Clustering in the Presence of Bridge-Nodes

Jerry Scripps
Computer Science and Engineering
Michigan State University
scripps@msu.edu

Pang-Ning Tan
Computer Science and Engineering
Michigan State University
ptan@msu.edu

Abstract

In this paper, we study the ill-effects of bridge-nodes, which causes many dissimilar objects to be placed together in the same cluster by existing clustering algorithms. We offer two new metrics for measuring how well a clustering algorithm handles the presence of bridge-nodes. We also illustrate how algorithms that produce overlapping clusters help to alleviate the effect of bridge-nodes and form more meaningful clusters. However, if there is too much overlap, the clusters become less informative. To address this problem, we present a novel clustering algorithm called MIN-CUT. Our experimental results with real data sets show that the MIN-CUT algorithm leads to purer clusters that have very little overlap.

1. Introduction

Clustering is a broad field of practice and study. The data set used for clustering can often be represented as a graph, where the nodes are objects to be clustered and the edges represent relationships among the objects. In this paper, we investigate a phenomenon known as *bridge-nodes*, which can pose significant problems for clustering algorithms. A bridge-node is a node that is very similar to two or more other nodes, which are not very similar to each other. Bridge-nodes appear often as in the following examples:

- In social networks one person may be close friends with two other people who hardly know or dislike each other.
- In a bibliographic database there are many prolific authors who have co-authored papers with other authors who have no common interests.
- In text documents, homonyms such as mint can be associated with disparate words such as candy and treasury.
- In a database of movie ratings, two reviewers may not share the same interest in movies but may have common interest with a third reviewer.

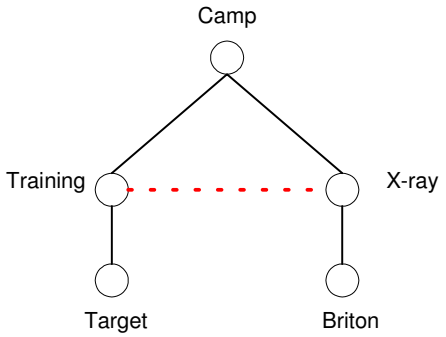

Figure 1: Bridge-node example.

Figure 1 shows a portion of a graph of words extracted from news stories that appeared in the Daily Mirror newspaper between 2001 and 2003. A solid line represents strong similarity between a pair of words and a dotted line represents strong dissimilarity. The figure suggests that there is a strong association between the word pairs (X-Ray, camp), (training, camp), (training, target), and (X-Ray, briton), but a very weak connection between (training, X-Ray). (Many other words are, of course, connected to these five words but we only show these for the sake of simplicity.)

The relationships observed in the graph can be explained in the following way. During 2002, there were numerous articles about British citizens detained at Camp X-Ray, a prisoner of war camp in Guantanamo Bay, Cuba. There were also many stories about the terrorist training camps that were potential military targets. However, the articles about Camp X-Ray do not mention the word training, which explains the lack of association between the two words. The word camp in this case acts as a bridge-node between training and X-ray.

Clustering algorithms differ in terms of how they handle bridge-nodes. Most clustering algorithms would either put camp, X-ray, and training all in the same cluster or put X-ray and training in two separate

clusters and then place camp in one of those two clusters. The obvious problem with grouping them altogether is that the cluster combines words from two distinct stories. On the other hand, putting camp in one of the two clusters leads to incomplete description of clusters. For example, suppose the word camp is placed in the cluster with training. Analysts looking at the resulting clusters might mistakenly conclude that there was a story about a British citizen given an X-ray. Whenever a clustering algorithm must place a bridge-node in one of many deserving clusters the ones not chosen will be less descriptive.

In this paper we will:

1. define and discuss the phenomenon of bridge-nodes,
2. introduce some new metrics helpful in examining the effect of bridge-nodes on existing clustering algorithms, and
3. present a new algorithm that specifically address the problems inherent with bridge-nodes.

The remainder of the paper is organized as follows. In Section 2 we define the concept of bridge-nodes. Section 3 examines the effect of bridge-nodes on existing clustering algorithms, while Section 4 lays the groundwork for a new algorithm. Section 5 discusses our proposed algorithm and Section 6 shows the results of our experiments. A summary of the related work is given in Section 7. Section 8 concludes with a discussion of future work.

2. Preliminaries

This section presents our graphical representation of data and introduces the concept of bridge-node.

2.1 Graph Representation

Let $G = \{V, E\}$ be a graph, where V is the set of vertices (nodes) and E is the set of edges. Note that all of the graphs discussed in this paper are assumed to be undirected. A data set can be transformed into a graph representation, where the vertices correspond to objects and the edges correspond to relationships among objects. The edges can have weights to represent the similarity (distance) between two data objects.

A weighted graph can be transformed into an unweighted graph – based on thresholds or constraints imposed by users. The example below demonstrates the transformation process using thresholds. During the transformation, edges whose weights are above the upper threshold are converted into must-link or ML edges (solid lines); those below the lower threshold are transformed into cannot-link or CL edges (dotted lines). The weights that fall between the two thresholds

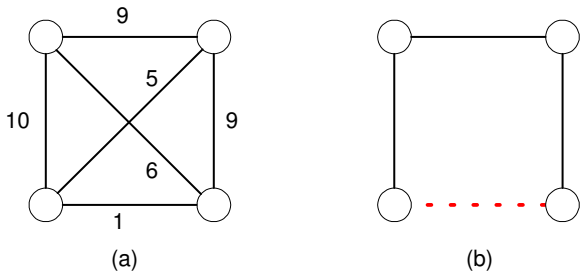

Figure 2: Transition from weights to constraints

are can-link edges and are represented by no line between the nodes.

