Microarrays Data Analysis for Cancer Disease on a Cluster of Computers

Amal Khalifa, Ph.D
College of Computer & Information Sciences
Princess Nora University
Riyadh, KSA

Dina Elsayad
Faculty of Computer & Information Sciences
Ain Shams University
Abbassyia, Cairo, Egypt

ABSTRACT
Clustering problem is one of the hottest research fields in microarrays data analysis. In Clustering, a set of observations are assigned into subsets (called clusters) such that observations in the same cluster are similar in some sense. One of the clustering approaches is based on the minimum spanning tree (MST). The MST-based clustering techniques consist of three main phases; MST construction, inconsistent edges identification and clusters identification.

The CLUMP algorithm (Clustering through Minimum spanning tree in parallel) is one of the MST-based clustering algorithms, which have been enhanced in the iCLUMP algorithm was improved using the cover tree data structure. This paper presents another improvement called iCLUMP-2 to enhance the edge inconsistency measure employed by both CLUMP and iCLUMP.

The performance of the implemented algorithm was tested on a 45 nodes cluster using cancer microarrays data sets. The results showed that the proposed algorithm outperformed both CLUMP and iCLUMP providing better speedup and efficiency. Furthermore the quality of cluster produced by the iCLUMP-2 algorithm is much better that those produced by both CUMP and iCLUMP.

General Terms
Bioinformatics, Microarrays data analysis, High performance computing.

Keywords
Clustering, microarrays, cancer, parallel algorithm, minimum spanning tree

1. INTRODUCTION
Bioinformatics is defined as the application of information technology to the field of molecular biology in order to manage the processing and the analysis of both genomic and molecular biological data. Bioinformatics is a very rich filed of research that includes so many areas such as: Genome annotation, sequence analysis, Computational evolutionary biology, Analysis of gene expression, Analysis of protein expression, Prediction of protein structure, and so on [1].
The microarray is a chip (usually made of glass or silicon) that assays large amounts of biological material using high-throughput screening methods [2]. This is a technology that allows studying the behavior of thousands of genes simultaneously under different conditions. The type of the microarray is determined based on the biological material used on the microarray chip: DNA microarrays [3], MMChips [4], Protein microarrays [5], Tissue microarrays [6], Cellular microarrays [7], Chemical compound microarrays [8], Antibody microarrays [9], and Carbohydrate arrays (glycoarrays) [1].

The biological material in DNA microarrays is DNA fragments, cDNA or oligonucleotides depending of chip construction technology [10]. In fact, this type of Microarrays is of special interest because it provides a useful tool in gene expression analysis that has been used effectively to discover the subsets of genes that are associated with occurrence of certain diseases such as cancer. However, the analysis of the microarrays data remains a big challenge because of the huge volume of data it produces. As shown in Fig. 1 The analysis process of microarrays data involves various computational tasks such as extracting differentially expressed genes, searching similar patterns of genes with a target gene, network analysis, clustering, and component analysis [11].
The clustering process aims to organize genes such that genes with similar expression patterns are grouped together to identify biologically relevant groups of genes inferring a common function or regulatory element [11].
A large number of clustering techniques have been proposed to solve the clustering process for the purpose of gene expression analysis. Generally speaking, clustering techniques can be classified into Hard and fuzzy clustering. In hard clustering each object belongs to only one cluster, while in Fuzzy clustering (also called soft clustering) an object can belong to more than one cluster with associated membership level [12]. Hard clustering can be further categorized into a number of subclasses including Hierarchical clustering [13][14][15], partitional clustering [16], graph based clustering [17] , and density-based clustering [18]. Many of these clustering algorithms are based on parallel computations because of the high dimensionality of the microarray data.
This paper presents a parallel Minimum spanning tree –based clustering algorithm. The algorithm; iCLUMP-2, is actually an enhancement over the CLUMP [19] in an attempt to improve its clustering quality. Another improvement was proposed previously by the authors in [20]

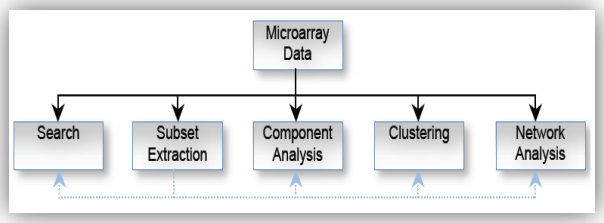

Figure 1. Computational tasks of microarrays data analysis.

The remaining of the paper is organized as follows; section2 provides a deep literature review on the minimum spanning tree (MST)-based clustering techniques. Section 3 provides an illustration of the used methods. Section 4 shows the experimental result. Finally section 5 shows the conclusion.

