Parallel Clustering Algorithm for Large Data Sets with applications in Bioinformatics

Victor Olman, Fenglou Mao, Hongwei Wu, and Ying Xu

Abstract—Large sets of bioinformatical data provide a challenge in time consumption while solving the cluster identification problem, and that’s why a parallel algorithm is so needed for identifying dense clusters in a noisy background. Our algorithm works on a graph representation of the data set to be analyzed. It identifies clusters through the identification of densely intra-connected subgraphs. We have employed a minimum spanning tree (MST) representation of the graph and solve the cluster identification problem using this representation. The computational bottleneck of our algorithm is the construction of an MST of a graph, for which a parallel algorithm is employed. Our high-level strategy for the parallel MST construction algorithm is to first partition the graph, then construct MSTs for the partitioned subgraphs and auxiliary bipartite graphs based on the subgraphs and finally merge these MSTs to derive a MST of the original graph. The computational results indicate that when running on 150 CPUs, our algorithm can solve a cluster identification problem on a data set with 1,000,000 data points almost 100 times faster than on single CPU, indicating that this program is capable of handling very large data clustering problems in an efficient manner. We have implemented the clustering algorithm as a software CLUMP.

Index Terms—Pattern recognition, clustering algorithm, genome application, parallel processing.

V. Olman, F. Mao, H. Wu and Y. Xu are with the Computational System Biology Laboratory, Department of Biochemistry and Molecular Biology and Institute of Bioinformatics, University of Georgia, Athens, Georgia, 30602. E-mail: {olman, fenglou, hongweiw, xyn} @csbl.bmb.uga.edu

© 2007 IEEE
1 INTRODUCTION

Data (object) clustering represents one of the most often encountered problems in data analyses. The basic problem is to partition a data set into "clusters" of data points that are "close" to each other but relatively "far from" other data points. A more general problem is the so called cluster identification problem [17], which is to identify "dense" clusters in a possibly noisy background. For such a problem, identified clusters don't necessarily cover the whole data set. There are many applications of the clustering problems in different fields, ranging from image analyses to pattern recognition, social science, biology, telecommunication and many other fields. Many methods with different application objectives have been developed to solve the clustering problems, including K-means [5], Single Linkage Algorithm (SLA) [3] and other hierarchical clustering methods [21], self-organizing map [5], Markov Cluster Algorithm [4], unsupervised clustering algorithm for graphs based on flows in graphs [10]. While each of these classes of methods has its advantages and has been shown to be useful for many application problems, there remains a few challenging problems facing most of the existing clustering algorithms. The most important one is the limit on the size of the data sets they can effectively handle. Often these algorithms are designed to deal with relatively small data sets, possibly ranging from hundreds to tens of thousands of data points. When applied on problems with sizes ranging from hundreds of thousands to millions, these algorithms are in general too slow or inefficient.
to be practically useful. Robustness of these algorithms represents another challenging issue. For example, the popular K-means algorithm works well only if to be identified clusters are embedded in disjoint convex sets, and the clusters are comparable in size, while self-organizing map-based methods work successfully only if a dataset intrinsically has the structure with ability for dimension reduction. For example, SOM would fail for the set of disjoint 10-dimensional spheres uniformly scattered in the 10-dimensional Euclidean space. The only algorithm that can be directly compared to ours is Single Linkage Algorithm (SLA) just because both use MST construction as a preliminary step, and therefore have close complexity. But in our approach the algorithm is a result of searching for entities defined as clusters and we offer the estimation of the statistical significance of a cluster, while SLA considers any subtree as a cluster without evaluation of a cluster quality. In this particular paper we don’t compare performance of our method with others because the evaluation of a performance is a very sensitive issue and depends on a goal of a research that should be established a priori. For example, if a data set consists of two disjoint spheres of very different radiiuses, the K-means will cut the big sphere while our algorithm will separate two spheres. But both results could be meaningful from different points of view.

Modern biology is a data rich field. Driven by the advent of high-throughput technologies such as sequencing technology, microarray gene chips, two hybrid arrays[15], and mass spectrometry [7], enormous amount of
data has been generated that reflect different aspects of living organisms at molecular, cellular, and even eco-systems levels. In addition, even larger amount of data have been derived from the experimental data by using the bioinformatics tools, such as predicted genes, promoters, gene families, protein structures, molecular interactions, biological networks, plus many intermediate results. Generation and application of these data often require a capability for data clustering in some fashion. In our own research, we often encounter problems of needing to cluster data sets up to hundreds of thousands to million data points, including our recent work on functional classification of genes [27], functional module identification [28] and microarray data analysis [30]. We have recently developed an effective parallel algorithm for solving the cluster identification problem, particularly for large data sets.