EXAMPLE 1. *Figure 2(a) shows an example of a weighted graph. The graph on the right is transformed from the graph on the left by using an upper threshold of 8 and a lower threshold of 2. The solid lines are the must-links (above 8) and the dotted line is a cannot-link (below 2). The diagonal edges that were removed are can-links. Nodes connected by a must-link should be clustered together while the ones connected by a cannot-link must not be clustered together.*

2.2 Bridge-path and Bridge-node

A bridge-path is a simple path of ML edges between two terminal nodes that are connected by a CL edge. Figure 3(a) shows an example of a bridge-path between two terminal nodes v_1 and v_3, while Figure 3(b) shows another bridge-path connecting the terminal nodes v_4 and v_7.

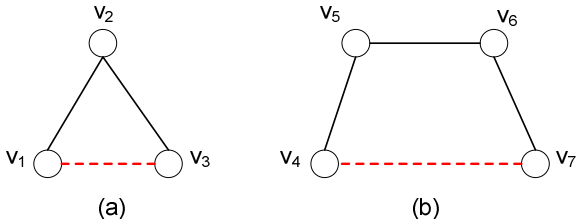

Figure 3: Bridge-node and Bridge-path

All intermediate (non-terminal) nodes located along the bridge-path can be considered as bridge-nodes. For example, v_2 is a bridge-node for the bridge-path shown in Figure 3(a). The nodes v_5 and v_6 along the bridge-path in Figure 3(b) are also considered to be bridge-nodes. In general, a node might be considered a bridge-node with respect to given CL edge but a non-bridge-node with respect to another CL edge.

3. Effect of Bridge-Nodes on Clustering

We will now use our definition of bridge-node to help us evaluate some typical clustering algorithms. Assume that we have a data set from which we create a transformed graph $G = (V, E)$, where $E = ML \cup CL$. Let C: $V \rightarrow \{1, 2, \ldots, k\}$ be a clustering function that maps each node $u \in V$ into a positive integer that represents the cluster id. To measure the effectiveness of an algorithm in terms of handling bridge-nodes, we first introduce the following terminology.

DEFINITION 1 [*Incomplete Edge*]: *An edge $(u,v) \in E$ is incomplete if $(u,v) \in ML$ and $C(u) \neq C(v)$.*

DEFINITION 2 [*Impure Edge*]: *An edge $(u,v) \in E$ is impure if $(u,v) \in CL$ and $C(u) = C(v)$.*

3.1 Metrics for Evaluating Effects of Bridge-Nodes

Our concern in this paper is bridge-nodes and the effects that they have on the clustering process. Bridge-nodes either bring together two dissimilar nodes or place two highly similar nodes in different clusters. We introduce the following two metrics to measure these effects.

DEFINITION 3 [*Incompleteness*]: Incompleteness is a cluster evaluation metric that measures the ratio of incomplete edges to the total number of ML edges.

DEFINITION 4 [*Impurity*]: Impurity is a cluster evaluation metric that measures the ratio of impure edges to the total number of CL edges.

In a simplistic data set where we have k well-separated groups of objects and we want k clusters, we would expect to find a clustering with zero incompleteness and zero impurity. Most data sets, though, have boundaries that are not so well defined.

Davidson and Ravi [6] have recently shown that determining whether a data set with CL constraints has a feasible solution is an intractable problem. Building upon their work, the metrics that we developed explicitly measure the amount of ML and CL constraint violations in the clustering produced by an algorithm.

In Figure 3(b), we see that if v_5 and v_6 cause v_4 and v_7 to be in the same cluster the number of impure edges increases by one. On the other hand if the CL link between v_4 and v_7 leads to breaking the nodes into two clusters then whichever ML edge is broken becomes an incomplete edge. In either case either the impurity or the incompleteness will increase.

In Section 3.2, we perform experiments using standard clustering algorithms to show a trade-off in cluster quality regarding incompleteness and impurity. Algorithms that do well grouping highly similar nodes together will probably fair poorly in keeping low-similarity nodes separate and vice versa. Some may do mediocre at both but we would not expect any to do well at both.

3.2 Evaluating Clusters Produced By Current Algorithms

To evaluate the effectiveness of existing algorithms, we compiled a database of news stories downloaded from the Daily Mirror newspaper using the Infotrac database. For the results presented below we extracted news articles about terror from July 2001 through June of 2003. A total of 6,303 stories were used. In addition to this data set, we also tested the algorithms on the Reuters-21578 and the 20 Newsgroups data sets, both available at the UCI KDD archive. For the newsgroups data we used the alt.atheism newsgroup rather than the entire set in order to get a more homogenous data set. Homogenous data sets are generally more difficult to cluster since they have more similarities than differences.

For each of the data sets we extracted a subset of 500 words. First we stemmed the words and then eliminated the common words. For the terror data set we used a database of non-terror news stories to identify the common words. We extracted the 500 words that occur most frequently in the terror stories but not in the non-terror ones. For the other data sets we simply took the top 500 most frequent, non-common words. We then build a similarity matrix that contains the number of times a pair of words occurred together within the same document.

Our experiments were performed using k-means and three agglomerative hierarchical clustering algorithms (complete-link, single-link and group-average) [10]. The number of clusters was varied from 20 to 500. To identify the ML and CL edges, we used thresholds based on the top 1% and the bottom 1% of the similarity values.

While it is possible to define the ML and CL edges based on other criteria beside the similarity measure (e.g., using domain knowledge), the algorithms considered in this section (k-means and hierarchical clustering) do not utilize the information about ML and CL edges when clustering the data. Instead they rely on a similarity or distance measure to determine which data points should be grouped together. Unless the ML

and CL edges are consistent with the similarity or distance measure, we might not be able to draw any definitive conclusions about the effectiveness of different clustering algorithms. This explains the rationale behind our approach for defining the ML and CL edges in this section. It is not necessary to define the ML and CL edges this way, though, for clustering algorithms that work directly with the ML and CL edges, such as the ones described in Section 5.

