Austin Journal of Proteomics, Bioinformatics & Genomics

Open Access

Austin Publishing Group

Research Article

An Efficient Hierarchical Clustering Algorithm for Large Datasets

Olga Tanaseichuk[1,2]*, Alireza Hadj Khodabakshi[1], Dimitri Petrov[1], Jianwei Che[1], Tao Jiang[2], Bin Zhou[1], Andrey Santrosyan[1] and Yingyao Zhou[1]
[1]Genomics Institute of the Novartis Research Foundation, 10675 John Jay Hopkins Drive, San Diego,CA 92121, USA
[2]Department of Computer Science and Engineering, University of California, Riverside, CA 92521, USA

***Corresponding author:** Tanaseichuk O, Genomics Institute of the Novartis Research Foundation,10675 John Jay Hopkins Drive, San Diego, CA 92121, USA

Received: September 24, 2014; **Accepted:** February 23, 2015; **Published:** February 25, 2015

Abstract

Hierarchical clustering is a widely adopted unsupervised learning algorithm for discovering intrinsic groups embedded within a dataset. Standard implementations of the exact algorithm for hierarchical clustering require $O(n^2)$ time and $O(n^2)$ memory and thus are unsuitable for processing datasets containing more than 20 000 objects. In this study, we present a hybrid hierarchical clustering algorithm requiring approximately $O(n\sqrt{n})$ time and $O(n\sqrt{n})$ memory while still preserving the most desirable properties of the exact algorithm. The algorithm was capable of clustering one million compounds within a few hours on a single processor. The clustering program is freely available to the research community at http://carrier.gnf.org/publications/cluster.

Keywords: Hybrid hierachical clustering; Hierachical clustering; K-means clustering; Large datasets

Introduction

Clustering is a popular unsupervised learning technique used to identify object groups within a given dataset, where intra-group objects tend to be more similar than inter-group objects. There are many different clustering algorithms [1], with applications in biocheminformatics and other data mining fields [2,3], including studies on protein families [4], functional genomics [2], chemical scaffolds [5], etc. In particular, clustering algorithms have been widely adopted in the bioinformatics fields after Treeview [6], a user-friendly visualization program, was made available following early studies on gene expression datasets.

Among all clustering methods, hierarchical clustering and *k*-means clustering are arguably the two most popular algorithms used due to their simplicity in result interpretation. In the cheminformatics field, Wards clustering [7] and Jarvis-Patrick clustering [8] are corresponding algorithms similar in spirit to hierarchical clustering and *k*-means clustering, respectively. Although there is no definitive answer as to which algorithm is more accurate, hierarchical clustering has been applied more often in bio-/cheminformatics research because of its deterministic property and flexibility in flattening the resultant tree at different cutoff levels.

However, applying hierarchical clustering to large datasets is rather challenging. First, compared to the linear complexity of the *k*-means algorithm, the most popular average-linkage hierarchical clustering requires $O(n^2)$ time; we even observed $O(n^3)$-time implementations in some popular bioinformatics tools [9]. Second, it requires $O(n^2)$ memory [10], which limits the number of input data points to ~ 20 000 for a typical desktop computer. In bioinformatics research, functional genomics profiling data approaching this limit are routinely generated for the human genome [11]. In cheminformatics research, modern drug discovery applies ultra-high-throughput screenings (uHTS) for several million compounds in one experiment [12]. Two problems arise from uHTS. First, to expand the screening compound collection, vendor catalogs of millions of compounds ideally should be hierarchically clustered and prioritized for acquisition. But in practice, informaticians resort to a greedy algorithm such as Sphere Exclusion [13], which relies on a predetermined similarity threshold. Second, instead of analyzing all compound profiles across a panel of screening assays, hierarchical clustering analyses have usually been compromised and restricted to ~ 20 000 top screening hits due to memory limitations. Therefore, there exists a significant need to develop a hierarchical clustering algorithm for large datasets.

Approximating hierarchical clustering in subquadratic time and memory has been previously attempted [14-18]. However, these methods either rely on embedding into spaces that are not biologically sensible, or they produce very low resolution hierarchical structures. Our goal is to produce hierarchical results with the same resolution as the exact hierarchical method, although with less accuracy, while maintaining the bio/chemically meaningful distance metrics. For a dataset over 20 000 objects, we are limited by both $O(n^2)$ memory and time. Therefore, a reasonable approximation needs to be introduced. We observe that if an exact hierarchical tree has been constructed, one can set a similarity cutoff such that tree branches above the cutoff are distant enough from each other and represent the coarse clusters of the dataset. The branches and leaves below the cutoff represent hierarchical structures within small-scale local vicinities. For a large dataset, we are often initially interested in a “zoomed out” view of the coarse clusters, then “zoom in” to neighborhoods of interest for a finer view of the intra-group structures. The two views are often considered to be the most beneficial properties of the hierarchical clustering. For example, in the aforementioned compound requisition problem, one would cherry pick vendor compounds from the coarse neighborhood if only a small number of compounds can be selected for purchasing. When budget and logistics permit, one could then lower the cutoff to pick more compounds within interesting coarse clusters.

