Clustering Social Networks using Competing Ant Hives

Pascal Held Alexander Dockhorn Benjamin Krause
Rudolf Kruse

Institute of Knowledge and Language Engineering
Department for Computer Science, Otto von Guericke University Magdeburg
Universitätsplatz 2, 39106 Magdeburg, Germany

E-Mail: {pheld, kruse}@iws.cs.uni-magdeburg.de
{alexander.dockhorn, benjamin.krause}@st.ovgu.de

May 19, 2015

Abstract Methods for clustering static graphs cannot always be transfered straight forward to dynamic scenarios. A typical approach is to reduce the number of updates by reusing results of previous iterations. But are there natural ways to implement dynamic graph clustering? This paper proposes a method which was derived by graph based ant colony algorithms. Similar to other clustering algorithms, multiple ant colonies are competing for the available nodes. Each hive creates ants, which will explore nearby graph structures and drop hive-specific pheromones on visited nodes. Over time, hives will collect nodes and will be relocated to the center of all collected nodes. In case of dynamic graph clustering, pheromone values can be reused in consecutive iterations. Our evaluation revealed that the proposed algorithm can lead to results on a par with the k-median algorithm and performs worse than Louvain clustering. However competing ant hives have the advantage of implicit noise detection, which comes at the cost of longer computation times. This can make it a suitable choice for certain clustering tasks.

1 Introduction

The rising development of web technology showed how fast networks can emerge and change. Social networks such as Facebook, MySpace and other websites were created and are very popular up to date. Their databases are source for a large set of mining tasks. Typical examples for network analysis are advert companies being more and more interested in providing user-specific content. Friendship relations or group dynamics can be of special interest, for example, when a friend of yours bought a product you might be interested as well. Especially if the person can be seen as leader of a social group, it could have more influence. However, the databases of all networks are growing rapidly and include millions of changes every minute. While static graph cluster analysis already resulted in multiple algorithms, adjustments for dynamic graphs are not always trivial.

Earlier graph analysis showed that ant-colony algorithms could be successfully applied to mining shortest paths between two instances [1] and finding clus-

ters [2]. An advantage of ant-colony algorithms is that they can be adjusted to dynamic scenarios by including a time-dependent decrease of pheromones. Results of previous runs are encoded in remaining pheromone information and can be reused in further iterations. We will show how a clustering algorithm based on k-median and multiple competing ant colonies can be implemented.
The three main goals of this paper are:

We want to propose a clustering algorithm based on competing ant hives. We will investigate the resulting cluster quality and time complexity for static graphs. See Section 3 for an explanation of the ant-hive algorithm.

Further on, Section 3.2 will show how the algorithm can be adopted to dynamic graphs by reusing hive locations of previous runs.

Finally our evaluation compares the algorithm to the well known alternatives k-median and the louvain method. Comparison will be done based on multiple cluster quality measures (see Section 2.2) and runtime of respective algorithms. The evaluation of different aspects will be explained in Section 4 followed by a detailed view on the results in Section 5.

2 Terminology

This chapter will introduce needed basic graph definitions for the rest of the paper. We will focus on defining clusters in social graphs and go through possible changes the graph can undergo in dynamic settings.

2.1 Social Networks/Graphs

The data can be modeled as a graph. Let a graph be a tuple $G = \{V, E\}$, where V is a set of vertices and E is a set of edges defined as $\{(u,v) \mid u,v \in V; u \neq v\}$. In case of a social network people will be presented as vertices and communication between them can be modeled as edges. Connections from one person to itself will be excluded, so the graph can be called simple since he does not include any self loops. Furthermore we will limit our analysis to undirected graphs where an edge $e = (u,v)$ also implies the existence of the edge $e' = (v,\ u)$.

The strength of communication between two instances could be modeled as weight $w(e) = (0,1]$, $e \in E$, where values close to 0 indicate nearly no communication and larger values indicate increasing levels of communication.

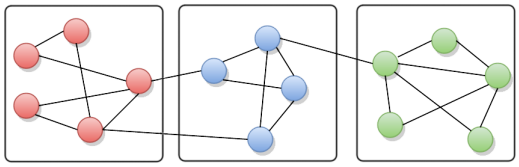


Figure 1: Graph with clustered structures. Framed node groups form a densely connected cluster. Only sparse connections can be found in between such clusters.

