

Vertex Reordering Algorithms for Cluster Identification

Christopher Mueller

Project Final Report

L529, "Algorithms for Bioinformatics"

May 6, 2005

ABSTRACT

In this paper we present two graph-theoretic algorithms for cluster identification and demonstrate the effectiveness of their combined use. Building on the results of our previous study using sparse matrix reordering schemes to generate clusters, we modify the most promising algorithms to take advantage of similarity scores to produce orderings that nearly recover known clusters. Both algorithms, built on connected components and depth first search are efficient and can be applied to large data sets. When combined with similarity matrix visualizations, complex inter- and intra-cluster relationships can be identified.

1. INTRODUCTION

Clustering is an important technique for identifying relationships across many related data items. Traditional clustering techniques such as k-means and hierarchical clustering exploit distance metrics between data elements to group similar elements together. However, most traditional techniques rely on metrics that must meet additional criteria. For instance, the distance metric for k-means must be valid in a Euclidean space. Often, it is unclear if a given similarity metric meets this requirement or it may not be possible to map the metric to a valid space. Additionally, there may not be a valid metric that builds the complete distance matrix between all nodes. In these cases, all that can usually be assumed is that two items are similar. Transitive relations between three items are not always valid and other special interpretations can be meaningless.

Given these constraints, it is desirable to develop clustering techniques that use domain knowledge in the form of similarity scores and make no other assumptions about relationships between data items.

In this paper, we explore two graph-theoretic approaches to linearly ordering data such that similar items are kept close together. Using visual similarity matrices, these orderings help in identifying relations between items and groups of items and can be used to generate clusters and view relationships between the clusters.

Previously, we studied the effectiveness of sparse matrix reordering schemes for cluster identification [1]. Reverse Cuthill-McKee (RCM), King's algorithm, modified minimum degree and connected components all produced orderings that exposed internal relationships between data items. However, most of the relationships were superficial with respect to the graph and had little to do with the actual similarities between the elements. In general, this was due to the fact that all of the algorithms only considered

connectivity and paid no attention to the weight of the connections.

To use these algorithms, data sets are characterized as graphs, with each item corresponding to a vertex in the graph and edges between two vertices if the items at each vertex are similar. The edges are weighted according to the degree of similarity between the nodes and the graphs can be processed as weighted or unweighted graphs. As mentioned, our previous results were on unweighted graphs.

To improve the results, we note that clusters in large data sets often form at different levels of similarity. For instance, a large, tightly connected cluster may exist at a low similarity score and, once removed, other clusters may be present at higher thresholds. This was observed when connected components was applied to different versions of the COG data set, each including proteins that are above a certain similarity threshold to other proteins. The first algorithm presented in this paper, weighted connected components exploits this observation by progressively filtering the graph and removing components if they meet a clustering condition.

The second algorithm builds on the superficial clusters generated by the more complex reordering algorithms, RCM and King. Both algorithms visit nodes based on out degree, giving higher priority to nodes with higher degree. Our weighted depth first search is a simple greedy algorithm that visits nodes based on edge weight and relevance to the current node. It makes no assumptions about the relationships between all the nodes and uses only the local relations.

The next sections detail related work, our methods and the results. In the conclusion, we highlight future directions for the algorithms to help them completely handle large data sets.

2. Related Work

Graph based clustering and node reordering schemes are not new. However, most focus on properties of the graph and exclude the extra domain knowledge available. Or, if they include domain knowledge, they often attempt to identify some larger graph property that may or may not correspond to an important property of the data set.

As mentioned in the introduction, traditional matrix reordering schemes such as reverse Cuthill-McKee modified minimum degree and King's algorithm use properties of the degree of the nodes to generate orderings.

These are effective at minimizing certain graph properties such as bandwidth (RCM, King) but fail to provide detailed insight into the local relationships among data items.

