Exploring Ant-Based Algorithms for Gene Expression Data Analysis

Yulan He[a,*], Siu Cheung Hui[b]

[a] *School of Engineering, Computing and Mathematics, University of Exeter North Park Road, Exeter EX4 4QF, UK*
[b] *School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798*

Summary

Objective: Recently, much research has been proposed using nature inspired algorithms to perform complex machine learning tasks. Ant colony optimization (ACO) is one such algorithm based on swarm intelligence and is derived from a model inspired by the collective foraging behavior of ants. Taking advantage of the ACO in traits such as self-organization and robustness, this paper investigates ant-based algorithms for gene expression data clustering and associative classification.

Methods and Material: An ant-based clustering (Ant-C) and an ant-based association rule mining (Ant-ARM) algorithms are proposed for gene expression data analysis. The proposed algorithms make use of the natural behavior of ants such as cooperation and adaptation to allow for a flexible robust search for a good candidate solution.

*Corresponding author. Tel.: +44 1392 263591; Fax: +44 1392 264067.
Email addresses: y.l.he.01@cantab.net (Yulan He), asschui@ntu.edu.sg (Siu Cheung Hui)

Results: Ant-C has been tested on the three datasets selected from the Stanford Genomic Resource Database and achieved relatively high accuracy compared to other classical clustering methods. Ant-ARM has been tested on the acute lymphoblastic leukemia (ALL) / acute myeloid leukemia (AML) dataset and generated about 30 classification rules with high accuracy.

Conclusions: Ant-C can generate optimal number of clusters without incorporating any other algorithms such as K-means or agglomerative hierarchical clustering. For associative classification, while a few of the well-known algorithms such as Apriori, FP-growth and Magnum Opus are unable to mine any association rules from the ALL/AML dataset within a reasonable period of time, Ant-ARM is able to extract associative classification rules.
Key words: Gene expression data analysis, Ant colony optimization, Clustering, Associative classification, Swarm intelligence.

1. Introduction

The break through in the development of DNA microarray technology examination of gene expression has created a new era for further exploration of living systems, source of disease and drug development. The study of thousands of genes to analyze their relations to cancer biology is one of the advantages harnessed from DNA microarray technologies. However, as the years go by, the explosion of genomic data in tens of thousands through microarray experiments has brought about problems in the development of efficient methods to pick out the information from such microarrays.

These gene expression data in microarray are presented in $M \times N$ matrix where M is the number of microarray experiments and N being the number of genes [1]. The number of experiments M can range from dozens to thousands. On the other hand, the number of genes N can range from hundred to tens of thousands. In some context, M can be referred to as number of transactions or itemsets where each gene represents an item. To add to the complexity of representation, each gene is measured in terms of absolute values. However, biologists are more interested in how gene expression changes under different environments in each respective experiment. Thus, these absolute values are discretized according to some predetermined thresholds and grouped under three different levels, namely unchanged, upregulated and downregulated.

With gene expression data, certain analysis needs to be performed on them to retrieve useful biological information. One of the techniques named cluster analysis has discovered useful biological information by detecting genes that have identical expression profile [2]. However, there are disadvantages in employing this technique. Firstly, the relationships of the genes revealed from clustering are only a fraction of the many valuable relationships that are present among various transcripts. Secondly, the user needs to have a good biological understanding of the pathway in question. Thus, for many of the pathways and rules that are not known to the user, clustering may not be a wise choice.

Such shortfalls provide the motivation for the development of association rule mining (ARM) [3]. ARM finds interesting associations and/or correlation relationships among a large set of data items. It is widely used for market

basket data to observe consumer spending patterns in retail markets. Its application is used in the analysis of expression data as well. Studies have revealed that association rules show the link between genes and environments/categories. Cancer diagnosis and diseases can also be predicted by the studies of the genes generated from association rules [4].

This paper explores ant-based algorithms for gene expression data analysis. Novel methods based on the ant colony optimization (ACO), a nature inspired algorithm emerging from the collective behavior of social ant colonies, are proposed for clustering (Ant-C) and associative classification (Ant-ARM). The rest of the paper is organized as follows. The related work on clustering and ARM on gene expression data is presented in Section 2. Section 3 briefly describes the ACO algorithm. The proposed Ant-C and Ant-ARM approaches are discussed in Section 4 and Section 5 respectively. Experimental results are presented in Section 6. Finally, Section 7 concludes the paper.

2. Related Work

Recently, there have been growing interests to apply the ACO algorithm on gene expression data analysis. Ressom *et al.* [5] proposed an algorithm that combines a recurrent neural network (RNN) and two swarm intelligence methods to infer a gene regulatory network (GRN) from time course gene expression data. ACO was used to identify the optimal architecture of an RNN, while the weights of the RNN were optimized using particle swarm optimization. Robbins *et al.* [6] applied ACO for feature selection in high dimensional gene expression

data for disease classification. The ACO was able to identify small subsets of highly predictive and biologically relevant genes without the need for extensive preselection of features. In this paper, we explore the ant-based algorithms for gene expression data clustering and associative classification.

2.1. Clustering

Clustering is a fundamental technique in exploratory data analysis and pattern discovery, aiming at extracting underlying cluster structures. Cluster analysis is concerned with multivariate techniques that can be used to create groups amongst the observations, where there is no *a priori* information regarding the underlying group structure. Clustering of the genes on the basis of the tissues can be used to detect a group of genes whose expression level changes follow a same pattern. Dozens of clustering algorithm exist in the literature and a number of *ad hoc* clustering procedures have been applied to microarray data. Available methods can be categorized broadly as being hierarchical such as agglomerative hierarchical clustering (AHC) [7, 8] or non-hierarchical such as K-means clustering [9] and clustering through self-organizing maps (SOMs) [10]. A major limitation of hierarchical methods is their inability to determine the number of the clusters. The limitation of K-means methods mainly lies in its inability to handle clusters with different sizes or densities and clusters with non-globular shapes [11]. SOMs are in fact conceptually different from clustering as they try to map high-dimensional input data onto a regular low-dimensional grid while clustering is to partition the input data into groups. Thus SOMs often fail to deliver satisfactory results especially when clusters exhibit arbitrary

shapes [12].

Early approaches in applying ACO to clustering [13, 14, 15] are to first partition the search area into low-dimensional regular grid (typically 2D grid). A population of ant-like agents then move around this 2D grid and carry or drop objects based on certain probabilities so as to categorize the objects. However, this may result in too many clusters as there might be objects left alone in the 2D grid and objects are still carried by the ants when the algorithm stops. Therefore, some other algorithms such as K-means are normally combined with ACO to minimize categorization errors [16, 17, 18]. More recently, variants of ant-based clustering have been proposed, such as using inhomogeneous population of ants which allow to skip several grid cells in one step [19], representing ants as data objects and allowing them to enter either the active state or the sleeping state on a 2D grid [20].

It has been formally proved in [21] that ACO clustering algorithms based on the grid structure are prone to produce bad topographic mappings, either too many or too small, and topographically distorted clusters. This paper proposes a novel Ant-C algorithm without relying on a 2D grid structure. In addition, it can generate optimal number of clusters without incorporating any other algorithms such as K-means or AHC.

2.2. Association Rule Mining

Apriori is the classic ARM algorithm [3] and is commonly used as a benchmark for improvements in ARM. Earlier work on applying Apriori in gene expression data can be found in [22]. The main disadvantage of Apriori is its high

memory space overhead in generating the frequent itemsets and association rules for large datasets which would exhaust the memory space limit that eventually results in the termination of generation of rules in many of the experiments [23]. It has been mentioned that Apriori-based algorithms only work if the largest frequent itemset has a size lower or equal to 15 when studied on human SAGE data [24].

In many of the gene expression datasets, the number of genes (columns) can range up to 10,000-100,000 while there are only 100-1000 experiments (rows). Many of the column-wise mining algorithms such as Apriori would have problems processing the large number of columns in order to identify the frequent itemsets as the search space consists of 2^i possible candidates where i is the number of columns. FARMER [25] uses a depth-first row-wise algorithm as opposed to a breath-first algorithm used by Apriori. It proves to be a lot faster than other algorithms that works on column enumeration space. However, no evaluations for the accuracy of the algorithm have been performed. Furthermore, most of the datasets used are artificial datasets whose association rules may not be of significance as compared to biomedical gene expression datasets.