2.2 Cluster-definition

Graph analysis can focus on finding clusters. A typical approach is to define a cluster as a set of vertices with high intra-cluster density and high inter-cluster sparsity, meaning the number of edges between nodes of the cluster is higher than to nodes outside the cluster. Figure 1 shows an example of clustered nodes in a graph.

Let a clustering be a set of disjoint sets of nodes $\zeta = \{C_1, ..., C_n\}$ and C_1 to C_n the clusters of a graph. We can define two functions for measuring density and sparsity of a cluster C_i.

Let the coverage [3] of a cluster C_i be defined as the weight of intra-cluster edges compared to the weight

of all edges and be defined as:

$$cov(C_i) = \frac{\sum_{e \in E(C_i,C_i)} w(e)}{\sum_{e \in E} w(e)} \quad (1)$$

In order to measure the sparsity of a cluster C_i we define the conductance [3] which is the sum over all edges between the cluster C_i and all vertices outside the cluster $V \backslash C_i$.

Let the conductance be:

$$cond(C_i) = \sum_{e \in E(C_i, V \backslash C_i)} w(e) \quad (2)$$

Another common graph measure for cluster evaluation is modularity $Q(\zeta)$ [4]. Networks with high modularity are densely connected within clusters, whereas nodes from different clusters are only sparsely connected. While modularity proved useful in a lot of network clustering applications, its properties are not well understood up to now [4].

$$Q(\zeta) = \sum_{C_i \in \zeta} \left[\frac{|E(C_i,C_i)|}{|E|} - \left(\frac{\sum_{C' \in \zeta} |E(C_i,C')|}{2|E|} \right)^2 \right] \quad (3)$$

Note that the second relation includes the sum of all edges of cluster C_i. Since modularity is defined for unweighted graphs only we can also use the node degree of all nodes contained in cluster C_i. This leads to a more readable version defined as:

$$Q(\zeta) = \sum_{C_i \in \zeta} \left[\frac{|E(C_i,C_i)|}{|E|} - \left(\frac{\sum_{v \in C_i} deg(v)}{2|E|} \right)^2 \right] \quad (4)$$

2.3 k-Median method

The k-Median method was originally defined for data points, but can also be used on graphs [5]. It starts by randomly choosing nodes as cluster centers. Iteratively, each node of the graph is assigned to the cluster with the closest cluster center. As a second step of the iteration each cluster center is reassigned to the cluster node with the shortest average path length to all nodes of the same cluster. This process will be repeated until convergence.

We made further adoptions for dynamic scenarios. First we use the final cluster centers of one time step as initialization for the next one. This leads to problems where previous cluster centers were seperated by large graph changes, such that the cluster included only few nodes. We fixed this by using only a portion of earlier cluster centers and initializing remaining cluster centers at random. Even if a cluster is separated, it is likely to be reinitialized in further time steps. This way overall quality of found clusters was improved.

2.4 Louvain method

The Louvain method [6] is based on a hierarchical clustering approach where each node starts in its own cluster. The used linkage criterion is the modularity of the resulting merge. In a second step clusters are aggregated into nodes. This process is repeated until no gain in modularity is obtained.

The result can be visualized as a dendrogram, where parts of it could be identified as substructures of clusters. Experiments showed that the greedy approach of Louvain method performs well in optimizing modularity. The process showed to be comparatively fast and experiment evaluations estimate the computational effort with $O(n \cdot log\ n)$. Exact modularity optimization is NP-hard. Experiments showed community sizes can be highly skewed [7].

Because the runtime is very low, we did not further optimize the method for dynamic graphs. Each time step is computed as usual. Therefore found community structures and numbers could largely diverge from previous time steps.

3 Competing Ant Hives

Swarm intelligence algorithms already showed viable for clustering tasks and, were, for example, used for determining shortest preference-based paths in networks [8]. We will follow a related approach and use multiple ant colonies, each represented by a hive at its center, to compete for available nodes of the network.

