Analysis and Visualization of dynamic clusterings

Pascal Held, Rudolf Kruse
Dept of Computer Sciences, Otto von Guericke University Magdeburg
email: {pheld, kruse}@iws.cs.uni-magdeburg.de

Abstract

Clusterings in dynamic networks are also dynamic. This means that they change over time. In this paper we present a visualization to show these changing behavior. For this purpose we used and modified the MONIC framework to track the clusters over time. Possible transactions during the lifetime of a cluster are birth, death, growth, contraction, splitting, and merging. We extend this list by rebirth, for clusters which were reactivated after death. In our evolution diagram we used lines to represent the lifetime of a cluster. Splitting and merging clusters have connecting lines. For the line adjustment we used a bandwidth reduction on connected components.

1. Introduction

Cluster analysis is a common method to analyze huge data sets. Often, these data set have a time component, so there is a dynamic behavior in the data. This results also in a dynamic clustering. So the clusters could change over time. These changes could be smaller changes like adding and removing elements in the cluster, or greater changes like splitting a cluster. There are different frameworks and algorithms to detect these changes. Recently, research changed in the direction on analyzing and understanding this changing behavior.

Nowadays, the most visualizations are focused on presenting the structure of a cluster, i.e. hierarchies of elements, or connections between clusters, like visualizations for graph clusterings. With our work we want to support the analysis of the dynamic clusters and their dynamic behavior. Our focus is more on the outer changes of the clusters over time, e.g. in which time frame a cluster exists, if it splitted up into several smaller cluster, or a result of a merge process.

We represent the clusters as lines during their lifetime. Different cluster sizes are represented by lines with different width. Clusters, that were split from others or merged into other clusters are connected with the interacting cluster by vertical lines. This allows the user to explore the dynamic data set. It is possible to see in which time step how many clusters exists, how long they exists, and how large they are. Also transactions between different clusters are visible.

This gives the possibility to get an overview of the represented data. It is also possible to find interesting points in time, e.g. when a lot of clusters are created or splitted, or if there are some dominant clusters which absorb a lot of the others. This could be easily recognized.

At first we give a brief overview of different graph clustering methods and frameworks for the tracking and monitoring of the changing behavior in Section 2. Afterward, in Section 3 we present our visualization in detail. We also present a modification on a tracking framework to handle clusters, which were reborn after they died. Following that, we tested our visualization on a dynamic clustering of the Enron mail corpus in Section 4. In this section we also propose some filters if there are too many clusters to visualize. We conclude this paper with results and critical review of our method in Section 5 and give a brief outlook in Section 6.

2. Related Work

2.1. Graph Clustering


The next step was the analysis of timestamped snapshots of dynamic graphs. Real world data sets were analyzed at different points in time. Since 2003, Kumar et al. studied the behavior of weblogs (blogs) and social networks like flickr [7, 8, 9, 10]. They analyzed the structure of the communities with different statistical measurements. In 2004, Hopcroft et al. [11] analyzed several snapshots of the NEC CiteSeer Database [12]. They used a hierarchical clustering to extract clus-
ters for the different snapshots. They matched the clusters from one time step to the clusters of the following time step. In this way they could follow the history of the communities over different time steps.

![Clusters and their transitions](image)

Figure 1. Main transactions of clusters over following time steps [13]

In the year 2007 Palla et al. [13] performed a systematical analysis of dynamic communities. They studied a graph of phone calls and a collaboration network between scientists. In Figure 1 the main recognized transactions of clusters between time steps are visualized. These are growth, contraction, merging, splitting, birth, and death. They matched clusters \( C(t) \) of following time steps \( t \) with clusters with a maximal relative overlap \( \text{C}(t) \), defined as follows:

\[
\text{C}(t) = \frac{|C(t)| \cap C(t+1)|}{|C(t)| \cup C(t+1)|} (1)
\]

2.2. Cluster Change Detection

For time-depending data a common approach is to analyze the data at given points in time. After this time stamp analysis clusters in different time stamps have to be matched against each other. In this section we present some of these algorithms.

Ganti et al. proposed the FOCUS framework [14] in 1999. This framework compares two data sets and computes a deviation measure between them. This comparison supports the user to understand how clusters are changing over time.

In 2001 Ganti et al. proposed the DEMON framework [15] for data evolution and monitoring across the temporal dimension. This framework focuses on detecting systematic and non-systematic changes in data. It also supports the user in extracting new patterns.

