Graph Clustering with Density-Cut

Junming Shao [*]   Stefan Kramer [†]

Abstract

How can we find a good graph clustering of a real-world network, that allows insight into its underlying structure and also potential functions? In this paper, we introduce a new graph clustering algorithm **Dcut** from a density point of view. The basic idea is to envision the graph clustering as a *density-cut* problem, such that the vertices in the same cluster are densely connected and the vertices between clusters are sparsely connected. To identify meaningful clusters (communities) in a graph, a density-connected tree is constructed in a local fashion. Owing to the density-connected tree, **Dcut** allows partitioning a graph into multiple densely tight-knit clusters directly. We systematically evaluate our new clustering approach on synthetic as well as real data to demonstrate its good performance.

1 Introduction

In recent years, the study of graph clustering has attracted a lot of attention, and many algorithms have been developed based on different criteria, e.g. *betweenness* [10], *normalized cut* ($Ncut$) [15], *minimum-cut tree* [4], *modularity* [12], to mention a few. Although many established approaches have already achieved some success, finding the intrinsic clusters in complex networks is still a big challenge [3]. Up to now, most previous studies struggle to find a good graph clustering by minimizing the similarity between clusters, e.g. *minimum cut* or *modularity*. However, the similarities of vertices in a graph are considered only little. For example, in Fig. 1, we show that the typical graph clustering algorithms, like *Ncut*, may produce a bad grouping without considering the topological similarities among nodes. A good cut should minimize the similarities of vertices between clusters while maximize the similarities of vertices within each cluster.

In this paper, we introduce a new density-based criterion for measuring the "goodness" of a graph clustering. The basic idea is to consider the graph clustering as a density-cut problem by removing the edges in a proposed density-connected tree (cf. Section 3.3). We expect that the vertices of resulting clusters are densely connected while the vertices between clusters are sparsely linked. The "good cut" in Fig. 1 is viewed
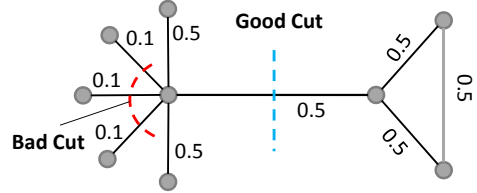
Figure 1: A simple fictitious weighted graph. Here traditional graph clustering algorithms may produce a bad clustering, which is indicated by the red dashed line. A better partitioning should also consider the similarity of vertices in groups.

as "good", because the vertices in the same group have the same or similar topological structures, instead of focusing on minimum cuts (e.g. *normalized cut*, *ratio cut*) or expected cuts (e.g. *modularity*) between two partitions only. Our proposal is to measure similarities of vertices in and between graph clusters by constructing a density-connected tree, where any two adjacent vertices with highest similarity (i.e. with strong edge weight and similar topological structure) are densely linked together. Based on the properties of the density-connected tree, a good graph partitioning with *density-cut* criterion is efficiently identified.

Contributions In this paper, we present a new graph clustering method *Dcut*, which partitions a graph from the density point of view. The major benefits of *Dcut* can be summarized as follows:

- *Density-cut criterion.* *Dcut* provides a density-cut criterion for graph clustering. The new criterion is capable of measuring the quality of a graph clustering in a more natural and precise way.
- *Good partitioning.* *Dcut* allows producing a good graph partitioning, thanks to the intuitive density-cut criterion. By characterizing the density of any two adjacent vertices in local fashion, *Dcut* can easily partition a graph into multiple clusters with densely connected vertices.
- *Intuitive graph structure.* *Dcut* generates an intuitive and interpretable density-connected tree, which provides a connectivity map of vertices in

a graph from a local density perspective.

The remainder of this paper is organized as follows: In the following section, we briefly survey related work. Section 3 presents our algorithm in detail. Section 4 contains an extensive experimental evaluation, before we conclude in Section 5.

2 Related Work

During the past several decades, many approaches have been proposed for graph clustering, such as [9], [15], [12] etc. Due to space limitations, we can only review the closest approaches from the literature. For detailed reviews of graph clustering, please refer to [14].

Minimum Cut. The minimum-cut criterion based graph clustering refers to a class of well-known techniques which seek to partition a graph into disjoint subgraphs such that the number of cuts across the subgraphs is minimized. Wu and Leahy [18] has proposed a clustering method based on such minimum cut criterion, where the cut between two subgraphs is computed as the total weights of the edges that have been removed. $k-$disjoint subgraphs are obtained by recursively finding the minimum cuts that bisect the existing segments. To avoid an unnatural bias towards splitting small-sized subgraphs based on the minimum-cut criterion, *ratio cut* [6] has been introduced, and it uses the second smallest eigenvalue of the similarity matrix to find the suitable cut. In the same spirit, Shi and Malik [15] has proposed the *normalized cut*, to compute the cut cost as a fraction of the total edge connections to all the nodes in a graph. To optimize this criterion, a generalized eigenvalue decomposition was used to speed up computation time.

