不完全な部品の組合せ問題について (II) —
誤差がベクトルの場合

嘉村 友作　中森 眞理雄　品野 勇治

東京農工大学工学部 情報コミュニケーション学科
〒 184-8588 東京都小金井市中町 2-24-16

概要

複数の部品から構成されるシステムについて考える．部品は個体の集合であるロット単位で供給される．各個体には固有の誤差があり, 誤差値は多次元のベクトルとして与えられる．システムの誤差は, システムを構成する個体誤差の総和とする．システムの誤差ができる限り 0 に近い組合せを見つけることを目標とする．この問題は, 著者らの以前の論文の発展である．

本論文では, 問題を整数計画問題として記述し, それを分割して解く方法を提案し, 計算例を示す.

Combining Imperfect Components (II) —
The Case of Multidimensional Error

Yuusaku Kamura　Mario Nakamori　Yuji Shinano

Department of Computer Science
Tokyo A & T University
Koganei,Tokyo 184-8588,Japan

Abstract

Suppose a system composed of several components. We have a lot of individuals for each component. Every individual of the components has its own error which is expressed as a multidimensional vector. The total system's error of a combination is the sum of the individual error vectors. We have to find a combination such that the error vector to be as close to zero as possible. The problem is an extension of the previous report of the authors.

In the present paper we formulate the problem as an integer programming problem and solve by decomposing it into appropriate size. Computational results are also reported.

1 Introduction

As we mentioned in the former report [1], our problem is related to the problem of lens combination for a semiconductor exposing equipment. Each equipment includes about 30 lenses. Although lenses are made carefully, each individual has its own aberation, because the required standard of precision is extraordinarily high. Since lenses in such equipments are very expensive, it is disired to make the number of lenses rejected as small as possible. Therefore, we have sometimes to combine "good" lenses with "bad" ones in order not to reject too much.

Thus, we restate the problem as follows. The equipment consists of m components (namely, lenses) $A_1, A_2, \ldots, A_m$. For each component A_i ($i = 1, 2, \ldots, m$) we have l individuals. So we have l^m combinations in total.

The error (namely, aberation) of the indi-

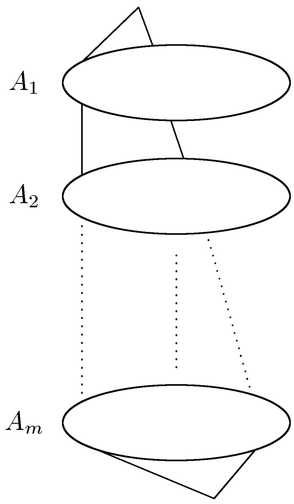

Figure 1: Lens system

vidual j for the component i is expressed as an n dimensional vector $(e_{ij}^1, e_{ij}^2, \ldots, e_{ij}^n)$. Let individuals $j_1, j_2, \ldots, j_m$ be chosen for components $A_1, A_2, \ldots, A_m$. The compound error of this combination $(j_1, j_2, \ldots, j_m)$ is the vector $(\sum_{i=1}^m e_{ij_i}^1, \sum_{i=1}^m e_{ij_i}^2, \ldots, \sum_{i=1}^m e_{ij_i}^n)$. Our aim is to find a combination or a set of combinations of individuals such that the compound error vector as close as zero.

In our previous report [1] we treated the error as scalar, i.e., we assumed that $n = 1$. In the present paper we reformulate the problem as an integer programming problem under the assumption that $n > 1$. We also show some computational results by an IP solver.

2 Formulation of the problem

In order to formulate the problem as an integer programming problem, we have to consider two important factors — the objective function and the constraints.

2.1 Objectives

Two typical cases of requirements can be considered, and the objective function depends on them.
Case 1: We have to use all individuals. In this case the objective is to minimize the worst error among combination. A slight modification gives the case that the objective to maximize the number of combinations that are within the bound of allowance.
Case 2: We have only to choose one combination of individuals. Suppose we have always l individuals as stocks for each component A_i. A new individual arrives for each component A_i every day. In such a case it is natural to select the best combination every day.

In the present paper we consider only the Case 2.

2.2 Constraints

Since we choose only one combination, we use integer variables x_{ij} $(i = 1, 2, \ldots, m; j = 1, 2, \ldots, l)$ such that $x_{ij} = 1$ if the individual j is chosen for the component A_i; $x_{ij} = 0$ otherwise.

As only one individual is chosen for each component A_i, we have

$$\sum_{j=1}^{l} x_{ij} = 1 \quad (i = 1, 2, \ldots, m). \tag{1}$$

The compound error $(u_1, u_2, \ldots, u_n)$ is expressed by

$$\sum_{j=1}^{l} \sum_{i=1}^{m} a_{ijk} x_{ij} = u_k \quad (k = 1, 2, \ldots, n) \tag{2}$$

where a_{ijk} is the error e_{ij}^k.