2. RELATED WORK

Generally speaking the gene expression data can be clustered in three ways [21]. First: genes clustering, similarly expressed genes across the samples are grouped together, where the gene function can be inferred. Second: samples clustering, the samples that have similar expression across the genes are grouped together. Third: Bi-Clustering (also called co-clustering or two-way clustering), cluster the genes and samples simultaneously based on their inter-relationship. In other words, if the data is represented as a 2-D matrix, the rows and the columns are clustered simultaneously where the result is a sub matrix. Using bi-clustering, we can extract genes that have similar behavior (co-express) under specific conditions.

Graph based clustering methods attracted a special interest in the field of gene expression analysis because of the intrinsic similarity between the microarrays data and the matrix representation of the adjacency information of a graph. Among the proposed techniques, minimum spanning tree (MST) based clustering has the advantage of the little impact of the cluster boundary shape, since the algorithm does not assume that data points are grouped around centers or separated by a regular geometric curve. As shown in Fig. 2, MST-based clustering algorithms consist of three main steps: MST construction, inconsistent edges identification, and finally the clusters are identified by removing the inconsistent edges from MST.

Figure 2 MST-based clustering algorithm steps

In this case, Microarrays data are represented as a fully connected undirected graph that consists of a set of N points in R^d where N is the number of genes and d is the number of samples. Using some distance measure; such as the Euclidian distance, the distance between each pair of genes is calculated. The resulting distance matrix is upper triangular matrix because the distance between the i^{th} gene and j^{th} gene $d(Gi, Gj)$ is the same as the distance between the j^{th} gene and i^{th} gene $d(G_j, G_i)$. Furthermore, the distance $d(G_i,G_i)$ is neglected because it has no meaning to measure the distance between the gene and itself. To construct the graph each gene is represented as a vertex where each pair of genes is connected by an edge whose weight is the distance between the two genes. Now the MST can be constructed from the resultant graph using either Kruskal or Prim algorithm [22].

The second phase of MST based clustering is the inconsistent edges identification. Inconsistent edges are the edges that may connect objects belonging to different clusters. The main difference between various MST-based clustering algorithms is the measure used to quantify edge inconsistency. One approach to do this is to exclude the edges with the highest weight [23]. That is, the removal of the longest edge results in two clusters, the removal of the next longest edge results in three clusters and so on. Sometimes the selection of these inconsistent edges depends on some threshold value. So, the clustering phase iteratively removes inconsistent edges from MST, calculate the ratio between the intra-cluster distance and inter-cluster distance and update the threshold value. This process is repeated until the threshold value is maximum and there are no edges to be deleted.

Although this approach has no specific requirements of prior knowledge of certain parameters nor the dimensionality of the data sets, its drawback resides in the fact that this edge removal policy may lead to a partition without sufficient evidence. To solve this problem, Zhong et al. proposed using two rounds of minimum spanning trees [24]. According to their approach two MSTs (T1 and T2) are constructed, then merged to construct the final MST (T). The clustering process works on the final T, where at least two edges must be removed in each step of which at least one edge comes from T1 and T2, respectively. This restriction provides more evidence in each cut. A two-round-MST based clustering is not affected by the sizes, shapes nor the densities of clusters but, it is not robust to outliers and cannot detect overlapping clusters.

Zhao and Zhang provided their yet robust MST-based clustering algorithm [25]. This algorithm is based on the direct clustering concept where the K needed clusters are constructed without the MST construction. The main idea of the direct clustering concept is implemented by selecting n-K (K ∈ [2, K]) shortest edges from the edge- weight matrix; then the nodes, linked by the same edge are combined together in a cluster. Although this algorithm performance is effective, but normally it's difficult to know the number of clusters in advance.

Xu et al. presented another MST-based clustering algorithm. The main idea of this approach is that each cluster corresponds to a sub-tree in the MST [26]. However the most challenging problems facing this kind of clustering algorithms is the limit on the size of the data sets they can effectively handle. Therefore, Olman et al [19] presented a parallel MST based clustering algorithm called CLUMP (clustering through MST in parallel). This algorithm identifies dense clusters in a noisy background and does not need prior information on the clusters or even the numbers of clusters. However, their parallelization effort focused only on the MST construction phase, leaving a lot of chances for more enhanced performance. Furthermore the clump algorithm still doesn't solve three types of clustering problems which are the well-separated cluster, connected cluster and the relaxed well-separated cluster.

Although, Wang et al [27] presented an algorithm that achieved satisfactory clustering results regarding these problems, the solution is still based on a sequential implantation. Hence, in this paper we focus on enhancing the performance of the CLUMP algorithm from two different points of view. The iCLUMP [20] algorithm enhances the parallel MST construction phase using efficient data structure called Cover Tree. Second, iCLUMP-2 enhances the clustering quality by using another metric for inconsistent edges identification.

