The following paper discusses the performance of ArrayMiner™ by Optimal Design as compared with the standard k-Means procedure. The paper was published in Proceedings of the 2001 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS'2001), Las Vegas, Nevada, USA, June 25-28, 2001.

Using k-Means? Consider ArrayMiner

Emanuel Falkenauer[*+] and Arnaud Marchand[+]

[*] Brussels University (ULB)
CAD Department
Av. F. D. Roosevelt 50
B-1050 Brussels, Belgium

[+] Optimal Design
Av. De l'Orée 14 bte 11
B-1000 Brussels, Belgium

Abstract - With the (near-)availability of complete sequences of human and other genomes, research effort is turning from obtaining the sequence itself towards determining the biological function of the genes within the sequence. One approach, made practically feasible with the advent of DNA microarray technology, consists of clustering of genes into groups of coexpressed genes, i.e. genes exhibiting similar behavior in some circumstances. Among the various methods used for identification of groups of coexpressed genes, k-Means is one of the most popular. However, we show here that k-Means is a highly unreliable method of clustering, yielding high-quality solutions with low *probability. We then present the ArrayMiner software based on Genetic Algorithms, which addresses the drawback, supplying high-quality solutions in short time with very high reliability.*

1. Clustering of Expression Profiles

With the (near-)availability of complete sequences of human and other genomes such as Drosophila and Arabidopsis, genomics has produced a significant wealth of sequence data, and the stage has been set for the next task, namely identifying the biological function of the genes within those sequences. Indeed, only this latter knowledge will enable researchers to establish correspondences between diseases and the genome, paving the route to new medication. A major difficulty lies with the fact that the detailed phenomena taking place within organisms are not fully understood and many probably remain to be discovered. Identifying the phenomena and the genes involved, together with the ways in which the genes interact, constitutes the major challenge of "post-sequencing" genetics.

One approach to achieve this consists in identification of groups of coexpressed genes, i.e. groups of genes that behave similarly in some conditions. The rationale behind this approach is that similarly behaving genes probably participate together in some phenomenon or have similar functions – identifying such groups may thus help in identifying the phenomenon, discovering previously unknown phenomena, or assigning previously unknown functions to genes. As an additional benefit, clustering gene expression data into groups reduces the unmanageable volume of data into data sets that can be more easily handled by biologists.

In this approach, each gene (or ORF) is represented by an expression profile, i.e. a series of numerical values of its activity. The profile of a gene may correspond to a set of readings of the gene's activity under various conditions, or be a time-series of the gene's activity after some event. The aim of clustering of the expression profiles being to decide which genes exhibit similar behaviors (i.e., are coexpressed), a measure of similarity between profiles must be defined. Several can be found in the literature, including the Euclidean distance, the correlation, and the Pearson coefficient, this last measure being widely used for the time-series profiles, as it measures the similarity of the *shapes*, rather than the absolute values of the measurements of the two profiles.

2. The k-Means Technique

2.1 Approach

One of the most popular clustering techniques is k-Means, to the extent that it is hardly possible to find gene expression clustering software that would not offer k-Means as a method of identification of groups of coexpressed genes. Given the user-supplied parameter of the number of groups, k-Means finds groups of genes such that within any group, all profiles in that group are closer to the average profile (centroid) of that group than to the average of any other group.

2.2 Operation

K-Means proceeds from an initial position of the given number of group centroids, by iterative assignment of the profiles to their closest centroid followed by adjustment of the centroids' positions into the centers of the resulting groups. The process is illustrated in Figure 1 on a simple example with 11 data points in three groups.

The algorithm terminates when a stable configuration is reached, i.e. when the partition of the profiles into groups does not change anymore. The process usually converges into a stable solution in a low number of iterations, making k-Means a very fast algorithm.