Our intention here is to show the tradeoff between the impurity and incompleteness metrics for different clustering algorithms when bridge-nodes are present in the data. Figure 4 shows the results for the newsgroup data. When the number of clusters is large the nodes that participate in CL edges can be easily separated into different clusters so all the algorithms have low impurity. As the number of clusters decreases we see that the impurity score for single-link increases markedly. This is because single-link computes the similarity between two clusters based on the similarity of the two closest data points that are in different clusters. As a result, it tends to place all the nodes that participate in a bridge-path into the same cluster including the endpoints that participate in the CL, thus increasing its impurity score. k-Means also creates impure clusters but not as many as single-link. This is because as the number of clusters gets smaller it is more likely to place an entire bridge-path including the endpoints in the same cluster. In contrast, complete-link and group-average still produce pure clusters even at 20 clusters. Complete-link determines the similarity between two clusters based on the similarity between the two furthest data points, whereas group-average uses the average similarities of all pairs of points from different clusters. Both approaches are therefore less likely to create impure CL edges than single-link.

Figure 4 also shows that with high numbers of clusters all of the algorithms produce clusters with high

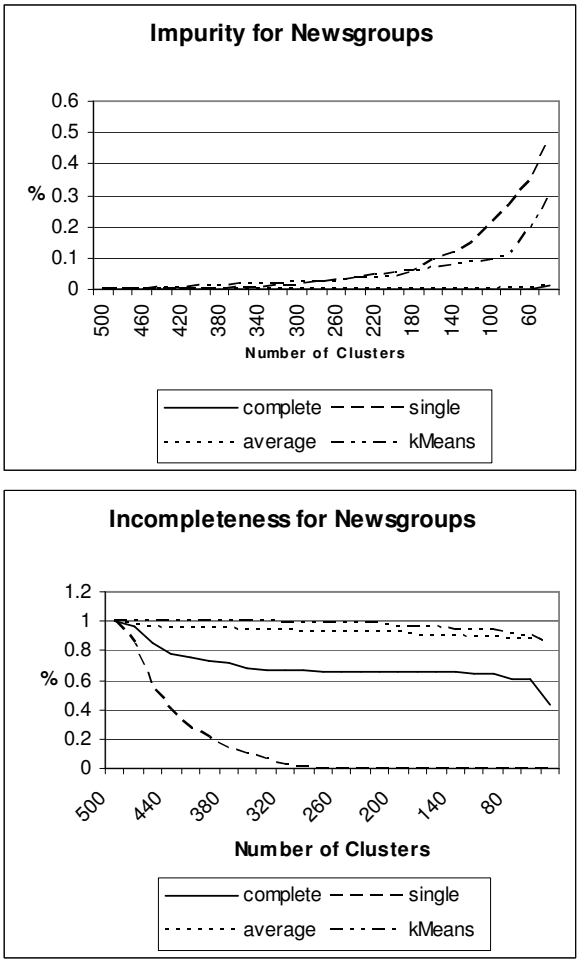

Figure 4: Impurity and Incompleteness for Newsgroup data.

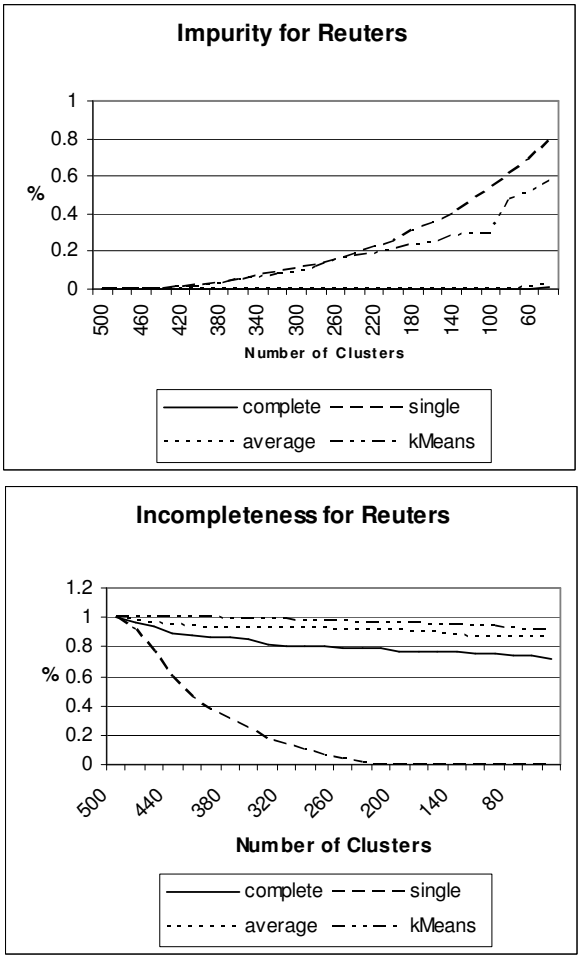

Figure 5: Impurity and Incompleteness for Reuters data.

incompleteness scores. Again this was not unexpected because with only 1 or 2 nodes per cluster many of the highly similar nodes that participate in the ML edges would not be found in the same cluster. As the number of clusters decreases, complete-link and group-average have considerably worse incompleteness scores than single-link because they tend to break the bridge-paths that are present in the data. For example, using complete-link the following word pairs that should have been clustered together were not (even when the ML threshold was raised from top 1% to top 0.2%): (afghan, taliban), (blair, minister), (iraq, war), and (laden, saudi). All of these pairs participate in a bridge-path triangle (Figure 3a). For example, minister is a bridge-node between blair and sharon.

For the most part algorithms that had relatively pure clusters also had high incompleteness scores, while those that had relatively complete clusters tend to be impure. This suggests a trade-off between the two evaluation metrics. The results for the other data sets in Figures 5 and 6 agree with our observations for the newsgroup data. The only other observation we can make besides the trade-off is that k-Means does not perform as well as the others – that is it performs worse than at least one other algorithm in both measures in all three data sets. Finally note that while there exist algorithms that explicitly deal with constraints [2,4,6,7], even these algorithms will need to violate some constraints in order to form clusters. Therefore there will still be a tradeoff between impurity and incompleteness scores among these algorithms.

