Collective, Hierarchical Clustering from Distributed, Heterogeneous Data

Erik L. Johnson[1] and Hillol Kargupta[2]

[1] School of Electrical Engineering and Computer Science
Washington State University
erikj@wsunix.wsu.edu
http://www.eecs.wsu.edu/~ejohnso1
[2] School of Electrical Engineering and Computer Science
Washington State University
hillol@eecs.wsu.edu
http://eecs.wsu.edu/~hillol

Abstract. *This paper presents the Collective Hierarchical Clustering (CHC) algorithm for analyzing distributed, heterogeneous data. This algorithm first generates local cluster models and then combines them to generate the global cluster model of the data. The proposed algorithm runs in $O(|\mathcal{S}|n^2)$ time, with a $O(|\mathcal{S}|n)$ space requirement and $O(n)$ communication requirement, where n is the number of elements in the data set and $|\mathcal{S}|$ is the number of data sites. This approach shows significant improvement over naive methods with $O(n^2)$ communication costs in the case that the entire distance matrix is transmitted and $O(nm)$ communication costs to centralize the data, where m is the total number of features. A specific implementation based on the single link clustering and results comparing its performance with that of a centralized clustering algorithm are presented. An analysis of the algorithm complexity, in terms of overall computation time and communication requirements, is presented.*

1 Introduction

The field of Knowledge Discovery from Data (KDD) emerged in the recent past as a result of the dramatic evolution of the technology for information storage, access, and analysis. The ability of various organizations to collect, store and retrieve huge amounts of data has necessitated the development of algorithms which can extract useful information from these databases. The field of KDD addresses this issue.

Distributed knowledge discovery (DKD) is a result of further evolution of the KDD problem. DKD embraces the growing trend of merging computation with communication. It considers all the new dimensions of the knowledge discovery process in the context of the emerging distributed computing environments. DKD accepts the fact that data may be inherently distributed among different loosely coupled sites connected by a network and the sites may have heterogeneous data. It offers techniques to discover new knowledge through distributed data analysis and modeling using minimal communication of data.

Account number	Amount	Location	Previous record	Unusual transaction
11992346	-42.84	Seattle	Poor	Yes
12993339	2613.33	Seattle	Good	No
45633341	432.42	Portland	Okay	No
55564999	128.32	Spokane	Okay	Yes

Table 1. Homogeneous case: Site A with a table for credit card transaction records.

Account number	Amount	Location	Previous record	Unusual transaction
87992364	446.32	Berkeley	Good	No
67845921	978.24	Orinda	Good	Yes
85621341	719.42	Walnut Creek	Okay	No
95345998	-256.40	San Francisco	Bad	Yes

Table 2. Homogeneous case: Site B with a table for credit card transaction records.

DKD must deal with different possibilities of data distribution. Different sites may contain data for a common set of features of the problem domain. In case of relational data this would mean a consistent database schema across all the sites. This is the homogeneous case. Tables 1 and 2 illustrate this case using an example from credit card transaction domain.[1] There are two data sites A and B, connected by a network. The KDD objective is to find patterns of fraudulent transactions. Note that both the tables have the same schema.

The data sites may also be heterogeneous. In other words, sites may contain data for different features. Let us illustrate this case with relational data. Table 3 shows two data-tables at site X. The table on the left contains weather-related data and the one on the right contains demographic data. Table 4 shows the content of site Y, which contains holiday toy sales data. The objective of the KDD process is to detect relations between the toy sales and the demographic and weather related features.

This paper presents an adaptation of the existing *Single Link Clustering Algorithm* to the heterogeneous case. The proposed algorithm first generates local cluster models of the data at each of the sites, and then transmits these models to a *facilitator* site, which combines the models into a global model. It will be shown that this process can be accomplished in the same asymptotic time as the existing optimal single link clustering algorithms, with a significant savings compared to naive methods in communication costs.

The rest of this paper is arranged as follows. Section 2 provides background information concerning agglomerative hierarchical clustering algorithms and a brief overview of other related efforts, including related work both in terms of clustering algorithms and DKD in general. Section 3 defines the problem of generating local models using known clustering algorithms in a heterogeneous space,

[1] Please note that the credit card domain may not always have consistent schema. The domain is used just for illustration.

City	Temp.	Humidity	Wind Chill
Pullman	20	24%	14
Spokane	32	48%	23
Seattle	63	88%	62
Portland	51	86%	46
Richland	47	52%	41

City	Size	Average earning	Proportion of small business owners
Pullman	Small	Low	0.041
Spokane	Medium	Medium	0.022
Seattle	Large	High	0.014
Portland	Large	High	0.017
Richland	Medium	Medium	0.031

Table 3. Heterogeneous case: Site X with two tables, one for weather and the other for demography.

City	Best Selling Item	Price	Number Items Sold (In thousands)
Spokane	Snarc Action Figure	47.99	23
Pullman	Power Toads	23.50	2
Richland	Light Saber	19.99	5
Seattle	Super Squirter	24.99	142
Portland	Super Fun Ball	9.99	24

Table 4. Heterogeneous case: Site Y with one table holiday toy sales.

and shows the flaws of a naive approach. Section 4 derives a set of bounding equations for generating a global model from a set of locally generated models, and uses these to derive the CHC algorithm. Section 5 examines the time complexity of the CHC algorithm. Section 6 provides empirical results. Ongoing and future work is discussed in Section 7. Finally, section 8 concludes this paper.

2 Background

This section presents a brief overview of agglomerative hierarchical clustering, and single link clustering in particular. Furthermore, this section presents an overview of other related efforts.

2.1 A Brief Overview of Agglomerative Hierarchical Clustering

Hierarchical clustering transforms a set of points, with an associated dissimilarity metric between two given points, into a tree structure, known as a *dendrogram* [1]. The dendrogram represents a sequence of combinations of sets of points when viewed from from the bottom of the tree. The dendrogram can also be viewed as a series of partitionings of the data into clusters when viewed from the top of the tree. Each leaf node of a dendrogram represents a single point from the data set, and each internal node represents a partitioning of the data set into two clusters. An agglomerative algorithm builds the dendrogram from the leaf nodes up, whereas a partitional algorithm builds the tree from the top down.

The algorithm presented in this paper is based upon the single link clustering algorithm, which is an agglomerative hierarchical clustering method.

As noted, in addition to the set of points that form the leaves of the dendrogram, a dissimilarity metric must be selected. The dissimilarity between two points or clusters is frequently measured in terms of a distance between the points or clusters. Common distance metrics used include the Euclidean distance metric and the "city-block" distance metric. The algorithm presented in this paper is based upon the use of the Euclidean distance metric for the measure of dissimilarity; however, it is extensible to other distance metrics as well.