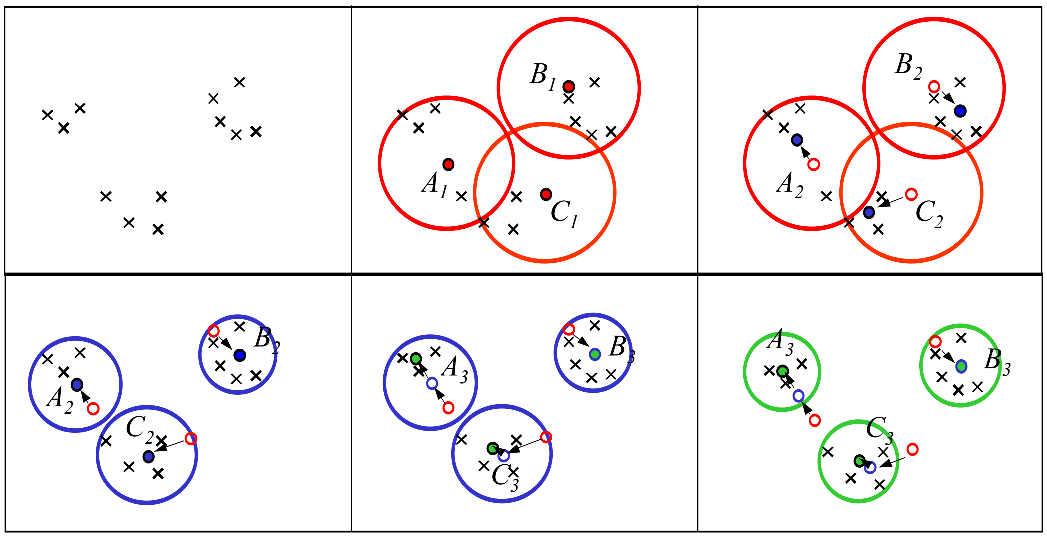

Figure 1: Successive stages of the k-Means algorithm

Intuitively at least then, the solution supplied by k-Means indeed groups together genes with the most similar behaviors.

Mathematically, k-Means is usually presented as supplying clusters with minimal total variance, i.e., a method that *minimizes* the sum of squares of distances between the data points and their associated centroids. The criterion of minimal total variance yields the most "compact" clusters, an intuitively appealing criterion of quality of a solution. However, as we show below, there is a "grain of salt" to this, as k-Means is only a *very* locally optimal technique.

2.3 The Caveat

The solution obtained in Figure 1 is at least intuitively the "right" one, as the three groups are well separated and of low variance (the profiles within each group are close to one another). However, recall that k-Means terminates as soon as a stable configuration is reached. Depending on the initial positions of the centroids, this may lead to very different solutions being supplied by the algorithm. Indeed, for the twelve profiles in Figure 1, there are at least seven different k-Means solutions in three groups, depicted in Figure

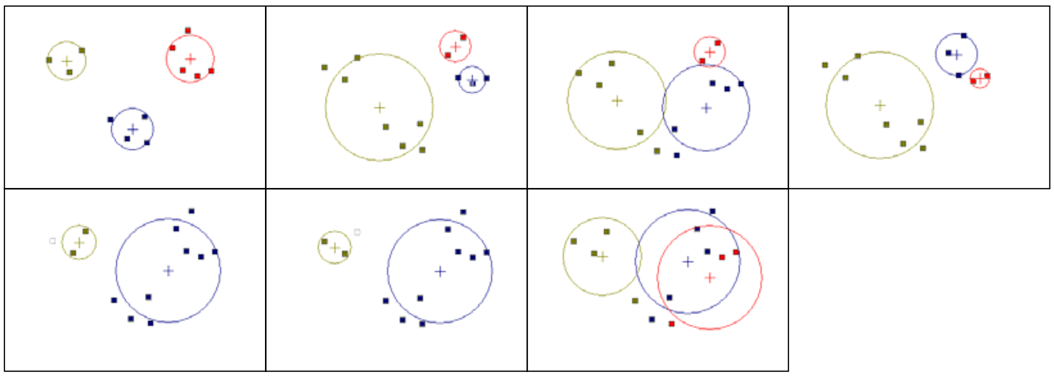

Figure 2: Seven k-Means partitions of the data in Figure 1

2[1]. They have been obtained by starting the algorithm with different initial positions of the centroids. The last of the seven solutions is especially noteworthy, as the members of the intuitively correct group C of Figure 1 are dispersed over three different groups there. Clearly, k-Means can supply solutions of wide diversity. Some, like the one in Figure 1, are intuitively appealing, while others, like the last one in Figure 2, are much less so.

