〈論　文〉

The Modified Bottleneck Assignment Problem in Vector Case

— An Idea to Apply a Clustering Method —

Yuusaku Kamura

Abstract In this study, we deal with the bottleneck assignment problem in vector case. This problem is NP-complete. We show an idea that we use a clustering method to divide the original problem into sub problems. Each set of vertices is divided to subsets by a non-hierarchical clustering method. We make the optimal combination of the subsets, then vertices in the subset are corresponded according to the subsets' combinations. We show the effect of this idea by the numerical experiments.

Keywords approximation algorithm; assignment problem; combinatorial problem; clustering method

1. INTRODUCTION

The most well-known assignment problem is the linear sum assignment problem. It has the polynomial time algorithm and many efficient algorithms have been proposed [1]. The problem is formulated as follows:

A, B : n vertices' sets respectively.We denote the element in A is a_i, and also in B is b_j,
c_{ij} : edge's cost between a_i and b_j,
π : a permutation $\{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$,
Objective function is

$$f = \sum_{i=1}^{n} c_{i\pi(i)},$$

then find $\tilde{\pi}$ such that $\tilde{\pi}$ minimizes f.

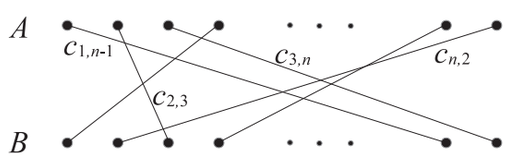

Fig. 1. Assignment problem. Each vertex $a_i (\in A)$ is assigned to distinct $b_j (\in B)$.

The assignment problem has various versions with respect to the objectives [2]. If we are going to minimize the maximum cost of the corresponded pair, the problem is called the bottleneck assignment problem [3]. The objective is

$$f = \max_{1 \le i \le n} c_{i\pi(i)} \,. \tag{1}$$

Also, polynomial time algorithms have been proposed for the bottleneck assignment problem.

In this paper we extend the bottleneck assignment problem to a case that costs are given by vectors. Here we assume that a cost vector $\boldsymbol{c}_{ij}$ is represented as a sum of the vectors $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$.

A formal description of our problem is as follows. Let A and B be sets of m dimensional vectors. We denote each element of A by $\boldsymbol{a}_i = (a_i^{(1)}, a_i^{(2)}, \ldots, a_i^{(m)})$ and each element of B by $\boldsymbol{b}_j = (b_j^{(1)}, b_j^{(2)}, \ldots, b_j^{(m)})$. We assume that $a_i^{(1)}, a_i^{(2)}, \ldots, a_i^{(m)}$ and $b_j^{(1)}, b_j^{(2)}, \ldots, b_j^{(m)}$ are nonnegative and define a sum of vectors $\boldsymbol{a}_i + \boldsymbol{b}_j$ as

$$\boldsymbol{a}_i + \boldsymbol{b}_j = (a_i^{(1)} + b_j^{(1)}, a_i^{(2)} + b_j^{(2)}, \ldots, a_i^{(m)} + b_j^{(m)}). \tag{2}$$

We denote this sum of vectors by $\boldsymbol{c}_{ij}$, namely $\boldsymbol{c}_{ij} = \boldsymbol{a}_i + \boldsymbol{b}_j$. Let us consider the following problem: find a permutation π of a set $\{1, 2, \ldots, n\}$ such that

$$T_m(\pi) = \max\{ \max_{1\leq i\leq n} c^{(1)}_{i\pi(i)}, \max_{1\leq i\leq n} c^{(2)}_{i\pi(i)}, \ \cdots, \max_{1\leq i\leq n} c^{(m)}_{i\pi(i)} \} \tag{3}$$

is the minimum.

More generally the problem is formulated as a multi-index bottleneck assignment problem. A multi-index bottleneck assignment problem is introduced in relation to the bus drivers' rostering problem and it is NP-complete [4]. That problem is multi-index scalar case one. Therefore for the vector case's bottleneck assignment problem, we deal with the problem in 2-index, that is, the combination between 2 sets of n vectors.

