MOPAC: MOtif Finding by Preprocessing and Agglomerative Clustering from Microarrays

R. GANESH[1], DEBORAH A. SIEGELE[2] and THOMAS R. IOERGER[1]
Department of Computer Science[1], and Department of Biology[2]
Texas A&M University, College Station, TX 77840, USA

We propose a novel strategy for discovering motifs from gene expression data. The gene expression data in our experiments comes from DNA Microarray analysis of the bacterium *E. coli* in response to recovery from nutrient starvation. We have annotated the data and identified the upregulated genes. Our interest is to find common regulatory motifs that are responsible for the upregulation of these specific genes. We assume that a common motif that a regulatory protein can bind to will be present in the upstream region of the upregulated genes and will not be present in the upstream regions of genes that showed a constant level of expression over time. Our objective is to find the common motifs that are present in at least some of the upstream sequences of upregulated genes and not present in the control set, which is the set of genes whose expression remained the same. Because it is possible that there could be several subsets of co-regulated genes under different control mechanisms among the co-expressed genes, we do not want to require motifs to be present in all upregulated sequences. Therefore, we propose a new algorithm for finding such motifs through stages of pre-processing, de-noising, agglomerative clustering and consensus checking. Through this process, we have found some motifs that are good candidates for further validation.

1 Introduction

Analyzing gene expression data from DNA Microarrays is a well-studied problem. There are several ways that gene expression data can be used, from profiling of genes to inferring gene regulatory networks. Microarray experiments reveal genes that are co-expressed and this is a good starting point to find co-regulated genes among the co-expressed genes. The most common work that has been done in analyzing the microarray data is by using different clustering techniques[1, 5]. We augment these analyses by searching for motifs that are shared by the upstream sequences of the genes that are co-expressed. There could exist short sequences that are shared by these co-regulated genes, which serve as the binding sites for those proteins that initialize transcription of these genes. We are interested in finding motifs from genes that have similar expression profiles. Previous work has been done in this area using probabilistic and Bayesian approaches[13].

Many of the methods that have been proposed to solve this problem are based on local search techniques like Gibbs sampling and Expectation Maximization. Thijis *et al*[20] have classified the computational methods that are used to identify regulatory motifs into two categories, viz. string analysis methods and methods based on probabilistic sequence models. The former method is based on frequency analysis of nucleotides in the upstream sequences of co-expressed genes and the latter develops a probabilistic model as a position-specific probability matrix.

McGuire *et al* [13] have worked on identifying regulatory elements from yeast and *E. coli* using AlignACE, which uses Gibb's sampling algorithm. Hu *et al* [9] have used constructive induction to analyze potential motif combinations. Pevzner and Sze [15] have proposed a challenge problem to find a signal in a sample of sequences each containing an unknown signal of length 15 with 4 mismatches and have come up with two algorithms WINNOWER and SP-STAR by interpreting the problem as a maximum-clique graph problem.

Working with prokaryotes in general is hard, as we have to take into consideration the operon relationships in finding upstream sequences and with respect to sharing the common signal. We cannot expect all upstream sequences to share the common signal and there could be many signals, which means many regulatory proteins can control the expression of a trait. The signals aren't expected to be identical in the genes that share them, and the proteins need just a partial consensus, tolerating some noise to bind to these signals [11]. The motifs are expected to be of variable length and they are typically between 5 and 15 nucleotides [10].

Most of the approaches that have been tried so far either do not use a background model or use a probabilistic background model. A probabilistic model would rule out motifs that have a high probability of occurring elsewhere in a large genome based on product of nucleotide frequencies. We define background empirically based on upstream sequences of other genes that did not show differential regulation, i.e. the genes whose expression profiles remained the same. A good background model would help reduce spurious signals being recognized as motifs. Another approach from Thijs *et al*[21] is INCLUSive, which is an integrated tool for clustering, retrieving upstream sequences and sampling motifs using Gibbs sampling. They have reported that using an organism-dependent background model can enhance the outcome of their motif finder. Another paper from the same authors[19] suggests using a higher-order background model that would update the probabilities of finding a motif at a certain position in the sequence. They have found that overall recovery of the motifs in the presence of a higher-order model has been significantly improved, and also the program better handles noisy data. YEBIS is another tool that was developed based on hidden Markov models using a weight matrix method designed for high-speed computation[24]. There has also been work that has been carried on removing artifacts from real motifs after finding the motifs first using a greedy approach.[3] Use of clustering techniques to solve the DNA motif problem has been rarely attempted. Guralnik and Karypis[7] have tried hierarchical and k means clustering for protein sequences.

