THE IDEA OF DE-CLUSTERING AND ITS APPLICATIONS

M. T. Fang*, R. C. T. Lee* and C. C. Chang**

* Institute of Computer and Decision Sciences, National Tsing Hua University , Hsinchu, Taiwan, CHINA
** Institute of Applied Mathematics, National Chung Hsing University, Taichung, Taiwan, CHINA

ABSTRACT

The idea of de-clustering is defined as follows: Given a set of data, divide them into two groups such that these two groups are similar to each other. We shall show that this de-clustering technique can be used to solve the multi-disk data allocation problem for parallel processing.

SECTION 1. INTRODUCTION

The problem of clustering is defined as follows: Given a set of data, divide these data into homogeneous groups. Clustering analysis and its applications have been studied by many researchers [Lee81, Zahn71, Sibs73, Rohl73, Dube80].

It often occurs that we have a situation which is exactly opposite to that where clustering is required. For instance, suppose that there are a group of students and we like to divide these students into two teams to play, say basketball. Certainly, in this case, clustering analysis can not be applied. If it is applied, one team will consist of tall and strong kids while the other team will consist of small and weak kids. Actually, in such a situation, we like to have two groups of kids which are similar to each other as a whole. That is, the average height, weight and skill of one team should be quite close to those of the other team. If there are strong and tall kids in one team, there should be such kids in the other team. Similarly, if there are weak and short kids in one team, such kids should also exist in the other team.

Consider another situation. We have a large company which has many branches in different cities. Some of the branches are located in very affluent cities and some are located in poverty-stricken cities. Let us assume that the company has two managers and each manager is in charge of a group of branches. Again, we have to be sure that these two groups are smilar to each other. That is, the number of affluent cities of one group should be roughly the same as that of the other group. Only through this arrangement we can truly compare the capabilities of these two managers.

In this paper, we shall discuss a de-clustering technique and then show how this de-clustering technique can be used to solve a multi-disk data allocation problem.

SECTION 2. A DE-CLUSTERING ALGORITHM BASED UPON MINIMAL SPANNING TREES

One way of defining two similar groups G_1 and G_2 is the following. For every point P in G_1, there exists at least one point Q in G_2 such that either P is a nearest neighbor of Q or Q is a nearest neighbor of P. To produce such two groups, we may use minimal spanning trees [Dijk59, Prim57, Yao75, Cher76, Bent78]. Let us assume that all of inter-point distances are different. In this case, if P is a nearest neighbor of Q, then P and Q are connected in the minimal spanning tree.

To produce two groups which are similar to each other, we may do the following.

(1) Construct a minimal spanning tree for a set S of points.
(2) Pick any point, say A, from S and mark the level of A as 1.
(3) For any other node on the minimal spanning tree whose level is not marked yet, mark its level as i+1 if it is linked to a node whose level is already marked as i.
(4) Repeat step 3 until every node is marked.
(5) Let G_1 consist of all points whose levels are odd numbers and G_2 consist of all points whose levels are even numbers.

Example 2-1.

Consider the points in Fig. 2-1.

Proceedings of the Twelfth International Conference on Very Large Data Bases **Kyoto, August, 1986**

Fig. 2-1 here

Suppose we choose C to start with. The levels of all nodes will be shown in Fig. 2-2.

Fig. 2-2 here

In this case, G_1 = {C, E, G, H} and G_2 = {A, B, D, F, I}. The reader can see that for every point in G_1, there is a point in G_2 such that these two points form a nearest neighbor pair. For instance, C, D is such a pair and {G, F} is also such a pair.

To illustrate the de-clustering effect, we shall use "x" and "o" to indicate points in G_1 and G_2 respectively and redraw Fig. 2-1 as Fig. 2-3. The reader can see that we have produced two groups of points which are similar to each other.

Fig. 2-3 here

Example 2-2.

Fig. 2-4 shows another example of using the de-clustering technique based upon the minimal spanning trees. The effect of de-clustering is easy to see and understand.

Fig. 2-4 here

The readers may have noticed that the number of points of G_1 is not equal to those of G_2. In the next section, we will propose another approach to avoid this problem.

SECTION 3. DE-CLUSTERING BASED UPON SHORT SPANNING PATHS

One disadvantage of using the minimal spanning tree approach is that the number of points of one group is not necessarily equal to those of the other group. In this section, we shall show that we can also use the concept of short spanning tree paths to de-cluster data.

