PERGAMON

Pattern Recognition 33 (2000) 849–858

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

A tabu-search-based heuristic for clustering

C.S. Sung*, H.W. Jin

Department of Industrial Engineering, Korea Advanced Institute of Science and Technology, 373-1 Kusong-dong, Yusong-gu, Taejon 305-701, South Korea

Received 24 April 1998; received in revised form 8 February 1999; accepted 8 March 1999

Abstract

This paper considers a clustering problem where a given data set is partitioned into a certain number of natural and homogeneous subsets such that each subset is composed of elements similar to one another but different from those of any other subset. For the clustering problem, a heuristic algorithm is exploited by combining the tabu search heuristic with two complementary functional procedures, called packing and releasing procedures. The algorithm is numerically tested for its effectiveness in comparison with reference works including the tabu search algorithm, the *K*-means algorithm and the simulated annealing algorithm. © 2000 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Tabu Search; Heuristic; Clustering; Packing; Releasing

1. Introduction

Clustering has been an important issue occurring widely in information data analysis areas. Many of its applications can be found in the literature analyzing marketing, medical, archaeology, or pattern recognition data [1]. This paper considers such a clustering problem stated as follows. Given n elements each of which has K attributes, the problem objective is to group (classify) all the elements into C clusters such that the sum of the squared Euclidean distance between each element and the center of its belonging cluster for every such allocated element is minimized. The clustering process can be represented as an assignment of all the n elements to C clusters, so that a $\{0, 1\}$ integer programming formulation can be derived. For the integer programming formulation, the following notation will be used throughout this paper.

n	number of elements
C	number of clusters
K	number of attributes of each element
α_{ik}	value of the kth attribute of element i
m_{ck}	average of the kth attribute values of all elements in the cluster c

$$y_{ic} = \begin{cases} 1 & \text{if data } i \text{ is contained in cluster } c, \\ 0 & \text{otherwise.} \end{cases}$$

It is noted that the number of clusters C is a fixed value within the range of from 2 to $(n-1)$. Each α_{ik} is given as an input, and each y_{ic} is a decision variable. The average m_{ck} is dependent on y_{ic}. The given problem is now expressed as the integer programming problem;

$$\min \quad \sum_{c=1}^{C} \sum_{i=1}^{n} \sum_{k=1}^{K} (\alpha_{ik} - m_{ck})^2 \cdot y_{ic}$$

$$\text{s.t.} \quad \sum_{c=1}^{C} y_{ic} = 1, \; i = 1, 2, \ldots, n,$$

$$\sum_{i=1}^{n} y_{ic} \geqslant 1, \; c = 1, 2, \ldots, C,$$

$$y_{ic} \in \{0, 1\},$$

where

$$m_{ck} = \sum_{i=1}^{n} y_{ic} \cdot \alpha_{ik} \Big/ \left(\sum_{i=1}^{n} y_{ic} \right), \quad k = 1, \ldots, K, \; c = 1, \ldots, C.$$

* Corresponding author. Tel.: +82-42-869-3102; fax: +82-42-869-3110.

E-mail address: cssung@sorak.kaist.ac.kr (C.S. Sung)

PII: S0031-3203(99)00090-4

The objective function is non-linear and non-convex [2], so that it seems to be very difficult to investigate the problem in an analytical approach. Referring to Ref. [3], it is seen that the above problem is NP-complete. This provides us with the motivation of developing an efficient heuristic procedure to find a *near optimal* solution in *reasonable time*.

For such clustering problems, several algorithms using the simulated annealing technique have appeared in the literature [4–6]. Klein and Dubes [5] have proposed a simulated-annealing-based algorithm, but without making any explicit discussion about the cooling strategy. Selim and Al-Sultan [6] have proposed a simulated annealing algorithm with a discussion about both a cooling strategy and a parameter selection strategy. Al-Sultan [7] has recently proposed an algorithm with a tabu search heuristic adapted and showed that the algorithm outperforms both the *K*-means algorithm and the simulated annealing algorithm. Thereby, for the integer programming problem, this paper wants to exploit a heuristic solution algorithm by employing a tabu search heuristic.

2. Motivation of proposing the algorithm

The heuristic algorithm of this paper consists of two parts. One is to find the initial solution and the other one is to improve the initial solution. For the initial solution derivation, both the procedure of packing element pairs and the procedure of clustering such packed elements are proposed. Moreover, for the initial solution improvement, a reallocation procedure is also proposed.