From the above assumptions, our problem is formulated as follows:

Problem 1　Given 2 sets of n vectors A and B, find the permuation π that minimizes

$$T_m(\pi) = \max_{1\leq k\leq m} \{\max_i c^{(k)}_{i\pi(i)}\}, \tag{4}$$

where $c^{(k)}_{i\pi(i)} = a^{(k)}_i + b^{(k)}_{\pi(i)}$.

In our former research [9] we dealt with the balanced assignment problem in 2 dimensional vector case. On the other hand, in [5], [6], [7], [8] we considered the bottleneck assignment problem in 2 dimensional vector case. In this paper we show the idea that we apply clustering methods to divide the problem into partial ones. We think it is more efficient than the partition method that I used in our previous studies [8],[9].

Remark　In general assignment problems, edges' costs are given independently. However, in our problem they are introduced by the vertices' costs. Hence we call this the 'modified' problem.

2. FORMULATION AS AN INTEGER PROGRAMMING PROBLEM

Problem 1 is formulated to the following 0-1 integer programming problem.

Problem 2

Minimize $f = t$

subject to

$$\begin{aligned}
&(a^{(k)}_i + b^{(k)}_j)x_{ij} \leq t, && (i, j = 1, 2, \ldots, n; k = 1, 2, \ldots, m);\\
&\sum_{i=1}^{n} x_{ij} = 1, && (j = 1, 2, \ldots, n);\\
&\sum_{j=1}^{n} x_{ij} = 1, && (i = 1, 2, \ldots, n);\\
&x_{ij} \in \{0, 1\}, && (i, j = 1, 2, \ldots, n).
\end{aligned}$$

Unlike the scalar case's assignment problem, no polynomial time algorithm exists to obtain the integer solution to this problem, and unfortunately, the relaxation method is not effective for this type of integer programming problems [10].

3. PROPERTY OF THE PROBLEM

Before describing our algorithm for Problem 1, let us consider the property of our problem.

At first, let us consider the case that $m = 1$, i.e., a_i and b_j are scalars. It is trivial that the total sum of $c_{i\pi(i)}$ does not depend on π, i.e.,

$$\sum_{i=1}^{n} c_{i\pi(i)} = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_{\pi(i)} = \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j = const. \tag{5}$$

Next, let us consider the case that $m = 2$. In this case again the total sum of $c^{(1)}_{i\pi(i)}$ and that of $c^{(2)}_{i\pi(i)}$ do not depend on π. We denote $\frac{1}{n}\sum_{i=1}^{n}(a_i^{(1)} + b_i^{(1)})$ by μ_1 and $\frac{1}{n}\sum_{i=1}^{n}(a_i^{(2)} + b_i^{(2)})$ by μ_2. For general m, the situation is unchanged.

Thus, we have the following Property 1.

Property 1 For each k, the total sum of $c^{(k)}_{i\pi(i)}$ does not depend on the selection of permutation π and is constant.

4. CLUSTERING

Clustering methods are being more and more important recently. It is used especially in the machine learning. Fig.2 shows the categorized diagram of clustering methods.

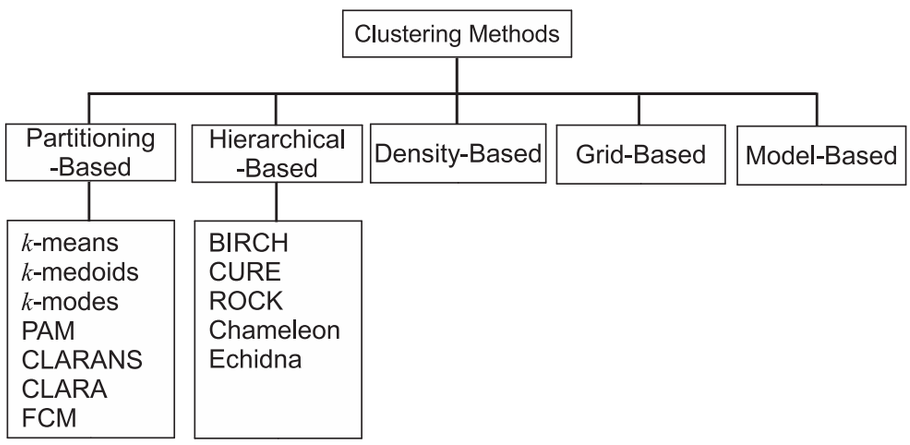

Fig. 2. Categorization of Clustering Methods [11] .

