Order Preserving Clustering over Multiple Time Course Experiments

Stefan Bleuler* and Eckart Zitzler

Swiss Federal Institute of Technology Zurich,
Computer Engineering and Networks Laboratory (TIK),
Gloriastrasse 35, CH–8092 Zürich, Switzerland
{bleuler, zitzler}@tik.ee.ethz.ch

Abstract. Clustering still represents the most commonly used technique to analyze gene expression data—be it classical clustering approaches that aim at finding biologically relevant gene groups or biclustering methods that focus on identifying subset of genes that behave similarly over a subset of conditions. Usually, the measurements of different experiments are mixed together in a single gene expression matrix, where the information about which experiments belong together, e.g., in the context of a time course, is lost. This paper investigates the question of how to exploit the information about related experiments and to effectively use it in the clustering process. To this end, the idea of order preserving clusters that has been presented in [2] is extended and integrated in an evolutionary algorithm framework that allows simultaneous clustering over multiple time course experiments while keeping the distinct time series data separate.

1 Motivation

A central goal in the analysis of genome wide gene expression measurements is to identify groups of genes with shared functions or shared regulatory mechanisms. To this end different clustering concepts have been developed. Standard clustering methods such as k-means, hierarchical clustering [6], self organizing maps [11], partition the set of genes into disjoint groups according to the similarity of their expression patterns over *all* conditions. Thereby, they may fail to uncover processes that are active only over some but not all conditions. In contrast, biclustering aims at finding subsets of genes which are similarly expressed over a *subset* of conditions, which often better reflects biological reality. The usefulness of this concept in the context of microarray measurements has been demonstrated in different studies [5, 12, 10].

A promising biclustering approach, which is especially useful in the context of time course experiments, is the order preserving submatrix (OPSM) method

* Stefan Bleuler has been supported by the SEP program at ETH Zürich under the poly project TH-8/02-2.

F. Rothlauf et al. (Eds.): EvoWorkshops 2005, LNCS 3449, pp. 33–43, 2005.

by Ben-Dor et al. [2]. This method concerns the discovery of one or several submatrices in a gene expression matrix in the which the expression levels of the selected genes induce the same linear ordering of the selected experiments. However, there are several potential drawbacks of this approach: the algorithm (i) does not allow to relax the clustering criterion, i.e., deviations from the perfect linear ordering will be not considered as clusters, (ii) needs excessive computing resources if applied to large gene expression matrices, and (iii) does not provide means to keep different types of experiments separate from each other. The first issue has been addressed in [7] by assigning similar expression levels equal ranks but still searching for perfect linear orderings. The third issue, which applies to most of the existing clustering approaches, is important insofar as the mixture of different types of experiments on the one hand side may blur the clustering outcome and on the other hand does not allow to study similarities and differences between the distinct experiment groups. Therefore in this paper, we

- propose a cluster scoring scheme that represents a relaxation of the strict order preserving criterion introduced in [2],
- present an evolutionary algorithm for clustering that uses the above scoring scheme, allows to treat strongly related experiments such as time series separately, and can be applied to arbitrarily large data sets, and
- demonstrate the usefulness of the suggested approach on various data sets for *Arabidopsis thaliana*.

The proposed method can generally be applied to combine data sets from different experiment groups which cannot be compared directly. It even is possible to use separate scores for measuring similarity in the different groups. In this paper, though, we focus on the analysis of time courses as an example for such data sets. This approach is conceptually different from other methods for the analysis of time course data as it focuses on the relation between different time courses while most other methods focus on the relation between the different time points within a single experiment, cf. [1].

In the following, we will first discuss the underlying clustering concept, before a corresponding implementation of an evolutionary algorithm is presented. Later, the proposed method is compared with the OPSM approach on various time series data, and especially the issue of mixed and separated time courses is investigated. The last section summarizes the main results of the study.

