DANCer User Manual

February 9, 2016

1 Interface

The **DANCer** generator is a synthetic graph generator for dynamic attributed networks with community structure that follows the properties of real world networks. The evolution of the network is performed using two kinds of operations: **micro operations** which are applied on the edges and vertices and **macro operations** applied on the communities. Moreover, the properties of real world networks such as preferential attachment or homophily are preserved during the evolution of the network. **DANCer** is available at `http://perso.univ-st-etienne.fr/largeron/DANC_Generator/`. It is a free software distributed under the terms of the GNU General Public Licence (version 3). It is implemented in Java and it can be executed on any platform with a Java virtual machine.

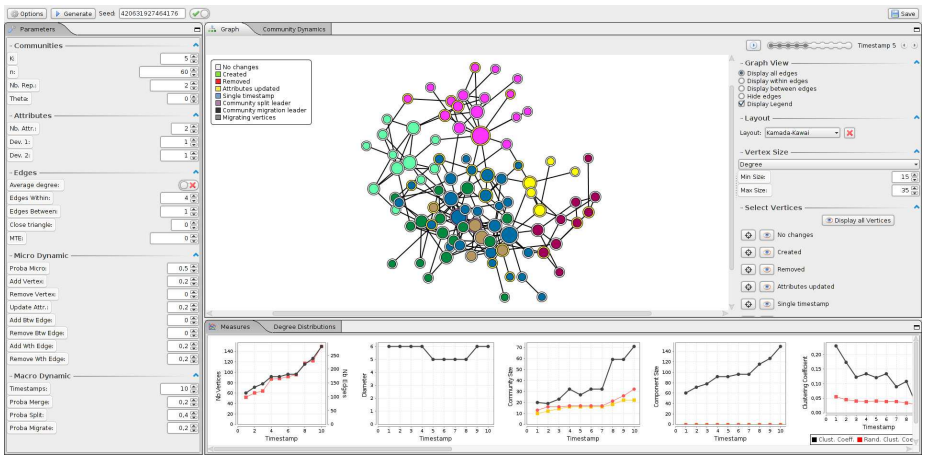


Figure 1: Parameters panel of the generator DANCer.

The user interface of **DANCer** generator is formed by three views as shown in Figure 1. The parameters are on the left panel, and are depicted in Figure 2. The user selects the generator parameters used as input for the main generator algorithm described in [1]. The central part, presented in Figure 5, displays the generated network and the graph evolution and changes in its communities at each time step. The graph measures are displayed at the bottom panel (see Figure 3) and are plotted for each graph in the sequence.

2 Parameters

The parameters panel allows to control the graph generation process as shown in Table 1. They are detailed below.