We use k-means method to divide the vertices sets A and B into k subsets. We denote the subsets by $A^{(p)}$ and $B^{(q)}$,respectively.

k-means method is the classical non-hierarchical clustering one. In unsupervised learning, k-means method is the most widely used. We find our problem conforms to the unsupervised learning. Because each data is not related to other qualities. We give only correlation coefficients to each two scalar's sequences of $a_i^{(k)}$ and $a_i^{(k+1)}$. Similarly, for $b_j^{(k)}$ and $b_j^{(k+1)}$. We have to notice that k-means does not divide the set equally to the subsets.

When we use k-means method, we give the integral value k which is the number of clusters. The optimal k-partitions is equivalent to the k-center problem. The k-center problem is NP-hard even for $k = 2$. So k-means method uses the greedy algorithm to find k clusters' each center point [12].

It means that k-means method gives an approximation k subsets' partition of A. Similarly, for B.

5. ALGORITHM FOR PROBLEM 1

Hereafter we consider the case that the dimension of error vector is 2. Firstly, we assume the following two properties.

Assumption 1

(i) Each vector's dimension is 2, that is, $m = 2$.

(ii) $\mu_1 = \mu_2$. We denote it μ.

(ii) is not essential but it is for treating the problem more easily.

Here we show the algorithm for Problem 1.

Algorithm 1

Step 1 :

1) Give clusters' number k. We assume $k \leq 10$.
2) Divide vertices set A to k subsets $A^{(1)}, A^{(2)}, \cdots, A^{(k)}$ by k-means method. We denote the center of $A^{(i)}$ as $cA^{(i)}$.
   Similarly for the set B and we get $B^{(1)}, B^{(2)}, \cdots, B^{(k)}$. Also $cB^{(j)}$ is the center of $B^{(j)}$.

Step 2 :

Make the optimal combination $\tilde{\pi}$ for $cA^{(l)}$ to $cB^{(\tilde{\pi}(l))}, (l = 1, 2, \ldots k)$.

Step 3 :

MaxVal: the maximum value of vectors' sums by the combination.

For each $l; (l = 1, 2, \ldots, n)$,

1) Sort $\boldsymbol{a}_{l,i} (\in A^{(l)})$ in ascending order by $||\boldsymbol{a}_{l,i} - cA^{(l)}||_{\max}$. Again we denote the sorted elements by $\boldsymbol{a}_{l,i}$ in i's order,
2) Sort $\boldsymbol{b}_{\tilde{\pi}(l),j} (\in B^{(\tilde{\pi}(l))})$ in descending order by $||\boldsymbol{b}_{\pi(l),j} - cB^{\tilde{\pi}(l)}||_{\max}$, Again we denote the sorted elements by $\boldsymbol{b}_{\pi(l),j}$ in j's order,
3) Define num$(A^{(l)})$ as the number of elements $A^{(l)}$, num$(B^{(\pi(l))})$ as the same for $B^{(\pi(l))}$.
   Set $MinNum \leftarrow \min\{\text{num}(A^{(l)}), \text{num}(B^{(\pi(l))})\}$,
4) For $i = 1, 2, \ldots, MinNum$, combine
$$\boldsymbol{c}_{l,i} \leftarrow \boldsymbol{a}_{l,i} + \boldsymbol{b}_{\tilde{\pi}(l),i},$$
5) If $\max\{c_{l,i}^{(1)}, c_{l,i}^{(2)}\} > MaxVal$ then $MaxVal \leftarrow \max\{c_{l,i}^{(1)}, c_{l,i}^{(2)}\}$.