In general, an agglomerative hierarchical clustering algorithm consists of three primary steps [1]:

1. Initialize each individual point as a single cluster.
2. Determine the smallest measure of dissimilarity between two clusters, and merge these clusters.
3. Continue to merge the two least dissimilar clusters until all points are contained in a single cluster.

The algorithm presented in this paper focuses on single link clustering, also referred to as *nearest neighbor* clustering. This method combines two sub-clusters into a single cluster by choosing the two remaining unconnected clusters that have the shortest distance as measured by the closest pair of points between the two clusters. In the case of single link clustering, the distance between two individual sub-clusters that have been merged is stored as the internal nodes of the dendrogram.

There are a number of other related methods of clustering that differ primarily in how the distance between points within a cluster and between two separate clusters is measured. An overview of various methods of hierarchical agglomerative clustering can be found in [1].

Once the dendrogram is generated, a level of dissimilarity is chosen. All subtrees that contain either a single point or a dissimilarity measure which is less than the chosen level of dissimilarity, and whose parent node has a measure of dissimilarity greater to or equal to the chosen level of dissimilarity become individual clusters. Note that once the dendrogram has been generated, this chosen level of dissimilarity may be changed to increase and decrease the granularity of the clusters.

As an example of monolithic single link clustering, let us consider a data set consisting of set of six points, p_1 through p_6, each described by four features f_1 through f_4, as shown in Figure 2.1. Note that the values of each feature are assumed to have the same significance.

Using the Euclidean distance metric, we find that the points p_1 through p_6 have the distances as shown in Figure 2.1. Note that for large data, we would not compute these distances and store them in a matrix such as shown, as this would entail an n^2 memory requirement; however, it is useful for the purpose of illustration.

To apply single link clustering, we would first initialize each point as an individual cluster. The smallest amount of dissimilarity between two clusters at

	Feature			
Point	f_1	f_2	f_3	f_4
p_1	1.00	1.00	1.00	1.00
p_2	1.25	1.00	1.25	1.25
p_3	1.00	1.25	2.00	2.00
p_4	3.00	3.25	2.25	2.25
p_5	3.25	3.00	3.00	3.00
p_6	3.25	3.50	3.25	3.25

Fig. 1. *Example data set consisting of 6 points, each described by four features*

	p_1	p_2	p_3	p_4	p_5	p_6
p_1	0.000	0.433	1.436	3.491	4.131	4.630
p_2	0.433	0.000	1.118	3.182	3.758	4.272
p_3	1.436	1.118	0.000	2.850	3.182	3.640
p_4	3.491	3.182	2.850	0.000	1.118	1.458
p_5	4.131	3.758	3.182	1.118	0.000	0.612
p_6	4.630	4.272	3.640	1.458	0.612	0.000

Fig. 2. *Distance matrix for points p_1 through p_6 as described in Figure 2.1, using the Euclidean distance metric.*

this point is between points p_1 and p_2. We would therefore join these two clusters and set the distance of the connecting node to 0.433. At this point, the smallest distance between two clusters is now between the single point clusters consisting of points p_1 and p_2, and that that distance is 0.621. We now find that the two pairs of clusters consisting of $\{\{p_1, p_2\}, \{p_3\}\}$ and $\{\{p_4\}, \{p_5, p_6\}\}$ both have a distance of 1.118, and we therefore combine these clusters. Finally, we find that the two clusters consisting of $\{p_1, p_2, p_3\}$ and $\{p_4, p_5, p_6\}$ have a shortest distance (in this case, between points p_3 and p_4, have a distance of 2.850. We merge these final two clusters, and our dendrogram is complete. By following these steps, the dendrogram shown in Figure 3 is generated.

The decision of what level of dissimilarity should be used to determine which internal nodes of the dendrogram, representing the partitioning of the data, should become the actual clusters is not firmly defined; rather, it is somewhat a matter of heuristics and a matter of the desired granularity. For the purpose of this example, we will choose a level of dissimilarity somewhere between 1.118 and 2.850, as shown by the heavy dashed line in Figure 3. This results in two clusters, consisting of points $\{p_1, p_2, p_3\}$ in the first cluster, and points $\{p_4, p_5, p_6\}$.

The monolithic single link clustering algorithm has been well known for a number of years. An early version, SLINK, presented in [2], is still considered to be the standard of comparison. The SLINK algorithm runs in $O(n^2)$ time and $O(n)$ space.

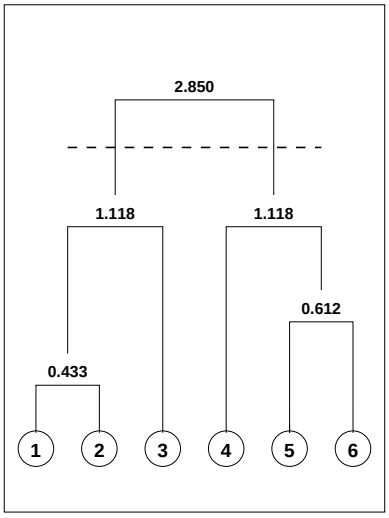

Fig. 3. *Dendrogram generated from the monolithic dataset shown in Figure 2.1.*

2.2 Related Work

There are numerous recent efforts directed towards scaling up clustering algorithms in order to allow practical use with the huge data sets commonly associated with KDD. In [3], an adaptation of the K-Means clustering algorithm is presented that is able to cluster huge monolithic data sets with a single scan of the data. The authors show that their approach is both faster and more accurate than a sampling based approach. Another approach toward scaling up clustering algorithms is presented in [4]. The authors present an algorithm applicable to clustering in a KDD environment, which in addition to being scalable, is well suited to handling noise in the dataset. Another such approach is presented in [5]. Here the authors present the CLARANS algorithm, that is also intended to scale up clustering algorithm for KDD applications. CURE, and algorithm which utilizes a combination of both random sampling and partitioning for clustering in the KDD domain is presented in [6].

A number of recent efforts have proposed parallel based approaches to the problem of scaling clustering algorithms up for use in KDD environments. In [7], the author shows an adaptation of the SLINK [2] and other agglomerative hierarchical clustering algorithms to a multiprocessor environment to parallelize the clustering process. In [8], the authors adapt the K-Means algorithm to run in a parallel environment.

The PADMA system [9, 10] achieves scalability by locating agents with the distributed data sources. An agent coordinating facilitator gives user requests to local agents that then access and analyze local data, returning analysis results to the facilitator, which merges the results. The high level results returned by the

local agents are much smaller than the original data, thus allowing economical communication and enhancing scalability. The authors report on a PADMA implementation for unstructured text mining.

