Knowl Inf Syst (2007) 12(1):25–53
DOI 10.1007/s10115-006-0027-5

Knowledge and Information Systems

REGULAR PAPER

Hisashi Koga · Tetsuo Ishibashi · Toshinori Watanabe

Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing

Received: 24 March 2005 / Revised: 25 February 2006 / Accepted: 11 March 2006 /
Published online: 21 July 2006

Abstract The single linkage method is a fundamental agglomerative hierarchical clustering algorithm. This algorithm regards each point as a single cluster initially. In the agglomeration step, it connects a pair of clusters such that the distance between the nearest members is the shortest. This step is repeated until only one cluster remains. The single linkage method can efficiently detect clusters in arbitrary shapes. However, a drawback of this method is a large time complexity of $O(n^2)$, where n represents the number of data points. This time complexity makes this method infeasible for large data. This paper proposes a fast approximation algorithm for the single linkage method. Our algorithm reduces the time complexity to $O(nB)$ by rapidly finding the near clusters to be connected by Locality-Sensitive Hashing, a fast algorithm for the approximate nearest neighbor search. Here, B represents the maximum number of points going into a single hash entry and it practically diminishes to a small constant as compared to n for sufficiently large hash tables. Experimentally, we show that (1) the proposed algorithm obtains clustering results similar to those obtained by the single linkage method and (2) it runs faster for large data than the single linkage method.

Keywords Hierarchical clustering · Single linkage method · Locality-Sensitive Hashing · Outlier removal

1 Introduction

Clustering is a powerful tool for data analysis; it provides an insight into the characteristics of the given data by classifying them into several groups. Because of its usefulness, clustering has been used in various application areas such as biology,

H. Koga (✉) · T. Ishibashi · T. Watanabe
Graduate School of Information Systems, University of Electro-Communications,
1-5-1 Chofugaoka, Chofu-si, Tokyo 182-8585, Japan
E-mail: koga@is.uec.ac.jp

medicine, and information science. Conventional clustering methods are categorized into two classes—hierarchical and nonhierarchical clustering. Hierarchical clustering algorithms are further classified into three types: agglomerative, divisive, and incremental hierarchical clustering. The agglomerative hierarchical clustering begins with one-point clusters and recursively merges the most similar pair of clusters. On the other hand, the divisive hierarchical clustering begins with one cluster of all data points and recursively divides the most appropriate cluster. The incremental hierarchical clustering builds the hierarchy in an on-line fashion, while scanning the data stream to reduce the frequency of the data scan.

The single linkage method [13] is one of well-known agglomerative hierarchical clustering algorithms. In this method, the distance between two clusters is defined as the distance between the two nearest points each of which is a member of each cluster. In the agglomeration step, the clustering algorithm first searches for the closest pair of clusters and merges them into a new single cluster. Then, the distances between this new cluster and the other unchanged clusters are updated. The agglomeration step is repeated until the number of clusters finally reduces to one. This algorithm generates clusters such that each member of a cluster is more closely related to at least one member of the same cluster than to any point outside it. This algorithm tends to gather a chain of points into a single cluster, and it can detect clusters in arbitrary shapes. The clustering result is typically drawn as a dendrogram that helps the analyzers to acquire proper understanding of the data. However, a significant drawback of the single linkage method is its large time complexity of $O(n^2)$, where n represents the number of points in the data. The bound of $O(n^2)$ originates from computing the distances between all the pairs in these n points. This time complexity becomes more serious as the data size grows large.

Hence, this paper proposes its fast randomized approximation algorithm. Our approximation algorithm utilizes *Locality-Sensitive Hashing* [8], abbreviated as LSH hereafter, which is a probabilistic approximation algorithm for the nearest neighbor search. We rely on LSH to efficiently search for the close clusters to be merged. Therefore, our algorithm is termed as LSH-link. The hierarchical structure is formed by combining the multiple clustering results derived with different parameter values. LSH-link achieves a time complexity of $O(nB)$, where B is the maximum number of points that enter a single hash entry. Although B becomes $O(n)$ such that n points are stored in the hash tables, $B \ll n$ holds practically if large hash tables are prepared. Thus, B may be treated as a small constant as compared to n.

LSH-link extracts fewer layers than the single linkage method; hence, it enables the analyzers to obtain simple dendrograms that can be easily understood. Note that as the amount of data increases, the dendrograms depicted by the single linkage method tend to become increasingly dense and complex. Thus, our algorithm is useful especially when analyzers need to obtain a coarse-grained hierarchical structure from the given data in a short period. It is noteworthy that LSH can also be utilized as a fast approximation algorithm to compute the minimum spanning tree for complete graphs which the single linkage method can compute exactly.

We show by experiment that (1) LSH-link obtains clustering results that are similar to those obtained by the single linkage method and (2) it runs faster for large data than the single linkage method.

This paper is organized as follows. In Sect. 2, we introduce LSH in detail. Section 3 explains our LSH-link algorithm, and its time complexity is discussed. In Sect. 4, LSH-link is compared to previous related research. In Sect. 5, the experimental result of LSH-link is reported. Section 6 gives a guideline on parameter adjustment in LSH-link. Section 7 concludes this paper and outlines the future work.

2 Locality-Sensitive Hashing (LSH)

LSH [8] is a randomized algorithm for searching approximate nearest neighbor points for a given query point q from a set of points P in the Euclidean space. LSH uses a hash function satisfying the property that near points are stored in the same hash entry (bucket) with a high probability, while remote points are stored in the same bucket with a low probability. Thus, it narrows down the search range to the points stored in the same bucket as q, thereby accelerating the nearest neighbor search. In addition, LSH prepares multiple hash functions to reduce the probability that a near point to q is unnoticed. It has been proved both in theory and by experiment that LSH runs efficiently even for high-dimensional data [5]. We will now explain its mechanism.

2.1 Hash functions

Let $p = (x_1, x_2, \ldots, x_d)$ be a point in a d-dimensional space, where the maximal coordinate value of any point is less than a constant C. We can transform p to a Cd-dimensional vector $v(p) = \mathrm{Unary}_C(x_1)\mathrm{Unary}_C(x_2), \ldots, \mathrm{Unary}_C(x_d)$ by concatenating unary expressions for every coordinate. Here, $\mathrm{Unary}_C(x)$ is a sequence of x ones followed by $C-x$ zeroes, as shown in Formula (1). For example, when $p = (3, 2)$ and $C = 5$, $v(p) = 1110011000$.

$$\mathrm{Unary}_C(x) = \underbrace{11\ldots11}_{x}\underbrace{00\ldots00}_{C-x}. \tag{1}$$

In LSH, a hash function is defined as follows. Let I be a set consisting of k distinct elements chosen uniformly randomly from a set of Cd arithmetic values $\{1, 2, \ldots, Cd\}$, where k is a parameter of LSH. A hash function for LSH computes a hash value for a point p by concatenating k bits from $v(p)$ corresponding to the set I. For instance, let $I = \{1, 3, 5, 6, 9\}$ and $v(p) = 1110011000$. Then the hash value $h_I(p)$ is 11010.

LSH prepares l distinct hash functions that differ from one another in the selection of I. k and l are important LSH parameters; their effects are discussed in Sect. 2.3. Throughout this paper, k denotes the number of sampled bits and l denotes the number of hash functions.

2.2 Approximate nearest neighbor search

The procedure to find the nearest point to q from the set of points P involves two steps.

Step 1: Generation of hash tables: l hash functions $h_1, h_2, \ldots, h_l$ are constructed from l sets of sampled bits $I_1, I_2, \ldots, I_l$. Next, l hash values are computed for each point in P. The l hash tables are built in this manner.

Step 2: Nearest neighbor search for q:

1. l hash values are computed for q.
2. Let P_i be the set of points in P stored in the same bucket as q in the hash table for h_i. In LSH, the points in $P_1 \cup P_2 \cup \cdots \cup P_l$ become the candidates for the nearest point to q. Among these candidates, the nearest point to q is determined by measuring the real distance to q.

2.3 Parameters in LSH

This section describes the intuitive significance of the LSH parameters.

2.3.1 Number of sampled bits k

A hash function in LSH cuts the d-dimensional space into cells (buckets) by k hyperplanes. Figure 1 illustrates a 2-dimensional case when $C = 5$ and $k = 2$. The hash value of each cell is written inside it, provided the third bit (i.e., the line $x = 3$) and the seventh bit (i.e., the line $y = 2$) are selected from the total $2 \times 5 = 10$ bits. This figure shows that the near points tend to take the same hash value. The sizes of the generated cells are expectedly large for small k values, while they are expectedly small for large k values. Thus, k determines how far points are likely to take the same hash value. Our algorithm in Sect. 3 exploits this LSH characteristic.

2.3.2 Number of hash functions l

The reason for the utilization of multiple hash functions in LSH is that two points may take different hash values depending on the result of the probabilistic space division, even if they are sufficiently close to each other. For example, in Fig. 1,

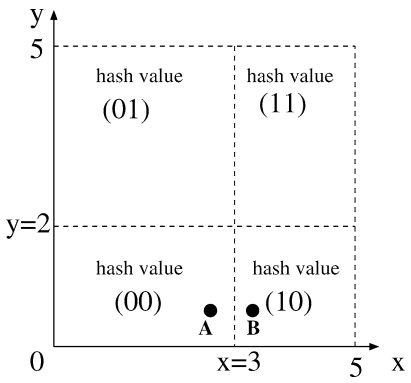


Fig. 1 Space partition by a hash function in LSH

although the two points A and B are close, the cell containing A does not cover B.