3.1 Competing Ant Hive Algorithm

Every cluster $C_1, \ldots, C_k$ will be represented with a hivenode as initial starting point. As a heuristic for initializing the hive positions, we use the k nodes with the highest node degree. For each iteration ants will be exploring the graph near a hive position and drop hive specific pheromones on visited nodes. These pheromones can be used for prioritized movement of ants and assigning nodes to a certain ant hive. At the end of an iteration each hive will be relocated given the current clustering.

The algorithm can be outlined by the following pseudocode:

```
1:  function COMPETINGANTHIVES
    (Graph G, clusters k, iterations i)
2:      intializeGraph(G)
3:      hives ← initializeHives(G,k)
4:      for 1 to i do
5:          pheromones ← walkAnts(G,hives)
6:          G.addPheromones(pheromones)
7:          clustering ← assignNodes(G,hives)
8:          hives ← relocateHives(G,clustering)
9:      end for
10:     return clustering
11: end function
```

Since the output of the clustering algorithm is highly dependent on the type of ant colony used, the following subsections present a detailed description and justification of crucial parts of the implemented algorithm.

3.1.1 Extending the graph definition

First we need an extended definition of a graph by including pheromone information. Therefor we introduce a pheromone graph $G = \{V, E, P_V\}$, where P_V is a matrix of $\mathbb{R}^{|V| \text{ x } k}$ pheromone weights per hive with respect to the vertices. We will use $pher_h(v)$ as shorthand for the pheromone value on node $v \in V$ of hive h.

An Ant Hive is defined as starting point for ants of one colony. Using multiple ant hives results in competing ant hives, which try to get hold of network nodes. Ants of an ant hive will wander on the graphs structure and drop pheromones on their way. Pheromones of different hives will be treated separately.

Graph analysis showed that social graphs tend to have a node degree distribution following a power-law distribution [9]. The hubs would be a plausible spawning position for the ant hives.

3.1.2 Ant Types

The ants' movement strategy can lead to a global exploration or a local exploitation of the graphs structure. We implemented three different types of ants and will compare the resulting pheromone graphs and clustering results in later sections. Our ant types are:

Random Ant: This ant-type will be used as a baseline and is of interest to compare more sophisticated ant types with a total random behavior. The ant starts at a certain vertex and has the option to walk along one randomly chosen edge to a neighboring vertex. Pheromones will be dropped by the ants for later hive assignments. An advantage of this ant type is the low computational complexity. Randomly choosing a node can be done with $O(1)$

Base Ant: Earlier analysis of ant swarm algorithms showed that it can be beneficial to relate the probability of choosing a direction by current pheromone values. For an ant of hive h let the probability of choosing a neighbor v be:

$$P(v) = \frac{w(u,v) \cdot pher_h(v)}{\sum_{(u,v') \in E} w(u,v') \cdot pher_h(v')} \quad (5)$$

where u is the current node and v a neighboring vertex connected by an edge $e = (u,v) \in E$. In comparison to the random ant approach this calculation has an increased complexity of $O(n)$, since every neighbor has to be iterated. In average the complexity will be dependent on the density of the graph, because the number of neighbors will be much less than n in large graphs. This should result in bearable computational effort.

Simple Path Ant: Since the base ant could be walking back and forth between two neighboring nodes, the amount of pheromones of both nodes could increase too fast. For this reason we introduce the simple path ant, which can only move to nodes and edges not visited yet. The simple path ant stores all visited nodes of the current time step. When choosing a direction, the nodes in the memory are not considered in the calculation of the probability. Since all directions can be blocked by the list of forbidden nodes, the ant just stops to move if it reaches a dead end. In addition to the high computational complexity of calculating the node probability, this ant has an increased memory usage for the visited nodes.

The algorithm was always used with one specified ant type. Each hive spawns the same amount of ants on its own node. Ant movement is determined by the above description of the chosen ant type. Pheromones were placed in a batch mode, where after each ant walked one step forward, pheromone values are increased at the current position of each ant. Hence movement of the first ants does not influence the movement of later ants for the same step number. See Section 4.1 for a comparison of clustering results depending on used ant types.

