DenGraph-HO: Density-based Hierarchical Community Detection for Explorative Visual Network Analysis

Nico Schlitter, Tanja Falkowski and Jörg Lässig

Abstract For the analysis of communities in social networks several data mining techniques have been developed such as the DenGraph algorithm to study the dynamics of groups in graph structures. The here proposed DenGraph-HO algorithm is an extension of the density-based graph clusterer DenGraph. It produces a cluster hierarchy that can be used to implement a zooming operation for visual social network analysis. The clusterings in the hierarchy fulfill the DenGraph-O paradigms and can be efficiently computed. We apply DenGraph-HO on a data set obtained from the music platform Last.fm and demonstrate its usability.

1 Introduction

DenGraph-HO was developed in order to fulfill the special needs of social network analysts. In most cases, the visual inspection of a network is the first step of the analytical process and helps to determine the basic graph characteristics and further actions. DenGraph-HO supports this early stage by providing a quick visual analysis of the network structure. It provides the ability of zooming into network clusterings and has proven its usefulness for our practical work.

The zooming feature is based on a cluster hierarchy that is computed by applying DenGraph-HO. Our approach differs from traditional hierarchical clustering methods in that DenGraph is a non partioning cluster algorithm. We consider the fact

Nico Schlitter
University of Applied Sciences Zittau/Görlitz, Group for Enterprise Application Development, e-mail: NSchlitter@hs-zigr.de

Tanja Falkowski
University of Göttingen, Göttingen International, e-mail: Tanja.Falkowski@zvw.uni-goettingen.de

Jörg Lässig
University of Applied Sciences Zittau/Görlitz, Group for Enterprise Application Development, e-mail: JLaessig@hs-zigr.de

that not all nodes are necessarily member of clusters. In addition, the proposed hierarchy is not strictly build up by the classic divisive or agglomerative approach that is known from literature. We generalize these methods and propose a top-down and bottom-up approach by extending the hierarchy paradigms. The proposed hierarchy supports superordinate clusters that contain subclusters and nodes which are not assigned to clusters due to their distance.

Each level of the hierarchy represents a clustering that fulfills the DenGraph paradigms which are described below. The levels, respectively the clusterings, differ in the density that is required to form a cluster. While lower level clusterings aggregate nodes with a lower similarity, higher level clusterings require a higher similarity between nodes. The density-based cluster criteria are controlled by the parameters η and ε which are iteratively applied for each level of the hierarchy. Thereby, an existing clustering is used to compute the clustering of the next level. The efficiency of our algorithm is based on this iterative sequence of cluster adaption instead of recalculating each cluster.

The remainder of this paper is organized as follows. Section 2 introduces the original DenGraph algorithm and its variations DenGraph-O and DenGraph-IO. Section 3 covers the proposed approaches of the DenGraph-HO algorithm. Its usability is demonstrated in Section 4 by applying DenGraph-HO on a dataset obtained from the online music platform Last.fm. Finally, a conclusion and an outlook are given in Section 5.

2 Related Work

DenGraph [5] is a density-based graph clustering algorithm developed in 2007 for community detection in social networks. It is inspired by DBSCAN [2], a well known clustering algorithm for spatial data, and applies a similar local cluster criterion. In the following, we briefly introduce the original DenGraph algorithm and its variations DenGraph-O and DenGraph-IO.

2.1 DenGraph

Given a graph $G = (V, E)$ consisting of a set of nodes V and a set of weighted, undirect edges E, the DenGraph algorithm produces a clustering $\zeta = \{C_1, \ldots, C_k\}$ whereas each cluster C_i $(i = 1 \ldots k)$ consists of nodes $V_{C_i} \subseteq V$. Since DenGraph is a non-partitioning clustering algorithm there can be *noise nodes* $V_N = \{u \in V \mid u \notin C_i\}$ that are not part of the clustering ζ. The remaining non-noise nodes are either *core nodes* or *border nodes* of a cluster. A node $u \in V$ is considered as core node if it has an ε-neighborhood $N_\varepsilon(u) = \{v \in V \mid \exists(u,v) \in E \wedge dist(u,v) \leq \varepsilon\}$ (where $dist(u,v)$ is the distance between u and v) of at least η neighbor nodes ($|N_\varepsilon(u)| \geq \eta$). Nodes which are in the ε-neighborhood of a core node but have not an own ε-neighborhood are called border nodes.

