JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 16, Number 6, 2009

Pp. 859–873
DOI: 10.1089/cmb.2008.0201

K-Boost: A Scalable Algorithm for High-Quality Clustering of Microarray Gene Expression Data

FILIPPO GERACI,[1] MAURO LEONCINI,[2] MANUELA MONTANGERO,[2]
MARCO PELLEGRINI,[1] and M. ELENA RENDA[1]

ABSTRACT

Microarray technology for profiling gene expression levels is a popular tool in modern biological research. Applications range from tissue classification to the detection of metabolic networks, from drug discovery to time-critical personalized medicine. Given the increase in size and complexity of the data sets produced, their analysis is becoming problematic in terms of time/quality trade-offs. Clustering genes with similar expression profiles is a key initial step for subsequent manipulations and the increasing volumes of data to be analyzed requires methods that are at the same time efficient (completing an analysis in minutes rather than hours) and effective (identifying significant clusters with high biological correlations). In this paper, we propose *K-Boost*, a clustering algorithm based on a combination of the furthest-point-first (FPF) heuristic for solving the metric *k*-center problem, a *stability-based* method for determining the number of clusters, and a *k*-means-like cluster refinement. *K-Boost* runs in $O(|N| \cdot k)$ time, where N is the input matrix and k is the number of proposed clusters. Experiments show that this low complexity is usually coupled with a very good quality of the computed clusterings, which we measure using both internal and external criteria. Supporting data can be found as online Supplementary Material at *www.liebertonline.com.*

Key words: algorithms, gene clusters.

1. INTRODUCTION

Several obstacles still lie in the way of exploiting the full potential of microarray technologies (Trent and Bexevanis, 2002). One issue is the scalability of the data processing software. In particular, a critical phase is often the clustering of gene expression data into groups with homogeneous expression profile. In this article, we tackle this problem by proposing *K-Boost*, a clustering algorithm based on a combination of the furthest-point-first (FPF) heuristic for the *k*-center problem (Gonzalez, 1985), a stability-based (SB) method for determining a plausible number of clusters *k* (Tibshirani et al., 2005), and a single-pass *k*-means (Jain et al., 2000). The experiments we report demonstrate that *K-Boost* is scalable to large data sets without sacrificing output quality.

[1]CNR, Istituto di Informatica e Telematica, Pisa, Italy.
[2]Dipartimento di Ingegneria dell'Informazione, Università di Modena e Reggio Emilia, Modena, Italy.

The scalability of our algorithm can find applications in a number of different settings. One aspect is the sheer dimension of a single data set: the technology of tiling arrays is capable of producing a complete profile of the transcript index of an individual genome—up to 50,000 transcript sequences and 60 tissues and cell lines can be mapped in a single chip experiment (Schadt et al., 2004). The second aspect is the trade-off between the number of experiments and the response time. Microarray technology, adapted towards the needs of personalized medicine, might be used to screen a vast range of different pathological conditions over large populations. In some applications there is the need to repeat the experiments many times and to have a prompt result. For example, data taken at different times from the same patient in a healthy state and in a pathological state could be clustered to highlight differences in the metabolism due to the pathology, filtering out the background effects of healthy individual metabolic profile. *K-Boost* might be useful in this context, where the great amount of data to be managed is one of the main bottlenecks for the existing techniques. *K-Boost* runs in $O(|N| \cdot k)$ time, where N is the input matrix (i.e., the matrix whose rows and columns represent genes and experiments, respectively) and k is the number of proposed clusters. This is essentially the time required to run FPF, as the determination of k is performed very efficiently interleaved with the computation of the cluster centers by FPF.

Scalability should by no means be paid for by a decrease in output quality. Ideally one would like new algorithms to be both faster and more accurate at the same time. Clustering is an inherently approximate activity and the issue of validating the quality of clusterings is an open area of research. Broadly speaking there are ''internal'' quality criteria (based on an inter-point metric that is assumed to be significant), and ''external'' quality criteria based on cross-referencing the produced clustering with a ''golden standard'' such as that derived by Gene Ontology annotations. We will use both methodologies to measure and compare clustering quality.

One of the most important subproblems in clustering is the determination of the number k of clusters that best fit the input data. Only very few methods are able to give this indication. In *K-Boost*, we use an information theoretic technique pioneered in Tibshirani et al., (2005). However, to the best of our knowledge, such a technique has not been previously applied to real biological data sets. The experiments with yeast and human data sets confirm that the overall algorithm and the module suggesting k are effective. In addition, if we feed our estimate of k to traditional methods, like k-means, we notice an increase in their effectiveness.

The FPF heuristic is known to attain a result that is within a factor two of the optimum clustering according to the k-center criterion (Gonzalez, 1985). This theoretical guarantee, coupled with a small computational complexity and with a careful implementation, makes this algorithm an ideal starting point for attaining scalability. The FPF algorithm constructs the clusters incrementally, thus it starts with no previous information about the final value of k and adopts some criterion to determine when to stop.

To detect the value of k we use a stability-based technique for cluster validation by *prediction strength* (Tibshirani et al., 2005) that guides the selection of the ''best'' number of clusters in which the data set should be partitioned. Such technique is well founded in an Information Theoretic framework. Moreover, as already pointed out, we interleave and make the computation of both FPF and SB incremental, adding only a small amount to the cost of pure FPF.

Finally, we use previously computed centers as centroids of clusters in an iterative loop. More precisely, we associate the other data-points to the closest centroid and iteratively update centroids of clusters.

1.1. State of the art

The seminal papers by Eisen et al. (1998), Alon et al. (1999), and Wen et al. (1988) have shown the rich potential of microarray gene expression data clustering as an exploratory tool for finding relevant biological relationships amidst large gene expression data sets. Since the late nineties a growing body of knowledge has been built up on several algorithmic aspects of the clustering task (Jiang et al., 2004; Shamir and Sharan, 2002; Draghici, 2003).

Among the most popular approaches we can broadly find those *distance based* such as k-means (Tavazoie et al., 1999), Self Organized Maps (SOM) (Tamayo et al., 1999), Hierarchical Agglomerative Clustering (HAC) (Eisen et al., 1998), and several variants thereof. A second broad class of algorithms is *graph-based*: CLICK (Sharan et al., 2003), CAST (Ben-Dor et al., 1999), and CLIFF (Xing and Karp, 2001) all use weighted graph min cuts (with variants among them). Excavator (Xu et al., 2002) is based instead on Minimum Spanning Tree clustering. Other families are those *models-based* (Ramoni et al., 2002), *fuzzy-logic-based* (Belacel et al., 2004), or based on *Principal Component Analysis* (PCA) (Hastie et al., 2000). For a recent

survey, see Kerr et al. (2008). We also mention the projective clustering approach (Aggarwal et al., 1999), where good clusters are uncovered by searching subspaces of the (usually very large) input space. Projective clustering is being investigated especially within the database community (Yip and Ng, 2004; Ng et al., 2005), as a typical relational database may contain hundreds of attributes (i.e., dimensions).

Among the main issues related to clustering there are *the problem of guessing the optimal number of clusters* (Tibshirani et al., 2001, 2005; Giurcaneanu et al., 2003) and *cluster validation* (Gibbons and Roth, 2000; Yeung et al., 2001; Gat-Viks et al., 2003). In biological data analysis a further issue is to provide metrics supported by ad hoc external biological knowledge (Huang, 2006; Hanisch et al., 2002). A large and promising area of research is that of feature selection (Dugas et al., 2004; Taylor, 2006), leading to the more general concept of bi-clustering (Tanay et al., 2006; Madeira and Oliveira, 2004). Special attention has been recently paid to particular kinds of microarray experiments, notably time-series, in which there is a natural ordering and correlation for the conditions tested (Ernst et al., 2005; Bar-Joseph, 2004).

