A New Clustering Technique using (k, w)-Core Decomposition for Restructuring Software Functions

Aftab Hussain and Md. Saidur Rahman
Graph Drawing and Information Visualization Laboratory
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Email: aftab.hussain46@gmail.com, saidurrahman@cse.buet.ac.bd

***Abstract*—In this paper, we introduce a hierarchical agglomerative clustering technique (HAC) based on a new graph theoretic algorithm, (k, w)-Core Decomposition. Previous HACs generate clustering trees or dendrograms with a large number of cut-points and bad clusters. The new technique generates lower readings for these two parameters, while giving results of competitive quality, efficiently. To establish this, we implemented our HAC and previous HACs for restructuring software functions, and made a comparative analysis of the results.**

I. Introduction

Hierarchical agglomerative clustering (HAC) algorithms have found wide utility in different applications primarily because of the efficiency with which they could be implemented. One such application is the restructuring of low-cohesive software modules[1]. HACs return a hierarchy of clusters, which is visualized as a dendrogram (Fig. 1): a 2D diagram in which a scale of similarity from 1 to 0 is represented in the vertical axis and entities are indicated in the horizontal axis. Each horizontal line in the dendrogram indicates a cluster, the height of which indicates the level of similarity of the cluster. A *cut-point* in the dendrogram is the level of similarity at which a dendrogram is cut to obtain a partition of the entities. Each cluster corresponds to a cut-point. Each cut-point yields a partition of clusters, which give advice on how to restructure the module. Previous HACs, Single Linkage Algorithm (SLINK), Complete Linkage Algorithm (CLINK), Weighted Pair Group Method of Arithmetic Averages (WPGMA), and Adaptive K-Nearest Neighbour Algorithm (A-KNN) ([2], [3], [4], [5]), return dendrograms with a large number of cut-points, of which only a few yield clusters that lead to a meaningful restructuring. We present an efficient HAC based on (k, w)-Core Decomposition, (k, w)-CC, that generates clustering trees which contain lower numbers of cut-points and bad clusters. The technique is implemented to restructure software functions and shown to give competitive results with respect to previous HACs.

II. Preliminaries

We shall first present definitions and lemmas related to two key elements of (k, w)-CC: the *k-core*, first introduced by Seidman [6], and the *(k, w)-core*, introduced in this work.

[1]The cohesion of a module is the degree to which the module performs a unique task [1].

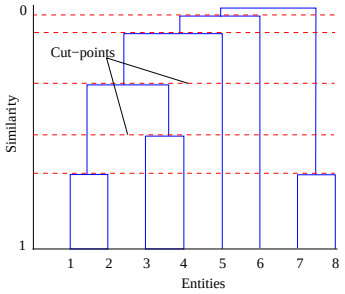

Fig. 1. A dendrogram with cut-points.

Definition 1: Let $G = (V, E)$, be a graph, where V is the set of vertices and E is the set of edges. A subgraph H_k of G induced by a vertex set $V \subseteq V$ is a *k-core* of G if every vertex in V has degree at least k in H_k, and H_k is the maximum subgraph with this property [7].

Lemma 1: If H_{k_1}, H_{k_2} are the k_1- and k_2-cores, respectively, of a graph G, where $k_2 > k_1$, then H_{k_2} is a subgraph of H_{k_1}.

Definition 2: Let W be the set of different edge weights of graph G, where $w \in W$. Then a *(k, w)-core* of G is a subgraph of G where the degree of each vertex of the subgraph is at least k and the weight of each edge of the subgraph is at least w, and this subgraph is the maximum subgraph with this property.

Lemma 2: A (k, w)-core of a weighted graph G is a subgraph of a k-core of G.

III. (k, w)-Core clustering ((k, w)-CC

Fig. 2 illustrates the overall approach of our novel clustering technique. The clustering technique operates on a weighted graph, where vertices represent entities and edges represent the presence of some similarity between the vertices (entities) joined by the edges[2]. Each edge carries a weight equal to the similarity value between the vertices joined by the edge. Firstly, (Fig. 2(a)), all possible (k, w)-cores are generated from the weighted graph. Then, (Fig. 2(b)), cores are systematically selected to form clusters, which together form a cluster hierarchy. We now discuss these steps in detail.

[2]In the context of software restructuring at the function level, entities denote statements of the function and similarity values denote the relationship between entities based on a resemblance coefficient. (See [5].)

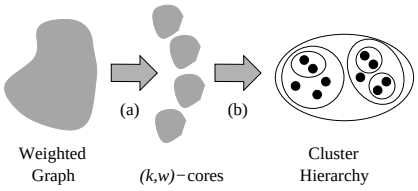

Fig. 2. Clustering approach using (k, w)-Core Clustering.

(k, w)-Core Decomposition. This decomposition algorithm is based on the k-core decomposition algorithm [7]. The algorithm begins by scanning every vertex v of input graph G to find the k-core for each $k \epsilon D$, where D is the ordered set of distinct degrees of the graph. If the degree of a vertex v is found to be less than k, v is deleted and the degrees of v's neighbours are decremented. In case the degrees of any of v's neighbours, fall below k, those vertices are also deleted recursively. In this manner, a k-core is generated. For generating k-cores for successive k values in D, it is sufficient to scan the k-core with the preceding k value rather than the entire graph (see Lemma 1).