The actual cluster criterion is based on the concepts *directly density-reachable*, *density-reachable* and *density-connected* which are defined below according to [3] and illustrated in Fig. 1.

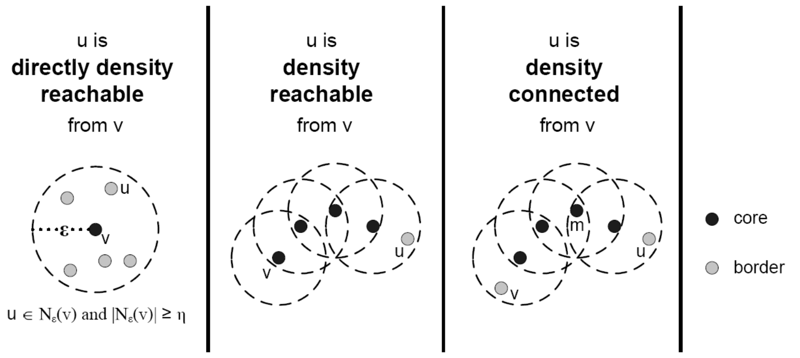


Fig. 1 The concepts *directly density reachability*, *density reachability* and *density connectedness* to determine whether nodes are density connected. (cf. [3])

Definition 1. Let $u, v \in V$ be two nodes. u is *directly density-reachable* from v within V with respect to ε and η if and only if v is a core node and u is in its ε-neighborhood, i.e. $u \in N_\varepsilon(v)$.

Definition 2. Let $u, v \in V$ be two nodes. u is *density-reachable* from v within V with respect to ε and η if there is a chain of nodes $p_1, \ldots, p_n$ such that $p_1 = v, p_n = u$ and for each $i = 2, \ldots, n$ it holds that p_i is directly density-reachable from p_{i-1} within V with respect to ε and η.

Definition 3. Let $u, v \in V$ be two nodes. u is *density-connected* to v within V with respect to ε and η if and only if there is a node $m \in V$ such that u is density-reachable from m and v is density-reachable from m.

In general, a set of core and border nodes V_C forms a cluster C if each node $u \in V_C$ is density-connected to each node $v \in V_C$.

The DenGraph algorithm itself is described in Alg. 1. It uses a stack in order to process the graph nodes. In a first step, all nodes V are marked as noise. Afterwards, each so far unprocessed node v is visited and checked if it has an ε-neighborhood. If the neighborhood contains at least η nodes ($|N(v)| \geq \eta$) v is marked as core and a new cluster is founded. Each of v's neighbors is marked as border, becomes a member of the new cluster and is pushed on the stack. After handling all neighbors, each node u from the stack is checked regarding having an ε-neighborhood and marked correspondingly. If u became core node, all of it's neighbors are marked as border and pushed on the stack. This procedure is repeated until all nodes of the graph are processed.

Algorithm 1: DenGraph (cf. [3])

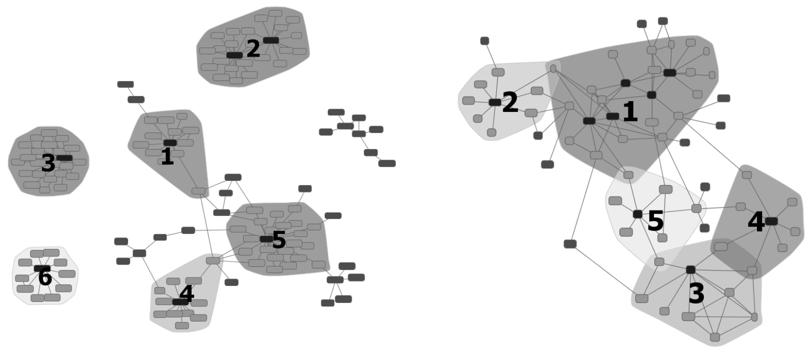
```
input  : Graph,$\eta$,$\varepsilon$
output: ClusterModel
begin
    foreach $r \in V$ do r.state=noise;
    foreach $(u \in V | u.state = noise)$ do
        if $(|N_\varepsilon(u)| \geq \eta)$ then
            Cluster=CreateNewCluster();
            Cluster.addNode(u);
            u.state=core;
            foreach $n \in N_\varepsilon(u)$ do
                Cluster.addNode(n);
                n.state=border;
                stack.push(n);
            repeat
                v=stack.pop();
                if $(|N_\varepsilon(v)| \geq \eta)$ then
                    v.state=core;
                    foreach $n \in N_\varepsilon(v) | n.state \neq core$ do
                        Cluster.addNode(n);
                        n.state=border;
                        stack.push(n);
            until stack is empty;
    return ClusterModel;
```

