giving a short introduction to the problem and detailing the vertices clustering and representation algorithms, the paper will introduce and discuss two applications from the social network field.
Introduction

Graph clustering has been studied in numerous scientific works and scientific communities. It has been presented as a way to simplify the structure of large graphs where the number of vertices – up to several thousand – goes beyond both human analytical capabilities and machine displaying capacities. The main target of such a clustering is to bring out graphs macroscopic structure (see Newman and Givan (2004)) and highlight “communities” – to use a terminology commonly used in the field of social networks. Although there is no consensual formal definition of what a community is, it generally refers to a subset of vertices that are strongly interconnected while being weakly connected to the rest of the vertices. There is no a priori Euclidean structure defined over the set of graph vertices. Yet, this is required prior to the implementation of any clustering method. This stresses the need for ad-hoc models and methods to find such subgraphs as usual multidimensional clustering methods can not be used directly with graph objects. There is a wide variety of methods that propose satisfactory solutions to the problem of partitioning a set of graph vertices into several clusters: Schaeffer (2007) presents several methods based on defining similarities between vertices or on the optimization of quality criteria. Among these methods, spectral clustering (see von Luxburg (2007)) is an approach that has become very popular but is very sensitive to the size of the data since it requires spectral decomposition of a square matrix whose size is equal to the number of vertices of the graph.

In this article, we propose an original method for graph vertices clustering. Our method is based on an ad-hoc quality measure: modularity. The originality of our approach is the use of a stochastic algorithm to optimize this modularity that allows its implementation for graphs whose size may be large. We show how this clustering method may be combined with a visualization algorithm to represent graphs whose vertices have been partitioned into classes (clusters) to produce an intelligible representation that emphasizes the structure of the whole graph, the main groups – set during the clustering phase – along with the relationships between clusters and between vertices belonging to a same cluster.

Clustering the vertices of a large graph

The modularity as a measure of quality of a clustering

In Schaeffer (2007), the author reviews the main methods for graph clustering: some have focused on the construction of a similarity measure or on a distance between vertices so that they can directly apply existing methods for the clustering of data defined by arrays of similarities (see, e.g., Capoccia et al. (2005), Dong et al. (2006) or Lakroum et al. (2005) for examples of such approaches). An alternative approach is to optimize a measure of quality of the clustering. This measure is derived from the structure of the graph, i.e. the data from its edges. Typically, we can minimize the “cut” of the edges between clusters: suppose $G$ is an undirected weighted graph of size $n$ of vertices set $V = \{1, 2, ..., n\}$ and where the set of edges, $E$, is described by the weight matrix $W$ (symmetric, having zero diagonal and with all values positive or null). For a clustering $C$, we denote $c(i)$ the cluster containing $i \in V$ with $c(i) \in \{C_1, ..., C_p\}$. The graph cut of $G$, according to the classification $C$, is:

$$\frac{1}{2} \sum_{i,j \in V: c(i) \neq c(j)} W_{ij}.$$

This measure of quality is strongly linked to the definition of a graph embedding in a Euclidean space (see von Luxburg (2007)) and thus, the optimization of the graph cut is often approximated by applying clustering methods to this embedding, such as the k-means algorithm. This approach is known as spectral clustering.

If the spectral clustering is appealing in its simplicity and its very straightforward interpretation, it can be, however, somewhat disappointing in practice. Several drawbacks of spectral clustering are described in Newman (2006): they are related to the fact that the strategy for edge pruning considers all the edges in a similar way although Newman points out that being tied to a
vertex of a high degree is less exceptional than being linked to a vertex of lower degree. To reflect this, Newman introduces a measure of quality based on the computation of a distance to a null model in which the distribution of the edges between and within clusters depends only on the degree of the vertices. For given vertices clusters, \( C_1, ..., C_p \), the modularity is the quantity:

\[
Q = \frac{1}{2m} \sum_{k=1}^{p} \sum_{i,j \in C_k, i \neq j} (W_{ij} - P_{ij})
\]

where \( m \) is the total number of weights in the graph and \( P_{ij} \) is the expected weight of intra-cluster edges between vertices \( i \) and \( j \) in the null model (the distribution of edges in the null model depends only on the degree not on the clusters). Thus, a high and positive \( Q \) value reflects an over-representation (in comparison to the null model) of edges within the clusters, whereas a strong and negative \( Q \) value reveals an over-representation of edges outside the clusters. The notion of communities, described above, corresponds to the first case (a high and positive \( Q \) value).