3.1.3 Clustering approach

Nodes will store an amount of pheromones per hive. The assignment of a node to a hive will be done by choosing the hive with the highest portion of pheromones on this node. Consequently the hive will be reassigned to the node with the minimal average shortest path length to all nodes assigned to the same hive. For the next iteration pheromones will be degraded by a certain percentage. Hence we can control the impact of previous pheromone information.

A special property of this assignment is implicit noise detection. Nodes without any pheromones can be separated as a noise cluster. This will automatically hold true for nodes further away than the maximal number of steps from the hive centers. Depending on the parameters it can be very likely to exclude nodes as noise at the border of the network.

3.2 Ant hives for dynamic graphs

An advantage of using ant-colony algorithms is that they are easily applicable on dynamic scenarios. As it can be observed in nature, pheromone density decreases over time if the path is not walked on that often. It is also observable that ant hives will be relocated to closer food sources if previous ones are depleted.

As of now we explained how the algorithm works for static graphs, but we can use a similar approach as it was used for k-Median in order to adapt to dynamic scenarios. Ant hive locations of the previous time step will be used as initialization for the current one.

Pheromone values can be degraded and reused as well.

4 Experiments

We divided our experiments in three stages. First we investigated how the ant type influences the clustering quality. Later evaluations will be based on the best ant-type. Secondly we compare the clustering results of our competing ant hive algorithm with other static graph clustering algorithms on different graph settings. Our third evaluation focused on a clustering comparison on dynamic graphs.

4.1 Ant Type Comparison

First we evaluated the qualitative clustering behavior of varying ant types. Therefor 100 graphs of the random Barabási-Albert-model [9], Watts-Strogatz-model [10] and powerlaw-cluster-model [11] were instantiated. Table 1 shows the experimental setup. The graphs vary in global and local graph properties. Especially the average path length and the node degree distribution will be of interest. While the Barabási-Albert-model and the powerlaw-cluster-model produce a node degree distribution following a power law distribution, the Watts-Strogatz-model produces a degree distribution resembling a Dirac-delta function centered at k. Every graph was clustered three times with the proposed competing ant hive clustering, each using a different type of ant. We created between 5 to 10 ant hives and measured coverage, conductance, modularity and time needed for the clustering. The evaluation will also show how robust the ant hive clustering algorithm works for different graph settings.

4.2 Static Graph Clustering Evaluation

In order to compare our algorithm with other clustering techniques we used the same experimental setup (see Table 1) and compared the results of the best ant type to other clustering algorithms.

Table 1: Experiment setup

Graphtype	$\|V\|$	k	p
Watts-Strogatz	100	8	0.5
Barabási-Albert	1000	5	-
powerlaw-cluster	1000	5	0.5

The louvain and the k-median method were used for comparison in these static scenarios. Since the louvain method is able to determine the number of clusters, we executed it first and initialized both other algorithms with the same amount of clusters. Other cluster numbers would lead to skewed evaluation results, since coverage computes the average over all clusters.

4.3 Dynamic Graph Clustering Evaluation

For the evaluation of a dynamic scenario the Enron-dataset was used. It comprises of e-mail communication between 150 Enron-employees. A sequence of 100 time steps of a preprocessed version of the Enron-dataset was used [12]. Clustering results were compared to the output of k-median and louvain clustering. The louvain method was used again to determine the number of clusters. For the case that the louvain clustering found less clusters than the last run, only the largest cluster centers were reused as initialization for k-median and ant-hive. If the cluster number increased in comparison to previous time steps additional centers were initialized at random.

As for the static scenario, coverage, conductance, modularity and time consumption was measured.

5 Results

5.1 Ant Type Comparison Results

Boxplots of all three graph types are shown in Figure 2(a). Since the results are fairly similar to each other, only the results of using Watts-Strogatz-graphs are shown in Figure 2(b) for a detailed comparison. This graph type yielded the least variance and makes it easier to compare the results.

It can be seen that the base ant type dominates the other ant types in coverage, conductance and modularity. This comes at the cost of a higher computational time in comparison to the random ant. Since the simple path ant type has to do additional checks it is even slower than the base ant. However this does not result in a large increase of clustering accuracy. The performance even decreased in case of modularity. This could be the case due to the larger spreading of ants from one starting point, since it is forbidden to walk back.

Based on the results of our comparison of different ant types, we chose the base ant type for comparison with other algorithms in the following subsections.