2 Order Preserving Clustering

In the remainder of this paper, we will assume that the measurements of several experiments are given in terms of an $m \times n$-matrix, E, where m is the number of considered genes and n the number of experiments. A cell e_{ij} of E contains a real value that reflects the abundance of mRNA for gene i under condition j relative to a control experiment (absolute mRNA concentrations are seldom used). A time course stands for a sequence of measurements that have been performed at different points in time under the same condition for the same organism. Such

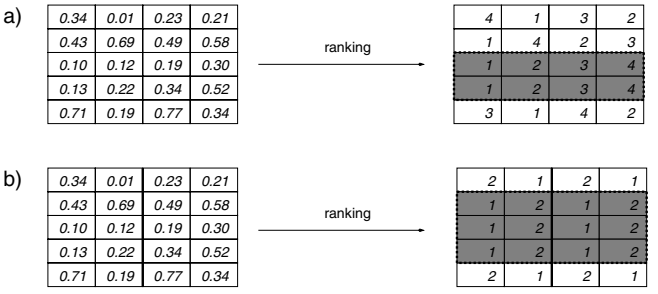

Fig. 1. (a) On the left hand side, a gene expression matrix is shown, on the right hand side, the corresponding expression levels are replaced by their ranks within each row; the shaded area marks the largest order preserving submatrix. (b) The same gene expression matrix as in a) is now divided into two time courses, each consisting of two experiments; the resulting ranking induces a larger order preserving submatrix compared to a), as each time course is treated separately

a time course measurement is represented by a gene expression matrix E where each column stands for one point in time and where the order of the columns reflects the order of the experiments in time.

2.1 The Order Preserving Submatrix Problem

Given the above notation, the order preserving submatrix problem, which has been introduced by Ben-Dor et al. [2], can be described as follows: find a subset G of genes ($|G| \leq m$) and a subset C of experiments ($|C| \leq n$) such that the submatrix D of E defined by G and C maximizes a given score $f(G, C)$ and is an order preserving submatrix (OPSM), see below. The score f reflects the probability of observing an OPSM of size $|G| \cdot |C|$ in a randomly chosen matrix of the same dimensions as E. A submatrix D is order preserving if there is a permutation of its columns (experiments) such that the sequence of (gene expression) values for each row (gene) is strictly increasing; this concept is illustrated in Fig. 1a.

A potential problem with the approach presented in [2] is that the running time of the proposed algorithm can increase considerably with the size of E. Another drawback with the OPSM concept is that the homogeneity criterion (perfect order) is a hard constraint. However, if slight deviations from a perfect ordering would be allowed, this often may better reflect biological reality.

2.2 A Score for the Degree of Order Preservation

The first question we address is how to relax the condition of order preservation such that slight disagreements between the genes are still acceptable. If the allowed error is adjustable by the user, it is possible to account for errors in the measurements and to adapt the cluster criterion to the current biological data set.

Several measures to quantify the unsortedness of a sequence of integers have been suggested in the literature, see, e.g., [9]. One potential measure is to compare all possible pairs of the sequence elements and count the number of pairs that appear in the wrong order. When we extend this concept to a submatrix, one could consider the total number of mismatches over all genes. However, the corresponding number strongly depends on the actual order of the selected columns and finding the order that minimizes this scores is itself an NP-hard problem [8]. Therefore, we propose a scoring scheme that is independent of the actual order of the columns; it is based on another biclustering approach proposed by Cheng and Church [5] that uses the mean squared residue score.

The optimization task in [5] is to find the largest bicluster that does not exceed a certain homogeneity constraint. The size $f(G, C)$ is simply defined as the number of cells in E covered by a bicluster (G, C), while the homogeneity $g(G, C)$ is given by the *mean squared residue score*. Formally, the problem is to maximize $f(G, C) = |G| \cdot |C|$ subject to $g(G, C) \leq \delta$ with $(G, C) \in X$, and where

$$g(G, C) = \frac{1}{|G||C|} \sum_{i \in G, j \in C} (e_{ij} - e_{iC} - e_{Gj} + e_{GC})^2$$