2.2 DenGraph-O

Practical work with DenGraph in the field of Social Network Analysis revealed a minor drawback: While in real world applications nodes - respectively human beings - might be part of more than one community, the DenGraph algorithm does not allow for clusters to overlap. This issue was addressed in [6] and the extended version DenGraph-O[1] allows border nodes that are part of more than one cluster.

Figure 2(a) shows an exemplary graph visualization of the Enron dataset, which encodes the communication frequency of Enron employees [8]. The graph was clustered by applying the original DenGraph. Core nodes are blue, border nodes are green and noise nodes are drawn in red color. An example for overlapping clusters is illustrated in Fig. 2(b).

[1] O stands for **O**verlapping



(a) Clustered with DenGraph (b) Clustered with DenGraph-O

Fig. 2 Visualization of the Enron graphs clustered with DenGraph and DenGraph-O (cf. [3])

2.3 DenGraph-IO

Falkowski et al. propose [4, 6] to analyze the dynamics of communities over time by comparing the changes between clusterings that are obtained in different time points. For this, it is necessary to compute the graph clusterings that are observed over time. A huge computational effort would be necessary if the original DenGraph was used to process multiple consecutive snapshots of social networks. However, as social structures often change slowly, the graphs G_t and G_{t+1} differ just slightly. Therefore, a total re-clustering, as the use of the original DenGraph would demand, would be quite inefficient. The incremental cluster algorithm DenGraph-IO[2] addresses this issue and updates an existing clustering based on the changes of the underlaying graph. Since the DenGraph-IO algorithm deals exclusively with the parts of the graph that changed from one point in time to the other, the computational complexity is dramatical reduced and even huge networks can be processed in reasonable time.

3 DenGraph-HO

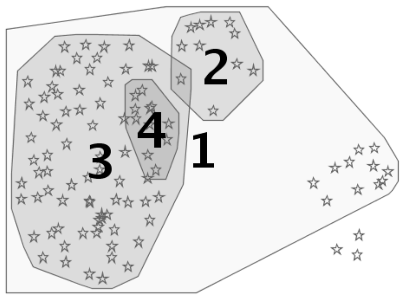
One challenge of the DenGraph algorithm is the choice of the parameters epsilon (ε) and eta (η). Several heuristics have been developed, however, the ”right” parameter combination mainly depends on the aim of the analysis. If the analyst is for example interested in observing strongly connected nodes rather than in clusterings that show the overall structure, the parameters need to be chosen accordingly. DenGraph-HO addresses this issue and allows for a quick visual representation of the clustering for

[2] IO stands for **I**ncremental and **O**verlapping

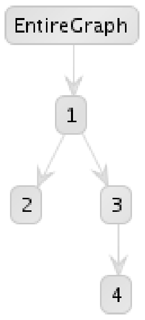
a chosen parameter combination. The process of zooming in or out of the network can be steered by the analyst.

The proposed algorithm returns a hierarchical clustering that describes the structure of the underlaying network. Thereby, the hierarchy provides multiple views of the network structure in different levels of detail. Consequently, the cluster hierarchy is an ideal basis for an efficient zooming implementation. Zooming-in is done by stepping up in the hierarchy. It provides a more detailed view of the current cluster by presenting its subclusters. A higher level of abstraction is reached by zooming-out, which is equivalent to merging similar clusters into superordinate clusters.