Modularity. Recently, modularity has been developed to measure the division of a network into communities. Unlike minimum-cut related approaches which investigate the number of edges or the total number of edge weights between two subgroups, modularity identifies a good cut by measuring the expected edges between clusters. Modularity-based graph clustering methods [11], [12] partition a network into groups to ensure the number of edges between two groups is significantly less than the expected edges.

Multi-Level Clustering. Metis is a class of multi-level partitioning techniques proposed by Karypis and Kumar [8], [9]. Graph clustering starts with constructing a sequence of successively smaller (coarser) graphs, and a bisection of the coarsest graph is applied. Subsequently, a finer graph is generated in the next level based on the previous bisections. At each level, an iterative refinement algorithm such as Kernighan-Lin (KL) or Fiduccia-Mattheyses (FM) is used to further improve the bisection. A more robust overall multilevel paradigm has been introduced by Karypis and Kumar [9], which presents a powerful graph coarsening scheme. It uses simplified variants of KL and FM to speed up the refinement without compromising the overall quality.

Markov Clustering. The Markov Cluster algorithm (MCL) [2] is a popular algorithm used in life sciences based on the simulation of (stochastic) flow in graphs. The basic idea is that dense regions in sparse graphs correspond to regions in which the number of random walks of length k is relatively large. MCL basically identifies high-flowing regions representing the graph clusters by using an inflation parameter to separate regions of weak and strong flow.

3 Graph Clustering based on Density-Cut

In this section, we present the *Dcut* algorithm in detail.

3.1 A Density-based Criterion for Graph Clustering
As stated in Section 1, we consider the problem of graph clustering from an intuitive perspective: *density*. We expect to find a good clustering if the vertices in each cluster are densely connected and the vertices between clusters are sparsely linked. In contrast to previous graph clustering algorithms, which treat "density" as the total number of links or edge weights in or between clusters, we consider whether vertices in and between clusters are densely connected based on the similarities among adjacent nodes in a local fashion. If the similarity between two adjacent vertices in a graph is high, they are viewed as densely connected, and vice versa. To identify tight-knit clusters, a density-connected tree is further proposed to look into the connection densities of vertices in a whole graph, where the two adjacent vertices with highest similarity (e.g. having strong edge weight and similar topological structure), are linked together in the tree. Built upon the density-connected tree, a good graph clustering based on the density-cut criterion is easily identified. In the following, we will first elaborate on how to measure the similarities between adjacent nodes.

3.2 Node Similarity Measure
For the purpose of graph clustering, a similarity measure needs to be defined so that similar nodes can be assigned into the same group from the density point of view. But before that, we start with some necessary definitions.

Definition 1 (UNDIRECTED WEIGHTED GRAPH)
Let $G = (V, E, W)$ be an undirected weighted graph, where V is the set of nodes, E is the set of edges and W is the corresponding set of weights. $e = \{u, v\} \in E$ indicates a connection between the nodes u and v. $w(u, v)$ represents the weight of edge e. $\forall e = \{u, v\} \in$

$E, w(u,v) = 1$, in case of unweighted graph.

Definition 2 (NEIGHBORS OF VERTEX u) Given an undirected graph $G = (V, E, W)$, the neighborhood of a node $u \in V$ is the set $\Gamma(u)$ containing node u and its adjacent nodes.

$$(3.1) \qquad \Gamma(u) = \{v \in V | \{u, v\} \in E\} \cup \{u\}$$

Unlike most previous graph clustering, which use edge weight to represent the similarity of two connecting nodes, we characterize the similarity between any two adjacent nodes by combining their edge weights and their local topological structures together.

In this study, the Jaccard coefficient [7] is used to quantify their local topological similarity. Generally, the more common neighbors two adjacent nodes have, the more similar they are.

Definition 3 (JACCARD COEFFICIENT) Given a graph $G = (V, E, W)$, the Jaccard coefficient of any two adjacent nodes u and v is defined as:

$$(3.2) \qquad \rho(u, v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$$

By considering the topological structure and edge weight simultaneously, the straight and intuitive way is to multiply the Jaccard coefficient with edge weight, where Jaccard coefficient captures the local structural similarity of two connecting nodes while the edge weight is the connection intensity between the two adjacent nodes. If the graph is unweighted (all weights are 1), node similarity measure is transferred as Jaccard coefficient. For more complicated weighted combination is not discussed in this study. Finally we define the similarity of any two adjacent nodes as follows.

Definition 4 (NODE SIMILARITY) Given any two adjacent nodes u and v in the graph G, the similarity of the two nodes u and v is defined as:

$$(3.3) \qquad s(u, v) = \rho(u, v) * w(u, v)$$

3.3 Density-connected Tree Based on the similarity measure, we construct a density-connected tree (DCT) for clustering. The basic idea is to generate a density-connected chain which connects all vertices in a local density fashion. In this tree, all vertices of an original graph are directly linked, and each edge is associated with a weight representing the density connection (similarity) between two connected nodes. It is expected that similar vertices are densely connected in the tree while vertices in different communities are lightly connected. Formally, the steps of constructing the density-connected tree are as follows.

