# Chapter 4
# Linear sum assignment problem

## 4.1 Introduction

The *linear sum assignment problem* (LSAP) is one of the most famous problems in linear programming and in combinatorial optimization. Informally speaking, we are given an $n \times n$ *cost matrix* $C = (c_{ij})$ and we want to match each row to a different column in such a way that the sum of the corresponding entries is minimized. In other words, we want to select $n$ elements of $C$ so that there is exactly one element in each row and one in each column and the sum of the corresponding costs is a minimum.

Alternatively, one can define LSAP through a graph theory model. Define a bipartite graph $G = (U, V; E)$ having a vertex of $U$ for each row, a vertex of $V$ for each column, and cost $c_{ij}$ associated with edge $[i, j]$ $(i, j = 1, 2, \ldots, n)$: The problem is then to determine a minimum cost perfect matching in $G$ (*weighted bipartite matching problem*: find a subset of edges such that each vertex belongs to exactly one edge and the sum of the costs of these edges is a minimum).

Without loss of generality, we assume that the costs $c_{ij}$ are nonnegative. Cases with negative costs can be handled by adding to each element of $C$ the value $\chi = -\min_{i,j}\{c_{ij}\}$: Since we need to select one element per row, any solution of value $z$ for the original cost matrix corresponds to a solution of value $z + n\chi$ for the transformed cost matrix. In this way we can manage the maximization version of the problem by solving LSAP on a transformed instance having costs $\widetilde{c}_{ij} = -c_{ij}$. Let us also observe that most preprocessing algorithms (see, e.g., Algorithm 4.1), which are preliminary executed on LSAP instances in order to accelerate the solution algorithms, produce a nonnegative cost matrix.

We also assume in general that the values in $C$ are finite, with some $c_{ij}$ possibly having a very large value ($< \infty$) when assigning $i$ to $j$ is forbidden. In the case of sparse matrices, where a very large number of $(i, j)$ assignments is forbidden, we denote by $m$ the number of admitted $(i, j)$ assignments and, in graph $G = (U, V; E)$ above, we only include in $E$ the edges corresponding to them, so $|E| = m$.

Most of the algorithms we present work for general cost matrices, although special approaches, such as those based on cost-scaling (see Section 4.2.3), require an integer cost matrix. For such cases we denote by $\mathcal{C}$ the maximum $c_{ij}$ value.

### 4.1.1 Mathematical model

By introducing a binary matrix $X = (x_{ij})$ such that

$$x_{ij} = \begin{cases} 1 & \text{if row } i \text{ is assigned to column } j, \\ 0 & \text{otherwise,} \end{cases}$$

LSAP can be modeled as

$$\min \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, 2, \ldots, n), \tag{4.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, 2, \ldots, n), \tag{4.3}$$

$$x_{ij} \in \{0, 1\} \qquad (i, j = 1, 2, \ldots, n). \tag{4.4}$$

Note that the resulting matrix $X$ is a permutation matrix (see Section 1.1).

The $2n \times n^2$ matrix defined by the left-hand sides of the assignment constraints (4.2) and (4.3) is the incidence matrix of the bipartite graph $G$ introduced above. It has a row $i$ for each vertex $i \in U$, a row $n + j$ for each vertex $j \in V$, and a column for each edge $[i, j] \in E$: The entries in this column are 1 in rows $i$ and $n + j$ and 0 elsewhere. This matrix, shown in Figure 4.1, has the nice combinatorial property of being totally unimodular.

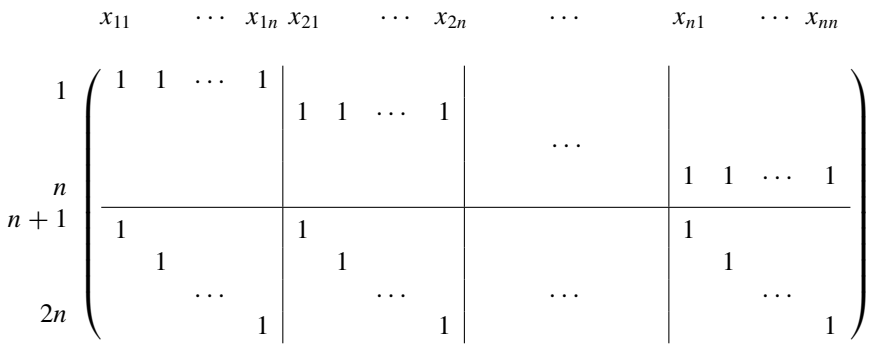


**Figure 4.1.** *Node-edge incidence matrix of G.*

**Definition 4.1.** *An integer matrix is* totally unimodular *if the determinant of every square submatrix has value 0,* $+1$*, or* $-1$*.*

We will use the following sufficient condition due to Heller and Tompkins [368].

