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 is 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 intraconnected 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 an 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 the software CLUMP.

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

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 do not 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, telecommunications, and many other fields. Many methods with different application objectives have been developed to solve the clustering problems, including K-Means [5], the Single Linkage Algorithm (SLA) [3] and other hierarchical clustering methods [21], a self-organizing map [5], the Markov Cluster Algorithm [4], and an 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 remain 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. The robustness of these algorithms represents another challenging issue. For example, the popular K-Means algorithm works well only if the clusters to be identified are embedded in disjoint convex sets and the clusters are comparable in size, while self-organizing-map-based methods work successfully only if a data set intrinsically has the structure with the 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 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 do not compare the performance of our method with that of others because the evaluation of 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 radii, the K-means will cut the big sphere, while our algorithm will separate the 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], an enormous amount of data has been generated that reflect different aspects of living organisms at molecular, cellular, and even ecosystem levels. In addition, an even larger amount of data have been derived from the experimental data by using bioinformatics tools such as predicted genes, promoters,

• *The authors are with the Computational System Biology Laboratory, Department of Biochemistry and Molecular Biology and Institute of Bioinformatics, University of Georgia, 120 Green Street, Athens, Georgia, 30602. E-mail: {olman, fenglow, hongweiw, xyn}@csbl.bmb.uga.edu.*

Manuscript received 24 Oct. 2006; revised 15 May 2007; accepted 3 Dec. 2007; published online 10 Dec. 2007.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-0194-1006.
Digital Object Identifier no. 10.1109/TCBB.2007.70272.

1545-5963/09/$25.00 © 2009 IEEE Published by the IEEE CS, CI, and EMB Societies & the ACM

gene families, protein structures, molecular interactions, and biological networks plus many intermediate results. The 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 millions of 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 an MST is that it generally preserves the structures of clusters in the sense that each cluster is generally presented as a subtree of an MST, as we have previously established [17], and hence, a clustering problem (or, generally, a 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 an MST [22]. In general, E is $O(V^2)$, which is often too large for practical applications with millions of data points. Despite the sequential essence of MST construction, a number of algorithms have been developed to parallelize calculations. A detailed analysis of this effort is done in the review in [16]. All suggested parallel algorithms present parallelization of calculations with heavy Message Passing Interface (MPI). In [20], the authors implemented a parallel version of the SLINK algorithm [23] using SIMD array processors. The parallel version of hierarchical clustering is presented in [14] on an n-node hypercube and an n-node butterfly. Another implementation of a parallel approach is reported in [9] with calculations distributed between processors and with the 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 an MST, while in [2], authors have dealt with the construction of a spanning tree using Symmetric Multiprocessors, though there is no guarantee that it will find an MST. The issue with these efforts is that they do not provide a practically useful and rigorous solution to the problem from the implementation point of view. In widely cited papers [3] and [19], the time complexity is $O(V\log(V))$ with $V/\log(V)$ processors, and in [13], it is $O(\log^{3/2} V)$ with $(V + E)$ processors, and while it presents an undisputed theoretical result, practically, it is not applicable with large data sets and today's technology. None of these papers provide Internet-accessible computer servers allowing a user to carry out MST construction with large data set. Theoretically, a more efficient algorithm represents a nontrivial 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 to implement, and while theoretical papers are dealing with a complexity $O(f(n))$, where n is a data set size, our goal was to minimize coefficient C in $O(f(n)) \sim Cf(n)$ for a large n with known function $f(n)$. Hence, we have implemented our own parallel algorithm for the MST construction, taking advantage of the Linux clusters that we have in our laboratory, which supports the data clustering computation server, based on the algorithms presented in this paper. The basic difference of our approach from those cited is that we do not 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 parallel MST constructions for subsets with moderate sizes. Using the parallel algorithm for the construction of an MST from a given data set, 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 the 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 "**clu**stering through **M**ST in **p**arallel."