However, by increasing the number of hash functions, A and B will enter the same bucket for at least one hash function. This reduces the probability of the oversight of the near neighbor points. On the other hand, for large l values, far points might be contained in the candidates for the nearest neighbors (that is, false positive). LSH considers the existence of false positives; therefore, it inspects the points in the candidate set for the nearest neighbors by measuring the real distances.

3 LSH-link algorithm

This section explains our new clustering algorithm LSH-link that exploits the hash tables generated by LSH. This algorithm outputs clustering results that approximate those obtained by the single linkage method.

3.1 Design concepts

Our algorithm works in multiple phases. A single phase corresponds to a single layer in the hierarchy.

In a single phase, our algorithm finds the clusters within a distance of r for each cluster by LSH and combines them into a single cluster. Here, r is a parameter that should be adjusted to the number of sampled bits k in LSH. The analyzer has to specify r only for the first phase. In all other phases, r is automatically configured. We discuss how to adjust r to k in Sect. 3.3.

After a phase ends, LSH-link increases r and advances to the next phase if more than one cluster remain (If only one cluster exists, LSH-link terminates). By increasing r, we anticipate that clusters not merged in the previous phase are likely to be unified in this phase. The repetition of multiple phases creates the hierarchical structure. We say that a phase i is *higher* than a phase j, if i is performed after j in the execution of LSH such that phase i corresponds to a higher layer than phase j in the final hierarchy.

Note that the single linkage method merges the closest pair of clusters in a stepwise manner. Hence, we can consider that a single phase of LSH-link puts together several steps in the single linkage method. LSH-link differs greatly from the single linkage method in that it rapidly detects the clusters to be merged while omitting the calculation of the distances between all clusters; this contributes to a reduction in the running time.

3.2 Details of the algorithm

The operation of LSH-link is described below. In the beginning, each point is viewed as a cluster, thus yielding n clusters containing only one point each.

3.2.1 LSH-link algorithm

Step 1: For each point p, l hash values $h_1(p), h_2(p), \ldots, h_l(p)$ are computed. In the i-th hash table, p is stored in the bucket with index $h_i(p)$. However, if another point belonging to the same cluster as p has already been saved in the very bucket, p is not stored in it.

Step 2: For each point p, from the set of points that enter the same bucket as p in at least one hash table, LSH-link finds points whose distances from p are less than r.

Step 3: The pairs of clusters, each of which corresponds to a pair of points obtained in Step 2, are connected.

Step 4: If the number of clusters is larger than 1 after Step 3, r is set to a new larger value and LSH-link advances to Step 5. Otherwise, the algorithm terminates.

Step 5: LSH-link decreases k in accordance with the increase in r. It returns to Step 1 after generating new l hash functions by using this new k value.

In Step 1, it is ensured that any points belonging to the same cluster do not enter the same bucket. This feature is essential in order to avoid performance degradation of hashing, which tends to occur in higher phases where many points take the same hash value against the small k values. Note that Step 1 and Step 2 are equivalent to the original LSH except that only one point per cluster is allowed to remain in one bucket. Thus, LSH-link examines at most l buckets for each point. In Step 2, LSH-link removes the point pairs with distances greater than r from the set of candidates for the near point pairs. Figure 2 illustrates an example where six clusters from a to f are registered to three hash tables. In the figure, each column represents one hash table. Because clusters a and c enter the same bucket as cluster b, the distance between a and b and that between c and b are examined in order to determine if they are smaller than r.

Step 3 is responsible for merging the clusters. An example of this step is presented in Fig. 3. Here, clusters a and b, clusters b and c, and clusters c and d form three pairs to be merged (i.e., the three areas surrounded by thin lines). In this case, all the four clusters compose one large cluster surrounded by the thick line.

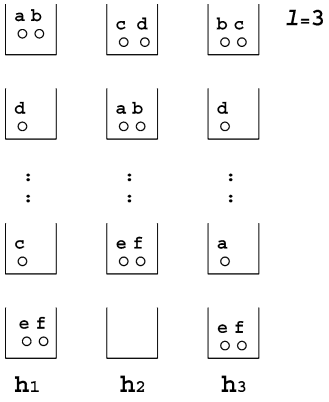


Fig. 2 The search for clusters to be connected

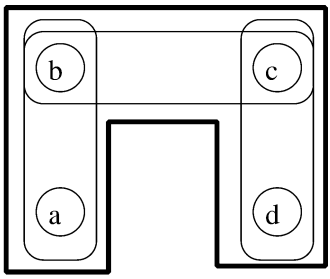


Fig. 3 Process of merging

Thus, similar to the single linkage method, LSH-link can detect clusters in various shapes.

3.2.2 Noise exclusion option

LSH-link allows the density-based noise exclusion to be integrated into the first phase as follows. In Step 2, for each point p, LSH-link examines the l buckets which p enters and counts the number of points whose distances from p are less than r, while allowing the repetitive counting of identical points. Let this number be rp. Because rp includes p in our implementation, $l \leq rp \leq nl$ holds. Although rp is a random variable, it stabilizes for a large l value due to the law of large numbers.

When rp is greater than the threshold parameter T, p is classified as a core point inside some cluster. Otherwise, p is classified as a boundary point of some cluster if at least one core point is less distant than r from p. Here, LSH can be used to search for the near neighbor of p rapidly. If p is neither a core point nor a boundary point, it belongs to low-density areas and ought to be excluded as noise. This technique is similar to DBSCAN [4] that examines how many points exist in the ϵ-neighborhood for each point.

3.3 Parameters in LSH-link

This section discusses the important parameters in LSH-link that influence the processing speed and quality of the clustering results.

- R, that is, the initial value of r: LSH-link connects the pairs of points whose distances are less than R in the first phase. This implies that in order to identify point pairs with small distances, R should also be set to a small value.
- The increase ratio A of r: Let r_c be the value of r in the current phase and r_n be its value in the next phase. When the current phase ends, LSH-link increases r such that $r_n = Ar_c \ \ (A > 1)$. A is a constant consistently used in all the phases. For large A values, the number of executed phases decreases and the execution time reduces. In contrast, for small A values, although the execution

time increases, the clustering result becomes detailed and more similar to that obtained by the single linkage method.

In this manner, R and A control the fineness of the clustering result. We present a guideline for adjusting R and A in Sect. 6.

– The relation between r and k: Although the sampled bits are selected randomly, the cell sizes are roughly determined when k is fixed. Therefore, r should be balanced with k. We select half the cell diagonal length as r. This is done under the assumption that the Euclidean space is partitioned at equal intervals by the sampled bits ideally. Let C be the maximal coordinate value in the given data. LSH treats the entire Euclidean space as a hypercube whose edge length is C. Since k sampled bits are uniformly distributed among the d dimensions in this ideal environment, the number of sampled bits per dimension is equal to $\frac{k}{d}$. Therefore, the section that starts at 0 and ends at C in each dimension is broken to $\frac{k}{d}+1$ subsections whose lengths are equally $\frac{C}{\frac{k}{d}+1}$. Thus, the cells constructed by these k sampled bits become hypercubes with edge lengths of $\frac{C}{\frac{k}{d}+1}$; the diagonal length of a hypercube is $\frac{dC}{k+d}\sqrt{d}$.
Therefore, if p and q are the two points that have entered the same bucket, the two associated clusters are merged if

$$||p-q|| \leq r = \frac{dC}{2(k+d)}\sqrt{d}. \tag{2}$$

Since Formula (2) is transformed into Formula (3), k decreases inversely with r.

$$k \approx \frac{dC}{2r}\sqrt{d}. \tag{3}$$

– Number of hash functions l: When l is small, a pair of points whose distance is smaller than r may be neglected by LSH-link. On the other hand, a large l value results in a wasteful overhead since many hash functions have to be computed. We can choose l appropriately based on the probability P_{miss} that a pair of points whose L_1 distance is r are separated into different buckets in all the l hash tables. When the L_1 distances are used in the implementation of LSH-link, P_{miss} accurately coincides with the probability that a pair of points whose distance is r is not found in Step 2 of LSH-link. When the L_2 distances are used instead as in the case of our implementation, P_{miss} does not coincide with the above-mentioned probability. However, P_{miss} still serves as a practical guide for selecting l appropriately.
In a hash table, a pair of points whose L_1 distance is r enter different buckets, if at least one sampled bit is placed between the pair of points. Since a sampled bit is uniformly randomly chosen from the set $\{1, 2, \ldots, Cd\}$, each sampled bit lies between the pair of points with probability $\frac{r}{Cd}$. Thus, at least one sampled bit from all the k sampled bits is placed between the pair of points with probability $1-(1-\frac{r}{Cd})^k$. Based on this argument, we have

$$P_{\text{miss}} = \left(1-\left(1-\frac{r}{Cd}\right)^k\right)^l. \tag{4}$$

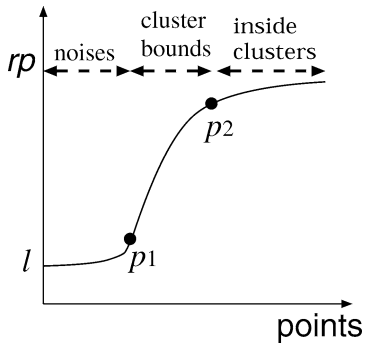


Fig. 4 rp-graph

The next equation is derived by substituting $\frac{dC}{2(k+d)}\sqrt{d}$ for r in Formula (4).

