Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2013, Article ID 891428, 8 pages
http://dx.doi.org/10.1155/2013/891428

Research Article

Spatial Cluster Analysis by the Bin-Packing Problem and DNA Computing Technique

Xiyu Liu and Jie Xue

School of Management Science and Engineering, Shandong Normal University, Jinan 250014, China

Correspondence should be addressed to Xiyu Liu; sdxyliu@163.com

Received 17 December 2012; Accepted 11 March 2013

Academic Editor: Bo Yang

Spatial cluster analysis is an important data mining task. Typical techniques include CLARANS, density- and gravity-based clustering, and other algorithms based on traditional von Neumann's computing architecture. The purpose of this paper is to propose a technique for spatial cluster analysis based on sticker systems of DNA computing. We will adopt the Bin-Packing Problem idea and then design algorithms of sticker programming. The proposed technique has a better time complexity. In the case when only the intracluster dissimilarity is taken into account, this time complexity is polynomial in the amount of data points, which reduces the NP-completeness nature of spatial cluster analysis. The new technique provides an alternative method for traditional cluster analysis.

1. Introduction

Spatial cluster analysis is a traditional problem in knowledge discovery from databases [1]. It has wide applications since increasingly large amounts of data obtained from satellite images, X-ray crystallography, or other automatic equipment are stored in spatial databases. The most classical spatial clustering technique is due to Ng and Han [2] who developed a variant PAM algorithm called CLARANS, while new techniques are proposed continuously in the literature aiming to reduce the time complexity or to fit for more complicated cluster shapes.

For example, Bouguila [3] proposed some model-based methods for unsupervised discrete feature selection. Wang et al. [4] developed techniques to detect clusters with irregular boundaries by a minimum spanning tree-based clustering algorithms. By using an efficient implementation of the cut and the cycle property of the minimum spanning trees, they obtain a performance better than $O(N^2)$, where N is the number of data points. In another paper, Wang and Huang [5] developed a new density-based clustering framework by a level set approach. By a valley seeking method, data points are grouped into corresponding clusters.

Adleman [6] and Lipton [7] pioneer a new era of DNA computing in 1994 with their experiments which demonstrated that the tools of laboratory molecular biology could be used to solve computation problems. Based on Adleman and Lipton's research, a number of applications of DNA computing in solving combinatorially complex problems such as factorization, graph theory, control, and nanostructures have emerged. There appeared also theoretical studies including DNA computers which are programmable, autonomous computing machines with hardware in biological molecules mode; see [8] for details.

According to Păun et al. [8], common DNA systems in DNA computing include the sticker system, the insertion-deletion system, the splicing system, and H systems. Among those, the sticker system has the ability to represent bits which is similar to the silicon computer memory. In a recent work, Alonso Sanches and Soma [9] propose an algorithm based on the sticker model of DNA computing [10] to solve the Bin-Packing Problem (BPP), which belongs to the class NP-Hard in the strong sense. The authors show that their proposed algorithms have time complexities bounded by $O(n^2)$ which are the first attempt to use DNA computing for the Bin-Packing Problem. Here the integer n is the number of items to be put in the bins.

Inspired by the work of Alonso Sanches and Soma [9], we propose a new DNA computing approach for spatial cluster analysis in this paper by the Bin-Packing Problem technique. The basic idea is to take clusters as bins and locate data points into bins. In order to complete evaluation of clustering, we need to accumulate dissimilarities within clusters. By the sticker system we can accomplish these tasks. We also show that our algorithm has a time complexity in polynomial in the case when only intracluster dissimilarity is considered, relative to the amount of data points. Notice that cluster analysis is NP-complete. It is interesting to notice that the method in this paper is new in cluster analysis.

The rest of this paper is organized as follows: in Section 2, we present the Bin-Packing Problem formulation of spatial clustering problem for the purpose of this paper. Then in Section 3 some basic facts on sticker model are presented with implementation of some new operations. The following two sections are devoted to the coding of the problem and the algorithms of clustering with sticker system. Finally, a brief conclusion is reached.

