two steps until no further changes occur in any of the k centroids:

- Assigning step: Assign each of n points to its nearest centroid.
- Updating step: Update each c of k centroids as the mean of points assigned to c.

Lloyd proposed the basic concept of the above procedure [34]. Suppose that it takes $\Theta(d)$ time to compute the distance between two d-dimensional points. A naïve implementation of the assigning step is to calculate the distance between each point and each centroid, which takes a $\Theta(dkn)$ time in total, while the updating step needs a $\Theta(dn)$ time. Thus, accelerating the assigning step is crucial. Here, we present a way of avoiding unnecessary computation in the assigning step by finding unchanged nearest centroids.

Selecting the distance between points is crucial in k-means clustering. The Euclidean and Pearson correlation distances are not always consistent and may produce different clustering results for an identical set of k initial centroids because during the assigning step, the centroid nearest to each vector can differ according to the distance selected. In contrast, the standardized Euclidean and Pearson correlation distances produce consistent orderings, and consequently the centroid closest to each vector is the same regardless of the distance selected. Using this property, we show that both distances yield the same clustering result.

Proposition. *For an identical set of k initial centroids, the k-means clustering algorithm produces the same clustering result for the standardized Euclidean distance as the Pearson correlation distance.*

Proof. We prove the inductive hypothesis stating that before each round of iteration, the set of k centroids for the standardized Euclidean distance is identical to that for the Pearson correlation distance. The hypothesis holds true before the first iteration simply because the same set of k initial centroids is the input for each distance. Assuming that the hypothesis is true before the ith iteration, after the assigning step, the centroid nearest to each vector is identical for each of the two distances because for any vector x and any centroids c_1 and c_2, $\mathrm{dis}(x, c_1) \leq \mathrm{dis}(x, c_2)$ if and only if $\mathrm{dis\_SE}(x, c_1) \leq \mathrm{dis\_SE}(x, c_2)$. Thus, after the updating step, the set of vectors closest to each centroid c is identical for the two distances, implying that the mean of the set, the revised centroid, is also identical. Consequently, the inductive hypothesis is true before the $(i+1)$ th iteration. □

This proposition allows us to perform k-means clustering with the Pearson correlation distance using optimization algorithms developed for the (standardized) Euclidean distance [41], [42], [44]; however, it is unclear whether the methods for the Euclidean distance are effective for accelerating the performance when using the standardized Euclidean distance. We show relevant experimental results in the next section.

For the following, we describe our new algorithm customized for the Pearson correlation distance. Centroids are updated frequently and are likely to move long distances in early stages of the repetitive steps. In contrast, in later steps, centroids are unlikely to move, and therefore, the assigning step has a tendency to reassign each point to the previous centroid as the nearest one, which should be avoided. Thus, we can accelerate the assigning step if we can test whether the nearest centroid for a point remains unchanged without recalculating the distances between the point and all centroids. Suppose that after the updating step, the centroid c_p nearest to x moves to c'_p for $p = 1, \ldots, k$, and any other centroid $c_q (q = 1, \ldots, k, q \neq p)$ moves to c'_q. We ask if x is still closest to cluster c'_p after the updating step:

$$\mathrm{dis}(c'_p, x) \leq \mathrm{dis}(c'_q, x),$$

for $q = 1, \ldots, k (q \neq p)$.

To check this test efficiently for any point x without recalculating the new distances on both sides of the inequality, we will develop an efficient method to estimate an upper bound of the new distance $\mathrm{dis}(c'_p, x)$ using the existing distance $\mathrm{dis}(c_p, x)$:

$$\mathrm{dis}(c'_p, x) \leq \mathrm{dis}(c_p, x) + \mathrm{an\_upper\_bound},$$

where we will define "an_upper_bound(≥ 0)" shortly. Similarly, we will derive a lower bound of $\mathrm{dis}(c'_q, x)$ using the previous distance $\mathrm{dis}(c_q, x)$:

$$\mathrm{dis}(c_q, x) + \mathrm{a\_lower\_bound} \leq \mathrm{dis}(c'_q, x)$$

for $q = 1, \ldots, k (q \neq p)$, where $\mathrm{a\_lower\_bound} \leq 0$.

Using these methods, we can implement a pruning procedure. If

$$\mathrm{dis}\,(c_p, x) + \mathrm{an\_upper\_bound} \leq \mathrm{dis}\,(c_q, x) + \mathrm{a\_lower\_bound} \quad \text{for } q = 1, \ldots, k (q \neq p), \quad (*)$$

we can confirm $\mathrm{dis}(c'_p, x) \leq \mathrm{dis}(c'_q, x) (q \neq p)$ without calculating the new distances, while retaining the final solution. In the next round of the assigning step, it might be necessary to calculate the new distances, but we can omit this step by substituting $\mathrm{dis}(c_p, x) + \mathrm{an\_upper\_bound}$ for new distances $\mathrm{dis}(c'_p, x)$ and $\mathrm{dis}(c'_q, x)$ respectively because this replacement does not violate the validity of the pruning procedure in the next assigning step. In cases in which the inequality (*) does not hold, we calculate $\mathrm{dis}(c'_p, x)$ and $\mathrm{dis}(c'_q, x)$ for $q = 1, \ldots, k (q \neq p)$, and determine the centroid nearest to x.

To facilitate the simple description of formula and derivations, we introduce a method of decomposing the Pearson's correlation coefficient $\rho(x, y)$ into two vectors called "correlation coefficient vectors."

Definition. *Correlation coefficient vectors are defined as*

$$\frac{1}{\sqrt{d}}\left(\frac{x[1]-\overline{x}}{\sigma_x}, \frac{x[2]-\overline{x}}{\sigma_x}, \cdots \frac{x[d]-\overline{x}}{\sigma_x}\right),$$

$$\frac{1}{\sqrt{d}}\left(\frac{y[1]-\overline{y}}{\sigma_y}, \frac{y[2]-\overline{y}}{\sigma_y}, \cdots \frac{y[d]-\overline{y}}{\sigma_y}\right),$$

for $x = (x[1], \ldots, x[d])$ and $y = (y[1], \ldots, y[d])$, respectively. Let CCx and CCy denote the respective correlation coefficient vectors.

Note that the Pearson's correlation coefficient $\rho(x, y)$ is equal to the inner product of CCx and CCy; i.e., $\rho(x, y) = (CCx, CCy)$. Any correlation coefficient vector CCx is of length 1; namely, $\| CCx \| = 1$, and similarly, $\| CCy \| = 1$.

To facilitate the discussion of calculating better upper and lower bounds, we introduce a new definition.

Definition. *Let c and c' be respective centroids before and after the updating step, and let CCc and CCc' be their correlation coefficient vectors. Let $\Delta\text{dis}(c, c', x)$ denote $\text{dis}(c', x) - \text{dis}(c, x)$, the distance variation of point x to c and c'.*

For example, $\text{dis}(c'_p, x) \leq \text{dis}(c_p, x) + \text{an\_upper\_bound}$ can be concisely described by

$$\Delta\text{dis}(c_p, c'_p, x) \leq \text{an\_upper\_bound}.$$

Another merit of this notation is that we are able to transform the distance variation into an inner product of $(CCc - CCc')$ and CCx:

$$\begin{aligned}\Delta\text{dis}(c_p, c'_p, x) &= \text{dis}(c'_p, x) - \text{dis}(c_p, x)\\ &= \rho(c_p, x) - \rho(c'_p, x),\\ &= (CCc_p, CCx) - (CCc'_p, CCx),\\ &= (CCc_p - CCc'_p, CCx).\end{aligned}$$

This inner product allows us to estimate an upper bound and a lower bound of $\Delta\text{dis}(c_p, c'_p, x)$ by analyzing the two vectors independently as well as by considering each dimension separately.