$$P_{\text{miss}} = \left(1 - \left(1 - \frac{\sqrt{d}}{2(k+d)}\right)^k\right)^l. \tag{5}$$

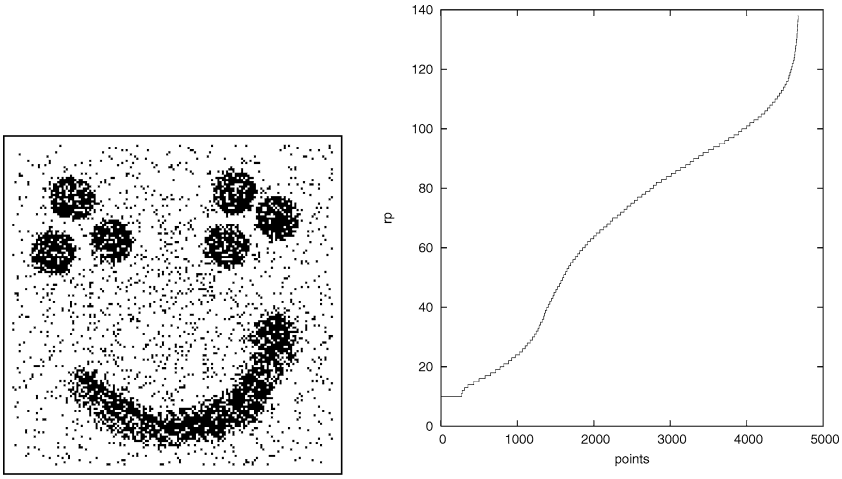
l should be determined such that P_{miss} may be sufficiently small after fixing the values of d and k.

– Threshold value T for rp: rp is the total number of points that enter the same bucket as a point p and that are less distant than R from p. rp measures the point-density of the R-neighborhood of p. Hence, T serves as the threshold point-density for noise exclusion. Below, we propose a heuristic approach for the analyzers to specify T in an interactive manner.
We sort all the points in the ascending order of rp[1] and plot them in this order on a 2-dimensional graph whose vertical axis shows the rp value. This graph increases monotonically, as shown in Fig. 4. This graph is named an *rp-graph*. If the rp-graph is observed along the horizontal axis, the following characteristics are visible:

1. Initially, the slope is moderate. This indicates the low-density areas containing noise points.
2. Next, the slope becomes steep. This corresponds to the boundaries of clusters lying between the noise areas and the interior of the clusters.
3. Finally, the slope becomes moderate again: This corresponds to the high-density areas inside the clusters.

LSH-link displays the rp-graph to the analyzers after computing the l hash tables in the first phase. Given the rp-graph, the analyzers will find two points p_1 and p_2 where the curvature becomes maximal. Finally, the analyzer can order LSH-link to set T to the rp value for p_2. We also need to consider the case in which only p_1 is found in the rp-graph. For example, when a cluster consisting of points generated by a nominal distribution, p_2 disappears from the rp-graph. In this case, T should be larger than the rp value for p_1. Figure 5b draws the real rp-graph for the data in Fig. 5a when $l = 10$. Based on this graph, we set T to 60 for this data. The clustering result is presented in Sect. 5.

[1] Bin sort finishes this sorting in $O(n)$ time because rp is a bounded integer.



(a) original data (b) rp-graph

Fig. 5 A real example of rp-graph. **a** Original data. **b** rp-graph

3.4 Time complexity

This section proves that the time complexity of LSH-link does not exceed $O(nB)$, where B is the maximum number of points which enter a single hash bucket over the execution of LSH-link. B is not a parameter of LSH-link. B is a function of the given data and LSH-link does not know its value before program execution. B depends on n because n points are distributed in the hash tables. For instance, suppose that the data points are uniformly distributed over the d-dimensional space and that the space is ideally divided uniformly by the sampled bits. Then, since the number of buckets is equal to $(\frac{k}{d}+1)^d$,

$$B = \frac{n}{\left(\frac{k}{d}+1\right)^d}. \tag{6}$$

Therefore, B becomes $O(n)$. However, in practice, B depends on the occupancy ratio of the hash tables and may be considered to be a constant by far smaller than n if we prepare sufficiently large hash tables. In fact, if k is augmented in Formula (6) for increasing the number of buckets, B decreases drastically. Although the hash table size decreases in the running of LSH-link as k gets smaller in higher phases, LSH-link maintains a small occupancy ratio by prohibiting multiple points from the same cluster entering the same bucket.

Our method repeats the following three operations until only one cluster remains. The number of repetitions is denoted by s. We clarify the time complexity incurred in all the three operations. Hereafter, let K be the k value for the first phase.

1. Generation of hash tables.
2. Search for clusters to be merged.
3. Updating clusters by connecting them.

Lemma 3.1 *Time complexity in generating hash tables is* $O(\frac{A}{A-1}nlK)$.

Proof The method of generating the hash tables is almost the same as that in LSH. However, when a point is saved in a bucket, an extra judgment must be made to check whether another point from the same cluster has already entered the bucket. This judgment finishes in $O(1)$ time by accompanying each bucket with an array indicating from which clusters the bucket has received points thus far. This computational overhead is lower than that for the calculation of hash functions mentioned below and may be ignored.

Now, let us evaluate the overhead for calculating the hash functions. With regard to the first phase, it takes $O(K)$ time to compute a hash function once because K sampled bits are looked into. Since we need to acquire l hash values for all the n points, $O(nlK)$ time is spent in the generation of hash tables in the first phase. From Formula (3), k is inversely proportional to r. Hence, k is reduced to $\frac{K}{A^{i-1}}$ in the i-th phase, in which r increases to $A^{i-1}R$. Therefore, the total time for calculating the hash functions over all the phases is at most

$$O\left(nlK + \frac{1}{A}nlK + \cdots + \frac{1}{A^{s-1}}nlK\right) \leq O\left(\sum_{i=0}^{\infty}\frac{1}{A^i}nlK\right) \leq O\left(\frac{A}{A-1}nlK\right).$$

Interestingly, since k decreases exponentially, the time complexity over all the phases is not more than constant times that in the first phase.

Because A, l, and K are constants independent of n, the generation of hash tables is completed in $O(n)$ time. □

Lemma 3.2 *Time complexity in searching for clusters to be merged is* $O(nlB)$.

Proof First, we consider the time complexity per phase. In a single phase, the set of point pairs (denoted by *SH*) entering the same bucket is scanned to find the point pairs with distances smaller than r. Since we have l hash tables and one bucket contains at most B points, the number of elements in *SH* associated with a single point is at most lB. Thus, the cardinality of *SH* never exceeds nlB. Therefore, the entire scan of *SH* costs at most $O(nlB)$.

Since there are s phases in total, the total time required for searching the clusters grows $O(snlB)$. We conclude this proof by showing that s is a constant of $O(1)$ size. In the i-th phase, pairs of clusters within a distance of $A^{i-1}R$ are connected. Let r' be the distance between the farthest point pair in the data and s' be the smallest integer satisfying $A^{s'-1}R \geq r'$. In other words, $s' \geq 1+\log_A(\frac{r'}{R})$. Because s' is the smallest integer satisfying this inequality, we have $s' \leq 2+\log_A(\frac{r'}{R})$. From the definition of r' and s', the number of phases s is smaller than s'. Hence, we obtain the next formula: $s \leq s' \leq 2+\log_A(\frac{r'}{R})$. Here, r' depends on the distribution of points; however, it does not depend on the data size. Because A and R are also independent of n, s is a constant of $O(1)$ size and it is independent of n.

As mentioned above, since the time complexity per phase is $O(nlB)$ and the number of phases s is $O(1)$, the operation over all the phases is completed in $O(nlB)$ time. Because l is a constant independent of n, the search for clusters is completed in $O(nB)$ time. □

Lemma 3.3 *Time complexity in updating clusters is* $O(nlB)$.

Proof Again, we consider the time complexity per phase. Consider the graph in which each vertex expresses a single cluster and an edge runs between two vertices such that these two vertices are a pair of clusters to be merged. Then, the construction of new large clusters is equivalent to the decomposition of the graph into connected components such that a connected component represents a new large cluster. The decomposition of a graph into connected components is realized by a depth-first search in $O(N+M)$ time by using the list expression of the graph, where N is the number of vertices and M is the number of edges. In our instance, clearly $N \leq n$. Note that each edge of the graph represents a pair of clusters to be merged that are selected from SH. Therefore, M is smaller than the cardinality of SH. Hence, $M \leq nlB$. In summary, the time complexity in constructing new large clusters becomes $O(n + nlB) = O(nlB)$. After connecting the present clusters, we need to make an array that indicates to which new cluster each point is assigned. This array is useful in the next phase, when the algorithm checks whether a bucket contains multiple points from the same cluster. $O(n)$ time is necessary for the arrangement of this array.