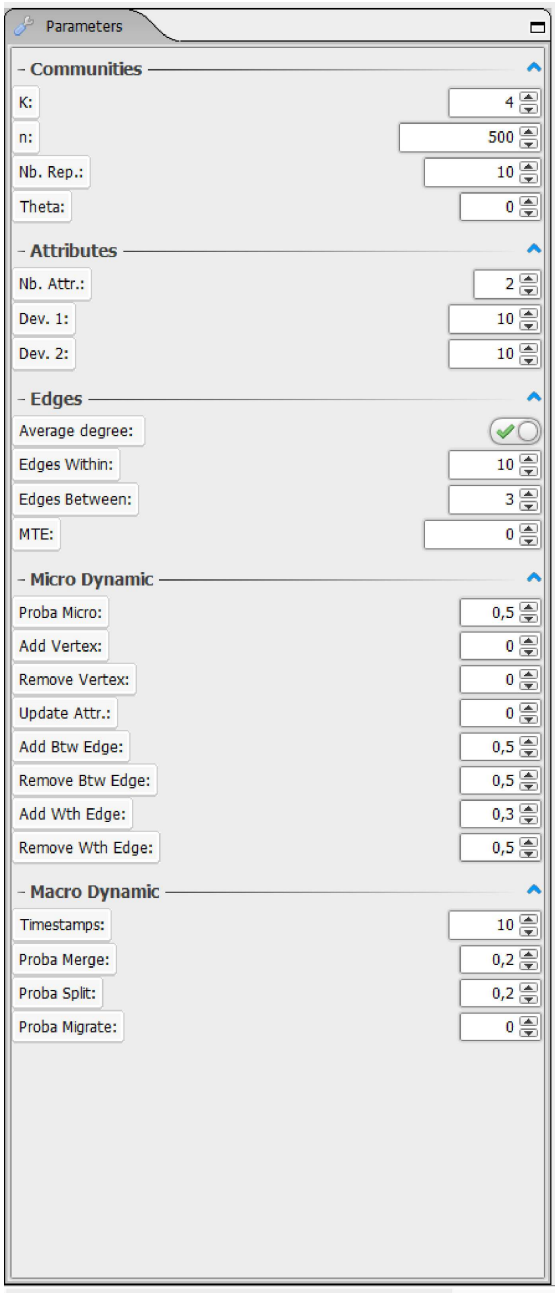


Figure 2: Parameters panel of the generator.

Parameter	Domain	Description
First timestamp (i.e., graph G_1)		
K	$\mathbb{N}^+$	Number of communities
n	$\mathbb{N}^+$	Number of vertices
Nb. Rep.	$\mathbb{N}^+$	Maximum number of representatives of each community
Theta	$[0, 1]$	A threshold to decide if a new vertex joins a randomly selected community or not
Nb. Attr.	$\mathbb{N}^+$	Number of numerical attributes
Dev. i		Standard deviations of the i^{th} attributes generated using centered normal distributions
Edges Within	$\mathbb{N}$	Maximum number of edges connecting a new vertex to vertices in its community
Edges Between	$\{0, \ldots, E_{wth}^{max}\}$	Maximum number of edges connecting a new vertex to vertices in a different community
MTE	$\mathbb{N}$	Minimum number of total edges
Micro operations		
Proba Micro	$[0, 1]$	A threshold to select if the micro dynamic updates are performed or not
Add Vertex	$[0, 1]$	Ratio defining the number of vertices inserted
Remove Vertex	$[0, 1]$	Ratio defining the number of vertices removed
Update Attr.	$[0, 1]$	Ratio defining the number of attributes updated
Add Btw. Edges	$[0, 1]$	Ratio defining the number of between edges inserted
Remove Btw. Edges	$[0, 1]$	Ratio defining the number of between edges removed
Add Wth. Edges	$[0, 1]$	Ratio defining the number of within edges inserted
Remove Wth. Edges	$[0, 1]$	Ratio defining the number of within edges removed
Macro operations		
Timestamps	$\mathbb{N}^+$	Number of graphs generated
Proba Merge	$[0, 1]$	Probability to perform the merge operation
Proba Split	$[0, 1]$	Probability to perform the split operation
Proba Migrate	$[0, 1]$	Probability to perform the migrate vertices operation

Table 1: Description of the dynamic network generator parameters

2.1 Communities

- K : Number of communities in the first graph;
- n : Number of vertices in the first graph;
- *Nb. Rep.* : Number of representents in each community. The higher is the value, the slower is the computation;
- *Theta* : Percentage of vertices assigned to a random community. The higher is this value, the less likely the community will be homogeneous w.r.t. the attributes.

2.2 Attributes

- *Nb. Attr.* : Number of real attributes associated to the vertices. Each attribute is distributed according to centered normal distribution with mean equals to 0;
- *Dev.* i : Standard deviation of the i^{th} attribute.

2.3 Edges

- *Average degree* : Select if $EdgesWithin$ or $EdgesBetween$ parameters correspond to the average degree of the vertices of the maximum number of edges added to a newly inserted vertex;
- $Edges\ Within$: Maximum number of within community edges added to a newly inserted vertex;
- *Edges Between* : Maximum number of between community edges added to a newly inserted vertex
- MTE : Minimum number of edges in the resulting graph (up to a graph where communities are cliques).