At present, there is no clear overall winner in the area of clustering algorithms for gene expression data, as any method has strong and weak points. However, one can safely say that the drive for higher quality results is always paid for by higher computational costs; therefore all these methods exhibit poor scalability or need educated guesses as to the setting of some critical parameter. In our approach we show that the two goals are not in sharp contrast: scalability need not entail lower quality.

Previously (Geraci et al., 2007), we proposed a scalable method for clustering gene expression data called *FPF-SB* that combines stability based computation of the number of clusters with the FPF heuristic for the k-center problem. FPF-SB used a single random sample and, as a consequence, there was some variance in the guess for k. *K-Boost* employs a more sophisticated multi-sample approach which leads in a more stable outcome. Combined with the final cluster refinement, this results in a noticeable quality increase of the clustering produced with respect to FPF-SB.

In general we can split the algorithms for clustering field into two large groups: methods that take a suggested number of clusters as input and methods that try to make an accurate guess of the ''optimal'' (or close to optimal) number of clusters, according to some internal criterion. Our proposal *K-Boost* is in the second category thus we will compare its performance with methods like CLICK and FPF-SB that are in the same class. We also do comparisons with methods like k-means and HAC, where we feed them a plausible number of clusters (as suggested by *K-Boost* or CLICK), thus giving them an advantage they do not have when applied in a stand-alone fashion.

The FPF heuristic has been applied also in the context of microarray clustering for time-series by Ernst et al. (2005). However, our approach is different. We apply the FPF algorithm directly to real-world input data. In Ernst et al. (2005), it is applied to a set of artificially generated data points that are meant to uniformly cover the parametric space of possible experimental outcomes. This second approach suffers of scalability problems as the cardinality of the discrete search space grows exponentially in the parameters of the experiment.

In Section 2, we discuss algorithmic tools and techniques, preliminary to the description of the *K-Boost* algorithm in Section 3. The data sets and measurements are described in Section 4. Some final remarks are given in Section 5.

2. PRELIMINARIES

2.1. Clustering and distance function

Let $N = \{p_1, \ldots, p_n\}$ be a set of n vectors in R^m, a partition $\tilde{N} = \{N_1, \ldots, N_k\}$ of N is a *clustering*, where each N_t, for $t \in \{1, 2, \ldots, k\}$, is called a *cluster*. Given a clustering $\tilde{N}$, two elements $p_i, p_j \in N$ are *mates* according to $\tilde{N}$ if they belong to the same cluster $N_t \subseteq \tilde{N}$, *non-mates* otherwise. Given two vectors $p_i, p_j \in N$ (with components $p_{s,t}, s \in \{i, j\}$ and $1 \leq t \leq m$), we denote with $d(p_i, p_j)$ their distance and we say that they are *similar* (respectively, *different*) if $d(p_i, p_j)$ is small (respectively, large). Our choice of $d(p_i, p_j)$ is based on the *Pearson Coefficient,* $P(p_i, p_j)$, given by

$$P(p_i, p_j) = \frac{\sum_{t=1}^{m} (p_{i,t} - \mu_i)(p_{j,t} - \mu_j)}{\sqrt{\left(\sum_{t=1}^{m} (p_{i,t} - \mu_i)^2\right)\left(\sum_{t=1}^{m} (p_{j,t} - \mu_j)^2\right)}}$$

where μ_i and μ_j are the means of p_i and p_j, respectively.

The Pearson Coefficient is a very popular measure of similarity in the context of gene expression microarray data clustering but it is not a distance. To come up with a measure suitable for a metric space method, we first define $\delta(p_i, p_j) = 1 - P(p_i, p_j)$, with $0 \leq \delta(p_i, p_j) \leq 2$ (since $-1 \leq P(p_i, p_j) \leq 1$). This quantity, which is in turn a widely accepted valid dissimilarity measure in gene expression analysis, violates the triangle inequality constraint, and thus is not a metric in a strict sense. However, the square root of $\delta(p_i, p_j)$ is proportional to the Euclidean distance between p_i and p_j (Clarkson, 2006), and hence can be adopted within algorithms (such as FPF) designed for metric spaces. Our definition of distance function is thus:

$$d(p_i, p_j) = \sqrt{\delta(p_i, p_j)}. \tag{1}$$

2.2. *Furthest-Point-First clustering algorithm*

The FPF algorithm (Gonzalez, 1985) computes a good clustering by finding a solution to the k-center problem, defined as follows:

Given a set of points N on a metric space M, a distance function $d(p_i, p_j) \geq 0$ satisfying the triangle inequality, and an integer k, a k-center set is a subset $C \subseteq N$ such that $|C| = k$. The k-center problem is to find a k-center set that minimizes the maximum distance of each point $p \in N$ to its nearest center in C, i.e., minimizes the quantity $\max_{p \in N} \min_{c \in C} d(p, c)$.

The approximation algorithm FPF is based on a greedy approach: it increasingly computes the set of centers $C_1 \subset \ldots \subset C_k$, where C_k is returned as the (approximate) solution to the problem. The set C_1 contains only one randomly chosen point $c_1 \in N$. Each iteration i, with $1 \leq i \leq k-1$, has the set of centers C_i at its disposal and works as follows:

1. for each point $p \in N \backslash C_i$ computes its closest center c_p, i.e., the point c_p satisfying $d(p, c_p) = \min_{c \in C_i} d(p, c)$;
2. determines $p \in N \backslash C_i$ that is farthest from its closest center c_p; *i.e.*, the point p that satisfies $\max_{p \in N \backslash C_i} d(p, c_p)$ and set $c_{i+1} = p$;
3. defines $C_{i+1} = C_i \cup \{c_{i+1}\}$.

At each iteration a new center is added to the set of centers being computed. The algorithm stops after $k-1$ iterations giving as result the set C_k.

Observe that, at iteration $i+1$, the first step, there is no need to recalculate the distance of each point p to all centers, but just $d(p, c_i)$, the distance to the unique center c_i added during the previous iteration. Then, just compare this distance with $d(p, c_p)$, the minimum distance to centers of C_i. According to the result of this comparison, c_p can be updated or not. Hence, if for each p the value $d(p, c_p)$ is stored, then each iteration can be executed in $O(n)$ space and a k-center set can be computed in $O(kn)$ distance computations.

To actually compute a clustering associated to such a k-center set, FPF simply partitions N into k subsets $N_1, \ldots, N_k$, each corresponding to a center in C_k and such that $N_j = \{p \in N | c_p = c_j\}$. In other words, the cluster N_j is composed of all points for which c_j is the closest center, for each $j = 1, \ldots, k$.

The basic FPF algorithm can be enhanced with an heuristic to obtain a speed-up in running time with no degradation of the quality of the solution (Geraci et al., 2006) (*i.e.*, with same approximation factor). Taking advantage of the triangle inequality, the modified algorithm avoids considering points that cannot change their closest center. To this aim, at each iteration i the algorithm maintains, for each center $c_j \in C_i$, the set N_j defined above of points for which c_j is the closest center, for $j = 1, \ldots, i$ (*i.e.*, it builds the clusters associated to intermediate solutions) storing the points in order of decreasing distance from c_j. Searching for the points closest to c_i, the algorithm scans the elements of N according to their order in each N_j. Given $p \in N_j$, with $1 \leq j < i$, if $d(p, c_j) \leq \frac{1}{2} d(c_j, c_i)$, then stops scanning N_j, as there cannot be any other point closer to c_i than to c_j. In this version, the distances between centers must be stored, requiring additional $O(k^2)$ space. As a consequence, storage consumption is linear in n only provided that $k = O(\sqrt{n})$, which is often the case, in practice.