Since there are s phases, the total time for updating clusters becomes $O(snlB)$. In a manner similar to Lemma 3.2, it is proved that the update of clusters is completed in $O(nB)$ time. □

From Lemmas 3.1, 3.2, and 3.3, we derive Theorem 3.1 with regard to the total time complexity of LSH-link.

Theorem 3.1 *Time complexity of LSH-link is* $O(nB)$.

4 Comparison to related research

This section reviews the previous works related to our approach. The application of LSH to clustering was attempted by Haveliwala et al. [6] in the context of web-page clustering. Whereas our objective is to develop an approximation algorithm for the single linkage method, they address a practical issue of clustering the whole web repository which would need to be stored in hard disks with a size of a few hundred GBs. With regard to the clustering algorithm, our approach is similar to theirs in that near point pairs are derived by LSH. However, the method of merging clusters differs. Our algorithm is unique in that (1) the hierarchical structure is obtained by repetitions of LSH with different parameter values and (2) the hash tables in LSH are exploited smartly to exclude noise points.

Our approach randomly divides the Euclidean space into grids that we have termed as cells or buckets. With regard to grid-based clustering, an approach that considers only high-density cells to be valid and neglects low-density cells as outliers is used in algorithms such as CLIQUE [1] and DENCLUE [7]. This approach is particularly effective in the case of high-dimensional data, which often contain noise data points. Whereas these algorithms do not handle hierarchical structures between clusters, LSH-link with the noise exclusion option combines hierarchical clustering with density-based outlier removal.

DBSCAN [4] is a density-based clustering algorithm which checks if the ϵ-neighborhood of a point contains less than *Minpts* points to detect outliers. ϵ is called a generating distance. When the noise exclusion option is used, LSH-link is closely related to DBSCAN. LSH-link checks if the total number of points

included in the R-neighborhood of a point over the l hash tables exceeds T. If we ignore the fact that LSH-link is a randomized algorithm, it yields a clustering result after the first phase, similar to running DBSCAN with the next parameter setting: $\epsilon = R$ and $Minpts = \frac{T}{l}$. In the subsequent phases, LSH-link outputs the hierarchical structure between the clusters found in the first phase. Thus, LSH-link can build a hierarchical structure among the clusters detected by a density-based clustering algorithm. While DBSCAN utilizes a tree-based spatial index, LSH-link relies on locality-sensitive hashing which adapts well in high-dimensional spaces.

OPTICS [2] is a clustering algorithm extended from DBSCAN. The parameters in OPTICS are the same as DBSCAN: ϵ and *Minpts*. By examining the ϵ-neighborhood of each point in detail, OPTICS works in principle like multiple instances of DBSCAN with different generating distances ϵ' such that $\epsilon' \leq \epsilon$. OPTICS exhibits an intrinsic clustering structure in the dataset by a novel graphical representation method named the reachability plot. OPTICS has the advantage that the choice of parameters does not change the reachability plot significantly. The running of OPTICS with the two parameters ϵ and *Minpts* differs from that of LSH-link with $R = \epsilon$ and $T = l \times Minpts$ as follows. Let the set of clusters found in the running of DBSCAN with the parameters ϵ and *Minpts* be *DBS*. As mentioned previously, LSH-link detects clusters comparable to *DBS* and constructs a hierarchical structure that reflects the distances between the elements in *DBS*. However, LSH-link does not give information about the interior of the elements in *DBS*. OPTICS can clarify the cluster structure that exists inside the elements in *DBS* via the reachability plot. However, the reachability plot does not illustrate the distances between the elements in *DBS*, because the distances are greater than ϵ. Of course, the analyzer can easily comprehend that the value of ϵ is extremely small from the reachability plot in OPTICS.

With regard to hierarchical clustering algorithms that attain low time complexity, BIRCH [15] achieves an $O(n)$ I/O cost. BIRCH is an incremental hierarchical clustering algorithm and constructs a data structure called a CF (cluster feature) tree by scanning the data only once. Each node in the CF tree maintains a data structure named CF that represents a cluster by a simple 3-tuple consisting of the number of points, the linear sum of the points, and the square sum of the points. The size of the CF is sufficiently small to manipulate the clusters in the fast main memory. In contrast, BIRCH performs poorly if the clusters are not spherical in shape, as reported in Sheikholeslami et al. [12], since it does not memorize the coordinate values of each point.

STING [14] is another fast hierarchical clustering algorithm that divides the data space into grids. This algorithm constructs a hierarchical structure as follows. In the highest layer, the entire space is divided into 2^d cells, each of which is further divided into 2^d smaller cells in the next layer. This division is repeated recursively. In the lowest layer, the statistical information is created per grid by scanning the original data. The hierarchical structure is built by exploiting the ancestor–descendant relation between the grids. Like BIRCH, STING scans the entire data only once. However, in the worst case, the time complexity of STING attains $O(G)$, because all the grids in the lowest layer must be scanned. Here, G denotes the number of grids in the lowest layer. Since G increases exponentially with respect to d, STING is the most suitable for handling 2-dimensional data such

as geographical data. Note that the time complexity of LSH-link does not explode even for high-dimensional data, because LSH-link does not require all the grids to be scanned. Another shortcoming of STING is that the boundary of the derived cluster is inevitably parallel to the space axes, since the minimum granularity of the clustering is a grid in the lowest layer.

Because the primary purpose of BIRCH and STING is to reduce the frequency of slow disk accesses for processing VLDBs by scanning the whole data only once, they are not better than LSH-link at extracting clusters of arbitrary shapes. On the contrary, LSH-link must scan the whole data every time a new layer is constructed; this implies that the data must be scanned multiple times. Therefore, unless the entire data fits in the main memory without resorting to slow disks, the performance of LSH-link will deteriorate. Based on the above observation, we assert that LSH-link is useful for applications with the following two features:

– clusters of various shapes are required to be discovered;
– the entire data can be stored in the main memory.

CHAMELEON [11] with $O(n \log n)$ time complexity is an agglomerative clustering algorithm which innovates two evaluation measures—relative interconnectivity and relative closeness—to judge whether to connect two clusters. By considering both of these measures, CHAMELEON is able to discover natural homogeneous clusters not recognized by other clustering algorithms. However, the analyzers need to be responsible for how to combine these two measures in order to decide the proximity between clusters, for example, by arranging different threshold values or a weighting parameter. Furthermore, the relative closeness treats the distance between two high-density clusters as relatively longer than that between two low-density clusters, even if they actually have the same length in the original space. Therefore, the practical effect of the relative closeness is influenced by the testing environment. Like other ordinal clustering algorithms, LSH-link is not involved in these difficulties.

5 Experimental results

This section evaluates LSH-link experimentally. First, we confirm that LSH-link approximates successfully the single linkage method from the three viewpoints below with synthetic datasets mainly.

– Similarity of the dendrograms to the single linkage method.
– Ability to extract clusters in various shapes.
– Execution time for large datasets.

Then, the experimental results with two real datasets are presented.

5.1 Similarity of dendrograms

For evaluating a pattern recognition algorithm, the Iris dataset is used very frequently (ftp://ftp.ics.uci.edu/pub/machine-learning-databases/iris/). We compare the dendrogram by LSH-link with that by the single linkage method for Iris. Iris

consists of 150 points in a 5-dimensional space. The five dimensions express the length and width of the petal, the length and width of the cup, and the type of the Iris. In the experiment, we only use the data on four dimensions; the type of the Iris is not considered.

The parameters of LSH-link are set as follows: The number of hash functions $l = 10$ and the number of the sampled bits in the initial phase $K = 100$. Under this parameter configuration $P_{\text{miss}} \approx 0.008$. We decide the value of R based on K, in accordance with Formula (2). The purpose of this experiment is to investigate how the quality of the obtained dendrogram changes when A (the increase ratio of r) alters.

Figure 6 is the dendrogram by the single linkage method. Figures 7 and 8 are the dendrograms by LSH-link for $A = 1.4$ and for $A = 2.0$, respectively. Our web page (http://sd.is.uec.ac.jp/~koga/KAISdata.html) displays the large images of these dendrograms. In the page, the dendrogram for $A = 1.2$ is also displayed.