3. METHODS

3.1 CLUMP Algorithm

Just like the rest of MST-based clustering algorithms, the CLUMP algorithm consists of three main steps as shown in Fig. 2. The graph construction phase is done according to Algorithm 1 in which the data is represented as a fully connected undirected graph G. Then according to Algorithm 2, the graph is partitioned into n sub-graphs Gi each of size k (k = number of vertices /n, if there is remaining vertices it's

added to the last sub-graph). The sub-graphs are combined in pairs (Gi , Gj) to construct a bipartite graph (Bij).
The number of the constructed sub-graphs and bipartite graphs is proportional to n requiring (n (n-1)/2) processing nodes to be working in parallel during the MST construction phase. More specifically, n processing nodes will be concerned with MST construction (Ti) for the sub-graphs Gi while the rest of the processing nodes will construct MST (Tij) for the bipartite graphs Bij. Then all the local MSTs are merged into one graph from which the final MST (T) is constructed providing the MST of the original graph. MST is constructed using Prim [22] algorithm and Fibonacci heap [22] data structure.
Thus, the complexity for constructing each sub-graph Gi can be expressed in eq. 1, while eq. 2 describes the complexity for each bipartite graph Bij.

$$O(|Ei|+|Vi| \log(|Vi|)) \quad (1)$$
$$O(|Vi||Vj|+(|Vi|+|Vj|) \log(|Vi|+|Vj|)) \quad (2)$$

The cluster identification phase is described in Algorithms 3 and 4. Notice that a cluster is identified by two edge indexes (left index and right index) where each cluster is partitioned recursively until the cluster size is less than or equal to the minimum cluster size. In this case, an inconsistent edge is defined as the one that has the max weight in the cluster range.

3.2 iCLUMP Algorithm

Even after a parallel implantation is employed, the MST construction phase is still considered the computational bottleneck of the CLUMP algorithm [19]. Therefore, an enhanced version of the CLUMP algorithm (called iCLUMP) has been proposed in [20]. The algorithm focused on speeding up the MST construction phase in CLUMP algorithm using the cover tree data structure [28]. The cover tree of a set S of n points is a leveled tree where each level is indexed by an integer i which decreases as the tree is descended. It was proved that the cover tree is constructed in O(c6 n log n) requiring O(n) space and answers the nearest neighbor query in O(c12 log n), where c is the expansion constant [29] and n is the number of nodes in the graph. Furthermore, William et al. succeed to use the cover tree to build the Euclidean MST in O(n log n) [28].

Hence, the iCLUMP algorithm proposed to enhance the nearest neighbor search step in the MST construction phase of the original CLUMP algorithm using the Cover tree (Algorithm 5) instead of the Fibonacci heap. More specifically the Algorithm 5 will replace steps 4.4.1, 4.4.2 and 5 in Algorithm 2. Therefore, the complexity of the MST construction will enhance (expressed in eq. 3 and, eq. 4) as compared with their respective ones in eq.1 and eq.2.

$$O(|Vi| \log(|Vi|)) \quad (3)$$
$$O((|Vi|+|Vj|) \log(|Vi| + |Vj|)) \quad (4)$$

3.3 iCLUMP-2 Algorithm

The edge inconsistency measure is the same in both CLUMP and iCLUMP. It is based on removing the longest edge resulting in two clusters, the removal of the next longest edge results in three clusters and so on. The removal of these edges depends on some threshold selection. The drawbacks of this strategy are the insufficient evidence, the difficulty of threshold selection and the inability to solve a clustering problem if one or more clusters are composed of sparse points [27]. Therefore, we propose another algorithm iCLUMP-2 that focuses on enhancing the inconsistency measure used in the CLUMP and iCLUMP algorithms. That is, to remove an edge 'i' it must satisfy the following criteria [27]:

- Wi>Wi-1 and Wi>Wi+1
- Wi > me + q σe

Where me and σe represent respectively the mean and standard deviation of all the edges that lie at most k steps away from the edge, and q is a predefined threshold. An inconsistent edge must simultaneously satisfy the two criteria.

Hence, here we propose Algorithm 6 to replace Algorithm 3 in order to enhanc the cluster identification phase in both the CLUMP and iCLUMP algorithm. Using this edge inconsistency measure, the proposed algorithm iCLUMP-2 can successfully solve the well-separated cluster, connected cluster and relaxed well-separated cluster problems [27].

Algorithm 1: Graph Construction	
Input	Data set of N points in R^d
Output	G(V,E): undirected fully connected graph, where V is the vertices set and E is the edges set
Step 1:	Add each point v as a vertex to V
Step 2:	d(u, v) = the distance between each pair of points u and v
Step 3:	Connect each pair of points (u, v) by edge e(u, v) where the weight of the edge is d(u, v)
Step 4:	Add e(u,v) to E

Algorithm 2 : MST Construction	
Input	G(V, E): undirected fully connected graph n: the partitions number
Output	MST T of G(V, E)
Step 1:	Calculate partition size K = |V|/n
Step 2:	Partition G into n sub-graphs $G_i = (V_i, E_i)$ each sub-graph of size K vertex
Step 3:	If |V| is not evenly divided by K Add the remaining vertices to the last sub-graph End if
Step 4:	Construct the MSTs for sub-graphs G_i and bipartite graphs B_{ij} in parallel
Step 4.1:	Assume that the distributed system consists of a set of processing nodes P_i ($1 \le i \le n(n-1)/2$)
Step 4.2:	Distribute the work among the nodes :
Step 4.3:	If Master Node:
Step 4.3.1:	x = n