Algorithm 1: $T = DCT(G)$

Input: $G = (V, E, W)$
Output: T

$T = null$;
Set $\forall v \in V$ as unchecked ($v.checked = false$);
Randomly selected one node $u \in V$;
Set $u.checked = $ true;
$u.connect = null$, and $u.density = null$;
$T.insert(u)$;

while $T.size < V.size$ **do**
  $maxv = -1; p = null; q = null$;
  for $i = 1$ **to** $T.size$ **do**
    $u = T.get(i)$;
    for $j = 1$ **to** $\Gamma(u).size$ **do**
      $v = \Gamma(u).get(j)$;
      if $v.checked == false$ **then**
        if $s(u, v) > maxv$ **then**
          $maxv = s(u, v)$;
          $p = v$;
          $q = u$;
        end
      end
    end
  end
  $p.checked = $ true;
  $p.connect = q; p.density = maxv$;
  $T.insert(p)$;
end

Let $G = (V, E, W)$ be the original graph. T is the density-connected tree under construction, and $T = null$ at the beginning. During the generation phase, two separate sets of vertices are maintained, where the first set is the vertices that have been inserted into the tree, and the second set is potential vertices for next insertion. For each step, the algorithm selects one node from the second set which maximizes the similarity to one existing node in the constructed tree. Specifically, in the initial phase, T is an empty set and the status of all vertices are set as *unchecked*. The construction starts with randomly selecting any node in V (e.g. u). We set $u.checked = true$, $u.connect = null$, $u.density = null$, and insert u into T at the first step. In the second step, we search all unchecked adjacent vertices of nodes (neighbors) in T, and find one node (e.g. v) which has the highest similarity to one node already in T (e.g. u) according to definition 4. Subsequently, we set $v.checked = true$, $v.conncet = u$, $v.density = s(u, v)$, and further insert v into T which directly links to the node u. The second step is repeated until all vertices have been inserted into the tree. Formally, the algorithm is described in Algorithm 1.

To illustrate DCT generation, Fig. 2 takes the
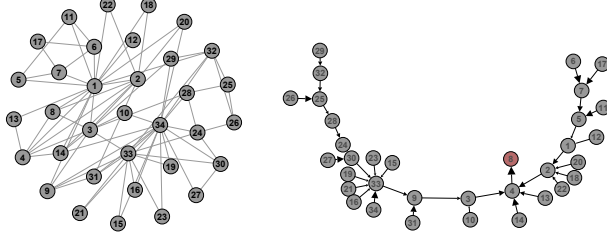
(a) Zachary's Karate Club (b) Density-Connected Tree

Figure 2: The illustration of construction of the density-connected tree. The thickness of the arrows in the density-connected tree indicates how densely connected two nodes are.

well-known Zachary's karate club network [19] as an example. To construct its density-connected tree, one node is first randomly selected (e.g. node "8" in this example, see Fig. 2 (b)). Next, all unchecked adjacent vertices of node "8" are viewed as the potential vertices for next insertion, namely, the nodes of "1", "2", "3" and "4". As the node "4" has the maximum similarity with node "8", it is further inserted into the tree, which directly connects it to the node "8" with the edge weight representing the similarity between the two nodes. For the next step, as there are already two vertices (node "8" and node "4") in this tree, all unchecked adjacent vertices are: $\Gamma(8) \bigcup \Gamma(4) \backslash \{4, 8\}$ (i.e. the nodes of "1", "2", "3", "13" and "14"). For the five potential vertices, the node "14" has the highest similarity with the node "4" in the constructed tree, and thereby node "14" is further inserted into the tree. Similarly, it directly connects to node "4" with the corresponding similarity. This procedure is repeated until all vertices have been inserted into the tree.

In the density-connected tree, all vertices with the highest similarity are densely connected in a local fashion to form tight-knit components. It thus provides a summarization of the graph structure from the density perspective (Fig. 2(b)).

Theorem 1 *The density-connected tree (DCT) is unique for a given graph, if any two adjacent nodes have a distinct similarity.*

Proof. Supposing there are two separate sets R and S during the generating phase at any step, where R is the set of vertices that have been inserted into the tree, and S is the set of unchecked neighbors of vertices that are already in R. The next node (e.g. v) is selected from S, which has maximum similarity with one node (e.g. u) from R. This means for each node, it is always connected with its most similar adjacent node. As any two adjacent nodes in the graph have a distinct similarity, the connection of nodes u and v is unique. Thus, the density-connected tree is unique.

3.4   The Dcut Algorithm Since DCT captures the density connectivity of vertices in a graph well, where the vertices with similar topological attributes and strong intensities of connections are densely linked together while the connections between components are lightly connected (the similarity of the two nodes connecting the two components is relatively low), it provides an intuitive solution to cut the edges in the tree directly to obtain a density-driven graph clustering.

Formally, we propose a new density-based criterion for measuring the "goodness" of a graph clustering. Instead of investigating the value of total (or normalized) edge weights connecting the two partitions, our measure computes its density connection between the two partitions based on the density-connected tree. We call this measure the *density cut* (*Dcut*):

$$(3.4) \qquad Dcut(C_1, C_2) = \frac{d(C_1, C_2)}{min(|C_1|, |C_2|)},$$

where C_1, C_2 are the two partitions, $d(C1, C2)$ means the corresponding density connecting the two partitions. The term of $min(|C_1|, |C_2|)$ is used to avoid the bias towards splitting small sets of vertices.