From these dendrograms, one can see that LSH-link yields a dendrogram that approximates the single linkage method for both $A = 1.4$ and $A = 2.0$. In

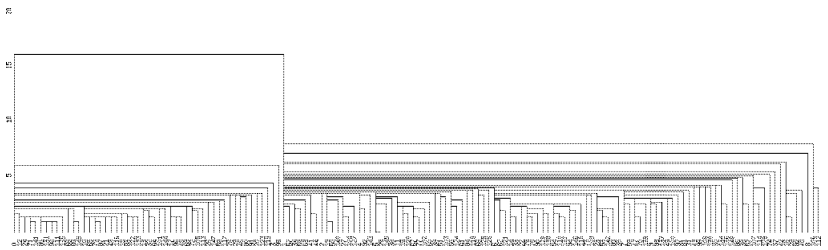


Fig. 6 Dendrogram by the single linkage method

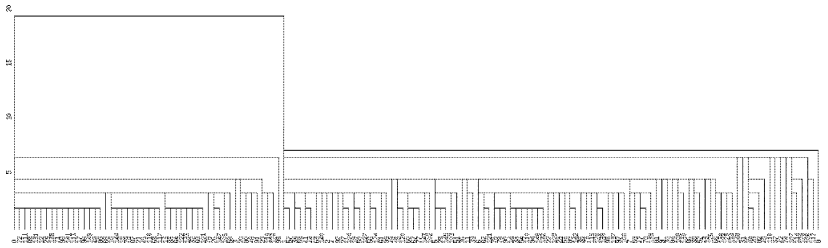


Fig. 7 Dendrogram by LSH-link ($A = 1.4$)

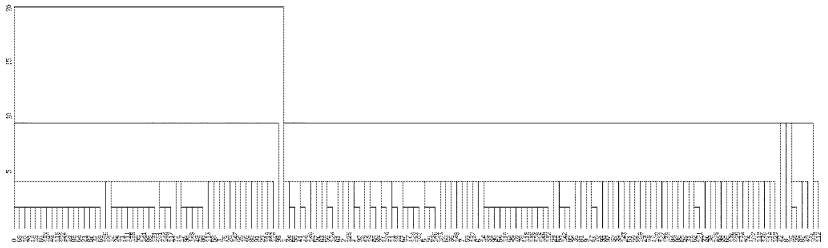


Fig. 8 Dendrogram by LSH-link ($A = 2.0$)

particular, the data are classified into the two large clusters at the top layer in all of the three dendrograms. We confirmed that the members of these two clusters match exactly between the single linkage method and our LSH-link. The clustering result for $A = 1.4$ resembles to the single linkage algorithm more than that for $A = 2.0$. This result claims that an analyzer can specify the granularity of the clustering result arbitrarily by adjusting A. LSH-link extracts fewer layers than the single linkage method; hence, the analyzers can obtain simpler and more comprehensive dendrograms. On the other hand, the dendrogram by the single linkage method is too dense.

In order to analyze qualitatively the similarity of the dendrogram drawn by LSH-link to that drawn by the single linkage method, the evaluation based on the next two measures is enforced.

– Cophenetic Correlation Coefficient (CCC): This coefficient calculates the correlation between two distance matrices. CCC is often used for measuring the similarity between two dendrograms. We denote the CCC between two distance matrices X and Y by $r_{X,Y}$ which is defined in Formula (7). Here, let $\bar{x}$ and $\bar{y}$ be the averages of the elements of the lower triangle matrices of X and Y, respectively. $r_{X,Y}$ takes a value between 0 and 1. The higher the correlation grows, the larger $r_{X,Y}$ gets. In particular, when X is identical to Y, $r_{X,Y} = 1$. From the viewpoint of statistics, when $r_{X,Y} \geq 0.7$, X is said to have a high correlation to Y. In this paper, the correlation between two distance matrices induced from the dendrograms generated by LSH-link and the single linkage method is investigated.

$$r_{X,Y} = \frac{\sum_{i<j}(X_{ij} - \bar{x})(Y_{ij} - \bar{y})}{\sqrt{\sum_{i<j}(X_{ij} - \bar{x})^2 \times \sum_{i<j}(Y_{ij} - \bar{y})^2}}. \tag{7}$$

– Ratio of the sum of edge lengths in the tree: The single linkage method produces the minimum spanning tree for the given data. Hence, we examine the number of times the sum of edge lengths in the tree generated by the LSH-link is larger than that generated by the single linkage method.

Table 1 shows these two measures for different A values. The arithmetic values in the table are averaged over 10 trials of experiments. From this table, it is evident that the correlation between the dendrogram by LSH-link and that by the single linkage method is quite high. The correlation becomes higher for smaller A values. A similar tendency is observed with regard to the ratio of the sum of edge lengths in the tree. The ratio of the sum of edge lengths in the tree is within 1.3 even if $A = 2.0$.

Next, we apply LSH-link to the 2-dimensional synthetic data in Fig. 9. 913 Points are plotted within an area of 100×100 size and there are six perceptible clusters in various shapes such as a bar and a ring. The minimum distance between two clusters that corresponds to the one between the ring cluster and the sphere cluster inside it is $\sqrt{53} \approx 7.28$, and the maximum distance between a pair of points belonging to the same cluster is $\sqrt{17} \approx 4.12$. Since these two values are quite close, a clustering algorithm has to be highly accurate in order to obtain a clustering result similar to that by the single linkage method.

Table 1 Evaluation result of LSH-link for Iris

	$A = 2.0$	$A = 1.4$	$A = 1.2$
CCC	0.9945	0.9981	0.9984
Ratio of the sum of edge lengths	1.259	1.180	1.164

Table 2 Evaluation result of LSH-link for the data in Fig. 9

	$A = 2.0$	$A = 1.4$	$A = 1.2$
CCC	0.9453	0.9601	0.9921
Ratio of the sum of edge lengths	1.390	1.326	1.308

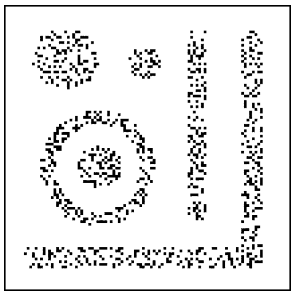


Fig. 9 Experimental data

We repeat the experiments under the condition that $l = 10$ and $K = 60$. In this setting, $P_{\text{miss}} \approx 0.001$. Table 2 shows the two evaluation measures—the CCC and the ratio of the sum of edge lengths in the tree—averaged over 10 trials for various A values. It is observed that the CCC takes a value greater than 0.94 for any A. The ratio of the sum of edge lengths in the tree never exceeds 1.4. We conclude that LSH-link gets more accurate, as A gets smaller. In fact, LSH-link succeeds in discovering the perceptible six clusters with a probability of $\frac{10}{10} = 100\%$ for any A, when the number of clusters remaining in the agglomeration step equals six. The clustering result is depicted in Fig. 10. This result also demonstrates the stability of our randomized algorithm.

5.2 Ability to extract clusters in various shapes

LSH-link is tested for five data in Fig. 11. All of them are 2-dimensional data plotted in a 150×150 area. They contain not only clusters in various shapes but also noise points. In addition, while the point-densities of the clusters are high in Fig. 11a–c, those in Fig. 11d and e are relatively low. The purpose of this experiment is to examine how LSH-link combined with the noise exclusion option detects clusters in various shapes from noisy data.

The parameters in LSH-link are set such that $l = 10$, $K = 80$, and $A = 1.4$. These parameters maintain P_{miss} at approximately 0.001. The threshold parameter T in the noise exclusion option is selected based on the heuristic algorithm

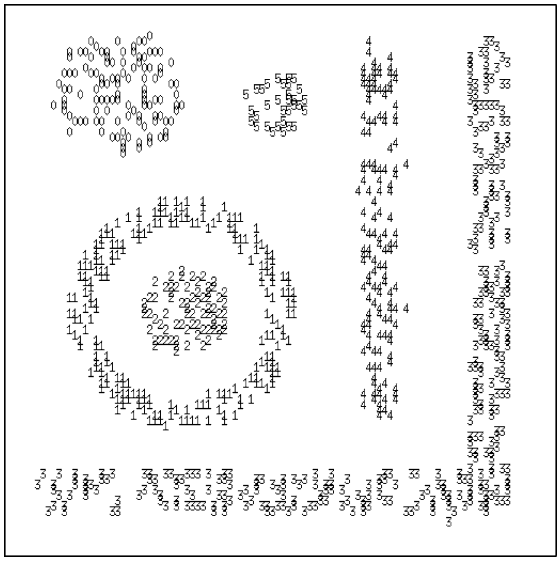


Fig. 10 Clustering result of LSH-link for the data in Fig. 9

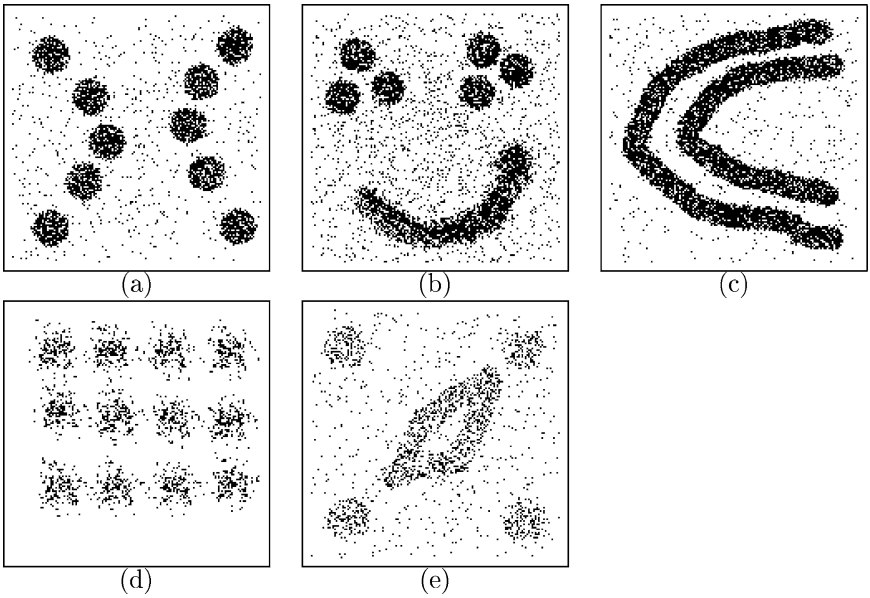


Fig. 11 Experimental data that contain noise points

described in Sect. 3.3. The behavior of the noise exclusion option is summarized in Table 3 where the second, third, and fourth rows describe the total number of points in the data, value of T, and the percentage of the noise points recognized by the noise exclusion option, respectively.