A shortest spanning path is defined as a shortest path connecting all of the data points. A shortest spanning path is hard to find because the problem of finding such a path is NP-complete [Papa76]. In Lee [Lee81] an algorithm to find a short, not necessarily the shortest, path was given.

A short spanning path produces a unique ordering of points. Let the points be labeled according to their positions in the path. We may then put points with odd numbers into one group and points with even numbers into another group. If this method is used, the difference between the numbers of these two troups is at most 1.

Example 3-1.

Consider the points in Fig. 2-1 again. A short spanning path is given in Fig. 3-1. In this case, we shall have two groups: {A, C, E, F, I} and {B, D, G, H}.

Fig. 3-1 here

Example 3-2.

Consider the points in Fig. 2-4 again. The result after de-clustering using the short spanning path approach is shown in Fig. 3-2. The reader can see easily that de-clustering using short spanning path produces two equal sized groups.

Fig. 3-2 here

SECTION 4. THE DE-CLUSTERING EFFECT

In this section, we should discuss whether our de-clustering techniques work or not.

If a de-clustering technique is effective, it should produce just exactly the opposite effect of clustering analysis. Therefore, in the following, we should first discuss the effects of clustering analysis.

Assume that we have a set S of data. If clustering analysis produces two groups S_1 and S_2 from S, then S_1 and S_2 will have the following properties:

(1) Both S_1 and S_2 will be quite different from the original set S.
(2) S_1 and S_2 should be quite different from each other.

Because de-clustering should produce opposite effects as clustering does, we expect that after de-clustering produces S_1 and S_2, S_1 and S_2 will have the following properties:

(1) S_1 and S_2 should be quite similar to the original set S.
(2) S_1 and S_2 should be quite similar to each other.

To measure the similarity between two sets S_1 and S_2 which are mutually exclusive, we may do the following: Let $S = S_1 \cup S_2$. For each point P_i in S_1, find Q_i which is a nearest neighbor of P_i. Let (P_i, Q_i) be denoted as a within-sets closest pair if Pi and Qi belong to the same set S_i. If S_1 and S_2 are similar to each other, the number of within-sets closest pairs should be small.

Therefore, we may use the number of within-sets closest pairs to indicate the similarity between S_1 and S_2. That is, if S_1 and S_2 are two mutually exlusive sets, then $d(S_1,S_2)$ is the number of within-sets closest pairs.

If de-clustering produces S_1 and S_2 from S, then we like both S_1 and S_2 to be able to represent S. Let X_i be a point in S. Let Y_j be a nearest neighbor of X_i in S_i. Let $d(X_i,Y_j)$ be the distance between X_i and Y_j. Let $d(S,S_i) = \sum_{i=1}^{N} d(X_i,Y_j)$. Clearly, if $d(S,S_i)$ is small, then S_i represent S. In other words, if S_i is a subset of S, then $d(S,S_i)$ measures the similarity between S and S_i.

Experiment 1.

In this experiment, we tested the effectiveness of using the minimal spanning trees to de-cluster data. For each data set, we used two methods to de-cluster these data. The first method was the minimal spanning trees approach. The second method was simply using the random approach. That is, the data were de-clustered by randomly assigning data into two groups. The purpose of this experiment was to test whether we can randomly produce two similar subsets of data and whether each subset can represent the original set of data.

Let the number of points be denoted as N. In this experiment, it was set to be euqal to 100, 200, ..., 1000. For each N, five data sets of 2-dimensional points were generated by a random number generator. Thus, there were totally fifty sets of data used.

For each set, we first used the minimal spanning tree approach to de-cluster the data. Let the two groups of data be denoted as A and B. Let the original set of data be denoted as S. Then d(A,B), d(S,A) and d(S,B) were all calculated. Since A and B were generated by using the minimal spanning trees approach, as expected, d(A,B) was zero for each set of data.

For each data set, let the sizes of A and B produced by the minimal spanning trees approach be N_A and N_B respectively. We then used the random number generator to produce two subsets A and B of sizes N_A and N_B respectively. Again, d(A,B), d(S,A) and d(S,B) where clculated.