3. Ant Colony Optimization

Nature inspired algorithms are problem solving techniques that attempt to simulate the occurrence of natural processes. Some of the natural processes that such algorithms are based on include the evolution of species [26, 27], organization of insect colonies [28, 29] and the working of immune systems

[30, 31]. Nature inspired algorithms are often characterized by techniques that manage a community of individuals and attempt to generate smarter individuals as the community evolves. Through means of selection, reproduction, feedback and heritability, each individual is able to offer a more productive solution. The ability of such stochastic population-based methods to explore large search space makes them highly suitable for overcoming complex problems such as clustering and optimization.

The ACO algorithm [28, 29, 32, 33] belongs to the natural class of problem solving techniques which is initially inspired by the efficiency of real ants as they find their fastest path back to their nest when sourcing for food. An ant searches for the fastest path based on the presence of pheromone deposited along the trail by either itself or other ants. An open loop feedback exists in this process as the chances of an ant taking a path increases with the amount of pheromone built up by other ants. This natural phenomenon has been applied to model the traveling salesman problem (TSP) [32].

Given a set of cities and the distances between them, the TSP is the problem of finding the shortest possible path which visits every city exactly once. More formally, it can be represented by a complete weighted graph $G = (N, E)$ where N is the set of nodes representing the cities and E is the set of edges. Each edge is assigned a value d_{ij} which is the distance between city i and j. When applying the ACO algorithm to the TSP, a pheromone strength $\tau_{ij}(t)$ is associated to each edge (i, j), where $\tau_{ij}(t)$ is a numerical value which is modified during the execution of the algorithm and t is the iteration counter. The skeleton of the

ACO algorithm applied to the TSP is:

```
procedure ACO algorithm for TSPs
    set parameters, initialize pheromone trails
    while (termination condition not met) do
        Tour construction
        Pheromone update
    end
end ACO algorithm for TSPs
```

At first, each of the m ants is placed on a randomly chosen city. At each *Tour construction* step, ant k currently at city i, chooses to move to city j at the tth iteration based on the probability $P_{ij}^k(t)$ which is biased by the pheromone trail strength $\tau_{ij}(t)$ on the edge between city i and city j and a locally available heuristic information η_{ij}. Each ant is associated with a *tabu list* in which the current partial tour is stored, i.e. $tabu_k(s)$ stores a set of cities visited by ant k so far at time s. After all the ants have constructed their tours, *Pheromone update* is performed by allowing each ant[1] to add pheromone on the edges it has visited. At the end of the iteration, the tabu list is emptied and each ant can choose an alternative path for the next cycle.

The design of an ACO algorithm involves the following specifications [34]:

- an appropriate representation of the problem, usually done by means of a "construction graph", which allows the ants to incrementally con-

[1]Note that in some ACO variants such as the Max-Min Ant System, only the best ant is allowed to update pheromone.

struct/modify solutions through the use of a probabilistic transition rule, based on the amount of the pheromone in the trail and on a local problem-dependent heuristic;

- a problem-dependent heuristic function (η) that measures the quality of items that can be added to the current partial solution;
- a rule for pheromone updating, which specifies how to modify the pheromone trail (τ);
- a probabilistic transition rule based on the contents of the pheromone trail (τ) and the value of the heuristic function (η) that is used to iteratively construct a solution.

4. Ant-Based Clustering (Ant-C)

We have proposed an ant-based clustering algorithm for document clustering based on the TSP scenario [36]. The advantages of our ant-based clustering approach are: 1) It does not rely on a 2D grid structure. 2) It can generate optimal number of clusters without incorporating any other algorithms such as K-means or AHC. 3) When compared with other document clustering algorithms such as K-means, AHC, and the artificial immune network based method, it shows improved performance when tested on the subsets of 20 Newsgroup data[2]. Here, we investigate the Ant-C algorithm for gene expression data analysis [37].

[2] `http://people.csail.mit.edu/jrennie/20Newsgroups/` (accessed: 17 March 2008)

Table 1: ACO design issues addressed for Ant-C and Ant-ARM.

Design Criteria	*Ant-C*	*Ant-ARM*
Problem representation	Each node represents a gene in the graph. N genes are connected by $\frac{1}{2}N \times (N-1)$ edges.	Each node on the graph corresponds to a 1-itemset rule whose support exceeds minSupport threshold
A problem-dependent heuristic function (η)	η_{ij} is defined as a distance measure $\text{dist}(\vec{g_i}, \vec{g_j})$ between two genes g_i and g_j.	η_j is defined as 1 if the support of the newly constructed rule $\hat{r}$ by including the itemset I_j is greater than minSupport and 0 otherwise.
A rule for updating the pheromone trail (τ)	$\tau_{ij}(t)$ represents the amount of pheromone associated with the gene pair gene_{ij} at iteration t. The initial amount of pheromone deposited at each path position is $\tau_{ij}(0) = 1/N$ where N is the total number of genes in the collection $\mathcal{D}$. At every generation of the algorithm, τ_{ij} is updated by $\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau$ where $\rho \in (0,1]$ determines the evaporation rate and the update of pheromone trail; $\Delta\tau$ is defined as the integrated similarity of a gene with other genes within a cluster which is measured by: $\Delta\tau = \begin{cases} \sum_{j=1}^{N_i}[1 - \frac{\text{dist}(\vec{c_i},\vec{g_j})}{\gamma}] & g_j \in c_i \\ 0 & otherwise \end{cases}$ where $\vec{c_i}$ is the centroid vector of the ith cluster, $\vec{g_j}$ is the jth gene vector which belongs to cluster i, N_i stands for the number of genes which belongs to the ith cluster. The parameter γ is defined as swarm similarity coefficient and it affects the number of clusters as well as the convergence of the algorithm.	$\tau_j(t)$ represents the amount of pheromone associated with the itemset $iset_j$ at iteration t. The initial amount of pheromone deposited at each path position is defined by $\tau_j(0) = 1/\sum_{i=1}^{k} b_i$ where k is the total number of attributes and b_i is the number of possible values that can be taken on by attribute A_i. At every generation of the algorithm, τ_j is updated by $\tau_j(t+1) = (1-\rho)\tau_j(t) + \Delta\tau$ where $\rho \in (0,1]$ determines the evaporation rate and the update of pheromone trail $\Delta\tau$ is defined as the *Foil Gain* [35] that measures the information gained from adding the itemset I_j to the current rule. Suppose there are $\lvert P\rvert$ positive examples and $\lvert N\rvert$ negative examples satisfying the current rule r's body. After the itemset I_j is added to r, there are $\lvert\hat{P}\rvert$ positive and $\lvert\hat{N}\rvert$ negative examples satisfying the new rule's body. Then the *Foil Gain* of I_j is defined as: $\Delta\tau = \lvert\hat{P}\rvert(\log\frac{\lvert P\rvert}{\lvert P\rvert+\lvert N\rvert} - \log\frac{\lvert\hat{P}\rvert}{\lvert\hat{P}\rvert+\lvert\hat{N}\rvert})$
A probabilistic transition rule	Ant k moves from gene i to an un-visited gene j at tth iteration by following probability $P_{ij}^k(t)$ defined by: $P_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \notin \text{tabu}_k(t)}[\tau_{il}(t)]^\alpha \cdot [\eta_{ij}]^\beta}$ The parameters α and β control the bias on the pheromone trail or the problem-dependent heuristic function.	Ant k adds the itemset I_j at tth iteration by following probability $P_j^k(t)$ defined by: $P_j^k(t) = \frac{[\tau_j(t)]^\alpha \cdot [\eta_j]^\beta}{\sum_{i=1}^{k} x_i \cdot \sum_{l=1}^{b_j}([\tau_l(t)]^\alpha \cdot [\eta_l]^\beta)}$ where k is the total number of attributes; x_i is set to 1 if the attribute A_j was not yet used by the current ant or to 0 otherwise; and b_j is the number of values in the domain of the jth attribute.
Stopping criteria	Either be a predefined maximum number of iterations or the change in the average gene distance to the cluster centroid between two successive iterations.	Either be a predefined maximum number of iterations or the predefined minimum number of cases covered by the rules generated.