The PANDA framework [16], proposed by Bartolini et al. in 2004, provides methods to compare simple and also more complex patterns. These patterns could be based on a single cluster or the complete clustering.

The Pattern Monitor (PAM) [17], proposed by Baron et al., models patterns as temporal, evolving objects. The main focus of PAM is on association rules.

In 2006, Spiliopoulou et al. proposed a framework MONIC [18] for modeling and monitoring cluster transactions. This framework tracks and analyzes the main transactions supposed by Palla et al. [13], see Figure 1.

**Algorithm 1 MONIC Algorithm (directly taken from [18])**

```plaintext
1: for \( X \in \zeta \) do
2:    splitCandidates = splitUnion = \emptyset
3:    survivalCandidate = NULL
4:    for \( Y \in \zeta \) do
5:      Mcell = overlap(X,Y)
6:      if Mcell \( \geq \tau \) then
7:        if g(X,Y) \( \geq g(X,\text{survivalCandidate}) \) then
8:          survivalCandidate = X
9:        end if
10:      end if
11:    end for
12:    splitCandidates += Y
13:    splitUnion = splitUnion \cup Y
14:  end for
15:  if survivalCandidate == NULL \( \lor \) splitCandidate == \emptyset then
16:    deadList += X // \( X \rightarrow \emptyset \)
17:  else if splitCandidates \( \neq \emptyset \) then
18:    if overlap(X,splitUnion) \( \geq \tau \) then
19:      for \( Y \in \text{splitCandidates} \) do
20:        splitList += (X,Y)
21:      end for
22:      \( X \rightarrow \text{splitCandidates} \)
23:    else
24:      deadList += X // \( X \rightarrow \emptyset \)
25:    end if
26:  else
27:    AbsorptionSurvivals += (X,survivalCandidate)
28:  end if
29: end for
30: for \( Y \in \zeta \) do
31:    if cardinality(absorptionCandidates) \( > 1 \) then
32:      for \( X \in \text{absorptionCandidates} \) do
33:        absorptionList += (X,Y) // \( X \rightarrow Y \)
34:        absorptionSurvivals += (X,Y)
35:      end for
36:    else if absorptionCandidates == X then
37:      survivalList += (X,Y) // \( X \rightarrow Y \)
38:    end if
39: end for
```

The main idea of this algorithm is to compare two following clusters. They define a minimum overlap \( \tau \). If a cluster in the second timestamp overlaps at least to this value, the clusters are marked as matching clusters. Additionally, they define a minimum split overlap \( \tau_{split} \). If a cluster overlaps at least to this value, it is marked as a split-candidate. If the union of all split-candidates of a cluster of the first time stamp overlaps at least \( \tau \) the cluster is marked as splitted into these split-candidates. Are there more matching clusters from the first time stamp...
to one cluster in the second time stamp, these clusters are marked as merging clusters. Clusters which have no (partial) matching clusters in the second time stamp are marked as death clusters. Analogous, clusters from the second time stamp with no matching clusters in the first one are marked as new clusters. More details on how this work, can be seen in Algorithm 1 or in the original paper [18].

3. Preparing Data and Visualization

In our visualization we want to show the behavior of clusters over time. Every transaction proposed by Palla et al. [13] should be visualized for clusters over time. To monitor the clusters evolution we use a modified version of the MONIC algorithm [18].

During the evolution of a cluster it is possible that a cluster dies in one time step, but a few time steps later a new cluster is born, which is similar to the dead one. In our opinion this relation between these clusters supports the user to recognize that a cluster not only exists for a special time frame. For this purpose we modified the MONIC algorithm. Clusters which would be created in the current time step, are checked for matching an already dead cluster. If there are more possible clusters the one which died most recently is used.

3.1. Visualization Elements

We visualize every cluster as a horizontal line in our evolution diagram. The horizontal axis represents the time. In the vertical direction all clusters in a time step are stacked one over the other. Each possible transaction is encoded as follows.

3.1.1. Birth and Death Each cluster either emerges from scratch or splits from another cluster. If a cluster is born there a simple line will start at the point of birth. The end of life is visualized analogous. An example for both is shown in Figure 2.

![Figure 2. Birth and Death of simple cluster](image)

3.1.2. Splitting In Figure 3 we show a cluster (the gray one) which splits up into two clusters. The new cluster (the black one) is connected with the old one by a vertical line at the beginning of the new cluster.

![Figure 3. One cluster (gray) splits up into two clusters (black and gray)](image)

3.1.3. Merging A cluster can be absorbed by another cluster. In this case the end of the cluster is connected with the absorbing cluster by a vertical line. An example is shown in Figure 4.