Note that for a good de-clustering technique, d(A,B), d(S,A) and d(S,B) should be small.

The experimental results are shown in Table 4-1. The last item of each row is the number of times that the distance produced by the minimal spanning trees approach is smaller than that produced by the random method. From Table 4-1, we can see that the minimal spanning trees approach was better than the random method in each of the three comparisons. That is, we may conclude that the minimal spanning trees approach produces two subsets which are very similar to each other (d(A,B) = 0) and they both represent the original data better than randomly selecting two groups of data.

Table 4-1 here

Experiment 2.

In this experiment, instead of minimal spanning trees, we used the short spanning paths. The experimental results are shown in Talbe 4-2. We may draw the following conclusions:

(a) As compared with the minimal spanning trees approach, the short spanning paths approach produces two groups which are less similar to each other. For the minimal spanning trees approach, d(A,B) is zero. This is not the case for the short spanning paths approach, as expected.

(b) The short spanning path de-clustering produced two subsets of data which were less representative than those produced by the minimal spanning trees approach. But the difference is very small, in general.

(c) The de-clustering method based upon the short spanning path method produced two groups of data which are much more similar to each other than the groups produced by the random method, as shown by the distance between A and B.

Table 4-2 here

SECTION 5. THE APPLICATION OF DE-CLUSTERING TO THE MULI-DISK DATA ALLOCATION PROBLEM

The multi-disk data allocation problem was first proposed by Du and Sobolewski [Du82] and studied by some researchers [Wu83, Du84a, Du84b, Chan85]. The problem can be explained by considering the following example. Assume that there are sixteen records as shown in Table 5-1.

Table 5-1 here

Table 5-2 shows one way of assigning the records into the two disks.

Table 5-2 here

Let us denote a partial match query which retrieves all records with the first attribute equal to A and the second attribute equal to anything as (A, *). Then when this query is issued, all of the records which are to be retrieved are in Disk 1. That is, Disk 2 is idling all the time as data are retrieved from Disk 1. Let us assume that the retrieving of one record takes one unit of time, Then, for the above configuration, the retrieval will take four units of time. The retrieval of (*, A) will take two units of time..

Let us consider another way of assigning data as shown in Table 5-3.

Table 5-3 here

In this case, for any partial match query, it will always take two units of time to complete. Thus, we may say that the above way of allocating data is more effective.

If one compares the above two ways of allocating data, one should note that the second method is more effective because it employs the de-clustering concept while the first allocation method employs the clustering method.

If we have a set of data and we have to allocate them onto some disks in such a way that we can retrieve them in parallel, we should distribute similar records onto different disk. This is why de-clustering can be applied.

Example 5-1.

Consider a 2-attribute Cartesian product file [Chan80, Chan84] as shown in Table 5-1. Applying the de-clustering algorithm based upon the minimal spanning tree, we obtained the data allocation scheme exactly as in Table 5-3, which is optimal.

Example 5-2.

Consider the data in Table 5-4. This set of data are from a paper written by Dayhoff [Dayh69] and are concerned with the sequence of amino acids in a protein molecule, cytochrome c, found in the mitochondria of animals and heigher plants. Only the positions which vary are recorded. The result after de-clustering based upon the minimal spanning tree is shown in Table 5-5. We measure the effectiveness of our method by examining every attribute. For instance, consider the first attribute. This attribute assumes two values: V and I. As can be seen, there are two records with the first attribute equal to I and distributed onto different disks. Similarly, there are eleven records whose first attributes are equal to V. Again, five of them in one disk and six of them in another disk. For attribute 1, our allocation is optimal as the distribution is totally balanced. By examining all of the attributes, we conclude that our allocation method is not optimal only for Atrributes 12, 16 and 18.

Table 5-4 here

Table 5-5 here

Example 5-3.

Let us consider a three-attribute file as shown in Table 5-6. In fact, they are data from [Lee79]. The three attributes are department code, skill code and salary code.

Table 5-6 here

We use Monte Carlo simulation method as the basis of the random allocation scheme. A random allocation is constructed by randomly assigning records into disks. A random number generator is used to put all the twenty eight records in Table 5-6 onto two available disks. This allocation is a kind of random allocation. The reader should note that talking about the performance of a particular random allocation is nonsense because a random allocation can be quite "good" or quite "poor" in its performance. So we have to calculate the expected performance of random allocation scheme. By the expected performance of random allocation scheme, we mean the expected number of buckets to be accessed in parallel over all possible partial match queries. In this case, we calculate the expected performance of random allocation scheme as the average performance of fifteen random allocations.

