# Clustering Nodes in Large-Scale Biological Networks Using External Memory Algorithms

Ahmed Shamsul Arefin[1], Mario Inostroza-Ponta[2], Luke Mathieson[3], Regina Berretta[1,4], and Pablo Moscato[1,4,5,*]

[1] Centre for Bioinformatics, Biomarker Discovery and Information-Based Medicine, The University of Newcastle, Callaghan, New South Wales, Australia
[2] Departamento de Ingeniería Informática, Universidad de Santiago de Chile, Chile
[3] Department of Computing, Faculty of Science, Macquarie University, Sydney Australia
[4] Hunter Medical Research Institute, Information Based Medicine Program, Australia
[5] ARC Centre of Excellence in Bioinformatics, Callaghan, NSW, Australia
{Ahmed.Arefin,Regina.Berretta,Pablo.Moscato}@newcastle.edu.au, Mario.Inostroza@usach.cl, Luke.Mathieson@mq.edu.au

**Abstract.** Novel analytical techniques have dramatically enhanced our understanding of many application domains including biological networks inferred from gene expression studies. However, there are clear computational challenges associated to the large datasets generated from these studies. The algorithmic solution of some NP-hard combinatorial optimization problems that naturally arise on the analysis of large networks is difficult without specialized computer facilities (i.e. supercomputers). In this work, we address the data clustering problem of large-scale biological networks with a polynomial-time algorithm that uses reasonable computing resources and is limited by the available memory. We have adapted and improved the MST*k*NN graph partitioning algorithm and redesigned it to take advantage of external memory (EM) algorithms. We evaluate the scalability and performance of our proposed algorithm on a well-known breast cancer microarray study and its associated dataset.

**Keywords:** Data clustering, external memory algorithms, graph algorithms, gene expression data analysis.

## 1 Introduction

The analysis of biological networks has become a major challenge due to the recent development of high-throughput techniques that are rapidly producing very large data sets. A number of algorithms, techniques and applications have been proposed to obtain useful information from various types of biological networks. Data clustering is perhaps the most common and widely used approach for the global network analysis. It helps to uncover important functional modules in the network. Numerous clustering algorithms for analyzing biological networks have been developed. These traditional algorithms/tools work well on moderate size networks and can produce

* Corresponding author.

Y. Xiang et al. (Eds.): ICA3PP 2011 Workshops, Part II, LNCS 7017, pp. 375–386, 2011.

informative results. Interestingly, the size and number of the biological networks are continuously growing due to extensive data integration from newly discovered biological processes and by novel microarray techniques that also consider ncRNAs. To handle the large-scale networks, existing algorithms are required to scale well and need to be re-implemented using cutting-edge software and hardware technologies.

In this work, we have enhanced and re-implemented a graph-based clustering algorithm known as MST*k*NN, proposed by Inostroza-Ponta et al. [1], to tackle the task of clustering large-scale biological networks. Given a weighted undirected graph ($G$) (or in its special case, given a non-negative square matrix of *distances* among a set of objects, i.e. a complete weighted graph) the MST*k*NN algorithm starts by building a proximity graph. It is defined as having the same set of nodes as the original graph, but has as the set of edges, the intersection of the edges of the minimum spanning tree (MST($G$)) and a the $k$-nearest neighbor graph ($k$NN($G$)). Gonzáles et al. [2] also used this proximity graph, with $k=\lfloor \ln(n) \rfloor$ where $n$ is the number of nodes. In the MST*k*NN algorithm, the value of $k$ is determined automatically and a recursive procedure partitions the graph until a stopping criteria stops this recursive partition of a cluster [3]. MST*k*NN does not require any fixed parameter (e.g., predetermined number of clusters) and it performs better than some other known classical clustering algorithms (e.g., *K*-Means and SOMs) in terms of homogeneity and separation [3] in spite of not using an explicitly defined objective function. In addition, it performs well even if the dataset has clusters of different mixed types (i.e. MST*k*NN is not biased to "prefer" convex clusters).

We propose here a different approach to allow the basic idea inherent to the MST*k*NN to be practically applicable on large datasets. In the worst-case situation, the input is a similarity/dissimilarity matrix at the start of the computation and, for a very large data set, this matrix may not fit in the computer's internal memory (in-memory) or even in the computer's external memory (EM). In order to overcome this problem, given $G$, we compute and store only a $q$NN graph (with $q=k+1$) of the similarity matrix and compute its MST (i.e. MST($q$NN)). Additionally, we annotate each edge of MST($q$NN)) with a non-negative integer value which is a function of the relative distance between the two nodes of that edge and their nearest neighbors. Finally, we recursively partition the MST($q$NN) using this set of annotations on the edges to produce the clusters. Unlike the MST*k*NN in [1], we compute the MST only once, instead of at each recursive step and we show that our clustering result is still the same to the previous proposed algorithm.