Traditional graph algorithms for identifying shortest paths (e.g. Dijkstra’s algorithm, see [2] for a comprehensive overview of graph algorithms) and minimum spanning trees do take into account edge weights. However, their goal is identifying a high-level property of the graph (e.g., short distance between two nodes) and again only provide partial details about the relationships between nodes.

Modern density-based clustering algorithms such as DB-SCAN and OPTICS [3] are build clusters by ordering elements. But, they require a Euclidean distance between all points to be successful.

K-cores [4] takes advantage of connected components and iteratively generates ‘cores’, or components that are connected at a certain ‘core’ level. The iterative approach used in k-cores is similar to our weighted connected components algorithm, but k-cores ignore the edge weight and focuses only on node degree.

Sun Kim’s BAG algorithm [5] for protein cluster identification is the most similar to ours. It uses an iterative bi-connected components algorithm that progressively builds clusters. Our algorithms do not attempt to directly assign cluster membership and thus do not process graphs in the same exact manner as BAG.

3. Methods

Two algorithms were developed for reordering the nodes of the graph. The first, weighted connected components (WCC), iteratively applies the connected components algorithm to the graph and extracts components at different threshold levels. WCC is designed to work on large, sparse graphs that have clusters at different thresholds. The second algorithm is designed to generate good orderings of highly connected components. It visits nodes based on edge weight and outputs the nodes in the order are visited.

The algorithms are evaluated by comparing the results to known clusters using the COG protein family database. The visual similarity matrix technique is described in detail in our earlier report.

3.1 Weighted Connected Components

The weighted connected components (WCC) algorithm is listed in Figure 1. Starting at a low threshold, it finds the connected components for the graph and tests each one to see if it forms a cluster. To form a cluster, each vertex must be connected to at least some percentage of the vertices in the component. If the component forms a cluster, the nodes are removed from the graph and output as cluster. Once all the components at a given threshold have been processed, the threshold is increased and the process repeated. If a preset maximum threshold is reached, all remaining nodes are output in a random order. This last step is essentially a check for no-progress and can have an adverse effect on the final results.

```
WEIGHTED_CC_CLUSTER(G, threshold, inc):
    // G is the dataset as a similarity graph
    // threshold is the starting score for similar
    // inc is the function that (in/de)creases the
    // threshold
    result = []

    while G has vertices:
        cc = connected_componets(G, threshold)

        for each component in cc:
            if IS_CLUSTER(component, ...):
                G.removeVertices(cluster.vertices)
                result.append(component)

        threshold = inc(threshold)

    return result
END WEIGHTED_CC_CLUSTER

IS_CLUSTER(cc, outdegree, accept):
    // CC is a component
    // outdegree is the % of nodes the source must
    // see
    // accept is the % if nodes that must pass the
    // outdegree test
    n = 0

    for vertex in cc.vertices:
        out = vertex.outEdges().targets
        i = 0
        // Determine the local out degree for
        // the vertex
        for target in out:
            if target in cc:
                i += 1
        if (i / cc.size()) > outdegree:
            n += 1

    if (n / cc.size()) > accept:
        return true
    else
        return false
END IS_CLUSTER
```

Figure 1 The Weighted Connected Components algorithm

WCC was inspired by the observation that for COG graphs with varying minimum thresholds for edge weights, connected components identifies a small number families perfectly. However, it fragments and over-clusters families with average similarity not near the threshold. Iteratively applying connected components to more specific graphs allows the families characterized by low similarity to get clustered first, essentially filtering the graph so higher similarity families are less affected by low noise. This progressive filtering using the clustering condition described above has the overall effect of keeping highly related vertices together at each threshold level.