Table 3 Behavior of the noise exclusion option

Dataset	(a)	(b)	(c)	(d)	(e)
Number of points	3376	4673	6184	1989	1829
T	60	60	60	35	30
Noise ratio	19%	32%	8.4%	29%	33%

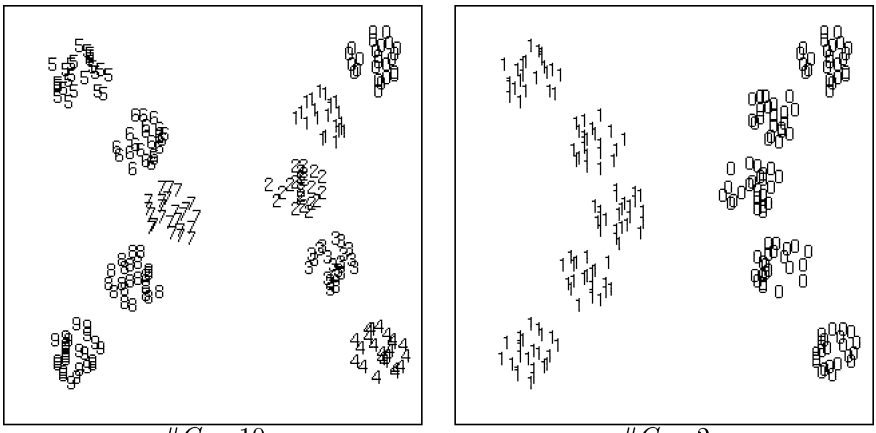


Fig. 12 Clustering result by LSH-link for Fig. 11a

Figures 12 and 13 depicts the effect of the noise exclusion option graphically. Here, the number of clusters that are requested to be detected from the hierarchical clustering result is denoted by $\#C$. Figure 12 is the clustering result for the data in Fig. 11a. When $\#C = 10$, 10 spherical clusters are detected. Furthermore, when $\#C = 2$, two clusters both of which consist of the five spherical clusters are found in the boomerang shape. In this way, LSH-link successfully grasps the interrelation between the underlying clusters. Figure 13 proves that the clusters in various shapes are separated well from the noise data points in the other datasets.

5.3 Execution time for large datasets

This section investigates how the execution time of LSH-link increases as the size of the dataset becomes larger. The experimental data consists of d-dimensional six clusters each of which is generated from a Gaussian distribution. The central points of these six clusters are listed below. Note that there exist two lumps of three clusters.

$$\underbrace{(10,\ldots,10)}_{d},\ \underbrace{(20,\ldots,20)}_{d},\ \underbrace{(30,\ldots,30)}_{d},\ \underbrace{(50,\ldots,50)}_{d},\ \underbrace{(60,\ldots,60)}_{d},\ \underbrace{(70,\ldots,70)}_{d}.$$

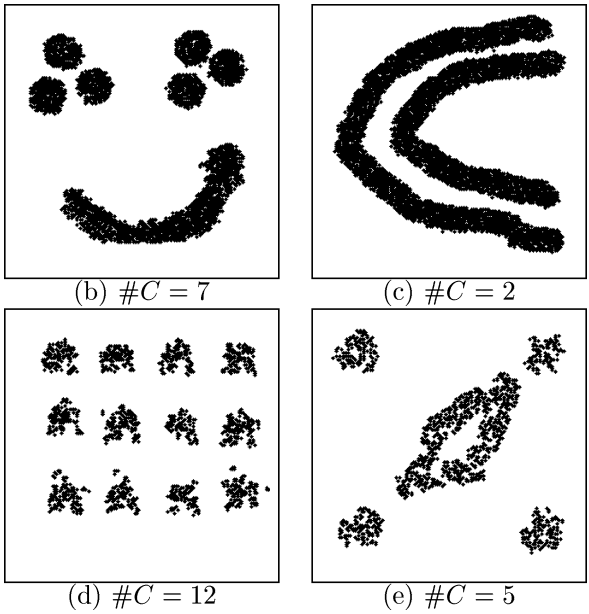


Fig. 13 Clustering result by LSH-link for Fig. 11b–e

5.3.1 Experiment 1

First, we set the variances of all the Gaussian distributions to 1, under the condition that $d = 20$. The execution time of LSH-link is compared with that of the single linkage method, when the number of data points changes from 600 (100 per cluster) to 7,500 (1,250 per cluster). The single linkage method is realized with the Partial Maximum Array [10]. With regard to the parameters in LSH-link, we set K to 220 and l to 30. Then we examine the two A values again, i.e., 1.4 and 2.0. In this experiment, P_{miss} takes a value of about 0.016.

Before discussing the length of the execution time, we briefly refer to the accuracy of LSH-link. LSH-link could extract six clusters correctly for both $A = 1.4$ and $A = 2.0$ without any error. In addition, when we set the number of the produced clusters to two, it was confirmed that the six clusters are divided into two groups of three clusters reliably. The dendrogram is displayed on our web page (http://sd.is.uec.ac.jp/∼koga/KAISdata.html).

Figure 14 shows the execution times averaged over ten trials of both LSH-link and the single linkage method. The horizontal axis represents the data size and the vertical axis represents the execution time in seconds. The experimental environment is built on a PC (CPU: Pentium IV 2.4 GHz, memory size 512 MB) with Redhat Linux OS (kernel 2.4.20-8).

From this graph, we conclude that, while the single linkage algorithm requires a time of $O(n^2)$, LSH-link runs almost linearly. Because the expectedly same hash table size determined by K is used throughout this experiment, B increases for large n values, although $B \ll n$. However, its effect does not appear in the graph, because the computation of hash functions is the dominant factor in the

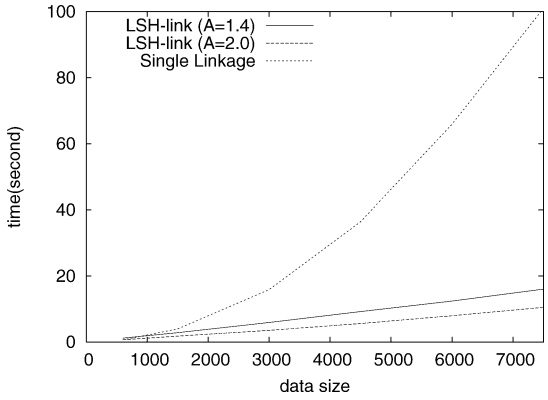


Fig. 14 Comparison of execution time

execution time of LSH-link. As the graph indicates, the speed of LSH-link is by far superior to that of the single linkage method for large data sizes. The averaged number of phases amounts to 9.2 for $A = 1.4$ and 5.2 for $A = 2.0$ independent of the data size. This difference appears as the gap in the execution time for the two parameter values. As shown in the proof of Lemma 3.2, the number of phases in LSH-link is independent of the data size.

5.3.2 Experiment 2

Next, we examine the dependency of the execution time on the dimension. A dataset that contains 6,000 points (i.e., 1,000 points per cluster) is prepared for each $d \in \{5, 10, 15, 20\}$. Here, the variance of each Gaussian distribution is 1. How d influences the execution time spent to produce the clustering result with the same fineness is investigated with these four datasets. In general, it is not easy to define the term "the same fineness." In this paper, we consider that LSH-link yields a clustering result for the d-dimensional dataset which has the same fineness as that for the d'-dimensional dataset, when the next two conditions are satisfied ($d \neq d'$).

- The number of sampled bits K in the first phase is proportional to the data dimension.
- P_{miss} is a constant independent of the data dimension.

The first condition implies that the average number of sampled bits per dimension is a constant. For a fixed K value, the number of hash functions l can be computed for a specific d so that P_{miss} may attain the desired value. Table 4 summarizes the values of K and l used in the experiment for each dimension. These parameters maintain P_{miss} at about 0.016. l is roughly proportional to d. Note that, for $d = 20$, the parameters are the same as Experiment 1.

