Visualization of k-connected Components and Minimum Separating Sets of Fixed Points of Degree Peeling

James Abello
Rutgers University
Piscataway, NJ, USA
Email: abelloj@cs.rutgers.edu

Daniel Nakhimovich
The Cooper Union
New York, NY, USA
Email: nakhimov@cooper.edu

***Abstract*—**
Project dates: June – July 2018
Graphs are an excellent form for the visualization of data because they clearly shows how individual data points are connected. However, for large sets of data, a direct visualization of a graph is indecipherable to the human eye. Separating sets and k-connected components are interesting structures in graphs that highlight critical data points and clusters of highly connected points respectively. In this paper, we develope an algorithm that uses minimum separating sets to decompose a graph into a hierarchy of k-connected components. The complexity of the algorithm depends linearly on the number of k-connected components in the graph. For each k-connected component $K = (V, E)$, however, the complexity of finding its minimum separating set is $O(nm\binom{n}{2})$ where $n = |V|$ and $m = |E|$. By using different approximate procedures this complexity can be improved to $O(nm)$ or at more cost to accuracy to $O(n + m)$. Performing the separating set decomposition creates a tree-structured map of the decomposed graph that more easily shows the connectivity of the graph and consequently the data it represents.

I. Project Description

With the ever expanding availability of data, good visualization tools are more important than ever to aid people in analyzing data quickly. A popular way to visualize data is through the use of graphs as they are excellent sturctures for showing connections among a set of data. Fortunately, the connectivity of graphs is a heavily studied topic and many algorithms exist to analyze and manipulate the structures of graphs. The graph structures that we will be focusing in this paper are minimum separating sets and k-connected components. There structures are insteresting to visualize as they often reveal critical data points or groupings of data points in the data that the graph represents. For example, minimum separating sets could show critical failure points in a computer network, bottlenecks in distribution system, or even influential individuals in a social network. On the other hand, k-connected components offer a way to measure the redundancy in a computer network and distribution system and the closeness of groups in a social network.

For the purpose of this paper we will specifically focus on these structures within Fixed Points of Degree Peeling. The peeling process and an algorithm to perform a decomposition of a graph into Fixed Points of Degree Peeling is described in detail in [1].

In this paper we first define basic graph-theortic concepts as well as k-connected components and minimum separating sets. Then, we introduce an algorithm for a graph decomposition into a hierarchy of k-connected components. Next, we propose some relaxations that allow for a faster algorithm implementation which still produces a decomposition of similar macro-structure to the ideal version. Finally we show the decomposition applied to a real data set and offer some interpretations for what it shows.

II. Basic Definitions

An **undirected graph** $G = (V, E)$ is a tuple of a vertex set V and an edge set E, which consists of unordered pairs of elements in V. A **path** in G is a sequence of vertices such that each consecutive pair of vertices is in E. A graph is said to be **connected** if there exists a path between any two vertices in V. A **directed graph** is very similar to a graph except that each element of E is an ordered pair of elements in V. Each edge is often described as (and represented visualy as) pointing from its **source vertex** to its **target vertex**. The **degree** of a vertex $v \subset V$ denoted by $deg(v)$ is the number of edges containing v. The **diameter** of a graph is the longest path of shortest distance between any two vertices.

A **tree** is an undirected graph in which any two vertices are connected by exactly one path. A **rooted tree** is simply a tree with one vertex chosen as the root. The choice of root is only significant in context. One such instance is for a **directed rooted tree** where all the edges point either towards the root or away from the root. A **leaf** of a tree is any vertex $v \subset V$ such that $deg(v) = 1$. A **node** of a tree is any vertex that is not a leaf.

A **separating set** $V' \subset V$ is a distinct set of vertices that, when removed from G, induces a subgraph that is not connected. A **minimum separating set** is any such V' where $|V'| \leq |S_i|, \forall S_i \in S$ the set of separating sets of G.

A graph G is said to be **k-connected** if $|V| > k$ and there doesn't exist a minimum separating set V' with $|V'| < k$.