In principle, the hierarchy is a tree of clusters. Figure 3 shows an exemplary graph clustering and the related cluster hierarchy. Since the root of the tree represents the whole graph, children represent subclusters of its parent cluster. Following this definition, the leaves of the tree correspond to the smallest clusters.



(a) Hierarchical Clusterings



(b) Hierarchy

Fig. 3 Visualization of the Enron graph clustered with DenGraph-HO

Here, we are proposing a hierarchy that is based on the concepts of the DenGraph algorithm. Each level of the tree (besides the root) represents a valid clustering that fulfills the DenGraph-O paradigms. The hierarchy could be built by repeatedly applying DenGraph-O while using specific parameter settings for each level of the tree. Thereby, the choice of the parameters η and ε is limited by constraints in order to ensure that lower level clusters are subclusters of the parent cluster.

Let us assume that the clustering ζ_l forms level l of the hierarchy and is computed by applying DenGraph-O with the parameters ε_l and η_l. Level $l+1$ represents a clustering that is based on ε_{l+1} and η_{l+1} and guarantees a higher similarity of nodes in the cluster. According to the description above, ζ_{l+1} has to contain subclusters of clusters that are element of ζ_l. In order to preserve this parent-child relation we have to ensure the following constraints:

1. The parameter ε_l that is used to generate the clustering ζ_l has to be bigger or equal than ε_{l+1} which is used to compute the clustering ζ_{l+1}:

$$\varepsilon_l \geq \varepsilon_{l+1}$$

2. The parameter η_l that is used to generate the clustering ζ_l has to be lower or equal than η_{l+1} which is used to compute the clustering ζ_{l+1}:

$$\eta_l \leq \eta_{l+1}$$

Increasing ε might lead to a transition of a node state from noise or border to core or from noise to border. By increasing ε a core node can not loose its state. This explains why increasing ε might create a new or expand an existing cluster and why it surely avoids cluster reductions or removals. The same argument holds for decreasing η and shows why the demanded cluster-subcluster relation can be guaranteed by the given constraints.

In the following, we discuss how the proposed cluster hierarchy can be efficiently generated based on a list of parameter settings that fulfill the discussed constraints. An obvious approach would be to perform multiple re-clusterings until each parameter setting is processed. However, this is very inefficient and would be a huge computational effort because for a re-clustering the whole graph needs to be traversed again.

The proposed DenGraph-HO algorithm addresses this issue and uses incremental parameter changes to generate the cluster hierarchy. Instead of computing a complete new clustering for each level, an existing clustering is used and adapted. In the following, we discuss how an existing clustering of level l can be used to compute the clusterings of level $l+1$ and $l-1$. We propose a bottom-up and top-down approach and analyze their efficiency depending on the graph structure. The input for both approaches is a graph $G = (V, E)$ and an existing clustering $\zeta_l = \left\{C_1^l, \ldots, C_k^l\right\}$ that fulfills the DenGraph paradigms.

3.1 Top-down Approach: Cluster Reduction, Split or Removal

The top-down approach performs a zoom-in operation and generates a new clustering for level $l+1$ of the hierarchy. Where in level $l+1$, nodes are clustered which have a higher similarity than clusters in level l. Thereby, clusters of level l might be reduced, split or removed. By decreasing ε and increasing η the state of nodes within a cluster might change. A former border node might become noise (Cluster Reduction). Former core nodes might get border state (possible Cluster Splitting) or noise state (Cluster Reduction, possible Cluster Splitting or Removal).

Due to the DenGraph paradigms, it is guaranteed that noise nodes can not reach border or core state by decreasing ε or increasing η. Thus, noise nodes will not change their state and do not need to be processed.

Consequently, the top-down approach traverses just border and core nodes and performs a reclustering for each existing cluster. For this purpose, we use a modified DenGraph-O that is shown in Alg. 2. Each cluster C of level l is reclustered by applying the parameters of level $l+1$. Regarding the cluster hierarchy, if new clusters emerge, they are subclusters of C.