The main feature of the heuristic algorithm is composed of *packing* and *releasing* procedures. The packing procedure is to bind a subset of elements together as a single element and the releasing procedure is to separate any packed elements from each other. In the initial solution step, any elements showing a high possibility to be grouped into a cluster is packed together, which is represented as the packing procedure. In the solution improvement step, each packed element is allowed to move together so as to make a drastic move (improvement) in the solution search effort. This is the major motivation of proposing the packing procedure. Another motivation is provided with that the solution space can be reduced by disregarding any solution in which any two elements having a short element-to-element Euclidean distance do not belong to the same cluster. Therewith, the packing procedure can help to promote the efficiency of the solution search.

On the other hand, there may be such a drawback in the packing procedure that the solution space may not be fully searched due to each packed element treated as a single element. This provides us with the motivation of proposing a releasing procedure such that any packed-elements are separated from each other (release of a packing relation). Once all the packed elements are tried for reallocation, the releasing operation is to be performed. The procedure of reallocating such packed elements and that of releasing any one of the packed elements are processed alternately in an iterative manner until all the packed elements are released. This way, the releasing procedure can help to promote the effectiveness of the solution search. More detail about the packing and releasing procedures will be discussed in Sections 3 and 4.

During the solution improvement effort, a reallocation trial of all the packed elements may become trapped at a local optimal solution. In order to avoid such a trapping situation, the tabu search method [8] is to be employed together with both the packing and releasing procedures to improve greatly the solution search efficiency and effectiveness. The tabu search method for clustering problems has already been proposed by Al-Sultan [7]. However, in this paper, the tabu search method is to be used as a sub-module in the whole solution algorithm. Moreover, as will be discussed later, the move for packed element and releasing is newly exploited for the tabu search, while Al-Sultan [7] has considered just the individual move.

3. Initial solution

3.1. Packing procedure

The basic idea of the proposing algorithm is that any pair of elements being close to each other are more likely to be grouped into a cluster. To implement this idea, the algorithm employs a packing procedure to treat a subset of elements as a single element. That is, a subset of elements are packed together as a single element and assigned to the same cluster.

The packing procedure is now described in detail. All the possible pairs of elements are first sorted in the increasing distance order. The two elements of the first ranked (closest) pair in the sorted sequence are packed together as a single element. This packing procedure is then repeated for the rest of all such ranked pairs until having a predetermined number of packings completed. If one of the elements of a current pair is a member of any earlier packed element, then the other element will also be packed into that earlier packed element. If both the elements of a current pair are separately contained as the members of two different packed elements, then the two packed elements will be agglomerated as one packed element. If both are contained as the members of the same packed element, then no change will be made. This way the packing procedure will be processed sequentially, while the method of determining the proper number of packings will be discussed later.

3.2. Packing property

This section characterizes a property named *packing property*. The property is used in the solution algorithm to determine the appropriate number of packings and to find an initial solution. Let us introduce the property called a generalized string property (GSP) which has been considered in Ref. [9] as the necessary condition for optimal clustering in one-dimensional Euclidean case.

Property (generalized string property).
Consider a clustering problem for various elements each being defined in a Euclidean space. Let d_{ij} be the Euclidean distance between elements i and j. If i and j are included together in a cluster, then every element k satisfying the relation $d_{kj} < d_{ij}$ or $d_{ki} < d_{ij}$ needs also to be included in the cluster.

Rao [10] has also studied a property similar to GSP, while it is neither a necessary condition nor a sufficient condition. Actually, the proposed clustering problem is heavily dependent on data structures, so that it seems difficult to find any good robust property. Therefore, this paper wants to exploit a packing property that can be used to determine the appropriate number of packings, while it is similar to GSP but more practical for determining the number of packings.

Property (packing property).
Let every cluster satisfy GSP and have more than m elements. Then each pair of elements having its element-to-element distance shorter than $D_{(2m-1)}$ belongs to such a cluster together, where $D_{(2m-1)}$ denotes the $(2m-1)$st shortest element-to-element Euclidean distance.

Proof. Suppose that d_{ij} is the kth shortest element-to-element distance, where $1 \leqslant k \leqslant 2(m-1)$ and $d_{ij} < D_{(2m-1)}$, and that elements i and j belong to different clusters, say I and J, respectively. Since each cluster has more than m elements, the size cardinality of each of the clusters I and J is at least m. Based on GSP, it holds that $d_{ii'} \leqslant d_{ij}$ for all $i' \in I, i' \neq i$, where the number of such element i' is at least $(m-1)$. Therefore, there are at least $(m-1)$ pairs having element-to-element distances shorter than or equal to d_{ij}. A similar logic can be applied to elements i and $j' \in J, j' \neq j$, having $(m-1)$ pairs with their distances shorter than or equal to d_{ij}. This implies that there are at least $2(m-1)$ pairs having their distances shorter than or equal to d_{ij}, and so d_{ij} is equal to or greater than $D_{(2m-1)}$. This contradicts the assumption that d_{ij} is the kth shortest distance.