In the microarray setting, N is represented as a $n \times m$ matrix, where n is the number of gene probes in the data set and m is the number of conditions tested on each probe, the metric space M is $\mathcal{R}^m$ with the distance function (1). In the rest of this article, when referring to the FPF algorithm, we refer to the enhanced version. Observe that, since the distance measure adopted can be computed in $O(m)$ time, the computational cost of the FPF algorithm is $O(knm)$ scalar operations.

2.3. Stability-based technique

Stability-based techniques are used to compute the number k of clusters into which N has to be partitioned, which can then be used by clustering algorithms that requires it as input. In this article, we refer to the prediction strength method developed in Tibshirani et al. (2005).

To obtain the estimate of k using the stability-based method, proceed as follows. Given a clustering algorithm, a set N of n elements, and an integer η, randomly choose a sample $S_r \subseteq N$ of cardinality η. Then, for increasing values of t ($t = 1, 2, \ldots$) repeat the following steps:

(i) cluster both S_r and $S_{ds} = N \backslash S_r$ into t clusters, obtaining the partitions N_r^t and N_{ds}^t, respectively;
(ii) measure how well the t centers of the clustering N_r^t predict co-memberships of mates in N_{ds}^t by counting how many pairs of elements that are mates in N_{ds}^t are also mates in the clustering of S_{ds} obtained according to the centers of N_r^t.

The measure computed in step (ii) at iteration t is obtained as follows: let $p_i, p_j \in S_{ds}$, then $D[i,j] = 1$ if p_i and p_j are mates both in N_{ds}^t and according to the centers of N_r^t, $D[i,j] = 0$ otherwise.
Let $N_{ds,\ell}^t, \ell = 1, \ldots, t$, be the clusters of N_{ds}^t, then the *prediction strength* $PS(t)$ of N_{ds}^t is defined as:

$$PS(t) = \min_{1 \le \ell \le t} \frac{1}{\#\text{pairs in } N_{ds,\ell}^t} \sum_{i,j \in N_{ds,\ell}^t, i<j} D[i,j], \tag{2}$$

where the number of pairs in $N_{ds,\ell}^t$ is given by its binomial coefficient over 2. In other words, $PS(t)$ is the minimum fraction of pairs, among all clusters in N_{ds}^t, that are mates according to both clusterings, hence $PS(t)$ is a worst case measure.

2.4. Parameter validation for the stability-based technique

To effectively and efficiently use the prediction strength approach, we had to make a couple of decisions: (1) using (possibly a subset of) the values $\{PS(t) | t = 2, \ldots, n\}$ decide which is the best candidate as the value of k; (2) decide whether to run just one or more than one prediction experiments. To this end, we first computed z_{score}, chosen as reference measure of quality, for increasing values of k. (Figure 1 shows the results obtained for the Cho et al. data set; however, the results for the other data sets are qualitatively the same.) Using this information we have then been able to globally evaluate the results obtained with the particular values of k determined by different choices for (1) and (2).

As for the first decision above, namely the choice of the candidate k for a single prediction experiment, Tibshirani et al. (2005) suggest (based on experiments on synthetic and well-separated data) to take the global maximum of the $PS(t)$, for $t > 1$. Clearly, this requires that $PS(t)$ be computed for all possible values of t. On our real data, we observe a qualitatively similar behavior as the one reported by Tibshirani et al., even though the signal is not as clean. In particular, when t is close to 1, a high value of $PS(t)$ does not always reflect a good partitioning. Considering the "essentially decreasing" shape of z_{score} (see the linear fit in Fig. 1), we then decided, as a good time/quality compromise, to take the first local maximum of $PS(t)$ after the first initial decrease of the function values. In other words, we stop the prediction strength computation as soon as $PS(t+1) < PS(t)$ and set $k = t$.

As for the second point, we repeated the prediction experiments r-times, for $r = 1, \ldots, 6$, taking the average results for $r > 1$. We then determined the corresponding values of z_{score} using the graph of Figure 1 (and those for the other data sets). Our findings indicate in a sufficiently clear way that the z_{score} increases with r. However, the differences are quite reasonable both in the values of k suggested and the corresponding z_{score} estimates. For instances, for the Cho et al. data set, the values of k suggested for the six different r-fold validations were all in the range of 8–14, but if we excluded 1-fold, the range shrank to 8–11. Very similar figures can be observed for the other data sets. Based on these findings, we then decided to let r be a user-definable parameter of *K-Boost* but assumed a default value of 2. Twofold validation is thus the choice we adopted to compare *K-Boost* with the other clustering algorithms.

3. *K*-BOOST ALGORITHM

Algorithm *K-Boost* works in two phases. Intuitively, in the first phase, it estimates a plausible number of clusters according to the stability-based techinique, working on a subset of elements. At the same time, it

determines a set of seeds that will be used in the next phase to produce the clustering. In the second phase, *K-Boost* adds the remaining elements, one by one, to the closest cluster, i.e., the one having closest centroid. At the beginning, centroids are the seeds identified in the first phase; then, each time a cluster is modified by the insertion of a new point, its centroid is updated accordingly. The use of centroids in place of centers has the positive effect to produce a smoother resulting clustering. On the other hand, as centroids depend on the order of inclusion, we reorder points to mitigate the effect of the initial ordering.

A detailed description of the Algorithm follows.

Phase I: Determining the number of clusters To determine the number k of clusters, *K-Boost* uses an approach similar to the one developed by the FPF-SB algorithm (Geraci et al., 2007), interleaving the Stability Based technique and the FPF algorithm using the *prediction strength* measure. *K-Boost* also performs a two-folds validation to produce a more robust prediction of k. This phase works in two steps:

I.1. Compute centers for random sample clustering: The input set N is partitioned in three sets S_l, S_r and S_{ds} such that $|S_l| = |S_r| = \eta = \sqrt{n}$ and $S_{ds} = N \backslash (S_l \cup S_r)$. Then, the FPF algorithm is run on sets S_l and S_r until $\mathrm{k} = \eta$. At the end of this procedure each cluster contains exactly one element, namely its center. Moreover, elements in S_l and S_r are ranked according to the order in which centers are extracted by FPF. In the following, for $w \in \{l, r\}$, we let $S_w^t \subseteq S_w$ denote the set of centers extracted up to the t-th iteration of FPF over S_w.

I.2. Evaluation of prediction strength: The set S_{ds} is clustered using FPF and centers of the computed clustering are returned as seeds to be used the second phase.

At the same time, to determine the number of clusters k (and to decide when to stop clustering S_{ds}), S_{ds} is also clustered into two distinct clusterings using the centers in S_w^t at iteration t, for $w \in \{l, r\}$, and value $PS_w(t)$ is computed. The clustering procedure applied to S_w stops as soon as, for $t > 2$, $PS_w(t-2) < PS_w(t-1)$ and $PS_w(t-1) > PS_w(t)$ hold (i.e., at the first local maximum different from $\mathrm{t} = 1$) or when $t_w = \sqrt{n}$. The guess of k for N is set to the average between $t_1 - 1$ and $t_r - 1$, rounded to the closest integer.

The evaluation of the prediction strength at each step can be implemented efficiently in the following way. Consider iteration t of FPF: compute clusters $N_{ds,1}^t, \ldots, N_{ds,t}^t$, and for each $p \in S_{ds}$, keep the index $i_w(p,t)$ of its closest center in S_w^t, for $w \in \{l, r\}$. Such index can be updated in $O(m)$ time given $i_w(p, t-1)$, by comparing the distance of p from its closest center in S^{t-1} and the one to point s_t; *i.e.,* $d(p, s_{i_w(p,t-1)})$ with $d(p, s_t)$. Now, for each cluster $N_{ds,\ell}^t$, $\ell = 1, \ldots, t$ count the number of elements that are closest to the same center of S^t in time $O(m|N_{ds,\ell}^t|)$, by means of indices $i_w(p,t)$. Hence, each term of the summation in formula (2) can be computed in time $O(m|N_{ds,\ell}^t|)$ and the prediction strength at each iteration in time $O\left(\sum_{\ell=1}^{t} m|N_{ds,\ell}^t|\right) = O(m|S_{ds}|)$ (leading to the same cost needed for clustering S_{ds}, when multiplied by the number of iterations).