Algorithm 2: TopDown

```
input  : Graph,Clustering,$\varepsilon$,$\eta$
output: Clustering
foreach ($C \in \zeta$) do
    foreach $r \in C$ do r.state=noise;
    foreach ($u \in C | u.state = noise$) do
        if ($|N_\varepsilon(u)| \geq \eta$) then
            Cluster=CreateNewCluster();
            C.addCluster(Cluster);
            Cluster.addNode(u);
            u.state=core;
            foreach $n \in N_\varepsilon(u)$ do
                Cluster.addNode(n);
                n.state=border;
                stack.push(n);
        repeat
            v=stack.pop();
            if ($|N_\varepsilon(v)| \geq \eta$) then
                v.state=core;
                foreach $n \in N_\varepsilon(v) | n.state \neq core$ do
                    Cluster.addNode(n);
                    n.state=border;
                    stack.push(n);
        until stack is empty;
return Clustering;
```

3.2 Bottom-up Approach: Cluster Creation, Absorption and Merging

Figuratively, the bottom-up approach performs a zoom-out operation and generates a new clustering for level $l-1$. Thereby, new clusters may be created, existing clusters of ζ_l might absorb new members or get merged with other clusters.

As discussed above, by increasing ε and decreasing η, the state of a core node remains unchanged. A former noise node may become core node (cluster creation) or border node (absorption). A former border node could become core node (absorption). In case a former border node is member of multiple clusters (overlapping clusters), its transition to core state leads to a merge of those clusters.

Since core nodes keep their state, there is no need to consider them in the bottom-up approach. Consequently, the proposed procedure processes only noise and border nodes in order to determine their new state and to adapt the clustering accordingly. Following this argumentation, the procedure's efficiency is based on the saved time that the original DenGraph-O would have spent for processing core nodes.

Algorithm 3 describes the procedure that performs the bottom-up step by dealing with the changes of η and ε. First, the algorithm iterates over all existing clusters in order to expand them. Therefore it traverses all border nodes and updates their state based on the number of nodes in the ε-neighborhood and η. In case a former border node becomes core, this new core node is pushed on the stack for further processing. After dealing with all border nodes, the procedure *Cascade_Expand* checks if new core nodes absorb their neighbors into the cluster. In case an absorbed neighbor has no own ε-neighborhood of cardinality η it becomes border node, otherwise it becomes a core node. The newly discovered core nodes are pushed on the stack and the procedure is repeated until no further nodes are absorbed into the cluster. Since we allow for clusters to overlap, a new core node might have been a member of multiple clusters before. Due to its new core state, the affected clusters are merged into a superordinate cluster.

After dealing with all border nodes the existing clusters are maximal expanded with respect to the changed η and ε. Now, the remaining noise nodes are processed to check whether their state has changed. In case a former noise node becomes core, a new cluster is created and the procedure *Cascade_Create* absorbs the ε-neighbors according to the DenGraph-O paradigms. If these neighbors become core nodes, this cascade is repeated until no new ε-neighbors are found. During the handling of noise nodes a newly created cluster will not merge with an existing one. If the new cluster would be in ε-distance to an other cluster, the nodes of the new cluster would have been already absorbed into the existing cluster during handling the border nodes.

Algorithm 3: BottomUp

```
input  : Graph,Clustering,$\varepsilon$,$\eta$
output: Clustering
foreach ($C \in \zeta$) do
    Cluster=CreateNewCluster();
    Cluster.addSubCluster(C);
    foreach ($u \in C | u.state = border$) do
        if ($|N_\varepsilon(u)| \geq \eta$) then
            u.state=core;
            stack.push(u);
    Cascade_Expand(stack,Cluster,$\varepsilon$,$\eta$);
foreach ($u \in V | u.state = noise$) do
    if ($|N_\varepsilon(u)| \geq \eta$) then
        Cluster=CreateNewCluster();
        Cluster.addNode(u);
        u.state=core;
        stack.push(u);
        Cascade_Create(Cluster,$\varepsilon$,$\eta$);
return Clustering;
```