Given N genes $g_i, i = 1, \ldots, N$ and their expression profiles $E_i = \langle a_{1i}, a_{2i}, \ldots, a_{Mi} \rangle$, $i = 1, \ldots, N$, we want to cluster these genes into several categories based on similarities between their expression profiles. The design issues listed in Section 3 can be addressed as shown in Table 1. The procedure of the Ant-C algorithm is illustrated in Fig. 1 where it consists of four main processes, *Initialization, Tour Construction, Pheromone Update*, and *Cluster Output*.

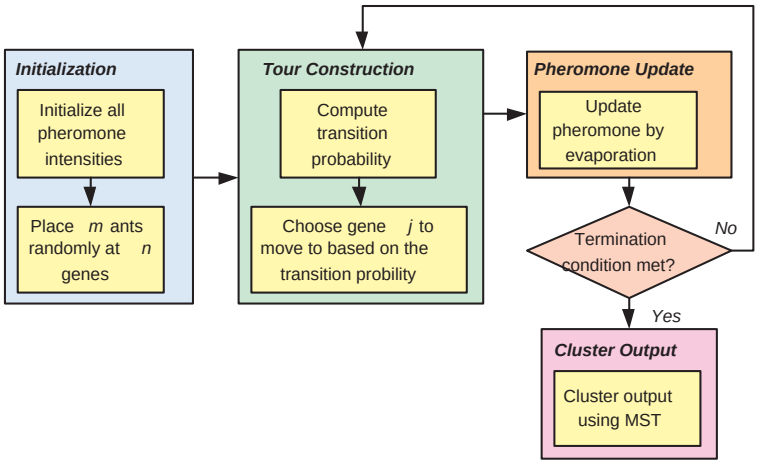


Figure 1: Ant-C: an ant-based clustering algorithm.

Once Ant-C has run successfully, a fully connected network of nodes will be formed. Each node represents a gene, and every edge is associated with a certain level of pheromone intensity. The next step is essentially to break the linkages in order to generate clusters. The simplest way is to use the average pheromone strategy where the average pheromone of all the edges is first computed and then used as a threshold in determining the meaningfulness of the edges. Edges with pheromone intensity less than the average pheromone will be removed from the network which results in a partially connected graph. Nodes will then be

separated by their connecting edges to form clusters.

Here, minimum spanning trees (MSTs) are applied to break the linkages of the fully connected network to generate clusters. When searching for the MST, pheromone intensity is used to measure the similarity between two genes instead of the commonly used Euclidean distance or correlation distance. Pheromone intensities associated with every edge in a fully-connected network records the collective memory of the ants through which self-organizing behavior could be easily discovered. After finding an MST T for a weighted graph, we can partition T into K subtrees, for some specified integer $K > 0$ [38]. These K subtrees would then correspond to K clusters.

5. Ant-Based Association Rule Mining (Ant-ARM)

Suppose a tuple t in the dataset $\mathcal{D}$ follows the schema $(A_1, A_2, ..., A_k)$ where $A_1, ..., A_k$ are called attributes. Let $\mathcal{I} = \{I_1, I_2, ..., I_n\}$ be a set of items in the dataset $\mathcal{D}$. Each item is an attribute-value pair, taking the form of (A_i, v) where A_i is an attribute and v is an value. Here, we assume that continuous attributes can be discretized into consecutive positive integers.

Let $\mathcal{C} = \{c_1, c_2, ..., c_m\}$ be a finite set of class labels. A training data set is a set of tuples such that for each tuple t, there exists a class label $c_t \in \mathcal{C}$ associated with it. A class association rule (CAR) is an implication of the form $iset \Rightarrow c$ where $iset \subseteq \mathcal{I}$ is a subset of items, and $c \in \mathcal{C}$. The confidence of the CAR is defined as the percentage of the cases in $\mathcal{D}$ containing $iset$ that are labeled with class c. The support of the CAR is defined as the percentage of

cases in $\mathcal{D}$ that contain $iset$ and are labeled with class c.

More formally, associative classification is to find all CARs that have support above minSupport. For each CAR, the support count of $iset$ is the number of cases in $\mathcal{D}$ that contain $iset$. The support count of the rule $iset \Rightarrow c$ is the number of cases in $\mathcal{D}$ that contain $iset$ and are labeled with class c. Therefore,

$$confidence(iset \Rightarrow c) = \frac{\text{supportCount}(iset \Rightarrow c)}{\text{supportCount}(iset)} \quad (1)$$

$$support(iset \Rightarrow c) = \frac{\text{supportCount}(iset \Rightarrow c)}{|\mathcal{D}|} \quad (2)$$

where $|\mathcal{D}|$ is the size of the dataset.

Ant-ARM is applied to discover classification rules for one particular class only. That is, the training set will contain cases from one particular class. One ant is used to construct one CAR and it adds one itemset at a time. The design issues listed in Section 3 can be addressed as shown in Table 1. The procedure of the Ant-ARM algorithm is illustrated in Fig. 2 where it consists of four main processes, *Initialization, Tour Construction, Pheromone Update, and Rule Set Update*.

A more detailed elaboration of the Ant-ARM algorithm is shown in Fig. 3. The purpose of the *Rule Set Update* step is to remove redundant and noisy information. It compares the newly constructed rule with the existing rules in R and decides whether to prune it or not. The *selection of the n-best rules* is to compare the rules constructed by all the ants at one iteration and only keep

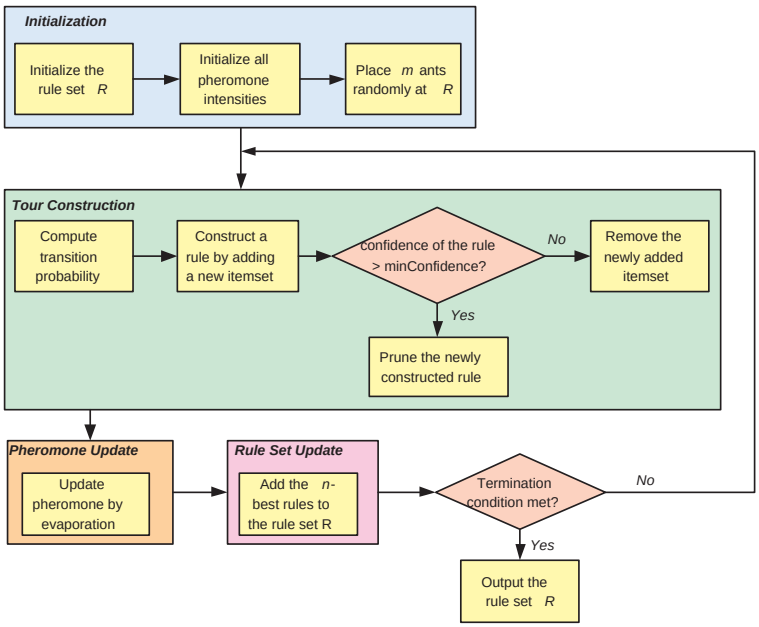


Figure 2: Ant-ARM: an ant-based association mining algorithm.

the top n rules. The criteria used here are listed below [39]:

1. Sort the rules in the rank descending order. Given two rules r_1 and r_2, r_1 is said having higher rank than r_2, denoted as $r_1 > r_2$ if and only if (a) $conf(r_1) > conf(r_2)$; or (b) $conf(r_1) = conf(r_2)$ but $supp(r_1) > supp(r_2)$; or (c) r_1 and r_2 have the same confidence and support value, but r_1 has fewer attribute values in its left hand side than r_2 does. In this case, r_2 will be pruned.
2. For each rule $r : iset \Rightarrow c$, test whether $iset$ is positively correlated with c by χ^2 testing. Only the rules with χ^2 value exceeding a significance level threshold are kept.