We have implemented our proposed algorithm by adapting the EM algorithmic approaches presented in [4-6], which give us an excellent performance improvement over the previous implementation. EM algorithms are very efficient when most of the data needs to be accessed from external memory. This approach improves the running time by reducing the number of I/Os between in-memory and the external memory. Further details on EM algorithms can be found in [7]. Additionally, we now have the benefits of employing parallel and distributed computing to calculate the similarity/distance matrix and computing the $q$NN graph that has made our data preprocessing reasonably fast on large data sets.

## 2 Related Work

Several graph-based clustering algorithms/tools have been developed in the past years and the advantages of them to analyse biological networks are clearly demonstrated in several publications [1, 8-9]. We can see graph-based clustering as a general domain of problems in which the task is often seen as an optimization problem (generally defined on a weighted graph). Given the graph, it is partitioned using certain pre-defined conditions. Each partition that represents a subgraph/ component of the graph is either further partitioned or presented as a cluster based on certain stopping criteria and guided by an objective function. In Table A.1, we present a brief list of the known graph-based clustering algorithm/tools for biological data sets along with the maximum test data set sizes in the relevant published literature. It is clear from the table that traditional graph-based clustering algorithms can serve as a primary/first tool for analyzing biological networks. However, new algorithms, designed with more advanced technologies, are necessary to deal with larger data sets. Surprisingly, EM algorithms, which are very convenient for handling massive data sets, have not yet been applied for clustering biological networks. We have found only few attempts in the published literature that exploit EM algorithms in bioinformatics, all of them seem to be related to sequence searching [10-11]. There exist several graph-based EM algorithms [12-13] that could be further investigated for their applicability on biological networks. In this work, we have adapted the EM computation of minimum spanning trees (EM MST) [4] and connected components (EM CC) [5-6]. These algorithms are capable of handling sparse graphs with up to billions of nodes.

## 3 Methods

### 3.1 The Original MST*k*NN Algorithm

The original MST*k*NN algorithm, presented in [3], takes an undirected complete graph ($G$) and computes two proximity graphs: a minimum spanning tree ($G_{MST}$) and a $k$-nearest neighbor graph ($G_{kNN}$), where the value of $k$ is determined by:

$$k = \mathbf{min}\{\lfloor \ln(n) \rfloor ; \mathbf{min}\ k/G_{kNN}\ \mathbf{is\ connected}\} \quad (1)$$

Subsequently, the algorithm inspects all edges in $G_{MST}$. If for a given edge $(x,y)$ neither $x$ is one of the $k$ nearest neighbors of $y$, nor $y$ is one of the $k$ nearest neighbors of $x$, the edge is eliminated from $G_{MST}$. This results in a new graph $G' = G_{MST} - \{(x,y)\}$. Since $G_{MST}$ is a tree, after the first edge is deleted, $G'$ becomes a *forest*. The algorithm continues applying the same procedure to each subtree in $G'$ (with a value of $k$ re-adjusted ($k=\lfloor \ln(n) \rfloor$), where $n$ is now the number of nodes in each subtree), until no further partition is possible. The final partition of the nodes of $G'$ induced by the forest is the result of the clustering algorithm.

### 3.2 MST*k*NN$^+$: The Modified MST*k*NN Algorithm

The original MST*k*NN algorithm requires $(n \times (n-1)/2)$ distance values (between all pairs of the $n$ elements) as the input. For a large data set, this could be too large to fit in the computer's in-memory and, for even larger values of $n$, it may not even fit in external memory. Even if we can store the distance matrix in the external memory, the computational speed will slow down dramatically because of the increased number of I/O operations. Therefore, we modified this step and instead of creating the complete graph from the distance matrix, we compute a $q$-nearest neighbor graph ($G_{qNN}$), where $q$ ($=\lfloor \ln(n) \rfloor + 1$). This procedure reduces the input graph size, but still creates a reasonable clustering structure of the data set. The value of the $q$ is determined from the *inclusion relationship* [2] of the $G_{MST}$ and the family of the nested sequence of graphs ($G_{kNN}$, where $k > \ln(n)$). Then, we compute the MST of the $G_{qNN}$ graph. We will call it $G_{MSTp}$. We first annotate each edge in $G_{MSTp}$ according to the following procedure. For each edge $(a,b)$ in $E(G_{MSTp})$ we assign an integer value $p$ to the edge as follows: let, $f(a,b)$ be the index of $b$ in the sorted list of nearest neighbors of $a$ in $G_{qNN}$. The value of $p$ is given by,

$$p = \textbf{min}\ \{f(a,b), f(b,a)\} \tag{2}$$

We define the maximum value of $p$ in the MST$p$ (or any of its components) as $p_{max}$ and then, we partition the $G_{MSTp}$ with the following criteria:

$C_1$. If $p > \lfloor \ln(n) \rfloor$; we remove the edge,

$C_2$. If $p_{max} < \lfloor \ln(n) \rfloor$; remove the edges with weight $p_{max} - 1$, and;

$C_3$. If $p_{max} = 1$ or $p_{max} = \lfloor \ln(n) \rfloor$; do not remove any edge, the result is a "cluster".

The final output of our algorithm is a set of partitions or clusters of the input data. The algorithm does not require any pre-determined value for $q$ but it is obviously possible to change the threshold from $\lfloor \ln(n) \rfloor$ to any other user-defined parameter. The algorithm can be understood as a recursive procedure (see below):

**Algorithm 1. PRE-MST*k*NN$^+$** ($D$: distance matrix)

1: Compute $G_{qNN}$.
2: Compute $G_{MSTp} = \text{MST}(G_{qNN})$.

**Algorithm 2. PRUNE-MST*k*NN$^+$** ($G_{MSTp}$)

1: $G' =$ **Partition** $G_{MSTp}$, using the criteria $C_1$, $C_2$ and $C_3$ described above.
2: $c =$ **connectedComponent**($G'$)
3: If $c > 1$ then
4: $G_{cluster} = \bigcup_{i=1}^{c}$ **PRUNE-MST*k*NN$^+$**(**components**($G'_i$))
5: End if
6: Return $G_{cluster}$

The function **connectedComponent**() gives the number of components in $G'$ and the function **components**() identifies and returns each of the components. Unlike the original algorithm in [1], we compute the MST only once (at the beginning), instead of at each recursive step. This change also gives a significant speed-up in terms of run-time performance over the previous algorithm. The following Lemma proves that this approach is sound (i.e., a partitioned subtree also represents an exact MST of the relevant component in the complete graph):

**Lemma 1.** Let $T$ be a minimum spanning tree for a weighted graph $G$. Then if we select an edge $e$ from $T$ and partition the graph according to the subgraphs induced by the subtrees induced by excluding $e$ from $T$, these subtrees are also minimum spanning trees for the subgraphs.

**Proof.** Let $T$ be a minimum spanning tree for a graph $G$. Let $T$ be partitioned into two subtrees $A$ and $B$ with vertex and edge sets $V(A)$, $V(B)$, $E(A)$ and $E(B)$ respectively. Furthermore, let $V(A) \cap V(B) = \phi$ and $V(A) \cup V(B) = V(G)$ and let $A$ and $B$ be connected by a single edge $e$ in $T$. Now consider the graph $G[V(A)]$ and let $T'$ be a minimum spanning tree for $G[V(A)]$. We define the weight function $w$ of a spanning tree to be the sum of the weights of the edges of the tree, and extend this in the natural way to any subtree. Then, $w(T) = w(A) + w(B) + w(e)$. Now, assume that $w(T') < w(A)$. Then, we could replace the subtree $A$ with $T'$, and join it to $B$ using $e$. As $V(A)$ and $V(B)$ are disjoint we cannot introduce any cycles, therefore $T'$ joined with $B$ via $e$ must be a tree, and further, a spanning tree for $G$. However, this new tree must have weight less than $w(T)$, contradicting the minimality of $T$. Therefore, $T'$ cannot exist.

The main advantage of this algorithm over the all other MST-based graph clustering algorithms (for example [8-9]) is that it prunes the MST edges using the local connectivity, instead of using the exact distance between the two nodes in an edge (e.g., deleting the longest edge). Our algorithm can produce better results in terms of local connectivity (i.e., homogeneity) which is a desirable characteristic in clustering biological networks.

### 3.3 Implementation

**The Test Environment.** The computational tests were performed on a 16 node cluster computer (with Intel Xeon 5550 processors, 2.67 GHz speed, 8 cores) and the programs were written in C++ with the support of STL, STXXL[1] and BOOST[2] library and compiled using the g++ 4.4.4 compiler on a Linux OS, kernel ver. 2.6.9.