3.2 Weighted Depth First Search

As we will detail in the results section, for dense areas of the graph, the ordering produced by WCC is too coarse-grained to distinguish subtle relationships at a given threshold level. Simply outputting the nodes in a random order ignores any intra-cluster relationships. In our previous work, both RCM and King generated decent

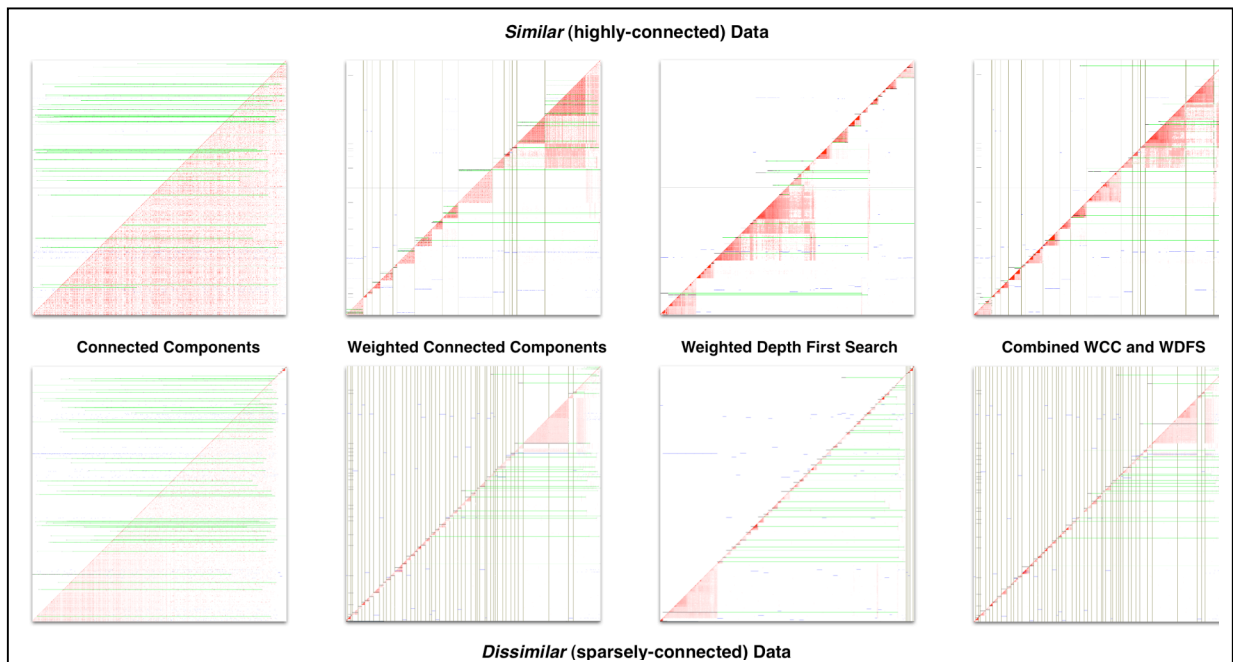


Figure 2 Similarity matrices for the reordering algorithms. The top and bottom rows contain the results for the *similar* and *dissimilar* data sets, respectively. The results for each algorithm are presented together. The nodes are ordered identically along each axis and each edge between node i and j is rendered as a shaded dot at position (i,j) in the plot with the shade of the dot corresponding to the weight of the edge. The span of each COG family is represented by a green line connecting the first and last elements in the family. For instances where the family was fully recovered, the span lines are not visible. The vertical lines delineate clusters identified by the weighted connected components algorithm. Each component falls between two

orderings based only on out degree for small connected components. However, the final orderings suffered from the lack of domain knowledge built into out degree. For most clusters, many inter-cluster edges exist at low threshold levels, giving too much weight to strongly connected nodes.

To address this, the nodes within a component are output based on a weighted depth first search. Starting with the node that has the greatest overall weight (i.e., the sum of the edge weights to all other nodes in the component), the nodes are traversed by always selecting the target of the edge with the largest weight. At each step, the current node is output and removed from consideration. This continues until no nodes remain. If the current node has no edges and nodes still remain, a new node is selected based on greatest overall weight.