![Figure 4. Two clusters (black and gray) merge into one cluster (gray)](image)

3.1.4. Growth During the evolution a cluster can grow, which means that the size of the cluster is increased. This would be visualized by increasing the line width, see Figure 5.

![Figure 5. Birth and Death of simple cluster](image)

3.1.5. Contraction If the cluster loses elements, the size of the cluster is decreasing. This is represented contrarily to the growth as decreasing of the line width. There is one example shown in Figure 6.

3.1.6. Rebirth If a dead cluster is reborn after a certain time, the life time frames are connected with a thin line. This shows that the cluster was dead during a specific time frame. An example is shown in Figure 7.

3.2. Cluster Ordering inside Connected Components

As written before every cluster is represented by a line in our evolution diagram. These lines could be connected in case of splitting or merging. The main focus in the visualization is due to the horizontal lines, which represent the lifetime of a cluster. The vertical lines are crossing all the horizontal lines between the concerning clusters. To avoid long connecting lines, which have no information besides showing clusters are connected, these lines should be as short as possible.
Therefor, we divided the whole problem in a) order the connected components, and b) place them. To find the connected components we consider the clusters as a graph, where every node is one cluster. Two nodes \( A \) and \( B \) are connected, if and only if a) \( A \) is split up from \( B \), b) \( B \) is split up from \( A \), c) \( A \) is merged into \( B \), or d) \( B \) is merged into \( A \). On this graph we compute a connected components analysis with a breadth-first-search. For each component we created an adjacency matrix. On this matrix we computed a bandwidth reduction with the Cuthill-McKee-algorithm \[19\]. The permutation of the rows and columns gives us the order of the containing clusters.

### 3.3. Packing of Connected Cluster Components

The next step is packing all connected components \( P \) on the evolution diagram. For simplification we consider every connected component as a rectangle. Every rectangle has a starting time \( s \), defined as the minimal starting time of all containing clusters \( c \), and an end time \( e \), defined as the maximal end time.

\[
s = \min_{c \in P} (\text{start}(c)) \\
e = \max_{c \in P} (\text{end}(c))
\]

The height is depending the space needed to stack all containing clusters. All rectangles shall be placed without overlapping.

We ordered all connected components first by their height and second by there starting time. For every rectangle in this ordered list we check the minimal vertical offset to place the rectangle without overlapping other placed rectangles. This heuristic works fine, because at first all big elements are placed. Second the place between them is used by the smaller components with less elements.

### 4. Experiments

The Enron mail corpus is a collection of email boxes from 150 employees of the Enron company. It contains the mail communication of these employees in a timeframe of about one year \[20\]. Like in every group of people there are subgroups (clusters) of people which are communicating together more often then with other employees. In previous work \[21\] we analyzed these data with respect to the communication frequency between the Enron employees. For the clustering we used a clustering threshold of 5, 10 and 20. The larger this threshold, the more details are represented in the clusters. The result of this analysis is a collection of clusterings, one per time step. To get a better overview of the development of these clusters we used the introduced visualization.

#### 4.1. Determine Parameters for MONIC

For monitoring and tracking of the clusters we have to provide a minimal overlap for matching clusters \( \tau \) and a minimal overlap for splitting clusters \( \tau_{\text{split}} \) to the MONIC algorithm. As proposed in \[18\] we first select \( \tau \). So we run the tracking with a fixed \( \tau_{\text{split}} = 0.2 \) and set \( \tau \in [0.50, 0.55, \ldots, 0.95] \).

The obvious observation is that with increasing \( \tau \) there will be less clusters which could be matched in the next time step. We observed that the number of dead and reborn clusters increases with increasing \( \tau \). Also the sum of all created clusters (birth and split) increases. Besides more clusters with no connections to any other clusters exist with an increasing \( \tau \).

Figure 8. Numbers of clusters that were created by birth for \( \tau \in [0.50, 0.55, \ldots, 0.95] \)

The more interesting fact is, that the minimal number of created clusters (birth) and the maximal number of clusters that were split are both between \( \tau = 0.7 \) and \( \tau = 0.8 \). Compare Figures 8 and 9. For this reason we chose \( \tau = 0.75 \) for further studies.
As the second step we studied the influence of \( \tau_{\text{split}} \). So we set \( \tau_{\text{split}} = [0.10, 0.15, \ldots, 0.50] \). In Figure 10 we show the influence of \( \tau_{\text{split}} \) on the different cluster transactions. This diagram is performed on a cluster threshold of 5. The threshold 10 and 20 result in similar measures. The value \( \tau_{\text{split}} = 0.25 \) produces the smallest number of clusters. Increasing \( \tau_{\text{split}} \) yields an increasing number of births, deaths, and rebirths and a decreasing number of split and merge transactions. We selected \( \tau_{\text{split}} = 0.25 \) as parameter for our experiments.