**Parallel /Distributed NN graph computation.** To compute the distance matrix we use a message-passing interface (MPI) to distribute the data set (row-wise) into $P$ parallel processors and then initiate the parallel computation of the distance metric, in each of them using Open MP (Multi-Processing). The method for efficiently distributing the computation of upper/lower triangle of the symmetric similarity matrix will be discussed later.

[1] http://stxxl.sourceforge.net/
[2] http://www.boost.org/

**The EM MST and CC computation.** We compute the MST using the EM MST algorithm in [4]. The I/O complexity of this algorithm is O(sort($m$)·log($n/M$)), where $n$ is the number of nodes of the original graph, $m$ is number of edges and $M$ number of nodes that fit into computer's internal memory, respectively, and the sort($m$) is the time required for sorting the $m$ edges. After partitioning the MST, we identify the connected components using the EM connected component algorithm in [5-6]. The I/O complexity of this algorithm is O($m$·log(log($n$))). Unlike other clustering tools, we store the connected components/clusters in external memory and only keep the *list* of the components in computer's in-memory. This eliminates the excessive use of the in-memory even when there are a large number of components or clusters. Additionally, we tuned the implementations of the adapted algorithms [4-6] for better performance with denser graphs.

# 4 Results

## 4.1 Data Description

We used two different data sets to demonstrate the performance of our proposed EM algorithm MST$k$NN$^+$. The first data set is used to illustrate the algorithm and contains a distance matrix between 10 Australian cities. The second data set is a breast cancer gene-expression data set from a study by van de Vijver et al. [14]. This microarray dataset contains the expression of 24,158 probe sets in 295 primary breast cancer patients. The data set also contains the clinical metastasis information (in terms of years to relapse) for all the patients. We also create a third larger dataset from van de Vijver et al. [14] as follows. First, we filter the probes sets using Fayyad and Irani's algorithm [15]. This step is supervised and aims at finding differentially expressed probe sets in the samples labeled *"metastasis"* versus the ones labeled "*non-metastasis"*. This does not mean that these patients had no relapse. Instead, we indicate with "*non-metastasis*" that the patients had no relapse within five years after the initial diagnosis, but indeed there is a presence of a metastasis during the duration of the study, up to 14 years in one case. Next, we use a feature selection algorithm to refine the selection of probe sets using the *(alpha-beta)-k-Feature set* methodology [16]. After we selected features based on this method, we obtain a set of 876 probes sets. Finally, we produce a new large data set by subtracting the expression values of each possible pair of probes. These unique probe pairs are termed as *metafeatures* as in Rocha de Paula et al. [17]. Subsequently, we have an artificial data set with 384,126 elements, including all the filtered probes and all metafeatures.

## 4.2 Application on the City Distance Data Set

Our first application is on a distance matrix that we have created by taking distances among 10 Australian cities. The data set is given is Table A.2. We first create a $q$NN graph from the data set (See Table A.3) for $q$ = 3 and an MST$p$, where we annotate each edge with an integer value ($p$) as described in equation (2). For example (See Figure 1(a) and Table A.3), *Adelaide* is the third nearest neighbor of *Melbourne*


**Fig. 1.** (a) The MST$p$ created from 10 Australian cities (actual locations of the cities in the map are schematic). The edge between "Albany" and "Adelaide" is the candidate for deletion as the neighborhood value $p > \lfloor \ln(10) \rfloor = 2$ (b) In the first iteration of MST$k$NN$^+$ the edge between "Katherine" and "Adelaide" is the next candidate to delete as $p > \lfloor \ln(7) \rfloor = 1$, where the number of elements in that component is 7 (c) Final clustering result.

and *Melbourne* is the first nearest neighbor of *Adelaide*. Therefore, we give a weight of 1 (the minimum) to the edge that connects *Adelaide* and *Melbourne*. Finally, we prune the MST edges using the criteria $C_1$, $C_2$ and $C_3$ on each of the components. The result of our algorithm is presented in Figure 1(c).

### 4.3 Application on the Breast Cancer Data Set

Our second application is on a dataset on breast cancer. It contains the gene expression values measured on 24,158 probe sets for 295 primary breast cancer patients [14]. We first compute a similarity matrix using Pearson's correlation and create a $q$NN graph that contains 24,158 vertices and 265,738 edges. Next, we create the MST$p$. Finally, we apply our proposed algorithm to partition the MST$p$ to obtain the clusters (see Figure 2).

