Bidirectional Agglomerative Hierarchical Clustering using AVL Tree Algorithm

Hussain Abu-Dalbouh[1] and Norita Md Norwawi[2]

[1] Faculty of Science & Technology
University Sains Islam Malaysia (USIM)
Bandar Baru Nilai, 71800 Nilai
Negeri Sembilan
Malaysia

[2] Faculty of Science & Technology
University Sains Islam Malaysia (USIM)
Bandar Baru Nilai, 71800 Nilai
Negeri Sembilan
Malaysia

Abstract

The hierarchy is often used to infer knowledge from groups of items and relations in varying granularities. Hierarchical clustering algorithms take an input of pairwise data-item similarities and output a hierarchy of the data-items. This paper presents Bidirectional agglomerative hierarchical clustering to create a hierarchy bottom-up, by iteratively merging the closest pair of data-items into one cluster. The result is a rooted AVL tree. The *n* leafs correspond to input data-items (singleton clusters) needs to *n/2* or *n/2+1* steps to merge into one cluster, correspond to groupings of items in coarser granularities climbing towards the root. As observed from the time complexity and number of steps need to cluster all data points into one cluster perspective, the performance of the bidirectional agglomerative algorithm using AVL tree is better than the current agglomerative algorithms. The experiment analysis results indicate that the improved algorithm has a higher efficiency than previous methods.

Keywords: Hierarchical, Clustering, Bidirectional algorithm, agglomerative**,** AVL tree

1. Introduction

Recently, dramatic increases in the amount of information or data are being stored in electronic format. This accumulation of data has taken place at an explosive rate [1]. It has been estimated that the amount of information in the world doubles every 20 months and the size and number of databases are increasing even faster [1]. These have transformed societies into one that strongly depends on information and knowledge. Huge volumes of data, that have accumulated and generated, contains important information. These databases contain not only known information, but also new knowledge as well and are not easy to be extracted and understood.

This essentially requires the development of reliable and scalable analysis procedures to extract the hidden rules, technology and dramatic growth in applications such as internet search, digital imaging, and video surveillance have created many high-volume and high dimensional data sets. It is estimated that the digital universe consumed approximately 281 exabytes in 2007, and it is projected to be 10 times that size by 2011 (1 exabyte is 1018 bytes or 1,000,000 terabytes). Many domains started collecting and sorting data from different sources and the massive amounts of information from many fields, such as math, biology, medical science, business, banking, engineering, education, medical and DNA technology, have led to the accumulation of tremendous amounts of data. However, traditional clustering algorithms become computationally expensive when the data set to be clustered is large. Clustering is an area where the analysis of large data sets becomes a problem. Analyzing large data sets via traditional methods has moved from being tedious, to being highly computational cost.

2. Proposed Algorithm for Bidirectional Agglomerative Hierarchical Clustering using AVL tree in the case of single-linkage clustering method

In this section, in depth discussion is presented on how bidirectional algorithm using AVL tree works in the case of single-linkage clustering. The algorithm is an agglomerative scheme that erases nodes in the tree as old clusters are merged into new ones.

The clustering are assigned sequence numbers 0,1,....., (*n/2*), (*n/2*) +1 and L(*k*) is the level of the *k*th clustering. A cluster with sequence number *m* is denoted (*m*) and the proximity between clusters (*r*) and (*s*) is denoted d [(*r*),(*s*)]. The algorithm composed of the following steps:

BIDIRECTIONAL AGGLOMERATIVE HIERARCHICAL CLUSTERING USING AVL TREE ALGORITHM

1. *Begin with the disjoint clustering having level L(0) = 0 and sequence number m = 0.*
2. *Arrange the pairs from minimum distance (similarities) to the maximum distance.*
3. *Count how many pairs, say (n) pairs.(If n >=3) do,*
   {

3.1 *Find the median/root*

3.2 *Divide the pairs in two sides according the median. Say left side and right side.*

3.3 *Find the least dissimilar pair of clusters in the left and right current clustering, say pair (r), (s), according to*
d[(r),(s)] = min d[(i),(j)]
where the minimum is over all pairs of clusters in the current clustering.