As DCT connects all vertices without cycle, each edge connects two components of a graph. Thereby, the intuitive bipartitioning of a graph in terms of density can be easily achieved by cutting one edge in the DCT. Instead of seeking to partition an original graph such that vertices in the same partition are densely connected and the vertices across different partitions are lightly connected, *Dcut* allows recursively finding the optimal cut on the DCT directly.

Generally, supposing we want to partition a graph into k disjoint clusters, the *Dcut* algorithm runs in the following steps.

1. Given a graph G, compute the similarities between adjacent vertices based on the node similarity measure (Definition 4).

2. Construct its density-connected tree (see Alg. 1).

3. Partition the DCT by removing the edge with minimum *Dcut* value of the resulting two components.

4. Recursively repartition the segmented DCT until k components of the graph are obtained.

Fig. 3 illustrates the graph clustering with density-cut criterion on the karate club network. Based on the constructed density-connected tree, the *Dcut* values for cutting all edges are computed, and the optimal cut with the minimum *Dcut* value is found between node "9" and node "3". Removing this edge from the DCT results in the two partitions shown in Fig. 3.
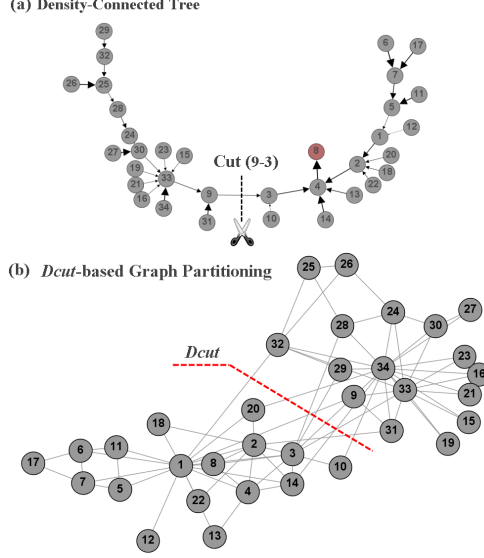
(a) **Density-Connected Tree**

(b) **Dcut-based Graph Partitioning**

Figure 3: The illustration of graph clustering based on the density-connected tree.

4 Experiments

In this section, we evaluate our proposed algorithm *Dcut* on synthetic as well as real-world data to demonstrate its benefits.

Comparison methods. To examine the performance of *Dcut*, we compare it to two closely related graph clustering algorithms: the normalized cut criterion based graph clustering method *Ncut* [15] and the *modularity*-based graph clustering algorithm by Newman [12] (in the following named *Modularity*). In addition, we compare to two representatives of graph clustering paradigms: the well-known multi-level partitioning algorithm *Metis* by Karypis and Kumar [9] and the Markov Cluster algorithm (*MCL*) by Dongen [2]. In the experiments, *Dcut*, *Ncut* and *Metis* assume the same number of clusters K for all data sets. *MCL* takes the default inflation parameter as indicated in the original paper. All experiments have been performed on a workstation with 2.0 GHz CPU and 8.0 GB RAM.

Evaluation measures: To compare different graph clustering algorithms with respect to effectiveness, we evaluate the clustering results in two ways. First, if class label information is available for the graph, the clustering performance is directly measured by three widely used evaluation measures: *Normalized Mutual Information (NMI)* [16], *Adjusted Rand Index (ARI)* [13] and *Cluster Purity*. For the graphs without ground truth, we adopt the well-known clustering coefficient proposed by Watts and Strogatz [17] as a cost function. Clustering results are measured by averaging the clustering coefficient of all subgraphs (clusters) obtained by
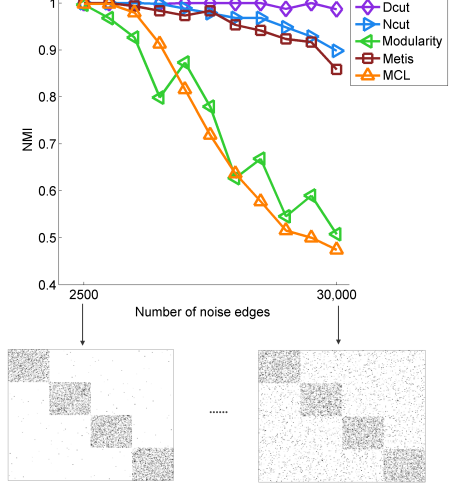

Figure 4: Varying the number of inter-cluster edges. Due to space limitations, the matrices only display 4 clusters, which is the same as in Fig. 5 and Fig. 6.

different approaches. Formally, the clustering coefficient for measuring a graph clustering is defined as follows.

$$(4.5) \qquad CC = \frac{1}{K} \sum_{i=1}^{K} \left(\frac{1}{|C_i|} \sum_{j=1}^{|C_i|} \frac{T_j}{L_j} \right),$$

where K is the number of clusters, C_i is the $i-th$ cluster, T_j is the number of triangles connected to node j, and L_j is the number of triples around node j.