The experimental results of applying De-clusterings based upon the minimal spanning tree and the short spanning path and random allocation scheme are as shown in Table 5-7.

Table 5-7 here

As can be easily seen, our de-clustering techniques based upon the minimal spanning tree and the short spanning path are better than their corresponding random allocations.

SECTION 6. CONCLUSIONS AND REMARKS

De-clustering is a new idea in analyzing the similarities between data set and is exactly opposite to clustering analysis. Clustering analysis divides data into homogeneous groups while de-clustering distributes similar data into different groups.

In database design, clustering analysis can be used to group similar

records into one bucket to shorten retrieval time. On the contrary, de-clustering is used to distribute similar records into different disks and also can be used to reduce response time to a query for parallel accessing.

From our experience, since de-clustering produces two subsets that represent the original data set, it appears to us that it can be used as a sampling technique in Statistics.

Finally, we believe that de-clustering is not only good for parallel partial matching, but also good for parallel nearest neighbor searching [Frie75, Shen78, Bent80, Lee82].

REFERENCES

[Bent78] J.L. Bentley and J.H. Friedman, "Fast algorithms for constructing minimal spanning trees in coordinate spaces", IEEE Transactions on Computers, C-27, 97-105 (1978).

[Bent80] J.L. Bentley, B.W. Weide and A.C.C. Yao, "Optimal Expected Time Analysis for Closest Point Problem", ACM Transactions on Mathematic Software, Vol. 6, No. 4, 563-580 (1980).

[Chan80] C.C. Chang, R.C.T. Lee and H.C.Du, "Some Properties of Cartesian Product Files", Proc. of ACM-SIGMOD 1980 Conf., Santa Monica, Calif., 157-168 (1980).

[Chan84] C.C. Chang, M.W. Du and R.C.T. Lee, "Performance Analysis of Cartesian Product Files and Random Files", IEEE Transactions of Software Engineering, Vol. SE-10, No. 1, 88-99 (1984).

[Chan85] C.C. Chang and J.C. Shen, "Performance Analysis of the Disk Modulo Allocation Method for Concurrent Accessing on Multiple-Disk Systems", Accepted by Journal of the Chinese Institute of Engineers (1985).

[Cher76] D. Cheriton and R.E. Tarjan, "Finding minimum spanning trees", SIAM Journal of Computing 5, 724-742 (1976).

[Dayh69] M.O. Dayhoff, "Computer Analysis of Protein Evolution", Scientific American, 232(7), July, 69-85 (1969).

[Dijk59] E.W. Dijkstra, "A note on two Problems in connexion with graph", Numerische Mathematik 1. 269-271 (1959).

[Du82] H.C. Du and J.S. Sobolewski, "Disk Allocation for Cartesian Product Files on Multiple-Disk System", ACM Transactions on Database Systems, Vol. 7, No. 1, 82-101 (1982).

[Du82a] H.C. Du and F. Naveda, "On the Complexities of Several Physical Database Design Problems", Technical Report 84-3, Department of Computer Science, University of Minnesota (1984).

[Du84b] H.C. Du "On the File Allocation for Power-2 Cartesian Product Files", Technical Report 84-8, Department of Computer Science, University of Minnesota (1984).

[Dube80] R. Dubes and A.K. Jain, "Clustering methodologies in exploratory data analysis", In Advances in Computers, Edited by M.C. Yovits, Vol. 19, Academic Press, New York (1980).

[Frie75] J.H. Friedman, F. Baskett and L.J. Shustek, "An Algorithm for Finding Nearest Neighbors", IEEE Transactions on Computers, Vol. 24, No. 10, 1000-1006 (1975).

[Lee81] R.C.T. Lee, "Clustering Anslysis and its applications", Advances in Information System Science, Edited by J.T. Tou, Vol. 8, 169-292, Plenum Press, New York (1981).

[Lee82] R.C.T. Lee, C.J. Lee and C.W. Shen, "Recent Developments of Nearest Neighbor Searching Techniques", Proc. First Conf. Comput. Algorithms, Hsinchu, Taiwan, 2-2-2-74 (1982).

