A parallel Clustering algorithm based on minimum spanning tree for microarrays data analysis

DINA ELSAYAD, AMAL KHALIFA, ESSAM KHALIFA, and EL-SAYED M. EL-HORBATY
Faculty of Computer and Information Sciences
Ain Shams University
Abbassia, Cairo - Egypt
dina.elsayad@fcis.asu.edu.eg, amal@fcis.asu.edu.eg, esskhalifa@eun.eg, shorbaty@cis.asu.edu.eg

Abstract: Clustering is partitioning a set of observation into groups called clusters, where the observation in the same group has a common characteristic. One of the best known algorithms for solving the microarrays data clustering problem using minimum spanning tree (MST) is CLUMP algorithm (Clustering algorithm through MST in Parallel) which identifies a dense clusters in a noisy background. The MST construction phase of the CLUMP is the time consuming phase. This paper presents an improved version of CLUMP algorithm called iCLUMP (improved Clustering algorithm through MST in Parallel). iCLUMP enhances the speedup of MST construction using the cover tree data structure. The implementation shows that iCLUMP is efficient than CLUMP in terms of complexity and runtime.

Key-Words: Clustering; Minimum spanning tree; Microarrays; Bioinformatics, Parallel algorithm.

1 Introduction

The Bioinformatics is the application of information technology to the molecular biology field. It helps the biologists in the data management and analysis. One of the bioinformatics research areas [1] is Microarrays.

A microarray is a chip made of glass or silicon or plastic. It assays large amounts of biological material using high-throughput screening methods [2]. There are a number of types of microarrays include DNA microarrays [3]. The biological material in DNA microarrays is DNA fragments, cDNA or oligonucleotides depending on chip construction technology. The DNA Microarrays is a useful tool in gene expression analysis for drug development and test, function annotation and diseases diagnosis such as cancer.

The analysis of microarrays data contains many tasks such as extracting differentially expressed genes, searching similar patterns of genes with a target gene, network analysis, clustering, and component analysis [4].

One of the microarrays data analysis tasks is the clustering. Clustering is a widely used technique in many fields such as machine learning, pattern recognition, image analysis, data mining and bioinformatics. Generally, clustering defined as the partitioning a set of observations into groups called clusters, where the observations in the same group have a common characteristic. There is a many research focused on the data clustering. One of the clustering techniques is the minimum spanning tree (MST) based clustering. Almost the recent clustering techniques is a sequential. But, the massive volume of produced data is a challenge. So, parallel algorithms are needed. One of the clustering parallel algorithms is CLUMP (Clustering through MST in parallel), which identifies a dense clusters in a noisy background. Although it's performance is efficient but the MST construction is a time consuming phase. So our focus is to enhance the performance of CLUMP through enhancing the MST construction phase.

The rest of the paper is organized as following. Section 2 provides the needed background and the related work. Section 3 provides the proposed algorithm. The implementation is discussed in section 4. Finally section 5 provides the conclusion and future work.

2 Background and Related work

Microarrays data clustering is a powerful tool for studying genes relationships and gene functions. It also plays an important role in discovering and understanding different classes and subclasses of diseases such as cancer.

Cluster analysis or clustering is the assignment of a set of observations into subsets (called clusters) so
that observations in the same cluster have common characteristics. As shown in Fig.1 clustering can be classified to

- Hard clustering where each object belongs to only one cluster.
- Fuzzy clustering also called soft clustering where an object can belongs to more than one cluster with associated membership level.

An example of the fuzzy clustering is the fuzzy c-means clustering algorithm [5, 6, 7]. It minimizes the weighted sum of distances between the gene and the cluster center.

The hard clustering can be further classified into a number of sub-types. The hierarchical algorithms [8] find successive clusters using previously established clusters. These algorithms can be either agglomerative ("bottom-up") or divisive ("top-down").

The K-means is partitional clustering algorithm which partitions the objects into K clusters where each object belongs to the cluster with the nearest mean [9].

The Density-based clustering algorithms [10, 11] are devised to discover arbitrary-shaped clusters. In this approach, a cluster is regarded as a region in which the density of data objects exceeds a threshold.

The Two-way clustering [12, 13] (also called co-clustering or bi-clustering) is clustering method where the genes and samples are clustered simultaneously based on their inter-relationship. Kanungo and et al. presented a bi-clustering algorithm based on graph representation especially the bi-partite graph [14].