2.4 Micro Dynamic

- $Proba\ Micro$: The probability to perform a micro update operation;
- $Add\ Vertex$: The ratio of vertices created at each timestamp. When set to 1, the number of vertices is doubled at each timestamp;
- $Remove\ Vertex$: The ratio of vertices removed at each timestamp;
- *Update Attr.* : The ratio of vertices having their attribute values updated;
- *Add Btw. Edges* : The ratio of edges inserted connecting two vertices in different communities;
- *Remove Btw. Edges* : The ratio of edges removed connecting two vertices in different communities;
- *Add Wth. Edges* : The ratio of edges inserted connecting two vertices in the same communities;

- *Remove Wth. Edges* : The ratio of edges removed connecting two vertices in the same communities;

2.5 Macro Dynamic

- *Timestamps* : The number of timestamp (i.e., the number of single graphs generated to form the dynamic network);
- *Proba Merge* : The probability to perform a merge operation at a single timestamp (i.e., merging two communities into a single one);
- *Proba Split* : The probability to perform a split operation at a single timestamp (i.e., split one community into two)
- *Proba Migrate* : The probability to perform a migrate operation at a single timestamp (i.e., migrate vertices from a community to either a new or an existing community).

2.6 Network reproduction

- *Seed* parameter : A seed is used for the random number generator. It allows to reproduce exactly the same network.

3 Measures

Several measures such as modularity or homophily are computed on each graph of the dynamic network to describe its properties, notably **P1**, **P2**, **P3**, **P4** and **P5** detailed in [1]. The changes in these different measures on the sequence of graphs is presented at the bottom of the interface as Figure 3 shows.

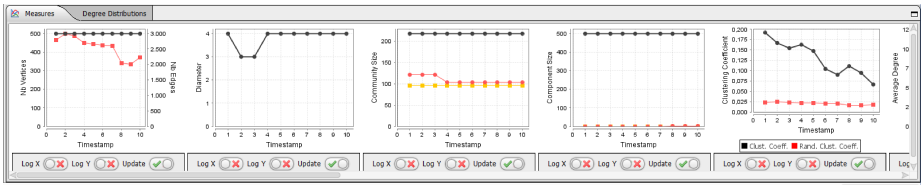


Figure 3: Measures panel of the generator.

3.1 Attribute Measures

- *Observed homophily* : Ratio of edges connecting similar vertices w.r.t. their attribute values;
- *Expected homophily* : Ratio of pair of similar vertices among all possible pairs of vertices;

  The difference between the expected and observed homophily allows to measure if similar vertices according to the attributes tend to be more connected than dissimilar vertices (cf. **P5**);

- *Within inertia* : Measure of the dispersion of the attribute values inside the communities (cf. **P4**). A low within inertia indicates that the communities are highly homogeneous with regard to the attribute values;

3.2 Structural Measures

- *Modularity* : gives the partition modularity measure as defined by M.E.J. Newman [2] (cf. **P3**);
- *Average clustering coefficient* : is given as an indication of the transitivity of connections in the network [3];
- *Random clustering coefficient* : gives the clustering coefficient in a Erdös-Renyi random graph having the same number of vertices and edges;

  The network average clustering coefficient is a measure of the clustering tendency of the network (cf. **P3**). This observed value can be compared with the expected value computed on a random graph having the same vertex set: an observed value higher than the expected value confirms the community structure;
- *Average degree* : the average number of neighbours of the vertices (cf. **P1**);
- *Average shortest path length* : the average minimum number of hops required to reach two arbitrary vertices (cf. **P2**). It is not computed when the graph is formed by several disconnected components (i.e., $E_{btw}^{max} = 0$);
- *Diameter* : length of the longest shortest path between any pair of vertices (cf. **P2**);
- *Nb. edges between* : number of edges connecting two vertices belonging to different communities;
- *Nb. edges within* : number of edges connecting two vertices belonging to the same community (cf. **P3**);
- *Nb. edges* : total number of edges in the graph, i.e., $\mathcal{E}$.