The computational cost of the first phase is given by the sum of the costs of the two steps: $O(m\eta^2)$ for the first step and $O(2km|S_{ds}|)$ for the second one. Summing up, the total cost of the first phase is $O(m\eta^2 + 2km|S_{ds}|) = O(m(n + k|S_{ds}|))$.

Phase II: Producing the clustering *K-Boost* incrementally builds a k-clustering starting from the seeds computed in the previous phase: centers $C = \{c_1, \ldots, c_k\}$ of N_{ds}^k re used as initial centroids and the points in $\bar{N} = N \backslash C$ are inserted one by one in the clustering, updating the centroids at each insertion accordingly. As the order in which the points are taken for insertion could affect the final output, elements in $\bar{N}$ are reordered using the procedure outlined below.

Initialize an empty list for each centroid $c_i \in C$, then insert the points of $\bar{N}$ in the list associated to the closest centroid. Keep the elements of each list ordered according to their distance to the associated centroid. This can be done efficiently by observing that the centroid c_i is the center of the cluster $N_{ds,i}^k$ and the list associated to c_i can be initialized with all the elements of cluster $N_{ds,i}^k$ except its center. Points are thus globally reordered by interleaving their relative rankings (see online Supplementary Material at www .liebertonline.com).

Note that the update of the centroid can be made in $O(m)$ time. Let N_j be a cluster of points $\{p_1, \ldots, p_{n_j}\}$ and let $p_{z,i}$ be the i-th component of the m-dimensional vector $p_z \in N_j$. By definition, the centroid, $cr = \{cr_1, \ldots, cr_m\}$, of N_j is a vector such that cr_i is the average of the values of $p_{z,i}$ for $z = 1, \ldots, n_j$, *i.e.*, equal to $\left(\sum_{z=1}^{n_j} p_{z,i}\right) \Big/ n_j$. When only one new point p_{n_j+1} is added to the cluster, each component of the

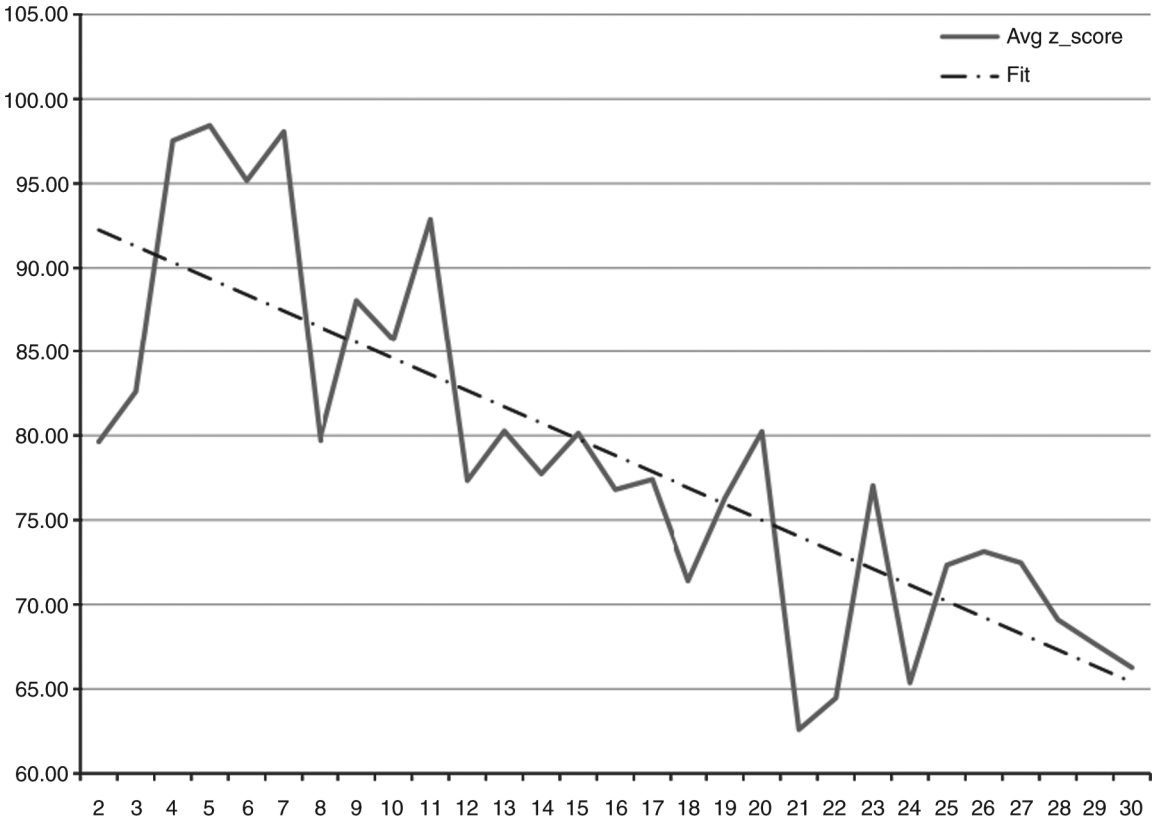

FIG. 1. Plottiing z-score as a function of k for the Cho et al. data set. Each value of z-score is the average of three experiments using Cluster Judge. One can notice a decreasing trend after a first peak.

centroid cr_i can be updated in constant time (see supplementary material). Thus, Phase II can be done in time $O(|S_{ds}|km)$.

In conclusion, the overall cost of *K-Boost* is the sum of the costs of the two phases: $O(m(n+k|S_{ds}|)+|S_{ds}|km) = O(nm+2k(n-\sqrt{n})) = O(nmk)$.

4. EXPERIMENTS

We compared *K-Boost* with CLICK and FPF-SB, two algorithms that are able, as *K-Boost* is, to determine a plausible number of clusters. Furthermore, since FPF-SB, CLICK, and *K-Boost* usually suggest a different clustering size, we decided to evaluate the ''robustness'' of their proposed values by feeding them to some of the most popular clustering algorithms for microarray gene expression data, namely FPF, HAC, and *k*-means.

CLICK has been run under EXPANDER (EXpression Analyzer and DisplayER) (Sharan et al. 2003) version 4.0.3, a java-based tool for the analysis of gene expression data, that is capable of clustering and visualizing the corresponding results. All the other algorithms have been executed under the multi clustering and visualization tool *AMIC@* (Geraci et al. 2008).

4.1. Evaluation

To evaluate the results we used both internal and external measures of quality. As internal measures we used *homogeneity* and *separation*, as defined in Sharan et al. (2003), while as an external measure we adopted the z_{score} computed by the *ClusterJudge* tool (Gibbons and Roth, 2000).

Homogeneity and separation. Let M be the set of indices of points forming unordered mate pairs; the *average homogeneity* is

TABLE 1. YEAST: SUMMARY OF DATA SET PROPERTIES

Data set	*Cho et al.*	*Eisen et al.*	*Spellman et al.*
Probes	6601	2467	6178
Conditions	17	79	82
Problem size	112,217	194,893	506,596

$$H_{ave} = \frac{1}{|M|} \sum_{(i,j)\in M} P(p_i, p_j),$$

while the *average separation* is

$$S_{ave} = \frac{1}{\binom{n}{2} - |M|} \sum_{(i,j)\notin M} P(p_i, p_j).$$

Both homogeneity and separation have values in the range [-1, 1]. Higher values of homogeneity and lower values of separation indicate higher quality. Note that singletons do not contribute to the average homogeneity, but do contribute to separation. Since both measures are greatly influenced by the number of clusters, they are most significant in comparing solutions having the same value of k.