We can derive an upper bound and a lower bound that are effective for any point x for which the nearest centroid is c_p. A simple approach is to derive two bounds from

$$\begin{aligned}||\text{dis}(c_p, c'_p, x)|| &= ||(CCc_p - CCc'_p, CCx)||\\ &\leq ||CCc_p - CCc'_p||||CCx||,\end{aligned}$$

where the inequality holds because of the Cauchy-Schwarz inequality. Because $\|CCx\| = 1$, we can use $\|CCc_p - CCc'_p\|$ and $-\|CCc_p - CCc'_p\|$ as the upper and lower bounds, respectively, and we define them as follows:

Definition.

$$\text{upperA}(c_p, c'_p) \stackrel{def}{=} ||CCc_p - CCc'_p||$$

$$\text{lowerA}(c_p, c'_p) \stackrel{\text{def}}{=} -||CCc_p - CCc'_p||$$

These upper and lower bounds are simple formulas but effective for eliminating unnecessary computation. It takes $\Theta(dk)$ time to calculate the lower and upper bounds for all k centroids, and $\Theta(k)$ space to store these bounds. We also design more complicated bounds by taking the sum of the differences at individual coordinates.

Definition. *Let S_{c_p} denote the set of all points for which the nearest centroid is c_p*

$$\text{upperB}\ (c_p, c'_p, S_{c_p}) \stackrel{\text{def}}{=} \sum_{j=1}^{d} \text{maximum}(CCc_p[j] - CCc'_p[j], S_{c_p}),$$

where

$$\text{maximum}(z, S_{c_p}) \stackrel{\text{def}}{=} \begin{cases} z \times \max\{CCx[j] | x \in S_{c_p}\} & z \geq 0 \\ z \times \min\{CCx[j] | x \in S_{c_p}\} & z < 0 \end{cases}$$

For $q = 1, \ldots, k (q \neq p)$, define

$$\text{lowerB}\ (c_q, c'_q, S_{c_p}) \stackrel{\text{def}}{=} \sum_{j=1}^{d} \text{minimum}(CCc_q[j] - CCc'_q[j], S_{c_p}),$$

where

$$\text{minimum}(z, S_{c_p}) \stackrel{\text{def}}{=} \begin{cases} z \times \min\{CCx[j] | x \in S_{c_p}\} & z \geq 0 \\ z \times \max\{CCx[j] | x \in S_{c_p}\} & z < 0 \end{cases}.$$

Proposition. *For any $x \in S_{c_p}$*

$$\begin{aligned}&\Delta\text{dis}(c_p, c'_p, x) \leq \text{upperB}(c_p, c'_p, S_{c_p}) \quad \text{and}\\ &\text{lowerB}(c_q, c'_q, S_{c_p}) \leq \Delta\text{dis}(c_q, c'_q, x)(q \neq p).\end{aligned}$$

It takes $\Theta(dn + dk^2)$ time and $\Theta(dk + k^2)$ space in order to calculate $\text{upperB}(c_p, c'_p, S_{c_p})$ and $\text{lowerB}(c_q, c'_q, S_{c_p})(p = 1, \ldots, k, q = 1, \ldots, k, q \neq p)$ for every cluster c_p.

Proof.

$$\begin{aligned}\Delta\text{dis}(c_p, c'_p, x) &= ((CCc_p - CCc'_p), CCx)\\ &= \sum_{j=1}^{d} (CCc_p[j] - CCc'_p[j]) \times CCx[j]\\ &\leq \sum_{j=1}^{d} \text{maximum}(CCc_p[j] - CCc'_p[j], S_{c_p})\\ &= \text{upperB}(c_p, c'_p, S_{c_p})\\ \Delta\text{dis}(c_q, c'_q, x) &= ((CCc_q - CCc'_q), CCx)\\ &= \sum_{j=1}^{d} (CCc_q[j] - CCc'_q[j]) \times CCx[j]\\ &\geq \sum_{j=1}^{d} \text{minimum}(CCc_q[j] - CCc'_q[j], S_{c_p})\\ &= \text{lowerB}(c_q, c'_q, S_{c_p}).\end{aligned}$$

For efficiency, we first compute the maximum and minimum of $\{CCx[j] | x \in S_{c_p}\}$ for each dimension $j = 1, \ldots, d$ and for each cluster $c_p (p = 1, \ldots, k)$, and store this information in a table of size $\Theta(dk)$. This tabulation process takes $\Theta(dn)$ time. Looking up the table, it is possible to calculate $\text{upperB}(c_p, c'_p, S_{c_p})$ for any cluster c_p in $\Theta(d)$ time, and $\text{lowerB}\ (c_q, c'_q, S_{c_p})$ for $(k-1)$ clusters $c_q (q = 1, \ldots, k, q \neq p)$ in $\Theta(d(k-1))$ time. Repeating this calculation for each cluster $c_p = c_1, \ldots, c_k$ requires $\Theta(dk^2)$ time and $\Theta(k^2)$ space for storing upper and lower bounds. □

Using the above two calculations for upper and lower bounds, we devise the pruning procedure that checks

$$\text{dis}(c_p, x) + \text{upperA}(c_p, c'_p) \leq \text{dis}(c_q, x) + \text{lowerA}(c_q, c'_q),$$

or

$$\begin{aligned}&\text{dis}(c_p, x) + \text{upperB}(c_p, c'_p, S_{c_p})\\ &\quad \leq \text{dis}(c_q, x) + \text{lowerB}(c_q, c'_q, S_{c_p})\end{aligned}$$

TABLE 1
Comparison of the Asymptotic Overhead Spent by Calculating Lower and Upper Bounds in Addition to Lloyd's Algorithm in Terms of Time and Space Complexity

	time / iteration	memory
BoostKCP(boundA)	$\Theta(dk)$	$\Theta(k)$
BoostKCP(boundB)	$\Theta(dn + dk^2)$	$\Theta(dk + k^2)$
Elkan	$\Theta(dk^2)$	$\Theta(kn + k^2)$
Hamerly	$\Theta(dk^2)$	$\Theta(n)$

for each x of n points ($x \in S_{c_p}$ for each $p = 1, \ldots, k$) and for each $q = 1, \ldots, k (q \neq p)$. If x meets one of the inequalities, we can confirm $\mathrm{dis}(c'_p, x) \leq \mathrm{dis}(c'_q, x) (q \neq p)$ by skipping the calculation of the new distances. The total computation time of checking the above inequality is $\Theta(kn)$. Using upperB and lowerB requires additional computational time $\Theta(dn + dk^2)$ and space $\Theta(dk + k^2)$, which is constantly required to calculate the two bounds in each iteration. In contrast, computing upperA and lowerA needs $\Theta(dk)$ time and $\Theta(k)$ space.

For each x that violates the above inequality, we compute new distances $\mathrm{dis}(c'_p, x)$ and $\mathrm{dis}(c'_q, x)$ for $q = 1, \ldots, k (q \neq p)$ to find the centroid nearest to x. In the best case, no calculation is needed. In the worst case, however, it is necessary to compute new distances $\mathrm{dis}(c'_p, x)$ for $p = 1, \ldots, k$ and n points, and the worst time complexity is $O(dkn)$. Recall for comparison that the assigning step of Lloyd's algorithm requires $\Theta(dkn)$ time.

We have defined two heuristic algorithms: one uses upperA and lowerA, and the other upperB and lowerB to prune unnecessary computations when performing k-means clustering using the Pearson correlation distance. We call the former BoostKCP (boundA) and the latter BoostKCP (boundB), where BoostKCP stands for Boosting K-means Clustering for Pearson correlation distance.

We compare the performance of Elkan's and Hamerly's methods, BoostKCP(boundA), and BoostKCP(boundB) with respect to time and space complexity. Although individual method accelerates Lloyd's algorithm using lower and upper bounds to prune unnecessary computation, each iteration requires $O(dkn)$ time in the worst case. Thus, we summarize the overhead of computing lower and upper bounds in terms of time and space complexity (Table 1). The entries of "time/iteration" show the asymptotic overhead computation time required to calculate lower and upper bounds in each iteration by individual algorithms. The entries for BoostKCP have been described, while those for Elkan's and Hamerly's algorithms are detailed in [42]. Table 1 shows that the time and space complexity of BoostKCP(boundA) are smaller than those of the other methods. In the experimental results, we will show that BoostKCP(boundA) also outperforms the others in terms of computational performance using real biological data sets, confirming that BoostKCP(boundA) is a simple and powerful heuristic method for accelerating k-means clustering when using Pearson correlation and standardized Euclidean distances.