2. Formulation of the Problem

Let R^n be the real Euclidean space of dimension n. A subset $\Omega \subset R^n$ is called a spatial dataset with N points and $\Omega = \{x_1, x_2, \dots, x_N\}$, where $x_i = (\xi_{i1}, \xi_{i2}, \dots, \xi_{in}) \in R^n$ for each $i = 1, \dots, N$. A clustering problem over Ω is to group the dataset Ω into k partitions called clusters where the intracluster similarity is maximal and the intercluster similarity is minimal. In this sense, clustering is an optimization process in two levels: one is maximization and the other is minimization. Here the integer k indicates the number of clusters. There are two kinds of clustering when we consider k as a parameter. The first kind is fixed number clustering, where the number of clusters k is *a priori* determined. The second kind is flexible clustering where the number k is chosen as one of the parameters to meet the two level optimization problem.

Now we denote a partition of Ω by $\mathscr{C} : \mathscr{C} = (C_1, \dots, C_k)$ with $\Omega = C_1 \cup \cdots \cup C_k$ and $C_i \subseteq \Omega$ for $1 \le i \le k$. If we define $\operatorname{Asim}(C_i)$ as the intracluster dissimilarity measure for $C_i \in \mathscr{C}$ and $\operatorname{Simm}(C_i, C_j)$ as the intercluster similarity measure for $C_i, C_j \in \mathscr{C}$, then the two kinds of clustering problems are formulated as follows:

$$\begin{gathered} \min_{\mathscr{C}} \operatorname{Asim}(C_i) \quad \text{for } 1 \le i \le k, \\ \min_{\mathscr{C}} \operatorname{Simm}(C_i, C_j) \quad \text{for } 1 \le i, j \le k,\ i \ne j, \end{gathered} \tag{1}$$

$$\begin{gathered} \min_{\mathscr{C},k} \operatorname{Asim}(C_i) \quad \text{for } 1 \le i \le k, \\ \min_{\mathscr{C},k} \operatorname{Simm}(C_i, C_j) \quad \text{for } 1 \le i, j \le k,\ i \ne j. \end{gathered} \tag{2}$$

To simplify the multiplicity of optimization, we often use the following variation of the above problems:

$$\begin{gathered} \min_{\mathscr{C}} \sum_{i=1}^{k} \operatorname{Asim}(C_i), \\ \min_{\mathscr{C}} \sum_{1 \le i, j \le k, i \ne j} \operatorname{Simm}(C_i, C_j), \end{gathered} \tag{3}$$

$$\begin{gathered} \min_{\mathscr{C},k} \sum_{i=1}^{k} \operatorname{Asim}(C_i), \\ \min_{\mathscr{C},k} \sum_{1 \le i, j \le k, i \ne j} \operatorname{Simm}(C_i, C_j). \end{gathered} \tag{4}$$

Next we only consider the cluster problem (1) or (3). In order to unite the two optimization formula: we introduce the following total energy function:

$$\begin{gathered} \min_{\mathscr{C}} E_T(\mathscr{C}) : \\ E_T(\mathscr{C}) = \sum_{i=1}^{k} \operatorname{Asim}(C_i) + \sum_{1 \le i, j \le k, i \ne j} \operatorname{Simm}(C_i, C_j). \end{gathered} \tag{5}$$

For the purpose of this paper, we will use a simplified version of the total energy as shown in the following equation:

$$\min_{\mathscr{C}} E(\mathscr{C}) : E(\mathscr{C}) = \sum_{i=1}^{k} \operatorname{Asim}(C_i). \tag{6}$$

In the case when the number of clusters k is a variable, the total energy is computed for nonempty clusters and the optimized number k of clusters is the counting of nonempty bins:

$$\min_{\mathscr{C}} E(\mathscr{C}) : E(\mathscr{C}) = \sum_{1 \le i \le k, C_i \ne \emptyset} \operatorname{Asim}(C_i). \tag{7}$$

We now propose a *Bin-Packing Problem* (BPP) formulation of the clustering problem as stated above. The classical one-dimensional BPP is given as a set of N items $x_1, \dots, x_N$ with respective weights $w(x_i) = a_i \in (0, c]$, $1 \le i \le N$. The aim is to allocate all items into N bins with equal capacity c and by using a minimum number of bins [9]. For clustering purpose we assume that there are k empty bins and we allocate all items into the bins with least energy. If we consider k as a variable, then the problem is to allocate N points into N bins with least energy. The capacity restriction c is removed. For the two cases of clustering, there are altogether k^N (N^N, resp.) combinations of allocation and the best solution can be achieved by brute force search.