The basic idea of this clustering algorithm is that it first represents a target data set as a weighted undirected graph, with each data point represented as a vertex and each pair of data points is connected by an edge with its weight being the “distance” between the two data points. Then it builds a minimum spanning tree (MST) of the graph. A key property of a MST is that it generally preserves the structures of clusters in the sense that each cluster is generally presented as a subtree of a MST as we have previously established [17], and hence a clustering problem (or generally cluster identification problem) can be solved on the MST representation of a data set. Such a cluster-preserving
representation facilitates efficient algorithms for identifying clusters in a data set. We have previously developed a rigorous and linear algorithm for identifying clusters based on the MST representation [17] of the data set. Among numerous nice properties of the MST-based clustering algorithm, one is that the algorithm is highly robust in terms of the specific distance function for solving the data clustering as long as the relative order among the edge distances is preserved, a highly useful property for dealing with noisy data sets. The computational bottleneck of the above outlined MST-based clustering algorithm is in the step of construction of an MST representation of a data set. Since our clustering algorithm is based on a sequential representation of MST, we require the use of Prim’s algorithm [22]. As we know, Prim’s algorithm for a graph with E edges and V vertices requires \( O(|E| + |V| \log(|V|)) \) steps to compute a MST [22]. In general, E is \( O(V^2) \), which is often too large for practical applications with millions of data points. Despite of the sequential essence of MST construction, a number of algorithms have been developed to parallelize calculations. Detailed analysis of this effort is done in the review [16]. All suggested parallel algorithms present parallelization of calculations with heavy Message Passing Interface (MPI). In [20], authors implemented parallel version of the SLINK algorithm [23] using SIMD array processors. The parallel version of hierarchical clustering is presented in [14] on n-node hypercube
and n-node butterfly. Another implementation of a parallel approach is reported in [9] with calculations distributed between processors and with use of MPI. Using different types of Parallel Random Access Memory (PRAM) model, people have found [8] a good solution for minimizing the time of communication using external memory when constructing a MST, while in [2] authors have dealt with the construction of spanning tree using Symmetric Multiprocessors though there is no guarantee that it will find a minimum spanning tree. The issue with these efforts is that they don’t provide the practically useful and rigorous solution to the problem from implementation point of view. In widely cited papers [3] and [19] the time complexity is O(Vlog(V)) with V/log(V) processors, and in [13] it’s O(log^{3/2}V) with (V+E) processors, and while it presents an undisputed theoretical result, practically it’s not applicable with large datasets and today’s technology. None of these papers provide Internet accessible computer servers allowing a user to carry out MST construction with large dataset. Theoretically more efficient algorithm represents a non-trivial challenge for implementation. In addition, there is no guarantee that these theoretically more efficient algorithms will solve the MST construction faster in practice. Our approach is easy for implementation, and while theoretical papers are dealing with a complexity O(f(n)), where n is a dataset size, our goal was to minimize coefficient C in O(f(n))~Cf(n) for large n with known function f(n). Hence we have imple-
mented our own parallel algorithm for the MST construction, taking advantage of the Linux clusters we have in our lab, which supports the data clustering computation server, based on the algorithms presented in this paper. The basic difference of our approach from cited is that we don’t parallelize the known algorithm but rather use a procedure that merges results precalculated in a parallel way and ends up with the MST for the original graph.

The key effort in designing the parallel algorithm is to minimize the communication efforts among processors. We have used MPI as the way of communication across processors on a Linux cluster, popularly used for bioinformatics applications. Compared to previously developed parallel algorithms for MST construction [13], which are generally based on Boruvka’s algorithm [6], our new parallel algorithm is based on a parallel MST constructions for subsets with moderate sizes. Using the parallel algorithm for the construction of an MST from a given dataset, we have developed an efficient algorithm for solving the cluster identification problem on large data sets. For each identified cluster, we assign a $p$-value to measure statistical significance of the cluster in terms of its data compactness versus the distance to its neighbors. The software package for this data clustering algorithm is called CLUMP for “clustering through minimum spanning tree in parallel”.
Before starting a description of the parallel algorithm we shortly describe the idea of cluster identification using MST, presented in our previous paper [17].

Let $S$ be the set of elements $s \in S$ and $W(s_1, s_2)$ be a distance between any two elements of $s$. We expand our definition of $W$ to the distance $W(S_1, S_2)$ between sets $S_1, S_2$ as the shortest distance between elements of sets

$$W(S_1, S_2) = \min \{W(s_1, s_2) | s_1 \in S_1, s_2 \in S_2\}$$

According to definitions in [12] and [17], the Necessary Condition for a subset $C \subset S$ to be a cluster is that for any partition $C = C_1 \cup C_2$, $C_1 \neq \emptyset, C_2 \neq \emptyset, C_1 \cap C_2 = \emptyset$,

$$W(C_1, S - C_1) = W(C_1, C_2) \quad \text{(NC)}$$

In other words regardless of partition of a cluster, the two parts of a cluster will still be closer to each other than to any other element of $s$. A key idea of our MST-based clustering algorithm is to represent the given dataset using a linear representation $LR$ through construction of a MST as follows. $LR$ is a list of the elements of $s$ whose sequential order is the same as the order that these elements got selected by Prim’s algorithm into the MST during its construction. In addition, each element $s$ has a numerical value associated with it, which is the $w(.,s)$ value of the edge that Prim’s algorithm used to add $s$ into the MST. A highly useful property of this linear representation is that data clusters in the given dataset, as defined [30], have a one-to-one correspondence with the “valleys” in this Linear Representation if we view it in a 2D coordinate system with the sequential order as the x-axis and the values of
individual elements as the y-axis [30]. Hence data clusters in the given dataset can be identified through identifying valleys in this linear representation. In comparison to Single Linkage Algorithm, our approach searches for clusters satisfying the special condition of a cluster (NC) that essentially narrows the set of all subtrees of MST as it’s done in SLA.