To capture both distant and close views of the hierarchical structure for a large dataset, we propose a hybrid hierarchical clustering algorithm (Figure 1). Initially, the *n* objects are clustered

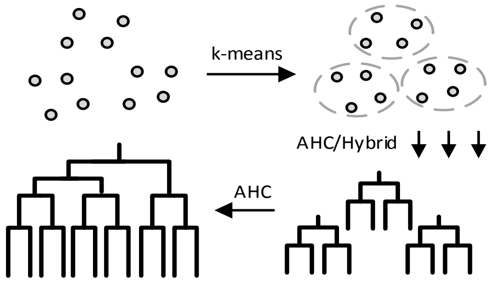

Figure 1: Hybrid hierarchical clustering pipeline. First, the objects are clustered by a *k*-means algorithm. Then, objects within each cluster are hierarchically clustered by either the exact agglomerative hierarchical clustering algorithm (AHC), or the hybrid method is applied recursively for large clusters. Finally, AHC is preformed to cluster *k* centroids and combine the trees into a complete tree.

by a *k*-means clustering algorithm, where *k* is chosen to be reasonably large, into roughly *k* coarse neighborhoods. We then apply the exact hierarchical clustering algorithm to cluster the *k* centroids into a coarse tree, as well as to the objects within each of the *k* clusters into k detailed trees. By replacing the *k* centroids in the coarse tree by the corresponding detailed trees, this two-step hybrid algorithm assembles a complete tree of *n* objects that can be cut, i.e., zoomed in and zoomed out, at various levels. The number *k* can be selected by the user and controls the cutoff reflecting the average similarities of objects within each coarse neighborhood. Practically, we cannot reliably distinguish data points positioned closer than the magnitude of the intrinsic noise of the data. Therefore, these data could be treated as one aggregated data object without losing meaningful interpretation of the data set. If the value of *k* is large enough, the size of individual *k*-means clusters approaches the intrinsic noise, and the *k*-means clustering step retains most of the essential information, thus the tree resulted from the hybrid algorithm could be considered as accurate as the one resulted from the exact AHC algorithm. If optimized for clustering speed, $k \sim \sqrt{n}$ can be chosen to yield an approximate running time of $O\left(n\sqrt{n}\right)$ and storage of $O\left(n\sqrt{n}\right)$ as discussed later in detail.

In the past few years, other attempts have been made to combine hierarchical clustering with *k*-means. For example, hierarchical *k*-means [19] is a well-known divisive hierarchical clustering algorithm that constructs a tree by recursively bisecting the data with k-means. This method has a low time complexity of $O(n\log(n))$, however, it may produce low quality clustering results. Specifically, for small values of *k* the algorithm inherits disadvantages of divisive clustering methods-high likelihood that similar objects may be separated during early stages of clustering, leading to low local accuracy of the clusters. Choosing larger values of *k* would potentially fix this, but at the expense of poor global clustering structure, since at the same level of recursion all clusters are connected at the same distance from the root in the complete tree. In contrast, in our hybrid algorithm we try to preserve both local and global clustering structure of the data simultaneously.

Materials

As our aim is to develop an algorithm for practical biomedical research applications, three real datasets encountered in our routine analyses were chosen. Dataset *D*1 is an activity matrix consisting of 2117 compounds profiled across 398 cancer cell lines. A subset of this matrix was previously published as the Cancer Cell Line Encyclopedia project and was described in detail by Barretina et al. [20]. This dataset provides an example of a typical medium-size clustering problem involved in bioinformatics and cheminformatics research.

Dataset *D*2 is a larger high-throughput screening activity matrix of 45 000 compounds across 178 assays. This is a subset of the larger matrix described in a published HTS frequent hit study [21]. A total of 45 000 compounds that hit the most number of assays were selected, because this size approaches the upper limit of what an exact hierarchical clustering algorithm can handle on a typical desktop computer. This large dataset provides a test case to compare the speed of clustering and the qualities of resultant trees, when both the exact hierarchical clustering algorithm and the proposed hybrid algorithm are applied.