First we consider the case when k is fixed. To solve the problem, we consider an array $\mathscr{C}$ of integers

$$\mathscr{C} = c_1 c_2 \cdots c_N, \quad \text{each } c_i \in \{1, \dots, k\}. \tag{8}$$

The ith bin (cluster) C_i is defined as $C_i = \{x_p : c_p = i,\ p = 1, \dots, N\}$ for $i = 1, \dots, k$. We will identify the allocation $\mathscr{C}$ with its corresponding partition. Therefore the energy function is defined on $\{\mathscr{C}\}$ of all allocations. In order to guarantee the bins are nonempty, we need to add a restriction that $\#(C_i) > 0$ for $i = 1, \dots, k$, where $\#(C_i)$ denote the cardinality of the set C_i.

Then the final problem is

$$\begin{gathered} \min E(\mathscr{C}), \\ \mathscr{C} = c_1 c_2 \cdots c_N, \quad \text{each } c_i \in \{1, \dots, k\}, \\ \#(C_i) > 0 \quad \text{for } i = 1, \dots, k. \end{gathered} \tag{9}$$

```
weigh($T, b, d, T_l, T_g, T_e$)
  $i \leftarrow 1; T_l \leftarrow \emptyset; T_g \leftarrow \emptyset$
  repeat
    $T_0 \leftarrow -(T, b+i), T_1 \leftarrow +(T, b+i)$
    if $d_i = 0$ then
      $T_g \leftarrow T_g \cup T_1; T \leftarrow T_0$
    else
      $T_l \leftarrow T_l \cup T_0; T \leftarrow T_1$
    endif
    $i \leftarrow i + 1$
  until $(i = q + 1)$ or $(detect(T) = \text{no})$
    $T_e \leftarrow T$
clearq($T, b$)
  $i \leftarrow 1; T_0 \leftarrow \emptyset;$
  repeat
    $T_0 \leftarrow clear(T, b+i)$
    $T \leftarrow T_0$
    $i \leftarrow i + 1$
  until $(i = q + 1)$
```

Algorithm 1: Algorithms of **weigh** and **clearq**.

Next when k is a variable, the array $\mathscr{C}$ is

$$\mathscr{C} = c_1 c_2 \cdots c_N, \quad \text{each } c_i \in \{1, \ldots, N\}. \tag{10}$$

The ith bin (cluster) C_i is defined as $C_i = \{x_p : c_p = i,\ p = 1, \ldots, N\}$ for $i = 1, \ldots, N$. The energy function defined on $\{\mathscr{C}\}$ to be optimized is

$$\min E(\mathscr{C}),$$
$$\mathscr{C} = c_1 c_2 \cdots c_N, \quad \text{each } c_i \in \{1, \ldots, N\}. \tag{11}$$

3. A Sticker DNA Model

First we recall some standard operations of DNA computing as shown in [8]. They are *merge, amplify, detect*, *separate*, and *append*.

(i) *merge*: $T \leftarrow merge(T_1, T_2)$. Two given tubes T_1 and T_2 are combined into one T without changing the strands which.

(ii) *amplify*: Given a tube T, *amplify* (T, T_1, T_2) produces two copies T_1, T_2 of T and then make T empty.

(iii) *detect*: Given a tube T, return *true* if T contains at least one DNA strand, otherwise return *false*.

(iv) *separate*: $T \leftarrow +(T, w)$ and $T \leftarrow -(T, w)$. Given a tube T and a word w, a new tube $+(T, w)$ (or $-(T, w)$) is produced with the strands in T which contain w as s substring (resp., do not contain).