is called the mean squared residue score and

$$e_{iC} = \frac{1}{|C|} \sum_{j \in C} e_{ij}, \; e_{Gj} = \frac{1}{|G|} \sum_{i \in G} e_{ij}, \; e_{GC} = \frac{1}{|G||C|} \sum_{i \in G, j \in C} e_{ij}$$

denote the mean column and row expression values for (G, C) and the mean over all cells, respectively. The threshold δ needs to be set by the user and defines the maximum allowable dissimilarity within the cells of a bicluster. Roughly speaking, the residue expresses how well the value of an element in the bicluster is determined by the column and row it lies in. A set of genes whose expression levels change in accordance to each other over a set of conditions can thus form a perfect bicluster even if the actual values lie far apart.

In our scenario, we use a scoring function h which combines the mean squared residue score with the OPSM concept. Given a submatrix D, we first rank the values in D per row and then replace the expression values with their ranks; afterwards, we apply the mean squared residue score to the transformed submatrix D' in order to assign a score to D. Since the ranks in each row of D' sum up to the same value, the row mean e_{iC} and the total mean e_{GC} cancel each other out and the scoring scheme is reduced to measure the unsortedness of the column means of the ranks. It can be easily shown that a score of 0 using this modified scheme h is equivalent to D being an OPSM. Additional tests with small random matrices showed that h correlates well with the abovementioned count of unordered pairs.

2.3 Clustering Scores for Time Course Data

This paper focuses on time series data, although the presented concepts can be used for other types of experiments as well. As time series often consist of

a few experiments only (usually 6 to 10), in the following we will consider the optimization task to find the largest subset G of genes that has a score $h(G) \leq \delta$, where δ is a constraint set by the user and all experiments in E are considered, i.e., the submatrix D extends over the rows of E specified by G and all columns of E. As we will show later in Section 4.2, this restriction is reasonable if n is small.

The situation changes, though, if multiple time series data are taken into account. The common way is to combine several time courses into a single matrix; however, thereby information is lost and the resulting OPSM can be small. Here, we propose to treat each of the time courses separately as depicted in Fig. 1b. We still aim at maximizing the number of genes in the cluster, but the constraint on h is now computed for each time series separately. That is, given G, for each time course the resulting submatrix D is computed and it is checked whether the corresponding h score is lower than or equal to the constraint δ; only if the score constraint is fulfilled for all time series, the cluster G is considered a feasible solution. In the next section, we present an EA implementation for this problem.

3 Evolutionary Algorithm

The main idea is to use an evolutionary algorithm to explore the search space of possible gene sets systematically. As we will see, the representation and most of the operators are generic while the local search procedure and the environmental selection are more specific to the proposed optimization problem.

Each individual encodes one cluster. For reasons of simplicity we have chosen to use a binary representation with a bit string of length m where a bit is set to 1 if the corresponding gene is included in the cluster. We apply uniform crossover and independent bit mutation. As to environmental selection, a special diversity maintenance mechanism was used which is described later in this section. For mating selection, a tournament selection is used.

Since the objective is to find large clusters the fitness of an individual is calculated as the inverse of the number of genes included in the cluster which leads to a minimization problem. The threshold on h is used as constraint and a local search is performed before the fitness assignment which produces only feasible solutions.

Local Search. In order to increase the efficiency of the EA a local search procedure is applied to each individual before evaluation. This procedure is based on a greedy heuristic which tries to reduce h while keeping a maximum number genes in the cluster. The algorithm is similar to the one proposed in [5] and consists of two main steps: First genes are removed from the cluster until the homogeneity constraints $h(G) < \delta$ for all time courses are met and in a second step all genes which can be added without increasing $h(G)$ are included in the cluster. This procedure guarantees to produce a feasible solution because in the extreme case a cluster is reduced to a single gene which always has perfect homogeneity ($h = 0$).

```
while number of genes > threshold do
  calculate e_Gj for all columns j for all time
  courses
  calculate h for all time courses
  for all genes in the cluster do
    calculate d for all time courses separately:
    d = 1/m Σ_{j=1..m} (e_ij − e_Gj)^2
    if d > αh for any time course then
      remove the gene
    end if
  end for
  if nothing was removed then
    switch to Single Gene Deletion
  end if
end while
```