Step 4 :

1) Collect remainders that are not combined in Step 3. We represent them as $A_{\mathrm{r}}, B_{\mathrm{r}}$ respectively. Here we notice that the number of elements in A_{r} and that of in B_{r} is equal.
2) Find the optimal combination of each element in A_{r} to that in B_{r}. For that purpose we solve the mixed integer programming problem(MIP) of A_{r} and B_{r}. It is obvious that solving this MIP does not require much time, because the number of remainders is much smaller than that of original A and B.

Step 5 :

Output the solution: *MaxVal* and the combination.

6. NUMERICAL EXPERIMENTS

We prepare the test data for the numerical experiments as follows : For vectors $\boldsymbol{a}_i = (a_i^{(1)}, a_i^{(2)})$ and $\boldsymbol{b}_j = (b_j^{(1)}, b_j^{(2)})$, $(i, j = 1, 2, \ldots, n)$, let $a_i^{(1)}, a_i^{(2)}$; $b_j^{(1)}, b_j^{(2)}$ follow the normal distribution that the mean is 10 and the variance 1, respectively. Here Cor_a is the correlation coefficient of $a_i^{(1)}$ and $a_i^{(2)}$, similarly Cor_b is that of $b_j^{(1)}$ and $b_j^{(2)}$. When the number of vectors n is not so large, that is, at most 100, $0 \leq Cor_a < 0.5$ means the $a_i^{(1)}$ and $a_i^{(2)}$ have little correlative relation. So we consider that it is sufficient to vary Cor_a to $0.0, 0.5, 0.7$, and 0.9. It is same for Cor_b and $(b_j^{(1)}, b_j^{(2)})$.

Each pair of Cor_a and Cor_b, we prepare several different data sets for $n = 40, 80, 120$. For each data set, we solve the problem by Algorithm 1, then compare to the exact solutions. Exact solutions are found by solving Problem 2, that is MIP, using the solver SCIP[1]. We denote results here for $n = 40, 80, 120$ in Table I,II,III, respectively.

In almost all cases Rel. Error are in $+5\%$, so we can say that the proposed algorithm, dividing the original bottleneck assignment problem to sub problems by the clustering method, is the effective idea.

TABLE I. $n = 40$

Cor_a,Cor_b	Solution		Rel.Error	Num. of combined data in clusters
	Exact	Approx.		
0.0, 0.0	22.039116	22.688025	0.0294	28
0.5	21.018795	21.778922	0.0362	28
0.7	20.900123	21.743491	0.0404	28
0.9	20.862537	21.461411	0.0287	29
0.5,0.5	20.996586	21.632746	0.0303	28
0.7	20.827939	21.538548	0.0341	30
0.9	20.826877	21.574567	0.0359	28
0.7,0.7	20.807930	21.3162220	0.0244	30
0.9	20.808607	21.4280423	0.0298	29
0.9,0.9	20.722833	21.430077	0.0341	30

Rel. Error shows the relative error of aprroximation solution, that is, $(\mathrm{Approx.} - \mathrm{Exact})/\mathrm{Exact}$.

We execute only one case for each n here. We should apply our algorithm to many data sets and get more results.

TABLE II. $n = 80$

Cor_a,Cor_b	Solution		Rel.Error	Num. of combined data in clusters
	Exact	Approx.		
0.0, 0.0	21.196701	21.766160	0.0269	70
0.5	20.697286	21.687800	0.0479	70
0.7	20.996616	21.873512	0.0418	69
0.9	21.171483	21.977872	0.0381	68
0.5,0.5	21.076558	22.112139	0.0491	65
0.7	20.846579	21.523213	0.0325	73
0.9	20.743654	21.523213	0.0376	66
0.7,0.7	20.753700	21.6102689	0.0413	71
0.9	20.677560	21.870456	0.0577	69
0.9,0.9	20.753692	21.289524	0.0258	73