[Lee79] R.C.T. Lee and S.H. Tseng, "Multi-key Sorting", Policy Analysis and Information Systems, Vol. 3, No. 2, 1-20 (1979).

[Papa76] C. Papadimitriou and K. Steiglitz, "Some complexity results for the traveling salesman problem", Proc. Eightn Annual ACM Symposium on Theory of Computing, 1-9 (May 1976).

[Prim57] E.C. Prim, "Shortest connection network and some generalization", Bell System Tech. J., 36, 1389-1401 (Nov. 1957).

[Rolh73] F.J. Rolhf, "Algorithm 76: hierarchical clustering using the minimum spanning tree", The Computer Journal 16, 93-95

(1973).

[Shen78] C.W. Shen and R.C.T. Lee, "A Nearest Neighbor Search Technique with Zero-in Time", IEEE Proc. 1978 Int. Comput. Software Appl. Conf., 470-475 (1978).

[Sibs73] R. Sibson, "SLINK: an optimally efficient algorithm for the single-link cluster method", The Computer Journal 16, 30-34 (1973).

[Wu83] C.T. Wu, "Associative Searching in Multiple Storeage Units", Computer Science Technical Report No. CS-072, University of California, San Diego (1983).

[Yao75] A.C. Yao, "An O(|E| log log |V|) algorithm for finding minimum spanning trees", Information Processing Letters 4, 21-23 (1975).

[Zahn71] C.T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters", IEEE Transactions on Computers, C-20 (1) 68-86 (Jan. 1971).

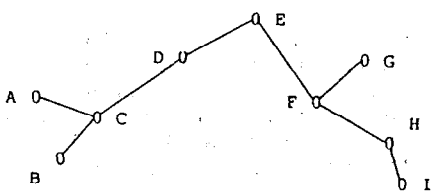

Fig. 2-1. Points in the minimal spanning tree

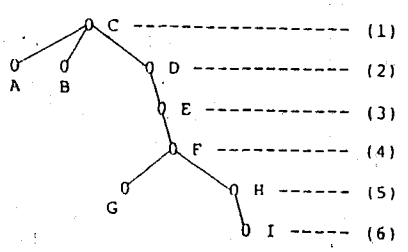

Fig. 2-2. Levels of the minimal spanning tree

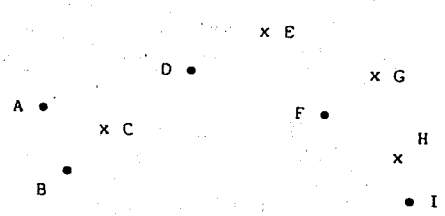

Fig. 2-3. The de-clustering effect of Fig. 2-1.

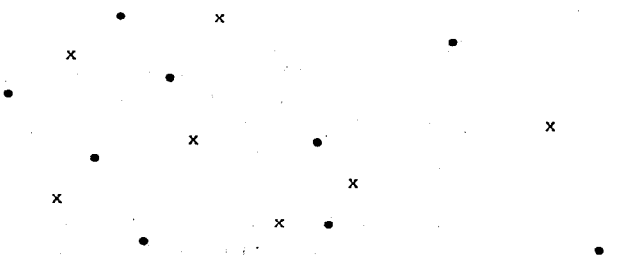

Fig. 2-4. De-clustering effect based on minimal spanning tree

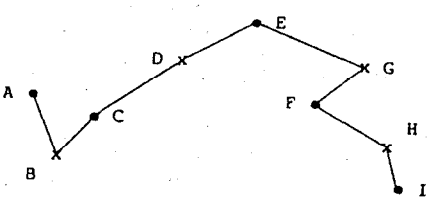

Fig. 3-1. Short spanning path of Fig. 2-1.

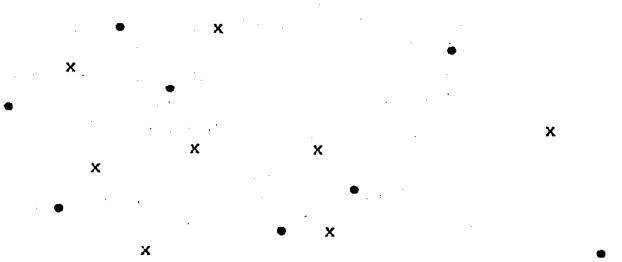

Fig. 3-2. De-clustering effect of Fig. 2-4. based on short spanning path

Table 4-1. The results of Experiment 1.