Thus, the proof is completed. □

3.3. Number of packings

This algorithm uses $2(m-1)$ initial packings provided that each cluster has at least m elements. Therefore, determining the minimum cardinality m is equivalent to determining the number of packing operations. Surely, the lower bound of m is 1. The logic for calculating the upper bound can also be characterized easily. If a packing is made, then two elements (or packed elements) are agglomerated as one packed element whose cardinality is the sum of the cardinalities of all the original elements (or packed element). Since the number of packings is $2(m-1)$, in the extreme case the maximum cardinality of a packed element can be $2m-1$ (the number of packings plus its original element). In the case, $2m-1 \geqslant m$ for $\forall m \geqslant 1$ and all the elements in a packed element should belong to the same cluster, so that it is possible to make a cluster being composed of all the elements in such packed elements and having its cardinality at $2m-1$. This gives the maximum cardinality of a cluster in the initial solution. Moreover, since all the other clusters should contain more than m elements, the following relation should be satisfied; $n-(2m-1) \geqslant (C-1)m$. This leads to $m \leqslant (n+1)/(C+1)$, and so $m \leqslant \lfloor (n+1)/(C+1) \rfloor$ since m is an integer.

Thus, m has its upper bound, $\lfloor (n+1)/(C+1) \rfloor$, and so can take an integer number on the range between 1 and $\lfloor (n+1)/(C+1) \rfloor$. This implies that there are many possible choices for m. Therefore, this algorithm wants to choose the maximum cardinality of m. The reason is that the biggest m within the allowed integer range can maximize the advantage of the packing. Thus, the algorithm considers the number of packing operations at $2(\lfloor (n+1)/(C+1) \rfloor - 1)$.

3.4. Initial solution search procedure

In the initial solution each cluster should contain more than m elements, and each pair of elements having the element-to-element distance shorter than $D_{(2m-1)}$ should be packed together, at $m = \lfloor (n+1)/(C+1) \rfloor$. Moreover, it may be desirable to allow the variance in cluster cardinality to be small. These requirements are put together to derive the following initial solution search procedure.

Step 1: Sort each pair of elements in the increasing distance order, and pack every pair which has the element-to-element distance shorter than the $(2\lfloor (n+1)/(C+1) \rfloor - 1)$st shortest distance. Let each of such packed elements form an individual cluster, and a non-packed single element also form an individual cluster. Let C' denote the number of such individual clusters and k of them have more than $\lfloor (n+1)/(C+1) \rfloor$ elements. It can easily be shown that $C' > C$ for $n \leqslant 2$ and $2 \leqslant C < n$. Sort the C' clusters in the decreasing order of their cardinalities.

Step 2: If $k \geqslant C$, go to Step 3.

Otherwise, go to Step 4.

Step 3: Merge the Cth cluster with the $(C+1)$st cluster to become a single cluster.

$C' = C' - 1$.

If $k > C$, then $k = k - 1$.

Go to Step 5.

Step 4: Merge the $(k+1)$st cluster with the $(k+2)$th cluster to become a single cluster.

$C' = C' - 1$.

If the cardinality of the new cluster is greater than or equal to $\lfloor (n+1)/(C+1) \rfloor$, then $k = k + 1$.

Step 5: If $C' = C$, then stop.

Otherwise, sort the clusters in their cardinality order and go to Step 2.

4. Improvement of initial solution

Each element of the initial solution is to be reallocated among the C clusters for any better clustering. For the reallocation, the tabu search method is to be employed to avoid any search trapping at a local optimal solution and also to make an efficient and thorough reallocation search, such that all the elements of a current intermediate solution are reallocated so as to find any improved solution. Then one of the packed elements of the current solution is released from its packing relation. Thus, the reallocation and the releasing processes are sequentially operated. In an iterative manner, such that all the elements of the improved solution with such one packed element released are again reallocated among the improved C clusters (solution).

4.1. Reallocation of elements

Tabu search is a metaheuristic suggested by Glover [11,12] to solve combinatorial optimization problems by introducing a memory-based strategy to prevent the solution search from becoming trapped at a local optimal solution. The heuristic either generates or starts with some initial solutions and proceeds iteratively from one solution to another one (in neighborhood) until some termination conditions are satisfied. To improve the effectiveness of the heuristic, various intensification and diversification strategies have been developed [8,13,14]. One of them is the vocabulary building strategy [14].