1. **Initialization.**
   Set the iteration counter $t = 0$
   Initialize the rule set R to be the set of all 1-itemset rules, i.e. $R = \{\{iset\} \Rightarrow c | iset \in \mathcal{I}\}$
   forall rule r in R do
       if $supp(r) < minSupport$ then Remove r from R end if
   end for
   For every remaining itemset I_j, set an initial value $\tau_j(t)$ for trail intensity and $\Delta\tau_j = 0$.
   Place m ants randomly on the rule set R.
2. Set the tabu list index $s = 1$.
   for $k = 1$ to m do
       Place starting itemset of the kth ant in $\text{tabu}_k(s)$.
   end for
3. **Tour Construction.**
   repeat until tabu list is full
       Set $s = s + 1$
       for $k = 1$ to m do
           Construct rule r_t^k by adding the itemset I_j to the current
               rule with probability $P_j^k(t)$
           Insert the itemset I_j to the tabu list $\text{tabu}_k(s)$
           if $conf(r_t^k) > minConfidence$ then Perform rule pruning
           else Remove rule r_t^k end if
       end for
   end repeat
4. **Pheromone Update.**
   for every itemset I_j do
       $\Delta\tau_j = \sum_{k=1}^{m} \Delta\tau_j^k$
       compute $\tau_j(t+1) = (1-\rho)\tau_j(t) + \Delta\tau_j$
       set $\Delta\tau_j = 0$
   end for

5. **Rule Set Update.**
   Choose the n-best rules among all rules constructed by all the ants
   Add the n-best rules to the rule set R

6. Set $t = t + 1$
   if $t >$ max iteration or the cases covered by the constructed rules $>$ minCoverage then Stop
   else Empty tabu lists, go to 2 end if

Fig. 3. The procedure of Ant-ARM.

6. Experiments

An analysis of the experimental results obtained from the Ant-C and the Ant-ARM algorithms is performed in this section. For each algorithm, the experimental setup is first explained followed by an analysis of the results produced.

6.1. Ant-C Experiments

After the investigation of the suitability of various datasets in Stanford Genomic Resource Database[3], three datasets were chosen to evaluate the performance of our algorithms. The dataset I is a subset of gene expression data in the yeast Saccharomyces cerevisiae (SGD)[4], which is commonly known as baker's or budding yeast. A set of 68 genes with each gene having 79 data points is chosen. The dataset II is a temporal gene expression dataset in response of human fibroblasts to serum[5]. It consists of 517 genes and each gene has 18 data points. In this dataset, genes are listed according to their cluster order along with their Gene bank Accession number and Clone IDs. Gene names with the SID prefix are not sequence verified. The expression changes are given as the ratio of the expression level at the given time-point to the expression level in serum-starved fibroblasts. The dataset III is the rat central nervous system development dataset[6]. It is obtained by researchers using the method of reverse transcription-coupled PCR to study the expression levels during rat central nervous system development.

Two most commonly used similarity measures for gene expression data are Euclidean distance and Pearson correlation coefficient [40]. Euclidean distance is sensitive to scaling and difference in average expression level. This problem can be addressed by standardizing each gene object with zero mean and unit variance before calculating the distance between genes. Pearson correlation coefficient

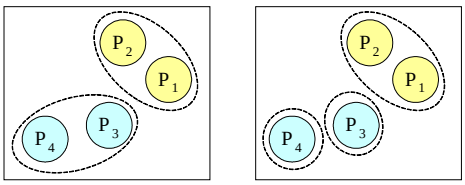
[3] `http://genome-www.stanford.edu/` (accessed: 17 March 2008)
[4] `http://www.yeastgenome.org/` (accessed: 17 March 2008)
[5] `http://genome-www.stanford.edu/serum/` (accessed: 17 March 2008)
[6] `http://www.arclab.org/node_pages/265.html` (accessed: 17 March 2008)

is not robust with respect to outliers [41]. Nevertheless, it has been shown in [42] that if standardization is performed on data objects, then the effectiveness of a clustering algorithm is equivalent whether Euclidean distance or Pearson correlation coefficient is chosen as a similarity measure. In our experiments reported here, all the gene objects were standardized with zero mean and unit variance prior to clustering and Euclidean distance was chosen as the similarity measure.

6.1.1. Evaluation Metric

The Rand index [43] is used to evaluate the performance of the clustering algorithm. In statistics, the Rand index is a measure of the similarity between two data clusters. Let V_1 and V_2 be two partitions of n objects. V_1 and V_2 may contain a different number of clusters. Let a be the number of pairs of objects that are placed in the same cluster in partition V_1 and in the same cluster in partition V_2, and d be the number of pairs of objects in different clusters in both partition V_1 and V_2, The Rand index is a fraction of agreement, i.e., Rand index $= (a + d)/\binom{n}{2}$. The Rand index lies between 0 and 1, with 0 indicating that the two data clusters do not agree on any pair of points and 1 indicating that the data clusters are exactly the same.

An illustration on how to calculate the Rand index between two partitions of 4 points is given in Fig. 4. Partition V_1 contains 2 clusters, $V_1 = \{(P_1, P_2), (P_3, P_4)\}$ and partition V_2 contains 3 clusters, $V_2 = \{(P_1, P_2), (P_3), (P_4)\}$. From the statistics of the point pairs shown in Fig. 4, it can be easily seen that $a = 1$ and $d = 4$, thus, Rand index $= (1 + 4)/\binom{4}{2} = 0.83$.



Point pair	P_1P_2	P_1P_3	P_1P_4	P_2P_3	P_2P_4	P_3P_4	Total
Together in both	√						1
Separate in both		√	√	√	√		4
Mixed						√	1

Figure 4: An illustration of calculating the Rand index.

Table 2: Statistics on experimental data where each entry denotes the numbers of genes belonging to the corresponding cluster.

	Gene Cluster					
Dataset	A	B	C	D	E	F
I	28	17	15	8	-	-
II	305	43	7	162	-	-
III	27	20	21	17	21	6

6.1.2. Results

The detailed "expert" cluster information of the three datasets is shown in Table 2 where dataset I and II have 4 clusters each and dataset III has 6 clusters. Table 3 lists the experimental results using Ant-C on different datasets. Results using the classical clustering algorithms such as AHC and K-means are also presented. It can be observed from Table 3 that the performance of the Ant-C algorithm is better than that of AHC and K-means on dataset I and II. The Rand index value achieved is 0.936 on dataset I and 0.811 on dataset II. However, the Rand index values obtained using Ant-C on dataset III are lower than that of K-means though Ant-C slightly outperforms AHC. One possible reason of

Table 3: Comparison of experimental results on different algorithms.

	Rand Index		
Methods	**Dataset I**	**Dataset II**	**Dataset III**
Ant-C	**0.936**	**0.811**	0.582
AHC	0.803	0.628	0.575
K-means	0.701	0.565	**0.676**

better performance of K-means on dataset III is that the exact cluster number 6 was preset by the user while in practice it is hard to predict the correct cluster number.

The optimal number of clusters is determined automatically based on the transition profile values [38] which is defined as follows:

$$T(k) = \frac{\|Q(k-1) - Q(k)\|}{\|Q(k) - Q(k+1)\|} \tag{3}$$

where k is the number of clusters, $T(k)$ is the transition profile value for k clusters, $Q(k)$ is a cluster cohesion value which measures the closeness among objects within a cluster and it is defined as $Q(k) = \sum_{i=1}^{k} \sum_{j=1}^{N_i} dist(\vec{g_j}, \vec{c_i})$,where $\vec{c_i}$ is the centroid vector of the ith cluster, $\vec{g_j}$ is the jth gene vector which belongs to cluster i, N_i stands for the number of genes which belongs to the ith cluster. $Q(0)$ always equals to 0. The highest transition profile value determines the optimal number of clusters.