There exists very little literature for analyzing data from heterogeneous sites. Learning from heterogeneous data sites is discussed in [11] from the perspective of inductive bias. This work notes that such partitioning of the feature space can be addressed by decomposing the problem into smaller sub-problems when the problem is site-wise decomposable. The WoRLD system [12] addressed the problem of concept learning from heterogeneous sites by developing an "activation spreading" approach. This approach first computes the cardinal distribution of the feature values in the individual data sets. Next, this distribution information is propagated across different sites. Features with strong correlations to the concept space are identified based on this first order statistics of the cardinal distribution. The selected features are used for learning the appropriate concept. Since the technique is based on the first order statistical approximation of the underlying distribution, it may not be appropriate in general for non-convex concept space. The propagation of marker activation records from one site to another is accomplished through basic database operations. This makes the approach easily implementable in database systems. Nevertheless, a general methodology for learning functions from distributed, heterogeneous data sites with guaranteed control of accuracy and minimal communication overhead is still an open issue.

In the recent past Kargupta and his colleagues [13] considered this case and proposed the *collective data mining* (CDM) framework that makes use of orthonormal basis functions for correct local analysis. This work describes the Collective Data Mining methodology that can learn different popular data models such as regression, decision trees in a distributed environment. Interested readers may refer to [14] for additional experimental and theoretical analysis of this framework. The main motivation behind this framework is that direct application of existing machine learning and statistical algorithms to local data sites may produce partials models that are completely incorrect and possibly ambiguous. CDM makes use of orthonormal representations to guarantee accurate learning of local models and their subsequent aggregation.

To the best knowledge of the authors, there exists no published work on distributed clustering techniques for heterogeneous data sites.

The following section defines the problem of hierarchical clustering in a heterogeneous environment.

3 Problem Definition

Suppose we have some set of sites $\mathcal{S}$, each of which has access to some subset of the total number of features $\mathcal{F}$, our goal is to generate a global model that is equivalent to the model that would be generated if the data were centralized; that is, we wish our dendrogram to have the same structure. In general, there

are four criteria that are critical to the generation of a global model from some set of local models:

1. The dendrogram representing global model should have the same general structure as the dendrogram that would be generated if the data set was centralized and a known clustering algorithm run on this centralized data.
2. As an added constraint, it is not acceptable to transmit the entire dataset to a single site for processing; rather, local models could be generated at each of the $|\mathcal{S}|$ sites, and these local models transmitted to a *facilitator* site, which will generate our global model. Obviously, it is not appropriate to transmit the entire dissimilarity matrix, as the cost of such transmission would be $O(|\mathcal{S}|n^2)$.
3. Furthermore, asymptotic time complexity of the algorithm should be less than or equal to the asymptotic time complexity of generating the model on a single site from monolithic data.
4. Finally, the global model should not only indicate the sets of clusters and their membership, but also should have a metric of the dissimilarity between the points and between the clusters. We will show that there exists a naive method of generating a global model that meets some of the above requirements, but gives no information about the proximity of the points.

We shall assume in the following discussion that there is some unique key associated with each element in the overall dataset that is accessible to each of the local sites. For example, if a local telephone company and an insurance company wished to build local models on of their customers in a given area, a good choice for a unique key to identify members the elements of the dataset (customers) might be the phone number. Given that there exists a unique key of this nature to identify the rows of the database, there also must exit some function $f(key)$ that, given the set of keys, will return an linear enumeration between [0, n-1] of the keys. This fact will be utilized in the following algorithm.

As an example, suppose there are two sites, $\mathcal{S} = \{s_1, s_2\}$. We will use the data set from the previous example, shown in Figure 2.1. Suppose site s_1 has access to the subset $\{f_1, f_2\}$ of the overall feature set $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$, and s_2 has access to the subset $\{f_3, f_4\}$.

If the single link clustering algorithm is applied at each of the two sites s_1 and s_2, with no interaction or information sharing between the two sites, the dendrograms shown in Figures 4 and 5 would be generated.

In this example, a dissimilarity between 0.354 and 2.282 was chosen for site s_1, and between 0.354 and 1.061, shown as a bold dashed line in figures 4 and 5. Heuristically, this is somewhat a natural place to split the dendrogram into the clusters, since this is the point where the dissimilarity between the clusters begins to increase rapidly. Note that this is a common heuristic for choosing the point at which to divide the dendrogram into the various clusters. Given this choice, site s_1 has two clusters, $c_{1,1} \in \{p_1, p_2, p_3\}$ and $c_{1,2} \in \{p_4, p_5, p_6\}$. In this case, the local model agrees with our dendrogram generated from the monolithic data, with the exception that there is a difference in the hierarchical structure of cluster

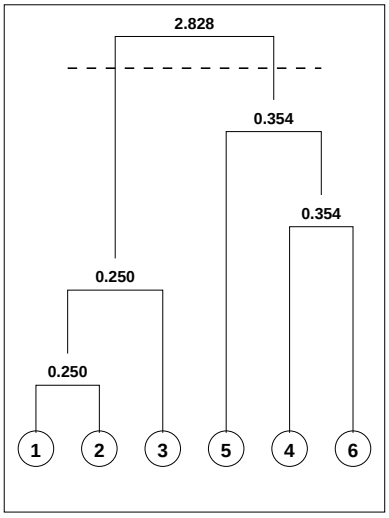

Fig. 4. *Dendrogram generated at site s_1, with access to features $\{f_1, f_2\}$ from the data set shown in Figure 2.1.*

$c_{1,2}$. The dendrogram generated at site s_2, however, is significantly different from the monolithically generated dendrogram. At site s_2, three clusters have been generated: $c_{2,1} \in \{p_1, p_2\}$, $c_{2,2} \in \{p_3, p_4\}$ and $c_{2,3} \in \{p_5, p_6\}$ The choice of dissimilarity used to split the dendrogram is insignificant; there is no choice of the measure of dissimilarity that will result in a clustering such as was generated from the monolithic data. In other words, no matter where the dendrogram is split up to generate the boundaries between clusters, the membership of the clusters will be different from the membership of the clusters in the monolithically generated model.