3 Experimental Results

3.1 Data Sets

We generated a synthetic data set of vectors whose elements were randomly selected from 0 to 1 using the Mersenne twister [46], a widely used pseudorandom number generator with an extraordinarily long cycle of $2^{19.937}$-1. We generated data sets of 50,000 vectors of dimension $d =$ 10, 20, 50, 101, 201, 501, 1,001, and 2,001. This random data set was an extreme example from which meaningful clusters were difficult to extract. We used these sets to compare the effectiveness of BoostKCP (boundA) and BoostKCP (boundB) for pruning unnecessary computation.

In order to compare BoostKCP with other available state-of-the-art pruning methods, we used three different types of high-dimensional real biological data sets rather than random data sets. The first real data set was a set of vectors with human nucleosome positioning signals at genomic positions surrounding transcription start sites (TSSs). A nucleosome positioning signal at a genomic position is a real value and represents the possibility of the presence of nucleosome centers at that position. From the GENCODE database (version 7) [47], we obtained human nucleosome positioning signals using MNase-sequencing and the TSSs of the human reference genome hg19. We repeated the process of merging neighboring TSSs within 1,000 bp into a group, and we selected representative TSSs whose expression levels were maximal in individual groups. From the representative TSSs, we excluded those having any other TSSs within 1,000 bp on the reverse strand to eliminate their effect. Subsequently, from the nucleosome positioning signal data, we generated a base set of 56,772 vectors of dimension 2,001 (~400M bytes) such that their elements were real-valued nucleosome positioning signals within 1,000 bp around representative TSSs and more than half of the elements within 50, 100, 250, and 500 bp of the TSSs were nonzero. To monitor how the algorithms behave for data of different dimension, from the base set, we generated sets of vectors of dimension $d =$ 101, 201, 501, 1,001, and 2,001 by selecting the elements within 50, 100, 250, 500, and 1,000 bp of the TSSs. The last digit "1" of dimension d indicates the TSS position. Because of the construction of the base set, more than half of the elements in each vector are guaranteed to be nonzero. For smaller dimensions $d = 10, 20$, and 50, we selected every $(2000/d)$-th element from the base set; e.g., elements at $-1{,}000, -800, -600, \ldots, +600$, and $+800$ bp for $d = 10$. The second real data set was a typical example of gene expression data, a set of 54,613 genes from 180 glioma samples [48]. The third real data set was a set of 60,000 gray-level images of handwritten letters in the MNIST database [49]. Each image consisted of 28 x 28 pixels, and we set dimension $d = 28^2 = 784$. As letters were categorized into 78 types, we set $k = 78$.

3.2 Comparison of Computational Performance

We compared the following five methods:

- Lloyd's algorithm [34].

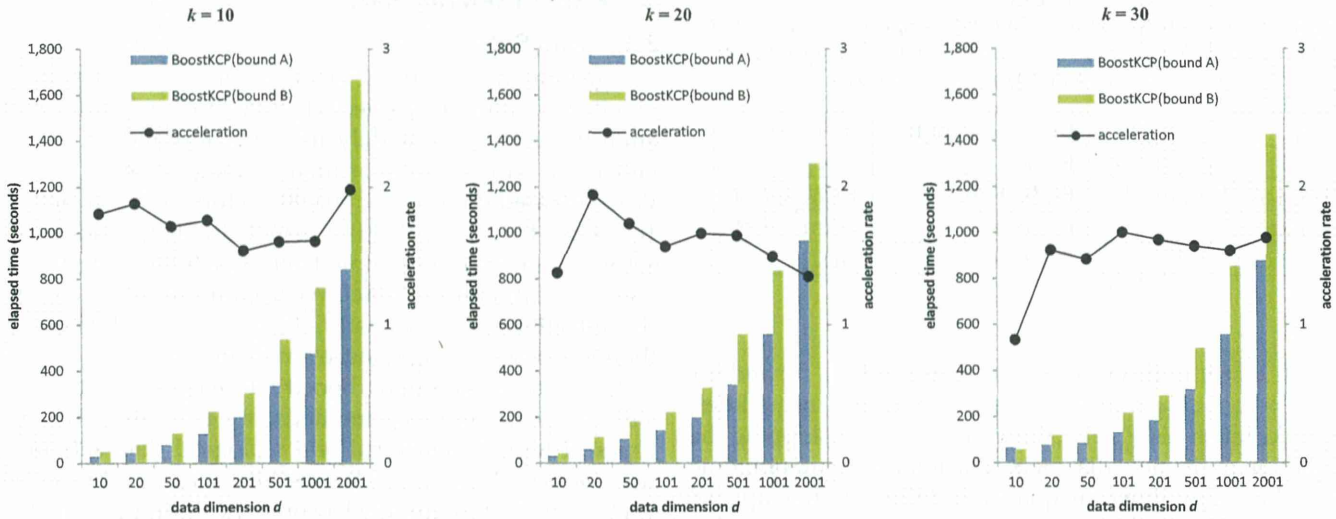


Fig. 1. Comparison between BoostKCP (boundA) and BoostKCP (boundB). Randomly generated 50,000 vectors of dimension $d = 10$, 20, 50, 101, 201, 501, 1,001, and 2,001 were grouped into k (= 10, 20, and 30) clusters. The first y-axis and second y-axis show the elapsed time and acceleration rate, respectively.

- BoostKCP (boundA).
- BoostKCP (boundB).
- Elkan's algorithm [41].
- Hamerley's algorithm [42], [43].

We used the first three methods to compute k-means clustering using the Pearson correlation distance. In contrast, since the latter two algorithms were designed to process the Euclidean distance, we used these to calculate k-means clustering using the standardized Euclidean distance, the results of which are equal to those using Pearson correlation distance as described in the previous section. For any initial centroid set, the above five methods give the same final clustering result.

Selecting the initial set of k centroids largely affects the final result, and for this purpose, we used Bradley and Fayyad's method [31] because it performed better than the other applicable initialization methods for several criteria [26]. After selecting the initial centroids, we measured the elapsed time during the application of each method towards the same initial centroid set derived from different types of data. We excluded the time required to compute the initial set of centroids because it was typically much less than the time used to compute k-means clustering. We monitored the computational performance using an Intel Xeon CPU E5-2670 processor with a clock rate of 2.60 GHz and 66 GB of main memory.

We first compared the performances of BoostKCP (boundA) and BoostKCP (boundB) using 50,000 random vectors of dimension $d = 10$, 20, 50, 101, 201, 501, 1,001, and 2,001. We calculated the average elapsed time by executing 10 trials for $d = 10, 20, 50, 101, 201, 501$, and 1,001, but five trials for $d = 2{,}001$, due to the large amount of computation. We observed that BoostKCP (boundA) outperformed BoostKCP (boundB). Specifically, we calculated the performance improvement by BoostKCP(boundA) as the acceleration rate; i.e., the elapsed time for BoostKCP (boundB) divided by that for BoostKCP (boundA). Fig. 1 displays the elapsed time and acceleration rate for each dimension and for $k = 10, 20$, and 30. In all cases except where $d = 10$ and $k = 30$, BoostKCP (boundA) was faster than BoostKCP (boundB) partly because computing lower and upper bounds for BoostKCP(boundA), $\Theta(dk)$, is less expensive than computing those for BoostKCP(boundB), $\Theta(dn + dk^2)$, where d is the dimension, n is the number of data, and k is the number of clusters (Table 1). We therefore used BoostKCP (boundA) for our comparisons with the other four algorithms using real data sets.

We next compared BoostKCP (boundA) with Lloyd's, Elkan's, and Hamerly's algorithms using real biological data sets. For measuring the performance improvement by BoostKCP(boundA), we again defined the acceleration rate as the average elapsed time of each algorithm divided by that of BoostKCP (boundA).