Method:		MST			RANDOM			SCORE
N	Itr	d(A,B)	d(S,A)	d(S,B)	d(A,B)	d(S,A)	d(S,B)	
100	1	0	13.1	14.1	43	16.5	21.3	3
	2	0	13.2	14.8	46	14.8	21.4	3
	3	0	13.6	13.4	62	20.9	20.9	3
	4	0	12.8	12.0	48	17.0	18.5	3
	5	0	13.2	14.0	51	23.9	20.9	3
200	1	0	9.5	8.8	101	13.4	12.2	3
	2	0	8.6	9.9	100	12.1	14.5	3
	3	0	9.8	8.4	105	14.2	11.7	3
	4	0	9.2	8.8	104	13.3	13.5	3
	5	0	8.7	8.7	114	12.4	12.3	3
300	1	0	7.7	7.6	159	11.3	10.5	3
	2	0	7.2	7.6	166	10.5	11.8	3
	3	0	7.8	7.2	143	10.6	9.4	3
	4	0	7.1	6.9	146	10.4	9.9	3
	5	0	7.0	7.4	133	9.1	9.8	3
400	1	0	6.8	6.0	211	9.5	8.4	3
	2	0	6.4	6.0	180	8.3	7.9	3
	3	0	6.5	6.9	192	8.8	9.6	3
	4	0	6.7	6.4	220	9.8	9.6	3
	5	0	6.1	6.5	195	8.6	9.4	3
500	1	0	5.4	5.6	231	7.7	8.0	3
	2	0	5.6	5.7	262	7.8	7.9	3
	3	0	5.9	5.6	278	8.6	8.5	3
	4	0	5.5	5.7	268	7.9	8.6	3
	5	0	6.3	5.5	249	8.6	7.9	3
600	1	0	5.4	5.2	314	7.6	7.2	3
	2	0	5.3	5.1	294	7.7	7.2	3
	3	0	5.0	5.2	312	7.0	7.9	3
	4	0	5.2	5.2	315	7.6	7.5	3
	5	0	5.0	5.1	301	6.9	7.3	3
700	1	0	4.7	4.8	349	6.6	7.0	3
	2	0	5.0	5.9	368	6.9	6.9	3
	3	0	4.4	4.6	391	6.8	6.9	3
	4	0	4.7	4.9	333	6.7	6.8	3
	5	0	4.9	4.5	349	6.8	6.5	3
800	1	0	4.7	4.4	377	6.3	6.2	3
	2	0	4.6	4.7	428	6.7	7.0	3
	3	0	4.3	4.5	372	6.0	6.3	3
	4	0	4.5	4.4	432	6.7	6.3	3
	5	0	4.4	4.6	369	6.2	6.4	3
900	1	0	4.1	4.4	483	5.8	6.5	3
	2	0	4.3	4.3	453	6.3	6.2	3
	3	0	4.2	4.4	440	6.2	6.2	3
	4	0	4.3	4.1	449	6.0	6.0	3
	5	0	4.2	4.3	433	5.9	5.9	3
1000	1	0	3.9	4.2	495	5.7	5.8	3
	2	0	3.9	4.0	489	5.6	5.3	3
	3	0	3.7	4.1	501	5.5	5.8	3
	4	0	4.1	3.8	507	6.1	5.4	3
	5	0	4.1	3.9	515	5.9	5.7	3

Table 4-2. The results of Experiment 2.