If we are to minimize the worst error, we have the constraint

$$|u_k| \leq z \quad (k = 1, 2, \ldots, n)$$

or

$$-z \leq u_k \leq z \quad (k = 1, 2, \ldots, n) \tag{3}$$

and the objective function is z.

Sometimes it is preferred that the total sum of $|u_k|$'s is the minimum. In such a case, we have the constraint

$$-v_k \leq u_k \leq v_k \quad (k = 1, 2, \ldots, n) \tag{4}$$

and the objective function is

$$s = \sum_{k=1}^{n} v_k \tag{5}$$

which is to be minimized.

3 Decomposition method

When the problem is of practical size, e.g., $m = 30$, $l = 30$, $n = 300$, the number of variables and that of constraints are $900, 930$, respectively. It is not practical to solve the problem directly.

As is well known, many kinds of algorithms have been proposed for combinatorial optimization problems. Among them "greedy" algorithm has been observed to give comparatively good solution, if we make consideration of the shortness of computing time. Thus, we decided to decompose the problem and solved step by step in greedy-like way. Our method is as follows.

Step 0 : Decompose the set $\{1, 2, \ldots, m\}$ into p subsets

$\{1, 2, \ldots, i_1\}$,
$\{i_1+1, i_1+2, \ldots, i_2\}$,
$\{i_2+1, i_2+2, \ldots, i_3\}$,
$\ldots$,
$\{i_{p-1}+1, i_{p-1}+2, \ldots, i_p\}$,

where we assume that $i_1 < i_2 < \cdots < i_p = m$.

Step 1 : Solve the problem by restricting the range of i of the constraint (1) to be $1, 2, \ldots, i_1$, i.e.,

Minimize z
subject to

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad (i = 1, \ldots, i_1)$$

$$x_{ij} = 0 \quad \text{or} \quad 1 \qquad (i = 1, \ldots, i_1; j = 1, \ldots, m)$$

$$\sum_{j=1}^{m} \sum_{i=1}^{i_1} a_{ijk} x_{ij} = u_k \qquad (k = 1, \ldots, n)$$

$$-z \le u_k \le z \qquad (k = 1, \ldots, n)$$

Let us denote the value of u_k of the optimal solution by $\hat{u}_k^1 (k = 1, \ldots, n)$. Also, let us denote the value of x_{ij} of the optimal solution by $\hat{x}_{ij} (i = 1, \ldots, i_1; j = 1, \ldots, m)$.

Step 2 : Solve the problem by restricting the range of i to be $i_1+1, i_1+2, \ldots, i_2$, i.e.,

Minimize z
subject to

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad (i = i_1+1, \ldots, i_2)$$

$$x_{ij} = 0 \quad \text{or} \quad 1 \qquad (i = i_1+1, \ldots, i_2; j = 1, \ldots, m)$$

$$\sum_{j=1}^{m} \sum_{i=i_1+1}^{i_2} a_{ijk} x_{ij} + \hat{u}_k^1 = u_k \qquad (k = 1, \ldots, n)$$

$$-z \le u_k \le z \qquad (k = 1, \ldots, n)$$

Let us denote the value of u_k of the optimal solution by $\hat{u}_k^2 (k = 1, \ldots, n)$. Also, let us denote the value of x_{ij} of the optimal solution by $\hat{x}_{ij} (i = i_1+1, \ldots, i_2; j = 1, \ldots, m)$.

$\vdots$

Step p : Solve the problem by restricting the range of i to be $i_{p-1}+1, i_{p-1}+2, \ldots, i_p$, i.e.,

Minimize z
subject to

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad (i = i_{p-1}+1, \ldots, i_p)$$

$$x_{ij} = 0 \quad \text{or} \quad 1 \qquad (i = i_{p-1}+1, \ldots, i_p; j = 1, \ldots, m)$$

$$\sum_{j=1}^{m} \sum_{i=i_{p-1}+1}^{i_p} a_{ijk} x_{ij} + \hat{u}_k^{p-1} = u_k \qquad (k = 1, \ldots, n)$$

$$-z \le u_k \le z \qquad (k = 1, \ldots, n)$$

Let us denote the value of x_{ij} of the optimal solution by $\hat{x}_{ij} (i = i_{p-1}+1, \ldots, i_p; j = 1, \ldots, m)$.

Thus, we have the final solution $\hat{x}_{ij}$ $(i = i_1, \ldots, i_p; j = 1, \ldots, m)$.