Step 4.3.2: For i = 0 to n-1
Send G_i to P_i where i > 0
For j = i+1 to n-1
Send G_i and G_j to P_x
Increment x
End for
End for
Step 4.3.3: Construct MST T_0 for G_0
Step 4.3.4: Wait till all worker nodes send their MSTs
Step 4.3.5: Reduce and merge the whole MSTs in one graph M
End if
Step 4.4: If Worker node:
Step 4.4.1: If receive a graph G_i
Construct MST T_i
Send T_i to the master node
End if
Step 4.4.2: If receive a two graphs G_i and G_j
Define a bipartite graph $B_{ij} = (V_i \cup V_j, E_{ij})$ where $E_{ij} \supset E$ is the set of edges between V_i and V_j
Construct MST T_{ij} for bipartite graph B_{ij}
Send T_{ij} to the master node
End if
End if
Step 5: Construct MST T of M

Algorithm 3: Cluster identification	
Input	N: the number of edges in MST min_size: cluster minimum size
Output	C: the set of the data clusters where each cluster C_i is defined by edge ranges {Li, Ri}
Step 1:	Initialization: let the first cluster C_0 consists of the whole MST edges $L_0 = 1$, $R_0 = N$
Step 2:	Call the routine Cluster_Partition (L_0,R_0,min_size)
Step 3:	Build the hierarchical structure of the clusters in C
Step 4:	Clean the clusters (C`)
Step 5:	Rebuild the hierarchical structure of clusters in (C`)

Algorithm 4: "Cluster_Partition" Routine	
Input	L_i, R_i: the left and right cluster ranges min_size: cluster minimum size
Output	C: set of clusters
Step 1:	Let max_index be the index of edge with maximum weight in range {L_{i+1}, R_i}
Step 2:	Let Left_valley contains all edges in the range {L_i , max_index -1}
Step 3:	Add the cluster Left_valley to C
Step 4:	If size of Left_valley >= min_size
Step 4.1:	Cluster_Partition(L_i, max_index -1, min_size)
	End if
Step 5:	Let Right_valley contains all edges in the range { max_index, R_i}
Step 6:	Add the cluster Right_valley to C
Step 7:	If size of Right_valley >= min_size
Step 7.1:	Cluster_Partition(max_index , R_i, min_size)
	End if

Algorithm 5: Prim algorithm using Cover Tree	
Input	G(V,E): undirected fully connected graph, where V is the vertices set and E is the edges set
Output	MST T(V_T,E_T)
Step 1:	Construct the cover tree CT
Step 2:	Initialization: Let V_T consists of an arbitrary node from V $V_T = \{X\}$
Step 3:	While $\|V_T\| \neq \|V\|$
Step 3.1:	Let p an arbitrary node from V_T
Step 3.2:	Let Q_∞ points to the root level of CT $Q_\infty = C_\infty$
Step 3.3:	For i from ∞ to $-\infty$
Step 3.3.1:	Let Q be the children of the ith level of CT $Q = \{\text{Children}(q) : q \in Q_i \text{ and } q \notin V_T\}$
Step 3.3.2:	Find the cover set and exclude the points from Q that may not contain the nearest neighbor $Q_{i-1} = \{ q \in Q: d(p,q) \leq d(p,Q) + 2^i\}$
	End for
Step 3.4:	Find the node $q \in Q_{-\infty}$ such that d(p,q) is minimum for all

Step 3.5:	Add q to V_T
Step 3.6:	Add e(p,q) to E_T
	End while

Algorithm 6: iCLUMP-2 Cluster identification	
Input	N: the number of edges in MST q: threshold value k: max step size
Output	C: the set of the data clusters where each cluster C_i is defined by edge ranges {Li, Ri}
Step 1:	Initialize Q to be an empty set Q = {}
Step 2:	Q1 = set of edges that satisfy inequality $W_i > W_{i-1}$ and $W_i > W_{i+1}$
Step 3:	For each edge e in Q1
Step 3.1:	Let Q2 contains all neighbors of e that are at most K steps away m = mean of edges in Q2 var = variance of edges in Q2 If w_e > m + q * var Add e to Q End if
	End for
Step 4:	Get the clusters
Step 4.1:	let the first cluster C_0 consists of the edges from first edge till the first edge e in Q L_0 = 1, R_0 =(index of the first edge e) -1
Step 4.2:	For each e in Q
Step 4.2.1:	Li =(index of e) +1
Step 4.2.2:	Ri = (index of the next e) -1
Step 4.2.3:	Add the cluster C_i = {Li,Ri} to C
Step 5:	Build the hierarchical structure of the clusters in C
Step 6:	Clean the clusters (C`)
Step 7:	Rebuild the hierarchical structure of clusters in (C`)