In fact, the packing procedure functionally corresponds to the vocabulary building strategy of the tabu search. The vocabulary building strategy has been designed to identify any common attributes of a chosen set (solution) and also to search for (or generate) other solutions having the same common attributes. Similarly, the packing procedure is more likely to get any pairs having a shorter element-to-element distance to be included in the same cluster.

Now, the design issues of our tabu search are briefly described. A detail of the tabu search can be found in Ref. [8].

Move. In a clustering problem, a move indicates that an element changes its cluster to another one to which it is newly assigned. Specifically, each move represents a change of the form $\{y_{ij} = 1 \to y_{ij} = 0$ and $y_{ik} = 0 \to y_{ik} = 1$ for $j \neq k\ j, k = 1, \ldots C, i = 1, \ldots, n\}$. It has been shown in Ref. [7] that by ignoring all the solutions of a neighborhood at some predetermined ignorance probability, the performance of the associated algorithm increases as the ignorance probability gets lowered. This implies that such element ignorance is not appropriate. Therefore, this paper does not allow any neighborhood element ignorance.

There is a fundamental difference between the reference work [7] and ours. The reference work allows *individual move* such that any cluster change induced by a move is applied only to one element. However, in performing our packing operation, such a cluster change is induced by a move of the whole elements of a packed-element. That is, *packed-element-together move* occurs so as to make a drastic move in the solution search. Therefrom, several advantages can be gained. The first advantage is that a greater improvement in each intermediate solution, if any, is possible so as to get a faster approach to any anticipated good solution space. The second advantage is that the drastic move makes it possible to search a broader area of the solution space at each search iteration. Another important characteristic of the packing is that it can contribute to reduce the solution space. This is done by its helping to remove any non-interesting solution (regarded to be bad) from further consideration in the search effort. In fact, it is desired to disregard any solution in which any two elements having the short element-to-element distance do not belong to the same cluster. This way, the packing can help a lot to search for any better solution space more thoroughly (conforming to an intensification strategy).

Tabu list. In our algorithm, the tabu list contains each element index and its cluster index. For example, a tabu list element (i, j) indicates that at some iteration, any move for changing element i from its belonging cluster $k(\neq j)$ to another cluster j is prohibited because it is in a tabu state. The advantage of such tabu lists is in memory saving. In fact, for the integer data type, each element of a tabu list needs only 16 bits for computer programming, so that the tabu list is not a burden on computer memory.

In this paper, the size of the tabu list is determined dynamically to reduce the risk of trapping at cycling [15]. That is, the tabu list size is changed to a random integer number in the range between 7 and n after n iterations.

Secondary tabu list. In the clustering problem, there exist two types of cycle to worry about. One is the ordinary cycle appearing as the replication of the same order of mathematical solutions. The other one is the cycle associated with the labels attached to each cluster. The second type of cycle appears as the replication of

the same order of clusterings, but not as that of the mathematical solutions. This is because the label assigned to each cluster is used only to distinguish one cluster from the others. That is, in the mathematical formulation, any different label attached to the same cluster may be treated as a different solution. For example, consider the instance with $n = 4$ and $C = 2$. The solution $Y_1 = (y_{11} = y_{21} = y_{32} = y_{42} = 1,\ y_{12} = y_{22} = y_{31} = y_{41} = 0)$ and the solution $Y_2 = (y_{12} = y_{22} = y_{31} = y_{41} = 1,\ y_{11} = y_{21} = y_{32} = y_{42} = 0)$ mean the same clustering, where elements 1 and 2 constitute one cluster, and elements 3 and 4 constitute another. However, in terms of mathematical view they are different solutions. From this respect, the solution space of the mathematical formulation is over-expanded $C!$ times more than the real solution (clustering) space. In the above example, if the search process starts from Y_1 and reaches to Y_2, it will form a cycle in terms of clustering, so that the search process will move back again from Y_2 to Y_1. This trajectory will then be repeated. The tabu list described in the preceding section has the role of preventing the first type of cycle, but it cannot prevent the second type (clustering) of cycle. In fact, the second type of cycle was experienced very often during the experiment work of this paper. Therefore, it is desired to design some technique to prevent the associated search from becoming trapped at the second type of cycle.

This paper uses the secondary tabu list [16] to escape from the second type of cycling trap. The sum of the Euclidean distances within each cluster of any current solution forms a vector recorded as an element of the secondary tabu list. All the C values in such an element are then sorted in the decreasing order. The sum of the Euclidean distances within a cluster is not altered unless the elements composing the cluster are changed. Moreover, because the values (representing each cluster) are sorted in their size, it is not necessary to keep the label attached to each cluster. This makes it possible to prevent from the second type of cycle.