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 and 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 $C = C_1 \bigcup C_2$, where $C_1 \neq \emptyset$, $C_2 \neq \emptyset$, and $C_1 \bigcap C_2 = \emptyset$

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

In other words, regardless of the 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 data set using a linear representation (LR) through the construction of an 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 LR is that data clusters in the given data set, as defined in [30], have a one-to-one correspondence with the "valleys" in this LR 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 data set can be identified through identifying valleys in this LR. In comparison to SLA, 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 is done in SLA.

2 Parallel Algorithm of MST Construction

We first present our parallel algorithm for the 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 the vertices of V_j;
- defining bipartite graphs $B_{ij} = \{V_i \bigcup V_j, E_{ij}\}$, 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 the vertices of V_i and the vertices of V_j, $i \neq j$, for each such pair of subgraphs from $\{G_i\}$;
- constructing an MST T_{ii} on each G_i and T_{ij} on each B_{ij} in parallel;
- building a new graph $G^0 = \bigcup T_{ij}$, $1 \leq i \leq j \leq s$, 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}, $1 \leq i \leq j \leq s$; and
- constructing an MST of G^0.

A mathematical proof is given in Appendix A to show that $\mathrm{MST}(G^0)$ is an $\mathrm{MST}(G)$, an MST of the original graph G. The key idea employed here is that we calculate in parallel an 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 an MST of G^0.

We now provide some analysis on the computational runtime 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_j and each bipartite graph B_{ij} to facilitate the 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 $O(|V_i||V_j| + (|V_i| + |V_j|) \log(|V_i| + |V_j|))$ for each bipartite 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

Dist	$T_C(D,V)$	R	$T_B(D,V)$	R
EUC	$(.192D+1.61)\ v^2$	.999	$(.385D+3.80)\ v^2$	.999
MAN	$(.206D+1.58)\ v^2$	.999	$(.413D+3.62)\ v^2$	.999
PEA	$(.173D+2.03)\ v^2$	.999	$(.346D+4.35)\ v^2$	.999

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 the computing time of the MST construction of the original graph.

The time for constructing the final $\mathrm{MST}(G^0)$ after merging the MSTs (G^0 is the original graph G without edges that do not belong to T_{ij}, $1 \leq i \leq j \leq s$), $T_M(s,V)$, accurately fits the following linear regression model with $R = 0.9931$:

$$T_M(s,V) = S*v*(0.401967*\log(v) - 0.464002) + 2.826016*v, \quad (1)$$

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 A, 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 $0.5V(V-1)$ edges is much faster than for a bipartite graph with $2V$ vertices and V^2 edges, the total computing time is

$$T(D,V,s) = T_B(D, V/s) + T_M(s,V). \quad (2)$$

Our test results showing the accuracy of (2) are presented in Fig. 1.

One can see that the total time is a monoextreme 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^0 (increasing the time for MST construction at the last step).

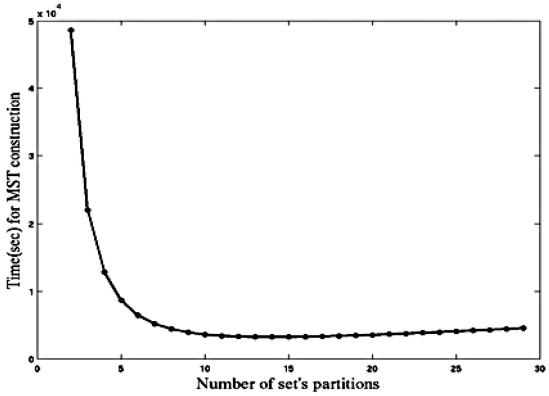


Fig. 1. The continuous line is a theoretical graph (based on the regression model) of the computing time for MST construction for a set of 1,000,000 40-dimensional vectors with the number of partitions ranging from 2 to 30, while circles are experimental results. The optimal number of partitions for this case is 14.