III. Separating Set Decomposition

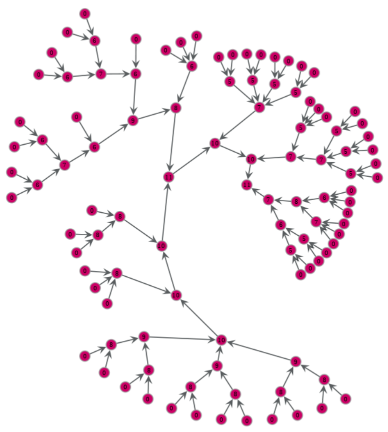

Fig. 1. Separating Set Decomposition of Peel Layer 11 of Rutgers MS students' study plans.

A. Decomposition Algorithm

The Separating Set Decomposition produces a directed rooted tree with the edges pointing towards the root. Each node in the Separating Set Decomposition represents a k-connected component of the graph, k being the label of the vertex (see Fig. 1). The children of each node are the connected components resulting from spliting the graph along the minimum separating set of the given k-connected component (see Ftn. split). The leaves of the tree either represent k-connected components that don't have a minimum separating set (complete graphs) or trivial components (trees).

```
Ftn. split
Input: A graph $G$ with separating set $S$
Output: A set of $n$ connected components
        $C = \{G_1, G_2, ...G_n\}$, $G_i \subset G \ \forall 1 \leq i \leq n$
function split $(G, S)$:
    $G' \leftarrow G \setminus S$
    $C \leftarrow \{\}$
    for $Q$ in connected components of $G$ do
        $C \leftarrow C \cup \{Q \cup S\}$
    end
    return $C$
end
```

To produce the full Separating Set Decomposition one needs to create the tree by recursively spliting a graph along the minimum separating set (see Alg. 1).

Since the Separating Set Decomposition is recursive, its complexity depends largely on the topology of the graph in question; namely, the number of k-connected components. Each recursive step has a complexity bounded by the complexity of finding the minimum separating set as the rest of the algorithm's operations are either constant time or linear with respect to number of vertices. As you'll see in the next section the worst case complexity of finding the minimum separating set is $O(n^2\binom{n}{2})$ where $n = |V|$.

```
Alg. 1: Separating Set Decomposition
Input: A graph $G$, a directed rooted tree $T$ (possibly
       empty)
Output: A directed rooted tree $T'$ or nothing
function decomposition $(G, T)$:
    if $T = \emptyset$ then
        $r \leftarrow G$
        $V'_G \leftarrow \{r\}$
        $E' \leftarrow \{\}$
        $T' \leftarrow (V'_G, E')$
        decomposition $(G, T')$
        return $T'$
    end
    $S \leftarrow$ minSeparatingSet $(G)$
    $k \leftarrow |S|$ $C \leftarrow$ split$(G, S)$
    if $|C| = 1$ then
        return
    end
    $V'_G \leftarrow \{\}$
    $E' \leftarrow \{\}$
    for $Q$ in $C$ do
        $V'_G \leftarrow V'_G \cup \{Q\}$
        $E' \leftarrow E' \cup \{(\{Q\}, \{G\})\}$
        decomposition $(Q, T)$
    end
    $(V_G, E) \leftarrow T$
    $T \leftarrow (V_G \cup V'_G, E \cup E')$
end
```

B. Finding the minimum separating set

To find the minimum separating set, we can compare the minimum separating set between every pair of vertices. An algorithm for finding the minimum separating cut between to vertices is described in detail in [2] but the basic idea is to find the minimum edge cut of an auxilariy graph using your favorite flow algorithm and establishing a correspondance between that minimum edge cut of the auxilariy graph to a minimum vertex cut (or separating set) of the original graph. The auxilary graph is formed by replacing each vertex v with a vertex pair v_1, v_2 and an edge pointing from v_1 to v_2. Then for each edge $e = (u, v)$ in the original graph, replace that edge with two directed edges $e_1 = (u_2, v_1)$ and $e_2 = (v_2, u_1)$. See Fig. 2 for a visual illusration.

Since the algorithm to find the minimum separating set in [2] is bounded in complexity by the max flow algorithm, [3] garauntees us a complexity of $O(nm)$ where $n = |V|$ and $m = |E|$ if we consider all edge weights to be 1. Thus, including the fact that we need to check every pair of vertices, the complexity for the finding the minimum separating set of a graph is at worst $O(nm\binom{n}{2})$.