The role of the secondary tabu list is similar to that of the ordinary tabu list. At some search iteration, when a solution has the least objective function value, among the solutions in neighborhood, but it is not restricted by the tabu list, the algorithm is initiated to check whether or not the solution is contained in the secondary tabu list. If it is contained in the secondary tabu list, it cannot be any candidate for the next solution.

Aspiration condition. Since the tabu list restricts some moves, there is a risk even to restrict a good move. In order to get rid of this risk, the solution can be selected as the candidate for next solution, if the solution in a tabu state satisfies the aspiration condition. Accordingly, this algorithm considers the aspiration condition of solution $f(y) < f_{\min}$. This implies that if the objective function value of solution y is less than the current minimum objective function value, the solution is set free from the tabu and can rather be the candidate for the next solution.

Stopping condition. The reallocation procedure ends after a pre-determined number of iterations. In the experimental test, the number of iterations is set to $1000/\{2\cdot(\lfloor (n+1)/(C+1)\rfloor - 1)\}$ to compare with the algorithms in the literature.

4.2. Releasing procedure

As stated above, the packed-element-together move has some advantages. However, it may have such a drawback as a missing of some good solutions. Because the elements of a packed element move together, the solution space may not be fully searched to find any better solution. To compensate for such a drawback, this paper proposes a strategy of releasing such packing relations in an approach of releasing the packed pairs one at each iteration of the reallocation operation. In the approach, the last ranked (most distant) pair is first selected as a released pair whose elements are allowed separately to belong to different clusters. In other words, the released elements are allowed to move separately. Thus, any solution space reduced by the packing operation can rather be expanded by the releasing operation (conforming to a diversification strategy).

5. Step-by-step solution procedure

The proposed algorithm can be characterized as designed to improve the effectiveness of the tabu search by incorporating both the packing and releasing procedures together such that the packing procedure tends to play a role of focusing the solution search on good solution spaces and the releasing procedure tends to play a complementary role of making an intensive search on the focused solutions spaces.

The whole algorithm is now constructed below in a step-by-step procedure.

Step 0: Sort k pairs of elements in the increasing distance order, where $k = 2(\lfloor (n+1)/(C+1)\rfloor - 1)$.

Step 1: Find the initial solution (refer to Section 3.4).

Step 2: Search for an improved solution by keeping the k pairs packed (refer to Section 4.1.).

Step 3: If $k = 0$, stop, and the best solution up to now becomes the final solution.

Otherwise, go to Step 4.

Step 4: Release the kth packed pair.

$k = k - 1$.

Let the initial solution be the best solution found up to now.

Go to Step 2.

6. Experiment

In this paper, four algorithms are numerically tested for their effectivenesses with 360 data sets. They include the *K*-means algorithm [17], the simulated annealing algorithm by Selim [6], the tabu search algorithm by Al-Sultan [7], and the proposed algorithm of this paper in Section 5. They are implemented in *C* language on a Pentium II 400MHz Personnel Computer.

6.1. Data set

Since the effectivenesses of any clustering algorithms are usually dependent on data set, how to design their test data set is important as well. In the literature, some authors [18,19] have paid attention to their strategies of designing test data sets. In this paper, the design strategy of Milligan [19] is adapted to generate our test data set by incorporating five design factors including number of elements, number of clusters, number of dimensions (attributes), ratio of cardinality, and error type.

To design the data set, the boundaries for each dimension of the cluster is pre-determined. In this data set, no overlap is permitted. In order to satisfy the non-overlapping restriction, any cluster overlap is not permitted on the first dimension of the space. That is, the clusters are required to occupy disjoint regions of space. Then, all the elements assigned to a given cluster are required to fall within the boundaries for each dimension of the variable space.

Two types of data sets are considered, one having 50 elements and the other having 100. Four different clusters including 2, 3, 4, and 5 clusters are considered. Each element is allowed to vary in dimension such as to have four, six, and eight attributes. The cardinality factor is allowed to vary in element weight to have three different distribution patterns. In the first pattern, all elements are distributed among clusters as equally as possible. In the second and third patterns, one cluster in the data set has 10 and 60% of all the elements, respectively, while the other clusters have the rest of the elements being equally distributed among them. Five different types of error are considered. The first type of error represents an error-free data set. The second and third type of errors are concerned with some random noise elements added to the error-free data set. The second type error data set has 10 such random noise elements, and the third type of error data set has 20 such random noise elements. Each of the random noise elements has the same *K* attributes as the ordinary (error-free) input element, but it is like a random error in the sense that it is not clearly dedicated to any specific cluster. The fourth and fifth type errors are concerned with random noise attributes (dimensions) added to each error-free element so as to have additionally one and two more dimensions, respectively.