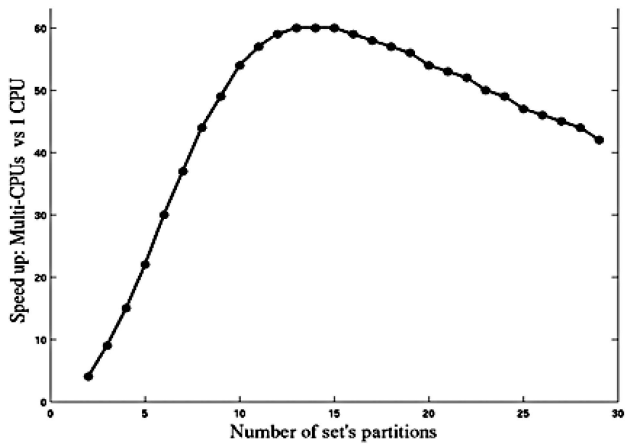


Fig. 2. The graph is an analytical form of speeding up (3), while circles are experimental points.

Given a complete graph with V D-dimensional points, (2) gives a tight estimate on the runtime of our parallel algorithm, as a function of s. The optimal runtime can be obtained by minimizing (2) with respect to s. We get the optimal value of s_o by differentiating (2) and equaling the derivative to zero as $s_o(D,V) \approx \sqrt[3]{2V(c_0 + c_1 D)/(0.401967 * \log(v) - 0.464002)}$, 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 speedup 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 the theoretical estimation of the speedup.

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

3 Cluster Identification

After building an MST for a given data set, our algorithm will construct an LR of the MST [30]. Given n data points, let $\mathrm{LR}[i]$ be the distance between the data point that gets added into the MST at the ith 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, $(i, \mathrm{LR}[i])$, $i = 1, \ldots, |V|$. A *valley* in the LR is defined as a list of indices s, $s+1, \ldots, s+t$ $(1 \le t \le |V| - s, s \ge 0)$ such that

$$\min(LR[s], LR[s+t+1]) > \max\{LR[i] | s < i < t+1\}. \quad (4)$$

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 the identification of valleys in the LR of a graph. We have employed the same simple algorithm that we have used for the identification of conserved binding motifs in genomic sequences for the identification of valleys [17]: recursive calculation of the maximum of LR in the valley belonging to $[0, |V|]$. The proof of correctness of this search is based on the fact that the starting and ending steps of a valley have the largest edges. It should be noted that an LR of a graph is not unique, but the aforementioned property holds for any LR of a graph. Fig. 3 shows a simple example for a set of two-dimensional data points in euclidean 2D space.

TABLE 2
Theoretical (SU_T) and Experimental (SU_E) Speedup Factors of the Parallel MST Construction Compared to the 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

Distance		EUC	MAN	PEA
V=250,000 D=20	SU_T	20	21	20
	SU_E	19	22	20
	s_o	9	9	9
V=500,000 D=40	SU_T	42	43	40
	SU_E	40	39	39
	s_o	12	12	12
V=750,000 D=60	SU_T	65	67	62
	SU_E	60	65	63
	s_o	15	15	14
V=1,000,000 D=80	SU_T	90	93	85
	SU_E	86	94	89
	s_o	17	17	17
V=1,100,000 D=100	SU_T	98	99	95
	SU_E	92	96	91
	s_o	18	18	17

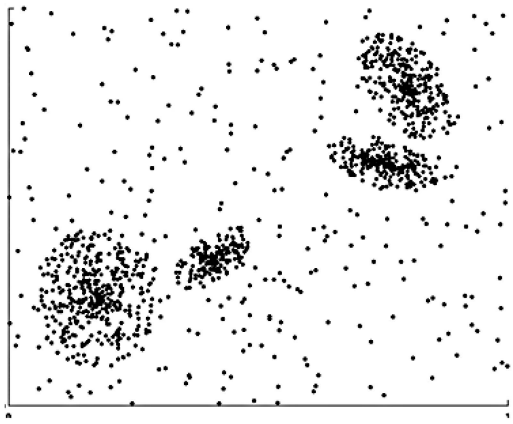


Fig. 3. Two-dimensional points with clustering structure.