By Lemma 2, for a particular value of k, say k_n, all (k_n, w)-cores of G can be obtained from the k_n-core of G. In order to do this the algorithm scans all edges of the k_n-core. For each weight $w \epsilon W$, where W is the set of distinct weights of G, edges with weights less than w are deleted. We thus get a set of intermediate graphs for each value of w. Next, the degrees of the vertices of each intermediate graph is checked and deleted if their degrees fall below k. We thus obtain all the (k_n, w)-cores of G. Carrying the same steps on all k-cores gives us all the (k, w)-cores of G.

Core Selection and Clustering Tree Generation. The basic idea of this phase is to select the (k, w)-cores as clusters in order to form a hierarchy. Cores are ranked using a new metric that relies on the k and w value of the core[3]. The selection process terminates until all entities have been selected.

IV. Experimental results

In our experiments we restructured 11 functions (ranging from 9 to 41 LOCs) extracted from published papers ([5], [2], [8]) and a real-life software, Sweet Home 3D ([9]), using SLINK, CLINK, WPGMA, A-KNN, and (k, w)-CC. We recorded the **number of cut-points** and the **number of bad clusters** that were observed in their respective dendrogram outputs. We also measured the **maximum cohesion improvement** that was attainable through each technique and the **execution time** taken by each technique to generate the dendrograms. We present the results as follows,

Number of cut-points and bad clusters. On average, (k, w)-CC gave 29.23%, 39.47%, 52.58%, 31.34% fewer number of cut-points than did SLINK, CLINK, WPGMA, and A-KNN, respectively. In addition, (k, w)-CC gave 57.41%, 62.90%, 68.49%, 54.00% fewer number of bad clusters than did SLINK, CLINK, WPGMA, and A-KNN, respectively.

[3]A detailed explanation of this metric has been omitted in this short version.

Maximum Cohesion Improvement. Overall, the maximum cohesion improvement level that was possible to achieve with SLINK, CLINK, WPGMA, A-KNN, (k, w)-CC was 99.88%, 99.92% 99.92%, 99.88%, 99.88%. Thus, the quality of (k, w)-CC's results competes well against those of the other techniques.

Execution Time. We measured the time taken, in milliseconds (ms), by each clustering technique to generate dendrograms for each of the functions that were analyzed [4]. Overall, (k, w)-CC performed 59.72% percent faster than SLINK, CLINK, and WPGMA. However, A-KNN performed the fastest among all the techniques. (It was 94.77% faster SLINK, CLINK, and WPGMA).

V. Conclusion

In this work, we developed a new hierarchical clustering technique based on (k, w)-core decomposition, (k, w)-Core Clustering ((k, w)-CC), for restructuring functions. We compared the performance of (k, w)-CC with four previous HACs (SLINK, CLINK, WPGMA, and A-KNN). The techniques were implemented on functions extracted from published papers and real-life software. Our technique gave competitive restructuring solutions through dendrograms that contained a smaller number of cut-points and a smaller number of bad clusters. As a result, (k, w)-CC's dendrograms were easier to analyze, from which meaningful suggestions were more readily retrievable. Performance-wise, although (k, w)-CC was slower than A-KNN, it was considerably faster than SLINK, CLINK, and WPGMA.

Acknowledgments

We would like to thank Mohammad Tanvir Parvez, Abdulaziz Alkhalid, Mohammad Alshayeb, and Sabri Mahmoud for providing the implementation details of the A-KNN algorithm.

References

[1] R. Pressman, *Software Engineering: A Practitioner's Approach*, 6th ed. McGraw-Hill, 2004.

[2] A. Alkhalid, M. Alshayeb, and S. Mahmoud, "Software refactoring at the function level using new adaptive k-nearest neighbor algorithm," *Journal of Advances in Engineering Software*, vol. 41, no. 10-11, pp. 1160–1178, 2010.

[3] ——, "Software refactoring at the package level using clustering techniques," *IET Software*, vol. 5, no. 3, pp. 276–284, 2011.

[4] N. Anquetil and T. C. Lethbridge, "Experiments with clustering as a software remodularization method," in *Proceedings of 6th Working Conference on Reverse Engineering*, 1999, pp. 235–255.

[5] C. H. Lung, X. Xu, M. Zaman, and A. Srinivasan, "Program restructuring using clustering techniques," *Journal of Systems and Software*, vol. 79, no. 9, pp. 1261–1279, 2006.

[6] S. B. Seidman, "Network structure and minimum degree," *Social Networks*, vol. 5, no. 3, pp. 269–287, 1983.

[7] V. Batagelj and M. Zaversnik, "An o(m) algorithm for cores decomposition of networks," in *CoRR (Computing Research Repository), cs.DS/0310049*, 2003.

[8] J. M. Bieman and B.-K. Kang, "Measuring design-level cohesion," *IEEE Transactions on Software Engineering*, vol. 24, no. 2, pp. 111–124, 1998.

[9] "Sweethome3d," http://sourceforge.net/projects/sweethome3d/?source=directory, July 2012.

[4]All executions were carried out in a system with a 2.4 GHz processor and a 4096 Mb RAM.