An example of the data generation procedure is illustrated in Fig. 1. At the first step, the boundaries for each dimension of the cluster is set. Then, the elements considering the first four design factors are allocated within the pre-determined boundaries. At the last step, the error types are considered. Fig. 1d shows the test data set having 50 elements, three clusters, two attributes, equal cardinality, and second type of error.

6.2. Results

It is known that the effectivenesses of the *K*-means algorithm, the simulated annealing algorithm, and the tabu search algorithm are all dependent greatly on initial solutions. Therefore, for every data set, those three algorithms are performed 10 times individually for their own effectiveness tests, each time with a new initial solution generated randomly from the data set. However, our proposed algorithm is tested only one time, since it has a fixed procedure to generate its own initial solution. Each test is made of at most 1000 iterations of the associated solution search procedure by each of the four different algorithms giving an equal chance for a fair performance comparison among the four algorithms.

The results show that the proposed algorithm of this paper finds a solution with the objective function value less than or equal to the mean of the objective function values obtained from 10 test runs by each of the other three algorithms. It is also seen that the tabu search

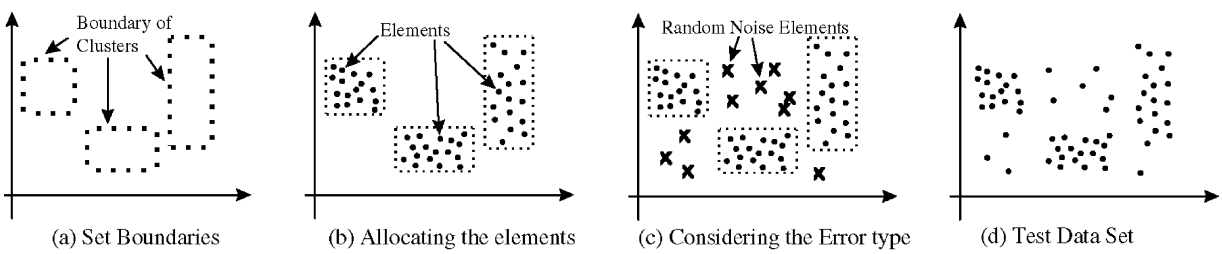

Fig. 1. Generating test data set.

algorithm outperforms the other two algorithms including the *K*-means algorithm and the simulated annealing algorithm, while the *K*-means algorithm and the simulated annealing algorithm show similar performances. This implies that our proposed algorithm shows its superiority in effectiveness to that of the tabu search algorithm.

In the above performance comparison, 10 test results are gathered, by use of a given data set, and averaged for each reference algorithm. In other words, multiple samples are gathered for each reference algorithm. Therefore, it may be desirable to investigate how statistically confident the superiority of the proposed algorithm is. This provides a motive of making a statistical hypothesis test for the three reference algorithms with the following null hypothesis;

H_0: the objective function value found by each of those reference algorithms is equal to that of our proposed algorithm

However, the reference algorithms are individually performed only 10 times and thier resulting objective function values are all discrete, so that the objective values are unlikely distributed in a Gaussian distribution. Thus, this paper wants to apply the Wilcoxon rank sum test, a non-parametric test. The SAS package is used for the test, and its results are summerized in from Tables 1–6.

The values in those tables denote the number of data sets in which the hypothesis is rejected at the confidence level 0.99 (rejected at 0.99), rejected at the confidence level 0.95 but accepted at the confidence level 0.99 (accepted at 0.99), accepted at the confidence level 0.95 (accepted at 0.95), respectively. The rejection at the confidence level 0.99 means that the solution found by the associated test algorithm gives a strong evidence of having a greater (not better) objective function value than the solution found by the proposed algorithm.

According to Table 1, the number of data sets rejected at 0.99 by the tabu search algorithm is apparently smaller than each of those by the *K*-means algorithm and the simulated annealing algorithm. This means that the solutions obtained by the tabu search algorithm in general have a tendency of being closer to the solutions obtained by the proposed algorithm than those by the *K*-means algorithm or the simulated annealing algorithm.