3.4 *Check if left and right side have at least one similar object(element)then merge it together in one cluster, and find minimum is over all pairs of clusters in the current clustering.*
   }
   Else
   {

3.5 *Find the least dissimilar pair of clusters in the left and right current clustering, say pair (r), (s), according to*
d[(r),(s)] = min d[(i),(j)]
where the minimum is over all pairs of clusters in the current clustering.
   }

4. *Increment the sequence number: m = m +1. (In both sides) Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to*
L(m) = d[(r),(s)]

5. *Update the tree, T, by deleting the nodes corresponding to clusters (r) and (s) and adding a node corresponding to the newly formed cluster. The proximity between the new cluster, denoted (r,s) and old cluster (k) is defined in this way:*
d[(k), (r,s)] = min d[(k),(r)], d[(k),(s)].
6. *If all objects are in one cluster, stop. Else, go to step 2.*

Fig. 1: Pseudo code of the bidirectional agglomerative hierarchical clustering using AVL tree algorithm

2.1 Complexity of bidirectional agglomerative hierarchical clustering using AVL tree algorithm

Initially each of the *n* objects to be clustered is in a cluster by itself, in step 1 of each loop iteration the Tree *T* has nodes for each of the m remaining clusters. The number of clusters decreases by one for step 8 and 9. When step 9 completes, the revised tree T has a nodes for each of the (*m-1*) remaining clusters. Table 1 shows the complexity to the major steps of the algorithm.

Table 1: Paradigmatic bidirectional agglomerative hierarchical clustering using AVL tree algorithm

Algorithm	***Time complexity***
1. *Begin with the disjoint clustering having level L(0) = 0 and sequence number m = 0.*	
2. *Arrange the pairs from minimum distance (similarities) to the maximum distance.*	
3. *Count how many pairs, say (n) pairs. .(If n >=3) do,*	O(1)
3.1 *Find the median/root.*	O(1)
3.2 *Divide the pairs in two sides according the median. Say left side and Right side.*	O(1)
3.3 *Find the least dissimilar pair of clusters in the left and right current clustering, say pair (r), (s), according to d[(r),(s)] = min d[(i),(j)] where the minimum is over all pairs of clusters in the current clustering.*	O(1)
3.4 *Check if left and right side have at least one similar object (element) then merge it together in one cluster, and find minimum is over all pairs of clusters in the current clustering.* *Else;* 3.5 *Find the least dissimilar pair of clusters in the left and right current clustering, say pair (r), (s), according to d[(r),(s)] = min d[(i),(j)] where the minimum is over all pairs of clusters in the current clustering.*	O(1) O(1)
4. *Increment the sequence number: m = m +1. (In both sides) Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to: L(m) = d[(r),(s)]*	O(1)
5. *Update the tree, T, by deleting the nodes corresponding to clusters (r) and (s) and adding a node corresponding to the*	O(logn)

	newly formed cluster. The proximity between the new cluster, denoted (r,s) and old cluster (k) is defined in this way: *d[(k), (r,s)] = min d[(k),(r)], d[(k),(s)].*	
6.	*If all objects are in one cluster, stop. Else, go to step 2.*	

The complexity = Max { O(1), O(1), O(1), O(1), O(1), O(1), O(1), O(logn)} = O(logn).

In bidirectional agglomerative clustering using AVL tree, the distance of each cluster to all other clusters, and at each step the number of clusters decreases by one. Considering bidirectional *agglomerative* hierarchical clustering using AVL tree in the case of single-linkage clustering, if the number of objects are *n*, there are *n/2* (*in the best case*) or *n/2+1* (*in the worst case*) levels. Each level involves finding a minimum from tree *T* with time complexity O(1). Merging two clusters into a single cluster need O(1) then updating the proximity tree, *T*, by deleting the nodes corresponding to clusters need O(*logn*).

3. Related Works

Clustering is considered as an unsupervised classification process that means no predefined classes [4-6]. Clustering large data sets of high dimensionality has always been a serious challenge for clustering algorithms. Clustering of large datasets can be very difficult with the available clustering algorithms mainly due to the time complexity. Hierarchical methods rely on a distance function to measure the similarity between clusters. These methods do not scale well with the number of data objects. Their computational complexity is usually O(n^2). [7-9].

To solve the complexity problem, many improved algorithms are proposed [10-12]. That aimed to improve performance, some of these partition algorithms. It was chosen to reduce the distance calculation process. Like the method based on the k-d tree structure and pruning function proposed by [10], P-CLUSTER, the parallel clustering algorithm utilizes three kinds of pruning methods proposed by [13] and the parallel algorithm based on the k-d tree structure proposed by [14].

According to [15] by reducing distance or similarity calculation, the algorithm does not guarantee accuracy. [15] use parallel computing, assign the distance computing to show different nodes in a distributed environment, which improved the efficiency and ensure the effectiveness.