External measure. The on-line available tool *ClusterJudge* scores yeast *(Saccharomyces cerevisiae)* genes clusterings by evaluating the mutual information between a gene's membership in a cluster, and the attributes it possesses, as annotated by the Saccharomyces Genome Database (SGD) and the Gene Ontology Consortium. In particular, ClusterJudge first determines a set of gene attributes, among those provided by Gene Ontology, that are independent and significant; then it computes the mutual information of the proposed clustering and that of a reference random clustering. Finally it returns the z_{score}, i.e., $z_{\text{score}} = (MI_{\text{real}} - MI_{\text{random}}) / \sigma_{\text{random}}$, where MI_{random} is the mean of the mutual information score for the random clustering used, and σ_{random} is the standard deviation. The higher the z_{score} the better the clustering. Given the randomized nature of the test, different runs produce slightly different numerical values, although the ranking of the methods is stable and consistent across different applications of the evaluation tool. In absolute values we may observe a variation typically in the range ± 5. For this reason, for each data set used we repeated three times the evaluation of the output of all the different algorithms, reporting the average z_{score} only. ClusterJudge methodology is available only for yeast genes, but is independent of both the algorithm and the metric used to produce the clustering, and thus is in effect validating both choices.

Platform. The results reported here have been obtained on a 3.2-GHz Intel Pentium D dual-core workstation with 3.2-GB RAM, running Linux kernel 2.6.18.2-34. *K-Boost* is written in Python version 2.5.

TABLE 2. EXPERIMENTAL RESULTS COMPARING ALGORITHMS THAT SUGGEST A PLAUSIBLE VALUE OF K

	Data set											
	Cho et al.				*Eisen et al.*				*Spellman et al.*			
Method	k	*Sg*	T	z_{score}	k	*Sg*	T	z_{score}	k	*Sg*	T	z_{score}
CLICK	30	136	540	62.47	8	0	165	42.26	27	17	3000	73.93
FPF-SB	14	0	16	61.93	12	0	19	57.90	16	0	59	58.20
K-Boost	10	0	21	**79.00**	8	0	23	**69.03**	19	0	134	**78.60**

For each algorithm and data set, we report the number k of clusters, the number *Sg* of singleton data points, the running time in seconds, and the z_{score} computed by ClusterJudge. The results shown for z_{score} are the average of three independent runs. *K-Boost* achieves a significantly better z_{score} on all the three yeast data sets using far less computation time.

TABLE 3. CHO ET AL. DATASET

	k selection														
	FPF-SB estimate					*CLICK estimate*					*K-Boost estimate*				
Method	*k*	*T*	z_{score}	*Hom*	*Sep*	*k*	*T*	z_{score}	*Hom*	*Sep*	*k*	*T*	z_{score}	*Hom*	*Sep*
FPF	14	5	56.5	0.572	0.011	30	10	52.43	**0.645**	0.016	10	4	**60.63**	0.548	**−0.025**
HAC	14	103	53.1	0.517	−0.143	30	103	56.57	**0.617**	−0.059	10	102	**61.00**	0.511	**−0.151**
k-means	14	18	74.6	0.655	−0.035	30	38	67.80	**0.703**	−0.001	10	10	**95.33**	0.631	**−0.056**

Experimental results comparing algorithms that take *k* as input with the values computed by FPF-SB, CLICK, and *K-Boost*. For each algorithm and data set, we report the number *k* of clusters, the running time *T* in seconds, the z_{score} computed by ClusterJudge, Homogeneity *Hom*, and Separation *Sep*. The results shown for z_{score} are the average of three independent runs. HAC has been run with average linkage. *k*-means has been run for 30 iterations.

TABLE 4. EISEN ET AL. DATASET

	k selection									
	FPF-SB estimate					*CLICK and K-Boost estimate*				
Method	*k*	*T*	z_{score}	*Hom*	*Sep*	*k*	*T*	z_{score}	*Hom*	*Sep*
FPF	12	6	53.4	0.483	0.079	8	4	**56.87**	**0.524**	**−0.076**
HAC	12	7	34.3	0.440	0.042	8	7	**37.10**	0.439	**−0.292**
k-means	12	14	62.3	0.528	0.102	8	10	**64.86**	**0.572**	**−0.021**

Experimental results comparing algorithms that take *k* as input with the values computed by FPF-SB, CLICK, and *K-Boost*. For each algorithm and data set, we report the number *k* of clusters, the running time *T* in seconds, the z_{score} computed by ClusterJudge, Homogeneity *Hom*, and Separation *Sep*. The results shown for z_{score} are the average of three independent runs. HAC has been run with average linkage. *k*-means has been run for 30 iterations. Note that FPF, HAC, and *k*-means attain significantly better performance when fed with *K-Boost*'s estimate of *k*.

TABLE 5. SPELLMAN ET AL. DATASET

	k selection														
	FPF-SB estimate					*CLICK estimate*					*K-Boost estimate*				
Method	*k*	*T*	z_{score}	*Hom*	*Sep*	*k*	*T*	z_{score}	*Hom*	*Sep*	*k*	*T*	z_{score}	*Hom*	*Sep*
FPF	16	19	62.2	0.456	0.188	27	32	46.16	**0.489**	0.066	19	22	**62.47**	0.481	**0.054**
HAC	16	92	55.8	0.420	0.176	27	92	56.63	**0.463**	−0.018	19	92	**57.00**	0.448	**−0.017**
k-means	16	83	80.10	0.507	0.149	27	130	79.66	**0.559**	0.047	19	75	**81.07**	0.538	**0.035**

Experimental results comparing algorithms that take *k* as input with the values computed by FPF-SB, CLICK, and *K-Boost*. For each algorithm and data set, we report the number *k* of clusters, the running time *T* in seconds, the z_{score} computed by ClusterJudge, Homogeneity *Hom*, and Separation *Sep*. The results shown for z_{score} are the average of three independent runs. HAC has been run with average linkage. *k*-means has been run for 30 iterations. Note that FPF, HAC, and *k*-means attain significantly better performance in terms of z_{score} and separation when fed with *K-Boost*'s estimate of *k*, while maintaining high level of homogeneity.

4.2. Data sets and experiments on yeast

The algorithms were tested on three well-studied yeast data sets. The first is the yeast cell cycle data set described in Cho et al. (1998). In their work the authors monitored the expression levels of 6218 *Saccharomyces cerevisiae* putative gene transcripts (ORFs). Probes were collected at 17 time points taken at 10 min intervals (160 minutes), covering nearly two full cell cycles. The second data set, described in Spellman et al. (1998), is a comprehensive catalog of 6178 yeast genes whose transcript levels vary periodically within the cell cycle (for a total of 82 conditions). The third data set, described in Eisen et al. (1998), contains 2467 probes under 79 conditions, and consists of an aggregation of data from experiments

TABLE 6. IYER ET AL. DATASET

	k selection														
	FPF-SB estimate					*CLICK estimate*					*K-Boost estimate*				
Method	*k*	*Sg*	*T*	*Hom*	*Sep*	*k*	*Sg*	*T*	*Hom*	*Sep*	*k*	*Sg*	*T*	*Hom*	*Sep*
FPF-SB	6	0	0.17	0.769	−0.135	5	0	0.14	0.727	**−0.181**	9	0	0.25	**0.777**	−0.090
CLICK	—	—	—	—	—	5	21	38	0.711	−0.384	—	—	—	—	—
K-Boost	6	0	0.34	0.564	−0.014	5	0	0.28	0.762	−0.148	9	0	0.50	**0.803**	−0.055

Experimental results on 517 probes and 18 conditions comparing algorithms that suggest a plausible value of k. For each algorithm, we report the number *k* of clusters, the number *Sg* of singleton data points, the running time *T* in seconds, Homogeneity *Hom*, and Separation *Sep*. *K-Boost* achieves a significantly better homogeneity than FPF-SB and CLICK, while maintaining good level of separation. FPF-SB than *K-Boost* attain better performance when fed with *K-Boost*'s estimate of *k*. Note that CLICK cannot be fed with an estimate for *k*.

on the budding yeast *Saccharomyces cerevisiae* (including time courses of the mitotic cell division cycle, sporulation, and the diauxic shift). Table 1 summarizes the properties (number of probes, number of conditions, and overall problem size) of the three data sets.