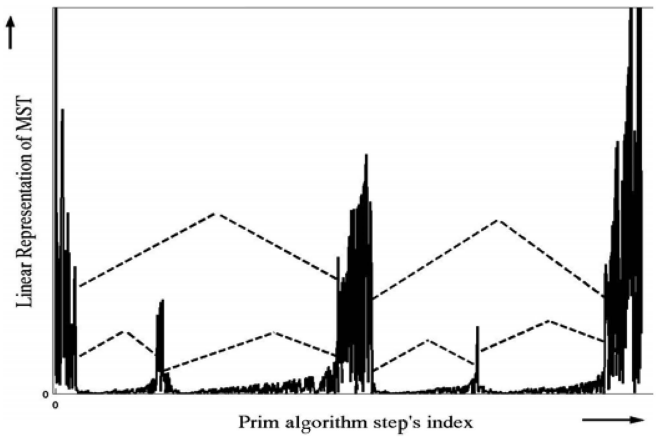


Fig. 4. The x-axis is the LR of the MST (indices of steps), and the y-axis represents the length of the edge that is used to include the corresponding vertex into MST. The lower V-shape dash lines represent four separate clusters, while the upper two V-shapes cover the aggregated clusters.

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

For each identified valley in the LR, 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 $z_m=\sum_{j=s}^{s+m}(LR[j]-L)/T$, $m=1,\ldots,t+1$, $z_0=0$, $T=\sum_{j=s}^{s+t+1}(LR[j]-L)$ 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 $\Pr obability(X\leq\rho)$, where $X=\max(x_i-x_{i-1}|1\leq i\leq t)/\min(1-x_t,x_1)$, and $0\leq x_1\leq x_2\leq\cdots\leq x_t\leq 1$ are sorted independent uniform observations in the 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 the 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-bit 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 an MST construction for a given graph. A Web interface for CLUMP is developed with popup 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 an arbitrary distance measure or provide precalculated 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 (the number of data points in a cluster), and the value for ρ (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 cluster C where $A\subset C\subset B$, 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 $(<=25)$; 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 a MySQL database, a php server-side script language, a postfix mail system, and an apache Web server. The back-end computer cluster is managed by a 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 that control the clustering or use 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 stand-alone CLUMP package and can be used to build clusters from MST, by using different parameters to create the 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 multiresolution 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,

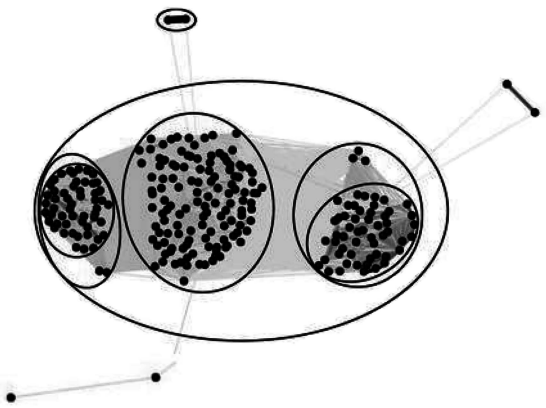


Fig. 5. The zoom-in view of one of the subgraphs of $G(V,E)$, where each node represents a gene, each edge indicates that the BLASTP e-value between the two connected genes is < 1.0, and the weight on the edge is proportional to the similarity measure $f(,)$ between the two genes. Each ellipse compasses a subset of genes that are functionally equivalent at a certain level.