Fig. 2 shows the experimental results obtained by applying the four algorithms to the nucleosome positioning data for dimension $d = 10, 20, 50, 101, 201, 501, 1{,}001$ and 2,001 and for number of clusters $k = 10$, 20, and 30. We set these values for k because nucleosome positioning signal vectors can be categorized into 10–30 groups with biologically meaningful characteristics [24]. We computed the average elapsed time by performing 10 trials with the exception of five trials where $d = 2{,}001$. Figs. 2A, 2B, 2C show the BoostKCP (boundA) acceleration rates compared with those of the Lloyd's, Elkan's, and Hamerly's algorithms. BoostKCP (bound A) clearly outperformed Lloyd's and Hamerly's algorithms for all parameter value combinations, and it was also faster than Elkan's algorithm in most cases.

It has been reported that Hamerly's algorithm is often faster than Elkan's algorithm for various low-dimensional ($d < 50$) data using the Euclidean distance [42], [43]; however, Hamerly's algorithm did not work as well for nucleosome positioning data using the standardized Euclidean distance (Figs. 2A, 2B, 2C). We remark here that the standardized Euclidean distance between two points is likely to be much smaller than the Euclidean distance between the two points, implying that the points are densely distributed in standardized Euclidean space. When handling more densely distributed points, greater care has to be taken for pruning unnecessary computation. In each iteration, Elkan's algorithm carefully maintains the lower and upper bounds

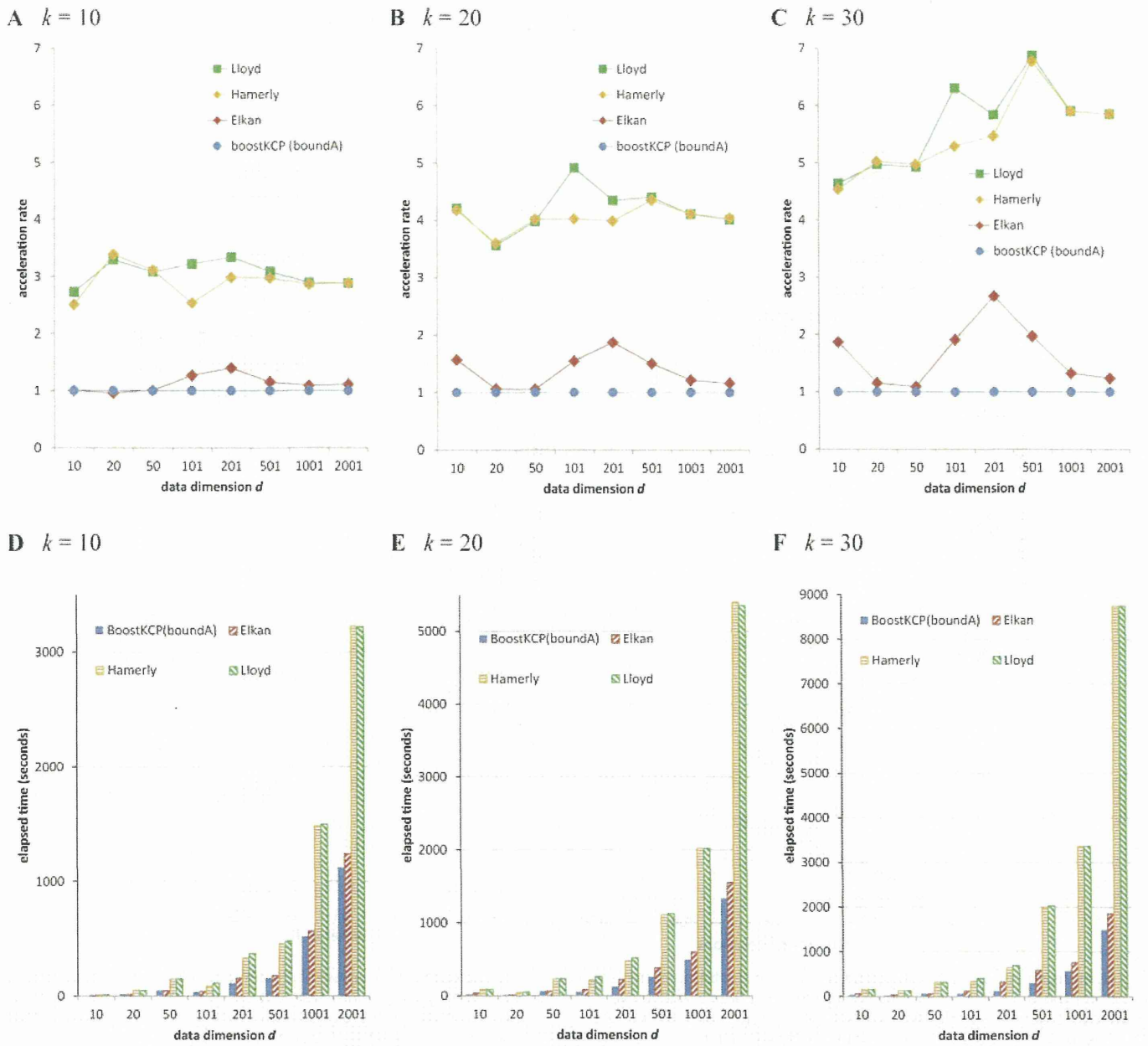


Fig. 2. Performance improvement by BoostKCP (boundA) using nucleosome positioning data of dimension $d = 10, 20, 50, 101, 201, 501, 1{,}001$, and 2,001. (A-C) Acceleration rates by BoostKCP (boundA) for each of Lloyd's, Hamerly's, and Elkan's algorithms. The lines for BoostKCP(boundA) show the constant rate of 1, the elapsed time for BoostKCP (boundA) divided by itself. Nucleosome positioning data were grouped into k clusters where $k = 10$ (A), 20 (B), and 30 (C). To make the comparison fair, we supplied all the algorithms with the same set of initial centroids that we generated using Bradley and Fayyad's method. (D-F) The average elapsed time of BoostKCP (boundA), Lloyd's, Hamerly's, and Elkan's algorithms.

for the distance between each point and each centroid, while Hamerly's algorithm considers the closest and second closest centroids only. For pruning unnecessary computation, put another way, Elkan's algorithm requires more time and space to estimate tighter bounds than does Hamerly's algorithm, allowing the former to be more effective in removing unnecessary computation than the latter.

Figs. 2D, 2E, and 2F display the average elapsed time when using each combination of d and k values; however, there is insufficient information as to how these times differed, since the elapsed time in each trial largely depended on the selection of the initial k vectors. To understand this further, we investigated how the elapsed time in each trial changed depending on the number of iterations when we applied BoostKCP (boundA), Elkan's, and Lloyd's algorithms to the nucleosome positioning signal data of dimension $d = 501$ for $k = 10, 20$, and 30. We did not consider Hamerly's algorithm because its performance was similar to that of Lloyd. Fig. 3A shows that how elapsed time of individual algorithm changes for ten different initial sets of centroids. The figure shows that the elapsed time of each algorithm increased in proportion to the number of iterations. A major difference between the three algorithms was that the elapsed time of Elkan's and Lloyd's algorithms increased for larger values of k, but that of our pruning method was almost independent of k, which explains why the acceleration rate increased for larger values of k, as seen in Fig. 2.