Dataset *D*3 consists of one million compounds randomly selected from our in-house compound collection, where the average Tanimoto structure similarity determined by ChemAxon two-dimensionalfingerprinting is merely 0.3 [22]. As structural redundancy of the collection is low, these compounds are expected to form numerous clusters of fairly small sizes. This set is chosen to represent the more challenging problem of identifying structurally diversified compounds from a large vendor catalog as well as to enable us to study the robustness of the hybrid algorithm.

Results

The hybrid algorithm

In this section, we introduce a hybrid algorithm for hierarchical clustering of large datasets. Our approach combines the advantages of the partitioning and agglomerative hierarchical clustering algorithms.

Hierarchical clustering organizes the data into a dendrogram that represents the clustering structure of the data. We only consider the bottom-up clustering approach here due to its ability to capture the local clustering structure of the data. The classic agglomerative hierarchical clustering (AHC) method [23] requires computation of all pairwise distances, which has a quadratic complexity. Therefore, the construction of the distance matrix creates a bottleneck, especially for high dimensional data and expensive distance functions. Since AHC algorithms greedily merge pairs of nearest data points (clusters) into tree nodes, the exact computation of pairwise distances is important for data points that are close enoughto each other, while the computation of distances between remote points is unlikely to contribute and should be avoided whenever possible. Therefore, it makes sense to partition the dataset to avoid the fulldistance matrix computation.

In the first step of the algorithm, we partition the data with *k*-means [24], a simple and effective clustering algorithm that generates a locally optimal partitioning of the data. The number of components *k* is predefined.The choice of *k* and performance of our algorithm with respect to *k* are discussed later in the paper. We apply the optimized version of the exact *k*-means algorithm, which utilizes a triangle inequality to avoid unnecessary distance computations [25]. The clusters are initialized uniformly at random from the data points. In the second step, at the first level, AHC is applied to cluster each individual component P_i obtained by *k*-means into an individual detailed tree T_i. At the second level, each T_i is treated as a leaf and clustered by AHC into a coarse tree T. T, therefore, reflects both the coarse relationships among components as well as detailed relationships among members of each component.

A few questions arise in the above procedure and require careful consideration: (1) How are distancesdefined between the components for the second level of clustering?(2) What should be done when the distance between a pair of component centroids is smaller than the radii of associated components?

Regarding the first question, the naive idea of taking the distance between the centroids of components as a pairwise distance between these components is undesirable. Consider two pairs of components, where the distance between the two centroids within each pair is the same. Additionally, assuming that the first pair of components have small radii while the other two components have large radii and may overlap. Clearly, the above naive approach would not capture the intuition that the second pair of components should be considered closer. We adopt the idea of data bubbles [26] and define the distance of twocomponents P_1 and P_2 as follows:

$$d(P_1,P_2)=\begin{cases} d(C_1,C_2)-(R_1,R_2)+d_{NN}(P_1)+d_{NN}(P_2) & \text{if } d(C_1,C_2)-(R_1,R_2)^{3}0 \\ max\, d_{NN}(P_1)+d_{NN}(P_2) & \text{otherwise} \end{cases}$$

Here, C_i is the centroid of the partition P_i, and R_i is the radius of the component (most of the objects are located within the radius R_i around the centroid C_i), $d_{NN}(P_i)$ is the average 1-nearest neighbor distance within the component P_i.

Regarding the second question, for each component P_i, we define the distance threshold r_i so that all points that are farther than r_i away from the centroid are considered outliers and removed from the component. Outliers are added as individual points and used in the second level of hierarchical clustering.

In addition, due to the nonuniform distribution of objects within a real dataset, the *k*-means clusteringmight result in components that exceed the size limit of AHC. Therefore, the hybrid algorithm might need to be recursively applied in a divide-and-conquer manner. Occasionally, when the height of a detailed tree T_i exceeds its corresponding level-two centroid distance, its height should be propagated up to its ancestral nodes along the tree branches during the assembly of T.

Our hybrid algorithm is outlined in Algorithm 1.