In short our approach requires the motif to be present only in a subset of the differentially regulated genes and absent in the genes that do not show a significant level of upregulation or downregulation.

2 Input Processing

2.1 Annotation of Input Data

The gene expression data has signal levels measured for each spot in the Microarray chip. For the nutrient starvation response, signals were recorded from the microarrays for RNA samples of *E. coli* culture in exponential growth phase, starved, 5 minutes and 15 minutes after recovery from starvation[12,18]. We assigned 1 or 0 or –1 as a regulation index for every gene based on whether they show a significant level of upregulation, no change, or downregulation, relative to growth phase (starved, recovery after 5 min, recovery after 15 min). We used 0.03 as a threshold to determine whether the signal is significant compared to the background, based on the signal levels of the genes that were known to be deleted from the genome. We used a threshold of two-fold increase or decrease in signal intensity compared to the exponential phase culture in order to determine whether it is an upregulation or downregulation; most internal variations of the signals of a same gene were less than this threshold. We classified six patterns as upregulation viz. (0,1,1), (0,1,0), (0,0,1), (-1,1,1), (-1,1,0), (-1,0,1) and another six patterns as downregulation viz. (0,-1,-1), (0,-1,0), (0,0,-1), (1,-1,-1), (1,-1,0), (1,0,-1) based on the relationships among the expression levels at the three time points, wherein the three numbers denote the regulation indices at time points 0, 5, and 15 minutes after recovery from starvation. For making the control (non-regulated) set, we grouped the genes that showed a constant level of expression in the three different time intervals viz. (0,0,0), (1,1,1) and (-1,-1,-1).

2.2 Extraction of Upstream Sequences

The next challenge in annotating the data was to extract the upstream nucleotide regions for these genes, since that is where common putative regulatory elements are likely to occur. In order to retrieve the upstream sequences, we needed the coordinates of the genes in the whole genome of *E. coli*. Like most prokaryotes, *E. coli* has operons, wherein more than one gene shares a common upstream sequence, which is located before the start of the first gene in the operon. In order to correctly extract the upstream sequence, we need to know its position in the operon and the upstream sequence of the first gene in the operon. We used the Linkage Map of *E.coli*[2] to determine the operon relationships. We used the NCBI ftp site[25] for information about the orientation and starting point of the ORFs of interest. We used the complete *E.coli* genome from the GOLD[26] database. We used the start of the predicted protein coding sequence as the boundary for extracting upstream sequence. We extracted 600 bp ahead of the start of the gene for upstream region [22]. If the gene were part of an operon, we used 300 bps before the start of the gene and

300 bps before the start of the first gene in operon to suit the total size. We also took the orientation of the gene into consideration in extracting the upstream sequences and we took the reverse complement of the sequences that were known to be transcribed in the counterclockwise direction.

3. Algorithm

An exhaustive approach of screening all potential patterns up to a certain length k by depth first search, especially with wildcards, is computationally intractable. Therefore, we decomposed the problem to a smaller problem of finding motifs of fixed length that are frequent in the upregulated set and not present in the non-regulated set. We can describe the functioning of our algorithm in the following steps.