Once the local models have been generated, we wish to transmit these models to a central facilitator site in order to combine them into a global model. One possible solution to this problem would be to transmit the cluster membership lists to the facilitator, that in turn would take the element-wise intersection of all of the clusters between each site. In the above example, this would result in a total of four clusters:

$$\{\{p_1, p_2, p_3\}, \{p_4, p_5, p_6\}\} \cap \{\{p_1, p_2\}, \{p_3, p_4\}, \{p_5, p_6\}\} =$$

$$\{\{p_1, p_2\}, \{p_3\}, \{p_4\}, \{p_5, p_6\}\}$$

Note that this approach does indeed generate a set of clusters that, given a choice of a level of dissimilarity between 0.612 and 1.118 could have been generated by the monolithic model. Thus, the first criteria is partially met; the clusters generated from the local models are indeed a subset of those generated by the monolithically generated model. However, we do not have a dendrogram

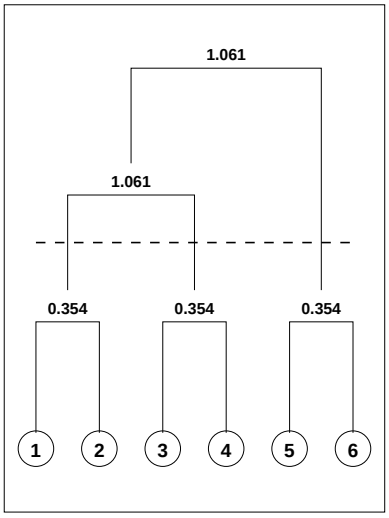

Fig. 5. *Dendrogram generated at site s_2, with access to features $\{f_3, f_4\}$ from the data set shown in Figure 2.1.*

representing the structure of the clustering. The second criteria has also been met; the transmission cost for such a scheme is $O(n)$, as all that has to be transmitted are the unique keys of the points. Likewise, the third criteria has been met, as the cost of generating the local models is identical to the cost of generating the monolithically generated model; in fact, this scheme allows for parallelization of the clustering, although, there is the added cost of the intersection of the clusters. However, the fourth criteria is not met in any way; there is no more information that can be gained from the global model except which points fall into which clusters. Therefore, this scheme is not adequate given the stated criteria for generating a global model from the local models. Therefore, we will examine a technique for bounding the dissimilarity between two points and/or clusters given two or more dendrograms. The following section derives an algorithm that meets all of the listed criteria.

4 Collective Hierarchical Clustering

In this section, we will derive a lower and upper bound for the distance between two points given only the set of local models. Furthermore, we will show how these bounds can be used to generate a global model of the data. Finally, we will provide a specific implementation of the CHC algorithm that meets all of the specified criteria.

4.1 Bounding of Euclidean Distances Between Points

For the following discussion, we will assume that the Euclidean distance metric will be the measure of dissimilarity (see Section 7 for a discussion on ongoing work concerning other distance metrics). Furthermore, we will also assume that any necessary feature selection and scaling of the features have already been performed.

Given two points, p_1 and p_2, with $m = |\mathcal{F}|$ features each consisting of $\mathcal{F} \in \{f_1, f_2, ...f_n\}$, the Euclidean distance between these two points is defined as:

$$dist_{actual}\left(p_1, p_2\right) = \sqrt{\sum_{i=1}^{m} \left(f_{p1,i} - f_{p2,i}\right)^2} \tag{1}$$

This can be expanded as follows:

$$dist_{actual}\left(p_1, p_2\right) = \sqrt{\sum_{i=1}^{m} \left(f_{p1,i}^2 - 2 f_{p1,i} f_{p2,i} + f_{p2,i}^2\right)} \tag{2}$$

$$dist_{actual}^2\left(p_1, p_2\right) = \sum_{i=1}^{m} \left(f_{p1,i}^2 - 2 f_{p1,i} f_{p2,i} + f_{p2,i}^2\right) \tag{3}$$

Hence, if the distance matrix generated at each local site were transmitted to the facilitator site responsible for generating the global model, the distances could be approximated by:

$$dist_{approx}\left(p_1, p_2\right) = \sqrt{\sum_{i=1}^{m} \left(f_{p1,i}^2 + f_{p2,i}^2\right)} \tag{4}$$

and the clustering algorithm being used could be applied to the resulting approximate distances. However, this approach does not take into account the cross terms, which in many cases are significant. Furthermore, this approach would require the transmission of the distance matrices generated at each of the $|\mathcal{S}|$ local sites, which would have a transmission cost of $O\left(|\mathcal{S}|n^2\right)|$, not to mention that these matrices can very easily be prohibitively large to be generated at the local sites, let alone gathered at the facilitator site.

We can, however, given a dendrogram in which the leaves contain only the unique key identifying the point that is represented by the leaf, generate both a minimum and a maximum bound of the distance between the two points. Noting that the transmission cost is $O(n)$ to send each of the locally generated models to the facilitator (see Section 5), such a bounding of the actual distance would allow the generation of a global model from the local models with the desired dissimilarity information and within the transmission cost constraints.

If we have more than one site, each of which has some subset of the features, we can rewrite Equation 1 as follows:

$$dist_{actual}\left(p_1, p_2\right) = \sqrt{\sum_{j=1}^{|\mathcal{S}|}\left(\sum_{i \in \mathcal{F}_j}\left(f_{p1,i} - f_{p2,i}\right)^2\right)} \tag{5}$$

We wish to find an upper and a lower bound for the $|\mathcal{S}|$ individual summations in Equation 5.

Given any two points, the shortest distance between these two points is represented in the dendrogram as the distance value stored in the lowest root of the subtree connecting the two leaves. This follows from the definition of how the dendrogram is generated.

Given a single dendrogram representing a single link clustering, the shortest distance between two points is represented in the dendrogram as the value stored the lowest common subroot connecting the the two leaves that represent the two points. Hence, we can place a lower bound for each of the $|\mathcal{S}|$ summation terms in Equation 5. Therefore, we may state that:

$$\begin{aligned} dist_{actual}\left(p_1, p_2\right) &= \sqrt{\sum_{j=1}^{|\mathcal{S}|}\left(\sum_{i \in \mathcal{F}_j}\left(f_{p1,i} - f_{p2,i}\right)^2\right)} \\ &\geq \sqrt{\sum_{j=1}^{|\mathcal{S}|} d^2_{commonroot,j}} \end{aligned} \tag{6}$$

where $d_{commonroot,j}$ is the distance value stored in the lowest connecting subroot of j^{th} locally generated dendrogram. This leads directly to the equation for the lower bound of the distances:

$$dist_{min}\left(p_1, p_2\right) = \sqrt{\sum_{j=1}^{|\mathcal{S}|} d^2_{commonroot,j}} \tag{7}$$

In order to generate an upper bound on the distance between two given points in a single dendrogram, we observe that the maximum possible distance between these two points is the sum of the distances on the shortest path connecting the two leaf nodes of the dendrogram. In the case that the path connecting two leaf nodes is of length one, this is obvious. However, this is not immediately intuitive when the path is longer. We will show by example why this is the case.