The running time and memory analysis

We theoretically and experimentally evaluate the running time of the hybrid algorithm. First, let us show that with a reasonable choice of the partitioning parameter k, the algorithm runs in $O\left(N\sqrt{N}\right)$ time for datasets of randomly distributed objects. The running time of the algorithm is affected by (1) the time to partition the data in the k-means phase and (2) the running time of the hierarchical clustering phase. The traditional *k*-means algorithm requires computing *kNL* distances, where *L* is the number ofiterations. However, in the optimized version of *k*–means, only the first few iterations require distance computations from all the data points to all the centroids. The time needed for subsequent iterations drops significantly, because most of the distances are not computed. Thus the overall number of distance computations becomes closer to *kN* than to *kNL*. The *k*-means phase runs in $O(kNL')$, where *L*<*L* and can be estimated

```
begin
  P ← ∅
  Perform optimized k-means clustering to partition the data into components $P_i$.
  for each component $P_i$ do
      $r_i$ = arg min$_j$ dist($P_i$, $P_j$)
      Compute centroid $C_i$
      for each point p in $P_i$ do
          if dist(p, $C_i$) > $r_i$ then
              Remove p from $P_i$. Add p to P.
  for each component $P_i$ do
      if size($P_i$) > n then
          Recursively apply hybrid clustering to generate a tree $T_i$.
      else
          Apply AHC to generate a tree $T_i$.
  Compute a combined distance matrix for all $P_i$ and all points in P.
  Perform AHC to generate a tree T.
  return T
```

Algorithm 1: Hybrid clustering of *N* data points. Given *k*, the algorithm partitions the dataset and performs two-level hierarchical clustering to construct a tree *T*. (The maximum size of the input for the agglomerative hierarchical clustering (AHC) algorithm is *n*. It can be supplied by the user, or estimated automatically).

experimentally. In our experiments, *L* was in the range of 2 to 5. The running time of the hierarchical clustering phase includes the time required to hierarchically cluster *k* subsets of approximate sizes *N/k* and to cluster *k* centroids. Assuming quadratic time complexity for the AHC algorithm, the overall running time of the hybrid algorithm is $O(k^2 + N^2/k + kNL)$. As the first term k^2 is dominated by *kNL*, our algorithm runs in $O(N^2/k + kNL)$. time. Thus, the minimum expected running time is achieved when *k* is set to *N=L'*, leading to $O\left(N\sqrt{N}\right)$ time complexity. The same analysis applies to the memory complexity which is also bounded by $O\left(N\sqrt{N}\right)$.

We measured the experimental running time of the hybrid algorithm for different values of *k*, for both the partitioning phase and the hierarchical clustering phase. The results are shown in Figure 2. Even though the real data is not uniformly distributed, trends in the experimental results agree with the theory. Note, that the exact algorithm matches the cases of *k=1 and k = N*. Clearly, the larger the data size, the more we gain in clustering speed compared to the exact algorithm. For example, when the parameters are optimized,

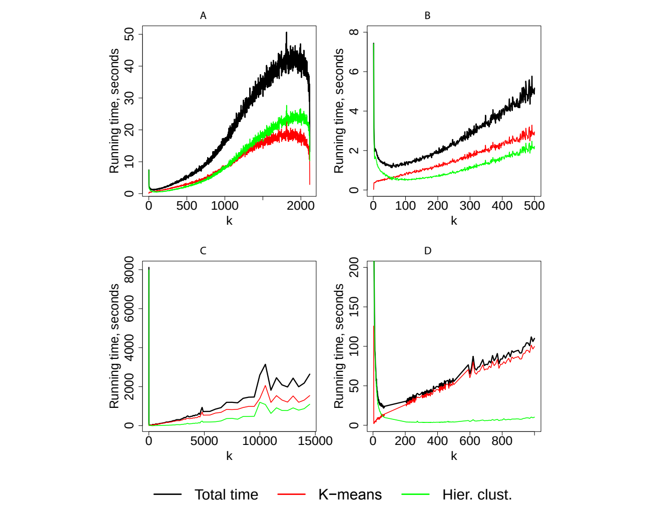

Figure 2: Running time of the hybrid algorithm for datasets *D*1 and *D*2. (A) Dataset *D*1, for all values of *k*. (B) Dataset *D*1, zoomed in for *k* < 500. (C) Dataset *D*2. (D) Dataset *D*2, zoomed in for *k* < 1000.

the hybrid algorithm is only 5 times faster on *D*1 while it is 370 times faster on *D*2, running in 22 seconds compared to 8117 seconds for the exact algorithm.