The choice of \( (P_{ij}) \) values is based on assumptions corresponding to the null model. Reasonable constraints for the choice of these values are usually:

- \( \sum_{i \in V} P_{ij} = 2m \) (i.e., the sum of the expected weights in the null model is identical to the sum of the weights in the graph);
- \( \sum_{j \in V} P_{ij} = d_i \) where, for each \( i \in V, d_i \) is the degree of the vertex \( i: d_i = \sum_{j \in V} W_{ij} \). This assumption means that the degree of each vertex of the null model is the same as the degree in the considered graph.

We assume, by hypothesis, that \( P_{ij} \) is of the form \( f(d_i) f(d_j) \) which means, on the one hand, that the expected weights in the null model depend only on the degrees of considered vertices, not on their distribution among different clusters while, on the other hand, the weights between two given edges are independent. Newman (2006) shows that in this case, \( P_{ij} \) are given by:

\[
P_{ij} = \frac{d_i d_j}{2m}.
\]

The optimization of \( Q \) over all partitions of the graph vertices cannot reasonably be achieved by an exhaustive search over all possible partitions of the graph into \( p \) clusters (see Newman (2006)). Various approaches to approximate this optimization problem have recently been proposed. Among these approaches, Newman and Girvan (2004) propose a hierarchical optimization, which may lead to rather unsuitable solutions (see the example given in Newman (2006)). Newman (2006) presents an approach based on a spectral decomposition, and Lehmann and Hansen (2007) develop a deterministic annealing optimization.

In the following, we present a simulated annealing approach as a way to optimize the modularity. This stochastic approach has the advantage of requiring only a few operations of low complexity: it is therefore particularly well suited for large data.

**Optimizing modularity using simulated annealing**

Simulated annealing (see Kirkpatrick et al. (1983)) is a stochastic algorithm designed to optimize a cost function when other simpler algorithms (such as gradient descent) are inaccurate. It is particularly useful when the possibility space is discrete or when the cost function has many local maxima. The optimization of the modularity in the space of possible partitions of the graph is a typical framework for applying a simulated annealing approach. In the following, we briefly present the algorithm of simulated annealing. We advice the reader interested in this approach to refer to Kirkpatrick et al. (1983) for further details.

Similarly to the algorithm of Metropolis-Hastings (see Hastings (1970)), the principle of the simulated annealing algorithm consists in simulating, using a Markov chain, a variable taking its values in the set of all possible partitions of the graph vertices. This Markov chain has the invariant probability law:

\[
\mu_t(C) = \frac{e^{Q(C,G)/T}}{Z}
\]
where $Q(C, G)$ is the modularity over the clustering $C$ of the graph $G$. $T$ is a positive temperature parameter and $Z$ a normalization constant which is not to be computed. The interest of this law appears when temperature parameter $T$ tends toward 0: the Markov chain has a tendency to freeze on states of high probability, i.e., on maxima of $Q$. A generally accepted choice for the decreasing scheme of temperature parameter is $T_l = \frac{\gamma}{\log l}$, where $l$ is the $l$th state of the Markov chain (i.e., the number of iterations of the algorithm) and $\gamma$ is a positive constant.

The practical implementation of this Markov chain is done through the method of “acceptance / rejection”: given a clustering $C_i$ of the graph, a neighboring clustering, $C_F$, is randomly chosen according to a reversible process. The clustering $C_F$ is chosen from $C_i$ with a probability equal to $P(C_F, C_i)$ (for the sake of simplicity, we will assume that $P(C_F, C_i) = P(C_i, C_F)$).

Finally, the clustering $C_F$ is accepted (that is to say, it is used as the new state of the Markov chain) with a probability equal to

$$
\min \left(1, e^{\frac{\Delta Q(C_i, C_F, Q)}{T}} \right)
$$

where $\Delta Q(C_i, C_F, Q)$ is the modularity difference between the clusterings $C_i$ and $C_F$ ($\Delta Q(C_i, C_F, Q) = Q(C_F, G) - Q(C_i, G)$). The algorithm for optimizing modularity by simulated annealing is described in Algorithm 1.