Method:		SSP			RANDOM			SCORE
N	Itr	d(A,B)	d(S,A)	d(S,B)	d(A,B)	d(S,A)	d(S,B)	
100	1	14	14.2	14.1	41	19.5	17.2	3
	2	10	14.2	14.7	53	20.1	18.8	3
	3	11	13.9	14.2	52	21.7	20.0	3
	4	12	13.3	12.9	50	17.7	17.2	3
	5	16	14.6	13.8	57	20.4	19.4	3
200	1	26	9.4	9.9	92	12.5	12.7	3
	2	24	9.2	9.6	92	12.5	12.3	3
	3	21	9.3	9.8	101	14.6	13.2	3
	4	27	9.3	9.3	82	12.3	12.1	3
	5	32	9.1	8.9	92	12.8	12.9	3
300	1	38	8.0	8.0	143	10.6	10.6	3
	2	53	7.9	7.7	141	10.5	9.9	3
	3	35	7.7	7.6	154	11.2	11.1	3
	4	47	7.4	7.2	159	9.6	10.3	3
	5	32	7.4	7.5	161	10.2	10.3	3
400	1	56	6.4	6.6	189	8.7	8.5	3
	2	53	6.5	6.4	179	8.8	8.1	3
	3	44	6.9	6.8	203	9.3	9.2	3
	4	65	6.8	6.9	190	9.0	9.0	3
	5	52	6.7	6.8	199	9.1	9.0	3
500	1	54	5.8	5.8	257	8.3	8.0	3
	2	68	5.7	5.9	321	7.5	7.8	3
	3	80	5.8	5.7	234	7.7	7.8	3
	4	62	5.8	5.8	259	8.3	8.3	3
	5	56	5.9	5.9	246	8.0	8.1	3
600	1	68	5.5	5.4	301	7.3	7.5	3
	2	88	5.4	5.5	284	7.4	7.4	3
	3	91	5.4	5.5	277	7.2	7.1	3
	4	73	5.5	5.4	307	7.3	7.3	3
	5	89	5.2	5.3	278	7.4	7.2	3
700	1	83	5.1	5.0	348	6.9	7.0	3
	2	93	4.9	5.0	341	6.6	6.5	3
	3	92	4.8	4.8	355	6.7	6.7	3
	4	87	4.9	5.0	342	7.1	7.0	3
	5	101	4.9	4.7	349	6.3	6.8	3
800	1	111	4.8	4.7	422	6.5	6.7	3
	2	102	4.6	4.6	419	6.7	6.4	3
	3	116	4.6	4.6	380	6.0	6.2	3
	4	103	4.6	4.7	395	6.3	6.2	3
	5	116	4.8	4.7	421	6.3	6.4	3
900	1	130	4.3	4.4	478	6.2	5.8	3
	2	108	4.4	4.5	487	6.1	6.3	3
	3	111	4.4	4.4	429	5.8	5.8	3
	4	132	4.5	4.3	461	6.1	6.1	3
	5	132	4.4	4.4	432	6.0	6.1	3
1000	1	149	4.2	4.2	412	5.6	5.9	3
	2	170	4.3	4.3	463	5.5	5.4	3
	3	116	4.1	4.0	522	5.9	5.7	3
	4	134	4.2	4.2	479	5.6	5.5	3
	5	117	4.0	4.1	534	5.8	5.9	3

(A,A)	(A,B)	(A,C)	(A,D)
(B,A)	(B,B)	(B,C)	(B,D)
(C,A)	(C,B)	(C,C)	(C,D)
(D,A)	(D,B)	(D,C)	(D,D)

Table 5-1. 2-attribute records

Disk 1	Disk 2
(A,A)	(C,A)
(A,B)	(C,B)
(A,C)	(C,C)
(A,D)	(C,D)
(B,A)	(D,A)
(B,B)	(D,B)
(B,C)	(D,C)
(B,D)	(D,D)

Table 5-2. One allocation result of Table 5-1.

Disk 1	Disk 2
(A,A)	(A,B)
(A,C)	(A,D)
(B,B)	(B,A)
(B,D)	(B,C)
(C,A)	(C,B)
(C,C)	(C,D)
(D,B)	(D,A)
(D,D)	(D,C)

Table 5-3. Optimal allocation of Table 5-1.