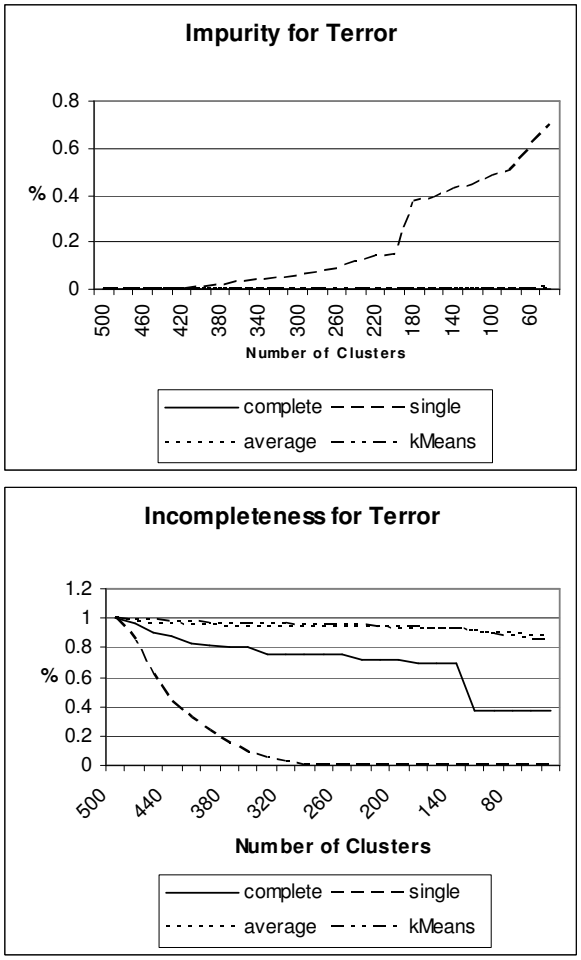

Figure 6: Impurity and Incompleteness for Terror news data.

4. Methodology for Handling Bridge-Nodes

In this section we explore how we can reduce the ill-effects of bridge-nodes by node cloning to produce overlapping clusters.

4.1 Cloning

Consider a graph $G = (V, E)$ where each node $u \in V$ has a unique identifier. For example, in figure 7(a) v_1, v_2, v_3, are the identifiers. Let $G' = (V', E)$ be a cloned graph of G where $V' \supset V$ and the identifiers assigned to the nodes in V' may not be unique.

DEFINITION 5 [*Cloned Node*]: *A node $u \in V'$ is a cloned node if $\exists\, v \in V'$ such that u and v have the same identifiers and $u \neq v$.*

DEFINITION 6 [*Cloned Set*]: *A cloned set for node w, S(w), is defined as the set of all nodes (including w) with the same identifier as w.*

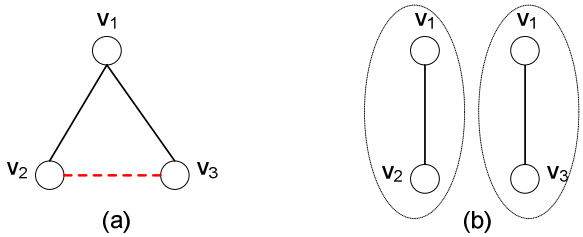

Figure 7: Node cloning

Cloning nodes, that is, creating multiple copies of a node will be helpful for reducing impurity and incompleteness. A simple example helps to illustrate. Given three nodes v_1, v_2 and v_3 (Figure 7), where there is a ML between v_1 and v_2 and between v_1 and v_3 but a CL between v_2 and v_3 is there any way to put them into 2 clusters without raising the impurity or

incompleteness? In the example above if we can clone node v_1 so that we create a cluster with v_1 and v_2 and another with v_1 and v_3 our clusters would be pure because v_2 and v_3 does not belong to the same cluster. Our definition of impurity can remain intact.

Our old definition of incompleteness, though, no longer applies. Because the vertex v_2 in the first cluster is linked to v_1 in the same cluster, the cloned vertex v_1 in the second cluster is not placed in the same cluster as v_2. We argue that the incompleteness score for this situation is zero because the ML constraint is still preserved in one of the clusters.

When cloning is allowed, we need to redefine an incomplete edge as follows:

DEFINITION 1B [*Incomplete Edge*]: *An edge $(u,v) \in E$ is incomplete if $(u,v) \in ML$, $\forall y \in S(u)$, $\forall w \in S(v)$: $C(y) \neq C(w)$, where $S(v)$ and $S(u)$ are the cloned sets for u and v respectively.*

To support this definition we give the following intuitive example. Suppose v_1, v_2 and v_3 in Figure 7 correspond to the words union, soviet, credit, respectively. Assume also that (soviet, union) and (credit, union) are both ML edges and (soviet, credit) is a CL edge, which means that union is a bridge-node. A reasonable clustering will have soviet and union together in the same cluster and credit and union together in another cluster. This means that every cluster that contains soviet must also contain union but not the other way around – that union can belong to some clusters that do not contain soviet. In terms of Figure 7, v_1 must appear in every cluster that contains v_2 and every cluster that contains v_3 but v_2 and v_3 do not have to belong to every cluster that contains v_1 (since v_1 is the cloned node).

The above definition suggests that if a node does not have a clone, it must belong to the same cluster as other nodes that have a ML relationship to it. If the node has a clone, then it must appear at least once in a cluster with every node with which it has a ML relationship.

4.2 Measure of Overlap

With cloning nodes, there is a danger of cloning too much. Considering the bridge-node in Figure 7 it is informative to clone the node v_1 but if the other nodes are cloned as well we could end up with clusters with many overlapping nodes which would be disastrous. So we could use another measure to help us determine whether the overlap is acceptable. With the overlap measure we are only concerned with nodes that appear in more than one cluster.