The graph in Fig. 15 presents the execution time for $d \in \{5, 10, 15, 20\}$. The execution time increases quadratically with respect to d. The reason for this phenomenon is that both K and l are proportional to d. Since LSH-link is still by far faster for $d = 20$ than the single linkage method whose time complexity becomes

Table 4 Parameters in LSH-link for different d values

Dimension	$d = 5$	$d = 10$	$d = 15$	$d = 20$
K	55	110	165	220
l	10	16	23	30

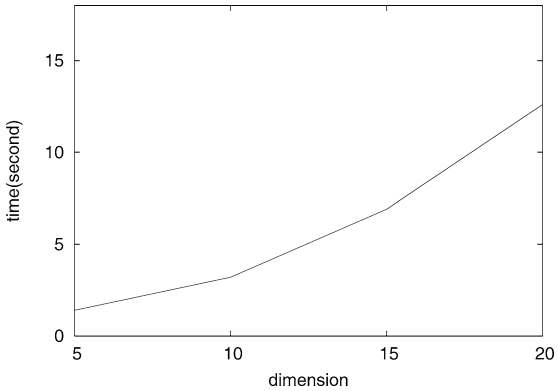


Fig. 15 Dependence on data dimension

a linear function of d, we believe that LSH-link will be useful for a dataset whose dimension does not go beyond about 40.

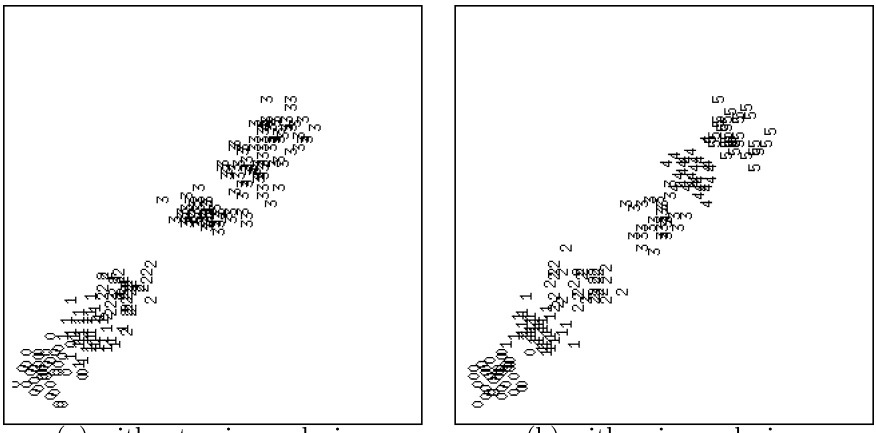
5.3.3 Experiment 3

Finally, we set the variance of each Gaussian distribution to 4 for $d = 20$. This large variance causes the six clusters to overlap with other clusters at their boundaries. Therefore, both the single linkage method and LSH-link without the noise exclusion option fail to detect the six clusters correctly as in Fig. 16a, where the number of data points equals 6,000. This image is obtained by projecting each 20-dimensional data point on a 2-dimensional space by using the first 2-dimensional coordinate values.

We investigate whether the noise exclusion option works well for this dataset. The experimental result for the case in which $K = 150$, $l = 30$, and $T = 40$ is presented in Fig. 16b. Since the six clusters are obtained correctly, it is confirmed that the noise exclusion option works effectively.

5.4 Experiments with real data

We evaluate the performance of LSH-link by using two real datasets. The first dataset is gathered from a gene database. The second dataset arises from an earthquake database. Again, the experiments are carried out on a PC (CPU: Pentium IV 2.4 GHz, memory size 512 MB) with Redhat Linux OS (kernel 2.4.20-8).



(a) without noise exclusion (b) with noise exclusion

Fig. 16 Effect of noise exclusion for six Gaussian clusters

5.4.1 Application to the gene database

The Gene Expression Omnibus [3] abbreviated as the GEO is a gene database managed by the National Center for Biotechnology Information. The GEO contains various gene expression datasets and provides a web interface which presents their precomputed clustering results in the form of dendrograms (http://www.ncbi.nlm.nih.gov/projects/geo/info/cluster.html).

From the GEO database, we select a dataset GDS10 named "Type 1 diabetes gene expression profiling." GDS10 consists of 39,114 gene expression data collected from diabetic mice. Each data is defined by the numerical values of 28 types of cDNAs. In other words, GDS10 contains 28-dimensional 39,114 data points. In the experiment we use the first 14,000 data in the dataset, so that the execution of LSH-link may fit in the main memory. Here, we do not try the noise exclusion option, because it is difficult to judge its effect without expertise on the testing data. Thus, we focus on how LSH-link obtains a clustering result similar to the single linkage method in a reasonable time. With respect to the configuration of LSH-link, we set $K = 450$ and $l = 45$. P_{miss} is about 0.02 for these parameters.

Table 5 presents the CCC for various A values, when the number of data is 6,000. It is observed that LSH-link approximates the single linkage method quite accurately.

Figure 17 compares the execution times of LSH-link for $A = 1.4$ and that of the single linkage method, when the size of the data increases up to 14,000. The graph shows that the execution time of LSH-link increases almost linearly

Table 5 Evaluation result of LSH-link for the GEO

	$A = 2.0$	$A = 1.4$	$A = 1.2$
CCC	0.943	0.961	0.970

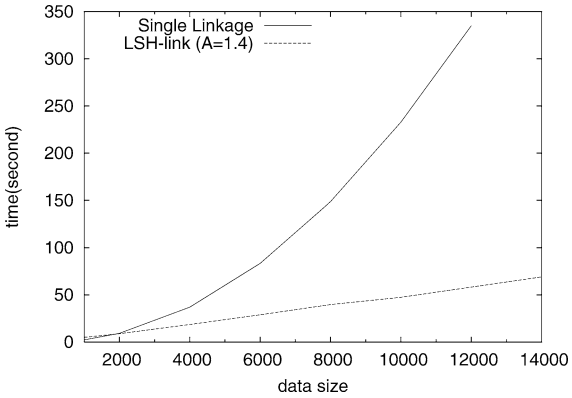


Fig. 17 Execution time for the GEO dataset

with respect to the data size, while that of the single linkage method increases quadratically.

When the size of the dataset is 14,000, we could not measure the execution time of the single linkage method due to performance degradation caused by the memory shortage. The implementation of the single linkage method holds the integer distance matrix of 14,000 × 14,000 size. This matrix alone consumes $\frac{14{,}000 \times 14{,}000}{2} \times 4$ byte = 392 MB memory, even if only the lower triangle matrix is stored. Since LSH-link does not rely on the distance matrix, its memory consumption is lower than the single linkage method. However, since our implementation keeps 45(= l) hash tables simultaneously in the main memory, the extent of memory saving is limited. Seeking an implementation in which only a single hash table is stored in the memory at a time is left as an open problem of this research.

5.4.2 Application to an earthquake database

The National Earthquake Information Center has made a worldwide earthquake database open to the public (http://neic.usgs.gov/neis/epic/epic_global.html). Each record of the database lists information on a single earthquake such as the time of occurrence, the location of the epicenter (latitude and longitude), and the magnitude. From this database, we gather records of earthquakes that occurred between the years 2000–2005 with magnitude greater than M4.8. Here, the number of corresponding earthquakes is 10,449.

We apply LSH-link to classify these 10,449 epicenters each of which is represented by a 2-dimensional vector consisting of the latitude and longitude. Especially, we expect the noise exclusion option to extract seismically active areas. In the experiment, the parameters of LSH-link are configured as follows: $K = 180$, $l = 15$, and $A = 1.2$. The threshold parameter T for the noise exclusion is set at 120.

Figure 18 presents the seven typical clusters discovered by LSH-link. In order to identify the seven clusters, tracking six or seven division points from the root of the dendrogram is necessary, which depends on the outcome of the randomness in LSH-link. By observing this picture, the analyzer would easily recognize the

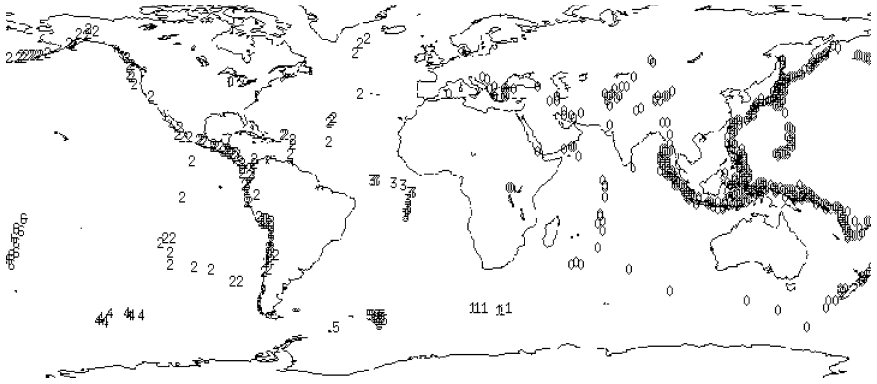


Fig. 18 Seven clusters obtained by LSH-link

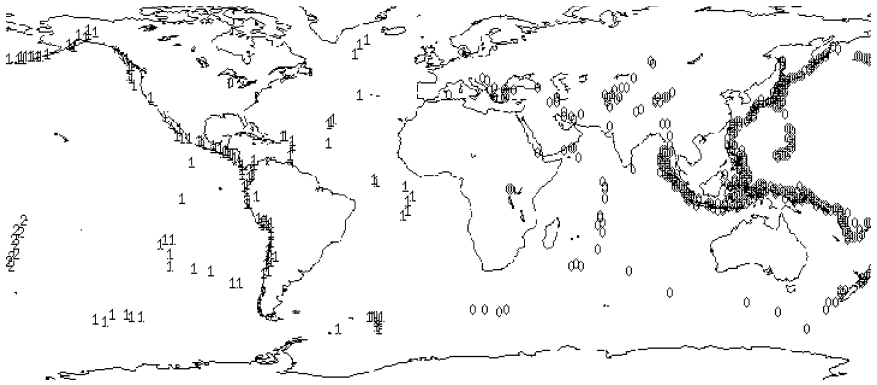


Fig. 19 Three clusters obtained by LSH-link

seismically active areas. Though epicenters have a tendency to lie continuously along an oceanic trench, the noise exclusion makes it possible to classify them as separated clusters by eliminating the low-density areas. Clusters No. 1, No. 3, and No. 5 in Fig. 18 are the instances of this phenomenon.