**Algorithm 1: Simulated annealing for the optimization of the modularity**

1. Randomly select an initial partition of $V$ into $p$ clusters: $C_i^0, \ldots, C_p^0$.
2. Compute the modularity associated to this partition: $Q(C_0^0, \ldots, C_p^0, G)$.
3. For $l = 1 \ldots L$, repeat:
   a. Randomly select two clusters $C_i^{l-1}$ and $C_j^{l-1}$ among $(C_i^{l-1})_{i=1}^p$.
   b. If $C_i^{l-1} \cup C_j^{l-1} \neq \emptyset$, randomly select a vertex $x \in C_i^{l-1} \cup C_j^{l-1}$; in the remainder, we assume, without loss of generality, that $x \in C_i^{l-1}$.
   c. Define a new clustering so that $C_k = C_k^{l-1}$ for all $k \neq i, j$, $C_i = C_i^{l-1} \setminus \{x\}$ and $C_j = C_j^{l-1} \cup \{x\}$ (x shifts from cluster $i$ to cluster $j$).
   d. Compute $\Delta Q = Q(C_1^l, \ldots, C_p^l, G) - Q(C_1^{l-1}, \ldots, C_p^{l-1}, G)$.
   e. In the case where:
      - $\Delta Q > 0$, accept the new clustering: $C_k^l = C_k^l$ for all $k$ and compute $Q(C_1^l, \ldots, C_p^l, G) = \Delta Q + Q(C_1^{l-1}, \ldots, C_p^{l-1}, G)$.
      - $\Delta Q < 0$, accept the new clustering with the probability $e^{\gamma \log(l) \Delta Q}$ ($\gamma$ is a positive constant) or keep the previous state. Compute $Q(C_1^l, \ldots, C_p^l, G)$. 


Step 3.d of Algorithm 1 fortunately does not require the full calculation of the modularity $Q((C_1,...,C_p),G)$. Indeed, Lehmann and Hansen (2007) shows that the modularity can be used in the matrix form

$$Q((C_1,...,C_p),G)=\frac{1}{2m}\text{Tr}(S^TBS)$$

where $S$ is the matrix of size $p \times n$ whose elements are $S_{ik}=\begin{cases} 1 & \text{si } x \in C_k \\ 0 & \text{ sinon} \end{cases}$ for all $x \in V$ and all $k \in \{1,...,p\}$ (in other words, $S$ is the matrix that associates each vertex to its corresponding cluster), $B$ is $(W-P)$ and $\text{Tr}(S^TBS)$ is the trace of matrix $S^TBS$. Thus, step 3.d of Algorithm 1 can be reduced to the computation of:

$$\Delta Q = \frac{1}{m} \sum_{y \neq x} (S_{xy}^1 - S_{xy}^0) = \frac{1}{m} \left( \sum_{y \in C_k} B_{xy} - \sum_{y \in C_j \setminus \{x\}} B_{xy} \right).$$

This requires less than $(n-1)$ additions which is far less than a modularity computation over a given partition (O($n^2$) operations required).

**A simulated example**

We propose to analyze the quality of the clustering obtained using the approach described in Algorithm 1. To do so, we use a random graph model (undirected and unweighted) that is acceptable from the viewpoint of social networks paradigm: the “p-planted partition model” (see Condon and Karp (2001)). A random graph generated by this model depends on 4 parameters labeled $p$ (the number of classes that will be called natural classes in the following), $k$ (the number of vertices in each class), $r$ (the probability of intra-class edges) and $q$ (the probability of inter-class edges). In the remainder, we note this model $G(p,k,r,q)$. An illustration of this model with 3 classes of 4 vertices each is given in Figure 1.

![Figure 1: An example of a planted $p$-partition graph: this graph is made of 3 classes of 4 vertices each with a proportion of intra-class edges of about 58% and a proportion of inter-class edges of about 10%.](image-url)

To assess whether the Algorithm 1 is able to find the natural classes of a planted $p$-partition graph, we generated 50 random graphs for two fixed sets of parameters of this model. The two tested sets of parameters were:
• \( G(5, 100, 0.7, 0.02) \): the 50 generated random graphs have an average modularity according to their natural partitions of about 0.70. This value is consistent with the values found in real social networks.
• \( G(50, 10, 0.7, 0.2) \): the 50 generated random graphs have an average modularity according to their natural partitions of about 0.37. This value is below the values generally observed in real social networks. This demonstrates that the process of finding natural classes in this model may be hard.

In addition, the algorithm was tested with the correct number of natural classes \((p_{\text{algo}} = p)\) and with a higher number \((p_{\text{algo}} > p)\) in order to see whether the algorithm is sensitive to the knowledge of the adequate number of classes. Lastly, the quality of the rebuilt clustering was measured in two different ways:
• by measuring the rate of bad clustering of the vertices (percentage of vertices which are not clustered in the cluster assigned to most vertices of the natural class that they belong to),
• by measuring the difference between the modularity of the natural classes and the modularity of the clustering provided by the algorithm (this quality measurement will be called modularity defect).