3.3 Degree Distribution

The bottom of the user interface includes also a panel displaying the distribution of vertex degrees on each graph of the sequence as shown in Figure 4.

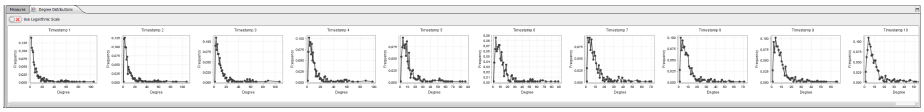


Figure 4: Degree distribution panel.

4 Graph Visualization and Manipulation

The central part of the user interface is shown in Figure 5. This panel allows to display the generated network and the changes in its communities at each time step. Each graph in the sequence can be viewed separately in the **Graph** tab. The sequence of graphs can also be visualized through the timestamp scrollbar at the right side of the panel.

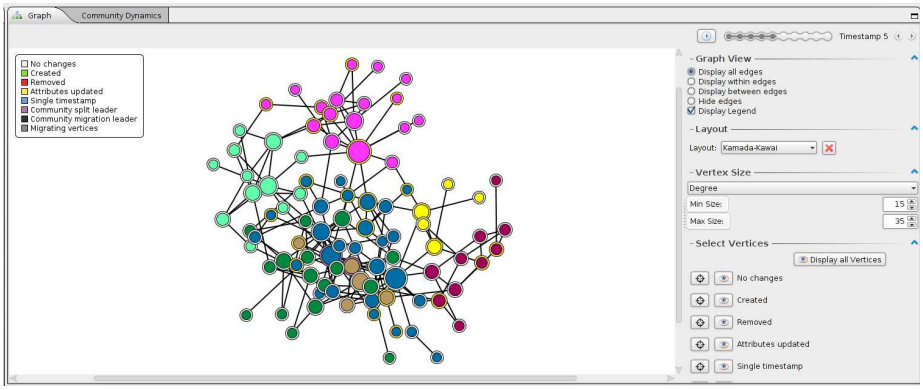


Figure 5: Graph visualization panel.

For each graph plotted, in the *Graph View* tab, we can set different options (see Figure 6) allowing for example to hide or display the edges and vertices through the *Graph View* section at the right side panel. The graph can then be displayed with different layout options (*kamada-kawai*, *fruchtman-reynolds* or *self organizing map*) where the sizes of the plotted vertices are chosen according to their degree, age or community membership. Moreover we can select or filter the displayed vertices according to their different events, as described in the micro dynamic operations, from the *Select Vertices* panel.

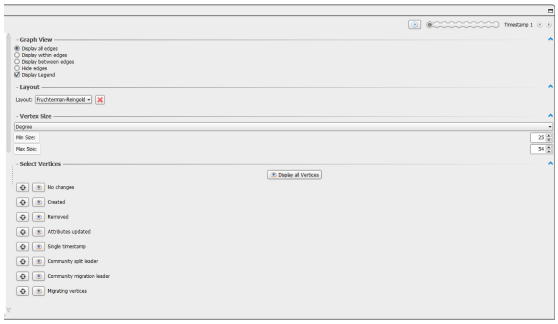


Figure 6: Graph options panel.

In the plotted graph, vertices of the same color are member of the same community, so each unique community have a unique color in the sequence of graphs. The user can then interactively select or manipulate a vertex (respectively a group of vertices) using the cursor. The informations for each node (id,

degree, attributes) are displayed when a vertex is pointed.

The community dynamics (Figure 7) are availlable through the *Community Dynamics* tabs, in the central part of the user interface. It displays the size and the evolution of the different communities in the sequence of graphs according to the macro dynamic operations (split, merge and migrate).