4. EXPERIMENTAL RESULTS

In this section, some experiments were conducted to evaluate the performance of the proposed MST-based clustering algorithms iCLUMP and iCLUMP-2 against the original CLUMP. The algorithms were implemented using C++ with MPI. The experiments were conducted on a 45 processing nodes cluster. Each node is Intel ® Xeon® CPU E5620 @ 2.40 GHZ. Six large microarrays datasets were used for comparison, five of them are breast cancer datasets and one ovarian cancer. These datasets are publicly available on the GEO database (http://www.ncbi.nlm.nih.gov/) through their accession numbers. Table 1 shows the accession number and the size of each dataset.

Table 1: Microarrays datasets used for comparison

No.	Accession No.	Size (No. genes x No. samples)
1	GSE7390	22283 x 189
2	GSE2034	22283 x 256
3	GSE3494	22645 x 252
4	GSE6532	54675 x 88
5	GSE9195	54675 x 78
6	GSE6008	22283 x 104

The run times of the three algorithms were measured for the six data sets using different numbers of processing nodes. The number of nodes (p) depends on the number of partitions n where p = n(n-1)/2 . Also the speedup (Sp) and efficiency (Ep) were calculated in each case. The speedup is calculated using eq. 5. The speedup ranges from 1 to p, reflecting how much a parallel algorithm is faster than the sequential one. When Sp reaches p it's an ideal speedup case. However, according to the Amdahl's law (expressed in eq. 6) the maximum expected speedup that can be achieved by N processing nodes is limited by the sequential part time (1-P).

$$Sp = Ts/Tp \quad (5)$$

$$S(N) = \frac{1}{(1-tp)+\frac{tp}{N}} \quad (6)$$

Where Ts is the runtime of the sequential algorithm and Tp is the runtime of the parallel one. While tp indicates the parallel portion of the algorithm and (1-tp) is the sequential part of the algorithm.

Efficiency is another performance metric that can be calculated using eq. 7. It ranges between zero and one to estimate how the processors are well-utilized to solve the problem in hand compared to how much effort is wasted in communication and synchronization.

$$Ep = Sp/p \quad (7)$$

Table 2 shows the runtime, the speedup and efficiency of the three algorithms applied on the fourth data set with accession number GSE6532. Where the measured sequential time of the CLUMP algorithm was Ts = 283.47 seconds. As indicated from the results, the iCLUMP-2 algorithm outperformed both iCLUMP and the original CLUMP achieving better speedup and efficiency for all the tested values of p. For example, at p = 45 the achieved speedup was 11.44 compared to 9.21 and 7.51 the iCLUMP and CLUMP achieved respectively. Although, this value seems to be too far from the ideal speedup (Sp=45), the upper bound of Sp specified by Amdahl's law (eq. 6) wouldn't exceed 12. That is, according to the CLUMP formulation only 93% of the algorithm would actually benefit from the parallel implantation pushing the speedup away from the ideal one. Although, the Sp increases as the number of processing nodes increase, the efficiency Ep decreases due to the synchronization overhead. That is, as discussed before, the parallel part of the CLUMP algorithm focuses only on the MST construction phase (Algorithm 2) assigning the partitioned graphs Gi to n processing nodes while the rest of the nodes ((n-1)/2) work on the bipartite

graphs Bij. Since, the bipartite graph contains double the number of nodes in any partition graph Gi, it needs double the time to construct their MSTs. This inherent load imbalance greatly affects the efficiency as the number of partitions increase.

With the variety of the clustering algorithms, there is a strong need for cluster assessment techniques to decide which clustering algorithm is suitable than the others. One of these assessment techniques is called the Silhouette index [36], which is a confidence indicator for the membership of the sample to a specific cluster. This method assigns to each sample a quality measure called Silhouette width s(i) that is expressed in eq. 8. When s(i) is close to 1, the sample has been well-clustered, the value close to -1 implies that the sample has been misclassified; while the value close to zero indicates that the sample may be assigned to the nearest neighbor cluster. At the cluster level, the cluster Silhouette Sj; expressed in eq. 9, characterizes the heterogeneity and isolation properties of a cluster. Thus, the global Silhouette value GS; expressed in eq. 10, can be used to indicate the quality of the clustering algorithm as a whole. In fact, the greater the value of GS, better the clustering result achieved by that specific algorithm.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (8)$$

$$Sj = \frac{1}{m} \sum_{i=1}^{m} s(i) \quad (9)$$

$$GS = \frac{1}{c} \sum_{i=1}^{c} Sj \quad (10)$$

Where a(i) is the average distance between the ith sample and all of the samples included in the same cluster, b(i) is the minimum average distance between the ith sample and all the samples included in other clusters, m is the number of samples in the cluster, and c is the number of clusters.