2. Parallel Algorithm of MST Construction

We first present our parallel algorithm for MST construction of a graph representation $G = (E, V)$ of a given data set $S$, which has the following key steps:

- Partitioning $G$ into $s$ subgraphs, $\{G_j = (V_j, E_j)\}, j=1,\ldots,s$, where the value of $s$ is determined later in this Section, $V_j$ is the set of vertices in $G_j$, and $E_j \subset E$ is the set of edges connecting vertices of $V_j$;

- Defining bipartite graphs $B_{ij} = (V_i \cup V_j, E_{ij})$, where $V_i$ and $V_j$ are vertex sets of $G_i$ and $G_j$, and $E_{ij} \subset E$ is a set of edges between vertices of $V_i$ and vertices of $V_j$, $i \neq j$, for each such pair of subgraphs from $\{G_i\}$;

- Constructing a MST $T_{ii}$ on each $G_i$ and $T_{ij}$ on each $B_{ij}$ in parallel;

- Building a new graph $G^0 = \bigcup_{1 \leq i \leq j \leq s} T_{ij}$, by merging all the MSTs from the previous step. A result of the merging operation is a subgraph $G^0$ of $G$ with a vertex set $V$ and edges from trees $T_{ij}, 1 \leq i \leq j \leq s$;

- Constructing a MST of $G^0$.

A mathematical proof is given in the Appendix A to show that $\text{MST}(G^0)$ is a
MST\((G)\), MST of the original graph \(G\). The key idea employed here is that we calculate in parallel MST for each subgraph and each auxiliary bipartite graph formed by each pair of subgraphs. Then we build a highly sparse graph \(G^0\) by merging the constructed MSTs, and build a MST of \(G^0\).

We now provide some analysis on the computational running time of the algorithm. For our current implementation of Prim’s algorithm for building MSTs on the subgraphs, we have used Fibonacci heap \([22]\) for each subgraph \(G_i\) and each bipartite graph \(B_j\) to facilitate efficient implementation of the “finding the next smallest edge” operation in Prim’s algorithm, which gives an \(O(|E_i| + |V_i| \log(|V_i|))\) time for each subgraph \(G_i\), and 
\[O(|V_i| \cdot |V_j| + |V_i| + |V_j|) \log(|V_i| + |V_j|))\] for each bipartite graph.

We have assessed the actual computing time of the parallel algorithm, using the following data sets. The data sets we have used consist of 10,000 up to 500,000 (with a step 10,000) vectors ranging from 10, 20, 30, ..., to 100 dimensions (D). Each component of a data point is an independent uniformly distributed real value taken from the interval [0,1]. For our test, we have tried three distance measures, namely, Euclidean, Manhattan and 1-Pearson correlation. For each distance type, we have used a linear regression model to summarize the dependence of the MST construction time, \(T_{(D,V)}\), on \(V\) and \(D\).

These results, as summarized in Table 1, are used in (1) and (2) for the analysis of computing time of MST construction of original graph.
Table 1. Linear regression model for the MST construction time for a complete graph, $T_C(D,V)$, and for a bipartite graph, $T_B(D,V)$, as functions of the number of objects ($V$), and the dimensionality of the object ($D$), where $v=V*0.0001$, and R is the multiple regression correlation coefficient.

<table>
<thead>
<tr>
<th>Dist</th>
<th>$T_C(D,V)$</th>
<th>$T_B(D,V)$</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>EUC</td>
<td>$(.192D+1.61);v^2$</td>
<td>$(.385D+3.80);v^2$</td>
<td>.999</td>
</tr>
<tr>
<td>MAN</td>
<td>$(.206D+1.58);v^2$</td>
<td>$(.413D+3.62);v^2$</td>
<td>.999</td>
</tr>
<tr>
<td>PEA</td>
<td>$(.173D+2.03);v^2$</td>
<td>$(.346D+4.35);v^2$</td>
<td>.999</td>
</tr>
</tbody>
</table>

The time for constructing the final MST $(G^0)$ after merging the MSTs ($G^0$ is the original graph $G$ without edges that don’t belong to $T_{ij}, 1 \leq i \leq j \leq s$), $T_M(s,V)$, accurately fits the following linear regression model with $R=.9931$:

$$
T_M(s,V) = s^*v^*(0.401967*\log(v)-0.464002) + 2.826016^*v^* .
$$

(1)

where $s$ is the number of subsets of vertices, $v=0.0001*V$, $V$ is the number of vertices in the original graph, and $s^*(s+1)/2$ is the number of CPUs used at the previous step. As it is shown in Appendix, the best partition is the partition of $V$ into subsets such that $|V_i|=|V_j|$ if $s>2$, and $|V_1|=2|V_2|$ for $s=2$. Since the MST-construction for a complete graph with $V$ vertices and $.5V(V-1)$ edges is much faster than for a bipartite graph with $2V$ vertices and $V^2$ edges, the total computing time is

$$
T(D,V,s) = T_B(D,V/s) + T_M(s,V)
$$

(2)

Our test results showing the accuracy of formulae (2) are presented in Fig. 1.
for MST construction for a set of 1,000,000 40-dimensional vectors with the number of partitions ranging from 2 to 30, while circles are experimental results. The optimal number of partitions for this case is 14.

One can see that the total time is a mono-extreme function that has a simple explanation: the larger the number of partitions (reducing the time for simultaneous MST construction on all partitioned subgraphs) the larger number of edges in graph $G''$ (increasing the time for MST construction at the last step).