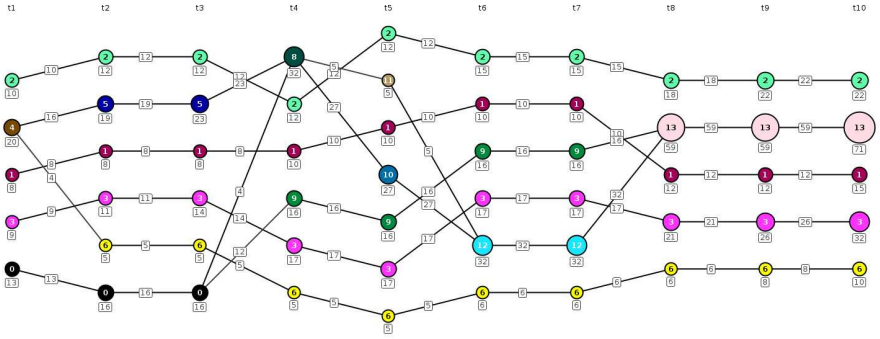


Figure 7: Community dynamics panel.

5 Output File Format

The generated dynamic network can be saved as a collection of files, one for each timestamp, under the **out** directory located in the same working directory as the generator. For each graph of the sequence, the file with the extension **.graph** indicates the composition of the graph (vertices and edges) and the **parameters** file enumerates all the parameters used by the generator.

- **Parameters** : The parameters are output in a separated file. Each line starts by the parameter name and its value.

  Example : $nbVertices$: 500.

  A *Seed* value is also saved in this file to be able to reproduce exactly the same dynamic notwork. Example *seed* : 28862418610579

- **Vertices** : In the graph file, the vertices section starts with the line *# Vertices*. Each consecutive line describes a vertex. A line consists of an integer corresponding to the vertex *id*, the list of its attribute values separated by a ; and an integer corresponding to the vertex community *id*. Each value is separated by a ;.

  Example : the line $1; 0.31; 0.49; 2$ corresponds to the vertex with $id = 1$ associated to two attributes having values 0.31 and 0.49 and being in community 2.

- **Edges** : This section starts with the line *# Edges*. Each consecutive line corresponds to an edge. A line is composed of two vertex ids separated by a ;.

  Example : the line $1; 4$ corresponds to an edge between the vertex with $id = 1$ and the vertex with $id = 4$.

The graph measures and community dynamics can also be saved in separated files.

- **Measures** : the measures are saved in a separated file. Each line gives the measure name and its consecutive values at each time step. Example : *modularities* : $0.58; 0.58; 0.59; 0.56; 0.55; 0.53; 0.53; 0.51; 0.49; 0.48$
- The evolution of communities partition are saved in a pdf file named **communityEvolution.pdf**.

6 Example

As an example, we consider a graph which will be generated with the following parameters:

- 1000 vertices in the first graph of the sequence ($n = 1000$), with 10 graphs in the sequence ($Timestamps = 10$)
- The number of communities in the first graph is set to 5 communities ($K = 5$) with 10 representents in each community ($Nb.\ Rep. = 10$).
- No vertices will be assigned to a random community ($Theta = 0$).
- Two real attributes are associated to each vertex ($Nb.\ Attr. = 2$), the standard deviation of the 1^{st} attribute is equal to 1 ($1\ Dev. 1 = 1$) and 7 for the 2^{nd} attribute ($Dev.\ 2 = 7$).
- At most 6 within community edges and 3 between community edges can be added to a newly inserted vertex ($Edges\ Within = 6$ and $Edges\ Between = 3$) and no additional edges will be inserted in the first graph of the sequence ($MTE = 0$) after the generation process.
- A micro update operation will be performed during the evolution of the network with a probability equal to 1/2 ($Proba\ Micro = 0.5$)
  - The ratio of newly created or removed vertices at each timestamp is set to 20% ($Add\ Vertex = Remove\ Vertex = 0.2$)
  - The ratio of vertices having their attribute values updated is set to 10% ($Update\ Attr. = 0.1$);
  - The ratio of edges inserted connecting two vertices in the same communities or in different communities is about 30% ($Add\ Wth.\ Edges = Add\ Btw.\ Edges = 0.3$)
  - The ratio of edges removed connecting two vertices in the same communities or in different communities is set to 30% ($Remove\ Wth.\ Edges = Remove\ Btw.\ Edges = 0.3$)
- All the macro dynamic operations are performed with a probability equals to 0.2 ($Proba\ Merge = Proba\ Split = Proba\ Migrate = 0.2$)

The generated dynamic network and its measures are shown in the Figure 8, Figure 9 and Figure 10.