DEFINITION 7 [*Additional Nodes*]: *Let $G'=(V',E)$ be a cloned graph of $G=(V,E)$. The number of additional nodes created by cloning is $|S(w)|-1$ for all $w \in V$ where $S(w)$ is the cloned set of w in V'.*

DEFINITION 8 *[Overlap]: The overlap measure is defined as the ratio of additional nodes to the total number of nodes in the cloned graph.*

Note that this measure is zero when there is no overlap and approaches but never reaches one when there is excessive overlap.

4.3 Measuring Effectiveness of Handling Bridge-Nodes

Consider again the diagram shown in Figure 7(a). We have three rational choices:

1. we can cluster v_1 and v_2 together or v_1 and v_3 together and put the other node in a separate cluster,
2. we can cluster all three node together, or
3. we can clone v_1 and put it in two clusters one with v_2 and the other with v_3.

Depending on which choice was selected we would add one to either the number of (1) incomplete edges, (2) impure edges, or (3) additional (cloned) nodes. While the trade-off among the choices may seem the same, we argue that having an extra clone node is still much better than having an incomplete or impure edge since the clone node provides us with a better interpretation of the clusters (unless the number of clone nodes is too large). This argument holds only if we do not make mistakes such as cloning irrelevant nodes that are not part of a bridge-path.

A clustering algorithm that handles bridge-nodes should minimize the following objective function:

$$Q = \lambda_1 |E_{\text{incomplete}}| + \lambda_2 |E_{\text{impure}}| + \lambda_3 |V_{\text{additional}}|$$

where λ_1, λ_2, and λ_3 are the cost of having an incomplete edge, an impure edge or an additional node. Based on our argument above the cost values are assigned in such a way that $\lambda_1 >> \lambda_3$ and $\lambda_2 >> \lambda_3$. Therefore in this paper we focus only on algorithms that produce clusters with zero impurity, zero incompleteness, and minimal amount of overlap. Developing an algorithm that minimizes the objective function for other costs is a subject for future work.

4.4 Selecting Bridge Nodes for Cloning

Finding the minimum number of bridge nodes to clone is not a trivial task. For the simple graph shown in

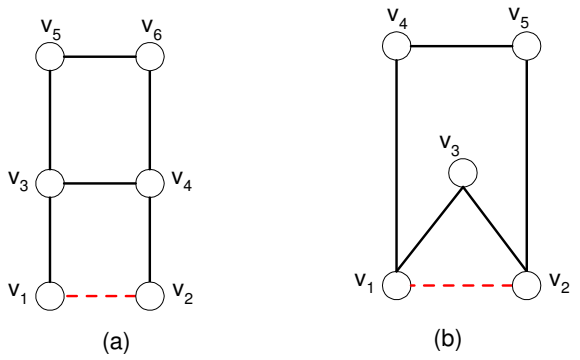

Figure 8: Selecting the nodes to clone

Figure 7(a), it is clear that the bridge node v_1 must be cloned to minimize the objective function. However, when there are multiple bridge nodes or bridge paths available, the situation becomes more complicated.

Consider the diagram shown in Figure 8(a). In this case, there are two bridge paths between the nodes v_1 and v_2. It is sufficient to clone one of the nodes, v_3 or v_4, to produce two clusters with zero impurity, zero incompleteness, and minimal amount of overlap. However, for the diagram shown in Figure 8(b), one has to clone at least two nodes to ensure that the impurity measure of the resulting clustering is zero.

It is also tempting to estimate the minimum number of nodes to clone as the sum of bridge-paths for each CL pair but some paths may have common nodes. Looking at Figure 9 we see that there are 3 CL links, GH, IJ and KL. Under the naïve assumption above that would mean we would have to clone 3 nodes. We can tell though, by looking at the figure that by cloning nodes D and B we could have 3 clusters {G,A,B,I}, {B,C,H,D,E,F,L}, and {J,D,K} with zero impurity, zero incompleteness, and two additional

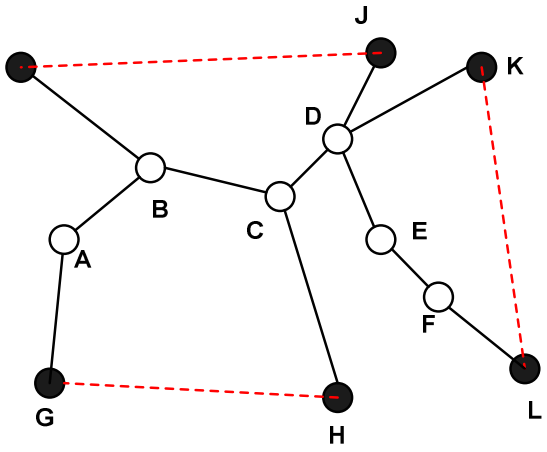

Figure 9: Graph with multiple CL edges.

nodes.

Therefore selecting the appropriate bridge nodes to clone is not a trivial problem. We refer to this problem as the node cloning problem. Theorem 1 shows that finding the minimum number of nodes to clone is an NP-Complete problem.

HITTING SET [9]
Instance: Collection C of subsets of a finite set S, positive integer $K \leq |S|$.
Question: Is there a subset $S' \subseteq S$ with $|S'| \leq K$ such that S' contains at least one element from each subset of C?