There are many recently developed representative of hierarchical clustering algorithm found in the literature that attempted and proposed for handling large data sets and to overcome the complexity time such as: i) Agglomerative Nesting (AGNES) [16] it with O(n^2) time complexity. ii) Divisive Analysis (DIANA) [16] it with O(n^2) time complexity. iii) Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [17] it with O(n) time complexity. iv) Clustering Using REpresentatives (CURE) [18] it with O(n^2) time complexity, v) RObust Clustering using links (ROCK) [19] it with O(O(n^2)+nmmma +n2logn)) time complexity.

4. An Example

The following discussion on bidirectional agglomerative hierarchical clustering using AVL tree algorithm is based on a simple example of distances in kilometers between some Malaysian states. The method used is single-linkage. There are eleven data points: JHR, KED, KTN, MLK, NSN, PHG, PRK, PLS, PNG, SGR, and TRG.

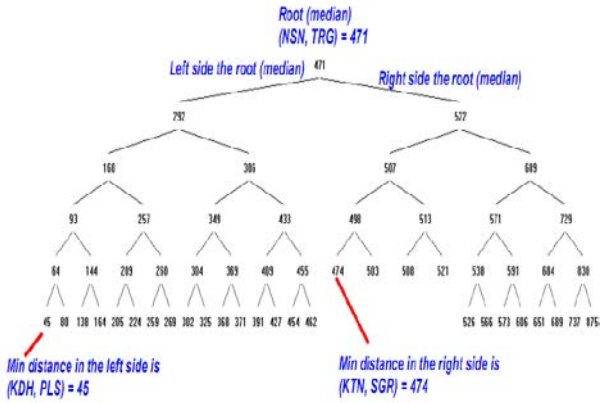

Fig. 2: First cluster in AVL tree

Based on Figure 2 the median/ root is NSN and TRG, at distance 471. The *Min* Left Side pair of states is KDH and PLS, at distance 45. These are merged into a single cluster called "KDH/PLS" and the Right Side pair of states is KTN and SGR, at distance 474. The level of the new cluster is L (KED/PLS) = 45, L (KTN/SGR) = 474, and the new sequence number is m = 1. In addition the left side and right side elements did not have any same element. Therefore no need to merge the left side cluster and right side cluster into a one cluster. The cluster, side, sequence number, elements and distances are shown in Table 2.

Table 2: First cluster

Cluster	***Side***	***Sequence No***	***Element***	***Distance***
KDH/PLS	Left side	1	KED and PLS	45
KTN/SGR	Right side		KTN and SGR	474

Then compute the distance from this new compound object in the left side to all other objects, and compute the distance from this new compound object in the right side to

all other objects. In single link clustering the rule is that the distance from the compound object to another object is *MIN* distance from any member of the cluster to the outside object. Therefore the distance from "KED/PLS" to JHR is chosen to be 830, which is the distance from "KED" to "JHR", and so on. The distances between this cluster and the remaining elements in the distance matrix are computed as shown below.

d (KED/PLS) JHR = *min* [*d* (KED, JHR), *d* (PLS,JHR)] = *d* (KED,JHR) = 830.
d (KED/PLS) KTN = *min* [*d* (KED, KTN), *d* (PLS, KTN)] = *d* (KED, KTN) = 409.
d (KED/PLS) MLK = *min* [*d* (KED, MLK), *d* (PLS, MLK)] = *d* (KED, MLK) = 609.
d (KED/PLS) NSN = *min* [*d* (KED, NSN), *d* (PLS, NSN)] = *d* (KED, NSN) = 526.
d (KED/PLS) PHG = *min* [*d* (KED, PHG), *d* (PLS, PHG)] = *d* (KED, PHG) = 684.
d (KED/PLS) PRK = *min* [*d* (KED, PRK), *d* (PLS, PRK)] = *d* (KED, PRK) = 257.
d (KED/PLS) PNG = *min* [*d* (KED, PNG), *d* (PLS, PNG)] = *d* (KED, PNG) = 93.
d (KED/PLS) SGR = *min* [*d* (KED, SGR), *d* (PLS, SGR)] = *d* (KED, SGR) = 462.
d (KED/PLS) TRG = *min* [*d* (KED, TRG), *d* (PLS, TRG)] = *d* (KED, TRG) = 521.