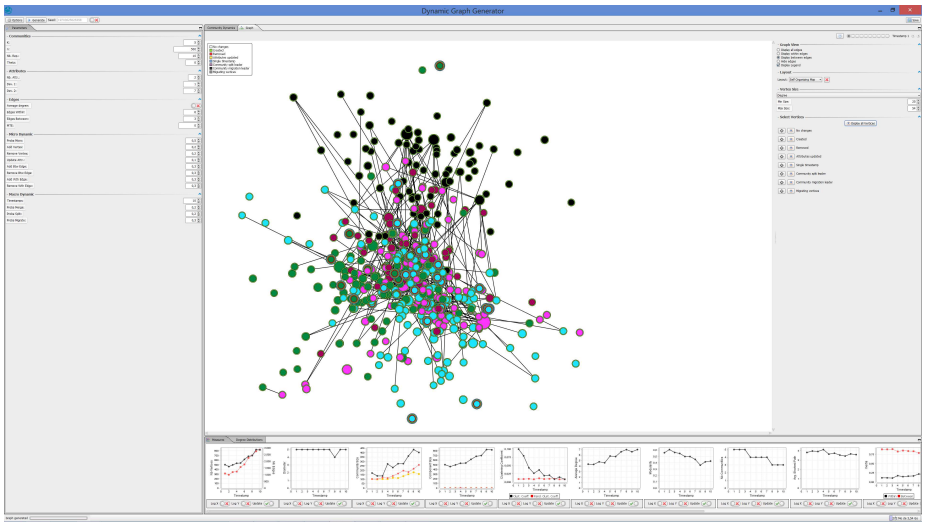


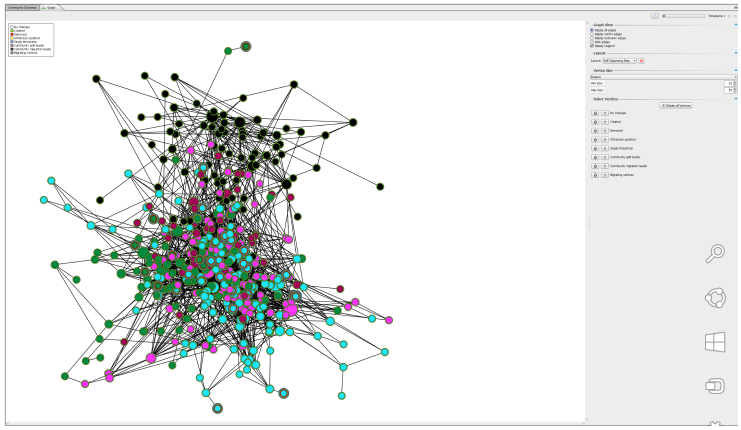
Figure 8: Example of a dynamic attributed network generated with **DANCer**.

References

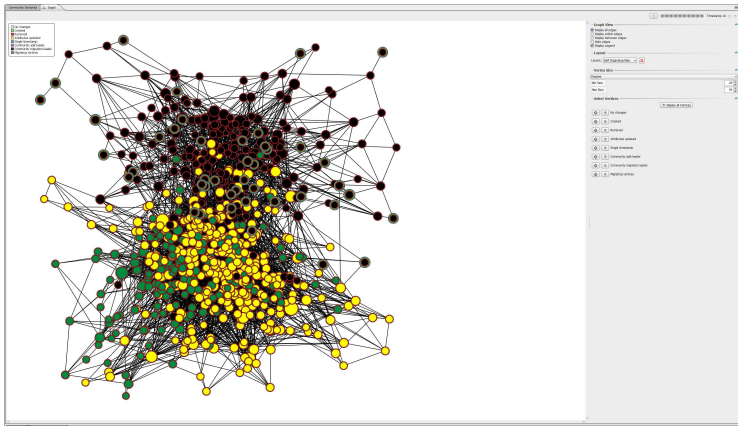
[1] Christine Largeron, Pierre-Nicolas Mougel, Reihaneh Rabbany, and Osmar R. Zaïane. Generating Attributed Networks with Communities. *PLoS ONE*, 10(4):e0122777, 04 2015.