Consider the set of points $\{A, B, C, D\}$ shown in Figure 6. Let $\overline{AB} < \overline{CD} < \overline{CB}$. This would result in the dendrogram shown in Figure 7. Say that we were trying to find the maximum distance possible between points between points A and D with only the information provided in the associated dendrogram. Certainly, we know precisely the distance between A and B, and also, between C and D. We also know the minimum of of the four distances $\overline{AC}$, $\overline{AD}$, $\overline{BC}$ and $\overline{BD}$, represented by $\overline{BC}$ in the dendrogram. However, we do not know that this distance is in fact $\overline{BC}$. For example, the distance represented as $\overline{BC}$ in the

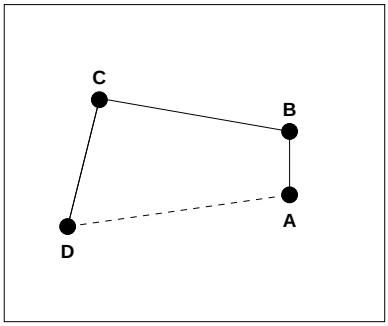

Fig. 6. *The set of points* $\{A, B, C, D\}$*, with* $\overline{AB} < \overline{CD} < \overline{CB}$.

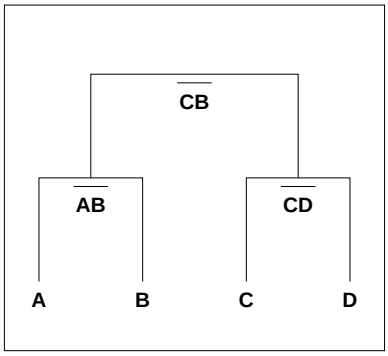

Fig. 7. *The dendrogram representing the points shown in Figure 6.*

dendrogram might actually be $\overline{AD}$, as shown by the dashed line in Figure 6. Now, if we were attempting to put an upper bound on the distance between A and D, given only the information in the dendrogram, it would make sense to sum up the distances along the shortest path in the dendrogram given the relative positions of the points in this example, as in the worst case, the points would actually lie on a straight line. However, if we were trying to find the upper distance bound between B and C, which is actually equal to the distance represented at the highest level of the dendrogram, we would still have to assume the worst case, that the points were in the straight line consisting of the sequence $\{C, D, A, B\}$. Therefore, we can bound the maximum possible distance between two points in the dendrogram as being less than the sum of the distances on the shortest path connecting the two leaf nodes of the dendrogram.

Now, consider that we have more than one dendrogram, and wish to find a total maximum distance between two points. In a manner similar to that used for the lower distance bound, we can place an upper bound on the total distance as follows:

$$dist_{actual}\left(p_1, p_2\right) = \sqrt{\sum_{j=1}^{|\mathcal{S}|}\left(\sum_{i \in \mathcal{F}_j}\left(f_{p1,i} - f_{p2,i}\right)^2\right)}$$

$$\leq \sqrt{\sum_{j=1}^{|\mathcal{S}|}\left(\sum_{shortpath,j} d_{pathnode}\right)^2} \tag{8}$$

where $shortpath, j$ is the shortest connecting path between the two leaf nodes representing the points in j^{th} locally generated dendrogram, and $d_{pathnode}$ is the distance contained in a given node along that path. Hence, the upper distance bound is defined as:

$$dist_{max}\left(p_1, p_2\right) = \sqrt{\sum_{j=1}^{|\mathcal{S}|}\left(\sum_{shortpath,j} d_{pathnode}\right)^2} \tag{9}$$

Hence, a lower and upper bound have been established for the distances between any two given points when the only information available are the locally generated dendrograms, each of which have only the unique keys associated with the points for data in the leaf nodes of the dendrogram which represent those points. The next step in the process is to use these bounds to build a global dendrogram from the locally generated dendrograms that meets the stated criteria for the global model.

4.2 Generation of the Global Model

Once the local models have generated dendrograms based upon the available features at each of the local sites and have transmitted these to the facilitator site, the facilitator is responsible for combining these locally generated models into a global model. In order to do this, the facilitator will use the bounded distances described previously and some function $f(min, max)$ to generate a metric of dissimilarity between the points, and, given this metric, will build the global dendrogram.

There are a number of possible choices for the function $f(min, max)$ to generate the dissimilarity metric (see Section 7 concerning future work for more details). One choice for the dissimilarity metric is the simple mean of the minimum and the maximum distances. While this will not result in an exact equivalent measure of dissimilarity between a dendrogram generated from a monolithic data set and the global dendrogram generated from some set of local models, it will give a reasonably good approximation of the relative dissimilarities between two given points. In the case of the Euclidean distance metric, using the average of the minimum and maximum possible distances between the points yields the dissimilarity function of:

$$dist_{ave}\left(p_1, p_2\right) = \frac{\sqrt{\sum_{j=1}^{|\mathcal{S}|} d^2_{commonroot,j}} + \sqrt{\sum_{j=1}^{|\mathcal{S}|} \left(\sum_{shortpath,j} d_{pathnode}\right)^2}}{2} \tag{10}$$

After the function $f(min, max)$ has been chosen, the facilitator must recluster the points in the dataset using $f(min, max)$ as the measure of dissimilarity. The resulting dendrogram is the global model of the data.

4.3 CHC - Collective, Hierarchical Clustering from Distributed, Heterogeneous Data

We are now ready to state the general form of the CHC algorithm:

1. At each local site, apply the chosen hierarchical clustering algorithm to the dataset and generate a local dendrogram.
2. Transmit the locally generated dendrograms to the facilitator site.
3. Using a statistic based on the bounds (e.g., average), generate the global dendrogram.

The following discussion concerns itself with one possible implementation of the CHC algorithm.

4.4 Implementation of the CHC Algorithm for Single Link Clustering

In order to generate the global model from the local models in a reasonable amount of time and given a memory constraint, we will take advantage of the *reducibility property* [15], which allows us to perform clustering in $O(n^2)$ time and in $O(n)$ space. The reducibility property requires that when two clusters i and j are agglomerated, the new cluster $i + j$ cannot be any closer to any other clusters than either i or j were. This property is satisfied in the case of single link clustering [7]. As a consequence of the reducibility property being satisfied, we need not store the entire $n \times n$ distance matrix containing every distance between every pair of points, but, rather, only two one-dimensional arrays of length n. Each element of the first of these arrays, $\mathcal{N}$, is initialized such that element i contains the key of the closest other point (cluster) j. Each element of the second of these arrays, $\mathcal{D}$, is initialized such that element i contains the associated distance $dist(i, j)$ for the key of j stored in $\mathcal{N}(i)$.