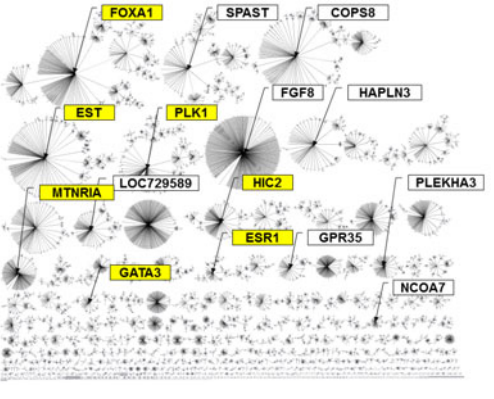


**Fig. 2.** Visualization of the clusters from the breast cancer data set in [12]. Some genes of interest are highlighted.

Additionally, we used iHOP[3] to find the importance of the genes that are in the central regulatory positions of some of the clusters (see Figure 3). Our results show that many of the genes that are in the central position seem to have been already discussed in breast cancer and its progression course (see Table 1). Additionally, the genes with less number of published papers can also be further investigated based on their conspicuous position in the clustering and adjacency relation with the genes that have already been implicated in breast cancer.

**Table 1.** The number of published literature associated with some of the observed most central genes and results using iHOP and Pubmed for searching the name of the gene and its aliases together with the words "breast" and "cancer" (ordered by gene symbol, highly referenced genes are in bold face)

| Gene Symbol | Gene Name | Breast | Cancer |
|---|---|---|---|
| COPS8 | COP9 constitutive photomorphogen | 10 | 57 |
| CPNE1 | copine I | 1 | 3 |
| **ESR1** | estrogen receptor 1 | 17,352 | 28,250 |
| **EST** | mitogen-activated protein kinase 8 | 165 | 879 |
| **FGF8** | fibroblast growth factor 8 | 27 | 156 |
| **FOXA1** | forkhead box A1 | 60 | 120 |
| **GATA3** | GATA binding protein 3 | 219 | 1399 |
| GPR35 | G protein-coupled receptor | 0 | 2 |
| HAPLN3 | hyaluronan and proteoglycan link 3 | 1 | 1 |
| HIC2 | hypermethylated in cancer | 13 | 122 |
| LOC729589 | hypothetical LOC729589 | 0 | 0 |
| **MTNR1A** | melatonin receptor 1A | 194 | 1193 |
| NCOA7 | nuclear receptor coactivator 7 | 1 | 3 |
| PLEKHA3 | pleckstrin homology domain 3 | 0 | 2 |
| **PLK1** | polo-like kinase 1 | 49 | 458 |
| SPAST | spastic paraplegia 4 | 0 | 3 |

### 4.4 Application on an Expanded Breast Cancer Data Set with 384,126 Vertices and 4,993,638 Edges

Finally, we apply our proposed algorithm (MST$k$NN$^+$) on a large-scale "artificial" data set that is an expanded version of the breast cancer data set [14]. This data set has 384,126 elements (the values of 383,250 metafeatures together with the values of 876 probe sets obtained by filtering the original data set). Additionally, we also include the clinical metastasis information as a "phenotypical dummy probe set". As previously described, we first create the $q$NN graph containing 384,126 vertices and 4,993,638 edges. Next, we apply MST$k$NN$^+$ to find the clusters. Due to the limitation of the existing visualization tools, it is impossible to provide a picture of the complete clustering. Instead, we present the group of metafeatures that closely cluster with the "phenotypical dummy probe set" (years for relapse), zooming in a part of that naturally strikes many as very interesting (see Figure 3). We find one metafeature (*BCAR1-SLC40A1*) that has better correlation with the metastasis information than either the individual probe sets alone (e.g., genes *BCAR1* or *SLC40A1*, see Figure 4).

[3] http://www.ihop-net.org/UniPub/iHOP/

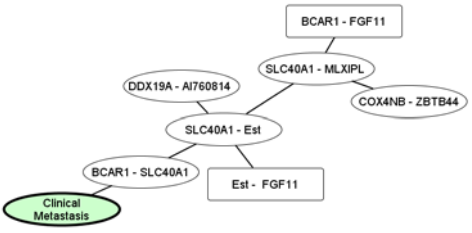


**Fig. 3.** The visualization (partial) of the cluster that contains the clinical metastasis information as a phenotypical gene. The rectangular shaped nodes indicate that the genes in these metafeatures share a common biological pathway (identified using GATHER[4]).