Experimental results are reported in Tables 2–5. For each experiment we report (all or some of) the following parameters: the number of clusters *k* (either computed or fed as input), the number of singletons *Sg* produced by the clustering procedure, the computation time *T* in seconds, the z_{score} (external measure), the homogeneity *Hom* and separation *Sep* (internal measures). Note that CLICK is the only algorithm, among those that we have tested, that sometimes produces singletons in its clustering and puts them into a single cluster labeled cluster zero.[1]

In Table 2, we observe that *K-Boost* achieves a significantly better z_{score} on all the three yeast data sets using far less computation time (by a factor of 4–30) than CLICK. FPF-SB is even faster, but it attains lower z_{score} than *K-Boost*. On larger data sets the time-gap is due to increase since CLICK asymptotically runs in $O(n^2m + n^3)$. The actual number of clusters computed by CLICK has little influence on its speed since the bulk of the cost is paid for in the set up and the initial iterations. In contrast, *K-Boost* (and FPF-SB) has a lower asymptotic cost $O(nmk)$. Note that in two out of three cases CLICK and *K-Boost* make significantly different choices as to the plausible value of *k*. In this case we prefer to use only external measures because homogeneity and separation have drift due to the number of clusters. Note also that the measured running times of *K-Boost* for the three data sets (of increasing sizes; Table 1) confirm the scalability properties of our algorithm, in almost perfect agreement with the $O(nmk)$ theoretical bound (which clearly ignores lower order terms).

In Tables 3–5, we report the results obtained by FPF, HAC and *k*-means on input the Cho et al., Eisen et al. and Spellman et al. data sets, respectively, and the values of *k* computed by FPF-SB, CLICK and *K-Boost*. In these tables, we can observe a uniform behavior of the three algorithms across the three data sets when changing the value of *k* in input. In fact, FPF, HAC, and *k*-means always attain significantly better performance in terms of z_{score} and separation when fed with *K-Boost*'s estimate of *k*; on the other hand, Click's estimates leads to better homogeneity figures.

4.3. Data set and experiments on human fibroblasts

Iyer et al. (1999) studied the temporal response of human fibroblasts in an experiment devised as follows. Initially a culture of fibroblasts from human neonatal foreskin is induced in a quiescent state by serum deprivation for 48 hours. At time zero the cells are stimulated by the addition of a medium containing 10% fetal bovine serum (FBS). During quiescence and subsequently in 12 preselected time intervals (in a range: 15 min - 24 hrs) samples were taken and the activity of 8613 genes measured via DNA microarray hybridization. Six more measurements are taken in a parallel experiments in which both cycloheximide and

[1]We report the number of singletons generated by each clustering algorithm only in Table 2. This is because CLICK, in these sets of experiments, is the only tested algorithm generating singletons.

TABLE 7. IYER ET AL. DATASET

	k selection														
	FPF-SB estimate					*CLICK estimate*					*K-Boost estimate*				
Method	*k*	*Sg*	*T*	*Hom*	*Sep*	*k*	*Sg*	*T*	*Hom*	*Sep*	*k*	*Sg*	*T*	*Hom*	*Sep*
FPF	6	0	0.05	0.769	−0.135	5	0	0.14	0.727	**−0.181**	9	0	0.25	**0.777**	−0.090
HAC	6	0	0.08	0.519	0.014	5	0	0.08	0.683	**−0.410**	9	0	0.08	**0.763**	−0.144
k-means	6	0	0.23	0.576	−0.023	5	0	0.22	0.769	**−0.155**	9	0	0.50	**0.811**	−0.031

Experimental results on 517 probes and 18 conditions comparing algorithms that take k as input with the values computed by FPF-SB, CLICK, and *K-Boost*. For each algorithm, we report the number *k* of clusters, the number *Sg* of singleton data points, the running time *T* in seconds, Homogeneity *Hom*, and Separation *Sep*. HAC has been run with average linkage. *k*-means has been run for 30 iterations.