5.2 Static Graph Clustering Results

Figure 3 shows the clustering results rated by coverage, conductance, modularity and time effort sorted by graph type. In general the results of the ant hive clustering are comparable to k-median. However both are performing worse than louvain clustering, which yielded the best results in all four evaluation measures.

The computational time of the ant hive algorithm could be further adjusted by changing the number of ants or the number of steps per ant. Increasing the ant number did not improve the clustering in our experiments, but decreasing it by a large degree can lead to a significant decrease in clustering quality. This is because many nodes, even nodes in the neighborhood of a hive, will not be part of a cluster and labeled as noise since no ant did visit it on its path.

5.3 Dynamic Graph Clustering Results

Louvain clustering seems to be superior to both other clustering algorithms. Conductance, modularity and time consumption are overall better than both competitors. For the first time steps louvain clustering results in good coverage values, but k-median and ant hive clustering result in the same close conductance values in later time steps.

6 Conclusion

In this paper we proposed a clustering algorithm based on ant colonies, where hives are competing for nodes in a graph. Multiple ant types were proposed and a comparison showed that a total random behavior performs much worse than a pheromone based direction preference. However forbidding loops in ant paths increases the computation and does not lead to an improvement. This could be due to a loss of cluster compactness.

The evaluation showed that our algorithm can produce comparable clustering results to other graph mining algorithms like k-median. However its performance is far behind the louvain clustering, which performed better in all used evaluation measures. The high performance differences could be a result of the greedy modularity optimization method used in louvain clustering, since modularity, coverage and conductance are related to each other.

It could be argued to use the ant hive clustering algorithm, if a certain number of clusters is desired. Also implicit noise detection can be beneficial. The first can also be achieved by k-median clustering but

both characteristics are not supported by the louvain clustering algorithm.

The proposed ant colony algorithm showed that a natural inspired clustering could yield a functioning dynamic graph clustering. Unfortunately the higher computation time and in generally worse results in the used evaluation measures showed our algorithm to be less effective than already known graph clustering algorithms. Future work could go into other ant walking behaviors dedicated for optimizing modularity or finding an optimal model parameterization depending on the graphs properties.

References

[1] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *Biosystems*, vol. 43, no. 2, pp. 73–81, Jul. 1997.

[2] P. S. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, "An ant colony approach for clustering," *Analytica Chimica Acta*, vol. 509, no. 2, pp. 187–195, May 2004.

[3] T. Falkowski, "Community analysis in dynamic social networks," Ph.D. dissertation, Otto von Guericke University Magdeburg, 2009.

[4] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On Modularity Clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 2, Feb. 2008.

[5] T. Saha, C. Domeniconi, and H. Rangwala, "Detection of Communities and Bridges in Weighted Networks," in *Machine Learning and Data Mining in Pattern Recognition*, P. Perner, Ed. Springer Berlin Heidelberg, 2011, no. 6871, pp. 584–598.

[6] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, Oct. 2008.

[7] J. M. Pujol, V. Erramilli, and P. Rodriguez, "Divide and Conquer: Partitioning Online Social Networks," May 2009.