(v) *append*: Given a tube T and a word w, *append* (T, w) affixes w at the end of each sequence in T.

The sticker model is based on the paradigm of Watson-Crick complementarity and was first proposed in [10]. There are two kinds of single-stranded DNA molecules, the memory strands and sticker strands, in this model. A memory strand is L bases in length and contains p nonoverlapping substrands, each of which is m bases long, where $L = pm$ [8]. A sticker is m bases long and complementary to exactly one of the p substrands in the memory strand. A specific substrand of a memory strand is either on or off and is called a bit. If a sticker is annealed to its matching substrand on a memory strand, then the particular substrand is said to be on. Otherwise it is said to be off. These partially double strands are called memory complexes.

The basic operations of the sticker model are merge, separate, set, and clear and are listed as follows [8]. Among these, *merge* is exactly as the standard operation as shown before.

(i) *separate*: $T \leftarrow +(T_0, i)$ and $T \leftarrow -(T_0, i)$. Given a tube T_0, a new tube $+(T_0, i)$ (or $-(T_0, i)$) is produced with the ith bit on (resp., off).

(ii) *set*: $T \leftarrow set(T_0, i)$. A new tube T is produced from T_0 by turning the ith bit on.

(iii) *clear*: $T \leftarrow clear(T_0, i)$. A new tube T is produced from T_0 by turning the ith bit off.

Now we consider a test tube T consisting memory complexes α. We define the length of α as the number of bits, that is, the number of substrands (stickers) contained in α denoted by $p(\alpha)$. Each numerical value is represented by q-bit stickers, where q is a constant designed for a certain problem. For a q-bit stickers α, the corresponding numerical value is denoted by $h(\alpha)$. The substring in a memory complex from the $(b+1)$th bit to the $(b+q)$th bit of α is denoted by $cut(\alpha, b)$, where b is an integer with $0 \leq b \leq p(\alpha) - 1$. Apart from the basic operations, we need more operations designed and inspired by Alonso Sanches and Soma [9] in order to handle numerical computations.

(a) *generate*: Generate multiple copies of all the k^N combinations as $\mathscr{C}$. Append $1, 2, \ldots, k$ as the position numbers of $\mathscr{C}$. Then append $E_1, \ldots, E_k$ and E to store the energies.
(b) *energy*: Compute the dissimilarities of the k clusters and store the energy.
(c) *prune*: Discard unfeasible partitions, that is, those where there exists empty clusters.
(d) *find*: Find the best solution.

Now we present algorithms to implement the above procedures.

(a) Generation of all the possible k^N solutions. Append k values in order to store the energies.

```
generate($T$)
    $T_0 \leftarrow T$
    for $i \leftarrow 1$ to $N$ do
        for $j \leftarrow 1$ to $k/2$ do
            $amplify(T_0, T_{2j-1}, T_{2j})$
        endfor
        for $j \leftarrow 1$ to $k$ do
            $append(T_j, seq(j))$
        endfor
        for $j \leftarrow 1$ to $k$ do
            $append(T_j, seq(0))$
        endfor
        for $j \leftarrow k$ downto $1$ do
            $T_{j-1} \leftarrow T_{j-1} \cup T_j$
        endfor
    endfor
    $T \leftarrow T_0$
    $append(T, seq(0))$.
    for $j \leftarrow 1$ to $k$ do
        $append(T, seq(0))$
    endfor
    $append(T, seq(0))$
```