Determining the minimum distances does not require that the entire $n \times n$ distance matrix is generated and stored all at one point in time; rather, only one row of this matrix needs be generated at any point in time. Once this row is generated, the minimum distance between the point represented by the index of the row and all points such that their key is less than the key of the current row is found, and this value is used for the appropriate cells in $\mathcal{D}$ and $\mathcal{N}$. The algorithm for generating $\mathcal{D}$ and $\mathcal{N}$ is shown in Figure 8.

```
(0)  Function BuildLocalModel( input: LocalModels )
(1)    For each {i : 0 ≤ i < n}
(2)      For each {k : 0 ≤ k < |s|}
(3)        Traverse tree k to compute upper and lower distance bounds
            between point i and each other point j, j < i
            Store as:
            LowerBound(j, k) ← lower bound between i, j
            UpperBound(j, k) ← upper bound between i, j
(4)       Apply Equation 10 to determine average distance metric for
           each (j, k)
           While doing so track the minimum distance and store as:
           N(i) ← key of the closest point
           D(i) ← distance to closest point
(5)  Initialize array Π(n) to be pointers to each of the initial
      clusters, e.g., the set of n points
(6)  Repeat n − 1 times
(7)    Cluster1 ← indexOf(min(D))
(8)    Cluster2 ← N(Cluster1)
(9)    Agglomerate( Cluster1, Cluster2 )
(10)    Update Π, N and D as necessary
(11) End
```

Fig. 8. *An efficient algorithm for building the global model from the local models.*

In order to demonstrate how the CHC algorithm may be applied to single link clustering, we will continue with the example started in Section 2.1. Recall that the features are heterogeneous, , $\mathcal{S} = \{s_1, s_2\}$, with the feature values given in Figure 2.1, and that site s_1 has access to the subset $\{f_1, f_2\}$ of the overall feature set $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$, and s_2 has access to the subset $\{f_3, f_4\}$.

First, each of the local sites s_1 and s_2 generates a dendrogram, using the single link clustering algorithm, based upon the feature information available at that site, as shown in Figures 4 and 5. These dendrograms are transmitted to the facilitator site.

Using the dissimilarity metric as given in Equation 10, the dendrogram shown in Figure 9 is generated from the two local models, using the given dissimilarity metric and the single link clustering algorithm.

The overall structure of the globally generated dendrogram is similar to the dendrogram generated from the monolithic data shown in figure 3. However, it should be noted that the dissimilarity metric stored in the internal nodes is an approximation of the dissimilarity between the partitions. In fact, the only case where an exact measure of the distance can be generated from the local models is the case where the length of the path between two given nodes in the local models is exactly one, as is the case between points p_1 and p_2. However, the measure of dissimilarity given does provide enough information to determine a level at which to split the global dendrogram into the individual clusters. In this case, the measure of dissimilarity is chosen to be between 2.017 and 3.326, as,

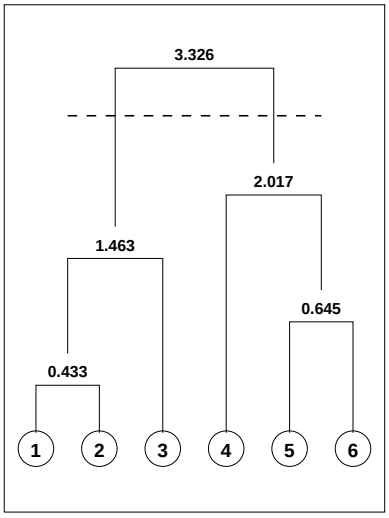

Fig. 9. *Global dendrogram as generated by the facilitator from data available from local dendrograms generated at sites s_1 and s_2, shown in Figures 4 and 5.*

heuristically, this is the point at which the level of dissimilarity rises most sharply. Using this level of dissimilarity to determine the individual cluster membership, the global model in this example results in two clusters, one containing the set of points $\{p_1, p_2, p_3\}$, the other containing the set of points $\{p_4, p_5, p_6\}$. This result is the desired one: the dendrogram generated from the local models is approximately equivalent to the dendrogram generated from the monolithic data set as shown in Figure 3. Furthermore, while not exact, the dissimilarity values stored in the internal nodes is allowed a division of the points into individual clusters without explicit knowledge of the exact coordinates of the points that is equivalent to the division of the points performed on the monolithically generated dendrogram. It should also be noted that, given the similarity of the structures of the two dendrograms, that if the dendrogram is divided at some lower point, for example between 0.612 and 1.118 for the monolithically generated dendrogram and between 0.645 and 1.463 for the global model that was built from the local models, the cluster membership will remain the same for both clusterings.

The CHC algorithm shown in in Figure 8 therefore meets the first requirement; that is, that the general structure of the global dendrogram generated from the local models is that of the dendrogram which would have been generated had the data been transmitted to a single site. Furthermore, the the global model meets the forth requirement, that the global model not only indicates the sets of clusters and their membership, but also has a measure of dissimilarity between the points and between the clusters. In the next section, the time complexity of the CHC algorithm is analyzed, and we will show that the second and third requirements for the algorithm are also met.

5 Time Complexity and Transmission Costs

There are two issues involved in the computation of the time complexity: 1. the cost of transmission of the local models to the facilitator site, and, 2. the computational cost of generating the global dendrogram from the locally generated dendrograms.

The first cost analysis, that of transmission of the local models to the facilitator site, turns out to be rather straight forward. Given the nature of the tree that represents the dendrogram, with a total of n points represented as the leaves of the tree, it follows that there will always be $n-1$ leaves. As it is not necessary to transmit any information about the points represented as the leaves, and as the keys of the points have been structured such that the leaves are linearly enumerated, the number of features is not relevant to the cost of transmitting the local models.

If the nodes of the tree are enumerated using a standard in-order (left-node-right) traversal of the locally generated dendrogram, the tree can be flattened into an array of length $2n-1$. This format is well adapted to transmission. When the tree is flattened, each internal node will be represented as a cell in the array, with four elements: 1. the node's number, 2. the node number of the left child, 3. the node number of the right child, and 4. the distance associated with the node. Each leaf node will contain the node number and the key, which will be used to position the node properly in the array representing the base of the dendrogram.

Given that there are $2n-1$ elements in this array, and that each element in the array contains at most 4 items, the cost of transmission of a given local model to the facilitator site is given by:

$$CT_{local} = O\left(4 \times (2n-1)\right) = O\left(n\right)$$

Therefore, the overall cost of transmission for all of the local models to the facilitator site is given by:

$$CT_{total} = O\left(|\mathcal{S}|n\right)$$

Which is effectively $O(n)$, when $|\mathcal{S}|$ is a constant and $|\mathcal{S}| \ll n$.