3.1 Preprocessing

First, we preprocess the experimental (upregulated) set and control (non-regulated) set by extracting all possible sub-sequences which could form motifs. We list all the over-lapping fragments in the experimental set, which are windows of length k nucleotides. If the new motif that we are adding is already in the list and is from a different gene, we increment the count of the motif. If it is a part of an operon, we make an additional check to ensure that we don't increment the count for the same motif from two genes belonging to the same operon. Similarly, we pre-process the control set and, for every motif in control set, we check whether the list of putative motifs contains it and remove them from the list, since we want the motif not to show up the in the control set.

a) Phase I

List = ϕ
For an upstream sequence Z of every gene G_z in the experimental set
    For every subsequence $S_i \in Z$ of length L
        If ($S_i \notin$ List) List = List $\cup$ S_i
        Else Let S_j be the other element in the List that matches S_i
            if ($G_z \in$ operon) and (operon(S_i) $\neq$ operon (S_j))
                Increment count for S_j
            Else if (not ($S_i \in G_z$ and $S_j \in G_z$)
                Increment count for S_j

b) Phase II

For an upstream sequence Z of every gene G_z in the control set
For every subsequence $S_i \in Z$ of length L
If ($S_i \in$ List) List = List – S_i

3.2 De-noising

After pre-processing, there might be a lot of nucleotide patterns that we need to find consensus for. We can easily identify patterns that do not have much similarity to the other patterns in the whole set. It is important to remove these outliers, as they can interfere with the accuracy of the clustering method, and also they add to the complexity of the clustering algorithm. Thus, for every pattern, we compute the ratio of sum of distances of that pattern to other patterns that do not belong to the same gene or operon to the number of such occurrences. The distance between two patterns is scored by the number of mismatches from lexicographic comparison. It gives a negative score for the nucleotides that match and a positive score for every mismatch. The total score is computed by adding the scores of the individual nucleotides. Then we use a threshold, which is based on the distribution of the scores, to exclude the patterns that are far away from the other patterns. The distance is computed by a scoring function that compares two strings lexicographically.

$$Score_p = \frac{\Sigma \text{ distances of S to } S_i \text{ where S and } S_i \text{ do not share the same gene or operon}}{\text{number of such occurences}}$$

3.3 Distance Graph

Next, we compute the similarity matrix of all the patterns we have identified. Our goal is to group the similar ones together and derive a wildcard motif representation for each cluster. We get a dense graph wherein each edge has a weight associated with it, which represents the distance between the two patterns. The distance is computed based on matches and mismatches between any two patterns. The weight associated with the edge between two patterns would be less if the patterns are more similar. This is an undirected graph, since the distance between each pair is the same in either direction.

3.4 Agglomerative Clustering

Now, we want to group the most similar patterns, which we can do by applying a clustering algorithm.[8] We start by coloring the edges in the graph from the lowest cost to highest cost. Initially, we represent every point in the space as a set. For

every edge in our graph, if the sets that each vertex belongs to are different, we make a union of them. If we iterate this we will get a Minimum Spanning Tree[4] (equivalent to single-linkage clustering [6]). However, if we set a threshold called Critical Cost over which we stop coloring edges, we will get a disjointed forest. Critical Cost is the value of the score that represents the similarity between two patterns at which we don't consider the two patterns to be similar enough to go in a single cluster.

```
Compute distance for each edge (pair of patterns)
Sort the edges of the graph in non-decreasing order
For every vertex v
    Do MakeSet(v)
For every edge m = [u_i, v_i]
    If distance(u_i, v_i) < θ (critical cost)
        R_1 = set-of(u_i);
        R_2 = set-of(v_i);
        If (R_1 != R_2) then
            Union(u_i, v_i);
```

3.5 Consensus Checking

The last step is to compute a consensus sequence for each cluster. We represent each pattern as a bit string. The valid nucleotides that can be represented in wild cards would be 2^4 and we keep turning on the bits as we encounter nucleotides that are observed to differ in a position. This approach is similar to the Find-S algorithm[14] for symbolic concept generalization.

```
For every pattern in a cluster
    Represent Pattern as bit string
    Make bitwise disjunction (OR) of its members
```

To construct the consensus, we relax the pattern. Though the individual members of the pattern are guaranteed to be absent in the control set, other instances matching the consensus pattern might show up sometimes. We discard the clusters that show hits in the control sequences with their consensus. We drop the most remote member within a cluster if the average distance to other points is about twice more than the next closest distance not involving that member.