(b) Energy computation. The problem is to compute totals of energy for those i where $c_i = j$. Hence $E_j = \sum_{r,s \in C_j, r \neq s} d_{rs}$ and $C_j = \{i \mid c_i = j\}$. The total energy is stored in E. At the same time, the counting number of each bin is stored in the following k stickers.

```
energy($T$)
    for $j \leftarrow 1$ to $k$ do
        $T_j \leftarrow \emptyset$
    endfor
    for $i \leftarrow 1$ to $N$ do
        for $\leftarrow 1$ to $k$ do
            $compare(T, qi, q(N+j), T_1', T_2', T_3')$
            $clearq(T_2', qi)$
            $T_j \leftarrow T_j \cup T_2'$
            if $detect(T_2')$ = yes then
                $incr(T_j, q(N+2k+1+j))$
            endif
        endfor
    endfor
    for $j \leftarrow 1$ to $k$ do
        for $i_1 \leftarrow 1$ to $N$ do
            for $i_2 \leftarrow 1$ to $N$ do
                $add(T_j, q(N+k+j), \sim (h(cut(T_j, i_1 q) \vee h(cut(T_j, i_2 q)))d_{i_1 i_2})$
            endfor
        endfor
        $add(T_j, q(N+k+j), q(N+2k+1))$
        $T \leftarrow T \cup T_j$
    endfor
```

(c) The third step is to eliminate unfeasible partitions. This is done by checking the last k stickers.

```
prune ($T$)
    $T_l' \leftarrow \emptyset, T_e' \leftarrow \emptyset\ T_g' \leftarrow \emptyset$
    for $i \leftarrow 1$ to $k$ do
```

Algorithm 2: Continued.

```
        $compare(T, q(N + 2k + 1) + i, q(N + 3k + 2), T_l', T_g', T_e')$
        $T \leftarrow T_g'$
    endfor
    (d) The last step is to find the best solution with least energy. If $detect(T)$ = yes in the final step,
        then we get the optimal solution.
find($T$)
    $T_0 \leftarrow \emptyset, T_1 \leftarrow \emptyset$
    for $i \leftarrow 1$ to $q$ do
        $T_0 \leftarrow -(T, q(N + 2k + 1) + i)$, $T_1 \leftarrow +(T, q(N + 2k + 1) + i)$
        if $detect(T_0)$ = no then
            $T \leftarrow T_1$
        else
            $T \leftarrow T_0$
        endif
    endfor
    $detect(T)$.
```

Algorithm 2

(i) *increment*: $T \leftarrow incr(T, b)$. For each $\alpha \in T$, generate a strand β with $h(cut(\beta, b)) = h(cut(\alpha, b)) + 1$ and let T be replaced by the collection of such new strands β.

(ii) *add*: $T \leftarrow add(T, b, d)$. For each $\alpha \in T$, generate a strand β with $h(cut(\beta, b)) = h(cut(\alpha, b)) + d$ and let T be replaced by the collection of such new strands β. Here d is an integer in binary form with length $p(\alpha) - b$.

(iii) *compare*: $compare(T, b_1, b_2, T_l, T_g, T_e)$. For each $\alpha \in T$, if $h(cut(\alpha, b_1)) < h(cut(\alpha, b_2))$, then let $\alpha \in T_l$; if $h(cut(\alpha, b_1)) > h(cut(\alpha, b_2))$, then let $\alpha \in T_g$; else let $\alpha \in T_e$.

(iv) *weigh*: $weigh(T, b, d, T_l, T_g, T_e)$. For each $\alpha \in T$, if $h(cut(\alpha, b)) < d$, then let $\alpha \in T_l$; if $h(cut(\alpha, b)) > d$, then let $\alpha \in T_g$; else let $\alpha \in T_e$.

(v) *clearq*: $clearq(T, b)$. For each strand in the tube T, turn all bits off from $(b + 1)$th bit to $(b + q)$th bit.

We only give a DNA algorithm for *weigh* and *clearq* as the other algorithms are presented in Alonso Sanches and Soma [9]. Suppose the binary digits of the integer d is $d_{b+1}d_{b+2}\cdots d_{b+q}$ (see Algorithm 1).