In order to evaluate the time and space complexity of the CHC algorithm, we first note that there are two primary loops, one represented by lines (1) through (4) in Figure 8, the other represented by lines (6) through (10). We will examine each of these separately. Line (5) is an $O(n)$ operation, and is not consequential to the overall analysis, as will be shown in the following discussion.

The first line of the first loop, line (1), iterates over each of the points in the tree in order to find the closest other point such that the key (represented by the index) of the other point is less than that of the point under consideration. Obviously, we must perform this n times; hence, the outer loop has a time complexity of $O(n)$.

Line (2) is a nested loop within the loop which began on line (1). As we must extract information from each of the local models, this will add a factor of $O(|\mathcal{S}|)$ to the overall time complexity of the first loop.

Line (3) is the process by which the bounds for the distances are determined. It is only necessary to traverse the tree once in order to determine the distance bounds between a given leaf node (point) and all other points. This traversal is an $O(n)$ operation. As this traversal is nested within the loop began on line line (2), this adds a further factor of $O(n)$ to the overall time complexity. Hence, the overall complexity of the loop began on line (2) is $O(|\mathcal{S}|n)$. Without considering line (4), then, the time complexity of the loop began on line (1) is then $O(|\mathcal{S}|n^2)$

Finally, the first primary loop contains line (4), which, it should be noted, is not contained within the loop defined on line (2). Hence, as the process of finding the minimum dissimilarity metric as defined in Equation 10 for each of the $O(n)$ elements for the under consideration, the components of which are stored in $O(|\mathcal{S}|)$ arrays of length $O(n)$ is itself $O(|\mathcal{S}|n)$. Therefore, line (4) adds nothing further to the overall time complexity of the loop began on line (1). Thus, the time complexity of the first primary loop, which began on line (1), remains $O(|\mathcal{S}|n^2)$.

Note that to this point, the total memory usage involves the following data structures:

- $2 \times |\mathcal{S}|$ reusable arrays to store the upper and lower bounds of the distances between the current point under consideration and those points whose index is less than the index of the point currently under consideration. Each of these arrays uses $O(n)$ space. Hence, the space complexity of these arrays is $O(|\mathcal{S}|n)$.
- $2 \times |\mathcal{S}|$ tree structures representing the locally generated models, each of size $2n - 1$. The space complexity for these data structures is thus $O(|\mathcal{S}|n)$.
- The arrays $\mathcal{N}$ and $\mathcal{D}$. These arrays are both of length n, and thus, have an overall space complexity of $O(n)$.

Thus, the overall space complexity of algorithm through the termination of the first primary loop, which includes lines (1) through (4) is $O(|\mathcal{S}|n)$.

Recall that line (5) is an $O(n)$ operation, and is not consequential to the overall analysis, as we have already determined that the time complexity to this point in the algorithm is of $O(|\mathcal{S}|n^2)$. Furthermore, the space involved on line (5) is $O(n)$, which is less than the $O(|\mathcal{S}|n)$ needed for the first primary loop.

The second primary loop begins on line (6) and includes the rest of the algorithm. Note that this section of the algorithm is an adaptation of the SLINK algorithm [2]. The loop that begins on (6) repeats $O(n)$ times.

Line (7) is a entails a traversal of the array $\mathcal{D}$. Note that the index of the array is the key of the point with which the element value is associated; therefore, it is not possible to sort this array and also maintain the $O(1)$ lookup time in line (8). As line (7) is nested within the second primary loop which began on line (6), and line (7) entails an $O(n)$ operation, the overall time complexity of the second primary loop to this point is $O(n^2)$.

As noted, line (8) is an $O(1)$ operation, and, therefore, adds nothing to the overall time complexity of the second portion of the algorithm.

By using the pointer representation for maintaining the current highest levels of agglomeration in the array Π, the agglomeration in line (9) becomes an $O(1)$

operation. This is also true of the updates necessary in line (10). (See [2] for details). Hence, neither of these lines add any significant amount of time to the overall time complexity of the second primary loop, and, hence, this loop remains with a time complexity of $O(n^2)$.

The space complexity of the second loop is easily shown to be $O(n)$. As the only new data element being utilized is the tree which is built using the Π array, and this dendrogram will contain, including the base nodes as initialized in the Π array, $2n-1$ elements, the overall space complexity of the second portion of the algorithm is $O(n)$.

Given that the first loop was shown to have a time complexity of $O(|\mathcal{S}|n^2)$, line (5) a time complexity of $O(n)$, and the second loop to have a time complexity of $O(n^2)$, our overall time complexity is given by:

$$O(|\mathcal{S}|n^2) + O(n) + O(n^2) = \tag{11}$$

$$O(|\mathcal{S}|n^2) \tag{12}$$

When $|\mathcal{S}|$ is a constant and $|\mathcal{S}| \ll n$, we can rewrite Equation 12 as:

$$time = O(n^2) \tag{13}$$

This time complexity is of the same order as that shown for the SLINK algorithm [2], and, hence, is considered to be the optimal possible.

Furthermore, the overall space complexity for the first primary loop was shown to be $O(|\mathcal{S}|n)$, and $O(n)$ for the second. Once again, when when $|\mathcal{S}|$ is a constant and $|\mathcal{S}| \ll n$, we can write:

$$space = O(n) \tag{14}$$

Hence, the overall time and space requirements are equivalent to that of the SLINK algorithm, and, therefore, equivalent to those which would be required for optimal clustering using the single link method on monolithic data. However, in order to transmit all of the data from the various remote sites would take $O(n^2)$ time and bandwidth, while our method uses only $O(n)$ time and bandwidth. Therefore, the second requirement specified for the algorithm has been met, that the transmission cost be less than or equal to that of transmitting the entire data set to a single site. Furthermore, the third requirement that the algorithms asymptotic time complexity should be less than or equal to the asymptotic time complexity of generating the model on a single site from monolithic data has also been met. As noted at the conclusion of the previous section, the first and forth requirements for the algorithm have also been met; therefore, the given algorithm meets all of the requirements specified.

The following section provides empirical results generated using the algorithm.

	B_1 Compared To:	
Number Divisions	B_2	B_3
2	92.4%	88.7%
3	82.2%	88.5%
5	83.0%	86.5%
10	79.8%	76.8%

Fig. 10. *Comparative results at different levels of division of the dendrogram between the monolithically generated dendrogram (B_1) and the global model generated from two sites (B_2) and the global model generated from three sites (B_3).*

6 Empirical Results

The data set used for testing the algorithm was the Boston Housing Data set, available from the UCI Machine Learning Repository, at

http://www.ics.uci.edu/AI/ML/MLDBRepository.html.