4. Results

After annotating the input data (microarray signals) using the above-mentioned criteria, we identified 22 genes that were upregulated for recovery from nutrient

starvation. We also arrived at a control set by pulling out the genes that did not show an appreciable upregulation or downregulation for this environmental condition. We retrieved the upstream sequences for the upregulated and downregulated genes using the operon relationships. Also, we retrieved the upstream sequences for the entire control set of 1361 genes. We did not use operon information to find upstream sequences from control set because the control set is just used to remove the spurious signals.

Given this well-defined experimental set and control set, we then ran our pre-processing program, which produced many patterns. We then ran the De-noising algorithm with a threshold of 5.87 to remove the outliers and keep the ones that are closely related. After de-noising, we ran the clustering algorithm by creating a similarity graph using a matrix representation and doing agglomerative clustering as explained above. We arrived at an optimum threshold of -6 to stop the clustering based on our expectation of similarity within a cluster (any two patterns can be atmost 25% dissimilar), so that we get a reasonable number of good quality clusters (at least 3 members that belong to different genes/operons). We constructed a consensus string for every cluster. We examined the results and identified the patterns that did not show any hits in the control set. We excluded some motifs whose hits in the control sets were too far from transcription start (more than 450 bp upstream) or downstream of transcription start. Table 1 lists the motifs found and the genes that share the motif, and Figure 2 displays the sequence logo[17] for these motifs.

Table 1: Results of MOPAC – Motifs discovered and their genes
(represented by IUB nucleotide symbols)

ID	Motif	Gene name
1	AAsAAwTTmAwA	CmtB, ygjR, cysD
2	CmwTTkTTyTTC	CysH, B3914, MetR
3	TTCTwHTgAwAT	B1587, MetF, FliY
4	wTVAACwThCAA	B1587, asnB, cysA,**P**,W
5	rAkTTTwTTCAT	B3914, MetR, MetF
6	CAArTwTTTwTr	CmtB, yhaV, cysD
7	ATwAATAATksw	B1587, yhaV, CmtB
8	ACsdTTTTTmTw	CmtB, asnB, b3914, ygjR
9	rAAwTTmATAAT	MetF, CmtB, ygjR
10	vwTTAATAATkC	CmtB, b1587, yhaV, MetF
11	ATwTTGAATTww	AsnB, metR, metF
12	yTTTkhGATATT	YfiA, cysD, fliY
13	AkTTTwTTCATy	B3914, metR, metF

5.Validation and Discussion

We believe that biological experiments such as mutating the motifs and looking for changes in expression would be the best way to validate our results. However, in

this paper, we use some heuristics to evaluate the likelihood of these patterns to be a motif.