Figure 19 shows the clustering result when the number of detected clusters is three. This figure indicates that the four clusters surrounding the South America Continent in Fig. 18 (Clusters No. 2, No. 3, No. 4, and No. 5) are united. In addition, the cluster at the southward of the Africa Continent in Fig. 18 is connected to that on the Eurasia Continent in Fig. 18 as Cluster No. 0 in Fig. 19. This proves that LSH-link maintains the geographical distances (interrelations) between clusters in the hierarchical structure.

6 Guideline on the adjustment of R and A

LSH-link positively approximates the single linkage method to enhance the visibility of the dendrogram by extracting fewer layers. The number of extracted layers is controlled by two parameters R and A. This section provides some

guidelines to adjust R and A, while taking the computational overhead in LSH-link into account.

LSH-link constructs the hierarchical structure in the following manner.

- In the first phase, point pairs whose distances are less than R are merged.
- In the i-th phase for $i > 1$, cluster pairs whose distances are greater than $A^{i-2}R$ and smaller than $A^{i-1}R$ are merged.

To pursue high accuracy of approximation, R should be set to the distance between the nearest point pair in the data. Let this distance be $\min dis$. Although it takes $O(n^2)$ time to compute $\min dis$ accurately, an approximate value of $\min dis$ may be obtained in $O(n)$ time using the random sampling technique that involves at most $\sqrt{n}$ data points. We recommend this method for small n values, for instance, a few hundred.

In contrast, if n is large, the analyzers hardly pay attention to the lowest layer in the hierarchy which often becomes too dense to be visible in the dendrogram. In such a case, the choice of R rather had better consider the computational overhead of LSH-link. On one hand, if R is small, K increases as per Formula (3); this large K value increases the overhead for computing hash functions for which the time complexity is $O(nlK)$. On the other hand, if R is large, K decreases; this small K value increases the cell sizes. In this case, B becomes large, which in turn increases the overhead for merging clusters for which the time complexity is $O(nlB)$. Based on the above discussion, R should be set such that the overhead for computing hash functions is balanced with that for merging clusters. Therefore, B should ideally have the same order of magnitude as K. The analyzer can check the maximum number of points b entering a single hash entry after the first hash table is constructed. If b differs extremely from K, it is worth adjusting R to balance b with K.

As for A, if A is too small, the analyzers will perceive the dendrogram to be too dense. In this case, they can apply LSH-link again with a larger A value in a reasonable time due to the fast speed of LSH-link.

In contrast, if A is large, some hierarchical layers will not be detected. For instance, in Fig. 20, the layer in the dendrogram drawn by the single linkage method that connects clusters B and C shrinks in LSH-link. Consider a particular phase i in which j clusters are merged. Since j clusters are merged, the merge operations are performed $j-1$ times. LSH-link is able to compute easily the distance between the merged clusters for all the $j-1$ merge operations, since this distance is simply that between the associated point pair entering the same hash entry in phase i. We define the shrinkage coefficient of phase i as the standard deviation of

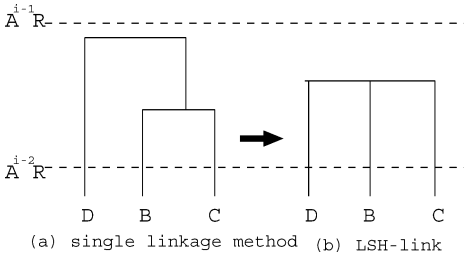


Fig. 20 A layer not detected in LSH-link

the $j-1$ distances normalized by $A^{i-1}R - A^{i-2}R$. When the shrinkage coefficient of phase i is large, the analyzer is worth applying LSH-link again with a smaller value of A.

7 Conclusion

This paper proposes a new fast approximation algorithm named LSH-link for the single linkage method. LSH-link attains a time complexity of $O(nB)$ by utilizing an approximate nearest neighbor search algorithm LSH, where B is the maximum number of points in a single hash entry. For practical use, B is a small constant as compared to n, if sufficiently large hash tables are prepared. By experiment, we show that (1) the proposed algorithm obtains clustering results similar to those obtained by the single linkage method and (2) it runs faster for large data than the single linkage method. In particular, this algorithm is suitable for the rapid discovery of the coarse-grained hierarchical structure from the given data. Furthermore, LSH-link can be easily combined with density-based noise exclusion.

One of the future scopes of this work is to devise a more sophisticated mechanism for selecting parameters automatically. Finally, it is important to seek application areas where LSH-link can be exploited to its full ability.

Acknowledgements The authors would like to thank anonymous reviewers for their very useful comments and suggestions.

References

1. Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high-dimensional data for data mining applications. In: Proceedings of ACM SIGMOD international conference on management of data, pp 94–105
2. Ankerst M, Breunig M, Kriegel H, Sander J (1999) OPTICS: Ordering points to identify the clustering structure. In: Proceedings of ACM SIGMOD international conference on management of data, pp 49–60
3. Barrett T, Suzek T, Troup D, Wilhite S, Ngau W, Ledoux P, Rudnev D, Lash A, Fujibuchi W, Edgar R (2005) NCBI GEO: mining millions of expression profiles – database and tools. Nucleic Acids Res 33:562–566
4. Ester M, Kriegel H, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd ACM SIGKDD, pp 226–231
5. Gionis A, Indyk P, Motwani R (1999) Similarity search in high dimensions via hashing. In: Proceedings of the 25th VLDB conference, pp 518–529
6. Haveliwala TH, Gionis A, Indyk P (2000) Scalable techniques for clustering the web. In: Proceedings of the 3rd international workshop on the web and databases, pp 129–134
7. Hinneburg A, Keim DA (1998) An efficient approach to clustering in large multimedia databases with noise. In: Proceedings of 4th ACM SIGKDD, pp 58–65
8. Indyk P, Motwani R (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of 30th ACM symposium on theory of computing, pp 604–613
9. Jain AK (1984) Handbook of pattern recognition and image processing. Academic Press, New York
10. Jung SY, Kim T (2001) An agglomerative hierarchical clustering using partial maximum array and incremental similarity computation method. In: Proceedings of the 2001 IEEE international conference on data mining, pp 265–272
11. Karypis G, Han E, Kumar V (1999) CHAMELEON: hierarchical clustering using dynamic modeling. IEEE Comput 32(8):68–75

12. Sheikholeslami G, Chatterjee S, Zhang A (1998) WaveCluster: a multi-resolution clustering approach for very large spatial databases. In: Proceedings of the 24th VLDB conference, pp 428–439
13. Sibson R (1973) SLINK: an optimally efficient algorithm for the single link cluster method. Comput J 16:30–34
14. Wang W, Yang J, Muntz R (1997) STING: a statistical information grid approach to spatial data mining. In: Proceedings of the 23rd VLDB conference, pp 186–195
15. Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering model for very large databases. In: Proceedings of the 1996 ACM SIGMOD international conference on management of data, pp 103–114

Hisashi Koga received the M.S. and Ph.D. degree in information science in 1995 and 2002, respectively, from the University of Tokyo. From 1995 to 2003, he worked as a researcher at Fujitsu Laboratories Ltd. Since 2003, he has been a faculty member at the University of Electro-Communications, Tokyo (Japan). Currently, he is an associate professor at the Graduate School of Information Systems, University of Electro-Communications. His research interest includes various kinds of algorithms such as clustering algorithms, on-line algorithms, and algorithms in network communications.

Tetsuo Ishibashi received the M.E. degree in information systems design from the Graduate School of Information Systems at the University of Electro-Communications in 2004. Presently, he is a system engineer at Fujitsu Broad Solution & Consulting Inc.

Toshinori Watanabe received the B.E. degree in aeronautical engineering in 1971 and the D.E. degree in 1985, both from the University of Tokyo. In 1971, he worked at Hitachi as a researcher in the field of information systems design. His experience includes demand forecasting, inventory and production management, VLSI design automation, knowledge-based nonlinear optimizer, and a case-based evolutionary learning system nicknamed TAMPOPO. He also engaged in FGCS (Fifth Generation Computer System) project of Japan and developed a new hierarchical message-passing parallel cooperative VLSI layout problem solver that ran on PIM (Parallel Inference Machine) in 1991. Since 1992, he has been a professor at the Graduate School of Information Systems, University of Electro-Communications, Tokyo, Japan. His areas of interest include media analysis, learning intelligence, and the semantics of information systems. He is a member of the IEEE.