$f(g_1,g_2)$, by incorporating both their sequence similarity and genomic context information as

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

where $h(,)$ denotes the sequence similarity measure, λ 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 $\sum_{ij}$ 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 are $\leq$ 1.0, we have obtained $\sim$ 46 million gene pairs involving 609,887 genes, which cover $\sim$92.7 percent 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 Fig. 5, $G(V,E)$ contains densely intraconnected subgraphs, each of which corresponds to a cluster of genes that are functionally equivalent to each other at a certain resolution level, and subgraphs (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 reflected not only 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. Therefore, we have defined the distance function for the MST-based clustering algorithm as $d(g_1,g_2)=[f^2(g_1,g_2)+\frac{\rho}{r}\sum_{k=1}^{r}f(g_1,g'_k)f(g_2,g'_k)]^{-1}$, 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 ρ determines the level of supporting evidence provided by such genes, and g'_k is the kth ranked gene in terms of the value $f(g_1,g'_k)f(g_2,g'_k)$ among such genes. The two parameters r and ρ provide flexibilities for a user to tailor the above distance function to his/her specific problems. In our case, we used $r=10$ and $\rho=0.6$.

When applied to $G(V,E)$ for clustering, CLUMP identified 51,205 gene clusters, which are organized into 5,339 multilevel and 15,770 single-level nonoverlapping trees, where the multilevel 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 one hand, that our clustering results are generally consistent with these two well-known classification systems, as reflected by the fact that $\sim$85 percent of those nontrivial COG clusters and $\sim$73 percent of those nontrivial 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 the functional annotation of unknown genes at different specificity levels. More discussions on this application can be found in [29].

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 subgraphs) is 42 times faster (8 minutes) than in the original analysis with one 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 a 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 minutes) was 36 times faster than that on the single CPU (144 minutes).

We have identified 1,100 statistically significant (with P value $<$ 0.001) nonoverlapping clusters (called CLUMP clusters in the rest of the discussion), covering 49,505 ($\sim$27 percent) genes. When comparing these CLUMP clusters with the COG [25] (3,827 clusters covering 85,013 genes) and Pfam [8] (1,087 clusters covering

76,020 genes) clustering results, we have found that 1) 307 out of all the 973 CLUMP clusters that have overlaps with the COG clustering results each has a Jaccard similarity coefficient with a COG cluster $\geq$ 2/3 and can therefore be considered to correspond to a COG cluster and 2) 219 out of all the 862 CLUMP clusters that have overlaps with the Pfam clustering results each has a Jaccard similarity coefficient with a Pfam cluster $\geq$ 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 completed 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 are the roles that 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 8t 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. Therefore, 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 the parallel clustering algorithm highly effective and practically useful.

6 Conclusion

The software CLUMP has proved to be a highly useful tool for clustering large quantities of biological data. Though the bottleneck in executing this program, i.e., the 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 x 86 processors. 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 $w(e)$ being the weight of $e \in E$, and $MST(G)$ be an MST of G, $MST(G) = \bigcup_{i=1}^{i=n-1} e_i$, 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 $V = U_i \bigcup \overline{U_i}$ in two subtrees formed by cutting an edge e_i from MST; then, $\min\{w(v,u)|v \in U_i, u \in \overline{U_i}\} = w(e_i)$.*

Proof. The proof is straightforward by the definition of MST. □

Lemma 2. *For any nonempty partition $V = U \bigcup \overline{U}$, there is an edge $e_0 = (v_0, u_0)$, $v_0 \in U$, $u_0 \in \overline{U}$, $e_0 \in MST(G)$, such that $\min\{w(v,u)|v \in U, u \in \overline{U}\} = w(e_0)$.*

Proof. Let us assume that it is not true, i.e., there are vertices $v_c \in U$ and $u_c \in \overline{U}$ such that $w(v_c, u_c) < w(e_0)$, where $e_0 = \text{argmin}\{w(e)|e \in MST(G)$ and e connects vertices of U and $\overline{U}\}$. By adding edge $e_c = \{v_c, u_c\}$ 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 $w(e_c) < w(e_0) <= w(e^0)$. Replacing e^0 by e_c, we get a spanning tree with a lower weight, and that contradicts to the fact that $MST(G)$ is an MST. □