The same experiment was repeated for all the datasets using 45 processing nodes. The results are listed in table 3, while fig. 4 and fig. 5, show respectively the computed speedup and efficiency for the three algorithms. The results show that the iCLUMP-2 provides the best performance in terms of both speedup and efficiency over all the datasets. Furthermore, the 4th and 5th data sets achieved a better performance over the other datasets, because the cover tree data structure works better on the data with high dimensionality [29].

Table 3 shows the global Silhouette values for the iCLUMP and iCLUMP-2 algorithms. It is worthy to note that the quality of clusters produced by CLUMP and iCLUMP is the same because they actually employ the same edge inconsistency measure. Nevertheless, the results show that the clustering quality of iCLUMP-2 is much better than the other two algorithms where the global Silhouette value GS reached 54.35 with approximately 85% improvement over both CLUMP and iCLUMP.

Table 2: The runtime, speedup and efficiency of the CLUMP, iCLUMP and iCLUMP-2 applied on dataset with accession number GSE6532

	CLUMP			iCLUMP			iCLUMP-2		
p	**Tp**	**Sp**	**Ep**	**Tp**	**Sp**	**Ep**	**Tp**	**Sp**	**Ep**
3	172.78	1.64	0.55	102.87	2.76	0.92	95.78	2.96	0.99
6	90.10	3.15	0.52	61.11	4.64	0.77	56.04	5.06	0.84
10	80.42	3.52	0.35	46.93	6.04	0.60	41.72	6.79	0.68
15	53.45	5.30	0.35	39.92	7.10	0.47	35.53	7.98	0.53
21	48.43	5.85	0.29	37.08	7.64	0.36	31.67	8.95	0.43
28	43.98	6.45	0.23	32.82	8.63	0.31	27.06	10.47	0.37
36	40.00	7.09	0.20	31.23	9.08	0.25	25.15	11.27	0.31
45	37.73	7.51	0.17	30.79	9.21	0.20	24.78	11.44	0.25

Table 3: The speedup, efficiency and GS of the CLUMP, iCLUMP and iCLUMP-2 when applied on 6 different datasets using 45 processing nodes

		CLUMP			iCLUMP			iCLUMP-2		
No.	**Ts**	**Sp**	**Ep**	**GS**	**Sp**	**Ep**	**GS**	**Sp**	**Ep**	**GS**
1	68.39	4.34	0.1	24.23	5.29	0.12	24.23	6.01	0.13	32.54
2	58.67	5.30	0.11	17.34	5.54	0.12	17.34	5.88	0.13	26.45
3	96.81	5.50	0.12	30.47	6.43	0.14	30.47	6.73	0.15	42.78
4	283.47	7.51	0.17	42.78	9.21	0.20	42.78	11.44	0.25	50.36
5	412.69	10.43	0.23	47.98	11.18	0.23	47.98	13.25	0.29	58.89
6	64.82	5.77	0.13	51.45	7.52	0.17	51.45	8.17	0.18	65.34

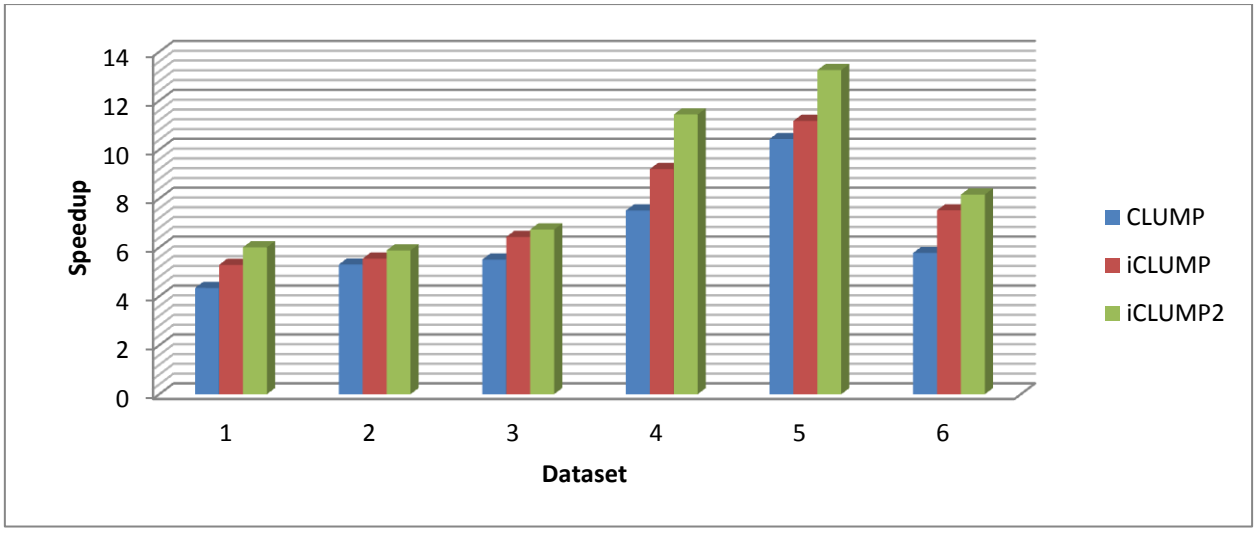

Figure 4. The speedup of the three algorithms CLUMP, iCLUMP and iCLUMP-2 on 45 processing nodes.