Performance analysis

There is no universal agreement on how clustering should be performed. Therefore, methods for validating clustering results vary significantly [27]. Since our primary goal is to accelerate AHC, the hierarchical tree *T* produced by the AHC algorithm is taken as the gold standard and is referred to as the exact tree. The tree produced by the hybrid algorithm is referred to as a hybrid tree or an approximate tree. Quantitative comparison between the exact tree and a hybrid tree remains an open problem and few results exist in the literature. One approach is to use a well-known tree edit distance [28], but it is computationally expensive and may produce counter-intuitive results [29]. Another popular approach is to cut trees at certain heights and measure similarity between the resultant clusters. The latter was chosen for this study, as it provides visualization that can be cross-examined by biological and chemical domain knowledge. Various similarity measurements are applicable to two sets of clusters resulting from tree cuts, e.g., Jaccard index [30], Rand index [31], Fowlkes-Mallows index [32], information theoretic measures [33], etc. Each method has its own advantages and weaknesses [34]. For example, the Rand index has an undesirable property of converging to 1 as the number of clusters increases, while the Fowlkes-Mallows index makes strong assumptions about the underlying distribution of data, making it hard to interpret the results. The information-theoretic approaches are promising for clustering validation, but require a more extensive evaluation. In our study, we chose the Jaccard index, one of the most common similaritymeasures for clustering.

For each of the two given datasets, we first cut the exact tree *T* at some height *g*, which was selected based on the combination of our domain knowledge of the bio- and cheminformatics problems and our visual inspection of the exact hierarchical tree *T*. This resulted in a set of clusters $C(g) = \{C_1, C_2, \ldots, C_{|C(g)|}\}$. The corresponding hybrid tree was then cut at different cutoff values *h* that would correspond to granularity. For each *h*, the Jaccard similarity index between *C*(*g*) and thehybrid clusters $\tilde{C}(h)$ was calculated according to:

$$J(C(g), \tilde{C}(h)) = \frac{N_{11}}{N_{11} + N_{10} + N_{01}},$$

Where N_{11} is the number of object pairs consisting of objects clustered together into the same cluster in both C and $\tilde{C}$. $N_{10} + N_{01}$ is the number of object pairs consisting of objects clustered together in either C or $\tilde{C}$ but not both. The *h* value that led to the highest Jaccard index was retained and used for the similarity score $S_g(T, \tilde{T})$:

$$S_g(T, \tilde{T}) = \arg\max_{h \in H} J(C(g), \tilde{C}(h)).$$

The set of cutoff values *H* was chosen to evenly cover different granularity levels of the resulting clusterings, where granularity is defined as a percent of object pairs that cluster together.

We are particularly interested in the approximation quality for biologically meaningful clusters with pronounced activity patterns. Therefore, in the computation of the similarity score, we disregarded clusters with low average Pearson correlation of the activity profiles (below 0.2) as well as small clustersthat contain less than 0.1% of the data. The selected clusters for datasets *D*1 and *D*2 are highlighted in Figures 3A and 4A, respectively. For the dataset *D*1, we additionally excluded a large cluster of low-activity compounds. Even though this cluster is well approximated by the hybrid algorithm, it dominates the resulting Jaccard index leading to an overall high similarity score. The results of the proposed similarity measures S_g on datasets *D*1 and *D*2 for the selected clusters are shown in Figure 5. It was observed that quality measurements for both datasets are rather in sensitive to the choice of *k* overa wide range. Since the hybrid tree T retains both the coarse and detailed structures within a datasetand provides approximate results for interpretations in-between, it is not surprising that *T* reasonably approximates the exact tree. Since high quality trees are produced for a wide range of the parameter values, it makes sense to optimize the parameter *k* mainly for improved running time in practice.

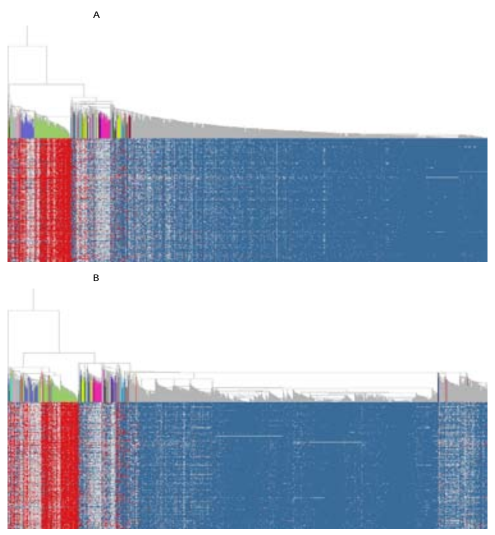

Figure 3: The hierarchical trees for the dataset *D*1. The trees are produced by (A) the exact algorithm and (B) the hybrid algorithm with *k* = 25. Highlighted are the biologically meaningful clusters selected for the evaluation of the approximation quality of the hybrid algorithm. The heat map illustrates the activity of compounds: red and green indicate active and inactive compounds, respectively.