To gain a better understanding of this, Fig. 3B presents an in-depth analysis, showing the elapsed time in each

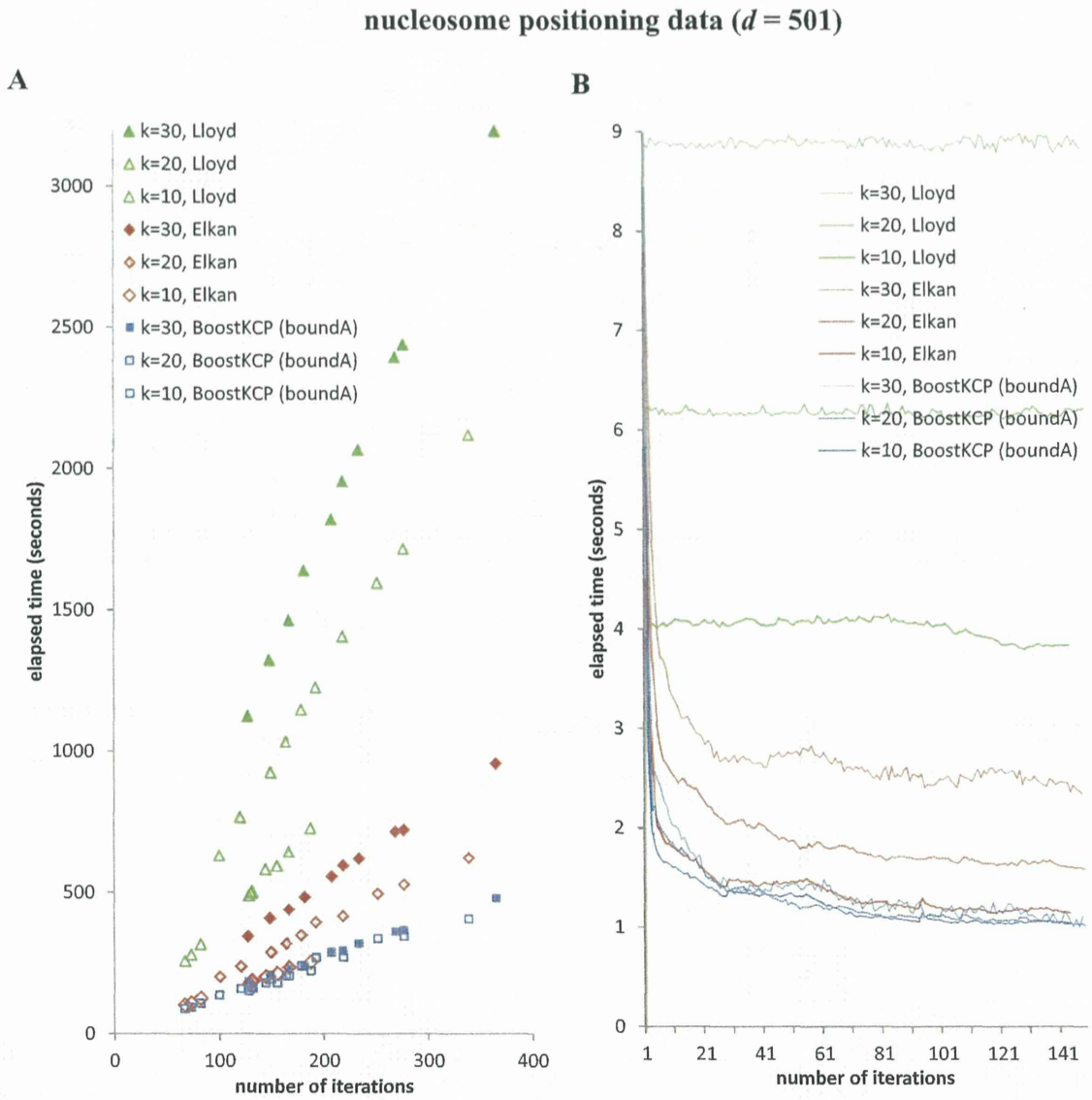


Fig. 3. In-depth performance analysis on k-means clustering of nucleosome positioning data. (A) Analysis of clustering nucleosome positioning data of dimension $d = 501$ by BoostKCP (boundA), Elkan's and Lloyd's algorithms. Hamerly's algorithm was not considered because Lloyd's and Hamerly's algorithms performed similarly. A dot represents the number of iterations (x-axis) and the elapsed time (seconds) of each experiment of 10 trials for $k = 10$, 20 and 30. (B) Elapsed time of each iteration (including the assigning and updating steps) in typical trials.

iteration of the three algorithms. Each iteration time for Lloyd's algorithm is almost constant because the algorithm does not avoid unnecessary computation, while each iteration time for BoostKCP (boundA) and Elkan's algorithm for $k = 10, 20$, and 30 decreased markedly after the first few steps. In later steps, the elapsed time of BoostKCP (boundA) became almost independent of the value of k, giving the account that its overall elapsed time was almost proportional to the number of iterations but independent of k, as shown in Fig. 3A. In contrast, the elapsed time of Elkan's algorithm in each iteration increased for larger values of k. This is because in each iteration, Elkan's algorithm maintains a large array of lower and upper bounds for the distance between each ~56K points and each k centroid at an expense. In contrast, BoostKCP (boundA) needs to calculate only the lower and upper bounds for each k centroid (Table 1).

We then applied the three algorithms to the gene expression data, a set of 54613 vectors of dimension $d = 180$. Because the dimension was fixed, we grouped the data into k ($= 2, 3, 10, 20, 30, , 70$) clusters of genes to determine if BoostKCP (boundA) achieved better performance with larger values of k. Fig. 4 shows the average elapsed time for ten trials and the acceleration rate of BoostKCP (boundA). The three algorithms used Bradley and Fayyad's method to generate the same set of initial centroids. BoostKCP (boundA) outperformed Elkan's and Lloyd's algorithms for each k except for the case that the acceleration rate by BoostKCP (boundA) for Elkan's algorithm was 0.988 when $k = 2$. The acceleration rates were 1.02, 1.13, and 1.32 when $k = 3$, 10, and 20, respectively. The acceleration rate increased for larger values of k, which was consistent with the performance improvement that we observed for the nucleosome positioning data in Fig. 2.

We also applied BoostKCP (boundA) and Elkan's algorithm to a data set of handwritten letters ($d = 784$) to obtain 78 ($= k$) groups (different letters). The average