Fig. 5 shows the transition profile diagrams for dataset I, II and III. In the transition profile diagram, the x-axis represents the number of clusters, while the y-axis represents transition profile values. It can be observed from Fig. 5(a)

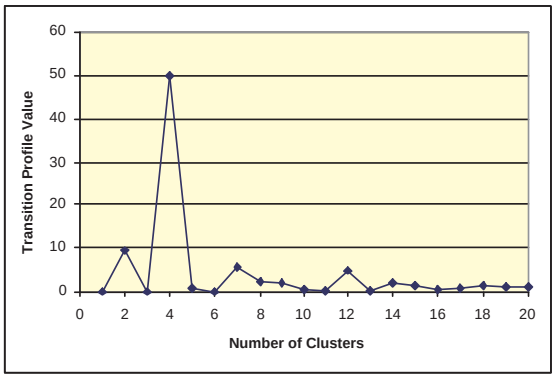
that the optimal number of clusters in dataset I is 4, which is same as the actual number of clusters as can be found in Table 2. While for dataset II, the optimal number of clusters is 3 as shown in Fig. 5(b). This is slightly different from the actual cluster number 4. Fig. 5(c) reveals that the optimal number of clusters in dataset III is 2 which is different from the actual cluster number 6. This also explains the worse performance of Ant-C in dataset III.

6.2. Ant-ARM Experiments

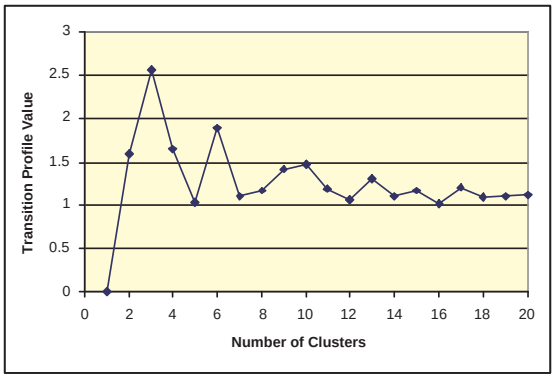
Associative classification experiments were conducted using a widely used Leukemia ALL/AML gene expression dataset [44]. The actual dataset can be found in Broad Institute Database[7]. The ALL/AML dataset used consists of 38 bone marrow samples obtained from acute leukemia patients. 27 of them are acute leukemia arising from lymphoid precursors (acute lymphoblastic leukemia, ALL) while 11 cases are acute leukemia arising from myeloid precursors (acute myeloid leukemia, AML). There are 7129 genes (attributes), 2 class labels (ALL/AML) with no missing attributes in any of the sample.

Each gene-sample pair has a gene expression value. When entropy-based discretization [45] is applied, the gene expression value is discretized into suitable intervals. For example, the gene AFFX-BioC-5_at (endogenous control) can be discretized into $(-\infty, 317], (317, \infty)$. We denote $(-\infty, 317]$ by 1 (downregulated gene) and $(317, \infty)$ by 2 (upregulated gene). The partition that separates the interval is known as cutting point. With the minimum entropy idea, the inter-

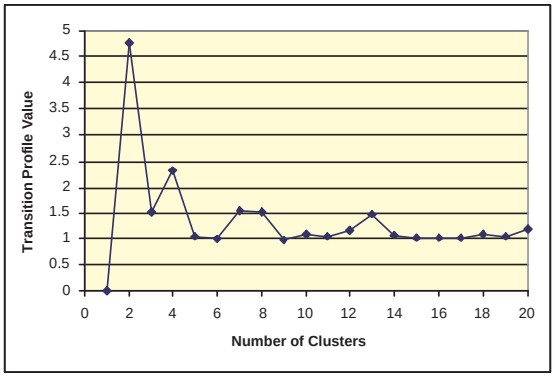
[7] `http://www.broad.mit.edu/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=43` (Accessed: 17 March 2008)



(a) Dataset I.



(b) Dataset II.



(b) Dataset III.

Figure 5: Transition profile diagrams of the three datasets.

vals are 'maximally' discriminatory between expression values from normal cells and those from cancer cells. This method automatically ignores those ranges which contain uniformly mixed normal and cancer cells' expression values. That is, those noisy features (genes) would not be of any use for the Ant-ARM algorithm and they are removed. As a result of the application of entropy-based discretization, ALL/AML dataset has a reduction of 7129 genes to 866 genes.

The tunable parameters in the Ant-ARM algorithm include the number of iterations i, the number of ants m, the evaporation rate ρ, and the parameters α and β that control the bias on the pheromone trail. A set of experiments have been conducted and it was found that the good value or range of each parameter is $i = 20$, $m = 10$, $\rho = 0.2$, $\alpha = 1$, and $\beta = 1$. From this section onwards, subsequent experiments will be carried out using these values for the parameters.

We have tried various ARM algorithms on the ALL/AML dataset, including Apriori [3], FP-growth [46], and Magnum Opus[8] which is based on OPUS (Optimized Pruning for Unordered Search) [47], a systematic search method with pruning. However, the CPU's memory space was exhausted when using Apriori. Similar results have been reported in other papers [48, 23, 24]. FP-growth has been declared as one of the fastest approaches to frequent item set mining. But still, it yielded no result despite running for more than 25 hours. Magnum Opus only processed less than 10% of the data after running for 13 hours.

[8] `http://www.rulequest.com/MagnumOpus-info.html` (Accessed: 17 March 2008).

6.2.1. Run Time Analysis

As most existing ARM algorithms were unable to process the ALL/AML datasets, we instead used other two benchmark datasets in our experiments for run time comparison. One comprises of synthetic data resemble market basket data with short frequent patterns generated using a transaction data generator obtained from IBM Almaden, typically designated as T10I4D100K[9]. Another dataset, Mushroom[10], is real data which are dense in long frequent patterns. These data sets were often used in the previous study of association rules mining. Table 4 shows the characteristics of these two datasets.

Table 4: Characteristics of the two datasets used for run time analysis.

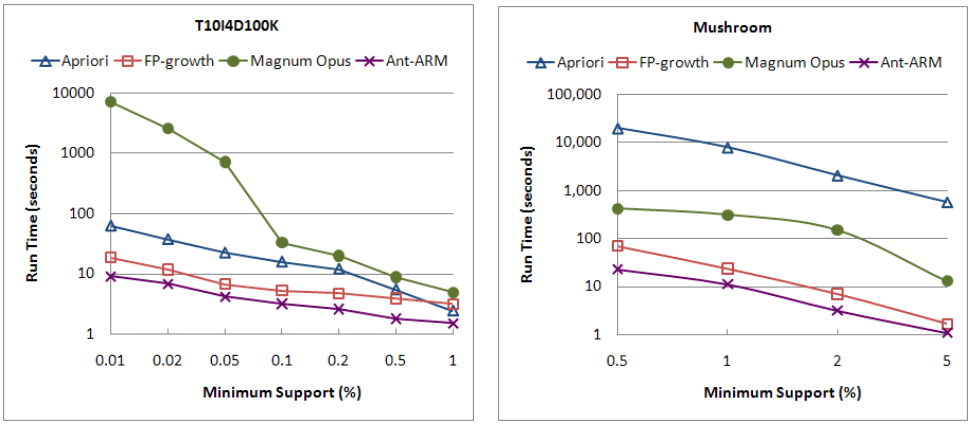
Dataset	*No. of Trans.*	*No. of Items*	*Avg Trans. Length*
T10I4D100K	100,000	1,000	10
Mushroom	8,124	120	23

The computer used to run the experiments had the configurations of Pentium CPU 2.66GHz with 3.25GB of memory, running Windows XP. Fig. 6(a) shows the run time of the compared algorithms on the T10I4D100K dataset with different minimum support values ranging from 0.01% to 1% while Fig. 6(b) shows the run time on the Mushroom dataset with minimum support values ranging from 0.1% to 5%. The run time is shown on a logarithmic scale (base 10).