4 Computational examples

We applied the above decomposition method to examples of practical data in industry. The computation was executed with branch and bound method of NUOPT Ver.4.0.1, a solver of mathematical programming and modelling by Mathematical Systems Institute, Inc.[1] Computational environment is as follows:

CPU	Intel PentiumIII 600MHz
Memory	384MB
OS	MS-Windows 98SE

Example 1.1

This is the case where $m = 28$, $l = 36$, $n = 329$, and the number of feasible solutions is $36^{28} \sim 3.8 \times 10^{43}$. We decomposed the set of components $\{A_1, A_2, \ldots, A_{28}\}$ into 7 blocks (subsets) such as

Block 1 : A_1, A_2, A_3, A_4;
Block 2 : A_5, A_6, A_7, A_8;
Block 3 : $A_9, A_{10}, A_{11}, A_{12}$;
Block 4 : $A_{13}, A_{14}, A_{15}, A_{16}$;
Block 5 : $A_{17}, A_{18}, A_{19}, A_{20}$;
Block 6 : $A_{21}, A_{22}, A_{23}, A_{24}$;
Block 7 : $A_{25}, A_{26}, A_{27}, A_{28}$.

[1] http://www.msi.co.jp/nuopt

Table 1: Result of Example 1.1

block	int. var.	time(s)	obj.
1	144	89.42	0.016465979
2	144	167.75	0.024760760
3	144	551.12	0.039422580
4	144	464.50	0.046844088
5	144	193.72	0.067523147
6	144	206.13	0.079441640
7	144	42.62	0.083060517

int. var. : Number of integer variables.
obj. : Value of the objective function.

Table 2: Result of Example 2

block	int. var.	time(s)	obj.
1	432	2981.41	0.015322444
2	432	395.13	0.023793190
3	432	20982.80	0.026066562
4	432	18191.22	0.035772508
5	432	1466.73	0.058210355
6	432	6840.42	0.060574506
7	432	1213.63	0.060115495

int. var : Number of integer variables.
obj. : Value of the objective function.

The computational result is shown in Table 1. The number of variables is 1008, and the computational time was 1715.26 seconds. The value of the objective function of the optimal solution (the compound error of the best combination) was 0.083060517.

Example 1.2
This is the same problem. We decomposed the set of components into 5 blocks, such that

Block 1 : $A_1,\ A_2,\ A_3,\ A_4,\ A_5,\ A_6$;
Block 2 : $A_7,\ A_8,\ A_9,\ A_{10}, A_{11}, A_{12}$;
Block 3 : $A_{13}, A_{14}, A_{15}, A_{16}, A_{17}, A_{18}$;
Block 4 : $A_{19}, A_{20}, A_{21}, A_{22}, A_{23}, A_{24}$;
Block 5 : $A_{25}, A_{26}, A_{27}, A_{28}$;

The number of the variables is 1008, and the computational time was 18398.49 seconds. The value of the objective function of the optimal solution (the compound error of the best combination) was 0.070845403. The error by a well trained engineer is 0.1165892.

Example 2
This is the case where $m = 28, l = 108, n = 329$, and the number of feasible solutions is $108^{28} \sim 8.6 \times 10^{56}$. We decomposed the set of components $\{A_1, A_2, \ldots, A_{28}\}$ into 7 blocks (subsets) such as

Block 1 : $A_1,\ A_2,\ A_3,\ A_4$;
Block 2 : $A_5,\ A_6,\ A_7,\ A_8$;
Block 3 : $A_9,\ A_{10}, A_{11}, A_{12}$;
Block 4 : $A_{13}, A_{14}, A_{15}, A_{16}$;
Block 5 : $A_{17}, A_{18}, A_{19}, A_{20}$;
Block 6 : $A_{21}, A_{22}, A_{23}, A_{24}$;
Block 7 : $A_{25}, A_{26}, A_{27}, A_{28}$.

The computational result is shown in Table 3. The number of the variables is 3024, and the computational time was 52084.21 seconds. The value of the objective function of the optimal solution (the compound error of the best combination) was 0.060115495.

5 Conclusions

An extension of the authors' previous research has been shown on combining imperfect individuals of components to compose a complex system. The problem has been formulated as an integer programming problem and has been solved by branch and bound method.

In practice it is desired to solve the problem in a few minutes in industry. Thus, we have to develop a practical algorithm that gives a suboptimal (quasi optimal) solution and to compare with the exact solution by integer programming techniques, which is left for further research.

References

[1] Y.Kamura and M.Nakamori. *Combining Imperfect Components to Minimize the System's error*, Proc. of the International Conf. on PDPTA'2001, pp.1277-1283, 2001.

[2] K.Fukuda and M.Nakamori. *Accelarating graph isomorphism algorithm by finer classification of layered graphs*, Proc. of the International Conf. on PDPTA'2001, pp.1271-1276, 2001.

[3] G.Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications* Springer-Verlag, 1994.

[4] M.Gen and R.Cheng. *Genetic Algorithms & Engineering Design,* John Wiley & Sons, 1997.

[5] R.L.Haupt and S.E.Haupt. *Practical Genetic Algorithms,* John Wiley & Sons, 1998.