**Theorem 4.2.** *An integer matrix A with* $a_{ij} = 0, \pm 1$ *is totally unimodular if no more than two non-zero entries appear in any column and if its rows can be partitioned into two sets,* $I_1$ *and* $I_2$*, such that*

1. *if a column has two entries of the same sign, their rows are in different sets;*
2. *if a column has two entries of different signs, their rows are in the same set.*

It is then easy to see that the matrix defined by (4.2) and (4.3) satisfies the condition above with $I_1 = \{1, 2, \ldots, n\}$ and $I_2 = \{n+1, n+2, \ldots, 2n\}$. Since it is known that a linear program with integer right-hand sides and totally unimodular constraint matrix always has an integer optimal solution, this shows that LSAP is equivalent to its continuous relaxation, given by (4.1)–(4.3) and

$$x_{ij} \geq 0 \qquad (i, j = 1, 2, \ldots, n). \tag{4.5}$$

The same result was proved, in a different way, by Birkhoff [100] (see Theorem 2.18 and Proposition 2.24).

### 4.1.2 Complementary slackness

By associating dual variables $u_i$ and $v_j$ with assignment constraints (4.2) and (4.3), respectively, the dual problem is

$$\max \quad \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \tag{4.6}$$

$$\text{s.t.} \quad u_i + v_j \leq c_{ij} \qquad (i, j = 1, 2, \ldots, n). \tag{4.7}$$

By duality theory, a pair of solutions respectively feasible for the primal and the dual is optimal if and only if (*complementary slackness*)

$$x_{ij}(c_{ij} - u_i - v_j) = 0 \qquad (i, j = 1, 2, \ldots, n). \tag{4.8}$$

The values

$$\overline{c}_{ij} = c_{ij} - u_i - v_j \qquad (i, j = 1, 2, \ldots, n) \tag{4.9}$$

are the linear programming *reduced costs*. This transformation from $C$ to $\overline{C}$ is a special case of what is known as "admissible transformation," which is formally stated in Chapter 6, Definition 6.19. Indeed, for any feasible primal solution $X$, the transformed objective function is

$$\begin{aligned} \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij} - u_i - v_j)x_{ij} &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{n} u_i \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} v_j \sum_{i=1}^{n} x_{ij} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{n} u_i - \sum_{j=1}^{n} v_j, \end{aligned} \tag{4.10}$$

i.e., for any feasible solution $X$ the objective function values induced by $C$ and $\overline{C}$ differ by the same constant $\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$.

The algorithms for LSAP are based on different approaches: a first class of methods directly solves the primal problem, a second one solves the dual, and a third one uses an intermediate approach (primal-dual). Most of these methods adopt a *preprocessing phase* to determine a feasible dual solution and a partial primal solution (where less than $n$ rows are assigned) satisfying the complementary slackness conditions. A basic $O(n^2)$ time implementation of this phase is given in Algorithm 4.1, which stores the partial assignment both in $X$ and in

$$row(j) = \begin{cases} i & \text{if column } j \text{ is assigned to row } i, \\ 0 & \text{if column } j \text{ is not assigned,} \end{cases} \qquad (j = 1, 2, \ldots, n). \tag{4.11}$$

Note that the reduced costs given by the resulting dual variables are nonnegative.

**ALGORITHM 4.1. Procedure Basic_preprocessing.**
Feasible dual solution and partial primal solution.

**for** $i := 1$ **to** $n$ **do for** $j := 1$ **to** $n$ **do** $x_{ij} := 0$;
**comment:** row and column reduction;
**for** $i := 1$ **to** $n$ **do** $u_i := \min\{c_{ij} : j = 1, 2, \ldots, n\}$;
**for** $j := 1$ **to** $n$ **do** $v_j := \min\{c_{ij} - u_i : i = 1, 2, \ldots, n\}$;
**comment:** find a partial feasible solution;
**for** $j := 1$ **to** $n$ **do** $row(j) := 0$;
**for** $i := 1$ **to** $n$ **do**
    **for** $j := 1$ **to** $n$ **do**
        **if** ($row(j) = 0$ **and** $c_{ij} - u_i - v_j = 0$) **then**
            $x_{ij} := 1$, $row(j) := i$;
            **break**
        **endif**
    **endfor**
**endfor**

In the following we will frequently store a (partial) assignment in $\varphi$ that implements the inverse of *row* (see Section 1.1), i.e.,

$$\varphi(i) = \begin{cases} j & \text{if row } i \text{ is assigned to column } j, \\ 0 & \text{if row } i \text{ is not assigned,} \end{cases} \qquad (i = 1, 2, \ldots, n). \tag{4.12}$$

The $u_i$ and $v_j$ values determined by the first two statements satisfy the dual constraints (4.7). The $x_{ij}$ values subsequently obtained ensure satisfaction of complementary slackness conditions (4.8), while for the primal constraints (4.1) and (4.2) the $\leq$ sign holds instead of $=$. Note that an alternative algorithm could first perform the column reduction and then the row reduction, generally obtaining different reduced costs and assignments. We illustrate Basic_preprocessing through a numerical example, to be resumed several times in the following.