4.1 Synthetic data In this section, we start with several experiments on synthetic data sets featuring various graph characteristics.

Noise Edges: First, we evaluate how well the different graph clustering algorithms can handle the additional edges in graphs, which we call noise edges. Here 20 clusters are generated, and each cluster consisting of 50 nodes are randomly interlinked with 60% intra-cluster edges. In addition to the approximately 15,000 intra-cluster edges, the number of noise edges, which are additional edges randomly added to random nodes, are present in the data varying from 2500 to 30,000. The noise in the data is represented by inter-cluster edges being added to the data, thus, introducing inter-cluster connectivity to hamper cluster separation.

With adding more noise edges into the graph data (Fig. 4), the performance of all five approaches degrades, as measured by NMI. *MCL* is only able to handle data with up to approximately 10,000 noise edges, and the performance starts to decrease dramatically as soon as more inter-edges are added. Like *MCL*, *Modularity* is sensitive to noise edges, which is indicated by large performance fluctuations. In contrast,
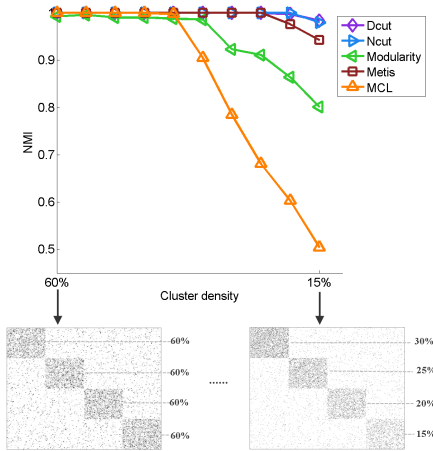
Figure 5: Varying the densities of clusters in the data.

the performances of *Dcut*, *Ncut* and *Metis* are more stable and robust to noise edges. *Dcut* starts to achieve relatively better results than *Ncut* and *Metis* for up to 20,000 noise edges.

Cluster Density: Next, we evaluate how the algorithms respond to a change of the intra-cluster edges of different clusters in the graph data, which we call cluster density. Here 10 clusters are first generated with 5000 inter-cluster edges, and 100 nodes in each cluster are randomly interlinked with 60% intra-cluster edges. We gradually change the number of intra-cluster edges for one cluster step by step with 5% decrease until all clusters have different densities of intra-connectivity (Fig. 5).

Fig. 5 shows that all algorithms perform well when the densities of clusters are above 50%. *MCL* and *Modularity* are not able to achieve convincing results when clusters with lower densities are included in the graph data. As soon as the cluster density is lower than 35%, *Metis* starts to exhibit a slightly decreasing performance. Gradually changing the density of intra-cluster edges, both *Dcut* and *Ncut* achieve relatively high clustering performance.

Network Types: Finally, we evaluate how the algorithms depend on different types of networks. Here we generate several graph data of clusters representing different types of networks with various preferential attachments: random network, small-world network and scale-free network (Fig. 6). Specifically, the first graph data including 10 clusters representing a random network is generated, and each cluster consisting of 100 nodes is randomly interlinked with intra-cluster edges from the density of 55% to 10% with a stepwise decrease of 5%. The second graph data consisting of 10 clusters exhibit small-world properties following the SW model [17]. Every node in each cluster connects to its k nearest

neighbours with rewiring probability of 20%. k ranges from 2 to 20 (step size of 2) for the different 10 clusters. Similarly, we generate the third graph data of 10 clusters representing a scale-free network [1] with various degrees of density. For each cluster of 100 nodes, the initial nodes are first generated and randomly linked, and for the next new node, m edges are added to ensure that the degree distribution of each cluster follows the power law. The value of m increases for clusters varying from 2 to 20. Moreover, graph data including clusters representing different types of networks are generated, where two clusters correspond to a random network, four clusters correspond to a small-world network and the remaining four clusters are scale-free networks. For all four graph data, 5000 inter-edges are additionally added.

From Fig. 6(a), we can observe that most approaches perform well on the random network except for *MCL*. For the small-world network, the scale-free network and the mixed network, different methods exhibit different preferences. For all graph data, *MCL* and *Modularity* are not able to achieve convincing clustering results. *Dcut* is the only algorithm which performs well for all graph data including mixed network types (Fig. 6(a)). To understand the reason behind that, we plot the corresponding DCTs for the graph data representing different network types. For random network, vertices are connected with very similar density indicated by the similar thickness of arrows. The DCT of small-world networks are chain-like, while for scale-free networks, many vertices form a hub, and the strengths of connections decrease from the center to the outside. These features of DCTs capture the intrinsic characteristics of different network types well.

4.2 Real World Data In this section, we evaluate the performances of different graph clustering algorithms on several real-world data which are all publicly available from the UCI network data repository (https://networkdata.ics.uci.edu/index.php).

(1). Networks with class information

We first investigate the networks for which the ground truth of community structure are already known. The external evaluation measures such as *NMI*, *ARI* and *purity* are applied.