Another graph based clustering is the minimum spanning tree (MST) based clustering [15, 16, 17, 18]. The main advantage of MST-based clustering is the little impact of the cluster boundary shape since, these algorithms 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, removing the inconsistent edges from MST to get the clusters. To construct the MST from microarrays data, the data is resented as undirected fully connected graph where the genes is represented as a vertices and each pair of genes is connected by edge where the edge weight is the distance between the two genes measured in any distance measure such as Euclidean distance. The MST constructed using Prim or Kruskal algorithm [19]. Then the clusters are identified using the MST representation through the inconsistent edges removing. The inconsistent edges are the edges that may connect objects belong to different clusters.

Jana and et al. presented MST-based clustering algorithm [17] where the MST is constructed using Kruskal algorithm and use it in the clustering process, which uses longest edges as the inconsistent edge measure. Removing of these edges depend on the threshold selection. Although this approach has no specific requirement of prior knowledge of some parameters and the dimensionality of the data sets, the drawback is that every edge's removal will result in two clusters but this may lead to a partition without sufficient evidence. To avoid this problem, Zhong and et al. provided their clustering method based on two rounds of minimum spanning trees [18]. In the first round MST T1 is constructed, in the second round another MST T2 is constructed by considering only the edges that do not belong to T1. Then, merge the T1 and T2 to construct the final MST T. The clustering process works on the final MST T. To partition a two-round MST based graph, at least two edges must be removed, of which at least one edge comes from T1 and T2, respectively. This restriction provides more evidence in each cut. Although, the two-round-MST based clustering is not affected by the sizes, shapes and the densities of clusters, it is not robust to outliers and cannot detect the overlapping clusters.

Zhao and et al. provided their robust MST based clustering algorithm [17]. It based on the direct clustering concept where K needed clusters are constructed without the construction of MST.
Although, this algorithm performance is effective, but normally it’s difficult to know the number of clusters in advance.

Olman and et al. [18] presented parallel MST based clustering technique called CLUMP (Clustering through MST in Parallel) which identifies dense clusters in a noisy background. They avoid the challenging problems facing most of the recently clustering algorithms that is the limit on the size of the data sets they can effectively handle and the robustness of theses algorithms. Also, this algorithm does not need prior information on the clusters or even the numbers of clusters. The MST construction phase is still considered the computational bottleneck of the CLUMP algorithm. Thus, the complexity for constructing MST of each subgraph is still a limitation. The original CLUMP algorithm constructs the MST of each subgraph in parallel. The proposed algorithm iCLUMP consists of three main steps as shown in Fig.2. In the graph construction phase, the data is represented as an undirected fully connected graph G. Then the graph is partitioned into n subgraphs Gi. The sub-graphs are combined in pairs (Gi, Gj) to construct a bipartite graph (Bi). After that, the MST is constructed for each Gi and Bji in parallel. 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. The original CLUMP algorithm constructs the MST using Prim algorithm and Fibonacci heap [19] data structure. Thus, the complexity for constructing MST of each sub-graph Gi can be expressed in equation 1, while equation 2 describes the complexity for each bipartite graph Bij.

\[
O(|E_i|+|V_i| \log(|V_i|)) \quad (1)
\]

\[
O(|V_i||V_j|+(|V_i|+|V_j|) \log(|V_i|+|V_j|)) \quad (2)
\]

iCLUMP construct MST using Prim algorithm and Cover tree [20] data structure (Algorithm 1) instead of Fibonacci heap data structure used in CLUMP algorithm. Thus, the complexity for constructing MST of each sub-graph Gi can be expressed in equation 3, while equation 4 describes the complexity for each bipartite graph Bij. Obviously, this is much less than the corresponding complexity in the original CLUMP algorithm expressed in equation 1 and equation 2 respectively.

\[
O(|V_i| \log(|V_i|)) \quad (3)
\]

\[
O((|V_i|+|V_j|) \log(|V_i|+|V_j|)) \quad (4)
\]

The cover tree of a set S of n points is a levelled tree where each level is indexed by an integer i which decreases as the tree is descended. Each level covers all nodes in the level beneath it. Let Ci be the set of points in level i, the cover tree has the following invariants

- Nesting invariant: Ci ⊂ Ci+1
- Covering invariant: For every p ∈ Ci+1, there exists q ∈ Ci such that d(p, q) < 2i and q is the parent of p.
- Separation invariant: For all p, q ∈ Ci, d(p, q) > 2i.

The cover tree is constructed in O(c6 n log n) requiring O(n) space and answers the nearest neighbour query in O(c12 log n), where c is the expansion constant[21]. William et al. succeed to use the cover tree to build the Euclidean MST in O(n log n) [22].

After the MST is constructed, it’s used in the clusters identification phase [18]. It is a recursive process in which the first cluster consists of the whole edges in MST and each cluster is partitioned recursively until the cluster size is less than or equal to the minimum cluster size. The cluster is partitioned according to the selection policy of an inconsistent edge. In this case an inconsistent edge is defined as the one that has the max weight in the cluster range. Hence the cluster is partitioned into a left valley and a right valley and the process is repeated recursively in each one of them.