The distances between the right side cluster and the remaining elements in the distance matrix are computed as shown below.

d (KTN/SGR) JHR = *min* [*d* (KTN, JHR), *d* (SGR,JHR)] = *d* (SGR,JHR) = 368.
d (KTN/SGR) KTN = *min* [d(KTN, KED/PLS), *d* (SGR, KED/PLS)] = d (KTN, KED/PLS) = 409.
d (KTN/SGR) MLK = *min* [*d* (KTN, MLK), *d* (SGR, MLK)] = *d* (SGR, MLK) = 144.
d (KTN/SGR) NSN = *min* [*d* (KTN, NSN), *d* (SGR, NSN)] = *d* (SGR, NSN) = 64.
d (KTN/SGR) PHG = *min* [*d* (KTN, PHG), *d* (SGR, PHG)] = *d* (SGR, PHG) = 259.
d (KTN/SGR) PRK = *min* [*d* (KTN, PRK), *d* (SGR, PRK)] = *d* (SGR, PRK) = 205.
d (KTN/SGR) PNG = *min* [*d* (KTN, PNG), *d* (SGR, PNG)] = *d* (SGR, PNG) = 369 .
d (KED/PLS) TRG = *min* [*d* (KTN, TRG), *d* (SGR, TRG)] = *d* (KTN, TRG) = 168.

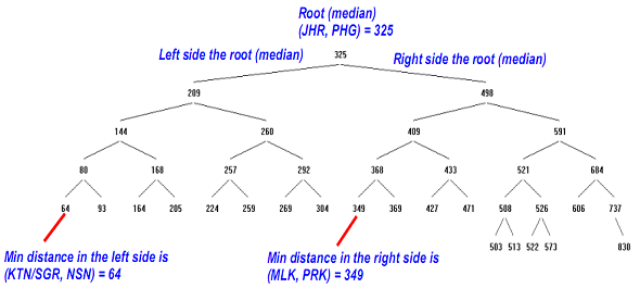

Fig. 3: Second cluster in AVL tree

Based on Figure 3 the root or the median is JHR and PHG, at distance 325. The minimum Left Side pair of states is KTN/SGR and NSN, at distance 64. These are merged into a single cluster called "KTN/SGR/NSN", and the minimum Right Side pair of states is MLK and PRK, at distance 349. The level of the new cluster is L (KTN/SGR/NSN) = 64, L (MLK/PRK) = 349, and the new sequence number is m = 2. In addition the left side and right side elements did not have any same element. Therefore no need to merge the left side cluster and right side cluster. The cluster, side, sequence number, elements and distances are shown in Table 3.

Table 3: Second cluster

Cluster	***Side***	***Sequence No***	***Element***	***Distance***
KTN/SGR/NSN	Left side	2	KTN, SGR and NSN	64
MLK/PRK	Right side		MLK and PRK	349

The distances between this cluster and the remaining elements in the distance matrix are computed as shown below.

d (KTN/SGR/NSN) JHR = *min* [*d*(KTN/SGR, JHR), *d* (NSN,JHR)] = *d*(NSN,JHR) = 304.
d (KTN/SGR/NSN) KTN = *min* [*d*(KTN/SGR, KED/PLS), *d* (NSN, KED/PLS)] = *d* (KTN, KED/PLS) = 409.
d (KTN/SGR/NSN) MLK = *min* [*d* (KTN/SGR, MLK), *d* (NSN, MLK)] = *d*(NSN, MLK) = 80.
d (KTN/SGR/NSN) PHG = *min* [*d* (KTN/SGR, PHG), *d* (NSN, PHG)] = *d*(SGR, PHG) = 259.
d (KTN/SGR/NSN) PRK = *min* [*d* (KTN/SGR, PRK), *d* (NSN, PRK)] = *d* (SGR, PRK) = 205.
d (KTN/SGR/NSN) PNG = *min* [*d* (KTN/SGR, PNG), *d* (NSN, PNG)] = *d* (SGR, PNG) = 369 .
d (KTN/PLS/NSN) TRG = *min* [*d* (KTN/SGR, TRG), *d* (NSN, TRG)] = *d* (KTN, TRG) = 168.

The distances between the right side cluster and the remaining elements in the distance matrix are computed as shown below.

d (MLK/PRK) JHR = *min* [*d*(MLK, JHR), *d* (PRK,JHR)] = *d* (MLK,JHR) = 224.
d (MLK/PRK) KTN = *min* [*d*(MLK, KED/PLS), *d* (PRK, KED/PLS)] = *d* (PRK, KED/PLS) = 257.
d (MLK/PRK) KTN/SGR/NSN = *min* [*d* (MLK, KTN/SGR/NSN), *d* (PRK, KTN/SGR/NSN)] = *d* (MLK, KTN/SGR/NSN) = 80.
d (MLK/PRK) PHG = *min* [*d* (MLK, PHG), *d* (PRK, PHG)] = *d* (MLK, PHG) = 292.
d (MLK/PRK) PNG = *min* [*d* (MLK, PNG), *d* (PRK, PNG)] = *d* (PRK, PNG) = 164 .
d (MLK/PRK) TRG = *min* [*d* (MLK, TRG), *d* (PRK, TRG)] = *d* (PRK, TRG) = 503.