Algorithm 4: Cascade_Create

```
input  : Graph,Cluster,$\varepsilon$,$\eta$
output: Clustering
while Stack is not empty do
    u=stack.pop();
    foreach ($n \in N_\varepsilon(u)$) do
        if n.state=core then
            if $n \notin Cluster$ then
                oldCluster=n.getCluster();
                oldCluster.absorb(Cluster);
                cluster=oldCluster;
            continue;
        Cluster.addNode(n);
        if ($|N_\varepsilon(n)| \geq \eta$) then
            n.state=core;
            stack.push(n);
        else
            n.state=border;
return Clustering;
```

Algorithm 5: Cascade_Expand

```
input  : Graph,Clustering,Cluster,$\varepsilon$,$\eta$
output: Clustering
while Stack is not empty do
    u=stack.pop();
    foreach $(n \in N_\varepsilon(u) | n.state \in \{noise, border\})$ do
        Cluster.addNode(n);
        if $(|N_\varepsilon(n)| \geq \eta)$ then
            if $n.state = noise$ then
                n.state=core;
                stack.push(n);
            else
                n.state=core;
                //Cluster Merge
                foreach $(C \in \zeta | n \in C)$ do
                    Cluster.addSubCluster(C);
                    foreach $(p \in C | p.state = border)$ do
                        if $(|N_\varepsilon(p)| \geq \eta)$ then
                            p.state=core;
                            stack.push(p);
                        else
                            p.state=border;
        else n.state=border;
return Clustering;
```

3.3 *Creating the Cluster Hierarchy for Explorative Visual Network Analysis*

Algorithm 6 describes how the top-down and bottom-up methods are used to generate the final cluster hierarchy. The list of parameter settings is generated by an heuristic-based function and should reflect the demands of the network analyst. For each parameter setting, the clusterings are computed by using either the bottom-up or the top-down approach.

4 Application

Last.fm[3] is a music community with over 20 million active users based in more than 200 countries. After a user signs up, a plugin is installed and all tracks a user listens to are submitted to a database.

From the *Last.fm* Website we obtained the user listening behavior of 1,209 users over an interval of 130 weeks (from March 2005 to May 2008). *Last.fm* provides

[3] http://www.last.fm/

Algorithm 6: CreateClusterHierarchy

```
input  : Graph
output: Clustering

PS=CreateParameterSettingList();
Direction=DecideDirection();
if Direction=BottomUp then
    ClusterHierarchy=DenGraph(Graph,PS[0].$\eta$,PS[0].$\varepsilon$);
    for $i \leftarrow 1$ to $(size(PS)-1)$ do
        ClusterHierarchy=BottomUp(ClusterHierarchy,PS[$i$].$\eta$,PS[$i$].$\varepsilon$);
else
    ClusterModel=DenGraph(Graph,PS[$size(PS)$].$\eta$,PS[$size(PS)$].$\varepsilon$);
    for $i \leftarrow (size(PS)-1)$ to $size(0)$ do
        ClusterHierarchy=TopDown(ClusterHierarchy,PS[$i$].$\eta$,PS[$i$].$\varepsilon$);
return Clustering;
```

for each user and interval (here one week) a list of the most frequently listened artists and the number of times the artist was played. Based on this information we determine user profiles for each interval and calculate the similarity of music preferences between users. Based on these similarities, we generate a graph in which the nodes represent the users and the edge weights code the distance between users based on the similarity of their music listening behavior [7].

For our analysis we used a Last.fm graph consisting of 1,209 nodes and 12,612 edges. The graph has an average degree of 20.8 and a density of 0.017. We applied DenGraph-HO and obtained the cluster hierarchy shown in Figure 4. The calculated cluster labels are based on the profiles of the users in the same cluster. Table 1 gives more details about the clustering of each hierarchy level. Shown are the parameters ε and η, the cluster label and the number of nodes in each cluster.