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 $\overline{U_e}$, $V = U_e \bigcup \overline{U_e}$, and a corresponding nonempty partition of $V_0 = V_e \bigcup \overline{V_e}$, where $V_e = U_e \bigcap V_0$, and $\overline{V_e} = \overline{U_e} \bigcap V_0$. The partition V_0 is not empty because the edge e connects vertices in V_0. From Lemma 1, $w(e) = \min\{w(v,u)|v \in U_e, u \in \overline{U_e}\}$, and because of $V_e \subseteq U_e$ and $\overline{V_e} \subseteq \overline{U_e}$, we get $w(e) = \min\{w(\{v,u\})|v \in V_e, u \in \overline{V_e}\}$ 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. □

A.1 Method for MST Construction

Let us consider an s-way partition $V = \bigcup_{i=1}^{i=s} V_i$ 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 $V_{ij} = V_i \bigcup V_j$, and E_{ij} is the set of edges connecting vertices between V_i and V_j, $1 <= i < j <= s$, and G_M be the graph formed by merging MSTs $MST(G_i)$ and $MST(Q_{ij})$, $1 \leq i, j \leq s$. Then, $MST(G) = MST(G_M)$.*

Proof. It is 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. □

A.2 Complexity of the Method

By using Fibonacci heap in the implementation of Prim's algorithm, the runtime 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 and 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: a complete graph, a bipartite graph, and a sparse graph (merger of MSTs).

The runtime (RT) for the preprocessing step of our parallel algorithm (see Section 2), using $s * (s+1)/2$ processors, is $\max\{\max_{1<=i<=s} RT(G_i), \max_{1<=i<j<=s} RT(B_{ij})\}$. Our

goal is to get a partition of the set V into s subsets that would minimize $\max\{\max_{1<=i<=s} RT(G_i), \max_{1<=i<j<=s} RT(B_{ij})\}$. We introduce new variables $a_1, a_2, \ldots, a_s$, $\sum_{i=1}^{s} a_i = 1$, $a_i > 0$, such that $|V_i| \sim= a_i|V|$, i.e., $a_1, a_2, \ldots, a_s$ represent a partition of vertices. Therefore, we have $|E_i| \sim= |V|^2 a_i^2/2$ and $|E_{ij}| \sim= |V|^2 a_i a_j$. By ignoring the much smaller second term in RT, we obtain a minimization of $F(a_1, \ldots, a_s) = \max\{\max_{1<=i<=s}(a_i^2/2), \max_{1<=i<j<=s}(a_i a_j)\}$ 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 $\alpha_2 = 2a_1$. 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 $s=2$, $F(a_1, a_2) = 0.5 \cdot \max\{a^2, (1-a)^2, 2 \cdot a \cdot (1-a)\}$, and a simple analysis proves the correctness of the statement. For $s > 2$, since $F(a_1, \ldots, a_s) >= F_1(a_1, \ldots, a_s) = \max_{1<=i<j<=s}(a_i \cdot a_j)$, it is sufficient to prove the lemma for $F_1(a_1, \ldots, a_s)$. Let us assume that the statement is not a true, and without loss of generality, we have $a_1 = a_2 = \cdots = a_k > a_{k+1} >= \cdots >= a_s$, $1 <= k < s$. If $a_s = a_{s-1} = \cdots = a_{s-t} = 0$, $0 <= t < s - k$, we consider a new solution $a_i' = a_i - \varepsilon$, $i = 1, \ldots, k$, and $a_i' = a_i + \delta$, $i = s - t, \ldots, s$, where $\varepsilon \cdot s = \delta \cdot t$. By choosing a sufficiently small ε, we can reduce the value of $F'(a_1, \ldots, a_s)$, and it proves that for the optimal solution $a_i > 0$, $1 <= i <= s$. Let ε and δ be positive values such that $\varepsilon \cdot s = (s-k) \cdot \delta$. 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 a sufficiently small ε, we have $F'(a_1, \ldots, a_s) > F'(a_1', \ldots, a_s')$, which contradicts the optimality of the solution. Hence, we have the lemma. □