Given a complete graph with $V$ $D$-dimensional points, (2) gives a tight estimate on the running time of our parallel algorithm, as a function of $s$. The optimal running time can be obtained by minimizing (2) with respect to $s$. We get the optimal value of $s_0$ by differentiating (2) and equaling the derivative to 0 as

$$s_0(D,V) = \sqrt{2V(c_0 + c_1D)/(0.401967*log(V)-0.464002)},$$

where $c_0$ and $c_1$ are the regression coefficients for MST-construction on the bipartite graph (Table 1), and they depend on the time for distance calculation.

We calculate the theoretical factor in speeding up MST-construction by using
the parallel implementation versus a single-processor, $SU_T(D,V)$, as a ratio of the time consumption by:

$$SU_T(D,V) = \frac{T(D,V,1)}{T(D,V,s_o(D,V))} \quad (3)$$

The experimental speed-up factor, $SU_E(D,V)$, is defined as the ratio between the real computing time of the parallel and single-processor implementations. Fig.2 shows the accuracy of theoretical estimation of speed-up.

**Fig.2.** The graph is an analytical form of speeding-up (3), while circles are experimental points.

We have summarized $SU_T(D,V)$, $SU_E(D,V)$ and the corresponding $s_o$ for different distance types in Table 2.

**Table 2.** Theoretical ($SU_T$) and experimental ($SU_E$) speed-up factors of the parallel MST construction compared to single-processor implementation, with the optimal number of partitions $s_o$, where $V$ represents the number of vertices, and $D$ represents the dimensionality of the vertices.
### 3. Cluster identification

After building an MST for a given data set, our algorithm will construct a linear representation (LR) of the MST \[30\]. Given \(n\) data points, let \(LR[i]\) be the distance between the data point that gets added into the MST at the \(i\)-th step of Prim’s algorithm through an edge connecting this data point with the current MST. In this way, each data point is associated with a two-dimensional representation, \((i, LR[i]), i=1,\ldots, |V|\). A valley in the LR is defined as a list of indices \(s,s+1,\ldots,s+t\), \((1 \leq t \leq |V|, -s, s \geq 0)\) such that

\[
\min(LR[s], LR[s + t + 1]) > \max\{LR[i] | s < i < t + 1\}
\]

In other words the set of vertices form a valley if they are consecutive in steps of Prim’s algorithm and edges that open and close the valley, are longer than any edge inside the valley. We have previously shown \[30\] that there is a one-to-one correspondence between valleys and data clusters that satisfy our cluster definition (NC), and the hierarchical structure among clusters is well preserved among the corresponding valleys. Based on this result, each cluster can be identified through identification of valleys in the Linear Representation of a graph. We have employed the same simple algorithm that we have
used for identification of conserved binding motifs in genomic sequences for identification of valleys [17]: recursive calculation of maximum of LR in the valley belonging to $[0, |V|]$. The proof of correctness of this search is based on the fact that starting and ending steps of a valley have largest edges. It should be noted that a linear representation of a graph is not unique but the aforementioned property holds for any linear representation of a graph. Fig. 3 shows a simple example for a set of two-dimensional data points in Euclidean 2D space.

**Fig. 3.** Two-dimensional points with clustering structure.

On Fig. 4 we can clearly see an apparent correspondence between clusters of data points and valleys in the Linear Representation of the data set with a simple hierarchical structure.

**Fig. 4.** The x-axis is the linear representation of the MST (indices of steps) and the y-axis represents the length of the edge that is used to include the corresponding vertex into MST. The lower V-shape dash lines represent four separate clusters while the upper two V-shapes cover the aggregated clusters.
For each identified valley in the linear representation, we can assess the statistical significance of the valley using the following model. We assume that if there is no cluster (or a valley) in the data set (data points are scattered uniformly), the LR values for the set of indices $s, s+1, \ldots, s+t$ ($t>1$) are actually distances between sorted observations from the uniform distribution. In other words, the values $z_m$ ($m=1, \ldots, t+1$) are sorted uniformly distributed observations on $(0,1)$, where $z_m = \frac{\sum_{j=s}^{m} (LR[j]-L)}{T}, m=1, \ldots, t+1, z_0 = 0$, $T = \frac{\sum_{j=s}^{s+t+1} (LR[j]-L)}{s+t+1}$ is a normalized constant, and $L = \min(LR[j] | s < j \leq s + t)$. The ratio $\rho = \max(z_i - z_{i-1} | 1 \leq i \leq t) / \min(z_1, 1 - z_t)$ and the number of data points in a cluster, $t+1$, are the only parameters defining the statistical significance ($P$-value) of a cluster. We calculate P-value as the $Pr(X \leq \rho)$, where $X = \max(x_i - x_{i-1} | 1 \leq i \leq t) / \min(1-x_t, x_1)$, and $0 \leq x_1 \leq x_2 \leq \ldots \leq x_t \leq 1$ are sorted independent uniform observations in interval $(0,1)$. The calculations are based on the fact that a random $t$-dimensional vector with components $x_i - x_{i-1}, i=1, \ldots, t$, follows Dirichlet distribution [26].