V	I	M	S	V	T	H	L	F	P	Y	S	A	I	G	D	V	K	E	A	K	T	N	E
V	I	M	S	V	T	H	L	F	P	Y	S	A	T	G	D	V	K	E	A	K	T	N	E
V	V	Q	A	V	T	H	L	F	P	F	T	D	T	K	E	A	K	T	E	K	T	N	E
V	V	Q	A	V	T	H	L	F	P	F	S	D	T	G	E	A	K	G	E	K	T	N	E
V	V	Q	A	V	T	H	L	F	P	F	S	D	T	G	E	A	T	G	A	K	T	K	E
V	V	Q	A	V	T	H	L	F	P	F	S	D	T	G	E	A	K	G	A	K	T	N	E
V	V	Q	A	V	T	H	L	F	P	F	S	D	T	G	D	A	K	D	A	K	T	N	E
V	V	Q	A	V	T	N	L	F	P	F	T	D	I	G	D	A	K	G	A	K	T	N	E
I	V	Q	S	V	T	H	L	F	E	F	S	D	T	G	D	A	K	S	V	D	T	S	K
I	V	Q	S	V	T	H	L	F	E	F	S	D	T	G	D	A	K	S	A	D	T	S	K
V	V	Q	S	V	T	H	L	F	E	F	S	D	T	G	D	A	K	S	A	D	T	A	K
V	V	Q	A	V	T	N	L	I	E	F	S	D	T	G	E	A	K	A	A	D	T	S	K
V	V	Q	A	C	V	Y	L	I	A	F	S	E	T	G	D	A	K	G	O	S	C	S	K

Table 5-4. Data from Dayhoff

Attr.	0 1	0 2	0 3	0 4	0 5	0 6	0 7	0 8	0 9	1 0	1 1	1 2	1 3	1 4	1 5	1 6	1 7	1 8	1 9	2 0	2 1	2 2	2 3	2 4
D	V	I	M	S	V	T	H	L	F	P	Y	S	A	T	G	D	V	K	E	A	K	T	N	E
i	V	V	Q	A	V	T	H	L	F	P	F	T	D	T	K	E	A	K	T	E	K	T	N	E
s	V	V	Q	A	V	T	H	L	F	P	F	S	D	T	G	E	A	K	G	A	K	T	N	E
k	V	V	Q	A	V	T	N	L	F	P	F	T	D	I	G	D	A	K	G	A	K	T	N	E
	I	V	Q	S	V	T	H	L	F	E	F	S	D	T	G	D	A	K	S	V	D	T	S	K
1	V	V	Q	S	V	T	H	L	F	E	F	S	D	T	G	D	A	K	S	A	D	T	A	K
	V	V	Q	A	C	V	Y	L	I	A	F	S	E	T	G	D	A	K	G	O	S	C	S	K
D	V	I	M	S	V	T	H	L	F	P	Y	S	A	I	G	D	V	K	E	A	K	T	N	E
i	V	V	Q	A	V	T	H	L	F	P	F	S	D	T	G	E	A	K	G	E	K	T	N	E
s	V	V	Q	A	V	T	H	L	F	P	F	S	D	T	G	E	A	T	G	A	K	T	K	E
k	V	V	Q	A	V	T	H	L	F	P	F	S	D	T	G	D	A	K	D	A	K	T	N	E
	I	V	Q	S	V	T	H	L	F	E	F	S	D	T	G	D	A	K	S	A	D	T	S	K
2	V	V	Q	A	V	T	N	L	I	E	F	S	D	T	G	E	A	K	A	A	D	T	S	K

Table 5-5. The de-clustering result of Table 5-4.

Records	Department code	Skill Code	Salary Code
R 0	P	FI	A
R 1	Q	SE	C
R 2	R	PL	B
R 3	P	SE	D
R 4	Q	SE	C
R 5	R	PL	B
R 6	P	AD	D
R 7	P	PL	B
R 8	R	FO	C
R 9	R	AD	D
R10	Q	FI	C
R11	R	FO	C
R12	R	PL	B
R13	P	SE	D
R14	P	AD	D
R15	P	FI	A
R16	R	PL	B
R17	R	PL	B
R18	Q	SE	B
R19	R	PL	C
R20	P	FI	A
R21	P	MA	C
R22	P	MA	C
R23	Q	SE	E
R24	R	PL	B
R25	P	PL	C
R26	P	MA	C
R27	Q	SE	C

Table 5-6. Data from Lee and Tseng.

T_{MST}	T_{SSP}	T_{RA}	T_{MST}/T_{RA} (%)	T_{SSP}/T_{RA} (%)
2.2444	2.2222	2.5185	89.10	88.24

T_{MST}: The performance of de-clustering based upon the minimal spanning tree.

T_{SSP}: The performance of de-clustering based upon the short spanning path.

T_{RA}: The expected performance of random allocation scheme.

Table 5-7. The results of Example 5-3.