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

ACKNOWLEDGMENTS

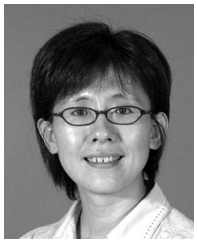
This work was supported in part by the US National Science Foundation (Grants NSF/DBI-0354771, NSF/ITR-IIS-0407204, and NSF/DBI-0542119) and also by a "Distinguished Scholar" grant from the Georgia Cancer Coalition.

REFERENCES

[1] S.F. Altschul et al., "Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs," *Nucleic Acids Research,* vol. 25, pp. 3389-3402, 1997.
[2] D.A. Bader and G. Cong, "A Fast, Parallel Spanning Tree Algorithm for Symmetric Multiprocessors (SMPs)," *J. Parallel and Distributed Computing,* vol. 65, no. 9, pp. 994-1006, 2005.
[3] J.L. Bentley, "Parallel Algorithm for Constructing Minimum Spanning Trees," *J. Algorithms,* vol. 1, pp. 51-59, 1980.
[4] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, 1981.
[5] C.M. Bishop, *Neural Networks for Pattern Recognition.* Oxford Univ. Press, 1995.
[6] O. Borůvka, "O jistém problému minimálnim. Práce Mor.Přorodověd," *Spol. v Brně (Acta Societ. Natur. Moravicae),* vol. 3, pp. 37-58, 1926.
[7] C. Dass, *An Introduction to Biological Mass Spectrometry.* John Wiley & Sons, 2002.
[8] R. Dementiev, P. Sanders, and D. Schultes, "Engineering an Eternal Memory Minimum Spanning Tree Algorithm," *Proc. Third IFIP Int'l Conf. Theoretical Computer Science (TCS '04),* pp. 195-208, 2004.
[9] Z. Du and F. Lin, "A Novel Approach for Hierarchical Clustering," *Parallel Computing,* vol. 31, no. 5, pp. 523-527, 2005.
[10] A.J. Enright, S. Van Dongen1, and S.A. Ouzounis, "An Efficient Algorithm for Large-Scale Detection of Protein Families," *Nucleic Acids Research,* vol. 30, no. 7, pp. 1575-1584, 2002.
[11] R.D. Finn et al., "PFAM: Clans, Web Tools and Services," *Nucleic Acids Research,* vol. 34, pp. 247-251, 2006.
[12] H.-R. Gregorius, "The Isolation Approach to Hierarchical Clustering," *J. Classification,* vol. 21, pp. 51-69, 2004.
[13] D.B. Johnson and P. Metaxas, "A Parallel Algorithm for Computing Minimum Spanning Trees," *Proc. Fourth Ann. ACM Symp. Parallel Algorithms and Architectures (SPAA '92),* pp. 363-372, 1992.
[14] X. Li and Z. Fang, "Parallel Clustering Algorithms," *Parallel Computing,* vol. 11, pp. 275-290, 1989.
[15] *Two-Hybrid Systems: Methods and Protocols (Methods in Molecular Biology),* P.N. Macdonald, ed., vol. 177. The Humana Press Inc., 2001.
[16] F. Murtagh, "Clustering in Massive Data Sets," *Handbook of Massive Data Sets,* pp. 501-543, 2002.
[17] V. Olman, D. Xu, and Y. Xu, "CUBIC: Identification of Regulatory Binding Sites through Data Clustering," *J. Bioinformatics and Computational Biology,* vol. 1, no. 1, pp. 21-40, 2003.
[18] V. Olman, C. Hicks, P. Wang, and X. Ying, "Gene Expression Data Analysis in Subtypes of Ovarian Cancer Using Covariance Analysis," *J. Bioinformatics and Computational Biology,* vol. 4, no. 5, pp. 999-1013, 2006.
[19] C.F. Olson, "Parallel Algorithms for Hierarchical Clustering," *Parallel Computing,* vol. V21, pp. 1313-1325, 1995.
[20] E.M. Rasmussen and P. Willet, "Efficiency of Hierarchical Agglomerative Clustering Using ICL Distributed Array Processors," *J. Documentation,* vol. 45, no. 1, pp. 1-24, 1989.
[21] H.C. Romesburg, *Cluster Analysis for Researchers,* 2004.
[22] *Handbook of Discrete and Combinatorial Mathematics,* K.H. Rosen, ed. CRC Press, 1999.
[23] R. Sibson, "SLINK: An Optimally Efficient Algorithm for the Single Link Cluster Methods," *Computer J.,* vol. 16, pp. 30-34, 1973.
[24] R.L. Tatusov, E.V. Koonin, and D.J. Lipman, "A Genomic Perspective on Protein Families," *Science,* vol. 278, pp. 631-637, 1997.
[25] R.L. Tatusov, D.A. Natale, I.V. Garkavtsev, T.A. Tatusova, U.T. Shankavaram, B.S. Rao, B. Kiryutin, M.Y. Galperin, N.D. Fedorova, and E.V. Koonin, "The COG Database: New Developments in Phylogenetic Classification of Proteins from Complete Genomes," *Nucleic Acids Research,* vol. 29, pp. 22-28, 2001.
[26] S.S. Wilks, *Mathematical Statistics.* John Wiley & Sons, 1962.
[27] H. Wu, F. Mao, V. Olman, and Y. Xu, "Accurate Prediction of Orthologous Gene Groups in Microbes," *Proc. IEEE Computational Systems Bioinformatics Conf. (CSB '05),* pp. 73-79, 2005.
[28] H. Wu, Z. Su, F. Mao, V. Olman, and Y. Xu, "Prediction of Functional Modules through Comparative Genome Analysis and Application of Gene Ontology," *Nucleic Acids Research,* vol. 33, pp. 2822-2837, 2005.
[29] H. Wu, F. Mao, V. Olman, and X. Ying, "Hierarchical Classification of Functionally Equivalent Genes of Prokaryotes," *Nuclear Acids Research,* vol. 35, pp. 2125-2140, 2007.
[30] Y. Xu, V. Olman, and D. Xu, "Clustering Gene Expression Data Using a Graph-Theoretic Approach: An Application of Minimum Spanning Tree," *Bioinformatics,* vol. 18, no. 4, pp. 526-535, 2001.