It can be observed that for T10I4D100K, when the minimum support is large, there is not much difference between various algorithms. However, when grad-

[9] `http://fimi.cs.helsinki.fi/data/T10I4D100K.dat` (accessed: 18 November 2008)
[10] `http://archive.ics.uci.edu/ml/datasets/Mushroom` (accessed: 18 November 2008)

ually reducing the minimum support, the difference in running time becomes more obvious. The run times of all algorithms increase exponentially as the support threshold is reduced. Magnum Opus is significantly slower than all the other algorithms especially with lower minimum support threshold. Ant-ARM and FP-growth outperform Aprior with a larger margin when the minimum support decreases. For the Mushroom dataset, Aprior runs significantly slower than the other three algorithms. For both datasets, Ant-ARM runs faster than all the other algorithms and there is a considerable gap in the performance of the second best algorithm, FP-growth, with respect to our algorithm. The difference is more significant for the Mushroom dataset.



(a) T10I4D100K. (b) Mushroom.

Figure 6: Run time of the algorithms for the two benchmark datasets.

6.2.2. Parameters Tuning

Parameters which are commonly used for the construction of association rules are the *minsup* (minimum support threshold), *minconf* (minimum confi-

dence threshold) and *minchi* (minimum chi-square threshold). Parameter tuning was performed based on the accuracy of the rules generated. As there is no gold standard rule set available for the ALL/AML dataset, one possible way is to compare the rules generated by Ant-ARM with those output by other existing algorithms. However, this seems not possible as none of the existing well-known ARM algorithms is able to process the ALL/AML dataset within a reasonable period of time. We found that Magnum Opus allows the user to choose the items in the LHS and RHS for the association rules to be generated. Thus, for each rule generated by Ant-ARM, we could manually specify the allowed LHS and RHS items in Magnum Opus to see whether the same rule could be generated. For example, if a rule output by Ant-ARM is $Gene1 \uparrow, Gene3 \downarrow \rightarrow AML$, then in Magnum Opus, the desired genes $Gene1$ and $Gene3$ are selected for LHS and AML and ALL are selected for values allowed on RHS. This guarantees that Magnum Opus can generate association rules within short time. Accuracy is calculated as the total number of the matched rules with those generated from Magnum Opus over the total number of the rules generated from the Ant-ARM algorithm.

The following settings were used in Magnum Opus: the default value of 0.001 for minimum leverage of interest, the default value of 0 for minimum coverage of interest, the maximum number of associative rules being 1000. The value of minimum support is adjusted according to the support used by Ant-ARM.

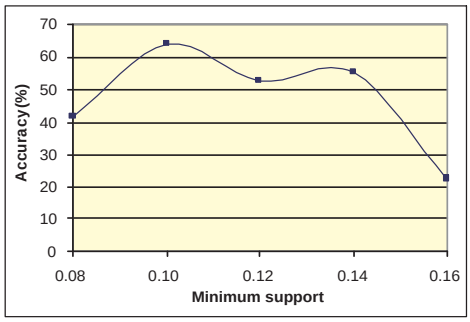
Fig. 7 shows the relationships between the ARM accuracy and the various parameters. The effect of varying *minsup* is displayed in Fig. 7(a). Both *minconf*

and *minchi* values are set to zero in this experiment so that rule pruning is disabled. It can be observed that the support that yields the greatest accuracy is at 0.1 where 63.9% accuracy is obtained. Increasing the support threshold value resulted in decreased performance. For *minsup* of 0.1, it is relatively low. However, it has been noted that only at low minimum support do interesting (predictive) findings being found [49]. A test conducted in [50] reveals that *minsup* of 0.1 gives the most effective operational results.

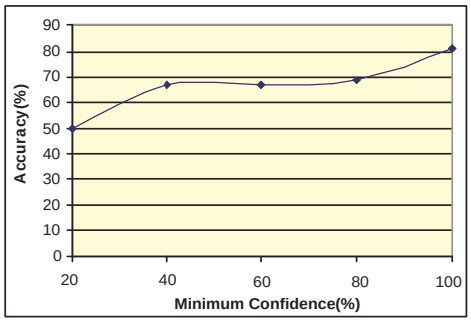
Fig. 7(b) shows the accuracy versus various minimum confidence values. The remaining parameters used are $minsup = 0.1, minchi = 0$. It can be observed that the *minconf* value that yields the highest accuracy is 100% where its accuracy stands at 80.7%. This is a significant improvement from Fig. 7(b) where the best accuracy is just 63.9%. As researchers are interested in rules with good classification power (high confidence) [51], a *minconf* of 100% is desirable. Such conclusion of $minconf = 100\%$ can also be found in [50].

Fig. 7(c) shows the accuracy versus various chi-square values. The remaining parameters used are $minsup = 0.1, minconf = 100\%$. The best accuracy 90% was obtained when *minchi* was set to 0.4. Increasing the *minchi* value degraded the performance. Finally, Fig. 7(d) shows that the optimal accuracy is achieved when 3-best rules were kept at each iteration of the Ant-ARM algorithm (Step 5 in Fig. 2).

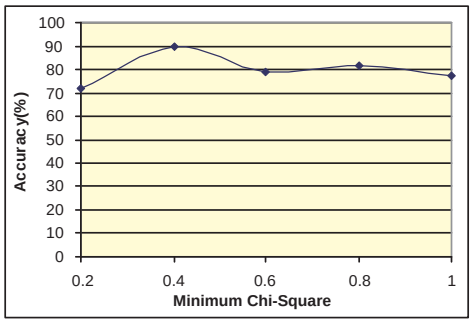
In summary, the best accuracy is obtained using a low support threshold. On the other hand, the choice of confidence threshold is also important. Low confidence threshold has resulted in bad performance as reported in [50].



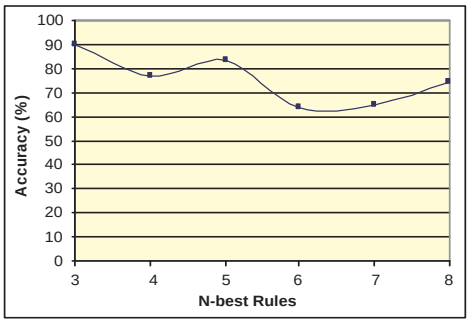
(a) Support vs. accuracy.



(b) Confidence vs. accuracy.



(c) Chi-Square vs. accuracy.



(d) N-best rules vs. accuracy.

Figure 7: ARM accuracy vs. various parameters.

6.2.3. Result Summary

Table 5 illustrates the number of rules generated and the number of terms per rule respectively as different parameters are tuned. As can be seen from these figures, the more optimum the parameters are tuned, the simpler are the rules generated. This is a desired result where simplicity is of importance compared to its predictive accuracy [52].

Table 5: Results on the no. of rules generated and the no. of terms per rule.

Support	0.08	**0.10**	0.12	0.14	0.16
No. of Rules	36	36	36	36	36
No. of terms/rule	2.58	2.72	2.69	2.75	2.56
Confidence (%)	20	40	60	80	**100**
No. of rules	36	36	36	35	31
No. of terms/rule	2.55	2.89	2.22	2.46	2.06
Chi-Square	0.2	**0.4**	0.6	0.8	1.0
No. of rules	25	30	29	33	31
No. of terms/rule	1.96	1.97	1.83	2	1.81

Fig. 8 shows the run time of Ant-ARM on the ALL/AML dataset by varying the minimum support value from 0.08 to 0.16. It can be observed that the run time of Ant-ARM decreases with the increasing minimum support value. Ant-ARM run for 4.2 hours to generate associative classification rules at the best minimum support value 0.1.

Experiments have also been conducted to evaluate the effect of pheromone update in Ant-ARM. Instead of choosing an itemset to add to the current rule based on the pheromone deposited at the path to that itemset, an ant randomly picked an itemset to add to the rule. Essentially, the Ant-ARM's search for rules is guided by the comparison with the minimum support threshold only. Table 6

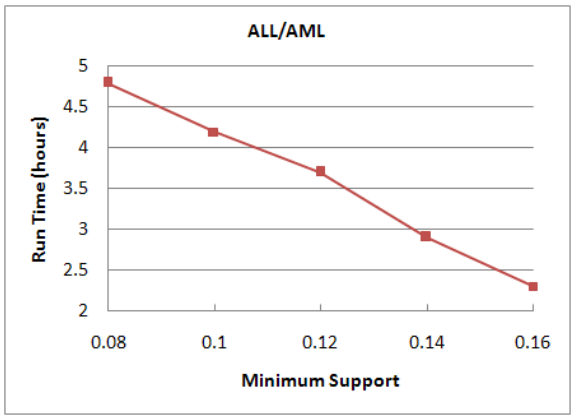


Figure 8: Run time of Ant-ARM for the ALL/AML dataset.

shows the results of Ant-ARM with or without pheromone update by varying n as in keeping the n-best rules at the end of each iteration. It can be observed that Ant-ARM without pheromone update consistently performed worse than Ant-ARM with pheromone update.