### 4.2. Preprocessing Clustering Results

We perform the cluster tracking with \( \tau = 0.75 \) and \( \tau_{\text{split}} = 0.25 \) for the clusterings with the threshold \( t_c = 5 \), \( t_c = 10 \), and \( t_c = 20 \). The numbers of the observed transactions is shown in Table 1.

<table>
<thead>
<tr>
<th>Transaction</th>
<th>( t_c = 5 )</th>
<th>( t_c = 10 )</th>
<th>( t_c = 20 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birth</td>
<td>3905</td>
<td>8538</td>
<td>12143</td>
</tr>
<tr>
<td>Death</td>
<td>2423</td>
<td>5887</td>
<td>9201</td>
</tr>
<tr>
<td>Rebirth</td>
<td>729</td>
<td>1843</td>
<td>2853</td>
</tr>
<tr>
<td>Merging</td>
<td>3322</td>
<td>5775</td>
<td>7189</td>
</tr>
<tr>
<td>Splitting</td>
<td>1740</td>
<td>3077</td>
<td>4210</td>
</tr>
<tr>
<td>Matching</td>
<td>67104</td>
<td>65085</td>
<td>63129</td>
</tr>
</tbody>
</table>

Table 1. Number of observed transactions

In Figure 11 all clusters from the clustering with \( t_c = 5 \) are visualized. In the data set there are a lot of clusters, which contain only a few number of elements, or exists only for a short time. For example, in one time step some elements of a huge cluster were separated into a new one and in the next time step this cluster will merge back to the original clusters. This produces a very small horizontal line, but this line has to be connected twice with the original cluster. This yields into a lot of (long) vertical lines, which produces nearly complete black areas. Thereby, the diagram gets really unclear.

To reduce this suggest to filter out some of the clusters. Good candidates are clusters which exists only a short time, e.g. one or two time steps, or clusters that contain only few elements. In Tables 2, 3, and 4 we present the number of clusters by a given life times and cluster sizes.

If we just filter out all clusters which exist only in one time step, we get a much clearer view on the evolution of the clusters (cf. Figure 11 and Figure 12). The black areas become much smaller. In Figure 13 is the in
<table>
<thead>
<tr>
<th>Items</th>
<th>Life time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>&gt;3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>823</td>
<td>343</td>
<td>216</td>
<td>823</td>
<td>2205</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>197</td>
<td>104</td>
<td>71</td>
<td>509</td>
<td>881</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>61</td>
<td>48</td>
<td>33</td>
<td>319</td>
<td>461</td>
</tr>
<tr>
<td></td>
<td>&gt;3</td>
<td>78</td>
<td>169</td>
<td>102</td>
<td>1779</td>
<td>2128</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>1159</td>
<td>664</td>
<td>422</td>
<td>3430</td>
<td>5675</td>
</tr>
</tbody>
</table>

Table 2. Number of clusters with given life time and maximal cluster size ($t_c = 5$)

<table>
<thead>
<tr>
<th>Items</th>
<th>Life time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>&gt;3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1872</td>
<td>722</td>
<td>371</td>
<td>1637</td>
<td>4602</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>430</td>
<td>244</td>
<td>154</td>
<td>1229</td>
<td>2057</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>152</td>
<td>121</td>
<td>81</td>
<td>855</td>
<td>1209</td>
</tr>
<tr>
<td></td>
<td>&gt;3</td>
<td>147</td>
<td>168</td>
<td>125</td>
<td>3251</td>
<td>3691</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>2601</td>
<td>1255</td>
<td>731</td>
<td>6972</td>
<td>11559</td>
</tr>
</tbody>
</table>

Table 3. Number of clusters with given life time and maximal cluster size ($t_c = 10$)

<table>
<thead>
<tr>
<th>Items</th>
<th>Life time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>&gt;3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2753</td>
<td>997</td>
<td>529</td>
<td>2391</td>
<td>6670</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>581</td>
<td>325</td>
<td>248</td>
<td>2072</td>
<td>3226</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>191</td>
<td>129</td>
<td>92</td>
<td>1533</td>
<td>1945</td>
</tr>
<tr>
<td></td>
<td>&gt;3</td>
<td>153</td>
<td>183</td>
<td>133</td>
<td>4151</td>
<td>4622</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>3678</td>
<td>1634</td>
<td>1002</td>
<td>10149</td>
<td>16463</td>
</tr>
</tbody>
</table>