3.3 Evaluation

We evaluated the algorithms by studying their behavior on the COG protein family database. We created two subsets of COG families and compared all the proteins in each set using FASTA. FASTA scores ranged from 2.0 to over 5000.0, with the majority between 50.0 and 200.0. The set *similar* included 50 closely related families containing 1770 proteins with many inter-family relationships at high-threshold values. The 50 family *dissimilar* set, on the other hand, had 2030 proteins with fewer high-threshold inter-

family relationships and the intra-family relationships existing at different threshold levels.

The algorithms were evaluated by determining the number of sub-clusters each COG family was distributed across and the distance between the first and last clusters. Sub-clusters were identified by traversing the nodes in order and marking the nodes with their COG family. A node flanked by nodes from different families was considered a single node sub-cluster and two or more adjacent nodes from the same family were considered a multi-node sub-cluster. For each sub cluster, its size and distance to the next sub-cluster from the same family were recorded. The distance provided an overall spanning metric for COG families. The span of a family is the distance from the first node in the first sub-cluster to the last node in the last sub-cluster. If the cluster was fully recovered (i.e., the entire COG family was contained in one sub-cluster), the size of the sub-cluster is equal to the span of the family.

Connected components, weighted connected components and weighted depth first search (WDFS) were tested independently on each data set and the combined algorithm WCC/WDFS was also tested. Connected components considered the entire graph and ignored the edge weights.

In the case where WCC reached its maximum threshold and dumped nodes, results were generated for the entire set of nodes and set of ordered nodes excluding the unprocessed nodes. As we will show, the results are striking and

Table 1 Cluster Recovery Statistics for each algorithm. WCC and Combined the stats with and without the unprocessed nodes, on the first and second line for each algorithm, respectively.

	Sub-clusters	Average Distance	Span	Families Recovered	
Similar					
WCC (all)	1087	18.6	325	2	(4%)
(processed)	994	6.5	121	3	(6%)
Combined	165	105.7	281	18	(36%)
	111	20.1	68	24	(48%)
WDFS	160	75.2	300	17	(34%)
Dissimilar					
WCC	186	177	389	25	(50%)
	47	2.7	41	43	(86%)
Combined	80	305	373	28	(56%)
	49	20.7	61	42	(84%)
WDFS	99	342	485	19	(38%)

provide a strong impetus for developing a more rigorous method for handling the unprocessed nodes.

4. Results

Figure 2 and Table 1 show the similarity matrices and statistics for each data set and each algorithm. The span of each family is rendered as a green line in the picture. For the two versions of weighted connected components (with and without the depth first ordering for the components), the component boundaries are displayed as vertical black lines.

4.1 Connected Components

Connected components performs poorly on both data sets, splitting the *similar* data into 1548 distinct sub-clusters and the *dissimilar* data into 1281 sub-clusters. This is mostly due to the fact that at the low background threshold, the *similar* graph contains only one connected component and *dissimilar* contains five (one large and four small components). With no other conditioning on the output and no consideration of the threshold values, connected components is not expected to perform well.

4.2 Weighted Connected Components

Visually, weighted connected components appears to perform well for both data sets. However, the numbers reveal a slightly less optimistic picture for the *similar* data set, where the COG families were split into 1087 sub-clusters (994 if the unprocessed nodes are ignored). This is an improvement over connected components, but nowhere near the optimal number of 50 sub-clusters. However, the average span for each COG family was only 121 and the average distance between sub-clusters within a family was 6.5. This visible in the graphic: very few clusters span the vertical component boundaries. Thus, while weighted connected components cannot provide a good ordering for families that are very similar to each other, it can group those families into clusters based on the threshold filter.

The results for the *dissimilar* data are much better. WCC generated only 186 sub-clusters including the unprocessed nodes and 47 sub-clusters without the unprocessed nodes. Of those 47 sub-clusters, all but two map fully to a COG family. That is, 43 COG families were recovered without being segmented. And, the segmented families were only split into two sub-clusters that were close to each other (with distances of 82 and 39). Note that the missing COG families were contained completely in the unprocessed nodes.