4. CLUMP implementation

The above algorithm has been implemented as a software package, CLUMP, using MPI and ANSI C. The executable code for 32 bits Intel x 86 compatible
Linux clusters is available at http://csbl.bmb.uga.edu/CLUMP/download.html. The core part is the set of functions that provide MST construction for a given graph. A web interface for CLUMP is developed with pop-up menus. Currently the clustering software supports three distances, namely Euclidean, Manhattan and 1- Pearson correlation (note that the distance does not have to satisfy triangle inequality, and hence the data set does not have to be defined in a metric space). In addition, a user can define arbitrary distance measure or provide pre-calculated distances in an input file.

There are other parameters that have default values or can be chosen by the user of CLUMP. The user can restrict the search for clusters based on their minimum and maximum size (number of data points in a cluster), as well as the value for $\rho$ (or P-value) can be bounded from above. Another parameter $\alpha < 1$ is used to prevent from reporting very similar clusters, so that if for two clusters $A \subset B$ there is no such a cluster $C, A \subset C \subset B$, then both clusters $A$ and $B$ will be reported iff $|A| \leq \alpha |B|$. The number of partitions for a given data set can also be defined by the user ($\leq 25$), otherwise the optimal number is used (see Section 2). The output of CLUMP is a text file with clusters of data points and the statistical significance for each cluster.

An online CLUMP server is available at
http://decoder.cc.uga.edu/paralell_MST/home.php, which provides large scale clustering ability for registered users. The online CLUMP is implemented by using MySQL database, php server-side script language, postfix mail system, and apache web server. The backend computer cluster is managed by Sun Grid Engine. In order to use the online CLUMP, a user should send an application to register for using CLUMP. After the application is approved, the user can submit jobs to the online CLUMP; the user can specify the parameters which control the clustering or using the default values. The user can download both the result MST and the identified clusters. The user can also run the hierarchical clustering program, which is in the standalone CLUMP package and can be used to build clusters from MST, by using different parameters to create most meaningful clusters.

5. Biological applications

We provide two applications of CLUMP in this section to demonstrate the power of this new clustering algorithm. All used processors are Intel x86.

**Application 1:** We have applied CLUMP for the hierarchical classification of functionally equivalent genes for prokaryotes at multi-resolution levels [29]. The framework can be described as follows.
We first define the functional equivalence relationship between a pair of genes, \( g_1 \) and \( g_2 \), of two different genomes, \( f(g_1, g_2) \), by incorporating both their sequence similarity and genomic context information as:

\[
f(g_1, g_2) = h(g_1, g_2) \left[ 1 + \lambda \sum_{i,j} P(g_1, g_i)P(g_2, g_j)I(h(g_1, g_j) \geq t_h) \right]
\]

where \( h( , ) \) denotes the sequence similarity measure, \( \lambda \) determines how much the genomic context information of the pair \( (g_1, g_2) \) should be considered over the pair's sequence similarity information, the summation \( \Sigma_{ij} \) is over all gene pairs \( (g_i, g_j) \) with \( g_i \) and \( g_j \) being likely to belong to the same genomic neighborhood of \( g_1 \) and \( g_2 \), respectively, \( P( , ) \) calculates the likelihood that two genes of the same genome are in the same genomic neighborhood, and the indicator function \( I( ) \) and the threshold \( t_h \) for the sequence similarity measure are introduced to make sure that only those sufficiently reliable gene pairs \( (g_i, g_j) \) in the genomic neighborhoods of \( (g_1, g_2) \) are considered as supporting evidence to the equivalence relationship between \( g_1 \) and \( g_2 \). The detailed definitions of these functions can be found in [29]. We have applied this scoring scheme to 224 prokaryotic genomes (ftp://ftp.ncbi.nih.gov/genomes/Bacteria/, March 2005) [29]. If we consider only those gene pairs whose BLASTP [1] e-values \( \leq 1.0 \), we have obtained \( \sim 46 \) million gene pairs involving 609,887 genes, which cover \( \sim 92.7\% \) of all the genes of the 224 genomes. We have constructed
a weighted graph $G(V, E)$, with $V$ and $E$ being the sets of the nodes and edges, respectively, to represent these 609,887 genes and their functional equivalence relationships; the weight on each edge is set to be proportional to the functional equivalence measure $f(, )$ between the two genes being connected [29].

As shown in Figure 4, $G(V, E)$ contains densely intra-connected sub-graphs, each of which corresponds to a cluster of genes that are functionally equivalent to each other at certain resolution level; and, sub-graphs (gene clusters) form a hierarchical structure. To identify these dense clusters, we have applied the MST-based hierarchical clustering algorithm on $G(V, E)$.

Note that the level of functional equivalence between a pair of genes $(g_1, g_2)$ is not only reflected by their own measure $f(g_1, g_2)$ but also through those genes $g'_k (k = 1, 2, \ldots)$ that are simultaneously equivalent to both $g_1$ and $g_2$. So we have defined the distance function for the MST-based clustering algorithm as

$$d(g_1, g_2) = \sqrt{f^2(g_1, g_2) + \rho \sum_{k=1}^{r} f(g_1, g'_k)f(g_2, g'_k)}^{-1},$$

where $r$ is the maximum number of genes allowed to be considered that are simultaneously equivalent to both $g_1$ and $g_2$, the parameter $\rho$ determines the level of supporting evidence provided by such genes, and $g'_k$ is the $k$-th ranked in terms of the value $f(g_1, g'_k)f(g_2, g'_k)$ among such genes. The two parameters $r$ and $\rho$ provide flexibilities for a user to tailor the above distance function to his/her specific prob-
hierarchy clustering be are levels. In our case we used r=10 and $\rho = .6$

When applied to $G(V, E)$ for clustering, CLUMP identified 51,205 gene clusters, which are organized into 5,339 multi-level and 15,770 single-level non-overlapping trees, where the multi-level trees totally contain 35,435 clusters covering 534,818 genes, and the single-level trees totally contain 15,770 clusters covering 75,067 genes. The statistical significance of each identified cluster has been assessed through the P-value computed for the alternative hypothesis that these genes do not form a cluster. [29].