This data set contains 506 instances with 14 features. All data was normalized to the range $[0, 1]$, and the feature values were considered to be of equal significance. A total of three dendrograms were generated. The first of these, B_1, was generated using the canonical single link clustering algorithm for monolithic data. The second, B_2, was generated from two local models, the first of which was built from features $[1, 7]$, the second from features $[8, 14]$. The third global model, B_3, was generated from three local models, the first of which was built from features $[1, 5]$, the second from features $[6, 10]$, the third from features $[11, 14]$.

In order to determine the accuracy of the CHC algorithm relative to the monolithic single link clustering algorithm, the dendrograms were "split" at different points resulting in different number of clusters, in a manner similar to the example followed throughout the discussion of the algorithm, in which, the monolithically generated dendrogram and the dendrogram generated in the distributed manner were split such that there were 2 clusters in each. The dendrograms were split such that there were 2, 3, 5 and 10 clusters. Figure 10 shows the accuracy for B_2 and B_3 at these levels of division. In order to measure the accuracy, a "best-fit" method was used. This method consisted of comparing the clusters generated from the monolithic data to those generated using the CHC algorithm. The clusters were compared in decreasing order of size of the monolithically generated clusters. The cluster generated from the distributed local models that contained most of the points also contained in given monolithically generated cluster was considered to be the best-fitting cluster. Any points that were not in this point-wise intersection of the best-fitting clusters were considered to be incorrectly placed. Hence, the values displayed in Figure 10 represent the ratio of points that resulted from the described intersection operation to the total number of points in the dataset. Note that the differing results are a consequence of the sequence of agglomeration at the higher levels of the dendrograms.

The following section addresses future work to be performed.

7 Future Work

This paper presented an algorithm for distributed single link clustering of heterogenous data. The general method for bounding of the distances presented in this paper are specific to the Euclidean distance metric and single link clustering. Therefore, future work will include the following activities:

- Expansion of the method of bounding distances to handle other agglomerative, hierarchical clustering methods, such as average link and complete link clustering.
- Expansion of the general form of the algorithm to handle other distance metrics, such as the Manhattan distance metric.
- Expansion of the general form of the algorithm to cover clustering methods such as density based clustering in a similar manner, with the same requirements as specified in Section 3.
- Inclusion of the algorithm into the existing Collective Data Mining (CDM) [13] system BODHI, a Java based communication and interface package.
- Further generation of results on larger data sets.
- Examination of the statistic given in Equation 10 to determine if a better metric of dissimilarity is achievable. This would result in a higher accuracy of classification for the distributed models when compared to the monolithically generated model.

The following section concludes this paper.

8 Conclusion

This paper has presented a set of requirements for a distributed clustering algorithm which operates on heterogeneous data, and the Collective Hierarchical Clustering (CHC) algorithm which meets, and in some cases, exceeds those requirements. The time complexity of this algorithm has been shown to be $O(|\mathcal{S}|n^2)$, and the space requirements of this algorithm have been shown to be $O(n)$. This indicates that the CHC algorithm is capable of performing the analysis of the data in the same order of time and space as the equivalent centralized version of the algorithm.

The transmission cost for assembling the local models at a central facilitator site have been shown to be independent of the number of features at each site, with an overall communication requirement of $O(n)$ and a time complexity for transmission of $O(n)$. This is a significant improvement over centralizing the data to a single site, which has a communication cost of $O(nm)$, when there are n elements and m features in the data set. Furthermore, the CHC algorithm has both significantly lower communication costs and lower time complexity than transmitting the entire distance matrix to a centralized cite, which has an $O(n^2)$ communication cost and time complexity.

The empirical results and examples shown in this paper demonstrate that the CHC algorithm is indeed a feasible approach to distributed, heterogeneous clustering.

Finally, the bounding metrics presented in this paper are adaptable by their nature to other hierarchical clustering algorithms. This indicates that this algorithm is adaptable to other clustering methods. Efforts are being directed to this end.

Acknowledgments

The authors would like to acknowledge support from the National Science Foundation Grant IIS-9803360 and the American Cancer Society.

References

1. Dubes, R., Jain, A.: Clustering methodologies in exploratory data analysis. Advances In Computers **19** (1980) 113–228
2. Sibson, R.: Slink: An optimally efficient algorithm for the single-link cluster method. The Computer Journal **16** (1973) 30–34
3. Bradley, P., Fayyad, U., Reina, C.: Scaling clustering algorithms to large databases. In: Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining, AAAI Press (1998) 9–15
4. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: An efficient data clustering method for very large databases. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, ACM Press (1996) 103–114
5. Ng, R., Han, J.: Efficient and effective clustering methods for spatial data mining. In: Proceedings of 20th International Conference on Very Large Data Bases, Morgan Kaufmann (1994) 144–155
6. Guha, S., Rastogi, R., Shim, K.: Cure: An efficient clustering algorithm for large databases. In: Proceedings ACM SIGMOD International Conference on Management of Data, ACM Press (1998) 73–84
7. Olson, C.: Parallel algorithms for hierarchical clustering. Parallel Computing **8** (1995) 1313–1325
8. Dhillon, I., Modha, D.: A data clustering algorithm on distributed memory multiprocessors. In: Workshop on Large-Scale Parallel KDD Systems. (1999)
9. Kargupta, H., Hamzaoglu, I., Stafford, B., Hanagandi, V., Buescher, K.: PADMA: Parallel data mining agent for scalable text classification. In: Proceedings Conference on High Performance Computing '97, The Society for Computer Simulation International (1996) 290–295
10. Kargupta, H., Hamzaoglu, I., Stafford, B.: Scalable, distributed data mining using an agent based architecture. In Heckerman, D., Mannila, H., Pregibon, D., Uthurusamy, R., eds.: Proceedings of Knowledge Discovery And Data Mining, Menlo Park, CA, AAAI Press (1997) 211–214
11. Provost, F.J., Buchanan, B.: Inductive policy: The pragmatics of bias selection. Machine Learning **20** (1995) 35–61
12. Aronis, J.M., Kolluri, V., Provost, F.J., Buchanan, B.G.: The world: Knowledge discovery from multiple distributed data bases. Technical Report ISL-96-6, Intelligent Systems Laboratory, Department of Computer Science, University of Pittsburgh, Pittsburgh, PA (1996)

13. Kargupta, H., Park, B., Hershbereger, D., Johnson, E.: Collective data mining: A new perspective toward distributed data mining. Accepted in the Advances in Distributed Data Mining, Eds: Hillol Kargupta and Philip Chan, AAAI/MIT Press (1999)
14. Hershberger, D., Kargupta, H.: Distributed multivariate regression using wavelet-based collective data mining. Technical Report EECS-99-02, School of EECS, Washington State University (1999)
15. Murtagh, F.: Multidimensional Clustering Algorithms. Physica-Verlag (1985)