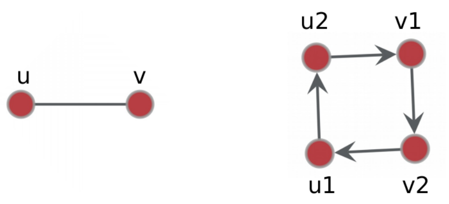

Fig. 2. Example transformation from an undirected source graph (left) to an auxilary graph (right)

IV. Approximate Minimum Separating Set Techniques

Although it is nice to know that finding the minimum separating set can be done with relatively low polynomial complexity with respect to the number of vertices and edges, for large data sets that complexity needs to be imporoved in order to work for visualization tasks. In this section we will discuss two approximate algorithms that are better in complexity and justify their loss in exactness.

A. Distant Vertex Pair

Finding the min vertex cut between every pair of vertices is rather wasteful. If there is only one minimum separating set than it will be found rather quickly as the majority of vertex pairs will be separated by it. On the other hand if there are multiple minimum separating sets than we have no reason to prefer one over the other. So, instead of finding the min cut between every pair of vertices one can simply check a pair of vertices that are far apart. This distant vertex pair will be separated by the same vertices as others near them if there is only one minimum separating set and if there are multiple minimum separating sets than the distant vertex pair is likely to have at least one of the minimum separating sets between them. Of course, there is no absolute garauntee that the min vertex cut between the distant vertex pair will be a minimum separating set for the entire graph. For instance, if the all minimum separating sets of the graph included those two vertices than this approuch would not find any of them.

Ideally, the vertex pair to pick would be a pair of vertices on the diameter of the graph but finding the actual diameter of a graph would actually introduce more complexity. Instead one can use a pseudo diameter algorithm as descibed and implemented in [4]. The complexity of this algorithm is $O(n+m)$ where $n = |V|$ and $m = |E|$. With the relaxation of only checking a distant vertex pair, the complexity for finding a pseudo minimum separating set is bounded by just one run of the max flow algorithm and thus becomes $O(nm)$.

B. Choosing Between Minimum Separating Set Candidates

The previous approuch, though offering a major improvement in complexity, is still bounded by the complexity of a max flow algorithm. Abandoning the use of a max flow algorithm altogether could potentially yield a great improvement in complexity for finding a pseudo minimum separating set. A simple way of finding a pseudo minimum separating set is by picking the smallest separating set out of a handfull of minimum separating set candidates. Below is a procedure for finding good candidates that borrows from the idea of distant vertex pairs and appears to work well empirically:

1) Find a distant vertex pair as described in section IV-A.
2) Using breadth first search label each vertex in the graph with the distance from each vertex in the distant vertex pair.
3) Each group of vertices that are labeled with the same distance from the same vertex is a separating set due to the nature of a breadth first search so choose them as candidates for a minimum separating set.
4) Check if the group of vertices that are equidistant from both vertices in the distant vertex pair are a valid separating set and if they are then add them as a candidate.

Although the accuracy of this procedure is not as well justified for finding a pseudo minimum separating set as the approach IV-A, when used for the purposes of Alg. 1 the overall macro-structure of the decomposition looks very similar to the decomposition using the pseudo minimum separating set algorithm of approach IV-A. Meanwhile the complexity of this procedure is only bounded by that of a breadth first search. Thus, if one accepts the loss in exactness, the complexity for the finding a pseudo minimum separating set can be reduced to $O(n + m)$ where $n = |V|$ and $m = |E|$.

V. Application and Interpretation

The data set we will be examining in this section is a set of protein structures. When viewed in graph form, the vertices represent different protein molucules and an edge between two vertices tells us that those two protein molucules share a high similarity in structure. Also, the subset of data which will be shown here is from Peel layer 5 of the Iterative Edge Core Decomposition described in [1] and contains only 1301 vertices and 4641 edges.