4. Sticker Algorithms for Fixed k

Now we consider solving the spatial clustering problem as described in Section 2, where the number of clusters k is fixed, and $\Omega = \{x_1, x_2, \ldots, x_N\}$, $x_i = (\xi_{i1}, \xi_{i2}, \ldots, \xi_{in}) \in R^n$ for each $i = 1, 2, \ldots, N$. A partition of the dataset Ω is denoted by $\mathscr{C}$ which is an array of integers

$$\mathscr{C} = c_1 c_2 \cdots c_N, \quad \forall c_i \in \{1, \ldots, k\}. \tag{12}$$

For two points x, y we use $\rho(x, y)$ to denote the Euclidean distance between them. We use $M = \max_{x,y\in\Omega}\rho(x, y)$ to denote the diameter of Ω. Let the dissimilarity measure of $x, y \in \Omega$ be $\widehat{d}(x, y) = \rho(x, y)/M \in [0, 1]$. Now we convert the dissimilarity measure into binary string consisting of "0"s and "1"s. For an acceptable given error rate to measure the dissimilarity $\varepsilon > 0$, divide the interval $[0, 1]$ into 2^{K_1} subintervals with equal width $2^{-K_1} < \varepsilon$. Now choose an integer K such that $kN^2 2^{K_1} < 2^K$. Then we can use a K bits string to represent the subintervals. For $z \in [0, 1]$ let its corresponding string be $s(z) = [2^{-K_1} z]$, where operator $[\cdot]$ is the largest integer without exceeding it. We will use a sticker system with q stickers in length that is capable of representing numbers between $0, 1, \ldots, N$ and $0, 1, \ldots, 2^K$.

Now we define the dissimilarity matrix as

$$\begin{gathered} D = \left[d_{ij}\right]_{N\times N}, \\ d_{ij} = s\left(\widehat{d}\left(x_i, x_j\right)\right), \\ i, j = 1, \ldots, N. \end{gathered} \tag{13}$$

For the partition $\mathscr{C}$, the ith bin (cluster) C_i is defined as $C_i = \{x_p : c_p = i,\ p = 1, \ldots, N\}$ for $i = 1, \ldots, k$. A partition is called feasible if $C_i \neq \emptyset$ for $i = 1, 2, \ldots, k$. The energy of a partition defined by (9) has the following form:

$$\begin{gathered} E_t \triangleq \operatorname{Asim}\left(C_t\right) = \sum_{i,j\in C_t} d_{ij}, \\ E\left(\mathscr{C}\right) = \sum_{t=1}^{k} E_t. \end{gathered} \tag{14}$$

For an integer j, we use seq(j) to represent the subsequence of q stickers corresponding to j. Conversely, if s is a sequence, we use $h(x)$ to denote the numerical value decoded by the q-bit sticker x. By the sticker model [9], a memory complex seq$(\mathscr{C})$ is designed as the coding of $\mathscr{C}$:

$$\operatorname{seq}\left(\mathscr{C}\right) = \operatorname{seq}\left(c_1\right)\operatorname{seq}\left(c_2\right)\cdots\operatorname{seq}\left(c_N\right). \tag{15}$$

Then we append $k+1$ stickers representing $E_1, \ldots, E_k$ and $E = E(\mathscr{C})$. Finally we append k stickers to store the cardinality of clusters. The structure of stickers for our problem is shown

(a) *generate*: Generate multiple copies of all the N^N combinations as $\mathscr{C}$. Append $1, 2, \dots, N$ as the position numbers of $\mathscr{C}$. Then append $E_1, \dots, E_N$ and E to store the energies.
(b) *energy*: Compute the dissimilarities of the N possible clusters and store in energy.
(c) *find*: Find the best solution.
(d) *count*: Count the number of clusters.

Now we present algorithms to implement the above procedures.