THEOREM 1. *Within a graph G formed by ML and CL edges, finding the minimum number of nodes to clone so that there are no ML paths connecting any two nodes in a CL link is NP-Complete.*

PROOF: It can be easily seen that given a non-deterministic algorithm that found potential solutions we could validate the solutions in polynomial time – so the problem is in NP. To prove that it is NP-Complete we will reduce HITTING SET to the node cloning problem. Given an instance (C,S,K) of HITTING SET we will transform it into a graph G in the following way. For each subset $c \in C$ we create an ML path in G using the elements in c as the nodes along the path. Each ML path is then converted into a bridge-path by adding two terminal nodes at both ends of the ML path with a CL link between them. Finding a number <= K of nodes that when cloned (cut) will separate ML paths between each CL pair in G is equivalent to finding an S' that contains at least one element from each subset of C. Note that the reduction from HITTING SET can be done in polynomial time. From the reduction it is clear that finding the minimum number of nodes to clone is at least as hard as HITTING SET. ■

As an example of the transformation in the proof, the collection C={ {A,B,C}, {B,C,D}, {D,E,F} } can be transformed into the graph in Figure 9.

5. Toward a New Algorithm

In this section we examine several node cloning strategies to handle the clustering with bridge-node problem.

5.1 NAIVE

First we consider a naïve approach where every node is cloned for each edge attached to it. The result is every ML constraint defines a 2-node cluster and any node not in a ML relationship is in a singleton cluster.

Table 1: MIN-CUT Algorithm

Input: Graph G = (V, ML, CL)
Output: Set of clusters, C

1. Ψ = { };
2. **for each** (u,v) ∈ CL
3. π = mincut(u, v);
4. Ψ = Ψ ∪ π;
5. **end**
6. C = { };
7. **while** ML is not empty
8. Select an edge (u,v) ∈ ML
9. D = {u}; D' = { };
10. **for each** (x,y) ∈ ML where (x ∈ D and y ∉ D)
11. **if** (x,y) ∈ Ψ
12. D' = D' ∪ {y};
13. Remove (x,y) and (y,x) from ML
14. **else if** (y,x) ∈ Ψ
15. Remove (y,x) from ML
16. **else**
17. add y to D
18. Remove (x,y) and (y,x) from ML
19. **end**
20. **end**
21. D = D ∪ D'
22. Remove all edges (x,y) and (y,x) ∈ ML where x ∈ D and y ∈ D.
23. C = C ∪ D;
24. **end**

Although this approach guarantees incompleteness and impurity of zero it may result in an extremely large number of clusters. For example, consider a 500 word clustering where we use the top 5% of the links for the must-link threshold. This would be approximately 6,275 ML constraints, which means there would be at least that many initial clusters. We may consider using an agglomerative clustering step to merge some of these initial clusters as long as no CL constraints are violated. During the agglomerative step, we will drop extra copies of clones in merged clusters. Despite the merging of clusters, the resulting clusters may still be poor due to the over use of clones.

5.2 MIN-CUT

Our goal is to find an algorithm that uses cloning to keep the ML nodes in the same cluster and separate the CL nodes into different clusters. Another way to look at the problem is that we need to ensure there are no paths from any node in a CL link to the other node in that link.

Before describing our algorithm we first describe a min-cut. In a graph G=(V,E), a cut between two nodes u and v are the edges which, when removed, separate the graph so that u and v are in different sub-graphs. A min-cut is a cut using the minimum number of edges.

Our MIN-CUT algorithm attempts to minimize the number of clonings by finding the min-cut vertices for each pair of CL nodes and then cloning those vertices. We used the Ford-Fulkerson [6] algorithm to find the min-cut for each pair of CL edges. Since the algorithm returns min-cut edges, we have to choose one of the two nodes in each min-cut edge for cloning.

Figure 10 shows an example of a graph with ML and CL edges. A min-cut for the CL edge Y could be edges A, I and D, while the min-cut for the CL edge X would need to be either L or N. For the min-cut edge I, we may either clone node 5 or node 10 because they are both part of the bridge-nodes for Y. On the other hand, for the min-cut edges A, D, and L (or N), we need to be careful about choosing which node to clone because one of the two vertices in the min-cut edge is part of the CL edge. In this example, the min-cut vertices 0, 1, and 7 (or 9) should not be cloned because they are not bridge-nodes (see the discussion in Section 4.1).

Table 1 summarizes the overall structure of the MIN-CUT algorithm. The algorithm consists of 2 stages: (1) clone identification stage (Steps 2-5), and (2) cluster generation stage (Steps 7-24). The clone identification stage determines the set of edges π that form the min-cut between the pair of nodes, (u, v), associated with a given CL edge. For each min-cut edge, (x,y) ∈ π, the second node y will be cloned during the cluster generation stage. At the end of Step 5, the set of nodes to be cloned (with respect to a particular edge) is stored in Ψ.

For this algorithm to work we need to ensure that for an edge in Ψ we will clone only one of the nodes (the second one). To do this we allow CL to contain

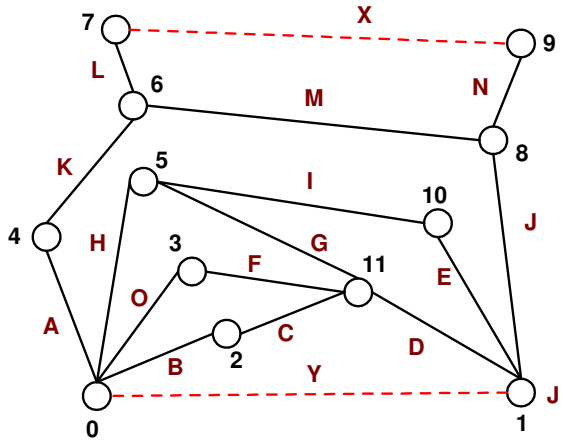

Figure 10: Finding the Min-Cut

undirected edges but ML must contain directed edges. For each must link edge that connects nodes x and y we will place both (x,y) and (y,x) in ML. Ψ will also contain directed edges but only one edge for each pair – either (x,y) or (y,x) – depending on which node is to be cloned.