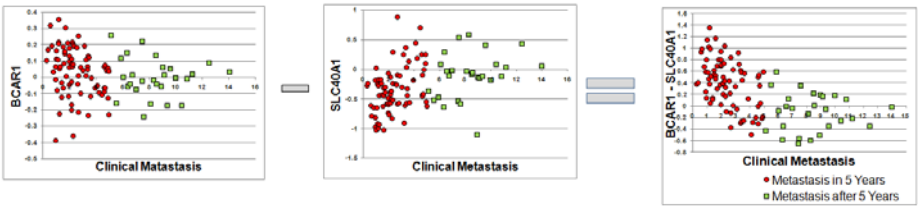


**Fig. 4.** The metafeature (*BCAR1-SLC40A1*) shows better correlation with the clinical metastasis values of each patient with respect to the feature (i.e., the *BCAR1*, Breast Cancer Anti-estrogen Resistance 1, or *SLC40A1*, Ferroportin-1) alone

It is also interesting to note the presence of *SLC40A1* in three of the metafeatures co-expressed with the time to relapse values (clinical metastasis "dummy probe set"). Jiang et al. suggested that "breast cancer cells up-regulate the expression of iron importer genes and down-regulate the expression of iron exporter *SLC40A1* to satisfy their increased demand for iron" [18]. This data indicates that, for those tumors that may relapse (and for which a different genetic signature may need to be found), the joint expression of *BCAR1* and Ferroportin may be associated to time to relapse. Similarly, other identified metafeatures could also be further investigated.

### 4.5 Performance Comparisons

We have compared the solutions of our clustering approach against *K*-Means, SOM, CLICK and the original MST*k*NN [1], using the homogeneity and separation indexes that give us an idea of how similar the elements in a cluster and dissimilar among the clusters, respectively (See Table 5). We used the implementation of the *K*-Means, SOM and CLICK available in the Expander tool[5] and the implementation of the MST*k*NN in [1] is obtained from *http://cibm.newcastle.edu.au*. The averages of homogeneity ($H_{avg}$) and separation ($S_{avg}$) were computed as in [19] and the *Pearson's correlation* is used as the metric for computing the similarity matrix.

[4] http://gather.genome.duke.edu/
[5] http://www.cs.tau.ac.il/~rshamir/expander/

**Table 2.** Performance comparisons with *K*-Means, SOM, CLICK and original the MST*k*NN approach in terms of homogeneity and separation

| Data | Algorithms | Param. | $H_{avg}$ | $S_{avg}$ | #Clust. | Time (min) | Mem. (MB) |
|---|---|---|---|---|---|---|---|
| Breast Cancer Filtered *n*=876 | *K*-Means | *K*=41 | 0.521 | -0.186 | 41 | ~ 1 | ~ 250 |
| | SOM | 3×3 | 0.501 | -0.015 | 9 | ~ 0.2 | ~ 200 |
| | CLICK | - | 0.538 | -0.281 | 8 | ~ 0.5 | ~ 250 |
| | MST*k*NN | - | 0.287 | 0.386 | 41 | ~ 0.5 | ~ 250 |
| | **MST*k*NN$^+$** | **-** | **0.288** | **0.389** | **45** | **~ 0.3** | **~ 156** |
| Complete *n*=24,158 | *K*-Means, SOM, CLICK | - | - | - | - | - | - |
| | MST*k*NN | - | 0.429 | 0.390 | 732 | ~ 12 | ~ 8,100 |
| | **MST*k*NN$^+$** | **-** | **0.430** | **0.398** | **745** | **~ 5^** | **~ 650$^\dagger$** |
| Expanded *n*=384,126 | *K*-Means, SOM, CLICK, MST*k*NN | - | - | - | - | - | - |
| | **MST*k*NN$^+$(ours)** | - | **0.630** | **0.410** | **2,587** | **~ 15^** | **~ 1,500$^\dagger$** |

^ Does not include the time for computing the similarity matrix.

† Internal memory consumption can be pre-defined with EM environment parameters.