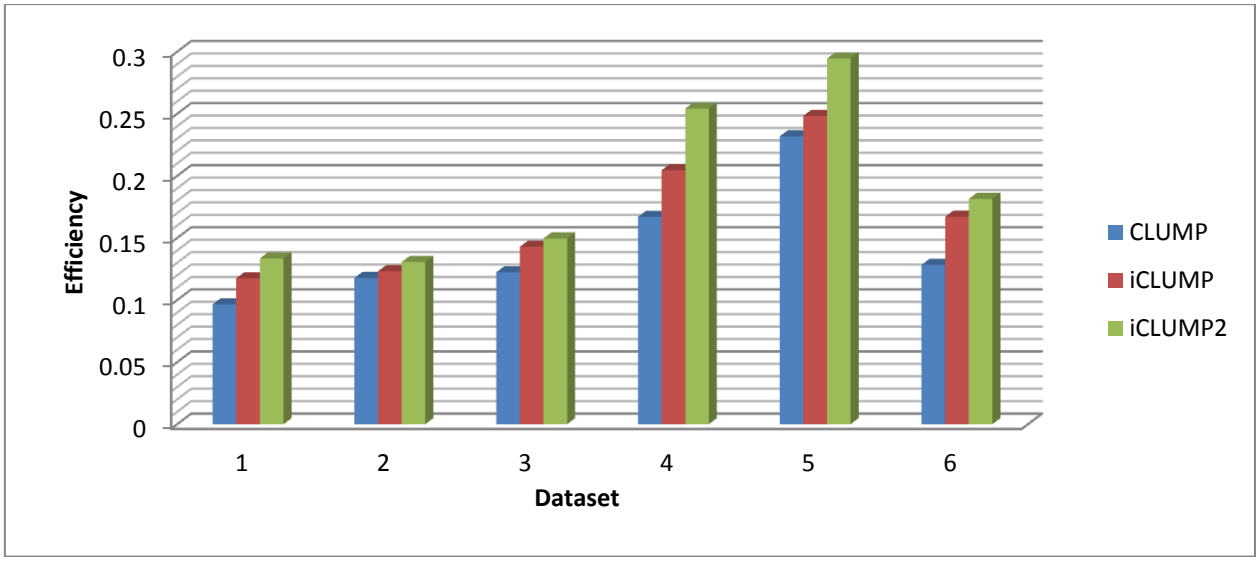

Figure 5. The Efficiency of the three algorithms CLUMP, iCLUMP and iCLUMP-2 on 45 processing nodes.

5. CONCLUSIONS

This paper presented another improvement over the iCLUMP algorithm providing lower complexity and higher clustering quality for microarrays datasets. Although, the iCLUMP algorithm successfully used the cover tree data structure to reduce the complexity of the MST construction phase from O(|Ei|+|Vi| log (|Vi|)) to become O(Vi log Vi), the proposed algorithm iCLUMP-2 successfully reduced the runtime with more cluster quality by employing another inconsistent edge measure other than the longest edge approach. Using a number of cancer microarrays data sets, the experimental results showed that iCLUMP-2 outperformed both CLUMP and iCLUMP algorithms in terms of speedup and efficiency. For example, on a cluster of 45 processing nodes, the speedup reached 11.44 compared to 9.21 and 7.51 achieved by the iCLUMP and CLUMP respectively. In addition, iCLUMP-2 enhanced the clustering quality of both CLUMP and iCLUMP by approximately 85%.

6. REFERENCES

[1] Aluru, S. Handbook of computational molecular biology. CRC Press, 2006.

[2] Culf, A.S. and Cuperlovic-Culf, M. and Ouellette, R.J. Carbohydrate microarrays: survey of fabrication techniques. OMICS: A Journal of Integrative Biology 2006; 10(3): 289-310.

[3] Schena, M. and Shalon, D. and Davis, R.W. and Brown, P.O. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. Science 1995; 270(5235): 467-470.

[4] Meenakshisundaram, K. and Carmen, L. and Michela, B. and Diego, D.B. and Rosaria, V. and Gabriella, M. Existence of snoRNA, microRNA, piRNA characteristics in a novel non-coding RNA: x-ncRNA and its biological implication in Homo sapiens. Journal of Bioinformatics and Sequence Analysis 2009; 1(2): 31-40.