The cluster generation stage forms an initial cluster D from a pair of nodes (u,v) associated with one of the must-links. The algorithm then grows the cluster D by examining all the must-links involving one of the nodes that belong to the current cluster (Steps 10-20). Given an edge (x,y) $\in$ ML, there are several possibilities to consider:

1. Both x and y do not belong to D. In this case, the edge (x,y) is ignored.
2. Both x and y belong to D. In this case, the edge (x,y) is removed in Step 22.
3. Node x belongs to D but not node y and (x,y) $\in \Psi$. Then y is a cloned node to be added to the cluster that contains x. Therefore, y is added to the cluster D'. We do not append D' to D until the entire search is exhausted so that neighbors of the cloned node y are not added to the cluster.
4. Node x belongs to D but not node y and (y,x) $\in \Psi$. Such an edge is ignored when processing node x. Both x and y will be included when we process the ML link (y,x).
5. Node x belongs to D but not node y and (x,y) $\notin \Psi$ and (y,x) $\notin \Psi$, then y is added to the current cluster D without worrying about violating any CL constraint.

When faced with several equally suitable nodes to clone the algorithm choose one somewhat randomly. Improvements could be made to break the paths in a way that provides more cohesive clusters but our concern is optimizing the clustering for impurity, incompleteness and overlap.

PROOF OF CORRECTNESS:

Theorem 4 proves that the algorithm puts every ML pair of nodes in at least one cluster together (so that incompleteness = 0) and theorem 5 proves that no CL pair is ever clustered together (so that impurity = 0).

THEOREM 4. *All nodes in every ML will be clustered together at least once.*

PROOF: *Every iteration of the loop that starts at step 7 assigns nodes to a new cluster D. Within the loop the edges (x,y) and (y,x) are removed from ML only when both x and y are in the same cluster (either D or D'). A single edge (x,y) can be removed from ML if (x,y) is also in Ψ but the other edge (y,x) remains. If Ψ contains (x,y) it cannot contain (y,x) so that when the ML edge (y,x) is processed both y and x will be added to that cluster.*■

THEOREM 5. *All nodes in every CL will be placed in separate clusters.*

PROOF(by contradiction): *Assume there is an edge $(x_0, x_n) \in CL$ such that $x_0 \in D$ and $x_n \in D$ then there must be at least one path $x_0, x_1, \ldots x_n$ such that $x_i \in D$ for $0 \le i \le n$ and $(x_i, x_{i+1}) \in ML$ for $0 \le i < n$. For any on the paths, the mincut step would ensure that for $0 \le i < n$ one of the edges (x_i, x_{i+1}) or (x_{i+1}, x_i) would be in Ψ. Without loss of generality lets assume that $(x_i, x_{i+1}) \in \Psi$. This would make the condition in step 11 true so that x_{i+1} is added to D'. Since x_{i+1} is not added to D within the loop, it is impossible for x_{i+2} to be added to D or D' which is a contradiction.*■

We illustrate the algorithm with the following example.

EXAMPLE 2: *Consider the graph shown in Figure 10. There are 12 vertices, 15 ML edges, and 2 CL edges in the graph. During the clone identification stage, the following edges are identified as min-cut edges:*

Ψ = {(0,4,A), (5,10,I), (1,11,D), (7,6,L)};

Suppose the ML edges are sorted in lexicographic order. During the cluster generation stage, suppose (0,4), is the first ML edge selected. Therefore, cluster D = {0} and D'={} (at Step 9). The algorithm then iteratively adds node 4 to D', nodes 2, 11, 3 and 5 to D and then 10 to D'. The first cluster is {0,2,3,4,5,10,11}. Links F and G are removed after the loop. Next the algorithm selects the edge (1,11) to create a new cluster. The cluster sets D and D' are initialized to {1} and {}, respectively. The algorithm then iteratively adds nodes 11, 10, 8, 6, 4 and 9 to D, so that the next cluster found is {1,4,6,8,9 and 11}. The only remaining edge is L, so the last cluster found by the algorithm is {6,7}.

5.3 Bounds for number of clusters

The number of initial clusters for both the NAÏVE and the MINCUT algorithm are bounded by the number of ML and CL edges. The NAÏVE algorithm will create |ML| + (n-|cloned nodes|) initial clusters with a maximum cluster size of 2. ML is controlled by the upper threshold. If the upper threshold is set lower than any link then essentially all of the links are must-link and ML=n*(n-1)/2. During the agglomeration the number of clusters will monotonically decrease to 1 while the cluster sizes will increase to n.

The number of clusters in MINCUT is more difficult to predict but will never get larger than n. In the extreme case where both the upper and lower thresholds are lower than the lowest link the algorithm will produce n singleton clusters. When they are both above the highest link it will produce one cluster of size n.

6. Experimental Evaluation

The overlapping clustering algorithms that we will test here should both be able to return clusters that are complete and pure. Of concern though is the number of clones that are generated by the two methods.

We compared the MIN-CUT and NAÏVE algorithms to the agglomerative, hierarchical algorithms using complete-link, single-link and group average. Note that the number of clusters found by the MIN-CUT algorithm depends only on the ML and CL edges present in the graph. This is somewhat different than standard clustering algorithms such as k-means in which the users have to specify the number of clusters they desire.

	Impurity	Incomplete	Overlap
threshold of .5%, 6 clusters			
Complete	0.016467	0.128411	0
Single	0.865269	0	0
Average	0.140719	0.727127	0
NAÏVE	0	0	0.753231
MINCUT	0	0	0.015842
threshold of 1%, 17 clusters			
Complete	0	0.359262	0
Single	0.779433	0	0
Average	0	0.809944	0
NAÏVE	0	0	0.880859
MINCUT	0	0	0.06705
threshold of 1.5%, 28 clusters			
Complete	0	0.451337	0
Single	0.735013	0	0
Average	0.02563	0.863102	0
NAÏVE	0	0	0.928768
MINCUT	0	0	0.108856
threshold of 2%, 42 clusters			
Complete	0	0.502408	0
Single	0.656777	0	0
Average	0.000733	0.897673	0
NAÏVE	0	0	0.953023
MINCUT	0	0	0.164336

Table 2: Metrics for 5 algorithms at various thresholds and number of clusters.