Table 1
The results of the statistical hypothesis test on the whole 360 data sets generated by considering all the design factors together

Algorithm	Number of data sets		
	Rejected at 0.99	Accepted at 0.99	Accepted at 0.95
K-means	224	47	89
S.A.	224	47	89
T.S.	71	43	246

The test results summarized in Tables 2–6 show the effects of the design factors. Table 2 shows that as the number of elements increases, the performance of the tabu search algorithm gets lowered, and the performance gap between the tabu search algorithm and the other two algorithms decreases. That is, as the number of elements increases, the proposed algorithm shows a more robust result than the tabu search algorithm. Table 3 shows that the number of data sets rejected at 0.99 increases as the number of clusters increases. Thus, we can infer from Table 3 that an increased number of clusters may induce a larger difference between the solution of the proposed algorithm and those of the other algorithms. Moreover, we can infer that the proposed algorithm may be more robust against any change in the number of clusters. Tables 4 and 5 show the effects of the number of dimensions and the ratio of cardinality. In both the tables, the solutions of the tabu search algorithm are closer to those of the proposed algorithm than any of those of the *K*-means algorithm and the Simulated Annealing algorithms. However, no

Table 2
The results of the statistical hypothesis test on the whole 360 data sets divided only by the "Number of Elements" factor

Number of elements	Algorithm	Number of data sets		
		Rejected at 0.99	Accepted at 0.99	Accepted at 0.95
50	*K*-means	107	23	50
	S.A.	128	11	41
	T.S.	29	22	129
100	*K*-means	117	24	39
	S.A.	96	36	48
	T.S.	42	21	117

Table 3
The results of the statistical hypothesis test on the whole 360 data sets divided only by the "number of clusters" factor

Number of clusters	Algorithm	Number of data sets		
		Rejected at 0.99	Accepted at 0.99	Accepted at 0.95
2	*K*-means	12	8	60
	S.A.	8	5	77
	T.S.	2	0	88
3	*K*-means	54	24	12
	S.A.	53	29	8
	T.S.	13	12	65
4	*K*-means	75	13	2
	S.A.	80	9	1
	T.S.	22	16	52
5	*K*-means	83	2	5
	S.A.	83	4	3
	T.S.	34	15	41

Table 4
The results of the statistical hypothesis test on the whole 360 data sets divided only by the "number of dimensions" factor

Number of dimensions	Algorithm	Number of data sets		
		Rejected at 0.99	Accepted at 0.99	Accepted at 0.95
4	*K*-means	73	19	28
	S.A.	81	11	28
	T.S.	24	14	82
6	*K*-means	74	15	31
	S.A.	72	17	31
	T.S.	22	17	81
8	*K*-means	77	13	30
	S.A.	71	19	30
	T.S.	25	12	83

Table 5
The results of the statistical hypothesis test on the whole 360 data sets divided only by the "ratio of cardinality" factor

Ratio of cardinality	Algorithm	Number of data sets		
		Rejected at 0.99	Accepted at 0.99	Accepted at 0.95
Equal cardinality	*K*-means	84	13	23
	S.A.	71	29	30
	T.S.	15	13	92
10% in one cluster	*K*-means	78	14	28
	S.A.	75	17	28
	T.S.	16	18	86
60% in one cluster	*K*-means	62	20	38
	S.A.	78	11	31
	T.S.	40	12	68

Table 6
The results of the statistical hypothesis test on the whole 360 data sets divided only by the "error-type" factor

Type of errors	Algorithm	Number of data sets		
		Rejected at 0.99	Accepted at 0.99	Accepted at 0.95
First-type error	*K*-means	28	7	37
	S.A.	41	11	20
	T.S.	11	6	54
Second-type error	*K*-means	52	8	12
	S.A.	46	12	14
	T.S.	12	6	54
Third-type error	*K*-means	44	15	13
	S.A.	46	11	15
	T.S.	19	10	43
Fourth-type error	*K*-means	42	10	20
	S.A.	42	10	20
	T.S.	15	8	49
Fifth-type error	*K*-means	47	7	18
	S.A.	49	3	20
	T.S.	14	12	46

peculiar trend associated with the design factors is shown. Table 6 shows the effect of the error types. The solutions of the first error type (error-free data set) of each algorithm show slightly closer to the solution of the proposed algorithm than those of the other error types of each algorithm. It means that the error perturbation deteriorates the effectiveness of each algorithm, so that the proposed algorithm is more robust in terms of error perturbation.

The average elapsed time for each algorithm is shown in Table 7. It shows that the proposed algorithm takes the longest time. This may be due to the additional time requirement for updating the packing relation. As discussed earlier, the number of the packing relation updatings depends on the number of the releasing operations. The elapsed time may be shortened by reducing the number of the releasing operations rather than by releasing all the packed elements as done in this paper. Such releasing-operation reduction is left as a further research issue.