The results obtained are summarized in Table 1. The rate of bad clustering of the vertices in their natural classes is very satisfactory, with less than 10% of vertices wrongly classified even in the worst situation (the one described by the last column of the table below). In the same way, the modularity is correctly optimized since the defect of modularity accounts for approximately 10% of the modularity expected for the most difficult context (last column) against approximately 5% for the easiest situation (first column). Lastly, even in the situation where the number of clusters sought by the algorithm is much higher than the number of natural classes (second column), the algorithm correctly behaves and tends automatically to decrease the suggested number of clusters.

<table>
<thead>
<tr>
<th>( R \times q )</th>
<th>( 0.7 \times 0.02 )</th>
<th>( 0.7 \times 0.02 )</th>
<th>( 0.7 \times 0.2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p \times k )</td>
<td>5 \times 100</td>
<td>5 \times 100</td>
<td>50 \times 10</td>
</tr>
<tr>
<td>( p_{\text{algo}} )</td>
<td>5</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Clustering error rates</td>
<td>1.36 %</td>
<td>2.34 %</td>
<td>6.32 %</td>
</tr>
<tr>
<td>Modularity defect</td>
<td>0.036</td>
<td>0.040</td>
<td>0.036</td>
</tr>
</tbody>
</table>

Table 1: Mean performances of the modularity optimization for the “\(p\)-planted partition” graph model.

The results of this section show that using simulated annealing gives good performances. In the remainder, we will explain how to use a relevant clustering obtained by this approach to represent the graph so as to emphasize its main dense subgroups.

A representation of clustered graphs

It is admitted that visualization strongly contributes to the effectiveness of the analysis and the exploration of data. As a consequence, several works have focused on graph drawing. Building a graph drawing or visualization mainly consists in positioning the vertices and the edges of a graph in a space of display according to criteria and constraints whose objective is to offer the most easily interpretable visualizations. The most known works are based on the so-called force-directed approach. In this approach, that can be accredited to Eades (see, for example, Eades (1984)), vertices are compared to rings and edges to springs. The vertices are often placed in a random initial position and are left free so that the forces of the springs move them towards a state of energy equilibrium of the system of rings and springs. Another variant of this method, called “attraction and repulsion”, consists in modifying the way in which the forces are applied: repelling forces are activated for all pairs of nodes while attraction forces are set only for connected vertices.

These force-based algorithms are successfully used for drawing graphs that have relatively few vertices and edges. The difficulty in visualizing a graph depends indeed on its size which is the
major obstacle that can prevent from completing the objective of visualization: achieving readable representation. The larger the number of edges and vertices is, the more the goals of visibility and of quality of interpretation are difficult to achieve. To deal with graphs containing more than a few hundreds of vertices, one has to resort either to more effective algorithms (see, for example Hachul and Juger (2006)), or to a preliminary clustering which goal is to reduce the complexity of the graph. This reduction of complexity often consists in considering the graph of the clusters rather than the initial graph. Indubitably, the reduction of complexity leads to a loss of information. Our purpose is to minimize this loss by using algorithms dedicated to the visualization of “clustered” graphs. This type of algorithm provides a visualization which makes it possible to appreciate not only the place of each vertex in the graph (with its relations to its neighbors) but also the community structure of the graph. Thus, this type of graph visualization emphasizes the relations between the communities (or clusters) and helps to understand the influence of the membership to the clusters on the position of each node. This type of graph visualization profits from the simplification of the graph structure brought by the clustering stage while preserving at most the representation of the graph as a whole.

Actually, very little works were dedicated to the visualization of clustered graphs. Among these works, those of Noack (2004), Chuang et al. (2004) and of Eades and Huang (2000) are the most representative ones. These papers are all based on Kamada and Kawai (1989), a traditional force model, and they add a virtual vertex at the center of each cluster and connect each of these new vertices to the vertices of the cluster to which it is associated. These models give good performances for graphs whose clusters are much denser than the entire graph. Thus, in addition to the adding of virtual nodes whose significance is not really defined in a concrete way, this approach appears unusable when the ratio of the density in the clusters to the total density of the graph is not significantly high. That can be the case in many concrete examples: this is often the case when the number of clusters is small – with respect to the size of the graph – or when the clustering does not depend on the graph structure (i.e., does not depend on the relationships that bind the graph vertices).