4.3 Weighted Depth First Search

The weighted depth first search (WDFS) algorithm was tested without WCC filtering against both data sets. It performed well for both sets. The families in the *similar* set were split into 160 sub-clusters with an average span of 300. The *dissimilar* families were split into 99 sub-clusters, or on average just two sub-clusters per family. The average spans, however, were fairly large. In the visualization, it is clear that portions of each cluster were pushed to the end of the ordering, effectively separating families by large distances.

While WDFS generated good orderings for these relatively small data sets, preliminary results on the entire collection of COG proteins suggest that it degrades quickly for larger graphs. Further study will be needed to fully understand its scaling performance.

4.4 Combined WCC and WDFS

The combined WCC and WDFS algorithms produced the best results for the *dissimilar* data and the best results for the *similar* data when the unprocessed nodes are ignored.

The *similar* data resulted in 165 and 111 sub-clusters, depending on whether the unprocessed nodes were included. The former, corresponding to the inclusion of the unprocessed nodes, is only 5 sub-clusters larger than the best results produced by weighted depth first search and a significant improvement over the 1087 sub-clusters

generated by WCC alone. Without the unprocessed nodes, the combined algorithm has the lowest number of sub-clusters for the *similar* data.

The *dissimilar* data was divided into 80 (with unprocessed) nodes and 49 (without unprocessed nodes) sub-clusters, with five missing families when the unprocessed nodes are excluded. In the latter case, only three families were split and the average distance between then split sub-clusters was only 20.7. In other words, the combined algorithm grouped almost all families perfectly and when it didn't, the sub-clusters were very close together.

5. Conclusion

We have described and evaluated two efficient algorithms for ordering data elements to reveal cluster structure, weighted connected components and weighted depth first search. Individually, WCC performs well for sparsely connected graphs and has reasonable coarse-grained performance for denser graphs. WDFS produces good orderings in both cases, but has a tendency to place some elements with low edge weights and connectivity a great distance from related elements. The combined algorithm leverages the strengths of both approaches, filtering the graph for dense regions and generating good orderings of the dense regions. This multi-scale approach yields *de novo* orderings that correspond well with known clusters in the COG family data set.

Currently, the combined approach suffers from one major drawback. The greedy nature of the cluster filtering leaves some nodes unprocessed. Generally, the unprocessed nodes have a weaker connection to some group of nodes. Future work to identify a strategy for including unprocessed in the final ordering is necessary in order to fully handle large data sets.

Overall, considering edge weights in common graph algorithms generates much better results than when they are ignored. This is common practice in flow and spanning tree algorithms, but is not discussed often in the context of node ordering schemes. These results, along with density based cluster approaches such as OPTICS, suggest that other reordering schemes can be effective and efficient clustering techniques.

6. References

- [1] Mueller, Christopher, *Sparse Matrix Reordering Algorithms for Cluster Identification*. December, 2004. <http://www.osl.iu.edu/~chemuell/projects/bioinf/sparse-matrix-clustering-chris-mueller.pdf>
- [2] Cormen, T., Leiserson, C., Rivest, R., Stein, C., *Introduction to Algorithms, Second Edition*, MIT Press. September 1, 2001.
- [3] M. M. B. Mihael Ankerst, Hans-Peter Kriegel, Jörg Sander, *OPTICS: Ordering Points To Identify the Clustering Structure*, *ACM SIGMOD'99*, 1999.
- [4] Batagelj, V., Zaversnik, M. *Generalized Cores*. <http://arxiv.org/abs/cs.DS/0202039>

- [5] Kim, Sun. *Graph Theoretic Sequence Clustering Algorithms and Their Applications to Genome Comparison in Computational Biology and Genome Informatics* (Ch. 4). Wu, C., Want, P., Want, J., eds. World Scientific, 2003.