Fig. 2. Algorithm for Multiple Gene Deletion

```
while constraint violated for any time course
do
  calculate e_Gj for all columns j for all time
  courses
  for all genes in the cluster do
    calculate d for all time courses separately:
    d = 1/m Σ_{j=1..m} (e_ij − e_Gj)^2
    sum up the d for all time courses into s
  end for
  remove gene with maximal s
end while
continue with Gene Addition
```

Fig. 3. Algorithm for Single Gene Deletion

The removal and addition of the genes is generally done one by one in a greedy fashion which requires to update the homogeneity score h after the addition or removal of each gene. If many genes need to be removed this recalculation can increase the running time heavily. To prevent this multiple genes are removed in each iteration while the number of genes in the cluster is above a certain threshold (default = 100). The complete local search thus consists of the three steps described in Figures 2, 3 and 5.

Additionally, it is necessary to specify what happens with the result of the local search. In this study, we use Lamarckian evolution which means that the solution found by the local search is kept and replaces the individual the local search started from as opposed to Baldwinian evolution where the locally optimized solution is just used to calculate the fitness of the individual.

Diversity Maintenance. As a whole population of clusters is evolved simultaneously it is possible not only to optimize one cluster but also to find a set of clusters which fulfill a desired property like total coverage or minimum overlap. To this end a special diversity maintenance mechanism is used. The general idea is to select the biggest cluster first and in each following step the cluster which adds most to the coverage of the set of all genes. The algorithm is described in detail in [3].

4 Results

In the simulation runs mainly two questions were investigated: (i) How does the EA compare to the OPSM algorithm proposed in [2] when applied to find perfectly ordered clusters, and (ii) what is the effect of separating the different time series?

4.1 Data Preparation and Experimental Setup

All simulations were performed on gene expression data generated with Affymetrix GeneChips from *Arabidosis thaliana*, a small plant. The data set

Table 1. Default parameter settings for this study

α	1.2
probability of 1 in initalization	0.001
mutation rate	0.001
crossover rate	0.1
tournament size	3
population size	100
number of generations	100

used in this study was provided by the ATGenExpress consortium [1] and used to investigate the response of Arabidopsis to different kinds of stresses. It consists of 8 time series with 6 time points each. The total expression matrix thus contains 22746 genes and 48 chips. All expression values were preprocessed using RMA [4] and the logratios with the measurement from an untreated control plant were calculated.

The EA parameter settings used in the following simulations are described in Table 1. The crossover rate refers to the percentage of parents involved in crossover. The mutation rate is the probability for bit flips in the independent bit mutation.

The EA was programmed in C++ while the OPSM algorithm was implemented in Java. The implementation closely follows the description in [2]. The OPSM algorithm takes a parameter l describing how many candidate solutions should be further investigated during the greedy search for OPSMs, see [2] for the details. Consistent with the value used in [2] we set l to 100.

All simulation runs were performed on a 3 GHz Intel Xeon machine. For each run, 30 replicates with different random number generator seeds were performed.

4.2 Single Time Series

As mentioned above, searching for perfect OPSMs which extend over all chips in the data set corresponds to setting the constraint on the inhomogeneity h to zero. It is thus interesting to compare the clusters found by the EA to the ones found by the OPSM algorithm. To this end we ran both algorithms on six time course data sets. The largest cluster found by the EA equaled the one found by the OPSM algorithm in all cases with sizes ranging from 290 to 662 genes. Often this cluster was found after only a few generations.

While the OPSM algorithm is tailored to find one maximal OPSM for each number of chips the EA can find several clusters in one run. Without the diversity mechanism described in Section 3 the population quickly converges to a set of clusters with large overlap with the best cluster. The diversity mechanism prevents this: for one data set, as an example, all 100 clusters in the final population were non overlapping and consisted of an average of 90 genes. These groups have different orderings of the expression levels. It makes sense to investigate these clusters as well and not just concentrate on finding the biggest OPSM.