(a) Generation of all the possible N^N solutions. Append N values in order to store the energies.

```
generate($T$)
    $T_0 \leftarrow T$
    for $i \leftarrow 1$ to $N$ do
        for $j \leftarrow 1$ to $N/2$
            $amplify(T_0, T_{2j-1}, T_{2j})$
        endfor
        for $j \leftarrow 1$ to $N$
            $append(T_j, seq(j))$
        endfor
        for $j \leftarrow 1$ to $N$
            $append(T_j, seq(0))$
        endfor
        for $j \leftarrow N$ downto 1
            $T_{j-1} \leftarrow T_{j-1} \cup T_j$
        endfor
    endfor
    $T \leftarrow T_0$
    $append(T, seq(0))$.
    for $j \leftarrow 1$ to $N$
        $append(T_j, seq(0))$
    endfor
    $append(T, seq(N))$.
```

(b) Energy computation. The problem is to compute totals of energy for those i where $c_i = j$. Hence $E_j = \sum_{r,s \in C_j, r \neq s} d_{rs}$ and $C_j = \{i \mid c_i = j\}$. The total energy is stored in E. At the same time, the counting number of each bin is stored in the following N stickers.

```
energy($T$)
    for $j \leftarrow 1$ to $N$ do
        $T_j \leftarrow \emptyset$
    endfor
    for $i \leftarrow 1$ to $N$ do
        for $j \leftarrow 1$ to $N$ do
            $compare(T, qi, q(N + j), T'_1, T'_2, T'_3)$
            $clearq(T'_2, qi)$
            $T_j \leftarrow T_j \cup T'_2$
            if $detect(T'_2)$ = yes then
                $incr(T_j, q(3N + 1 + j))$
            endif
        endfor
    endfor
    for $j \leftarrow 1$ to $N$ do
        for $i_1 \leftarrow 1$ to $N$ do
            for $i_2 \leftarrow 1$ to $N$ do
                $add(T_j, q(2N + j), \sim (h(cut(T_j, i_1 q) \vee h(cut(T_j, i_2 q)))d_{i_1 i_2})$
            endfor
        endfor
        $add(T_j, q(2N + j), q(3N + 1))$
        $T \leftarrow T \cup T_j$
    endfor
```

(c) The next step is to find the best solution with least energy. If $detect(T)$ = yes in the final step, then we get the optimal solution. The final number k of clusters in stored in the last sticker.

Algorithm 3: Continued.

```
find($T$)
  $T_0 \leftarrow \emptyset, T_1 \leftarrow \emptyset$
  for $i \leftarrow 1$ to $q$ do
    $T_0 \leftarrow -(T, q(3N+1)+i)$, $T_1 \leftarrow +(T, q(3N+1)+i)$
    if $detect(T_0)$ = no then
      $T \leftarrow T_1$
    else
      $T \leftarrow T_0$
    endif
  endfor
  $detect(T)$.
  (d) The final step is to count the number of clusters. It is stored in the last sticker while in the variable $k$.
count ($T$)
  $k \leftarrow 0$
  for $i \leftarrow 1$ to $N$ do
    $weigh(T, q(3N+1+i), 0, T_l, T_g, T_e)$
    if $detect(T_g)$ = yes then
      $incr(T, q(4N+2))$
      $k \leftarrow k+1$
    endif
  endfor
```

Algorithm 3

$\mathscr{C}$	Position	Cluster energies	Total energy	Counting of items	0-Flag
$c_1 c_2 \cdots c_N$	$1, 2, \ldots, k$	$E_1 \cdots E_k$	E	$I_1 \cdots I_k$	0

Figure 1: Coding structure of clusters.

$\mathscr{C}$	Position	Cluster energies	Total energy	Counting of items	Clusters
$c_1 c_2 \cdots c_N$	$1, 2, \ldots, N$	$E_1 \cdots E_N$	E	$I_1 \cdots I_N$	k

Figure 2: Coding structure when k is changing.

in Figure 1. The clustering algorithm consists of four steps as shown in Algorithm 2.