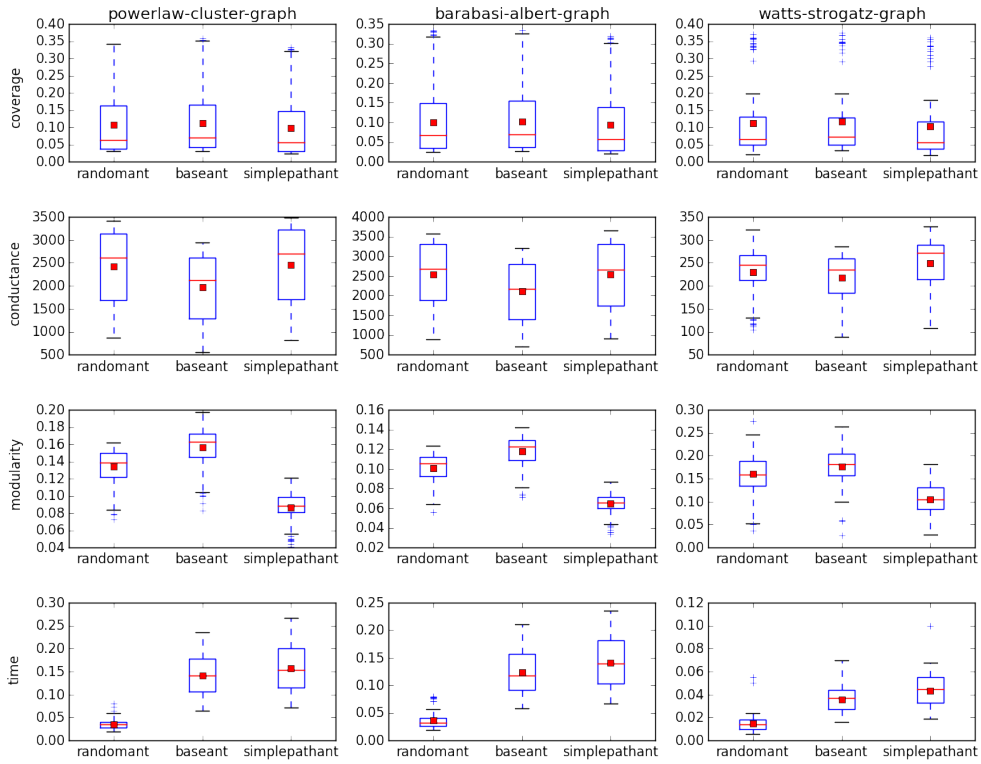
[8] S.-H. Ok, W.-J. Seo, J.-H. Ahn, S. Kang, and B. Moon, "An ant colony optimization approach for the preference-based shortest path search," *Journal of the Chinese Institute of Engineers*, vol. 34, no. 2, pp. 181–196, 2011.

[9] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

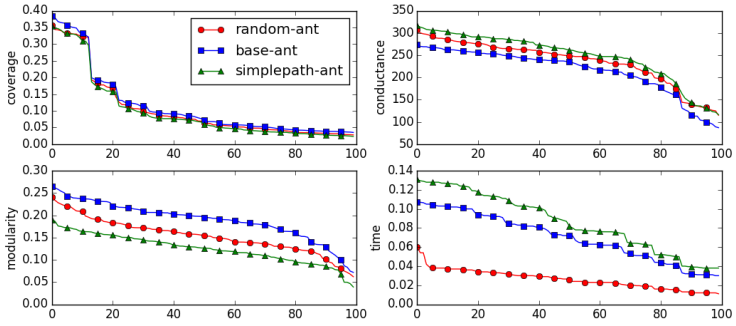
[10] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.

[11] P. Holme and B. J. Kim, "Growing scale-free networks with tunable clustering," *Physical review E*, vol. 65, no. 2, p. 026107, 2002.

[12] P. Held and K. Dannies, "Clustering on dynamic social network data," in *Synergies of Soft Computing and Statistics for Intelligent Data Analysis*, ser. Advances in Intelligent Systems and Computing (AISC), R. Kruse, M. R. Berthold, C. Moewes, M. Á. Gil, P. Grzegorzewski, and O. Hryniewicz, Eds., vol. 190. Heidelberg Berlin: Springer-Verlag, 2012, pp. 563–571.

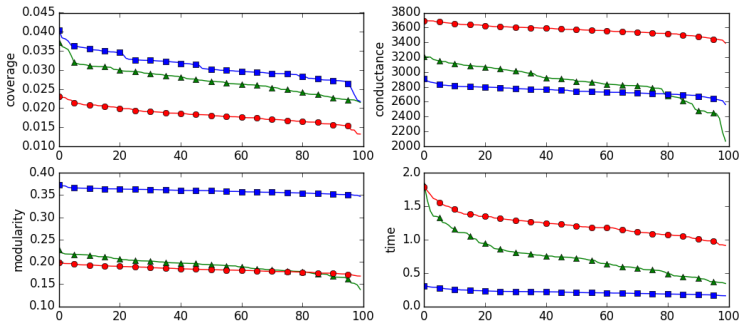


(a) Boxplots of clustering results per ant type measured on 100 random graphs per graph model. Qualitive comparison of all ant type per graph type. Scales of the columns differ.