American College football: The graph data derived from the American football games of the schedule of Division I during regular season Fall 2000, where 115 vertices in the graph represent teams, and edges represent regular-season games between the two teams they connect. The teams are divided into 12 conferences containing around 8-12 teams each, and thereby the real

(a) Clustering results   (b) random network (RN)   (c) SW network   (d) SF network
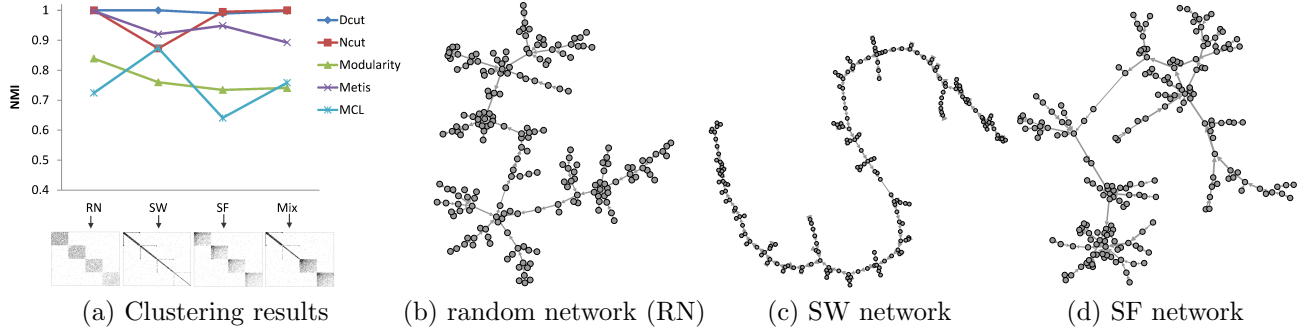
Figure 6: Clustering on graph data of clusters characterizing different network types. Here, (b)- (c) are the density-connected trees corresponding different network types. Due to space limitations, only the graph data including two clusters is illustrated.

(a) Density-connected Tree   (b) Graph partitioning with *Dcut*

AC: Atlantic Coast   MA: Mid-American
BE: Big East   MW: Mountain West
BT: Big Ten   PT: Pacific Ten
B12: Big Twelve   SE: Southeastern
CU: Conference USA   SB: Sun Belt
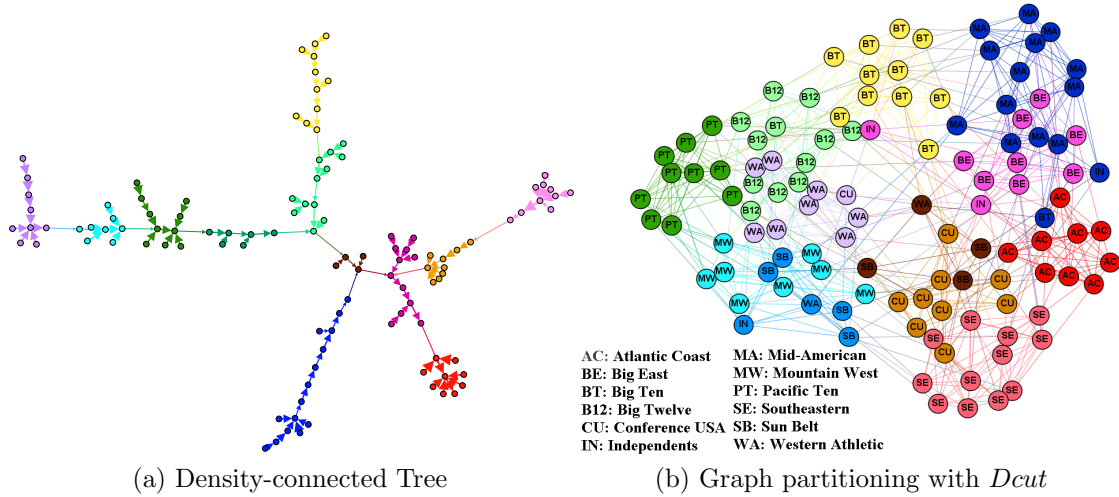IN: Independents   WA: Western Athletic

Figure 7: Performance of *Dcut* on the network of American college football, where the colors of nodes indicate different graph clusters.

community structure is already known.

Dcut ($K = 12$) identifies the graph clusters with a high degree of success (Fig. 7 and Table 1). Most teams are correctly grouped with the other teams in their conference with the highest cluster quality compared to the other four approaches (NMI = 0.924, ARI = 0.899, Purity = 93.0%). The good performance is due to the density-connected tree, where the most closely associated teams are densely connected together (Fig. 7(a)). *Ncut* and *MCL* also perform well, and most teams are correctly grouped. For *Metis* and *Modularity*, however, it is difficult to discover the community structure. The performance of the different algorithms is summarized in Table 1.

Books about US politics: The network consists of 105 nodes and 441 edges, which are derived from the books about US politics published around the time of the 2004 presidential election. Edges represent frequent copurchasing of books by the same buyers. Each book is categorized as "liberal", "neutral", or "conservative"

Table 2: Evaluation of different graph clustering algorithms with clustering coefficient on real-world data sets.