5. Sticker Algorithms for Variable k

In this section we consider cluster analysis when k is a variable. In this case a partition of the dataset Ω is denoted by $\mathscr{C}$ which is an array of integers

$$\mathscr{C} = c_1 c_2 \cdots c_N, \quad \forall c_i \in \{1, \ldots, N\}. \tag{16}$$

Similar to the previous section, the ith bin (cluster) C_i is defined as $C_i = \{x_p : c_p = i,\ p = 1, \ldots, N\}$ for $i = 1, \ldots, N$. Notice that C_i may be empty and the final number of clusters is the counting of nonempty clusters. The energy of a partition defined by (11) has the following form:

$$\begin{gathered} E(\mathscr{C}) = \sum_{t=1}^{k} E_t, \\ E_t = \begin{cases} \sum_{i,j \in C_t} d_{ij} & \text{for } C_t \neq \emptyset, \\ 0 & \text{for } C_t = \emptyset. \end{cases} \end{gathered} \tag{17}$$

Now the coding is seq($\mathscr{C}$). Then we append $N+1$ numbers of q bits, that is, $(N+1)q$ stickers representing $E_1, \ldots, E_N$ and $E = E(\mathscr{C})$. Next we append N values to store the counting number of the N clusters. Finally we append a value to store the number of valid (nonempty) clusters. The structure of stickers in this case is shown in Figure 2.

The clustering algorithm consists of four steps as shown in Algorithm 3.

6. Conclusion

In this paper we presented a new DNA-based technique for spatial cluster analysis. Two cases when the number of clusters k is predefined and not determined are considered. If we take the scale of data N, and the length of bits for a sticker q, as a variables, then clearly Algorithm 1 has a time complexity of $O(qN)$. Among the four steps of Algorithm 2, the operator generate (T) has a time complexity of $O(kqN)$, and the operator energy (T) has complexity of $O(kqN^2)$. The remaining two operators all have complexity of $O(kq)$. Thus the total time complexity for fixed number of clusters k is $O(kqN^2)$. In the other case when k is dynamic, time complexity for the four algorithms changes to $O(qN^2)$, $O(qN^3)$, $O(q)$, and $O(qN)$. Hence the total complexity is $O(qN^3)$. The reason why our complexity is worse than that of [9] (of course for a different problem) is that the summation of dissimilarity is time consuming. It is interesting if one can reduce this complexity to $O(qN^2)$.

Finally we will point out that up to the authors knowledge, this is the first research in cluster analysis by sticker DNA systems. It provides an alternative solution for this traditional knowledge engineering problem, which is *not* combinatorial in nature. Comparing many applications of DNA computing mainly in combinatorial problems, this is still interesting.

Acknowledgments

Research is supported by the Natural Science Foundation of China (no. 61170038, 60873058), the Natural Science Foundation of Shandong Province (no. ZR2011FM001), and the Shandong Soft Science Major Project (no. 2010RKMA2005).

References

[1] H. Jiawei and M. Kamber, *Data Mining Concepts and Techniques*, Elsevier, Singapore, 2nd edition, 2006.

[2] R. T. Ng and J. Han, "CLARANS: a method for clustering objects for spatial data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 5, pp. 1003–1016, 2002.

[3] N. Bouguila, "A model-based approach for discrete data clustering and feature weighting using MAP and stochastic complexity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 12, pp. 1649–1664, 2009.

[4] X. Wang, X. Wang, and D. M. Wilkes, "A divide-and-conquer approach for minimum spanning tree-based clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 7, pp. 945–958, 2009.

[5] X. F. Wang and D. S. Huang, "A novel density-based clustering framework by using level set method," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 11, pp. 1515–1531, 2009.

[6] L. M. Adleman, "Molecular computation of solutions to combinatorial problems," *Science*, vol. 266, no. 5187, pp. 1021–1024, 1994.

[7] R. J. Lipton, "DNA solution of hard computational problems," *Science*, vol. 268, no. 5210, pp. 542–545, 1995.

[8] G. Păun, G. Rozenberg, and A. Salomaa, *DNA Computing. New Computing Paradigms*, Texts in Theoretical Computer Science. An EATCS Series, Springer, Berlin, Germany, 1998.

[9] C. A. Alonso Sanches and N. Y. Soma, "A polynomial-time DNA computing solution for the bin-packing problem," *Applied Mathematics and Computation*, vol. 215, no. 6, pp. 2055–2062, 2009.

[10] S. Roweis, E. Winfree, R. Burgoyne et al., "A sticker based model for DNA computation," *Journal of Computational Biology*, vol. 5, no. 4, pp. 615–629, 1998.