[2] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

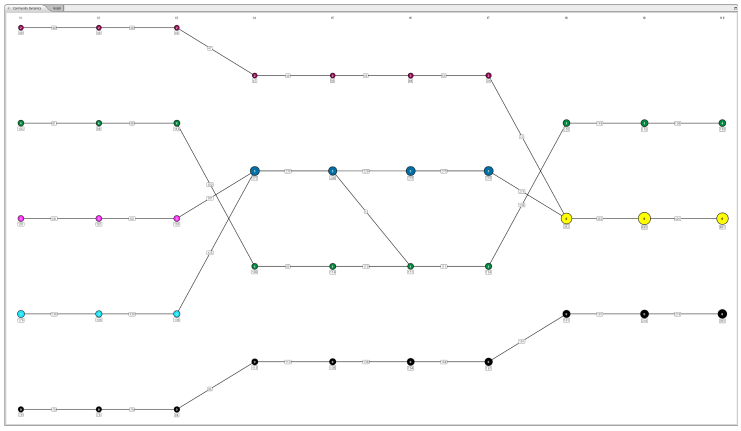
[3] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.



(a) First snapshot of the dynamic attributed network generated with **DANCer**.

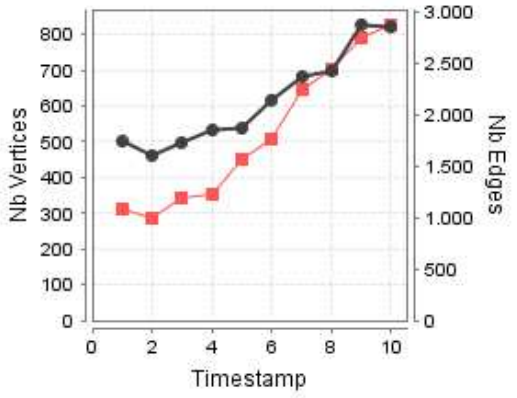


(b) Last snapshot of the dynamic attributed network generated with **DANCer**.

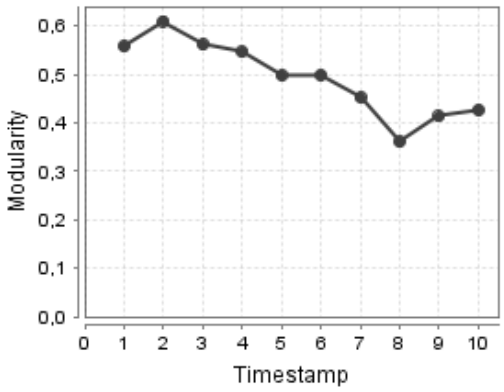


(c) Community dynamics.

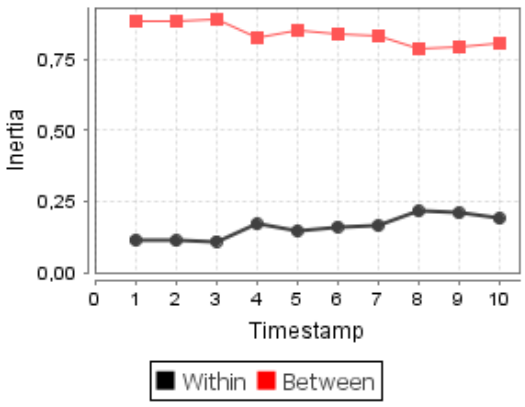
Figure 9: Example: properties of the dynamic attributed network.



(a) Number of vertices and edges.



(b) Modularity



(c) Inertia.

Figure 10: Example: measures on the dynamic attributed network.