A. Visualization

In Fig. 3 the entirety of Peel Layer 5 is shown through an ARF layout as described in [4]. Although a few clusters are visible it is relatively difficult to make much sense of how different protein molucules are related. In Fig. 4 we have the Separating Set Decomposition which is displayed in a radial tree format that is much easier for a person to follow. In effect the Separating Set Decomposition acts as a map where the leaves are the atomic components of a graph and the nodes shows connections between the atomic components or between other nodes (see Fig. 5). By looking at a leaf and working your way towards the root, one can get a much better sense of how that piece of the graph connects to the entire graph as a whole.

B. Interpretation and Conclusion

Although biology is out of our expertise, considering the nature of the Separating Set Decomposition and what the vertices and edges of the original graph represent, one could imagine that looking at separating sets could help one find proteins that were a turning point in some evolutionary process. At

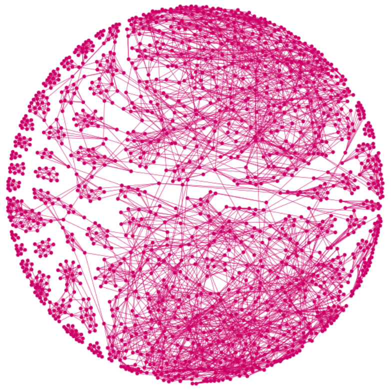

Fig. 3. Peel Layer 5 of Protein Structure Data.

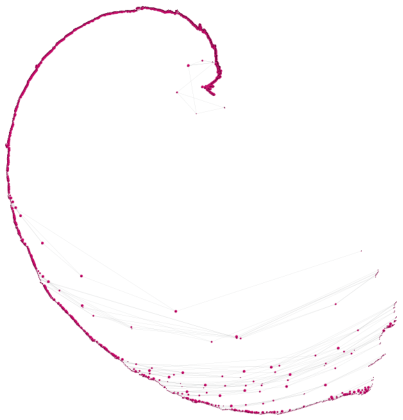

Fig. 4. Separating Set Decomposition of Peel Layer 5 of Protein Structure Data.

the very least, the k-connected components that the separating sets divide can highlight groups of protein molecules with very similar structures. And using the map, one can see how these groups share similarites between other groups.

Ultimately, biology aside, the power of the Separating Set Decomposition is to decompose a large graph that is hard to visualize. The decomposition creates a map of the graphs k-connectivity that can then be easily visualized by a computer and provide insight about the data represented in the graph to an informed user.

References

[1] J. Abello and F. Queyroi, "Fixed points of degree peeling," in *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Niagara, Ontario, Canada, Aug. 2013, pp. 256–263.

[2] S. Acid and L. M. de Campos, "An algorithm for finding minimum d-separating sets in belief networks," in *Twelfth international conference on Uncertainty in artificial intelligence*, Portland, OR, Aug. 1996, pp. 3–10.

[3] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *ACM (JACM)*, vol. 19, Apr. 1972.

[4] T. P. Peixoto, "The graph-tool python library," *figshare*, 2014. [Online]. Available: http://figshare.com/articles/graph_tool/1164194

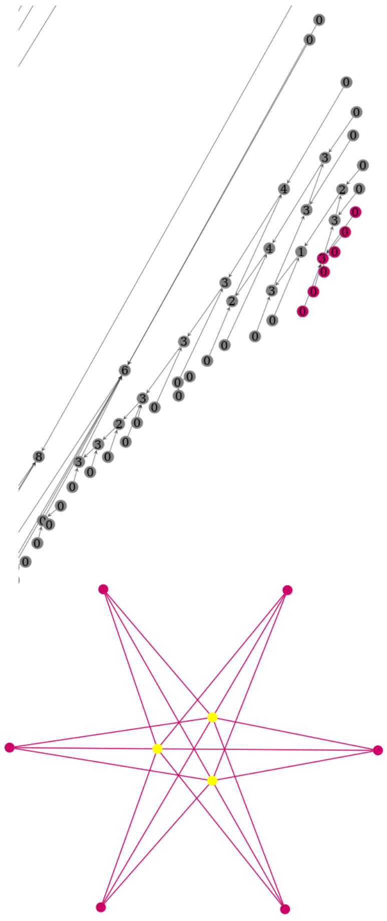

Fig. 5. Zoomed in View of the Separating Set Decomposition (top) and the subgraph corresponding to the highligted portion of the decomposition with the pseudo minimum separating set highlighted in yellow (bottom).