Discussion

The implementation of the exact hierarchical clustering algorithm

We have been using a non-trivial assumption that AHC requires an $O(n^2)$ running time. The running times of specific AHC implementations actually vary significantly from the expected $O(n^2)$. Cluster 3.0 [9] provides a popular AHC implementation that is used extensively in the bioinformatics field. For the average-linkage configuration, Cluster 3.0 implementation takes $O(n^3)$ time, as shown in Figure 6. For this study, we adopt the Murtagh reciprocal nearest neighbor idea [10], which offers a much improved $O(n^2)$ time. To test this, both Cluster 3.0 and Murtagh algorithms were implemented in Java and were applied to sample datasets sizing between 1 000 and 20 000 (40 000 for the Murtagh implementation), where each data object consisted of double vectors of length 80. As shown in Figure 6, the Murtaghmethod indeed performed at the scale of $O(n^2)$ and Cluster 3.0 at $O(n^3)$. These results are in agreement with the recent

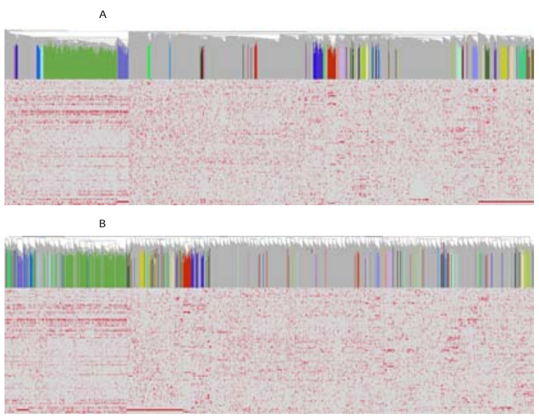

Figure 4: The hierarchical trees for the dataset *D*2. The trees are produced by (A) the exact algorithm and (B) the hybrid algorithm with *k* = 130. Highlighted are the biologically meaningful clusters selected for the evaluation of the approximation quality of the hybrid algorithm. The heat map illustrates the activity of compounds: the intensity of red is proportional to the compound's activity.

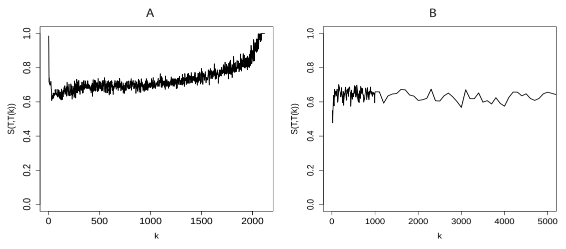

Figure 5: Approximation quality of the hybrid algorithm. The similarity score $O(n^2)$ between exact tree T and the hybrid trees $O(n^2)$ for different values of the parameter *k*. (A) Dataset *D*1. (B) Dataset *D*2.

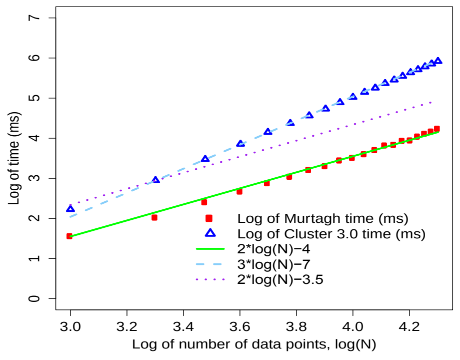

Figure 6: The performance comparison between the Murtagh method and the Java implementation of the Cluster 3.0 method. The running time of the Murtagh method matches a linear curve of slope 2 while the running time of the Cluster 3.0 method matches a linear curve of slope 3, showing that their running time are of $O(n^2)$ and $O(n^2)$, respectively. The green curve is a linear curve of slope 2 that crosses the curve of Cluster 3.0 running time included to make the comparison easier.

study [35]. It is worth mentioning that our Java implementation of Cluster 3.0 is two fold faster than the original C implementation, and the observation in Figure 6 is not an over-estimation. Note that although the Murtagh method has been used in the JKlustor program in the cheminformatics field [22], it is not widely adopted in bioinformatics. Therefore, bioinformatics researchers not using an $O\left(n^2\right)$ implementation of AHC could benefit from the release of our package.