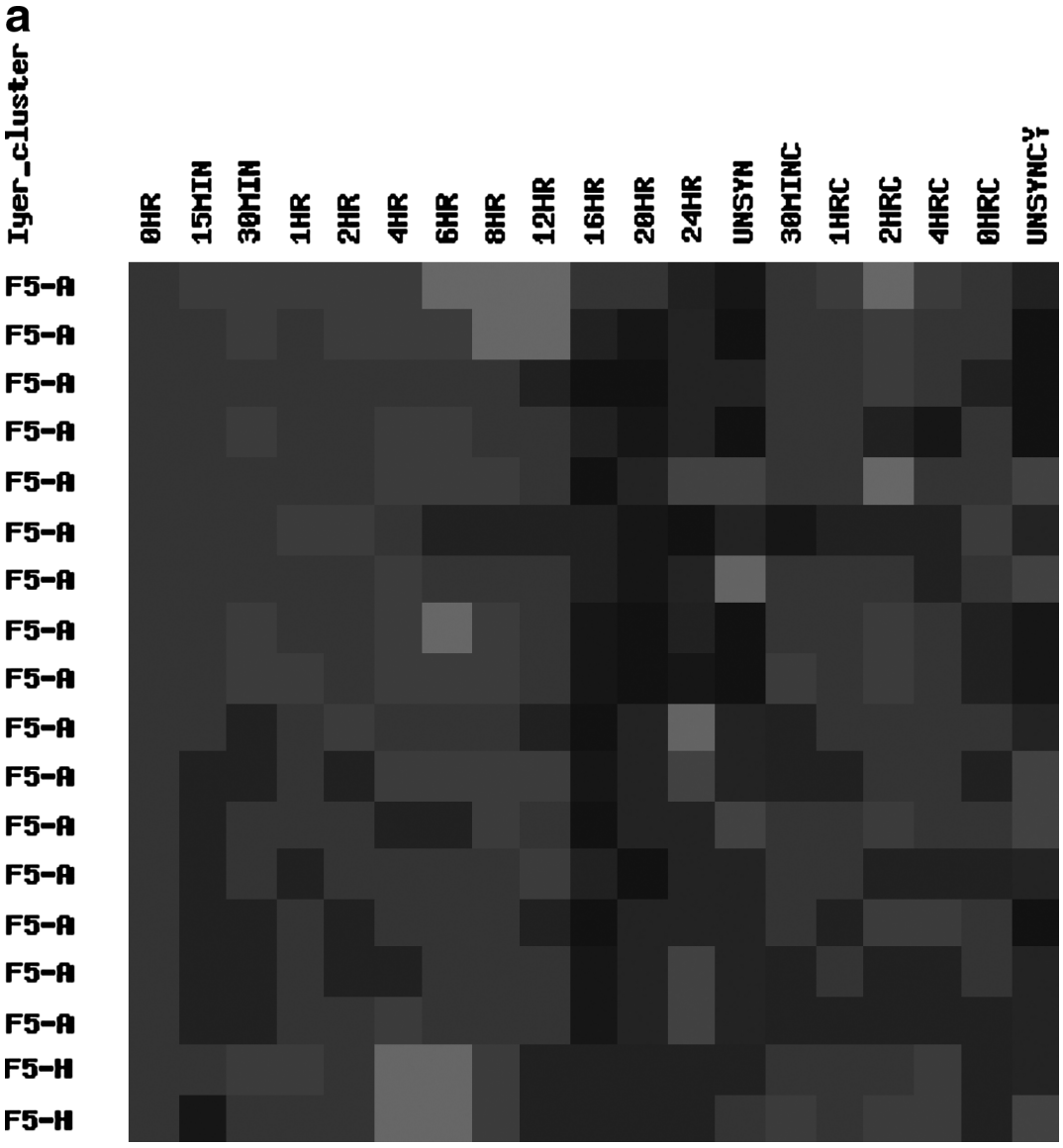

FIG. 2. Heatmaps for some of the clusters produced on data from Iyer et al. (1999): 517 genes and 18 conditions. *K-Boost* predicted nine clusters. Here we show three clusters (only the genes for which a biological function is indicated in Figure 5 of Iyer et al. [1999]). There is good agreement between the generated clusters and the biological functions. (**a**) Cluster III—Rich in (I) cholesterol biosynthesis genes. (**b**) Cluster I—Rich in (A) cell cycle and proliferation genes. (**c**) Cluster V—Rich in (D) angiogenesis genes and (G) re-epithelialization.

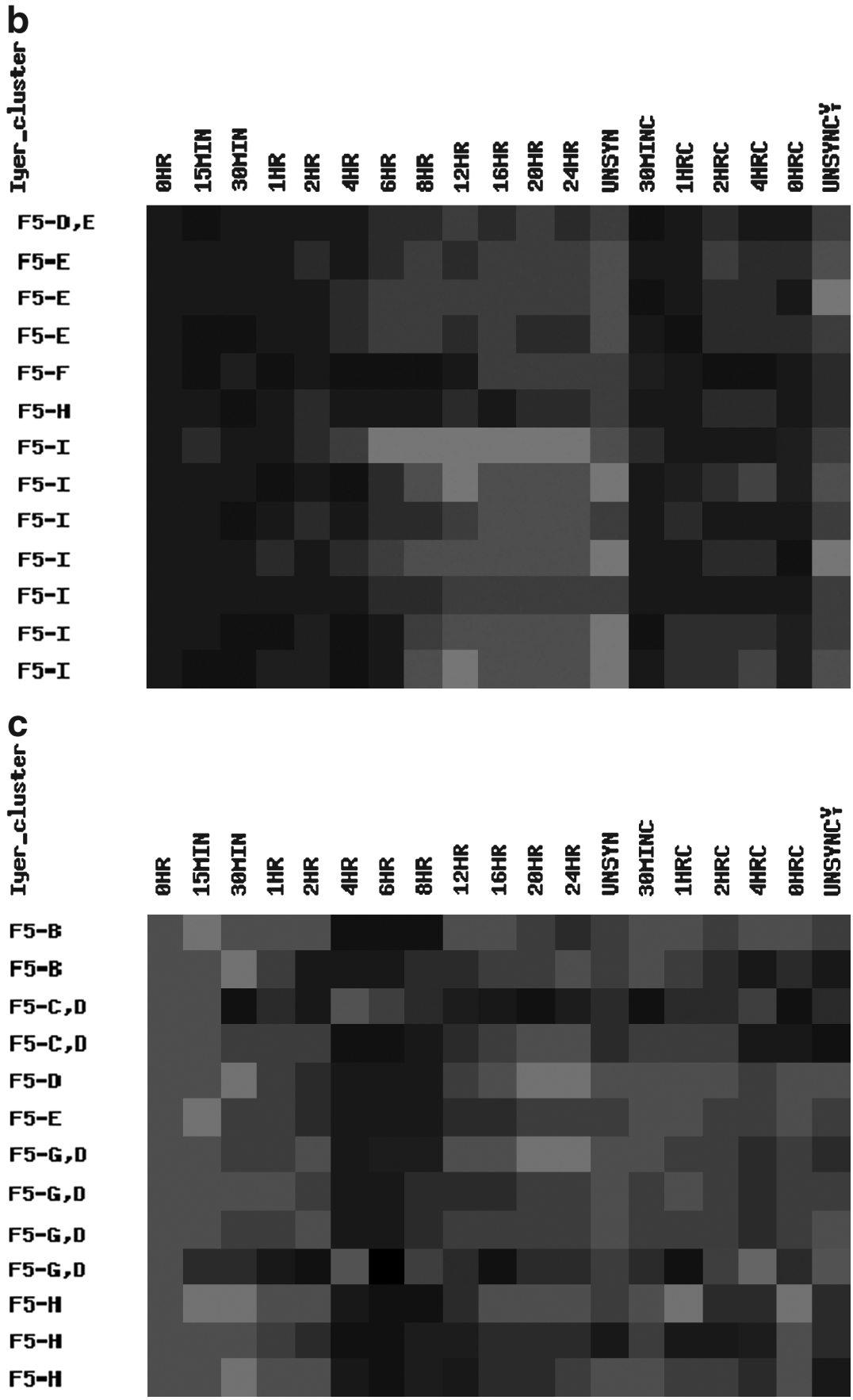

FIG. 2. (*Continued*)

FBS are added at time zero. A subset of 517 genes showing a sufficiently dynamic response was selected and analyzed using the hierarchical clustering technique in Eisen et al. (1998).

Clustering experiments reported in Tables 6 and 7 indicate that CLICK, FPF-SB, and *K-Boost* make different estimates of k. The results in Table 6 show that *K-Boost* achieves better homogeneity than FPF-SB and CLICK, while maintaining good level of separation. Furthermore, both FPF-SB and *K-Boost*[2] attain better performance when fed with *K-Boost*'s estimate of k.

Table 7 shows that FPF, HAC, and k-means always attain significantly better performance in terms of homogeneity when fed with *K-Boost*'s estimate of k, while maintaining good level of separation.

K-Boost's estimate, namely nine, is exactly the number of functionally different classes considered in Iyer et al. (1999). In Iyer et al. (1999), nine families of genes characterizing a well-defined biological function are shown: (A) cell cycle and proliferation, (B) coagulation and hemostasis, (C) inflammation, (D) angiogenesis, (E) tissue remodeling, (F) cytoskeletal reorganization, (G) re-epithelialization, (H) unidentified role in wound healing, and (I) cholesterol biosynthesis. We run *K-Boost* on all the 517 genes, obtaining nine clusters. Within these clusters, we identified 93 genes belonging to the nine families (a gene can belong to ore than one family). In most cases, clusters that have been produced have a prevalent biological function. Figure 2 shows heatmaps for some of the produced clusters.

5. CONCLUSION

Efficient and effective analysis of large data sets from microarray gene expression data is one of the keys to time-critical personalized medicine. The issue we address here is the scalability and quality of the data processing software for clustering gene expression data into groups with homogeneous expression profile. In this article, we proposed *K-Boost*, a new clustering algorithm that is computationally efficient for large data sets, computes a plausible estimate of the number of clusters, based on information-theoretic principles, and fares well in comparative experiments. Clustering is still an open and active research area, and there is no universally recommended method of choice. Accordingly, we used several quality measures and observed that no algorithm is the winner in all situations; so, depending on the purpose of the analysis, one might prefer methods that are stronger in separation or those that are stronger in homogeneity and z_{score}.

ACKNOWLEDGMENTS

Funding for this research has been provided by Italian Registry of ''.it'' ccTLD and by Italian Ministry of University and Research (PRIN number 2006018748_004, ''Metodi di ottimizzazione numerica nell'ambito dei problemi inversi'').