From Table 2, we can clearly see that MST*k*NN succeeds in producing small, precise clusters from the filtered expression data (*n*=876). Even for the same number of clusters it gives better performance (i.e., higher homogeneity and lower separation values) than *K*-Means (even when we intentionally set *K*=41 in *K*-Means), SOM and CLICK. Proposed MST*k*NN$^+$, showed better performance in terms of homogeneity, time and memory usage, but the separation value was slightly increased. For the complete breast cancer data set (*n*=24,158), only the MST*k*NN and our proposed algorithm were able to cluster the data set with high and low in-memory usage, respectively. The other algorithms were incomputable and were running indefinitely on the test machine. Finally, for the expanded breast cancer data set (*n*=384,126), only our proposed algorithm's implementation MST*k*NN$^+$ could successfully cluster the whole data set in 15 minutes and using reasonable amount of in-memory.

## 5 Conclusion and Future Work

In this paper, we have proposed a significant improvement to the existing MST*k*NN based clustering approach. Our implementation is faster (due to parallel/distributed pre-processing and algorithmic enhancement) and more memory efficient and scalable (due to the EM implementation) than the one in [1]. The clusters identified by our approach are meaningful, precise and comparable with other state-of-the-art algorithms. Our future work includes the design and implementation of a nearest neighbor-based MST algorithm so that we can eliminate the prohibitive computation of the similarity matrix when the data set is terribly large. Finding the nearest neighborhood of a point in space is being widely researched and one way to do so is to produce a *kd*-tree. Other approaches, such as a GPU based similarity matrix computation can be an aid to accelerate the clustering process.

## References


1. Inostroza-Ponta, M.: An Integrated and Scalable Approach Based on Combinatorial Optimization Techniques for the Analysis of Microarray Data, PhD thesis, The University of Newcastle, Australia (2008)
2. Gonzalez-Barrios, J.M., Quiroz, A.J.: A clustering procedure based on the comparison between the k nearest neighbors graph and the minimal spanning tree. Statistics and Probability Letters 62(3), 23–34 (2003)
3. Inostroza-Ponta, M., Mendes, A., Berretta, R., Moscato, P.: An integrated QAP-based approach to visualize patterns of gene expression similarity. In: Randall, M., Abbass, H.A., Wiles, J. (eds.) ACAL 2007. LNCS (LNAI), vol. 4828, pp. 156–167. Springer, Heidelberg (2007)
4. Dementiev, R., Sanders, P., Schultes, D., Sibeyn, J.: Engineering an external memory minimum spanning tree algorithm. In: 3rd IFIP Intl. Conf. on Theoretical Computer Science, pp. 195–208 (2004)
5. Sibeyn, J.: External Connected Components. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111, pp. 468–479. Springer, Heidelberg (2004)
6. Schultes, D.: External memory spanning forests and connected components, Technical report (2004), `http://algo2.iti.kit.edu/dementiev/files/cc.pdf`
7. Vitter, J.S.: External memory algorithms and data structures: Dealing with massive data. ACM Computing Surveys 33 (2001)
8. Xu, Y., Olman, V., Xu, D.: Clustering Gene Expression Data Using a Graph-Theoretic Approach: An Application of Minimum Spanning Tree. Bioinformatics 18(4), 526–535 (2002)
9. Grygorash, O., Zhou, Y., Jorgensen, Z.: Minimum Spanning Tree Based Clustering Algorithms. In: Proc. of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006), pp. 73–81. IEEE Computer Society, Washington, DC, USA (2006)
10. Doowang, J.: An external memory approach to computing the maximal repeats across classes of dna sequences. Asian Journal of Health and Information Sciences 1(3), 276–295 (2006)
11. Choi, J.H., Cho, H.G.: Analysis of common *k*-mers for whole genome sequences using SSB-tree. Japanese Society for Bioinformatics 13, 30–41 (2002)
12. Chiang, Y., Goodrich, M.T., Grove, E.F., Tamassia, R., Vengroff, D.E., et al.: External-memory graph algorithms, In. In: SODA 1995: Proceedings of the Sixth Annual ACM-SIAM, pp. 139–149. Society for IAM, Philadelphia (1995)
13. Abello, J., Buchsbaum, A.L., Westbrook, J.R.: A functional approach to external graph algorithms. Algorithmica, 332–343 (1998)
14. van de Vijver, M.J., He, Y.D., van't Veer, L.J., Dai, H., et al.: A gene-expression signature as a predictor of survival in breast cancer. N. Engl. J. Med. 347(25) (2002)
15. Fayyad, U.M., Irarni, K.B.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In: IJCAI, pp. 1022–1029 (1993)
16. Cotta, C., Sloper, C., Moscato, P.: Evolutionary Search of Thresholds for Robust Feature Set Selection: Application to the Analysis of Microarray Data. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 21–30. Springer, Heidelberg (2004)
17. Rocha de Paula, M., Ravetti, M.G., Rosso, O.A., Berretta, R., Moscato, P.: Differences in abundances of cell-signalling proteins in blood reveal novel biomarkers for early detection of clinical Alzheimer's disease. PLoS ONE 6(e17481) (2011)
18. Jiang, X.P., Elliot, R.L., Head, J.F.: Manipulation of iron transporter genes results in the suppression of human and mouse mammary adenocarcinomas. Anticancer Res. 30(3), 759–765 (2010)
19. Shamir, R., Sharan, R.: CLICK: A Clustering Algorithm with Applications to Gene Expression Analysis. In: Proc. of ISMB, pp. 307–316 (2000)