4 Implementation

In this section, we will conduct some experiments to evaluate the performance of the proposed MST-based clustering algorithm iCLUMP against the original CLUMP. The algorithms are 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 are used for comparison, five of them are breast cancer datasets and one ovarian cancer. These datasets are publicly available from 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.

The run times of the two algorithms are measured for the six data sets using different numbers of processing nodes. The number of
processing nodes (p) depends on the number of partitions n where \( p = n(n-1)/2 \).

Table 1: Microarrays datasets used for comparison

<table>
<thead>
<tr>
<th>No.</th>
<th>Accession Number</th>
<th>Size (number of genes x number of samples)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GSE7390</td>
<td>22283 x 189</td>
</tr>
<tr>
<td>2</td>
<td>GSE2034</td>
<td>22283 x 256</td>
</tr>
<tr>
<td>3</td>
<td>GSE3494</td>
<td>22645 x 252</td>
</tr>
<tr>
<td>4</td>
<td>GSE6532</td>
<td>54675 x 88</td>
</tr>
<tr>
<td>5</td>
<td>GSE9195</td>
<td>54675 x 78</td>
</tr>
<tr>
<td>6</td>
<td>GSE6008</td>
<td>22283 x 104</td>
</tr>
</tbody>
</table>

Fig. 4 and Fig.5 show the runtime for the two algorithms applied on the 4th and 5th data set. As indicated from the results, the iCLUMP algorithm outperformed the original CLUMP achieving better runtime for all the tested values of the processing nodes p. For example, at \( p = 45 \) the achieved runtime was 30.79 and 31.9262 compared to 37.73 and 39.55 by CLUMP algorithm for the 4th and 5th data set respectively. Fig. 5 and Fig.6 show the results of the MST construction and its effect on the overall execution time of the two algorithms at \( p = 45 \), respectively. The result indicate that iCLUMP is more efficient than CLUMP providing better performance reaching a average speedup of 1.24 and 1.19 for MST construction time and algorithm overall execution time respectively.

Fig. 3: Runtime for data set GSE6532, the x-axis represents the number of processing nodes and y-axis represents the runtime in seconds.

Fig. 4: Runtime for data set GSE9195, the x-axis represents the number of processing nodes and y-axis represents the runtime in seconds.

Fig. 5: MST construction time in seconds

Fig. 6: Algorithm overall execution time in seconds
5 Conclusion and Future work

The proposed algorithm iCLUMP is MST-based clustering technique that improves the MST construction phase of the original CLUMP algorithm using cover tree data structure. The iCLUMP is outperformed the CLUMP algorithm where the complexity for constructing MST for each sub-graph $G_i$ is $O(|V_i| \log(|V_i|))$, while the complexity for each bipartite graph $B_{ij}$ is $O(|E_{ij}| + |V_i| \log(|V_i| + |V_j|))$ instead of $O(|E_{ij}| + |V_i| \log(|V_i|))$ and $O(|V_i| |V_j| + |V_i| |V_j| \log(|V_i| + |V_j|))$ for the CLUMP respectively. Obviously, this is much less than the corresponding complexity in the original CLUMP algorithm.

The implantation was conducted on 45 processing nodes cluster, the runtime and MST construction time was measured for the two algorithms using 6 microarrays data sets. At 45 processing node the iCLUMP overall runtime was 30.79 instead of 37.73 for CLUMP using data set of size 54675 x 88 while it reached 31.93 instead of 39.55 using data set of size 54675 x 78. The MST average speedup reaches 1.24 and overall execution time reaches 1.19.

As a future work a different distance measure can be added such as Manhattan distance and Pearson correlation.

References:
Appendix:

<table>
<thead>
<tr>
<th>Input</th>
<th>G(V,E): undirected fully connected graph, where V is the vertices set and E is the edges set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>MST T(V_T,E_T)</td>
</tr>
</tbody>
</table>

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| ≠ |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_∞ = C_∞
Step 3.3: For i from ∞ to -∞
Step 3.3.1: Let Q be the children of the ith level of CT
Q = {Children(q) : q ∈ Q_i and q ∉ 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 = { q ∈ Q : d(p,q) ≤ d(p,Q) + 2^i }  
End for
Step 3.4: Find the node q ∈ Q_∞ such that d(p,q) is minimum for all q
Step 3.5: Add q to V_T
Step 3.6: Add e(p,q) to E_T
End while

Algorithm 1: Build MST algorithm using cover tree data structure