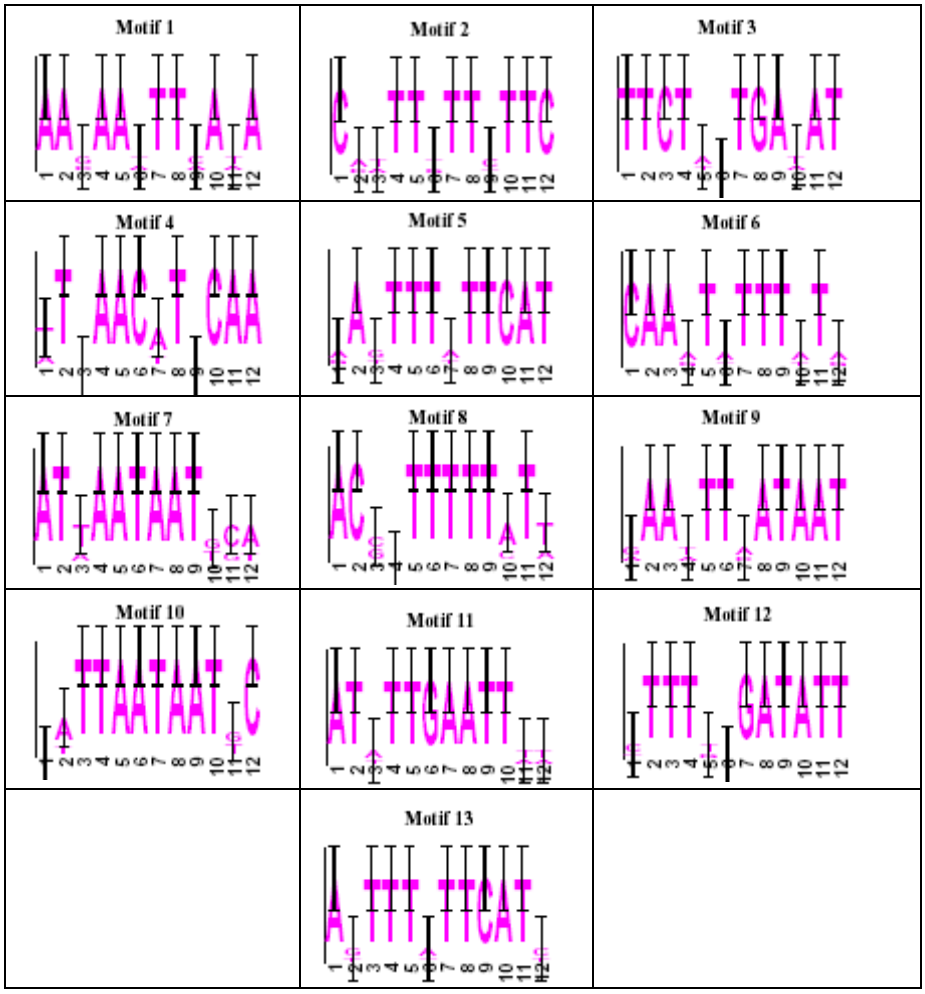

Figure 1: Sequence Logo[17] of motifs discovered using MOPAC

Also, we ran AlignACE[13] and MotifSampler[19] with our input data (upstream sequences of upregulated genes from starvation recovery response). AlignACE tried to find motifs that were shared by almost all the genes. It was difficult to compare the results of AlignACE because it didn't output a consensus, and when we tried to construct one, it seemed too general (with many wildcards, i.e. low specificity).

MotifSampler finds as many motifs as we want with a fixed length and gives the consensus too. The motifs identified by MOPAC were unique.

5.1 Distance from Transcription Start

We found most of the motifs to be close to and upstream of transcription start, which strengthens our belief that the candidate motifs are biologically significant. Since activators are often associated with upregulation, we hypothesize that they tend to bind upstream say between –30 to –300 and the strongest signals could be found within 100 base pairs from the transcription start. We used RegulonDB[16] to determine the transcription start for each gene (most of them were predictions). Figure 2 shows the distribution of position of motifs found using MOPAC.

We calculated the distances to the transcription start site based on the occurrence of the best motif found by AlignACE and MotifSampler. In MOPAC, motifs were typically found between 0-300 bp upstream. The motifs found by MotifSampler were mostly between –200 and –400 and in the case of AlignACE more than 60% of the upstream sequences that share a motif had their motifs concentrated around –300 to –500 bp upstream.

5.2 Palindromicity

Regulatory motifs are often observed to be palindromes[11]. So, we reverse complemented each motif and compared the resultant pattern with our motif and recorded their degree of palindromicity, which is defined as the ratio of the characters that match when the pattern is compared with itself after it is reverse complemented. Table 2 shows that the motifs found using MOPAC have average to high palindromicity.