# Appendix

**Table A.1.** A list of known graph-based clustering algorithms/tools for biological networks[6]

| Name | Approaches | Language | Max. test data ($n$) |
|---|---|---|---|
| cMonkey | Bi-clustering | R | 2,993 |
| GTOM | Topological overlap | R | 4,000 |
| SAMBA | Neighborhood search | C/C++ | 4,177 |
| CAST | Affinity search | Matlab | 6,000 |
| NNN | Mutual NN search | Java | 6162 |
| EXCAVATOR | MST | C, Java | 6,178 |
| HCS | Minimum cut | Matlab,LEDA | 7,800 |
| MST*k*NN | Intersect MST- *k*NN | Java | 14,772 |
| CLICK | Mincut | C/C++ | 29,600 |
| Ncut-KL | Mincut, | - | 40,703 |
| TribeMCL | MCL | C/C++ | 80,000 |
| MPI-MCL | MCL, dist. comp. | Fortran, MPI | 125,008 |

**Table A.2.** A distance matrix in KMs for 10 Australian cities[7]

| | Canb. | Syd. | Melb | Adel. | Perth | Darwin | Kath. | Hobart | Albany | Bunb. |
|---|---|---|---|---|---|---|---|---|---|---|
| Canberra | 0 | 240 | 473 | 967 | 3102 | 3141 | 2870 | 865 | 2838 | 3080 |
| Sydney | 240 | 0 | 713 | 1163 | 3297 | 3153 | 2882 | 1060 | 3046 | 3282 |
| Melb. | 473 | 713 | 0 | 654 | 2727 | 3151 | 2885 | 601 | 2436 | 2690 |
| Adelaide | 967 | 1163 | 654 | 0 | 2136 | 2620 | 2364 | 1165 | 1885 | 2118 |
| Perth | 3102 | 3297 | 2727 | 2136 | 0 | 2654 | 2562 | 3017 | 392 | 156 |
| Darwin | 3141 | 3153 | 3151 | 2620 | 2654 | 0 | 271 | 3743 | 2828 | 2788 |
| Katherine | 1962 | 2030 | 1892 | 1330 | 1995 | 1291 | 0 | 2470 | 1993 | 2688 |
| Hobart | 865 | 1060 | 601 | 1165 | 3017 | 3743 | 3478 | 0 | 2678 | 2951 |
| Albany | 2838 | 3046 | 2436 | 1885 | 392 | 2828 | 2702 | 2678 | 0 | 279 |
| Bunbury | 3080 | 3282 | 2690 | 2118 | 156 | 2788 | 2688 | 2951 | 279 | 0 |

**Table A.3.** Three nearest neighborhood ($q$=3) for 10 Australian cities

| City/ $q$ = | 1 | 2 | 3 |
|---|---|---|---|
| Canberra | Sydney (240) | Melbourne (473) | Hobart (865) |
| Sydney | Canberra (240) | Melbourne (713) | Hobart (1060) |
| **Melbourne** | Canberra (473) | Hobart (601) | **Adelaide (654)** |
| **Adelaide** | **Melbourne (654)** | Canberra (967) | Sydney (1163) |
| Perth | Bunbury (156) | Albany (392) | Adelaide (2136) |
| Darwin | Katherine (271) | Adelaide (2620) | Perth (2654) |
| Katherine | Darwin (1291) | Adelaide (1330) | Melbourne (1892) |
| Hobart | Melbourne (601) | Canberra (865) | Sydney (1060) |
| Albany | Bunbury (279) | Perth (392) | Adelaide (1885) |
| Bunbury | Perth (156) | Albany (279) | Adelaide (2118) |

[6] Details about the methods and test environments can be found in the relevant publications.
[7] Computed using the distance tool at *http://www.geobytes.com/citydistancetool.htm*