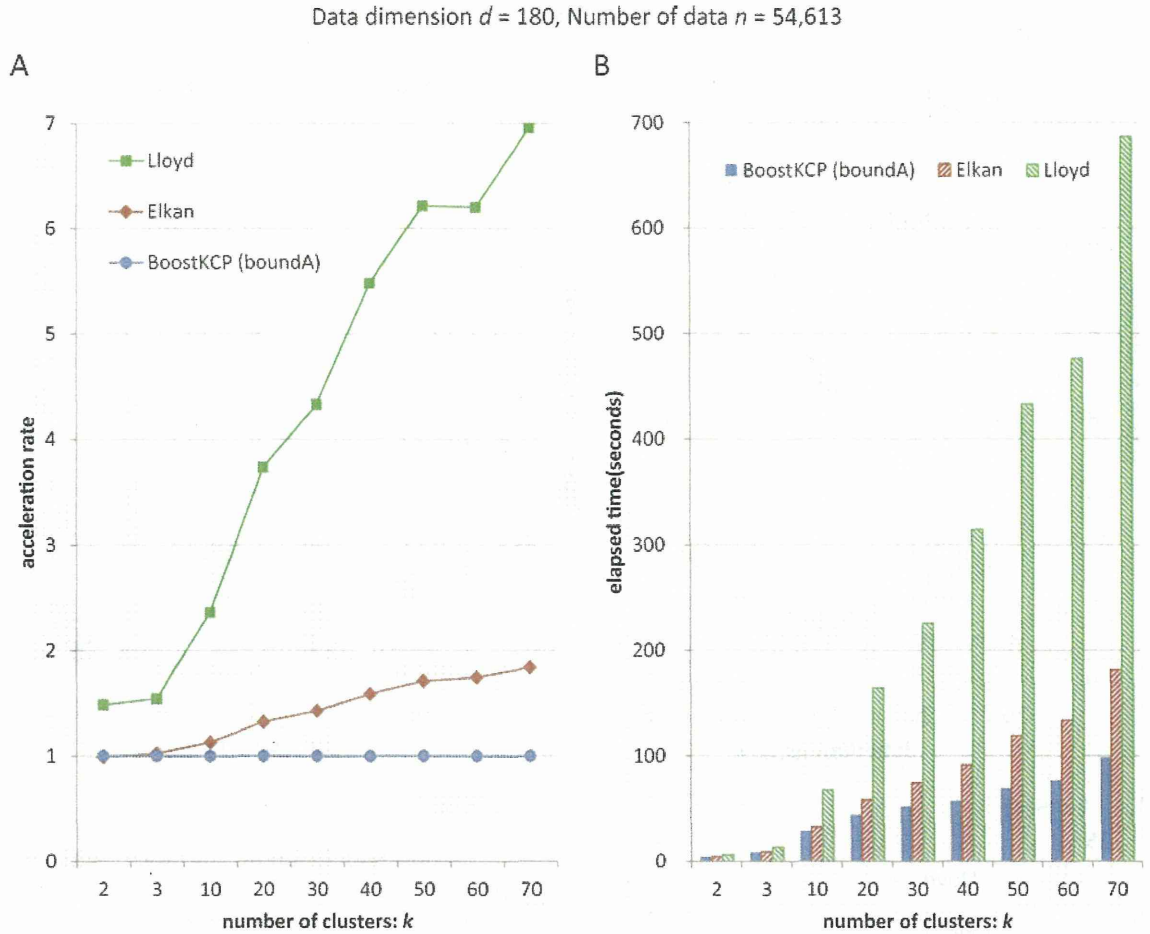


Fig. 4. Performance improvement by BoostKCP (boundA) using gene expression data of dimension $d = 180$ to group the data into k (=2, 3, 10, 20, 30, ..., 70) clusters. (A) Acceleration rates by BoostKCP (boundA) for each of Elkan's and Lloyd's algorithms. (B) Average elapsed time of 10 trials.

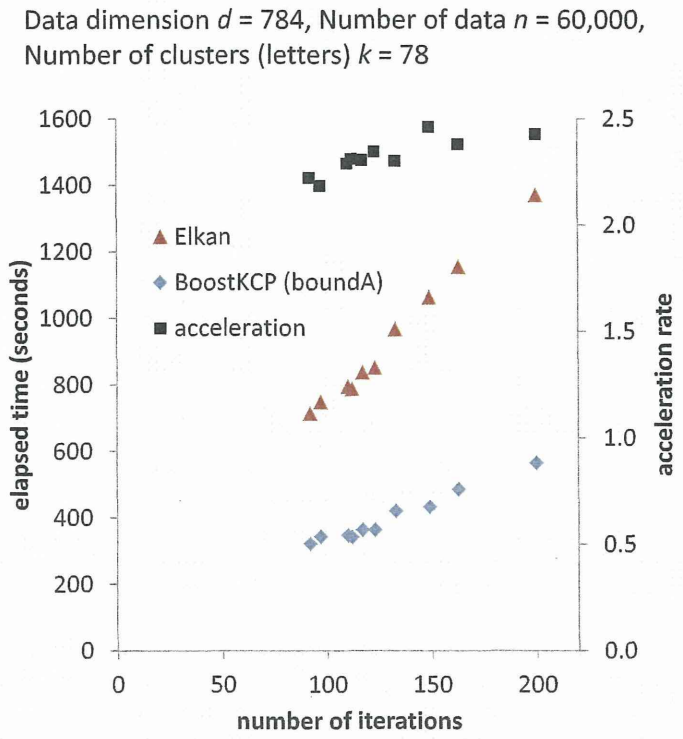


Fig. 5. The elapsed time, acceleration rate, and number of iterations of each of ten attempts to cluster handwritten letter images of dimension 784 (=d) into 78 (=k) groups using BoostKCP (boundA) and Elkan's algorithm.

acceleration rate of the 10 trials was high (2.18 – 2.46) presumably because the number of clusters was large. Fig. 5 shows the elapsed time, acceleration rate, and number of iterations for each of the ten trials. The iteration numbers are likely to be smaller than those in Fig. 3A because the images of the handwritten letters are grouped inherently. In general, the number of iterations depends on individual data, and it tends to be smaller when the focal data have inherently discriminating groups of similar vectors that are relatively easier to categorize. In contrast, randomly generated data avoid this data skewness; thus, the algorithms spend more time searching for centroids.

We have so far examined situations when the number of clusters (k) ranges from two to 78 simply because these numbers of groups are of interest in real biological applications. We here investigate whether BoostKCP (boundA) outperforms Elkan's and Lloyd's algorithms for larger values of k, such as $k = 100$ and 500. Indeed, Fig. 6 illustrates that BoostKCP (boundA) was the winner when the three algorithms were used to cluster the nucleosome positioning data of dimension $d = 10$, 20, 50, 101, and 201 into $k = 100$ and 500 groups.

4 Conclusion

High-dimensional data, such as epigenome data, nucleosome positioning, and gene expression patterns, are quite common in biological research. K-means clustering