5.3 Probability

Since the motifs that we have discovered show up in 3 or more genes out of 22 upregulated genes and do not show up at all in the 1361 control genes, we believe that these are significant signals. To quantify this, the probability P of finding a pattern of length n in a sequence of length L, allowing y positions in n to have x number of wildcards, is approximately $P = 1 - [1- \{1/4\}^{n-y}. \{x/4\}^{y}\}]^{L}$. For example, the probability of finding a pattern of length 12 with 4 wildcard positions in regions of size 600 bases with each of them having 2 possible wildcards is 0.000057. The probability of finding the motif in 3 sequences out of 22 sequences is $Q(3,22) = P^{3}.(1-P)^{19}.C^{22}_{3}$ which evaluates to approximately 10^{-10}. We can also see that the probability of finding it in 3 out of 22 sequences would be further reduced. Hence the chance of these motifs to be random occurrences is very minimal.

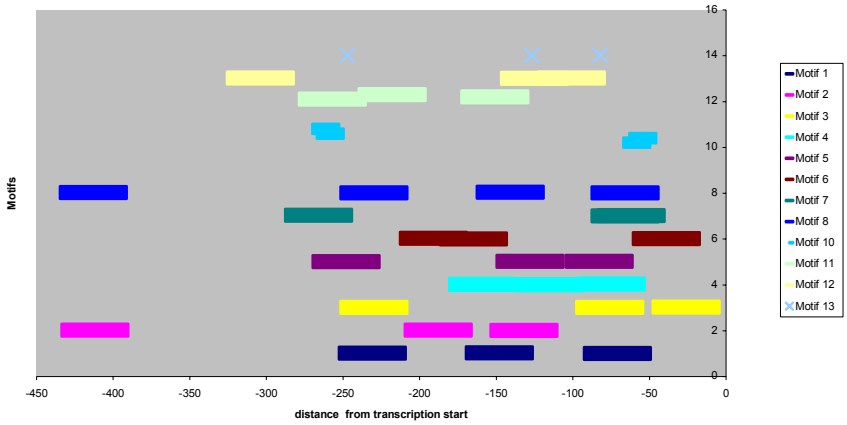

Figure 2: Relative locations of motifs upstream of transcription start [MOPAC]

Table 2: Palindromicity of motifs

MOPAC		MotifSampler	
Motif	Palindromicty	Motif	Palindromicity
1	0.5	1	0.5
2	0.33	2	0.33
3	0.5	3	0.33
4	0.83	4	0.5
5	0.5	5	0.66
6	0.75	6	0.33
7	0.66	7	0.33
8	0.5	8	0.58
9	0.66		
10	0.58		
11	0.66		
12	0.42		
13	0.5		

5.4 Relationship with known transcription factors

Using the prokaryotic part of the TRANSFAC database[23], we searched for sites with 2 or 3 mismatches from our patterns. We got several hits from the database. For example, the motif "CmwTTkTTyTTC" from MOPAC, which is shared by metF, picked up the consensus of the MetJ-MetF site. The other known sites that matched some of the motifs are OmpR-ompC-bc, BlaI-P(p)1, BlaI-P(p)2, ArgR-

carAB arg-box-1, AlgR1-algD, IHF-L1, IHF-L2, cI-Op72a, deoR-deoO(E), CRP-galO, XylR-UBS2, Nod box and CAP/CRP-lac. Many of the motifs predicted by MotifSampler had close homology to the binding site of lambda repressor.

5.5 Hits in the Control Set

The motifs discovered by MOPAC had zero hits in the control set in most cases. In some cases the motifs were still considered in spite of control hits, when the location of the motifs is far upstream or downstream of the transcription start, making them unlikely to be involved in regulation. When we tested for the occurrences of the consensus sequences of MotifSampler, they were found to occur in the control set occasionally. [Table 3] MOPAC, by design does not have any hits in the control set. If we were to make a consensus for AlignACE, it would pick up several hits because of its low specificity. MOPAC and MotifSampler found two different motifs for the genes asnB, metR and metF. The motif found by MOPAC lies between –150 and –250 in all three genes but the one that is found by MotifSampler lies between –300 and –450.