The sensitivity of k-Means to the choice of the initial positions of the centroids is a well known problem. Consequently, in order to limit the chances of landing in a suboptimal solution, several techniques have been proposed for selecting the "best" initial positions of the centroids, with the aim of inducing a high-quality solution at the end of the k-Means run. However, that approach can only be successful if the centroids are initially placed near the centers of the clusters that are unknown at the start of the procedure, and such an *a priori* determination of the positions of the clusters is difficult.

Given that the high speed of the algorithm allows for numerous restarts within a reasonable time, the standard solution to this caveat is running k-Means several times from different initial positions of the centroids and retaining the best clustering found, i.e., the one with the smallest total variance. The various initial positions are typically selected at random.

Supposing k-Means is a robust technique supplying a reasonably small number of different solutions, that approach should yield the best clustering in an acceptably short computation time. In order to assess whether such a way of proceeding with k-Means is practically feasible, we have considered the set of data depicted in Figure 3. This data set is again easily handled visually, but it is closer in size to the amounts of expression profile data usually processed in real world. The example was generated by a juxtaposition of 21 gaussian distributions, a property easily verified with the bare eye. Consequently, we looked at the number of different solutions in 21 clusters that k-Means could supply on those data.

In order to determine the number of different k-Means solutions, one would in principle be required to start the algorithm from every possible initial configuration of the centroids. However, there are infinitely many of them, so we restricted the experiment, considering only the configurations where each centroid is initially placed in one of the data points. Unfortunately, that restriction still leaves too many initial configurations to be tested, as there are more than $(2500\text{-}21)^{21} / 21! > 10^{51}$ ways of selecting 21 items among the 2500 in the figure. We have therefore generated 10000 initial configurations by placing the centroids in data points selected at random (without replacement). The outcome of the test was quite spectacular, since starting from the 10000 initial positions of the centroids, k-Means supplied a staggering 9874 different solutions. The solutions are not reproduced here, for obvious space reasons. However, they can be accessed at the URL given below.

[1] The radii of the circles in Figure 2 are proportional to the standard deviation.

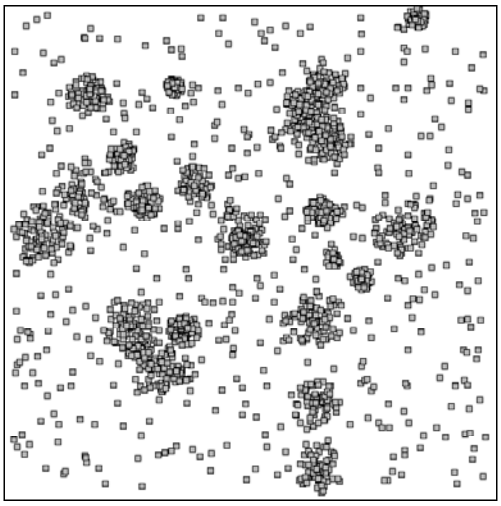

Figure 3: A more realistic 2D data set

The experiment illustrates that k-Means is *very* far from being a robust technique. Given that very few of the currently available software would run the procedure thousands of times, this implies that the solutions obtained with most current software using k-Means have in fact little chance of being of high quality.

3. The ArrayMiner Software

3.1 The Technique Used

The popularity of the k-Means method is witness to the fact that the minimal variance criterion for measuring clustering quality is widely adopted. Consequently, the current release of the ArrayMiner software follows that criterion, i.e., it searches for solutions of minimal total variance in a given number of clusters. However, unlike most other currently available methods, ArrayMiner does not rely on k-Means in its search for high-quality solutions. Rather, it is based on the technique of Grouping Genetic Algorithms [1]. Obvious space constraints do not allow us to explain here the details of the Grouping Genetic Algorithm (GGA), the interested reader should refer to the Falkenauer book. Nevertheless, let us at least explain the rationale behind the method.