Performance on a large dataset and robustness analysis

A major goal in proposing our algorithm is to provide a hierarchical method that is capable of clustering datasets that contain more than 40 000 objects. Here, we studied dataset *D*3, which consists of one million compounds randomly selected from our in-house compound collection. Running AHC on such a large dataset is infeasible and cheminformaticians have relied on greedy algorithms such as Sphere Exclusion (SE) [13] to partition the compounds into clusters. SE requires a fixed similarity cutoff value as its input. It randomly selects a query compound and extracts all remaining compounds, where their structural similarities to the query compound are above the predefined threshold. The extraction and exclusion process is iterated until the collection is exhausted. Because the exact tree is not available for adataset as large as *D*3, performance comparisons between SE and hybrid algorithms can not be conductedin a manner similar to what we presented in Sections The Running Time and Memory Analysis" and Performance Analysis". Nevertheless, we speculate that the hybrid method provides a result closer to the exact AHC tree than to SE. This is because no super-sized compound cluster is expected in *D*3 based on our domain knowledge, i.e., the sizes of chemically interesting clusters are small. The first *k*-means clustering step is expected to produce only large components of structurally diverse compounds and is unlikely to break down small groups of highly similar compounds. The SE algorithm, on the other hand, produced flattened clusters based on a rather subjective similarity threshold, which may not match the average similarities in small clusters.

A main criticism on SE is its greediness, which led to different clustering results in different runs in our experiment. As the hybrid algorithm also has a random component in the *k*-means stage, it would be interesting to compare the two methods for robustness in the results. We shuffled records in the one million compound dataset ten times and applied both algorithms. We then measured how well each method was able to reproduce its own results. In particular, we first applied a cutoff value to flatten hybrid trees into a similar number of clusters as in the output of the SE algorithm. Then, through random sampling of compound pairs in the output clusters, we estimated the probability that a pair of compounds will cluster together in consecutive runs to be 37.1% with a standard deviation of 0.9%15for the hybrid method, and 27.8% with a standard deviation of 1.6% for the SE methods (*p*-value *is* $1\times e^{-10}$). Similarly, we also estimated the probability that a pair of compounds will not cluster together in consecutive runs to be 99.8% and 99.9%, respectively. These results indicate the superior robustness of the hybrid algorithms across multiple runs.

Conclusion

We have introduced a hybrid hierarchical clustering algorithm that requires approximately $O\left(n\sqrt{n}\right)$ running time and $O\left(n\sqrt{n}\right)$ memory, producing hierarchical trees similar to what the exact hierarchical algorithm offers but applicable to much larger datasets. With three example datasets, the hybrid algorithm was demonstrated to be much faster, reasonably accurate and robust for clustering large datasets encountered in bioinformatics and cheminformatics research. The software package has been made available to the informatics community and should prove very useful when applied to a wide range of data mining problems.

Acknowledgments

We would like to thank Frederick Lo and Tom Carolan for their help proofreading the manuscript.