[5] Stoevesandt, O. and Taussig, M.J. and He, M. Protein microarrays: high-throughput tools for proteomics. Expert Review of Proteomics 2009; 6(2): 145-157.

[6] Camp, R. L. Charette, L. A. Rimm, D. L. Validation of Tissue Microarray Technology in Breast Carcinoma.

LABORATORY INVESTIGATION 2000; 80(12): 1943-1949.

[7] Chen, D.S. and Davis, M.M. Cellular immunotherapy: Antigen recognition is just the beginning. Springer seminars in immunopathology 2005; 27(1): 199-127.

[8] Ma, H. and Horiuchi, K.Y. Chemical microarray: a new tool for drug screening and discovery. Drug discovery today 2006; 11(13): 661-668.

[9] Rivas, L.A. and García-Villadangos, M. and Moreno-Paz, M. and Cruz-Gil, P. and Gómez-Elvira, J. and Parro, V. A 200-Antibody Microarray Biochip for Environmental Monitoring: Searching for Universal Microbial Biomarkers through Immunoprofiling. Analytical Chemistry 2008; 80(21): 7970-7979.

[10] Li, S. and Li, D. DNA microarray technology and data analysis in cancer research. World Scientific Pub Co Inc, 2008.

[11] Yang, Y. and Choi, J.Y. and Choi, K. and Pierce, M. and Gannon, D. and Kim, S. BioVLAB-Microarray: Microarray Data Analysis in Virtual Environment. IEEE Fourth International Conference on eScience, 2008, 159-165.

[12] D. Dembele and P. Kanstner. Fuzzy C-means method for clustering microarray data. Bioinformatics 2003; 19(1): 973-980.

[13] Ivan G. Costa, Francisco de A.T. de Carvalho and Marcilio C.P. de Souto. Comparative Analysis of Clustering Methods for Gene Expression Time Course Data. Genetics and Molecular Biology 2004; 27(4): 623-631.

[14] Carlos Cotta, Pablo Moscato. A memetic-aided approach to hierarchical clustering from distance matrices: application to gene expression clustering and phylogeny. Biosystems 2003; 72(1): 75-97.

[15] Sudip Seal, Srikanth Komrina, Srinivas Aluru. An optimal hierarchical clustering algorithm for gene expression data. Information Processing Letters, 2004; 39(3): 143-147.

[16] C.M. Bishop. Neural Networks for Pattern Recognition. Oxford Univ.Press, 1995.

[17] Kanungo, S. and Sahoo, G. and Gore, M.M. A Co-Clustering Technique for Gene Expression Data Using Bi-Partite Graph Approach. International Conference on Bioinformatics and Biomedical Engineering 2010; 1-5.

[18] De Bin, R. and Risso, D. A novel approach to the clustering of microarray data via nonparametric density estimation. BMC bioinformatics 2011; 12(1): 49-56.

[19] Olman, V. and Mao, F. and Wu, H. and Xu, Y. Parallel clustering algorithm for large data sets with applications in bioinformatics. IEEE/ACM Transactions on Computational Biology and Bioinformatics 2009; 6(2): 344-352.

[20] Elsayad, D. Khalifa, A. Khalifs, M.E, El-Horbaty, E.-S. An improved parallel minimum spanning tree based clustering algorithm for microarrays data analysis. 8th International Conference on Informatics and Systems (INFOS), May 2012; 66-72.

[21] Kerr, G. and Ruskin, H.J. and Crane, M. and Doolan, P. Techniques for clustering gene expression data. Computers in biology and medicine 2008; 38(3): 283-293.

[22] K.H. Rosen. Handbook of Discrete and Combinatorial Mathematics. CRC Press, 1999.

[23] Jana, PK and Naik, A. An efficient minimum spanning tree based clustering algorithm. Proceeding of International Conference on Methods and Models in Computer Science 2009; 1-5.

[24] Zhong, C. and Miao, D. and Wang, R. A graph-theoretical clustering method based on two rounds of minimum spanning trees. Pattern Recognition 2010; 43(3): 752-766.

[25] Zhao, W.L. and Zhang, Z.G. An Improved Algorithm for Clustering Gene Expression Data Using Minimum Spanning Trees. Applied Mechanics and Materials 2010; 29(1): 2656-2661.

[26] XY. Xu, V. Olman, and D. Xu. Clustering Gene Expression Data Using a Graph-Theoretic Approach: An Application of Minimum Spanning Tree. Bioinformatics 2001; 18(4): 526-535.

[27] Wang, G.W. and Zhang, C.X. and Zhuang, J. and Yu, D.H. Clustering based on sequential representation of minimum spanning tree. International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR), 2011.

[28] William B. March and Parikshit Ram and Alexander G. Gray. Fast Euclidean minimum spanning tree: algorithm, analysis, and applications. In Proceedings of KDD 2010; 603-612.

[29] D. Karger and M. Ruhl. Finding nearest neighbors in growth restricted metrics. Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC) 2002; 741–750.