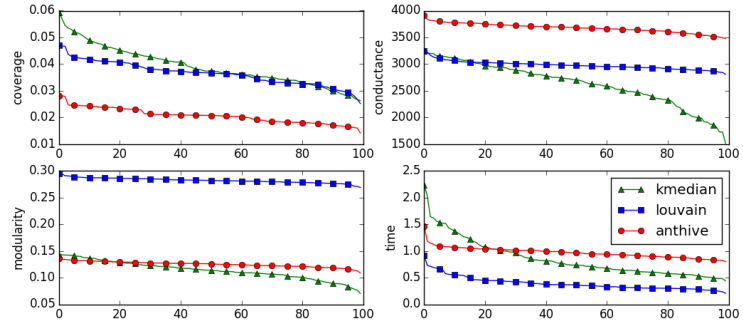


(b) Detailed results for 100 Watts-Strogatz-graphs per ant type.

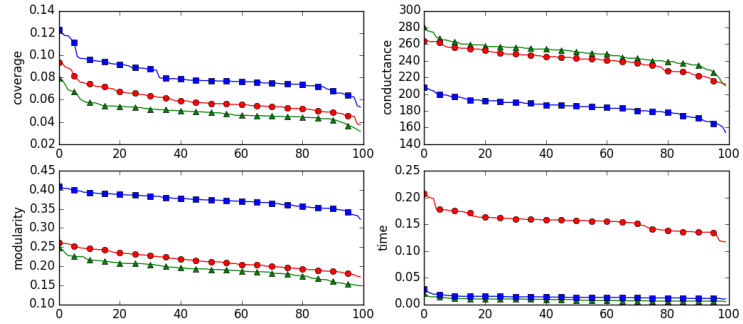
Figure 2: Comparison of clustering results using proposed ant types.



(a) Comparison of clustering results using louvain, k-median and ant-hive clustering on randomly generated powerlaw-cluster-graphs with parameters set to $n = 100, k = 5, \mathrm{p} = 0.2$.



(b) Comparison of clustering results using louvain, k-median and ant-hive clustering on randomly generated Barabási-Albert-graphs with parameters set to $n = 100, k = 5$.



(c) Comparison of clustering results using louvain, k-median and ant-hive clustering on randomly generated Watts-Strogatz-graphs with parameters set to $n = 100, k = 8, p = 0.5$.

Figure 3: Static graph clustering evaluation using multiple random generated graph.

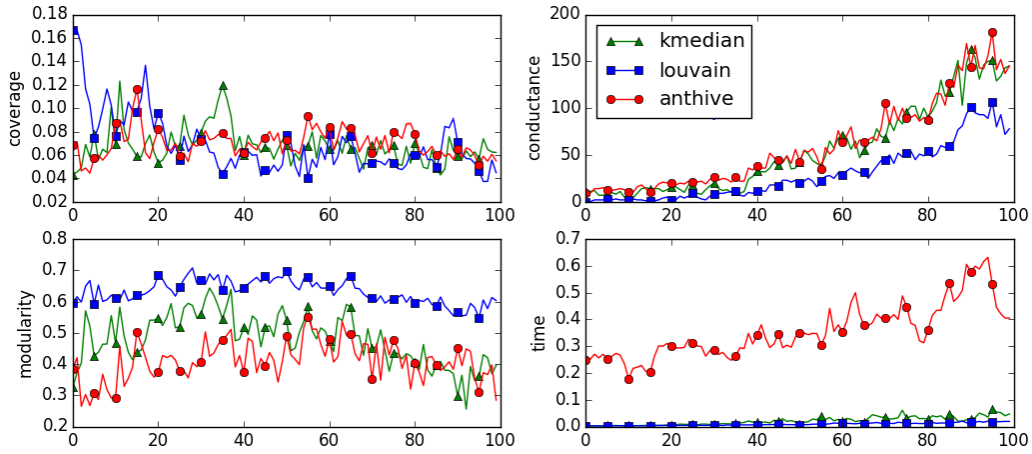


Figure 4: Comparison of clustering results using louvain, k-median and ant-hive clustering on consecutive time steps of the Enron-dataset.