References

1. Jain AK, Murty MN, Flynn PJ. Data clustering: a review. ACM Comput Surv 1999; 31: 264-323.
2. Jiang D, Tang C, Zhang A. Cluster analysis for gene expression data: a survey. IEEE Transactions on Knowledge and Data Engineering 2004; 16: 1370-1386.
3. Brohee S, van Helden J. Evaluation of clustering algorithms for protein-protein interaction networks. BMC Bioinformatics 2006; 7: 488.
4. Klimke W, Agarwala R, Badretdin A, Chetvernin S, Ciufo S, et al. The National Center for Biotechnology Information's Protein Clusters Database. Nucleic Acids Res 2009; 37: D216-D223.
5. Downs GM, Barnard JM. Clustering Methods and Their Uses in Computational Chemistry. John Wiley and Sons 2003; 1-40. doi:10.1002/0471433519.ch1.
6. Eisen MB, Spellman PT, Brown PO, Botstein D. Cluster analysis and display of genome-wide expression patterns. The National Academy of Sciences 1998; 95: 14863-14868.
7. Ward JH. Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association 1963; 58: 236-244.
8. Jarvis RA. EAP Clustering using a similarity measure based on shared near neighbors. Computers, IEEE Transactions 1973; 1024-1035.
9. de Hoon MJL, Imoto S, Nolan J, Miyano S. Open source clustering software. Bioinformatics 2004; 20: 1453-1454.
10. Murtagh F. Complexities of hierarchic clustering algorithms: state of the art. J omputational Statistic Quarterly 1984; 1: 101-113.
11. Konig R, Zhou Y, Elleder D, Diamond TL, Bonamy GMC, et al. Global analysis of hostpathogen interactions that regulate early-stage HIV-1 replication. Cell 2008; 135: 49-60.
12. Hertzberg RP, Pope AJ. High-throughput screening: new technology for the 21st century. Curr Opin Chem Biol 2000; 4: 445-451.
13. Gobbi A, Lee ML. DISE: Directed Sphere Exclusion. J Chem Inf Comput Sci 2003; 43: 317-323.
14. Krznaric D, Levcopoulos C. The first subquadratic algorithm for complete linkage clustering. 6th International Symposium, ISAAC; Cairns, Australia,December 4-6, 1995 Proceedings. Springer 1995; 392-401.
15. Krznaric D, Levcopoulos C. Optimal algorithms for complete linkage clustering in d dimensions. 22nd International Symposium, MFCS 97, Bratislava, Slovakia, August 25-29, 1997 Proceedings. Springer 1997; 368-377.
16. Kull M, Vilo J. Fast approximate hierarchical clustering using similarity heuristics. Bio Data Mining 2008; 1: 9.
17. Murtagh F, Downs G, Contreras P. Hierarchical clustering of massive, high dimensional data sets by exploiting ultrametric embedding. SIAM J, Sci, Comput 2008; 30: 707-730.
18. Murtagh F, Contreras P. Fast, linear time, m-adic hierarchical clustering for search and retrieval using the baire metric, with linkages to generalized ultrametrics, hashing, formal concept analysis, and precision of data measurement. P-Adic Numbers, Ultrametric Analysis, and Applications 2012; 4: 46-56.
19. Bocker A, Derksen S, Schmidt E, Teckentrup A, Schneider G. A hierarchical clustering approach for large compound libraries. J Chem Inf Model 2005; 45: 807-815.
20. Barretina J, Caponigro G, Stransky N, Venkatesan K, Margolin AA, et al. The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. Nature 2012; 483: 603-607.
21. Che J1, King FJ, Zhou B, Zhou Y. Chemical and biological properties of frequent screening hits. J Chem Inf Model 2012; 52: 913-926.
22. Chemaxon. Http://www.chemaxon.com.
23. Day WH, Edelsbrunner H. Efficient algorithms for agglomerative hierarchical clustering methods. Journal of Classification 1984 1: 7-24.
24. Mac Queen JB. Some methods for classification and analysis of multivariate observations. In: Procedings of the Fifth Berkeley Symposium on Math, Statistics, and Probability. University of California Press 1967; 1: 281-297.
25. Elkan C. Using the triangle inequality to accelerate *k* Means. In: Proceedings of the Twentieth International Conference on Machine Learning 2003 (ICML-2003).
26. Breunig MM, Kriegel HP, Kroger P, Sander J. Data bubbles: quality preserving performance boosting for hierarchical clustering. In: SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data. New York, NY, USA: ACM 2003; 79-90.
27. Handl J, Knowles J, Kell DB. Computational cluster validation in post-genomic data analysis. Bioinformatics 2005; 21: 3201-3212.
28. Bille P. A survey on tree edit distance and related problems. Theor Comput Sci 2005; 337: 217-239.
29. Zhang Q, Liu EY, Sarkar A,Wang W. Split-order distance for clustering and classi_cation hierarchies. 21st International Conference SSDBM 2009 New Orleans, LA, USA, June 2-4, 2009 Proceedings. Scientific and Statistical Database Management 2009; 10:517-534.
30. Hamers L, Hemeryck Y, Herweyers G, Janssen M, Keters H, et al. Similarity measures in scientometric research: The Jaccard index versus Salton's cosine formula. Information Processing and Management 1989; 25: 315-318.
31. Rand WM. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 1971; 66: 846-850.
32. Fowlkes EB, Mallows CL. A method for comparing two hierarchical clusterings. Journal of the American Statistical Association 1983; 78: 553-569.
33. Vinh NX. Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J Mach Learn Res 2010; 11: 2837-2854.
34. Wagner S, Wagner D. Comparing clusterings: an overview. Technical Report 2006-04, University at Karlsruhe 2007;
35. Mullner D. fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. Journal of Statistical Software 2013; 53: 1-18.

Austin J Proteomics Bioinform & Genomics - Volume 2 Issue 1 - 2015
ISSN : 2471-0423 | www.austinpublishinggroup.com
Tanaseichuk et al. © All rights reserved

Citation: Tanaseichuk O, Khodabakshi A, Petrov D, Che J, Jiang T, et al. An Efficient Hierarchical Clustering Algorithm for Large Datasets. Austin J Proteomics Bioinform & Genomics. 2015;2(1): 1008.