Data	Dcut	Ncut	Modularity	Metis	MCL
Coauthorship	0.1408	0.1349	0.1269	0.0984	0.1247
Power Grid	0.0309	0.0255	0.0086	0.0237	0.012

by Mark Newman based on a reading of the descriptions and reviews of the books posted on Amazon. With $K = 3$, most books can be correctly grouped by *Dcut* with $NMI = 0.572$ (Fig. 8). Two major clusters correspond to liberal and conservative books with high cluster purity (only four books are misclustered in the two clusters), respectively. The same types of books are linked together in the density-connected tree (Fig. 8(a)). Compared with other algorithms, *Dcut* achieves the best clustering results, as indicated in Fig. 8(b) and Table 1.

(a) Density-connected Tree
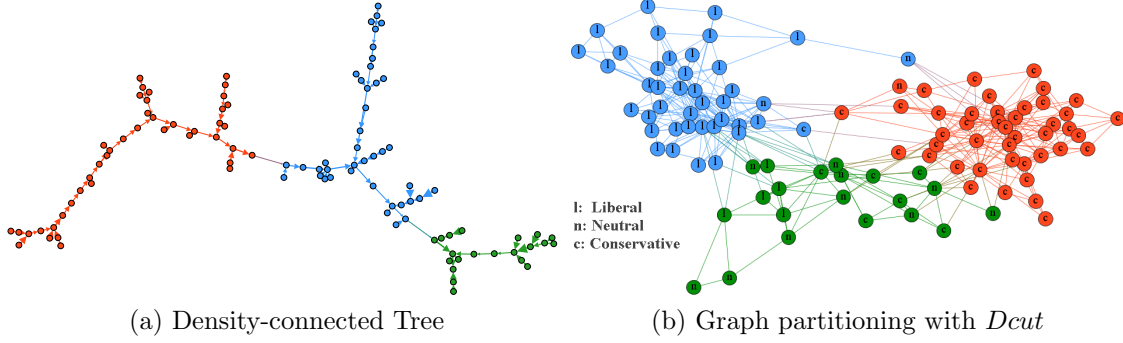
(b) Graph partitioning with *Dcut*

Figure 8: Performance of *Dcut* on the network of books about US politics.

Table 1: Performance of different graph clustering algorithms on real-world data sets.

Data	Dcut			Ncut			Modularity			Metis			MCL		
	NMI	ARI	Pur	NMI	ARI	Pur	NMI	ARI	Pur	NMI	ARI	Pur	NMI	ARI	Pur
Football	0.924	0.899	0.930	0.923	0.897	0.930	0.596	0.474	0.574	0.526	0.236	0.487	0.923	0.897	0.930
Books	0.572	0.680	0.857	0.534	0.645	0.829	0.508	0.638	0.838	0.382	0.425	0.781	0.455	0.594	0.857

Figure 9: Graph Clustering of *Dcut* on the coauthorship network of scientists (K=8).
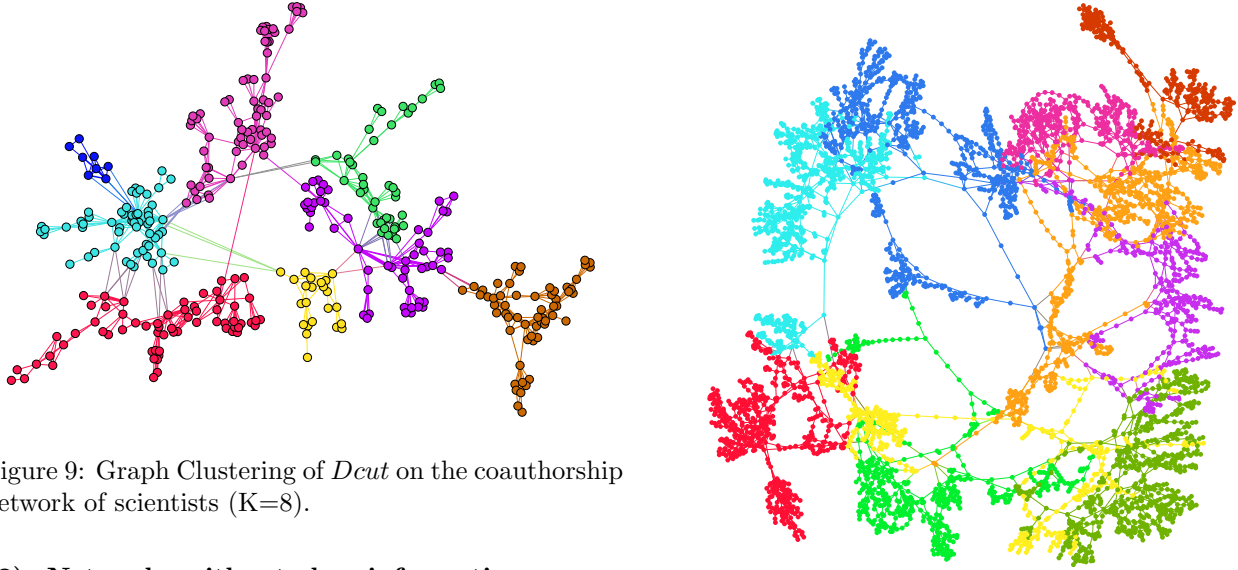

Figure 10: Graph Clustering of *Dcut* on the power grid network (K=10).