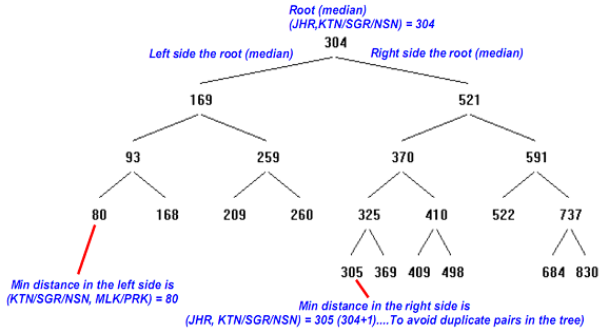

Fig. 4: Third cluster in AVL tree

Based on Figure 4 the root or the median is JHR and KTN/SGR/NSN, at distance 304. The minimum Left Side pair of states is KTN/SGR/NSN and MLK/PRK, at distance 80. These are merged into a single cluster called "KTN/SGR/NSN/MLK/PRK", and the minimum Right Side pair of states is JHR and KTN/SGR/NSN, at distance 305. The level of the new cluster is L (KTN/SGR/NSN/MLK/PRK) = 80, L (JHR/KTNSGR/NSN) = 305, and the new sequence number is m = 3. In addition the left side and right side have same element(s) in both sides. Therefore merge the left side cluster and the right side cluster in one cluster. The level of the new cluster is L (KTN/SGR/NSN/ JHR/MLK/PRK) = 385. The cluster, side, sequence number, elements and distances are shown in Table 4.

Table 4: Third cluster

Cluster	***Side***	***Sequence No***	***Element***	***Distance***
KTN/SGR/NSN/JHR/MLK/PRK	-	3	KTN, SGR, NSN, JHR, MLK, PRK	385

The distances between this cluster and the remaining elements in the distance matrix are computed as shown below.

d (KTN/SGR/NSN/JHR/MLK/PRK) JHR = *min* [*d* (KTN/SGR, JHR), *d* (NSN,JHR)] = *d* (NSN,JHR) = 304.
d (KTN/SGR/NSN/JHR/MLK/PRK) KTN = *min* [*d* (KTN/SGR, KED/PLS), *d* (NSN, KED/PLS)] = *d* (KTN, KED/PLS) = 409.
d (KTN/SGR/NSN/JHR/MLK/PRK) MLK = *min* [*d* (KTN/SGR, MLK), *d* (NSN, MLK)] = d(NSN, MLK) = 80.
d (KTN/SGR/NSN/JHR/MLK/PRK) PHG = *min* [*d* (KTN/SGR, PHG), *d* (NSN, PHG)] = *d*(SGR, PHG) = 259.
d (KTN/SGR/NSN/JHR/MLK/PRK) PRK = *min* [*d* (KTN/SGR, PRK), *d* (NSN, PRK)] = *d* (SGR, PRK) = 205.
d (KTN/SGR/NSN/JHR/MLK/PRK) PNG = *min* [*d* (KTN/SGR, PNG), *d* (NSN, PNG)] = *d* (SGR, PNG) = 369 .
d (KTN/SGR/NSN/JHR/MLK/PRK) TRG = *min* [*d* (KTN/SGR, TRG), *d* (NSN, TRG)] = *d* (KTN, TRG) = 168.