Table 6: Results with or without pheromone update by varying n as in keeping the n-best rules at the end of each iteration.

	With pheromone update	*Without pheromone update*
n	*Accuracy(%)*	*Accuracy(%)*
3	90.0	85.2
4	76.9	71.8
5	83.3	79.5
6	63.9	61.4
7	65.0	60.5
8	74.3	69.2

Table 7 shows some of the associative classification rules generated by Ant-ARM. Almost all of the genes in Table 7 can be found in [53] where a list of 50 genes are termed as marker genes that are involved in Leukemia and can distinguish AML from ALL.

Table 7: Example associative classification rules generated by Ant-ARM.

No.	*Associative Classification Rules*
1	{IL-7 receptor (M29696)↑} ⇒ ALL
2	{CD19 antigen (M28170)↑} ⇒ ALL
3	{LYN (M16038)↓} ⇒ ALL
4	{LTC4 synthase (U50136) ↓, CRADD (U84388)↓} ⇒ ALL
5	{IRF2 (X15949)↑} ⇒ ALL
6	{TFIEβ(X63469)↑} ⇒ ALL
7	{Leptin receptor (Y12670)↑} ⇒ AML
8	{p62 (U46751)↑ SNF2 (U29175)↓} ⇒ AML

Leptin receptor (Y12670) has been cited as one of the new markers of acute leukemia subtype [44]. It is originally identified through its role in weight regulation and has high relative expression in AML. Furthermore, this gene has been recently found to have antiapoptotic function in hematopoietic cells.

What is more interesting in the findings is that in all of the experiments conducted, more than half of the experiments show the presence of CD19 antigen (M28170) gene in one or more of the associative classification rules. According to [54], CD19 antigen is a cell surface molecule expressed only by B-lymphocytes and follicular dendritic cells of the hematopoietic system. It is present on most pre-B and most non-T acute lymphocytic leukemia cells. Furthermore, it is the earliest of the B-lineage-restricted antigens to be expressed. Thus, CD19 antigen may be an important gene that differentiates ALL from AML.

It should also be noted that Rule 4's "CRADD (U84388) ↓" cannot be matched in [44, 54] while "LTC4 synthase (U50136)↓ ⇒ ALL" can be matched. Such an observation on the co-relationship between these two genes and ALL can be used for future investigation.

7. Conclusions

This paper explored the Ant-C and Ant-ARM algorithms for gene expression data analysis. The proposed algorithms make use of the natural behavior of ants such as cooperation and adaptation to allow for a flexible robust search for a good candidate solution.

Unlike other existing ACO-based clustering approaches where ants move around in a 2D grid and carry or drop objects to perform categorization, the novel Ant-C approach proposed here does not rely on a 2D grid structure. In addition, it can also generate optimal number of clusters without incorporating any other algorithms such as K-means or AHC. Overall, Ant-C shows a relatively high accuracy compared to other classical clustering algorithms when tested on the three gene expression datasets.

For associative classification on gene expression data, first, the entropy-based discretization pre-processing method was used to remove many noisy features (genes) and explore the remaining most discriminatory features. Such pre-processing effectively reduces the search space required and thus reduces over-head time. The rules discovered by the Ant-ARM algorithm are in general quite simple with an average of 3 items per rule. Such simplicity improves the comprehensibility to a human who would be making intelligent decision based on these rules. Moreover, the algorithm delivers its ability to mine large datasets. A few of the well-known algorithms such as Apriori, FP-growth and Magnum Opus have been used to experiment on ALL/AML dataset. However, these algorithms are unable to mine any association rules from the dataset within

a reasonable period of time. Therefore, given the accuracy, simplicity of rules and ability to mine large datasets, the Ant-ARM algorithm has the advantage compared to most of the ARM algorithms.

Nevertheless, the run time of Ant-ARM needs to be further improved. In future work, we intend to incorporate other algorithms for attribute reduction such as the one proposed in [55] which is based on rough set theory to reduce the run time of Ant-ARM.

References

[1] A. Tuzhilin, G. Adomavicius, Handling very large numbers of association rules in the analysis of microarray data, in: O. Zaïane, R. Goebel, D. Hand, D. Keim, R. Ng (Eds.), Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA, 2002, pp. 396–404.

[2] P. Kotala, P. Zhou, S. Mudivarthy, W. Perrizo, E. Deckard, Gene expression profiling of DNA microarray data using peano count trees (p-trees), in: Online Proceedings of the 1st Virtual Conference on Genomics and Bioinformatics, North Dakota State University, USA, 2001, pp. 15–16.

[3] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, in: P. Buneman, S. Jajodia (Eds.), Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, 1993, pp. 207–216.

[4] X. Xu, G. Cong, B. Ooi, K.-L. Tan, A. Tung, Semantic mining and analysis of gene expression data (demo), in: M. Nascimento, M. Özsu, D. Kossmann, R. Miller, J. Blakeley, K. Schiefer (Eds.), Proceedings of the 30th International Conference on Very Large Data Bases (VLDB'04), Morgan Kaufmann, St. Louis, MO, US, 2004.

[5] H. Ressom, Y. Zhang, J. Xuan, Y. Wang, R. Clarke, Inference of gene regulatory networks from time course gene expression data using neural networks and swarm intelligence, in: D. Ashlock, C. Bauch (Eds.), IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, IEEE, Los Alamitos, CA, US, 2006, pp. 1–8.

[6] K. Robbins, W. Zhang, J. Bertrand, R. Rekaya, The ant colony algorithm for feature selection in high-dimension gene expression data for disease classification, Mathematical Medicine and Biology 24 (4) (2007) 413–426.

[7] M. Eisen, P. Spellman, P. Brown, D. Botstein, Cluster analysis and display of genome-wide expression patterns, Proceedings of the National Academy of Sciences of the United States of America 95 (14) (1998) 14863–14868.

[8] X. Wen, S. Fuhrman, G. Michaels, D. Carr, Large-scale temporal gene expression mapping of central nervous system development, Proceedings of the National Academy of Sciences of the United States of America 95 (1) (1998) 334–339.

[9] R. Herwig, A. Poustka, C. Müller, C. Bull, Large-scale clustering of cdna-fingerprinting data, Genome Research 9 (11) (1999) 1093–1105.

[10] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation, Proceedings of the National Academy of Sciences of the United States of America 96 (6) (1999) 2907–2912.

[11] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley, 2006.

[12] S. Wu, T. Chow, Self-organizing-map based clustering using a local clustering validity index, Journal Neural Processing Letters 17 (3) (2004) 253–271.

[13] J. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, L. Chretien, The dynamics of collective sorting robot-like ants and ant-like robots, in: J.-A. Meyer, S. Wilson (Eds.), Proceedings of the 1st international conference on simulation of adaptive behavior on From animals to animats, MIT Press, Cambridge, MA, USA, 1990, pp. 356–363.

[14] E. Lumer, B. Faieta, Diversity and adaptation in populations of clustering ants, in: D. Cli, P. Husbands, J. Meyer, S. Wilson (Eds.), Proceedings of the 3rd International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 3, MIT Press, Cambridge, MA, 1994, pp. 501–508.

[15] P. Kuntz, P. Layzell, D. Snyers, A colony of ant-like agents for partitioning in VLSI technology, in: P. Husbands, I. Harvey (Eds.), Proceedings of the 4th European Conference on Artificial Life, MIT Press, Cambridge, MA, USA, 1997, pp. 417–424.

[16] N. Monmarche, On data clustering with artificial ants, in: A. Freitas (Ed.), Data Mining with Evolutionary Algorithms: Research Directions, AAAI Press, Orlando, Florida, 1999, pp. 23–26.