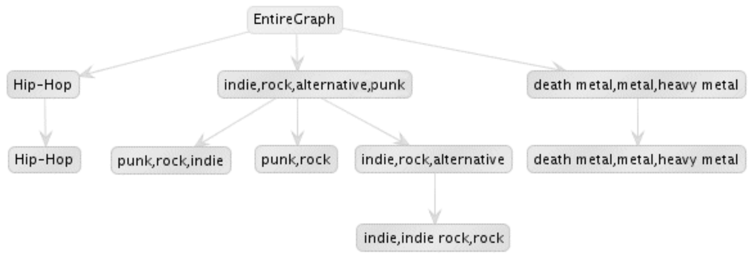


Fig. 4 DenGraph-HO: Hierarchy

The plausibility of the relation between a cluster and its subclusters is demonstrated by the *(indie, rock, alternative, punk)*-cluster that is divided in the clusters *(punk, rock, indie)*, *(punk, rock)* and *(indie, rock, alternative)*. Obviously, music gen-

Table 1 Overview about the cluster hierarchy

Level	ε	η	Cluster Labels	# Nodes
0	-	-	Entire Graph	1209
1	0.05	22	hip-hop	35
			indie, rock, alternative, punk	631
			death metal, metal, heavy metal	85
2	0.037	24	hip-hop	31
			punk, rock, indie	86
			punk, rock	33
			indie, rock, alternative	316
			death metal, metal, heavy metal	37
3	0.025	26	indie, indie rock, rock	120

res that are similar but separated in level 2 of the hierarchy are merged into one cluster in level 1.

The number of nodes per cluster increases, when going from a leave to the root of the cluster hierarchy. Due to the parameter setting of ε and η the clusters grow either through cluster merging or absorption of new nodes. These properties of DenGraph-HO and the efficient calculation of the cluster hierarchy are the base for the proposed zooming purpose.

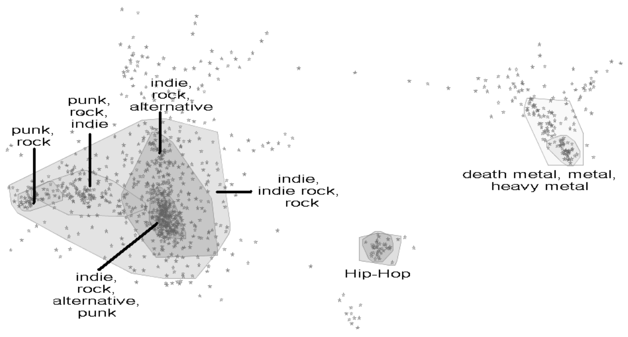
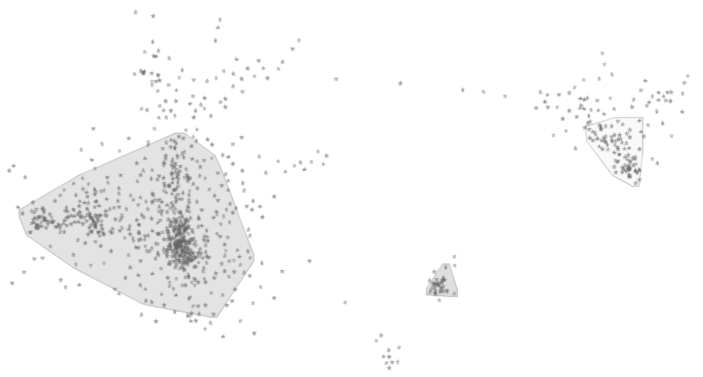
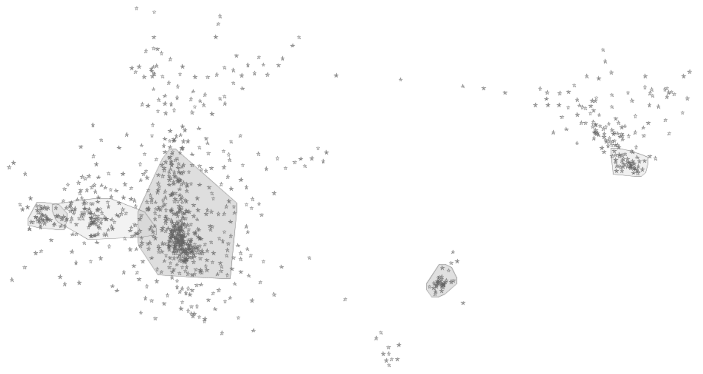


Fig. 5 DenGraph-HO: Graph and Clustering

Figure 5 shows the entire graph and the clusters produced by DenGraph-HO. For the sake of clarity edges are not drawn. Since we limited the number of hierarchy levels in our example and due to the small number of nodes, the graph can be easily understood. However, graphs with millions of nodes and a hierarchy depth greater than ten ask for appropriate tools.