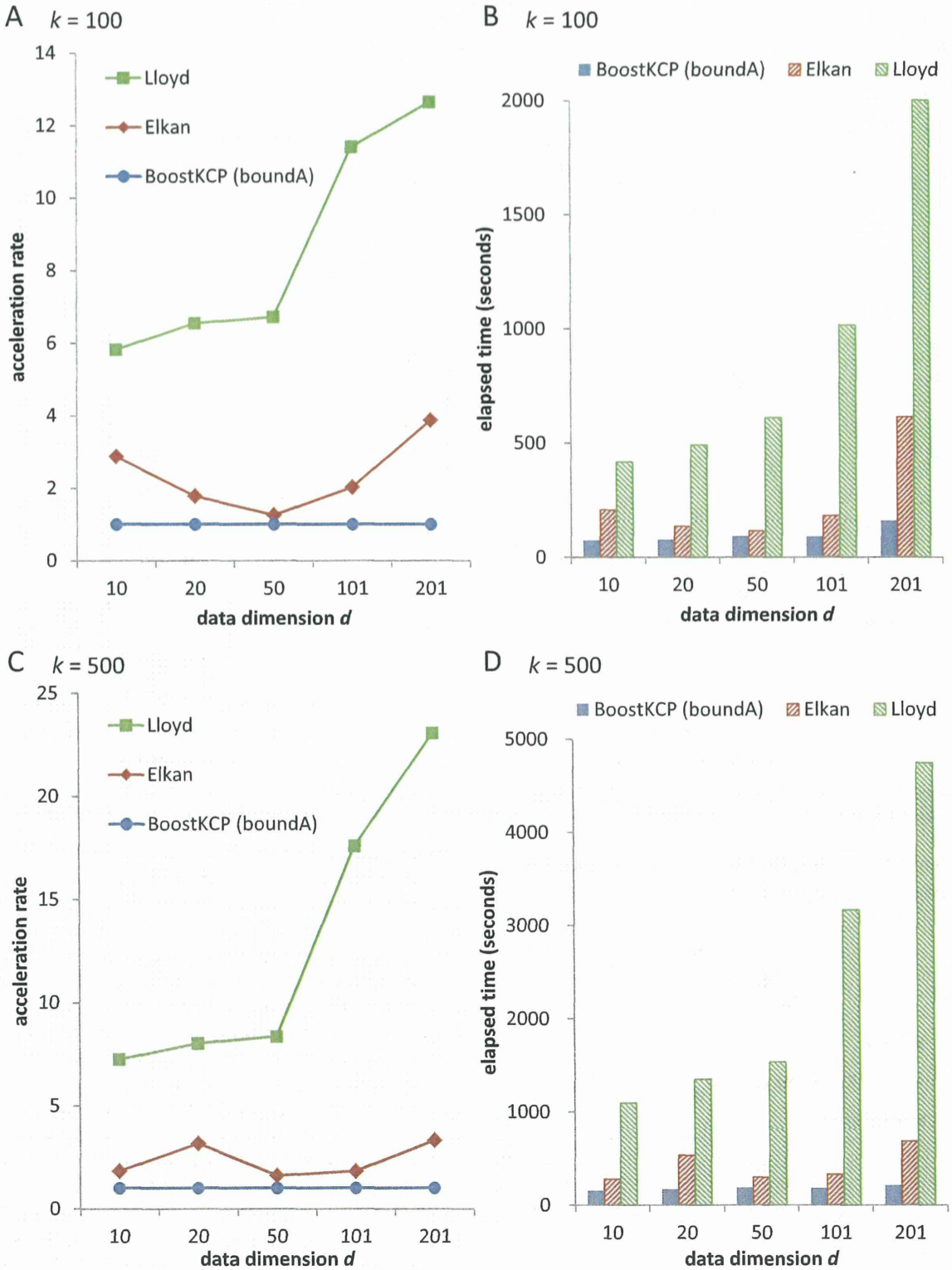


Fig. 6. Performance improvement by BoostKCP (boundA) using nucleosome positioning data of dimension $d = 10, 20, 50, 101$, and 201 to group the data into $k = 100$ and 500 clusters. (A, C) Acceleration rates by BoostKCP (boundA) for each of Elkan's and Lloyd's algorithms when $k = 100$ (A) and $k = 500$ (C). (B, D) Average elapsed time of ten trials for BoostKCP (boundA), Elkan's, and Lloyd's algorithms when $k = 100$ (B) and $k = 500$ (D).

using the Pearson correlation and standardized Euclidean distances has proven useful for obtaining novel insight from such large-scale biological data sets; however, it is likely to be a computationally intense task, thus demanding a method for accelerating computational performance for high-dimensional biological data. We have addressed the problem of eliminating unnecessary calculations associated with the k-means clustering algorithm. In this paper, we introduced BoostKCP, a simple but powerful heuristic method that has proved useful for reducing the computational time. We applied BoostKCP to three types of real biological data sets of dimension $d = 10, 20, 50, 101, 180, 201, 501, 784, 1{,}001$ and $2{,}001$ to perform k-clustering for $k = 2, 3, 10, 20, 30, 40, 50, 60, 70, 78, 100$, and 500.

BoostKCP outperformed Elkan's, Lloyd's, and Hamerly's algorithms in most cases. Our concept is also applicable to k-medians clustering, which uses the median of points in a cluster as the cluster representative, and this method is applied frequently to generate tight clusters.

Acknowledgments

The authors thank Yuta Suzuki for his valuable suggestions. This work was supported in part by MEXT (Grant-in-Aid for Scientific Research A 23241058, Global COE program "Deciphering Biosphere from Genome Big Bang", Innovative Cell Biology by Innovative Technology).

References

[1] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," in *Proc. Natl. Acad. Sci. USA*, 1998, vol. 95, no. 25, pp. 14863–14868.

[2] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub, "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation," in *Proc. Natl. Acad. Sci. USA*, vol. 96, no. 6, 1999, pp. 2907–2912, 1999.

[3] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: A survey," *IEEE Trans. Knowl. and Data Eng.*, vol. 16, no. 11, pp. 1370–1386, Nov. 2004.

[4] P. D'haeseleer, "How does gene expression clustering work?" *Nat. Biotechnol.*, vol. 23, pp. 1499–501, 2005.

[5] T. S. Mikkelsen, M. Ku, D. B. Jaffe, B. Issac, E. Lieberman, G. Giannoukos, P. Alvarez, W. Brockman, T.-K. Kim, R. P. Koche, W. Lee, E. Mendenhall, A. O'Donovan, A. Presser, C. Russ, X. Xie, A. Meissner, M. Wernig, R. Jaenisch, C. Nusbaum, E. S. Lander, and B. E. Bernstein, "Genome-wide maps of chromatin state in pluripotent and lineage-committed cells," *Nature*, vol. 448, pp. 553–60, 2007.

[6] N. D. Heintzman, G. C. Hon, R. D. Hawkins, P. Kheradpour, A. Stark, L. F. Harp, Z. Ye, L. K. Lee, R. K. Stuart, and C. W. Ching, "Histone modifications at human enhancers reflect global cell-type-specific gene expression," *Nature*, vol. 459, no. 7243, pp. 108–112, 2009.

[7] P. V. Kharchenko, A. A. Alekseyenko, Y. B. Schwartz, A. Minoda, N. C. Riddle, J. Ernst, P. J. Sabo, E. Larschan, A. A. Gorchakov, and T. Gu, "Comprehensive analysis of the chromatin landscape in Drosophila melanogaster," *Nature*, vol. 471, no. 7339, pp. 480–485, 2010.

[8] S. Roy, J. Ernst, P. V. Kharchenko, P. Kheradpour, N. Negre, M. L. Eaton, J. M. Landolin, C. A. Bristow, L. Ma, and M. F. Lin, "Identification of functional elements and regulatory circuits by Drosophila modENCODE," *Science*, vol. 330, no. 6012, pp. 1787–1797, 2010.

[9] L. Handoko, H. Xu, G. Li, C. Y. Ngan, E. Chew, M. Schnapp, C. W. H. Lee, C. Ye, J. L. H. Ping, F. Mulawadi, E. Wong, J. Sheng, Y. Zhang, T. Poh, C. S. Chan, G. Kunarso, A. Shahab, G. Bourque, V. Cacheux-Rataboul, W.-K. Sung, Y. Ruan, and C.-L. Wei, "CTCF-mediated functional chromatin interactome in pluripotent cells," *Nat. Genetics*, vol. 43, pp. 630–8, 2011.

[10] T. Liu, A. Rechtsteiner, T. A. Egelhofer, A. Vielle, I. Latorre, M. S. Cheung, S. Ercan, K. Ikegami, M. Jensen, and P. Kolasinska-Zwierz, "Broad chromosomal domains of histone modification patterns in C. elegans," *Genome Res.*, vol. 21, no. 2, pp. 227–236, 2011.

[11] J. Ernst and M. Kellis, "ChromHMM: Automating chromatin-state discovery and characterization," *Nat. Methods*, vol. 9, no. 3, pp. 215–216, 2012.

[12] M. M. Hoffman, O. J. Buske, J. Wang, Z. Weng, J. A. Bilmes, and W. S. Noble, "Unsupervised pattern discovery in human chromatin structure through genomic segmentation," *Nat. Methods*, vol. 9, no. 5, pp. 473–476, 2012.

[13] J. R. Dixon, S. Selvaraj, F. Yue, A. Kim, Y. Li, Y. Shen, M. Hu, J. S. Liu, and B. Ren, "Topological domains in mammalian genomes identified by analysis of chromatin interactions," *Nature*, vol. 485, no. 7398, pp. 376–380, 2012.

[14] S. M. Johnson, F. J. Tan, H. L. McCullough, D. P. Riordan, and A. Z. Fire, "Flexibility and constraint in the nucleosome core landscape of Caenorhabditis elegans chromatin," *Genome Res.*, vol. 16, no. 12, pp. 1505–1516, 2006.

[15] W. Lee, D. Tillo, N. Bray, R. H. Morse, R. W. Davis, T. R. Hughes, and C. Nislow, "A high-resolution atlas of nucleosome occupancy in yeast," *Nat. Genetics*, vol. 39, pp. 1235–1244, 2007.

[16] I. Whitehouse, O. J. Rando, J. Delrow, and T. Tsukiyama, "Chromatin remodelling at promoters suppresses antisense transcription," *Nature*, vol. 450, pp. 1031–5, 2007.

[17] A. Valouev, J. Ichikawa, T. Tonthat, J. Stuart, S. Ranade, H. Peckham, K. Zeng, J. a. Malek, G. Costa, K. McKernan, A. Sidow, A. Fire, and S. M. Johnson, "A high-resolution, nucleosome position map of C. Elegans reveals a lack of universal sequence-dictated positioning," *Genome Res.*, vol. 18, pp. 1051–63, 2008.

[18] T. N. Mavrich, C. Jiang, I. P. Ioshikhes, X. Li, B. J. Venters, S. J. Zanton, L. P. Tomsho, J. Qi, R. L. Glaser, S. C. Schuster, D. S. Gilmour, I. Albert, and B. F. Pugh, "Nucleosome organization in the Drosophila genome," *Nature*, vol. 453, pp. 358–362, 2008.