Table 3: Occurrences of motifs in upstream sequences using MotifSampler

Motif	Consensus	Expt hits	Control hits
1	mCGCwkCCGGCr	11	15
2	CGCCrGCGGwrA	10	4
3	GwCGTsnyTGAn	10	8
4	GCnTCTGsTnGs	7	7
5	wsCCGCkGyrCT	6	2
6	CCrCGCmGGAAr	5	2
7	TGTAGGCCGGAT	4	8
8	CGATATCnACCG	3	0

6.Conclusion

We have presented an algorithm, MOPAC that couples analysis of gene expression data with motif prediction in upstream sequences. Our approach is based on the assumption that motifs would be over represented in the genes that are upregulated for an environmental condition and not present in the genes that do not show any significant change in their expression for the same environmental condition. We also do not require the motif to be present in all the co-regulated genes, as some algorithms do. We feel that use of a real (empirically-defined) background sequence yields more biologically meaningful results, by the way it filters out spurious motifs unrelated to the environmental condition. Though our validations were purely computational, the motifs we discovered appear promising. Verifying these signals biologically is beyond the scope of this research. Our algorithm is not as fast as

AlignACE or MotifSampler as it uses an extensive background search and hierarchical clustering, which is slow and memory intensive. This method works only when we have results of a microarray experiment wherein we can clearly identify the genes that are affected and the genes that are not affected for a specific environmental condition. Several improvements can be made in reducing the complexity of the algorithm. Also, it might be better to associate a weight to every nucleotide in the wildcard instead of attaching equal importance when relaxing a pattern.

References

1. A. Ben-Dor and Z. Yakhini, in *RECOMB'99* (1999).
2. M. K. Berlyn, *Microbiology and Molecular Biology Rev*, **62**, 814 (1998).
3. M. Blanchette and S. Sinha in *ISMB 2001* (2001).
4. T.H. Cormen *et al., Introduction to Algorithms* (MIT Press, Cambridge, 1990).
5. M.B.Eisen *et al.,* Proc. Natl. Acad. Sci. USA, **95**, 14863 (1998).
6. B.S. Everitt, *Cluster Analysis* (Heineman Educational Books Ltd., London, 1980).
7. V. Guralnik and G.Karypis in *KDD-2001* (2001).
8. J. Han and M. Kamber, *Datamining: Concepts and Techniques* (Morgan Kaufmann Publishers, San Francisco, 2001).
9. Y. J. Hu *et al.*, *Bioinformatics*, **16(3)**, 222 (2000).
10. S. Keles *et al.*, Bioinformatics **18**, 1167 (2002).
11. B. Lewin, *Genes VII* (Oxford University Press Inc., New York, 2000).
12. J. McEthanon and D.A. Siegele, *Pers. Comm*.
13. A.M. McGuire *et al., Genome Research* **10(6)** 744 (2000).
14. T. M. Mitchell, *Machine Learning* (McGraw-Hill, 1997).
15. P.A. Pevzner and S. –H. Sze in *ISMB'2000* (2000).
16. H. Salgado *et al., Nucleic Acids Res*. **29(1)**,72 (2001).
17. T.D. Schneider and R.M. Stephens, *Nucleic Acids Res.* **18**, 6097 (1990).
18. D.A. Siegele and L.J.Guynn, J. Bacteriology, **178(21)**, 6352 (1996).
19. G. Thijis *et al., Bioinformatics* **17(12)**, 1113 (2001).
20. G. Thijis *et al.,* in *RECOMB'2001* (2001).
21. G. Thijis *et al., Bioinformatics* **18(2)**, 331 (2002).
22. J. van Helden *et al., J. Mol. Biol.* 281, 827 (1998).
23. E. Wingender *et al., Nucleic Acids Res.* **24**, 238 (1996). http://transfac.gbf.de/TRANSFAC/.
24. T. Yada *et al., Bioinformatics* **14(4)**, 317 (1998).
25. ftp://ftp.ncbi.nih.gov/genomes/Bacteria/Escherichia_coli_K12/
26. http://wit.integratedgenomics.com/GOLD/completegenomes.html