Genetic Algorithms (GAs) [3] are an optimization technique inspired by the processes of evolution of species in Nature. A GA proceeds by promoting high-quality *parts* of solutions in a process of inheritance when creating new solutions to the optimization problem. The parts are combined together to produce novel complete solutions of high quality with high probability. Starting with an initial pool of solutions generated largely at random, a GA iteratively improves the pool by combining the best members of the pool, yielding solutions of increasing quality.

The originality of the GGA is that it promotes *groups* of items during inheritance. Since the GGA's groups naturally correspond to the clusters of expression profiles, this proves to be an extremely effective technique for finding high-quality clusters in the expression profile data. Indeed, a good clustering of the data necessarily consists of at least some clusters fitting well the structure of the data, i.e., clusters of low variance, well separated from the other data points. Promoting such clusters during inheritance thus improves the overall fit of the clusters to the data, yielding solution of low total variance.

3.2 Why is ArrayMiner Different

ArrayMiner thus stands out in comparison with other currently available software for expression profile clustering in that it uses a well-defined *optimization* process (the GGA) in search of the best possible clustering solution, using a well-defined measure of quality (the total variance of the clusters, being minimized). In contrast, most other software uses a *heuristic* one-shot procedure (k-Means or other) in hopes of arriving at a good solution to the problem. If the heuristic fails and produces a mediocre solution, then that's what the user will have to do with. That this probably happens frequently is nicely illustrated by the fact that *none* of the currently available software we know of dares to report a measure of quality of the clusters they supply, such as the actual value of the total variance of the clusters.

3.3 Why does it Matter in Practice

The use of a sophisticated optimization algorithm in ArrayMiner is all-important. Indeed, it can be shown that finding a set of clusters with the lowest total variance is in fact a very difficult problem (it is NP-hard, [2]), which implies that fast one-shot heuristics simply cannot be expected to perform well. As a result, they will supply suboptimal solutions with high probability. Indeed, as we have shown above, k-Means is *extremely* prone to performing below expectations.

For the biologist who runs the clustering software, the quality of the clustering is of significant importance, as he or she interprets the clusters as associations of genes that behave similarly. Since the detailed phenomena underlying the observed expression profiles are often yet to be discovered, the fact that some genes are associated while others are not will typically prompt the biologist to examine hypotheses on *why* the genes associated in a cluster would behave similarly, and differently from the genes outside the cluster.

Suppose now that the clusters supplied to the biologist are of poor quality, such as in the last part of Figure 2. Considering such a solution will lead the biologist into a painful examination (and, hopefully, rejection) of hypotheses purportedly explaining the *bogus* associations suggested by the ill-formed clusters in that solution. Indeed, the genes in those clusters do *not* in fact behave similarly, they have just been mistakenly associated due to the poor performance of the clustering algorithm. Conversely, a poor solution obviously means that better ones are not supplied. It thus prevents the biologist from examining the probably more useful associations, such as the ones in the first part of Figure 1. This may cause the biologist to *miss* important biological phenomena, a potentially serious hindrance on their research.

3.4 ArrayMiner's Performance

3.4.1 Solution Quality and Speed

In order to be of real use to biologists, the solutions supplied by a clustering algorithm must be of high quality, which is why ArrayMiner exploits an algorithm performing a true optimization of the clustering. However, in order to be practically usable, an algorithm must also be reasonably fast. On difficult problems like the one discussed here, the challenge is to design an algorithm supplying high-quality solutions on short notice.

In this respect, the GGA incorporated in ArrayMiner fares very well. As an illustration, consider the data in Figure 3. Running the bare k-Means procedure 10000 times took about 35 minutes on a PIII-800 machine. The best among those 10000 solutions was found just once in those 10000 runs of k-Means, illustrating the high instability of the k-Means algorithm. Yet ArrayMiner's GGA found that same solution in just 4 minutes on average (see below).

3.4.2 Reliability

If the biologist is to have any confidence in the clusters of genes supplied by a clustering tool, the tool must *consistently* deliver high-quality solutions. Indeed, if it does not, then any given solution supplied by the tool can be considered as a largely *random* pick among numerous different solutions. In that case, the biologist would have good reasons to question why the one proposed should be worthy of extensive scrutiny, rather than any other.