In this work, we propose an alternative method of visualization which is able to manage the display of clustered graphs for any ratio between the densities of the clusters and those of the whole graph. This approach is presented in Truong et al. (2007). This model is based on the Fruchterman’s force model (see Fruchterman and Reingold (1991)) which associates to each cluster a predetermined area of visualization in which each vertex will be put. However, the proposed algorithm presents two differences comparing to the one of Fruchterman. First of all, implemented forces are of two types: those that take place “between the vertices of the same cluster” on the one hand and those corresponding to “any two vertices of different clusters” on the other hand. These forces are characterized by different intensities. Secondly, vertices are constrained inside the areas associated to the clusters they belong to by adding repelling forces between these vertices and the border of their associated areas. To do so, the vertices and the
borders of the areas associated with the various clusters are charged with an electric charge of the same polarity. This ensures that each node always remains inside the area to which it is initially associated – thanks to the electrically induced repulsion forces. Figure 2 summarizes the entire force model. This approach has been combined with the method of graph clustering previously described in order to obtain an overall representation of a social network divided into communities.

Applications

Representing a social network from Hugo’s novel, “Les Misérables”

In this first example, we deal with a simple and well known example to show how the combination of clustering and visualization tools leads to a simpler and more valuable representation of the underlying structures of a graph. This example is about a social network described in Knuth (1993) and resulting from “Les Misérables”, a novel by the famous french writer Victor Hugo. This weighted graph counts simultaneous appearances of the 77 characters of the novel.

The first phase of the analysis consists in using the algorithm of simulated annealing for clustering the vertices of the graph. This clustering was performed with:

- various values of the number of clusters (varying from 5 to 10). The modularity being a non monotonous function of the number of clusters this allows an optimal selection (according to modularity) of the number of clusters.
- various values of the parameter $\gamma$ of Algorithm 1.

Figure 3: Initial representation of the graph obtained by Tulip (force directed algorithm)

In addition, several iterations of the algorithm were carried out: the algorithm being based on a stochastic process, each iteration produces a potentially different solution. Repetitions aim at approaching as closely as possible, the optimal solution within a finite time.
Figure 4: Evolution of the optimal modularity (within values of $\gamma$ and repetitions) in function of the number of clusters.

Figure 4 presents the evolution of the optimal modularity according to the initial number of clusters. The optimal modularity is obtained for an initialization with 9 clusters while the actual optimal clustering contains only 8 clusters. The evolution of the optimal modularity (Figure 4) and the fact that the optimal solution presents a number of clusters lower than the initial number of clusters, show that the search for a better clustering with a number of clusters higher than 10 is useless.

Figure 5: Representation of the graph of clusters using a force directed algorithm (the size of the squares is proportional to the number of vertices in each cluster and the thickness of the edges is proportional to the number of edges between clusters).

Once a clustering is obtained, each cluster is represented using a symbol (here a square) whose surface is proportional to the number of vertices in this cluster. A force-directed algorithm has
been used to place these clusters in a readable way while emphasizing the relations existing between clusters: Figure 5 is the result of such a process. Here the visualization was carried out using Tulip\(^1\) (see Auber (2001)). The representation emphasizes an important and central group around which the other groups evolve. 3 other big sized groups are also easily distinguishable: they organize the relations around the main group.

Figure 6: Representation using the modified force-directed algorithm for clustered graphs

To represent the whole graph using the algorithm described in section 3, we use the positions of the clusters given by the force algorithm used for the representation of the graph of the clusters (Figure 5). The final representation of the clustered graph is given in Figure 6. We can compare this representation with the initial representation obtained by a traditional force-directed algorithm (Figure 3). If some dense groups are easily recognizable in the two representations (for example the group which is organized around Myriel), the clustered representation of the graph helps to identify more clearly groups in the most dense part of the graph. It emphasizes the central positions of several characters of the principal group: Valjean, Cosette, Marius, Javert (…), as well as the relative proximity of these central characters. Indeed, the principal frame of the novel is organized around the life of Jean Valjean, former convict, who is pursued for years by the uncompromising Javert. Valjean finally died in the arms of his protégée, young Cosette, and of Marius, a student who fell in love with her. Except for Valjean, this central position is not that clear in the initial graph. The three principal subgroups in Figure 6 are as many secondary stories revolving around this principal story. The subgroup located in the utmost bottom of the graph is organized around Gavroche, street child who is capable of real gestures of generosity towards Mabeuf (also in this group) and two lost children, called “Child 1” and “Child 2”. A second principal group is just above and on the right of Gavroche’s group. It is organized around Fantine, a little employee who was forced to entrust her daughter Cosette to some strangers. These strangers can be found in the third important group, located at the left

\(^1\) Tulip is a free software available at [http://tulip.labri.fr/](http://tulip.labri.fr/)
of the principal group: they are the Thénardier. Their presence in this group influences the position of Cosette at the left part of its own cluster.