The clustering results have been validated through comparisons with two existing classification systems, Clusters of Orthologous Groups (COG) [25] and Pfam [8]. The comparisons have indicated, on the one hand, that our clustering results are generally consistent with these two well-known classification systems, as reflected by the fact that ~85% of those non-trivial COG clusters and ~73% of those non-trivial Pfam clusters are essentially included in our clustering results; and, on the other hand, at a different level of our cluster hierarchy the functional annotations of genes belonging to the same cluster are consistent to different degrees, suggesting that our clustering results can be used for functional annotation of unknown genes at different specificity levels. More discussions on this application can be found in [29].
**Fig. 5.** The zoom-in view of one of the sub-graphs of $G(V, E)$, where each node represents a gene, each edge indicates that the BLASTP e-value between the two connected genes is < 1.0, and the weight on the edge is proportional to the similarity measure $f(\cdot, \cdot)$ between the two genes. Each ellipse compasses a sub-set of genes that are functionally equivalent at a certain level.

The clustering results given by CLUMP are identical to those given in [29], but the computing time by CLUMP using 55 CPUs (partition in 10 sub-graphs) is 42 times faster (8 min) than in the original analysis with 1 CPU (>5 hours).

Application 2: We have also applied CLUMP for the analyses of the Diversa Silage Soil metagenome (http://img.jgi.doe.gov/cgi-bin/m/main.cgi?page=taxonDetail&taxon_oid=2001200001). Our goal is to do functional classification of the genes encoded in this metagenome. This metagenome is obtained directly from the farm silage surface soil sample rather than from laboratory clonal cultures; and is predicted to contain 184,374 protein coding genes. For our preliminary study, we have used the BLASTP e-
value between a pair of genes as an assessment of their distance; and have then applied CLUMP on the complete graph consisting of 184,374 vertices, for gene clustering. When concentrating on constructing the MST, the time on 45 CPUs, 4 min, was 36 times faster than on the single CPU, 144 min.

We have identified 1,100 statistically significant (with P-value < .001) non-overlapping clusters (called CLUMP clusters in the rest of the discussion), covering 49,505 (~27%) genes. When comparing these CLUMP clusters with the COG [25] (3,827 clusters covering 85,013 genes) and Pfam [8] (1087 clusters covering 76,020 genes) clustering results, we have found that (i) 307, out of all the 973 CLUMP clusters that have overlaps with the COG clustering results, each has its Jaccard similarity coefficient with a COG cluster ≥ 2/3 and can therefore be considered to correspond to a COG cluster; and (ii) 219, out of all the 862 CLUMP clusters that have overlaps with the Pfam clustering results, each has its Jaccard similarity coefficient with a Pfam cluster ≥ 2/3 and can therefore be considered to correspond to a Pfam cluster. Our study on the Diversa Silage Soil metagenome data has not been complete yet. Nevertheless, these primary results have indicated that the MST-based clustering approach, the basis of CLUMP, can essentially capture the commonalities within a gene group and the differences across different gene groups.

As for metagenomic analyses, the most frequently asked questions often
include what organisms are present and what roles they play in the local ecosystem. Gene clustering can be used as one technique to answer these questions. For example, out of the CLUMP clusters that have overlaps with the COG and Pfam clustering results, 34 of them correspond to ribosomal proteins; and the number of genes in each such cluster ranges from 8 to 21. When combined with the clusters of other house-keeping genes, these ribosomal protein gene clusters can be used to analyze the taxonomic diversity of the Diversa Silage Soil metagenome. Also, by studying the functional diversity of these identified gene clusters, we may infer this metagenome’s metabolic capabilities and their effects on the environment. So for our ongoing project on the analyses of metagenomic data, we plan to use gene clustering as one of our main techniques. We have done similar applications on other biological data, including microarray gene expression data [18]. Overall we found the performance of parallel clustering algorithm highly effective and practically useful.

6. Conclusion

The software CLUMP has proved to be a highly useful tool for clustering for large quantities of biological data. Though the bottleneck in executing this
program, i.e., construction of MSTs, is a time-consuming step, our decomposition strategy and associated parallel algorithm have made the application of the algorithm practically useful. For a typical data clustering problem with 1,000,000 data points in 30 dimensions, CLUMP can finish the calculation in 40 minutes on 105 Intel x86 processors. The CLUMP is open source, and we will continue developing the software by adding new distance functions and other features.

Appendix A

Let \( G = (V, E) \) be a weighted undirected graph with vertex set \( V, \ |V| = n \geq 2 \), and edge set \( E \) with \( w(e) \) be the weight of \( e \in E \); and \( \text{MST}(G) \) be a MST of \( G \), \( \text{MST}(G) = \bigcup_{i=1}^{i=n-1} e_i \), consisting of edges \( e_1, e_2, \ldots, e_{n-1} \in E \). We will show that MST-construction can be parallelized as follows.