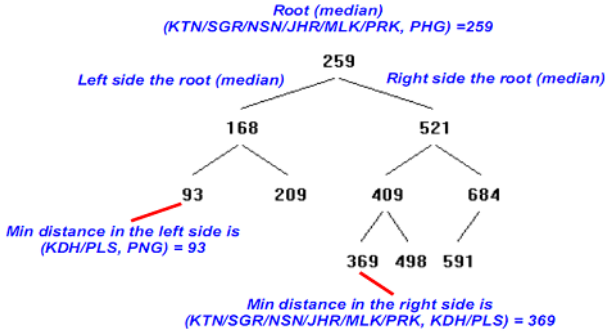

Fig. 5: Forth cluster in AVL tree

Based on Figure 5 the root or the median is KTN/SGR/NSN/JHR/MLK/PRK and PHG, at distance 259. The minimum left side pair of states is KDH/PLS and PNG, at distance 93. These are merged into a single cluster called "KDH/PLS/PNG", and the minimum Right Side pair of states is KTN/SGR/NSN/JHR/MLK/PRK and KDH/PLS, at distance 369. The level of the new cluster is L (KDH/PLS/PNG) = 93, L (KTN/SGR/NSN/JHR/MLK/PRK/KDH/PLS) = 369, and the new sequence number is m = 4. In addition the left side and right side have same element(s) in both sides. Therefore merge the left side cluster and the right side cluster in one cluster. The level of the new cluster is L (KTN/SGR/NSN/JHR/MLK/PRK/KDH/PLS/PNG) = 462. The cluster, side, sequence number, elements and distances are shown in Table 5.

Table 5: Forth cluster

Cluster	*Side*	*Sequence No*	*Element*	*Distance*
KTN/SGR/NSN/JHR/MLK/PRK/KDH/PLS/PNG	-	4	KTN, SGR, NSN, JHR, MLK, PRK, KDH, PLS, PNG	462

The distances between this cluster and the remaining elements in the distance matrix are computed as shown below.

d (KTN/SGR/NSN/JHR/MLK/PRK/KDH/PLS/PNG) PHG = *min* [*d* (KTN/SGR/NSN/JHR/MLK/PRK, PHG), *d* (KDH/PLS/PNG, PHG)] = *d* (KTN/SGR/NSN/JHR/MLK/PRK, PHG) = 259.
d (KTN/SGR/NSN/JHR/MLK/PRK/KDH/PLS/PNG) TRG = *min* [*d* (KTN/SGR/NSN/JHR/MLK/PRK, TRG), *d* (KDH/PLS/PNG, TRG)] = *d* (KTN/SGR/NSN/JHR/MLK/PRK, TRG) = 168.

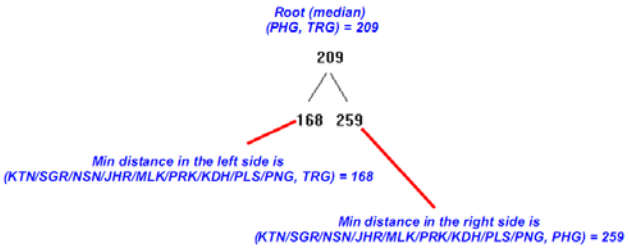

Fig. 6: Fifth cluster in AVL tree

Based on Figure 6 the minimum left side pair of states is KTN/SGR/NSN/JHR /MLK/PRK/KDH/PLS/PNG and TRG, at distance 168. These are merged into a single cluster called "KTN/SGR/NSN/JHR/MLK/PRK/KDH/PLS/PNG/TRG", and the minimum right side pair of states is KTN/SGR/NSN/JHR/MLK/PRK/KDH/PLS/PNG and PHG the level of the new cluster is L (KTN/SGR/NSN/JHR/MLK/PRK /KDH/PLS/PNG/TRG) = 168, and the new sequence number is m = 5. In addition the left side and right side have same element(s) in both sides. Therefore merge the left side cluster and the right side cluster in one cluster. The level of the new cluster is L (KTN/SGR/NSN/JHR/MLK/PRK /KDH/PLS/PNG/PHG/TRG) = 427. The cluster, side, sequence number, elements and distances are shown in Table 6.

Table 6: Fifth cluster

Cluster	*Side*	*Sequence No*	*Element*	*Distance*
KTN/SGR/NSN/JHR/MLK /PRK/KDH /PLS/PNG/ PHG/TRG	-	5	KTN, SGR, NSN, JHR, MLK, PRK, KDH, PLS, PNG,PHG,TRG	168

The final cluster is (KTN/SGR/NSN/JHR/MLK /PRK/KDH/PLS/PNG/PHG/TRG)

Table 7 shows an example of hierarchical clustering of eleven labeled points, namely JHR, KED, KTN, MLK, NSN, PHG, PRK, PLS, PNG, SGR and TRG. The example showing the following sequence of nested partitions:

Table 7: Overall cluster and sequence number

Sequence No	**Root**	**Cluster**		**Overall cluster**
		Left cluster	**Right cluster**	
m1	NSN, TRG	KDH/PLS	KTN/SGR	
m2	JHR, PHG	KTN/SGR /NSN	MLK/PRK	
m3	JHR, KTN/SGR/NSN	KTN/SGR /NSN/MLK/PRK	JHR and KTN/SGR /NSN	KTN/SGR/NSN/ JHR/MLK/PRK
m4	KTN/SGR/NSN/JHR/MLK/PRK,PHG	KDH/PLS/PNG	KTN/SGR /NSN/JHR /MLK/PRK and KDH/PLS	KTN/SGR/NSN/JHR/MLK/PRK/KDH/PLS/PNG
m5	PHG, TRG	KTN/SGR /NSN/JHR /MLK/PRK/KDH/PLS/PNG and TRG,	KTN/SGR /NSN/JHR /MLK/PRK/KDH/PLS/PNG and PHG	KTN, SGR, NSN, JHR, MLK, PRK, KDH, PLS, PNG,PHG,TRG

5. Results and Discussions

From the results of the manual analysis of applying bidirectional agglomerative hierarchical clustering using single-link method on Malaysian states example. Consider the number of objects are n, there are *n/2* (*in the best case*) or *n/2+1* (*in the worst case*) levels. Bidirectional algorithm in each level involves finding a minimum from tree *T* with time complexity O(1), then merge two clusters into a single cluster it need O(1), finally update the proximity tree, *T*, by deleting the nodes corresponding to clusters its need

O(*logn*). Therefore, this approach is more efficient in clustering huge amount of data and the performance of the bidirectional agglomerative hierarchical clustering using AVL tree algorithm is better from the bidirectional agglomerative hierarchical clustering using distance matrix and traditional agglomerative hierarchical clustering algorithms. In fact of the manual analysis on bidirectional agglomerative hierarchical clustering use distance matrix for single-link method. It is obvious *n* data point's need (*n/2*) in the best case or (*n/2+1*) in the worst case steps to merge all data points into one cluster. While the traditional agglomerative hierarchical clustering algorithms need (*n-1*) steps for merging all data points into single cluster. The bidirectional agglomerative hierarchical clustering using distance matrix and as for traditional agglomerative hierarchical clustering algorithms, the dissimilarity matrix *D* has a row and a column for each of the n elements. The overall complexity to merge all data points into single cluster is $O(n^2)$. Therefore the bidirectional use distance matrix and agglomerative hierarchical clustering method is a limitation to handle large datasets within a reasonable time and memory resources.

6. Conclusions

The complexity analysis is the algorithm performance in determining the resources such as execution time and memory usage necessary to execute it. Usually, the complexity of an algorithm is a function related to the input length/size to the number of fundamental steps. This paper proposes a hierarchical algorithm called bidirectional agglomerative hierarchical clustering algorithm based on the AVL tree by clustering the objects in left and right the median/root to enhance the complexity time of the current agglomerative hierarchical clustering algorithms and to reduce the gap between the flooding of information and the current agglomerative hierarchical clustering algorithm. One of the advantages of the proposed bidirectional agglomerative hierarchical clustering algorithm using AVL tree and that of other similar agglomerative algorithm is that, it has relatively low computational requirements. The overall complexity of the proposed algorithm is O(*logn*) and need (*n/2* or *n/2+1*) to cluster all data points in one cluster whereas the previous algorithm is $O(n^2)$ and need (*n-1*) steps to cluster all data points into one cluster.

Appendix

Table 8: Original distances

	JHR	***KDH***	***KTN***	***MLK***	***NSN***	***PHG***	***PRK***	***PLS***	***PNG***	***SGR***	***TRG***
JHR		830	689	224	304	325	573	875	737	368	521
KDH	830	0	409	606	526	684	257	45	93	462	521
KTN	689	409	0	609	538	371	391	454	386	474	168
MLK	224	606	609	0	80	292	349	651	513	144	508
NSN	304	526	538	80	0	259	269	571	433	64	471
PHG	325	684	371	292	259	0	427	729	591	259	209
PRK	573	257	391	349	269	427	0	302	164	205	503
PLS	875	45	454	651	571	729	302	0	138	507	566
PNG	737	93	386	513	433	591	164	138	0	369	498
SGR	368	462	474	144	64	259	205	507	369	0	455
TRG	521	521	168	508	471	209	503	566	498	455	0

Acknowledgments

The authors wish to thank Universiti Sains Islam Malaysia, Faculty of Science and Technology.

References

[1] S. Moran, Y. Hey, and K. Liu. An Empirical Framework for Automatically Selecting the Best Bayesian Classifier. Proceedings of the World Congress on Engineering 2009 Vol I WCE 2009, July 1 - 3, 2009, London, U.K.