[1] See http://web.uni-frankfurt.de/fb15/botanik/mcb/AFGN/atgenex.htm

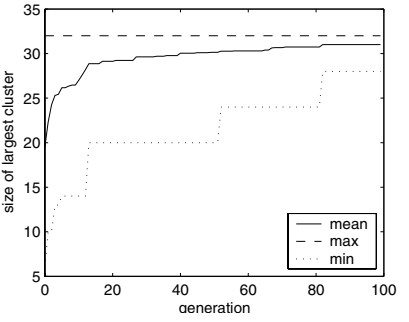

Fig. 4. Size of the largest cluster found by the EA on the two "cold" data sets. Mean, max and min over 30 runs

while a gene was added in the last iteration **do**
  calculate e_{Gj} for all columns j for all time courses
  calculate h for all time courses
  for all genes not in the cluster **do**
    calculate $d = \frac{1}{m} \sum_{j=1..m} (e_{ij} - e_{Gj})^2$
    if $d < h$ for all time courses **then**
      add gene
      recalculate all e_{Gj} and h
    end if
  end for
end while

Fig. 5. Algorithm for Gene Addition

The high number of large clusters found by the EA makes it unnecessary to relax the constraint in the case of such small data sets. The effect of relaxing the constraint will be discussed in the following section where several time courses are combined into one data set.

4.3 Multiple Time Series

In a second set of experiments we investigated the effect of keeping the time courses separated during the search for order preserving clusters. To this end we first combined two data sets and then tested the algorithms on the combination of all eight data sets.

Two Time Series. When comparing the performance of the OPSM algorithm to the EA on the combination of both data sets in same sense as in the previous section, both algorithms found OPSMs consisting of two genes and all twelve chips. When relaxing the constraint for the EA, larger clusters can be found; for instance, if the constraint on h is set to 0.5 which means that the average difference between the actual rank and the column mean of the ranks must be smaller than 0.5 the largest cluster found by the EA contained 6 genes (mean over 30 runs).

We then used the EA to search for groups of genes which fulfill the homogeneity constraint of $h = 0$ on both data sets separately. As shown in Figure 4 the largest cluster found by the EA contained 31 genes. Additionally the final population contained 100 clusters with an average size of 7 genes and no overlap between them. It is obvious that many significant clusters are missed if the data sets are mixed and the proposed EA is able to retrieve many of them.

Eight Time Series. When applying the algorithms to the total of 48 chips another drawback of the OPSM algorithm becomes apparent: the running time increases rapidly with the number of chips in the data set. While the OPSM

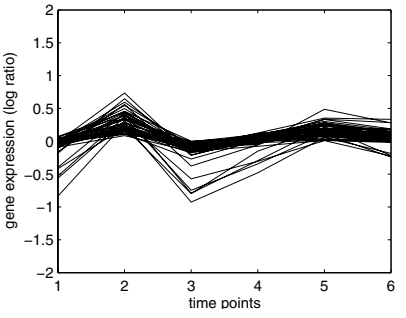

Fig. 6. Identifying differences: Expression values for the well ordered time course

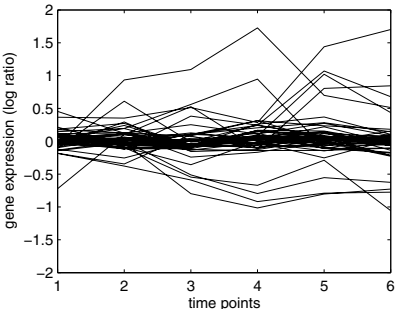

Fig. 7. Identifying differences: Expression values for one of the unordered time courses

algorithm takes 30 seconds to run on a data set of six chips is takes more than two hours to finish on the full data set. The EA run time lies between 10 and 20 mins. For the EA, however, the running time can be adjusted by changing the number of generations and the population size. The EA is thus still applicable for large data sets which cannot be analyzed using the OPSM algorithm.