**Lemma 1:** Let \( U_i \) and \( \overline{U_i} \) be subsets of vertices \( v = U_i \cup \overline{U_i} \) in two subtrees formed by cutting edge from MST, then \( \min\{w(v,u) | v \in U_i, u \in \overline{U_i}\} = w(e_i) \)

**Proof** is straightforward by the definition of MST.

**Lemma 2:** For any nonempty partition \( v = U \cup \overline{U} \), there is an edge \( e_0 = (v_0, u_0), v_0 \in U, u_0 \in \overline{U}, e_0 \in \text{MST}(G) \), such that
\[ \min \{w(v,u) \mid v \in U, u \in \overline{U}\} = w(e_0). \]

**Proof:** Let’s assume it’s not true, i.e. there are vertices \( v_e \in U \text{ and } u_e \in \overline{U} \) such that,

\[ w(v_e, u_e) < w(e_0), \]

where \( e_0 = \arg \min \{w(e) \mid e \in MST(G) \text{ and } e \text{ connects vertices of } U \text{ and } \overline{U}\}. \]

By adding edge \( e_c = \{v_e, u_e\} \) to \( MST(G) \) we get a cycle that contains at least one edge \( e^0 \in MST(G) \) connecting \( U \) and \( \overline{U} \). Now we have \( w(e_c) < w(e_0) \). Replacing \( e^0 \) by \( e_c \), we get a spanning tree with lower weight, and that contradicts to the fact that \( MST(G) \) is a minimum spanning tree.

**Lemma 3:** Let \( G_0 = (V_0, E_0) \) be any (connected) subgraph of the original graph \( G \). For any edge \( e \in MST(G) \), if \( e \in E_0 \), then \( e \) belongs to \( MST(G_0) \).

**Proof:** By cutting edge \( e \in MST(G) \) we get two trees with vertex sets \( U_e \) and \( \overline{U_e} \),

\[ V = U_e \cup \overline{U_e}, \]

and a corresponding nonempty partition of \( V_0 = V_e \cup \overline{V_e} \), where \( V_e = U_e \cap V_0 \) and \( \overline{V_e} = \overline{U_e} \cap V_0 \). The partition \( V_0 \) is not empty because the edge \( e \) connects vertices in \( V_0 \). From Lemma 1, \( w(e) = \min \{w(v,u) \mid v \in U_e, u \in \overline{U_e}\} \), and because of \( V_e \subseteq U_e, \overline{V_e} \subseteq \overline{U_e} \), we get \( w(e) = \min \{w(v,u) \mid v \in V_e, u \in \overline{V_e}\} \) since the edge \( e \) satisfies the right-hand side condition of the above equation. Applying Lemma 2 to the partition, we get the required result that \( e \in MST(G_0) \).

Hence the lemma follows.

**A1: Method for MST construction**
Let’s consider an $s$-way partition $V = \bigcup_{i=1}^{s} V_i$ and the corresponding subgraphs $G_i = (V_i, E_i)$ of $G$, where $E_i$ is a subset of $E$ consisting of edges connecting vertices of $V_i$, $i=1,\ldots,s$. We have the following result.

**Theorem:** Let $B_{ij}$ be a bipartite graph, $B_{ij} = (V_{ij}, E_{ij})$, where $V_{ij} = V_i \cup V_j$ and $E_{ij}$ is the set of edges connecting vertices between $V_i$ and $V_j$, $1 \leq i < j \leq s$, and $G_M$ be the graph formed by merging minimum spanning trees $MST(G_i), MST(Q_{ij}), 1 \leq i, j \leq s$. Then $MST(G) = MST(G_M)$.

**Proof:** It’s obvious that if a graph $G_0$ is a subgraph of $G$ and contains all edges of $MST(G)$, then $MST(G_0) = MST(G)$. From here the proof is straightforward based on Lemma 3.

**A2. Complexity of the method**

By using Fibonacci heap in the implementation of Prim’s algorithm, the running time $RT$ for MST construction is $O(|E| + |V| \log(|V|))$ [22], or more specifically, $c_0 |E| + c_1 |V| \log(|V|)$, where coefficients $c_0, c_1$ depend on the edge-density of a graph and the time for edge weight calculation. In our case we have three types of graphs: complete graph, bipartite graph and a sparse graph (merger of MSTs).

The running time ($RT$) for the preprocessing step of our parallel algorithm (see Section 2), using $s^*(s+1)/2$ processors, is
max{ max \(RT(G_i)\), \(\max_{1 \leq i < j \leq s} RT(B_{ij})\)} . Our goal is to get a partition of the set \(V\) into \(s\) subsets that would minimize max\{ max \(RT(G_i)\), \(\max_{1 \leq i < j \leq s} RT(B_{ij})\)\}. We introduce new variables \(a_1, a_2, \ldots, a_s, \sum_{i=1}^s a_i = 1, a_i > 0\), such that \(|V| \sim a_i |V|\), i.e. \(a_1, a_2, \ldots, a_s\) represent a partition of vertices. So we have \(|E_i| \sim |V|^2 a_i^2 / 2\), \(|E_{ij}| \sim |V|^2 a_i a_j\). By ignoring the much smaller second term in \(RT\), we obtain a minimization of

\[
F(a_1, \ldots, a_s) = \max_{1 \leq i < s} \left( a_i^2 / 2 \right), \max_{1 \leq i < j \leq s} (a_i a_j) \text{ on the simplex } \sum_{i=1}^s a_i = 1.
\]

