

Chapter 6

Other types of linear assignment problems

6.1 Introduction

In the two previous chapters we discussed in detail linear assignment problems with a sum objective function of the form

$$\min_{\varphi} \sum_{i=1}^n c_{i\varphi(i)}.$$

In various situations, however, it is meaningful to replace the sum objective by a so-called bottleneck objective function. This leads to a *linear bottleneck assignment problem* (LBAP) of the form

$$\min_{\varphi} \max_{1 \leq i \leq n} c_{i\varphi(i)}.$$

We have seen in the introduction that such LBAPs occur in connection with assigning jobs to parallel working machines. The goal is to assign the jobs such that the latest completion time is minimized. We discuss linear bottleneck assignment problems in detail in Section 6.2. In particular, we develop threshold algorithms, a dual method, and a shortest augmenting path method for solving the LBAP. A practically efficient method will be obtained by thinning out the underlying bipartite graph and exploiting the sparsity of the modified problem; see Section 6.2.5. Moreover, we discuss special cases of the LBAP which can be solved in a fast way. Finally, we describe the asymptotic behavior of LBAPs in Section 6.2.7.

Sum and bottleneck assignment problems can be viewed as special cases of a more general model, the so-called *algebraic assignment problem* which is discussed in Section 6.3. In Sections 6.4 and 6.5 we deal with assignment problems where we require to minimize the sum of the k largest cost coefficients in the assignment, or the difference between the largest and the smallest cost coefficient in the assignment. In Section 6.6 we deal with a modified objective: we order the n cost coefficients of an assignment decreasingly and ask for a solution which is lexicographically minimal.

6.2 Bottleneck assignment problem

6.2.1 Background

Linear bottleneck assignment problems were introduced by Fulkerson, Glicksberg, and Gross [292] and occur, e.g., in connection with assigning jobs to parallel machines so as to minimize the latest completion time. Let n jobs and n machines be given. The cost coefficient c_{ij} is the time needed for machine j to complete job i . If the machines work in parallel and we want to assign the jobs to the machines such that the latest completion time is as early as possible, we get an LBAP of the form

$$\min_{\varphi \in \mathcal{S}_n} \max_{1 \leq i \leq n} c_{i\varphi(i)}. \quad (6.1)$$

If we describe permutations by the corresponding permutation matrices $X = (x_{ij})$, an LBAP can be modeled as

$$\begin{aligned} \min \quad & \max_{1 \leq i, j \leq n} c_{ij} x_{ij} & (6.2) \\ \text{s.t.} \quad & \sum_{j=1}^n x_{ij} = 1 & (i = 1, 2, \dots, n), \\ & \sum_{i=1}^n x_{ij} = 1 & (j = 1, 2, \dots, n), \\ & x_{ij} \in \{0, 1\} & (i, j = 1, 2, \dots, n). \end{aligned}$$

The following lemma comprises two elementary observations concerning LBAPs which are helpful in solving problems in practice.

Lemma 6.1. *Let $C = (c_{ij})$ be the cost matrix of an LBAP. Then the following two statements hold:*

1. *The optimum value of the LBAP is taken by one of the cost coefficients c_{ij} .*
2. *The optimal solution φ^* depends only on the relative order of the cost coefficients and not on their numerical value.*

Example 6.2. Consider an LBAP with cost matrix

$$C = \begin{pmatrix} \pi & \sqrt{3} & 54392 \\ 0 & e^3 & e^3 \\ -2 & \sin \pi/8 & 4 \end{pmatrix}.$$

By ordering the cost coefficients we get

$$-2 < 0 < \sin \pi/8 < \sqrt{3} < \pi < 4 < e^3 = e^3 < 54392.$$

The smallest element is -2 which can be modeled by 0. The second smallest element is 0 which is replaced by 1. Then $\sin \pi/8$ is replaced by 2, and so on. Therefore, we can replace this cost matrix by

$$C = \begin{pmatrix} 4 & 3 & 7 \\ 1 & 6 & 6 \\ 0 & 2 & 5 \end{pmatrix}. \quad (6.3)$$

Thus, if the cost matrix of an LBAP has d different entries, we may model these entries by $0, 1, 2, \dots, d - 1$. The LBAP with cost matrix (6.3) has the optimal solution

$$\varphi^* = (2, 1, 3).$$

The largest value in this solution is $c_{33} = 5$, which corresponds to 4 in the originally given cost matrix. ■

If one is interested in solving the “opposite” LBAP, i.e., in finding an assignment in which (6.2) is replaced by

$$\max \min_{1 \leq i, j \leq n} c_{ij} x_{ij},$$

according to the second statement of Lemma 6.1, it is enough to model the elements in the opposite way, i.e., with the largest element modeled by 0, the second largest by 1, and so on.

Considering bottleneck assignment problems, Gross [342] proved the following min-max theorem, which was a starting point of the theory on blocking systems; see Edmonds and Fulkerson [249].

Theorem 6.3. (Gross [342], 1959.) *Let $N = \{1, 2, \dots, n\}$ and let S_n be the set of all permutations φ on N . Then the following min-max equality holds for an arbitrary $n \times n$ matrix $C = (c_{ij})$ with elements c_{ij} drawn from a totally ordered set:*

$$\min_{\varphi \in S_n} \max_{i \in N} c_{i\varphi(i)} = \max_{\substack{I, J \subseteq N \\ |I| + |J| = n + 1}} \min_{i \in I, j \in J} c_{ij}. \quad (6.4)$$

Note that the Frobenius theorem, Theorem 2.4, can be seen as a special case of Theorem 6.3, arising when C is a 0–1 matrix.

Proof. Let

$$c^* = \min_{\varphi \in S_n} \max_{i \in N} c_{i\varphi(i)}.$$

We define two 0–1 matrices \bar{C} and $\overline{\bar{C}}$ by

$$\bar{c}_{ij} = \begin{cases} 1 & \text{if } c_{ij} \leq c^*, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad \overline{\bar{c}}_{ij} = \begin{cases} 1 & \text{if } c_{ij} < c^*, \\ 0 & \text{otherwise.} \end{cases}$$

According to the construction, matrix \bar{C} contains a permutation matrix and matrix $\overline{\bar{C}}$ does not contain any permutation matrix. By applying Theorem 2.4 to matrix \bar{C} we get that any $(k + 1) \times (n - k)$ submatrix of C contains an entry $\leq c^*$. By applying Theorem 2.4 to matrix $\overline{\bar{C}}$ we get that there is a $(k + 1) \times (n - k)$ submatrix of C which contains only entries $\geq c^*$. Thus we have shown

$$c^* = \max_{\substack{I, J \subseteq N \\ |I| + |J| = n + 1}} \min_{i \in I, j \in J} c_{ij}. \quad \square$$