Table 7
The average elapsed time of each algorithm (s)

Our algorithm	T.S.	S.A.	*K*-means
2.893	0.856	0.782	0.169

7. Conclusion

This paper proposes an efficient heuristic algorithm for a clustering problem by employing the tabu search method which is combined with two newly exploited procedures (called packing and releasing procedures) for both the solution search efficiency and effectiveness. Moreover, a secondary tabu list is considered to prevent the associated search from becoming trapped at any local optimal solution. The heuristic algorithm is numerically tested for its effectiveness and shown that it outperforms over all the three reference works including the tabu search algorithm, the *K*-means algorithm, and the simulated annealing algorithm.

The results of this paper may be immediately applied to various information data analyses concerned with information classification and pattern recognition. For example, the proposed algorithm may be applied to various practical classification subjects including patient classification, product distribution center allocation, and government service branch organization. Moreover, it may be feasible to use the results of this paper as a guideline for designing any decision boundary (classifier).

An efficient adaptation of the proposed algorithm for any situation of allowing the number of clusters to be another decision variable may be an interesting subject for further study.

References

[1] J.A. Hartigan, Clustering Algorithms, Wiley, New York, 1975.

[2] S.Z. Selim, M.A. Ismail, *K*-means-type algorithms: a generalized convergence theorem and characterization of local optimality, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1984) 81–87.

[3] P. Brucker, On the complexity of the clustering problem, Optimierung und operations research, Lecture notes in Economics and Mathematical Systems, Springer, Berlin, 1978.

[4] W.S. DeSarbo, R.L. Oliver, A. Rangaswamy, A simulated annealing methodology for clusterwise linear regression, Psychometrika 54 (1989) 707–736.

[5] R.W. Klein, R.C. Dubes, Experiments in projection and clustering by simulated annealing, Pattern Recognition 22 (1989) 213–220.

[6] S.Z. Selim, K.S. Al-Sultan, A simulated annealing algorithm for the clustering problem, Pattern Recognition 24 (1991) 1003–1008.

[7] K.S. Al-Sultan, A tabu search approach to the clustering problem, Pattern Recognition 28 (1995) 1443–1451.

[8] F. Glover, M. Laguna, Tabu search, in: C. Reeves (Ed.), Modern Heuristic Techniques for Combinatorianl Problems, Blackwell Scientific Publishing, Oxford, 1993, pp. 70–141.

[9] H.D. Vinod, Integer programming and theory of grouping, J. Am. Statist. Assoc. 64 (1969) 506–519.

[10] M.R. Rao, Cluster analysis and mathematical programming, J. Am. Statist. Assoc. 66 (1971) 622–626.

[11] F. Glover, Tabu search-part I, ORSA J. Comput. 1 (1989) 190–206.

[12] F. Glover, Tabu search-part II, ORSA J. Comput. 2 (1990) 4–32.

[13] F. Glover, E. Taillard, D. Werra, A user's guide to tabu search, Ann. Oper. Res. 41 (1993) 3–28.

[14] F. Glover, Tabu search and adaptive memory programming – advances, applications and challenges, in: Barr, Helgason, Kennington (Eds.), Interfaces Comput. Sci. Oper. Res., Kluwer Academic Publishers, 1996.

[15] J. Skorin-Kapov, Extensions of a tabu search adaptation to the quadratic assignment problem, Com. O.R. 21 (1994) 855–865.

[16] M. Gendreau, P. Soriano, L. Salvail, Solving the maximum clique problem using a tabu search approach, Ann. Oper. Res. 41 (1993) 385–403.

[17] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[18] K.S. Al-Sultan, M. Maroof Kahn, Computational experience on four algorithms for clustering, Pattern Recogn. Lett. 17 (1996) 295–308.

[19] G.W. Milligan, An examination of the effect of six types of error perturbation on fifteen clustering algorithms, Psychometrika 45 (1990) 325–342.

About the Author—CHANG SUP SUNG received the M.S. degree in Industrial Engineering from Iowa State University, USA, in 1974 and the Ph.D. in Industrial Engineering from Iowa State University, USA, in 1978. He is a professor of Industrial Engineering at Korea Advanced Institute of Science and Technology (KAIST) in Taejon, Korea. His areas of research interests include Clustering, Production Planning and Control, Production Scheduling, Telecommunication Network Modeling and Evaluation, Combinatorial Optimization and Network Theory, and Logistics.

About the Author—HYUN WOONG JIN received the B.S. degree in Statistics from Yonsei University in Seoul, Korea, in 1993 and the MS degree in Industrial Engineering from KAIST, Korea, in 1996. He is a student at the doctoral level at the Department of Industrial Engineering of KAIST. His research interests include Clustering, Optimization Theory, and Network Theory.