Table 4. Number of clusters with given life time and maximal cluster size ($t_c = 5$)

Figure 12 marked detail presented. In the with 1 marked area the reason for the black areas is shown. There are too much connecting lines. Number 2 is an example for the rebirth of an clustter. At number 3 two effects are visible, the growing and contraction (thicker and thinner lines) and the split into two clusters (parallel line below the first line). Filtering out also clusters which exists only two and three time steps results in the visualization from Figure 14. There are no black areas and annoying crossing lines to see.

Figure 12. Visualization clusters, with lifetime > one time step, from the clustering with $t_c = 5$

Figure 13. Detail of Figure 12

Figure 14. Visualization clusters, with lifetime > three time steps, from the clustering with $t_c = 5$

Removing small clusters does not lead to a clearer visualization. In Figure 15 we filtered all clusters with a maximal size of three elements. We have sorted out
about 40% of all clusters, but there are still many clusters with short life times. If they are connected with others they produce a lot of vertical lines close to each other. So we propose to remove short time clusters, which only exists a few time steps.

Figure 15. Visualization clusters, with maximal cluster size \(>\) three elements, from the clustering with \(t_c = 5\)

In Figure 16 you see another phenomenon for clusters which could be removed from the diagram. Especially in the bottom part of the diagram there are a lot of clusters which exists for a short time, died, and are reborn again for a short time. These clusters are dead most of the time. In Figure 17 we also filtered the clusters with more then 95% death time. This eliminates a lot of clusters which only exist in the beginning and the end of the data set.

Figure 16. Visualization clusters, with lifetime \(>\) three time steps, from the clustering with \(t_c = 5\)

Figure 17. Visualization clusters, with lifetime \(>\) three time steps and maximal 95% death time, from the clustering with \(t_c = 5\)

5. Conclusion and Results

In this work we presented a method to visualize the evolution of dynamic clusters. The main graph transactions birth, death, splitting, merging, growth, and contraction [13] were considered in the visualization. We extended these transactions with rebirth, which means that a dead cluster is reactivated after some time. For the tracking of the generated clusters we used a modified version of the MONIC algorithm [18].

Every cluster is represented by a line during its lifetime. Clusters which are split off of or merged into another clusters were marked by a connecting line. This new visualization supports the analyst in finding interesting clusters. This clusters could be clusters which are very stable, so they exist in a long time frame. Also dominant clusters which absorb a lot of the others ones or clusters which have an high dynamic could be interesting. If the clusters are labeled in the visualization, it is pretty easy to find clusters for further analysis.

Another possibility is the analysis of events. There are some events, that changes the structure of a given communication network, like starting of a new semester in university, where a lot of new relationships occur or reorganizations in companies. With this visualization it is possible to see the impact of such events or, if you see a point in time where a lot of changes are happen, to find such events.

In Figure 18 one of these visualizations is shown. For this diagram we used the dynamic clustered Enron mail corpus [21] with a clustering threshold of \(t_c = 20\).

To optimize the adjustment of the lines in the diagram we used a bandpass reduction on the adjacency matrices of connected components.

If there are to many elements to visualize we propose to remove clusters which only exist for a very short time, like one time step. With this filtering we get a clear visualization of the dynamic clusters over time.

6. Outlook

The biggest handicap on this visualization is that the containing elements of a cluster have no influence on the position of the cluster in the diagram. This means, that if a cluster continuous exchange elements, the size of the cluster is constant over time. It is also possible to match the cluster in each time step, but the beginning of the cluster has nothing to do with the end of the cluster. Imagine the following cluster over time:

1. 1,2,3,4,5
2. 6,2,3,4,5
3. 6,7,3,4,5
4. 6,7,8,4,5
5. 6.7.8.9.5
6. 6.7.8.9.10
In each time step the cluster has an overlap of 80% with the previous one but the start and the end of this cluster has an overlap of 0.

To include this aspect, the line of the cluster could move in vertical direction, so we do not have a strict horizontal line, representing a cluster.

Another aspect is that if we have two clusters A and B, A can transform into B and at the same time B could transform in anything else, it is not visualized that cluster A now looks like cluster B at the beginning.

7. References