As we have seen above, the bare k-Means algorithm fares extremely poorly in this respect: it found the best solution to the simple problem in Figure 3 only once in 10000 trials. In other words, even on that simple example, the bare k-Means is about ten thousands times likelier to supply the *wrong* clusters than to supply the correct ones.

In order to assess the reliability of ArrayMiner, we have run it 20 times on the example, starting with different initial conditions. In those 20 runs, ArrayMiner found the best of the 10000-trials solutions *every* time. Finding that one solution[2] on numerous runs started from different initial conditions suggests that it is probably the globally optimal solution to the problem, yet ArrayMiner was able to find it in very reasonable time (row RND2500 in Table 1).

The two-dimensional example in Figure 3 is actually quite easy. Testing ArrayMiner on real gene expression data further demonstrates its advantage over simple k-Means. The following additional data sets, all clustered into 10 groups, have their corresponding entries in Table 1:

- YDSHIFT: 1446 profiles of the yeast diauxic shift (7 time points)

[2] Equality based on identical cluster memberships, rather than a mere equality of total variance.

Table 1: ArrayMiner Performance

Data set	10000 k-Means Trials			ArrayMiner		
	10kTime	10kNbrSols	10kBest	TimeTo10k	Best	TimeToBest
RND2500	35	9874	2751149	4	2751149	4
YDSHIFT	39	9650	21.4911	2	21.4875	8
RAT	1.5	9982	1.8527	0.04	1.8343	0.09
YCELL	14	10000	32.3130	6	32.2871	17

10kTime: Time to perform 10000 trials of k-Means; 10kNbrSols: Number of distinct solutions in 10000 trials; 10kBest: Total variance of the best k-Means trial; TimeTo10k: Average time over 20 successive runs of ArrayMiner to find a solution as good as or better than the best found in 10000 trials of k-Means; Best: Total variance of ArrayMiner's best solution; TimeToBest: Average time over 20 successive runs of ArrayMiner to find the best solution. All times in minutes on a PIII-800/Win2000.

- RAT: the LNP/NINDS/NIH set of 112 profiles of rat nervous system development (9 time points) available in the demo version of GeneSpring™ software by Silicon Genetics
- YCELL: the "ACGCGT in all ORFs" set of 507 profiles (16 time points) of the yeast cell cycle available in the demo version of GeneSpring™

Table 1 deserves several comments. K-Means produced well in excess of 9500 different solutions when started from 10000 initial configurations, in every case. Yet except for the easy RND2500, it never found the optimal solution. ArrayMiner was significantly faster in finding solutions of equal or better quality. Most importantly though, since the best solution found by ArrayMiner was *identical* in each of the 20 successive runs on each of the data sets, those solutions are probably the global optima. ArrayMiner thus *consistently* found the probable best clustering in each case, and it did so within very reasonable execution times.

4. Conclusions

Clustering of gene expression data into clusters of minimal variance is a difficult problem on which fast one-shot heuristics cannot be expected to perform adequately. We have shown that k-Means, one of the most popular techniques for tackling the problem, is a very unreliable method, extremely prone to supplying solutions of low quality. Consequently, biologists using k-Means may expect to work with bogus groups of genes that are in fact not coexpressed on the one hand, and miss important groups of coexpressed genes on the other. We have then presented the ArrayMiner software that uses a sophisticated optimization technique in searching for the best possible clusters. Experimental results show that ArrayMiner supplies excellent solutions with very high reliability, and it does so within reasonably short execution times.

Further information on ArrayMiner is available at http://www.optimaldesign.com.

5. References

[1] Falkenauer E. Genetic Algorithms and Grouping Problems. Wiley, 1998.

[2] Garey M. R. and Johnson D. S. Computers and Intractability - A Guide to the Theory of NP-completeness. Freeman, 1979.

[3] Holland J. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, 1975.

Post-publication notes:

ArrayMiner is also available as optional add-on to GeneSpring™ by Silicon Genetics, at http://www.sigenetics.com/cgi/SiG.cgi/Products/GeneSpring/Programs/arrayminer.smf.