TABLE III. $n = 120$

Cor_a,Cor_b	Solution		Rel.Error	Num. of combined data in clusters
	Exact	Approx.		
0.0, 0.0	21.216827	22.121165	0.0426	104
0.5	21.488096	22.484232	0.0464	92
0.7	21.420816	22.325154	0.0422	89
0.9	21.118222	22.525889	0.0667	92
0.5,0.5	21.118222	22.355431	0.0586	94
0.7	21.118222	22.300793	0.0560	95
0.9	21.118222	22.330413	0.0574	91
0.7,0.7	21.118222	22.044003	0.0438	89
0.9	20.987568	21.843143	0.0408	85
0.9,0.9	20.994517	21.538125	0.0259	91

7. CONCLUSIONS

In this paper, we considered a permutation that minimizes the maximum value in each sum of components given by 2 dimensional vectors. We first formulated this problem as an integer programming problem. We used the clustering method to divide the problem to sub problems and we show the approximation algorithm. We showed that idea is effect by the numerical experiments. Complete the numerical experiments of Algorithm 1 is left to further researches.

[1]SCIP (Solving Constraint Integer Programs) is a non-commercial MIP solver developed at Zuse Institute Berlin. http://scip.zib.de/

REFERENCES

[1] Du, D.-Z., Pardalos, P.M.(eds.): Handbook of Combinatorial Optimization. Kluwer (1999)

[2] Pentico, D.W.: Assignment problems : A golden anniversary survey. European J. Oper. Res. **176**, 774–793 (2007)

[3] Gabow, H.N., Tarjan, R.E.: Algorithms for Two Bottleneck Optimization Problems. J. of Algorithms. **9**, 411–417 (1988)

[4] Carraresi, P., Gallo,G. : A multi-level bottleneck assignment approach to the bus drivers' rostering problem. European J. Oper. Res. **16**, 163–173 (1984)

[5] Kamura, Y., Nakamori, M. : Combining Imperfect Components to Minimize the System's Error. Proc. PDPTA'01. 1277–1283 (2001)

[6] Kamura, Y., Nakamori, M., Shinano, Y. : Combining Imperfect Components (II) — The Case of Multidimensional Error. Proc. PDPTA'02. 228–232 (2002)

[7] Kamura, Y., Nakamori, M. : Combining Imperfect Components (III) — Minimax Optimization of Multidimensional Cost Error. Proc. PDPTA'04. 311–316 (2004)

[8] Kamura, Y., Nakamori, M. : A Simple Approximation Algorithm for the Modified Bottleneck Assignment Problem in Vector Case. International J. of Computer Theory and Engineering. **8**(2), 145–149 (2016)

[9] Kamura, Y., Nakamori, M. : Modified Balanced Assignment Problem in Vector Case: System Construction Problem. Proc. of The 2014 International Conf. on Comput. Sci. & Comput. Intelligence (CSCI ' 2014). **II**, 52–56, IEEE Comput. Soc. (2014)

[10] Mori, M., Matsui, T., : Operations Research (in Japanese). Asakura publishing (2004)

[11] Fahad, A, Alshatri, N., Tari, Z., Alamri A., Khalil, I., Zomaya, A.Y., Foufou, S., Bouras, A, : A Survey of Clustering Algorithms for Big Data: Taxonomy & Empirical Analysis. EMERGING TOPICS IN COMPUTING **2**, 267–279 (2014)

[12] Jain, Anil K. : Data clustering: 50 years beyond K-means. Pattern Recognition Letters. **31**,651-666, (2010)

[13] Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. SIAM J. Comput. **2**, 225–231 (1973)

[14] Burkard, R.E. : Selected topics on assignment problems. Discrete Appl. Math. **123**, 257–302 (2002)

Author

Yuusaku Kamura

Email: kamura.yuusaku@internet.ac.jp, kamura@u-gakugei.ac.jp

Joint Research Laboratory, Tokyo Online University,

1-7-3, Nishi-Shinjuku, Shinjuku, Tokyo 160-0023, Japan