[19] R. K. Chodavarapu, S. Feng, Y. V. Bernatavichute, P.-Y. Chen, H. Stroud, Y. Yu, J. a. Hetzel, F. Kuo, J. Kim, S. J. Cokus, D. Casero, M. Bernal, P. Huijser, A. T. Clark, U. Krämer, S. S. Merchant, X. Zhang, S. E. Jacobsen, and M. Pellegrini, "Relationship between nucleosome positioning and DNA methylation," *Nature*, vol. 466, pp. 388–92, 2010.

[20] A. Valouev, S. M. Johnson, S. D. Boyd, C. L. Smith, A. Z. Fire, and A. Sidow, "Determinants of nucleosome organization in primary human cells," *Nature*, vol. 474, pp. 516–520, 2011.

[21] X. Wang, G. O. Bryant, M. Floer, D. Spagna, and M. Ptashne, "An effect of DNA sequence on nucleosome occupancy and removal," *Nat. Publishing Group*, vol. 18, pp. 507–509, 2011.

[22] Z. Zhang, C. J. Wippo, M. Wal, E. Ward, P. Korber, and B. F. Pugh, "A packing mechanism for nucleosome organization reconstituted across a eukaryotic genome," *Science*, vol. 332, pp. 977–80, 2011.

[23] J. Wang, J. Zhuang, S. Iyer, X. Lin, T. W. Whitfield, M. C. Greven, B. G. Pierce, X. Dong, A. Kundaje, Y. Cheng, O. J. Rando, E. Birney, R. M. Myers, W. S. Noble, M. Snyder, and Z. Weng, "Sequence features and chromatin structure around the genomic regions bound by 119 human transcription factors," *Genome Res.*, vol. 22, pp. 1798–1812, 2012.

[24] A. Kundaje, S. Kyriazopoulou-Panagiotopoulou, M. Libbrecht, C. L. Smith, D. Raha, E. E. Winters, S. M. Johnson, M. Snyder, S. Batzoglou, and A. Sidow, "Ubiquitous heterogeneity and asymmetry of the chromatin environment at regulatory elements," *Genome Res.*, vol. 22, pp. 1735–1747, 2012.

[25] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recog. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[26] H. A. Kingravi, M. E. Celebi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," *Expert Syst. Appl.*, vol. 40, pp. 200–120, 2012.

[27] E. W. Forgy, "Cluster analysis of multivariate data: Efficiency versus interpretability of classifications," *Biometrics*, vol. 21, pp. 768–769, 1965.

[28] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.

[29] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Comput. Sci.*, vol. 38, pp. 293–306, 1985.

[30] I. Katsavounidis, C. C. Jay Kuo, and Z. Zhang, "A new initialization technique for generalized Lloyd iteration," *IEEE Signal Process. Lett.*, vol. 1, no. 10, pp. 144–146, Oct. 1994.

[31] P. S. Bradley and U. M. Fayyad, "Refining initial points for k-means clustering," in *Proc. 15th Int. Conf. Mach. Learn.*, 1998, pp. 91–99.

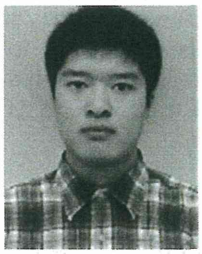
[32] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2007, pp. 1027–1035.

[33] T. Su and J. G. Dy, "In search of deterministic methods for initializing K-means and Gaussian mixture clustering," *Intell. Data Anal.*, vol. 11, no. 4, pp. 319–338, 2007.

[34] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[35] F. D. Gibbons and F. P. Roth, "Judging the quality of gene expression-based clustering methods using gene annotation," *Genome Res.*, vol. 12, no. 10, pp. 1574–1581, Oct. 2002.

[36] F. Geraci, M. Leoncini, M. Montangero, M. Pellegrini, and M. E. Renda, "K-Boost: A scalable algorithm for high-quality clustering of microarray gene expression data," *J. Comput. Biol.*, vol. 16, no. 6, pp. 859–873, Jun. 2009.
[37] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church, "Systematic determination of genetic network architecture," *Nat. Geneicst*, vol. 22, no. 3, pp. 281–285, Jul. 1999.
[38] R. Sharan and R. Shamir, "CLICK: A clustering algorithm with applications to gene expression analysis," *in Proc. Int. Conf. Intell. Syst. Molecular Biol.*, vol. 8, 2000, pp. 307–316.
[39] F. De Smet, J. Mathys, K. Marchal, G. Thijs, B. De Moor, and Y. Moreau, "Adaptive quality-based clustering of gene expression profiles," *Bioinformatics*, vol. 18, no. 5, pp. 735–746, May 2002.
[40] K. L. Clarkson, "Nearest-neighbor searching and metric space dimensions," *Nearest-Neighbor Methods Learning Vis.: Theory Practice*, pp. 15–59, 2006.
[41] C. Elkan, "Using the triangle inequality to accelerate k-means," in *Proc. Int. Conf. Mach. Learn.*, 2003, p. 147-153.
[42] G. Hamerly, "Making k-means even faster," in *Proc. Symp. Data Mining*, 2010, pp. 130–140.
[43] J. Drake and G. Hamerly, "Accelerated k-means with adaptive distance bounds," in *Proc. 5th NIPS Workshop Optimization Mach. Learn.*, 2012.
[44] V. C. Osamor, E. F. Adebiyi, J. O. Oyelade, and S. Doumbia, "Reducing the time requirement of k-means algorithm," *PLoS One*, vol. 7, no. 12, p. e49946, 2012.
[45] M. H. Fulekar, *Bioinformatics: Applications in Life and Environmental Science*. New York, NY, USA: Springer, 2009.
[46] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Modeling Comput. Simulation*, vol. 8, no. 1, pp. 3–30, 1998.
[47] J. Harrow, A. Frankish, J. M. Gonzalez, E. Tapanari, M. Diekhans, F. Kokocinski, B. L. Aken, D. Barrell, A. Zadissa, S. Searle, I. Barnes, A. Bignell, V. Boychenko, T. Hunt, M. Kay, G. Mukherjee, J. Rajan, G. Despacio-Reyes, G. Saunders, C. Steward, R. Harte, M. Lin, C. Howald, A. Tanzer, T. Derrien, J. Chrast, N. Walters, S. Balasubramanian, B. Pei, M. Tress, J. M. Rodriguez, I. Ezkurdia, J. van Baren, M. Brent, D. Haussler, M. Kellis, A. Valencia, A. Reymond, M. Gerstein, R. Guigo, and T. J. Hubbard, "GENCODE: The reference human genome annotation for The ENCODE project," *Genome Res.*, vol. 22, no. 9, pp. 1760–1774, Sep. 2012.
[48] L. Sun, A. M. Hui, Q. Su, A. Vortmeyer, Y. Kotliarov, S. Pastorino, A. Passaniti, J. Menon, J. Walling, R. Bailey, M. Rosenblum, T. Mikkelsen, and H. A. Fine, "Neuronal and glioma-derived stem cell factor induces angiogenesis within the brain," *Cancer Cell*, vol. 9, no. 4, pp. 287–300, Apr. 2006.
[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.



Kazuki Ichikawa received the BS degree from the Department of Bioinformatics and Systems Biology, Faculty of Science, The University of Tokyo, in 2012. He is a graduate student at the Department of Computational Biology, Graduate School of Frontier Sciences, The University of Tokyo.

Shinichi Morishita received the BS, MS, and PhD degree from the Department of Information Science, Graduate School of Science, The University of Tokyo. He is a professor in the Department of Computational Biology, Graduate School of Frontier Sciences, The University of Tokyo. He published more than 100 papers on logic programming, database systems, data mining, optimization algorithms, genome assembly, epigenomics, genome evolution, chromatin structure, RNAi, transcriptome, phenome, and personal genomics. He is a member of the ACM.