Victor Olman received the PhD degree in theory of probability and mathematical statistics in 1976 from the Saint Petersburg University, Russia. He is a senior research scientist in the Department of Biochemistry and Molecular Biology, University of Georgia. 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 more than 60 publications in mathematical and biology related scientific journals.

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



Hongwei Wu received BEng degree in automatic control and the MEng degree in pattern recognition and intelligent systems from the Tsinghua University, China, in 1997 and 1999, respectively, and the PhD and MS degrees in electrical engineering from the University of Southern California in 2004 and 2002, respectively. She is a postdoctoral research associate with the Computational Systems Biology Laboratory, Department of Biochemistry and Molecular Biology and the Institute of Bioinformatics, University of Georgia. Her current research interests include computational biology/bioinformatics with focus 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 the IEEE and the IEEE Computational Intelligence Society.

Ying Xu received the PhD degree in theoretical computer science from the University of Colorado at Boulder in 1991. He is a chair professor of biochemistry and molecular biology in the Computational Systems Biology Laboratory, Department of Biochemistry and Molecular Biology, and the director of the Institute of Bioinformatics, University of Georgia. His current research interests include the study of microbial genome structures and pathway inference, cancer bioinformatics, and development of computational tools in support of the aforementioned areas. He has more than 100 publications and has given more than 100 invited and contributed talks about his research work.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.