[2] S. Kanaujiya. Visual Data Mining. *Proceedings of 2nd National Conference on Challenges & Opportunities in Information Technology (COIT-2008) RIMT-IET, Mandi Gobindgarh. March 29, 2008.*

[3] J. *Gantz. F. 2008.* The diverse and exploding digital universe. Available online at: <http://www.emc.com/collateral/analyst-reports/diverse-exploding-digitaluniverse. pdf>.

[4] G. Bordogna, and G. Pasi. Hierarchical-Hyperspherical Divisive Fuzzy C-Means (H2D-FCM) Clustering for Information Retrieval. IEEE computer society 2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology -Workshops.

[5] M. Halkidi, D. Gunopulos, M. Vazirgiannis, N. Kumar, and C. Domeniconi. A Clustering

Framework Based on Subjective and Objective Validity Criteria. ACM Transactions on Knowledge Discovery from Data (TKDD). Vol 1, No.4, 2008.

[6] A. Jain, Murty, P.J. Flynn. 1999. "Data Clustering: A Review", *ACM Computing Surveys*. Vol. *31*, *No.* 3. 1999, pp.264-323.

[7] C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A Survey of Web Clustering Engines. *ACM* Computing Surveys. Vol. 41, No. 3, 2009.

[8] J. J. Hu, C. G. Tang, J. Peng, C. Li, C. A. Yuan, and A. L. Chen. A Clustering Algorithm Based Absorbing Nearest Neighbors. WAIM 2005, Volume 3739 of Lecture Notes in Computer Science, Springer, 2005, p.p 700-705.

[9] Ke-Bing, Z. 2007. Visual Cluster Analysis in Data Mining. (PhD Thesis). Macquarie University.

[10] K. Alsabti, S. Ranka, and V. Singh. An Efficient K-Means Clustering Algorithm, http://www.cise.ufl.edu/_ranka/, 1997.

[11] M. N. Joshi. Parallel K-Means Algorithm on Distributed Memory Multiprocessors. 2003

[12] L. Liping, and M. Zhi-Qing Meng. A method of choosing the initial cluster centers, Computer Engineering and Applications, pp.179-180.

[13] D. Judd, P. K. McKinley, and A. K. Jain. Large-Scale Parallel Data Clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, AUGUST 1998, pp.871-876.

[14] M. Hemalatha, P. Ranjith Jebah Thangiah, K. Vivekanandan. A Distributed and Parallel Clustering Algorithm for Massive Biological Data. JCIT: Journal of Convergence Information Technology, Vol. 3, No. 4, 2008, pp. 84 -88.

[15] H. Gao, J. Jiang, L. She, and Y. Fu. A New Agglomerative Hierarchical Clustering Algorithm Implementation based on the Map Reduce Framework. International Journal of Digital Content Technology and its Applications, Vol. 4 No. 3, 2010.

[16] L. Kaufman, and P. J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis", John While & Sons.

[17] T. Zhang , R. Ramakrishnan, and M. livny BIRCH: An efficient data clustering method for very large databases. In Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96). 1996, Pp.103–114.

[18] S. Guha, R. Rastogi, and K. Shim. An efficient clustering algorithm for large databases. In Proceedings of SIGMOD, June 1998, pp.73–84.

[19] S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes", In the Proceedings of the IEEE Conference on Data Engineering. 1999.

Hussain Mohammad Abu Dalbouh is a PhD student at the University Science Islam Malaysia (USIM), was born on the 26 of May 1982, his nationality Jordanian. He obtained his Bachelor's degree in Computer Information System in 2005 from the Al Yarmouk University, Jordan. He received his Master's degree in Information Technology from University Utara Malaysia (UUM) in 2009. His interest Areas: Artificial Intelligence (AI), Data Mining (DM), Visualization, Tree data structure, Data structure and algorithms.

Norita Md Norwawi is an Associate Professor at Universiti Sains Islam Malaysia. She obtained her Bachelor in Computer Science in 1987 from the University of New South Wales, Australia. She received her Master's degree in Computer Science from National University of Malaysia in 1994. In 2004, she obtained her PhD specializing in Temporal Data Mining and Multiagent System from University Utara Malaysia. As an academician, her research interests include artificial intelligence, multi-agent system, temporal data mining, text mining, knowledge mining, information security and digital Islamic application and content. Her works have been published in international conferences, journals and won awards on research and innovation competition in national and international level.