The ability of zooming through the graph enriches our tool set and is an important step for studying the graph structure. Figure 6 shows the three zooming steps that are provided by DenGraph-HO for the given example. Each level of the hierarchy is shown as a single clustering.


(a) Level 1


(b) Level 2

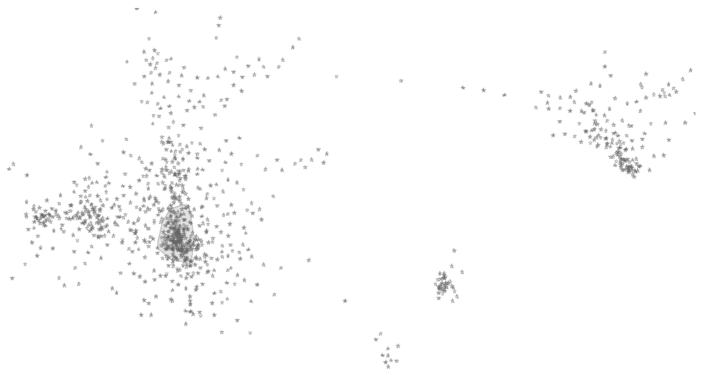

(c) Level 3

Fig. 6 Zoom-in Operation

5 Conclusion

In this paper we proposed the hierarchical density-based graph clustering algorithm DenGraph-HO. We demonstrated its practical use for explorative visual network analysis by applying the algorithm to a social network that we have obtained from the Last.fm music platform. The resulting clusters form groups of users that have similar music listening preferences. By calculating labels the clusters get a semantic meaning based on the music preferences of its members. The produced cluster hierarchy and clustered graph have shown that clusters with similar labels are located closely in the graph and the in hierarchy.

Since DenGraph-HO has proven its usefulness for our practical work, our next step is to integrate the incremental approach known from DenGraph-IO in a hierarchical incremental density-based graph clusterer DenGraph-HIO in order to analyze clusters over time.

Acknowledgements This work was supported by the members of the *distributedDataMining* BOINC [1] project (http://www.distributedDataMining.org).

References

1. Anderson, D.P.: Boinc: A system for public-resource computing and storage. In: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, GRID '04, pp. 4–10. IEEE Computer Society, Washington, DC, USA (2004)
2. Ester, M., Kriegel, H.P., Sander, J., Wimmer, M., Xu, X.: Incremental clustering for mining in a data warehousing environment. In: A. Gupta, O. Shmueli, J. Widom (eds.) VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA, pp. 323–333. Morgan Kaufmann (1998)
3. Falkowski, T.: Community Analysis in Dynamic Social Networks. Sierke Verlag, Gttingen (2009)
4. Falkowski, T., Barth, A.: Density-based temporal graph clustering for subgroup detection in social networks. Presented at Conference on Applications of Social Network Analysis (2007)
5. Falkowski, T., Barth, A., Spiliopoulou, M.: Dengraph: A density-based community detection algorithm. In: Proc. of the 2007 IEEE / WIC / ACM International Conference on Web Intelligence, pp. 112–115. IEEE Computer Society, Washington, DC, USA (2007)
6. Falkowski, T., Barth, A., Spiliopoulou, M.: Studying community dynamics with an incremental graph mining algorithm. In: Proc. of the 14 th Americas Conference on Information Systems (AMCIS 2008). Toronto, Canada (2008)
7. Schlitter, N., Falkowski, T.: Mining the dynamics of music preferences from a social networking site. In: Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining, pp. 243–248. IEEE Computer Society, Washington, DC, USA (2009)
8. Shetty, J., Adibi, J.: Enron email dataset. Tech. rep. (2004). URL http://www.isi.edu/adibi/Enron/Enron.htm