[17] B. Wu, Y. Zheng, S. Liu, Z. Shi, CSIM: a document clustering algorithm based on swarm intelligence, in: Proceedings of the 2002 congress on Evolutionary Computation, IEEE Computer Society, Washington, DC, 2002, pp. 477–482.

[18] Y. Peng, X. Hou, S. Liu, The k-means clustering algorithm based on density and ant colony, in: Proceedings of IEEE International Conference on Neural Networks and Signal Processing, IEEE, Los Alamitos, CA, 2003, pp. 457–460.

[19] J. Handl, B. Meyer, Improved ant-based clustering and sorting, in: J. Guervós, P. Adamidis, H.-G. Beyer, n. J.L. Fernández-Villaca H.-P. Schwefel (Eds.), PPSN VII: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature, Springer-Verlag, London, UK, 2002, pp. 913–923.

[20] L. Chen, X. Xu, Y. Chen, An adaptive ant colony clustering algorithm, in: D. Yeung, X. Wang, J. Su (Eds.), Proceedings of the 3rd International Conference on Machine Learning and Cybernetics, IEEE, Los Alamitos, CA, 2004, pp. 1387–1392.

[21] L. Herrmann, A. Ultsch, Explaining ant-based clustering on the basis of self-organizing maps, in: M. Verleysen (Ed.), Proc. of the European Sym-

posium on Artificial Neural Networks (ESANN 2008), Bruges, Belgium, 2008, pp. 215–220.

[22] C. Creighton, S. Hanash, Mining gene expression databases for association rules, Bioinformatics 19 (1) (2003) 79–86.

[23] Z. Zheng, R. Kohavi, L. Mason, Real world performance of association rule algorithms, in: D. Lee, M. Schkolnick, F. Provost, R. Srikant (Eds.), Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2001, pp. 401–406.

[24] C. Becquet, S. Blachon, B. Jeudy, J. F. Boulicaut, O. Gandrillon, Strong-association-rule mining for large-scale gene-expression data analysis: A case study on human sage data, Genome Biology 3 (12) (2002) 1465–6914.

[25] G. Cong, A. Tung, X. Xu, F. Pan, J. Yang, Farmer: Finding interesting rule groups in microarray datasets, in: G. Weikum, A. König, S. Deßloch (Eds.), Proceeding of the ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, 2004, pp. 143–154.

[26] E. Yu, K. Sung, A genetic algorithm for a university weekly courses timetabling problem, International Transactions in Operational Research 9 (6) (2002) 703–717.

[27] E. Burke, D. Elliman, R. Weare, A genetic algorithm based university timetabling system, in: P. Brusilovsky, V. Stefanuk (Eds.), Proceedings of

the 2nd East-West International Conference on Computer Technologies in Education, International Centre for Scientific and Technical Information, Moscow, Russia, 1994, pp. 35–40.

[28] M. Dorigo, V. Maniezzo, A. Colorni, Positive feedback as a search strategy, Technical report 91-016, Politecnico di milano, dip. Elettronica (1991).

[29] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics - Part B 26 (1) (1996) 29–42.

[30] L. de Castro, F. V. Zuben, Learning and optimization using the clonal selection principle, IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems 6 (3) (2002) 239–251.

[31] D. Dasgupta, Z. Ji, F. Gonzlez, Artificial immune system (AIS) research in the last five years, in: R. Sarker, R. Reynolds, H. Abbass, K. Tan, B. McKay, D. Essam, T. Gedeon (Eds.), Proceedings of the IEEE Congress on Evolutionary Computation Conference (CEC), IEEE, Los Alamitos, CA, US, 2003, pp. 123–130.

[32] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, US, 2004.

[33] W. Gutjahr, Mathematical runtime analysis of aco algorithms: survey on an emerging issue, Swarm Intelligence 1 (1) (2007) 59–79.

[34] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: from Natural to Artificial Systems, New York: Oxford University Press, 1999.

[35] J. Quinlan, R. Cameron-Jones, FOIL: a midterm report, in: P. Brazdil (Ed.), Proceedings of European Conference on Machine Learning, Springer, Vienna, Austria, 1993, pp. 3–20.

[36] Y. He, S. Hui, Y. Sim, A novel ant-based clustering approach for document clustering, in: H. Ng, M.-K. Leong, M.-Y. Kan, D. Ji (Eds.), Asia Information Retrieval Symposium, Springer, Singapore, LNCS 4182, 2006, pp. 537–544.

[37] D. Zhou, Y. He, C. Kwoh, H. Wang, Ant-MST: An ant-based minimum spanning tree for gene expression data clustering, in: J. Rajapakse, B. Schmidt, G. Volkert (Eds.), IAPR Workshop on Pattern Recognition in Bioinformatics, Springer, Singapore, LNCS 4774, 2007, pp. 198–205.

[38] Y. Xu, V. Olman, D. Xu, Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees, Bioinformatics 18 (4) (2002) 536–545.

[39] W. Li, J. Han, J. Pei, CMAR: Accurate and efficient classification based on multiple class-association rules, in: N. Cercone, T. Lin, X. Wu (Eds.), Proceedings of the IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, US, 2001, pp. 369–376.

[40] P. D'haeseleer, How does gene expression clustering work?, Nature Biotechnology 23 (2005) 1499 – 1501.

[41] D. Bickel, Robust cluster analysis of microarray gene expression data with the number of clusters determined biologically, Bioinformatics 19 (7) (2003) 818–824.

[42] D. Jiang, C. Tang, A. Zhang, Cluster analysis for gene expression data: A survey, IEEE Transactions on Knowledge and Data Engineering 16 (11) (2004) 1370–1386.

[43] W. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical Association 66 (1971) 622–626.

[44] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, E. Lander, Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring, Science 286 (5439) (1999) 531–537.

[45] U. Fayyad, K. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: R. Bajcsy (Ed.), Proceedings of the 13th International Joint Conference on Uncertainty in AI, Morgan Kaufmann, Chambery, France, 1993, pp. 1022–1027.

[46] C. Borgelt, An implementation of the FP-growth algorithm, in: B. Goethals, S. Nijssen, M. Zaki (Eds.), Proceedings of the 1st international Workshop on Open Source Data Mining, ACM, New York, NY, USA, 2005, pp. 1–5.

[47] G. Webb, Opus: An efficient admissible algorithm for unordered search, Journal of Artificial Intelligence Research 3 (1995) 431–465.

[48] G. Webb, Efficient search for association rules, Knowledge Discovery and Data Mining (2000) 99–107.

[49] R. Bayardo, The hows, whys, and whens of constraints in itemset and rule discovery, in: J.-F. Boulicaut, L. Raedt, H. Mannila (Eds.), Proceedings of the Workshop on Inductive Databases and Constraint Based Mining, Vol. 3848 of Lecture Notes in Computer Science, Springer, Berlin, 2005, pp. 1–13.

[50] F. Coenen, P. Leng, The effect of threshold values on association rule based classification accuracy, Data and Knowledge Engineering 60 (2) (2007) 345–360.

[51] R. Bayardo, Brute-force mining of high-confidence classification rules, in: D. Heckerman, H. Mannila, D. Pregibon, R. Uthurusamy (Eds.), Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, CA, 1997, pp. 123–126.

[52] R. Parpinelli, H. Lopes, A. Freitas, Data mining with an ant colony optimization algorithm, IEEE Transactions on Evolutionary Computation 6 (4) (2002) 321–332.

[53] M. Chow, E. Moler, I. Mian, Identifying marker genes in transcription profiling data using a mixture of feature relevance experts, Physiol Genomics 5 (2) (2001) 99–111.

[54] S. Bicciato, M. Pandin, G. Didone, C. D. Bello, Pattern identification and

classification in gene expression data using an autoassociative neural network model, Biotechnology and Bioengineering 81 (5) (2003) 594–606.

[55] L. Ke, Z. Feng, Z. Ren, An efficient ant colony optimization approach to attribute reduction in rough set theory, Pattern Recognition Letters 29 (9) (2008) 1351–1357.