(2). Networks without class information

Coauthorships in network science: The graph is a coauthorship network of 1589 scientists working on network theory and experiment. As the vertices of the network are not all connected, only the largest component of this network is used for graph clustering in this study. The graph clusters detected by *Dcut* (K = 8) are illustrated in Fig. 9. In the plot, the obtained clusters present a high degree of scientific community structures. For comparison, the adapted clustering coefficient is applied to measure the quality of graph clusters discovered by different clustering algorithms (Table 2).

Power Grid: This network consists of 4941 vertices and 6594 edges, which represents the power grid of the Western States of the United States, compiled by Duncan Watts and Steven Strogatz. With K = 10, the clustering result of *Dcut* is depicted in Fig. 10. We can observe that the power stations in each cluster show strong connections although the graph is very sparse, which results in the highest clustering coefficient of 0.031 compared to other approaches (Table 2).

Generally, *Dcut* allows identifying a good graph clustering, and outperforms the compared algorithms on these real-world data sets, as indicated by external measures or the clustering coefficient.
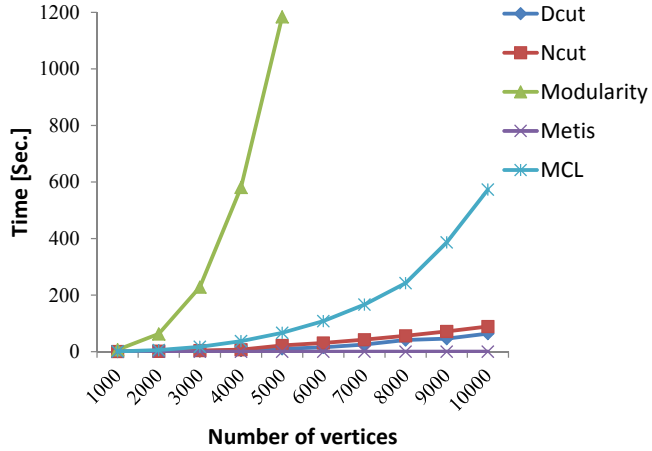
Figure 11: The runtime of the clustering algorithms.

4.3 Runtime For runtime comparisons, we generated several synthetic data sets, where the number of clusters k varied from 10 to 50, and each cluster contains 100 nodes. Approximately 30% of the intra-cluster edges were generated, and 1% inter-cluster edges were linked. To obtain more accurate runtime results, each method was run 10 times and the times were averaged. In Fig. 11, we can observe that $Dcut$ is faster than $Modularity$, while the time complexity of $Dcut$ is only super-linear. $Dcut$ is also better than MCL (approximately 8 times), and comparable to $Ncut$. Moreover, $Dcut$ is slightly slower than $Metis$ with VLSI parallel implementation.

5 Conclusions

In this paper, we introduce $Dcut$, a novel graph clustering algorithm. From a density point of view, the proposed $density\text{-}cut$ criterion offers an intuitive measure to quantify the "goodness" of a graph clustering. As the DCT provides a density connectivity map of vertices in a graph, it supports an effective way to bisect a graph directly. Our extensive experiments demonstrate that $Dcut$ has many desirable properties and outperforms several state-of-art graph clustering methods. However, one limitation of $Dcut$ is not designed for large network and does not scale well with graph size, although it is faster than $Ncut$, $Modularity$ and MCL (Fig. 11). In future work, we are planning to focus on exploring large networks containing millions or billions of nodes by combining the density-cut criterion and heuristic summarization techniques.

References

[1] R. Albert and A.L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

[2] S. Dongen. A cluster algorithm for graphs. Technical report, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, 2000.

[3] T. S. Evans. Clique Graphs and Overlapping Communities. *Journal of Statistical Mechanics*, P12037, 2010.

[4] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. Graph Clustering and Minimum Cut Trees. *Internet Mathematics*, 1(4):385–408, 2010.

[5] S. Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174, 2010.

[6] L. W. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 11(9): 1074-1085, 1992.

[7] C. Hennig and B. Hausdorf. Design of dissimilarity measures: A new dissimilarity measure between species distribution ranges. In *Data Science and Classification, Studies in Classification, Data Analysis, and Knowledge Organization, SpringerVerlag GmbH, Berlin, Germany*, pages 29–38, 2006.

[8] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *J. Parallel Distrib. Comput.*, 48(1): 96–129, 1998.

[9] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359–392, 1999.

[10] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002.

[11] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69(6):066133, 2004.

[12] M. E. J. Newman. Modularity and community structure in networks. *PNAS*, 103(23):8577–8582, 2006.

[13] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66 (336): 846–850, 1971.

[14] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27 – 64, 2007.

[15] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on PAMI*, 22:888–905, 1997.

[16] A. Strehl and J. Ghosh. Cluster ensembles–a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, 2003.

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

[18] Z. Wu and R. Leahy. An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation. *IEEE Trans. PAMI*, 15(11):1101–1113, 1993.

[19] W. Zachary. An information flow model for conflict and fission in small groups. *J. of Anthropological Research*, 33:452–473, 1977.