Lemma 4. For \(s=2\), \(\min\{F(a_1, a_2) | a_1 + a_2 = 1\} = 2/9\) and is achieved with \(a_1 = 1/3\) and \(a_2 = 2/3\), so a partition into two subgraphs is \(RT\)-optimal with \(a_2 = 2a_1\). For \(s > 2\), the optimal partition is achieved with \(a_i = 1/s, i = 1, 2, \ldots, s\) and

\[
\min\{F(a_1, \ldots, a_s) | \sum_{i=1}^s a_i = 1\} = |V|^2 / s^2.
\]

Proof. For \(s=2\), \(F(a_1, a_2) = 5: \max[a_i, (1-a_i)^2, 2a \cdot (1-a)]\), and a simple analysis proves the correctness of the statement. For \(s > 2\), since \(F(a_1, \ldots, a_s) \geq F_1(a_1, \ldots, a_s) = \max_{1 \leq i < j \leq s} (a_i \cdot a_j)\), it is sufficient to prove the Lemma for \(F_1(a_1, \ldots, a_s)\). Let’s assume that the statement is not a true, and without loss of generality, we have \(a_1 = a_2 = \ldots = a_k > a_{k+1} = \ldots = a_s, 1 \leq k < s\). If \(a_s = a_{s-1} = \ldots = a_{s-t} = 0\), \(0 \leq t < s-k\), we consider a new solution \(a_i' = a_i - \epsilon, \epsilon > 0\), \(a_i' = a_i + \delta, \Delta > 0\), \(a_i' = a_i + 1\), \(i = s-t, \ldots, s\), where \(\epsilon \cdot s = \delta \cdot t\). By choosing a sufficiently small \(\epsilon\), we can reduce the value of \(F(a_1, \ldots, a_s)\), and it proves that for the optimal solution \(a_i > 0, 1 \leq i \leq s\). Let \(\epsilon\) and \(\delta\) be positive values such that \(\epsilon \cdot s = (s-k) \cdot \delta\).
If $a_1 \cdot k > a_2 \cdot (s-k)$, we choose $a'_i = a_i + \varepsilon$, $i = 1,\ldots,k$, and $a'_i = a_i - \delta$, $i = 2,\ldots,s$, otherwise $a'_i = a_i - \varepsilon$, $i = 1,\ldots,k$, and $a'_i = a_i + \delta$, $i = 2,\ldots,s$. For both cases for sufficiently small $\varepsilon$, we have $F'(a_1,\ldots,a_s) > F'(a'_1,\ldots,a'_s)$, which contradicts to optimality of the solution. Hence we have the lemma.

The last step is to construct MST with $\varepsilon |V|$ edges (from preprocessing) and $|V|$ vertices, that takes $\gamma_1 k |V| + \gamma_2 |V| \log |V|$ running time. This concludes the computational complexity analysis of the total time of the parallel algorithm in Section 2.

Acknowledgements

This work was supported in part by National Science Foundation (#NSF/DBI-0354771, #NSF/ITR-IIS-0407204, #NSF/DBI-0542119), and also by a “Distinguished Scholar” grant from the Georgia Cancer Coalition.

References


of algorithms, 1,pp. 51-59.


Victor Olman is a Senior Research Scientist in Department of Biochemistry and Molecular Biology, University of Georgia. He received his Ph.D. degree in theory of probability and mathematical statistics in 1976 from the S. Petersburg University, Russia. His current research interests include the development and application of methods in cluster analysis, pattern recognition and statistics for analysis of microbial genome structures, pathway inference, binding sites prediction as well as for cancer bioinformatics. He has published over 60 publications in mathematical and biology related scientific journals.

Hongwei Wu is a post-doctoral research associate with the Computational Systems Biology Lab of the Department of Biochemistry and Molecular Biology and the Institute of Bioinformatics, University of Georgia. She received her Ph.D. and M.S., both in Electrical Engineering, from the University of Southern California in 2004 and 2002, respectively. She also received her M.Eng in Pattern Recognition and Intelligent Systems and B.Eng in Auto-
motic Control from the Tsinghua University, China, in 1999 and 1997, respectively. Her current research interests include computational biology/bioinformatics with focuses on comparative genomic analyses and computational reconstruction of biological networks, and computational intelligence theories and applications in computational biology/bioinformatics, signal processing, and pattern recognition. She is a member of IEEE and IEEE Computational Intelligence Society.

**Fenglou Mao** has received the PhD in Physical Chemistry from Peking University, China, in 2001. Currently he is an assistant research scientist at University of Georgia, USA. His main research interests include computational systems biology and computational structure biology.

**Ying Xu** is a chair professor of biochemistry and molecular biology and the director of the Institute of Bioinformatics, University of Georgia. He received his Ph.D. degree in theoretical computer science in 1991 from the University of Colorado at Boulder. His current research interests include (a) study of microbial genome structures and pathway inference, (b) cancer bioinformatics, and (c) development of computational tools in support of (a) and (b). He has over 100 publications and has given over 100 invited and contributed talks about his research work.