For our experiments we identified the ML and CL edges using the thresholds of .5, 1, 1.5 and 2 percent of the total links. This means that for the .5% the upper threshold was set to include the top .5% of the links and the lower threshold was set to include the bottom .5%. Once we established the number of clusters obtained by MIN-CUT we ran the other algorithms for that specific number of clusters.

Table 2 summarizes the results of our experiment for the terror data set. The number of clusters found by the MIN-CUT algorithm (when the threshold is varied from 0.5% to 2%) ranges from 6 to 42 clusters. The results for agglomerative hierarchical clustering agree with the analysis given in Section 3.2. Complete-link produces clusters with low impurity and zero overlap but may break some of the ML edges present in the original graph. In contrast, single-link has zero incompleteness and overlap scores, but it can produce impure clusters especially when the number of clusters is small.

As expected, the NAÏVE and MIN-CUT algorithms produce clusters with zero impurity and incompleteness scores. These algorithms are therefore more effective in terms of handling bridge-nodes due to ML and CL constraints. Furthermore, the MIN-CUT algorithm produces far fewer clones than the NAÏVE method. The number of clones also decreases with decreasing number of clusters.

7. Related Work

Overlapping clustering methods are not new. Shepard and Arabie[13] introduced Adclus in 1979. Their algorithm is based on grouping objects together that have common properties where an object can belong to more than one cluster. The concept of bridge-nodes is not really addressed by this algorithm since two objects having properties in common with a third object might get grouped together in spite of dissimilarities with each other.

Fuzzy clustering[3] creates overlapping clusters using the concept that every node belongs to every cluster with a certain weight (the weight for each node must sum to one). Because every node is in every cluster bridge-nodes are treated the same as non-bridge-nodes.

The advances in semi-supervised clustering[2, 4, 6, 7] in recent years have provided us with a good framework to start our discussion about bridge-nodes. This paper assumes a special case of ML and CL constraints based on thresholding. A more general case can be built for any ML and CL constraints provided by the user.

Using the min-cut method to cluster data was introduced by [8]. They applied min-cut repeatedly to create a hierarchical clustering. Another graph/clustering algorithm, Chameleon[11] uses the concepts of *closeness* and *inter-connectivity* (which have similarities to our concepts of impurity and incompleteness) to form clusters.

Overlapping clustering has been proposed using a mixture model by Banerjee, et al.[1]. Using a modified probabilistic relational model they assign objects to multiple clusters by using a threshold which is based on a distance from a central point. Like adclus, this does not directly address the bridge-node problem.

8. Conclusions and Future work

In this paper, we have studied the effect of bridge-nodes on clustering and present two metrics, impurity and incompleteness, to measure the effectiveness of a clustering algorithm in terms of its ability to handle bridge-nodes. We demonstrate how cloning helps to produce clusters with zero impurity and incompleteness scores, but such an approach may lead to an overwhelming number of cloned nodes unless it is done appropriately. We prove that finding the optimum solution with minimum number of cloned nodes is intractable and present an approximation algorithm called MIN-CUT that produces a reasonably low number of cloned nodes (with respect to the NAÏVE approach) without degrading the impurity and incompleteness scores.

The work presented in this paper assumes that the cost for adding a cloned node is significantly lower than the cost of having an impure or an incomplete node. If the relative costs among the impure, incomplete, and cloned nodes are known, an optimal algorithm should minimize the objective function given in Section 4.3. Designing such an optimization algorithm for a general cost function is a subject for future work.

Two other possible directions for future research would be to modify MIN-CUT to use weighted graphs and to run comparisons of the MIN-CUT algorithm to other overlapping methods and the recent semi-supervised algorithms. It would be interesting to compare them by the measures introduced in this paper.

9. Acknowledgements

The authors wish to thank the reviewers for their constructive and helpful comments.

10. References

[1] A. Banerjee, et al., *Model Based Overlapping Clustering* International Conference on Knowledge Discovery and Data Mining, 2005.

[2] S. Basu, M. Bilenko, R. Mooney, *A probabilistic framework for semi-supervised clustering* Conference on Knowledge Discovery in Data ., 2004.

[3] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981

[4] M. Bilenko, S. Basu, R. Mooney, *Integrating constraints and metric learning in semi-supervised clustering*, Proceedings of the twenty-first international conference on Machine learning, 2004.

[5] G. Chartrand, O. Oellermann, *Applied and Algorithmic Graph Theory* ,McGraw Hill, Inc. 1992.

[6] I. Davidson, S. Ravi, *Clustering with Constraints: Feasibility Issues and the k-Means Algorithm.* Siam Data Mining Conference 2005.

[7] I. Davidson, S. Ravi, *Towards Efficient and Improved Hierarchical Clustering With Instance and Cluster Level Constraints.* Technical Report, Department of Computer Science, University at Albany.

[8] G. Flake, K. Tsioutsiouliklis, R. Tarjan, Graph Clustering Techniques based on Minimum Cut Trees, Technical Report 2002-06, NEC, Princeton, NJ, 2002.

[9] M. Garey and D. Johnson, *Computers and intractability. A guide to the theory of NP-completeness*, W.H. Freeman and Company, New York-San Francisco, 1979

[10] A. Jain, R. Dubes. *Algorithms for clustering data.*, Prentice-Hall, Inc., 1988

[11] G. Karypis, E.-H. Han, and V. Kumar. *Chameleon: Hierarchical clustering using dynamic modeling*. IEEE Computer, 32(8):68--75, 1999.

[12] R. Shepard, P. Arabie, *Additive Clustering: Representation of Similarities as Combinations of Discrete Overlapping Properties*, Psychological Review, March 1979

[13] P. Tan, M. Steinbach, V. Kumar. *Introduction to Data Mining.*, Addison Wesley, Inc., 2005.