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Aggarwal, C.C., Wolf, J.L., Yu, P.S., et al. 1999. Fast algorithms for projected clustering. *SIGMOD Rec.* 28, 61–72.

Alon, U., Barkai, N., Notterman, D., et al. 1999. Broad patterns of gene expression revealed by clustering analysis of tumoral and normal colon tissues probed by oligonucleotide array. *Proc. Nat. Acad. Sci. USA* 1999, 6745–6750.

Bar-Joseph, Z. 2004. Analyzing time series gene expression data. *Bioinformatics* 20, 2493–2503.

[2]Note that CLICK cannot be fed with an estimate of k.

Belacel, N., Cuperlovic-Culf, M., Laflamme, M., et al. 2004. Fuzzy J-means and VNS methods for clustering genes from microarray data. *Bioinformatics* 20, 1690–1701.

Ben-Dor, A., Shamir, R., and Yakhini, Z. 1999. Clustering gene expression patterns. *J. Comput. Biol.* 6, 281–297.

Cho, R., Campbell, M., Winzeler, E., et al. 1998. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell* 2, 65–73.

Clarkson, K.L. 2006. Nearest-neighbor searching and metric space dimensions, 15–59. In: *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*. Shakhnarovich, G., Darrell, T., Indyk, P. (eds), MIT Press, Cambridge, MA.

Draghici, S. 2003. *Data Analysis Tools for DNA Microarrays*. Chapman & Hall/CRC, Boca Raton, FL.

Dugas, M., Merk, S., Breit, S., et al. 2004. mdclust-Exploratory microarray analysis by multidimensional clustering. *Bioinformatics* 20, 931–936.

Eisen, M., Spellman, P., Brown, P., et al. 1998. Cluster analysis and display of genome-wide expression patterns. *Proc. Nat. Acad. Sci. USA* 1998, 14863–14868.

Ernst, J., Naur, G., and Bar-Joseph, Z. 2005. Clustering short time series gene expression. *Bioinformatics* 21, i159–i168.

Gat-Viks, I., Sharan, R., and Shamir, R. 2003. Scoring clustering solutions by their biological relevance. *Bioinformatics* 19, 2381–2389.

Geraci, F., Leoncini, M., Montangero, M., et al. 2007. *FPF-SB*: a scalable algorithm for microarray gene expression data clustering. *Lect. Notes Comput. Sci*, 4561, 606–615.

Geraci, F., Pellegrini, M., and Renda, M.E. 2008. AMIC@: All MIcroarray Clusterings @ once. *Nucleic Acids Res.* gkn265.

Geraci, F., Pellegrini, M., Pisati, P., et al. 2006. A scalable algorithm for high-quality clustering of web snippets. *Proc. 21st Annu. ACM Symp. Appl. Comput.* 1058–1062.

Gibbons, F., and Roth, F. 2000. Judging the quality of gene expression-based clustering methods using gene annotation. *Genome Res.* 12, 1574–1581.

Giurcaneanu, C., Tabus, I., Shmulevich, I., et al. 2003. Stability-based cluster analysis applied to microarray data. *Proc. 7th Int. Symp. Signal Process. Appl.* 57–60.

Gonzalez, T. 1985. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.* 38, 293–306.

Hanisch, D., Zien, A., Zimmer, R., et al. 2002. Co-clustering of Biological networks and gene expression data. *Bioinformatics* 18, 145S–154S.

Hastie, T., Tibshirani, R., and Eisen, M.B., et al. 2000. ''Gene shaving'' as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biol.* 1, research00.

Huang, W.P. 2006. Incorporating biological knowledge into distance-based clustering analysis of microarray gene expression data. *Bioinformatics* 22, 1259–1268.

Iyer, V.R., Eisen, M.B., Ross, D.T., et al. 1999. The transcriptional program in the response of human fibroblasts to serum. *Science* 283, 83–87.

Jain, A.K., Murty, M.N., and Flynn, P.J. 1999. Data clustering: a review. *ACM Comput. Surv.* 31, 264–323.

Jiang, D., Tang, C., and Zhang, A. 2004. Cluster analysis for gene expression data: a survey. *IEEE Trans. Knowledge Data Eng.* 16, 1370–1386.

Kerr, G., Ruskin, H.J., Crane, M., et al. 2008. Techniques for clustering gene expression data. *Comput. Biol. Med.* 38, 283–293.

Madeira, S., and Oliveira, A. 2004. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 1, 24–45.

Ng, E.K.K., Chee Fu, A.W., and Wong, R.C.W. 2005. Projective clustering by histograms. *IEEE Trans. Knowl. Data Eng.* 17, 369–383.

Ramoni, M., Sebastiani, P., and Kohane, I. 2002. Cluster analysis of gene expression dynamics. *Proc. Natl. Acad. Sci. USA* 99, 9121–9126.

Schadt, E., Edwards, S., GuhaThakurta, D., et al. 2004. A comprehensive transcript index of the human genome generated using microarrays and computational approaches. *Genome Biol.* 5, R73.

Shamir, R., and Sharan, R. 2002. Algoritmic approaches to clustering gene expression data, 269–299. In: *Current Topics in Computational Molecular Biology.* MIT Press, *Cambridge, MA*.

Sharan, R., Maron-Katz, A., and Shamir, R. 2003. CLICK and EXPANDER: a system for clustering and visualizing gene expression data. *Bioinformatics* 19, 1787–1799.

Spellman, P., Sherlock, G., Zhang, M., et al. 1998. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell.* 9, 3273–3297.

Tamayo, P., Slonim, D., Mesirov, J., et al. 1999. Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc. Nat. Acad. Sci. USA* 1999, 2907–2912.

Tanay, A., Sharan, R., and Shamir, R. 2006. Biclustering algorithms: a survey. In: *Handbook of Computational Molecular Biology*. Chapman and Hall/CRC Press, Boca Raton, FL.

Tavazoie, S., Hughes, J., Campbell, M., et al. 1999. Systematic determination of genetic network architecture. *Nat. Gene.*, 22, 281–285.

Taylor, J. 2006. RT: a tail strength measure for assessing the overall univariate significance in a dataset. *Biostatistics* 7, 167–181.

Tibshirani, R., Walther, G., and Hastie, T. 2001. Estimating the number of clusters in a data set via the gap statistic. *J. R. Statist. Soc. Ser. B* 63, 411–423.

Tibshirani, R., Walther, G., Botstein, D., et al. 2005. Cluster validation by prediction strength. *J. Comput. Graphical Statist.* 14, 511–528.

Trent, J., and Bexevanis, A. 2002. Chipping away at genomic medicine. *Nat. Genet. (Suppl.)* 426.

Wen, X., Fuhrman, S., Michaelsdagger, G.S., et al. 1988. Large-scale temporal gene expression mapping of central nervous system development. *Proc. Natl. Acad. Sci. USA* 95, 334–349.

Xing, E., and Karp, R. 2001. CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. *Bioinformatics* 17, 306–315.

Xu, Y., Olman, V., and Xu, D. 2002. Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. *Bioinformatics* 18, 536–545.

Yeung, K., Haynor, D., and Ruzzo, W. 2001. Validating clustering for gene expression data. *Bioinformatics* 17, 309–318.

Yip, K.Y., and Ng, M.K. 2004. HARP: a practical projected clustering algorithm. *IEEE Trans. Knowl. Data Eng.* 16, 1387–1397.

Address reprint requests to:
Dr. Marco Pellegrini
CNR
Istituto di Informatica e Telematica
via Moruzzi 1
56124 Pisa, Italy

E-Mail: marco.pellegrini@iit.cnr.it