Thus, the representation of the clustered graph is easily related to the story in Victor Hugo’s novel. It clearly emphasizes the main characters and the secondary stories. This simple example shows how a preliminary phase of clustering can contribute to a representation of the graph that can improve the structures of the drawing and simplify the interpretation and the analysis of graphs.

Representing a scientific collaboration network

In this section, we examine another example, simple but more realistic, in order to show that the approach is effective for graphs of several hundreds of vertices. This example has been already used in Newman (2007). It deals with a network of scientists working around social networks. A weighted undirected graph is built to model the collaborations between them. The largest connected component of this graph contains 379 scientists. A representation based on a force-directed algorithm (using Tulip) is given in Figure 7 (without label) and in Figure 8 (with labels).

![Figure 7: Initial representation of the scientific collaboration network without labels](image)

First, the simulated annealing algorithm is applied to optimize the modularity. In order to obtain small-sized clusters, the algorithm was performed with an initial value of 20 clusters. Several random initializations were carried out and several values of the parameter $\gamma$ of the simulated annealing algorithm were tested. Best clustering yields a modularity value equals to 0.816, which is an optimal expected value for this type of network (see Newman and Girvan (2004)). Figure 9 shows the convergence speed of the simulated annealing algorithm and Figure 10 shows the representation of the graph of clusters by a traditional force-directed algorithm. We can notice that there are some isolated groups and very central groups around which the network is arranged in a star shape. This phenomenon was already perceptible on the initial representation of the graph but it is, on the clusters representation, more obvious and readable. This shows the validity and the usefulness of such an approach for the user.

Data are available at [http://www-personal.umich.edu/~mejn/netdata/netscience.zip](http://www-personal.umich.edu/~mejn/netdata/netscience.zip).
Figure 8: Initial representation of the scientific collaboration network with labels

Figure 9: Convergence of the simulated annealing algorithm while clustering the graph in 20 clusters
To represent the whole graph starting from the clustering step, we use the positions of the clusters given in Figure 10 that are generated by using a force-directed algorithm for the purpose of the representation of the graph of the clusters.

Figure 11 is the final result of the representation of the clustered graph. The advantage of the representation of the whole graph rather than the sole representation of the graph of clusters clearly appears: local densities of edges are more easily recognized, either inside or outside the clusters. Compared to the initial representation, this one seems to somewhat lose in clarity but has the advantage of emphasizing communities and relations between communities. It is a valuable important information for the user (sociologist, biologist,…), who can directly visualize and interpret the results of some graph clustering. The information given by visualizing both vertices and clusters helps to understand the reasons behind the formation and the organization of the clusters.

Conclusions

This paper describes a combination of a method for clustering the vertices of a graph and of an adaptation of a method for graphs visualization to clustered graphs. This combination aims at providing an easily readable representation of large graphs that highlights both communities structure and inter-vertices relationships. The graph clustering method is based on a stochastic algorithm and a graph specific quality measure. It allows the use of our approach for the visualization of large graphs. The use of such an approach for documents or biological data is under investigation: these fields are well-known sources of interesting challenges of graph mining, especially because of the size of data they involve.

Another way of investigation consists in using a self-organizing map as an algorithm to cluster the vertices of a graph instead of the clustering algorithm presented in this paper. Examples of such SOM like algorithms that have been adapted to graphs were presented in Rossi and Villa (2009) and Villa and Rossi (2007). A similar approach to the one we presented could be implemented for the visualization of graphs after that a SOM like step has been performed. This avoids the clusters positioning phase since SOM like clustering is topologically organized and thus it handles the clusters’ placement problem. However, the advantage of an a priori choice of
the clusters position is counterbalanced by the fact that the shape of the map is chosen \textit{a priori} and can be not well suitable to the representation of the studied graph.

![Figure 11: Representation of the clustered graph](image)

References