As expected neither OPSM nor EA found perfect cluster over all 48 chips. Also for the separated data sets no cluster was found which had a perfect ordering for all time series. Relaxing the constraint to 1 for each time course allowed the EA to find clusters of a maximal size of 9 genes. Higher constraints could be used to find larger but less coherent clusters. Also for the combined data set large clusters can be found but the constraint must be relaxed to about 25 which does not provide a good ordering anymore.

4.4 Identifying Differences in Multiple Time Series

Another problem which is of high biological relevance is to identify groups of genes that behave similarly in some of the time courses and exhibit inhomogeneous expression patterns in other time courses. There are different possibilities to include these objectives into the EA depending on the biological scenario. We chose the following: the expression values for at least one time course must be ordered, i. e., h must be zero. Under this constraint the maximum h over all time series is maximized. The local search procedure was changed accordingly so that only the constraint of the best ordered time course must be fulfilled.

We tested this method on the combination of all eight data sets. In the resulting clusters in general only the expression levels of one time course are well ordered. Figures 6 and 7 show one example where time course on the left hand side is well ordered and the one on the right hand side stands as example for all the others which are not well ordered.

This shows that the problem of finding differences in multiple time series can be address using the proposed approach, however, further investigations and biological analysis of the results are necessary.

5 Conclusions

The order preserving submatrix problem as defined in [2] consists of finding large submatrices in a gene expression matrix such that for each submatrix there is a permutation of the columns under which the sequence of expression values is for each row strictly increasing. We have extended this approach in three aspects:

- A more flexible scoring scheme was proposed that allows to arbitrarily scale the degree of orderedness required for a cluster.
- Based on this scoring scheme, a methodology to handle different set of experiments such as distinct time series in a systematic way was introduced.
- An evolutionary algorithm for this order preserving problem was presented that is capable of finding multiple non-overlapping clusters in a single optimization run.

The effectiveness of the suggested approach was demonstrated on eight time series experiments covering overall 48 measurements for about 20000 genes of *Arabidopsis thaliana*. Especially, the separation of different time series experiments has proven to be a valuable concept: the cluster sizes could be improved substantially in comparison to the common approach where all time series experiments are combined in a single gene expression matrix.

References

1. Z. Bar-Joseph. Analizing time series gene expression data. *Bioinformatics*, 20(16):2493–2503, 2004.
2. A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. In *Conference on Computational Biology (RECOMB 02)*, pages 49–57. ACM Press, 2002.
3. S. Bleuler, A. Prelić, and E. Zitzler. An EA framework for biclustering of gene expression data. In *Congress on Evolutionary Computation (CEC 2004)*, pages 166–173, Piscataway, NJ, 2004. IEEE.
4. B. M. Bolstad, R. A. Irizarry, M. Astrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array based on variance and bias. *Bioinformatics*, 19(2):185–193, Jan. 2003.
5. Y. Cheng and G. M. Church. Biclustering of gene expression data. In *ISMB 2000*, pages 93–103, 2000. http://cheng.ecescs.uc.edu/biclustering.
6. M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, Dec. 1998.
7. J. Liu, J. Yang, and W. Wang. Biclustering in gene expression data by tendency. In *Computational Systems Bioinformatics Conference (CSB 2004)*. IEEE, 2004.
8. G. Reinelt. *The Linear Ordering Problem*. Heldermann, Berlin, 1985.
9. J. Scharnow, K. Tinnefeld, and I. Wegener. Fitness landscapes based on sorting and shortest path problems. In *Parallel Problem Solving from Nature (PPSN VII)*, number 2439 in LNCS, pages 54–63. Springer-Verlag, 2002.

10. E. Segal, A. Battle, and D. Koller. Decomposing gene expresision into cellular processes. In *Pacific Symposium on Biocomputing*, pages 8:89–100, 2003.
11. P. Tamayo et al. Interpreting patterns of gene expression with self-organizing maps: Methods and applicataion to hematopoietic differentiation. *PNAS*, 96:2907 – 2912, March 1999.
12. A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(Suppl. 1):S136–S144, 2002.