**Example 4.3.** Given the input matrix $C$ below, we obtain the dual variables $u$ and $v$ (shown on the left and on the top) and the corresponding reduced cost matrix $\overline{C}$:

$$\begin{array}{c} \\ 7 \\ 2 \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{cccc} 0 & 2 & 0 & 0 \end{array} \\ \begin{pmatrix} 7 & 9 & 8 & 9 \\ 2 & 8 & 5 & 7 \\ 1 & 6 & 6 & 9 \\ 3 & 6 & 2 & 2 \end{pmatrix} \end{array} \qquad \begin{pmatrix} \underline{0} & 0 & 1 & 2 \\ 0 & 4 & 3 & 5 \\ 0 & 3 & 5 & 8 \\ 1 & 2 & \underline{0} & 0 \end{pmatrix}.$$

$C$ $\overline{C}$

We then obtain $row = (1,\ 0,\ 4,\ \underline{0})$ (thus $\varphi = (1,\ 0,\ 0,\ 3)$) and the partial assignment shown by the underlined zeroes in $\overline{C}$. ∎

### 4.1.3 Historical notes, books, and surveys

The first algorithm for LSAP was presented in 1946 by Easterfield [245]: it is a non-polynomial $O(2^n n^2)$ time approach, based on iterated application of a particular class of admissible transformations (see Section 6.3). Easterfield did not give a name to the problem (the title of the paper is "A combinatorial algorithm"), nor did it Thorndike [638] who, in a 1950 paper ("The problem of classification of personnel"), proposed three heuristic algorithms for LSAP. The current name appeared for the first time in 1952, in the paper "The personnel assignment problem" by Votaw and Orden [653]. Further historical details on the early years of LSAP and the Hungarian algorithm can be found in Schrijver [599, Chapter 17], Frank [277], and Schrijver [600].

*Primal-dual algorithms* have been the first polynomial-time methods for LSAP. The famous Hungarian algorithm, presented in the mid-1950s by Kuhn [438, 440], in its original formulation solves the problem in $O(n^4)$ time. A variation of this algorithm was presented by Munkres [502]: he showed that his method requires at most $(11n^3 + 12n^2 + 31n)/6$ operations, where, however, some operations are "scan a line," thus leaving an $O(n^4)$ overall time complexity. Other $O(n^4)$ time algorithms were later proposed by Iri [386] and Desler and Hakimi [232]. In 1960 Silver [609, 610] gave the first computer code for LSAP, a modified version of the Munkres algorithm, written in Algol. The best time complexity for a Hungarian algorithm is $O(n^3)$ (see the implementation proposed by Lawler [448] in 1976). The first $O(n^3)$ algorithm for LSAP had, however, appeared in a 1969 paper by Dinic and Kronrod [235] that was ignored for many years. In the early 1970s Tomizawa [640] and Edmonds and Karp [250] showed that shortest path computations on the reduced costs produce an $O(n^3)$ time algorithm for LSAP. Computer implementations of the Hungarian algorithm adopting shortest path techniques (e.g., the codes proposed in the 1980s by Burkard and Derigs [145], Jonker and Volgenant [392], and Carpaneto, Martello, and Toth [165]) have for many years been the most successful tools for the practical solution of LSAP instances. In the mid-1980s Gabow [295] used the *cost scaling* technique, developed in the early 1970s by Edmonds and Karp [250], for obtaining a cost-scaling Hungarian algorithm for LSAP. The weakly polynomial time complexity of the resulting algorithm, $O(n^{3/4} m \log C)$, was later improved by Gabow and Tarjan [297] to $O(\sqrt{n}\ m \log(nC))$. Detailed descriptions of primal-dual methods are given in Sections 4.2 and 4.4.

The first *primal simplex algorithms*, proposed in the mid-1970s by Cunningham [205] and Barr, Glover, and Klingman [68], required exponential time due to the high degeneracy of the linear program associated with LSAP. A few years later Roohy-Laleh [589] modified

the Cunningham algorithm to obtain an $O(n^5)$ time complexity, while in 1993 Akgül [19] proposed an $O(n^3)$ time primal simplex algorithm. Primal methods are described in Section 4.5. The first *primal* (non-simplex) *algorithm* was proposed in 1964 by Balinski and Gomory [64]. It iteratively improves, through alternating paths, a feasible assignment and a dual (infeasible) solution satisfying complementary slackness and solves the problem in $O(n^4)$ time. Other primal algorithms were given in the following years by Srinivasan and Thompson [619, 620] and Klein [421]. The latter paper introduced the "cycle canceling" technique, which was very important for the solution of min-cost flow problems. (Such a technique had however been studied for the first time by Robinson [587] in 1949.) An $O(n^3)$ primal algorithm was obtained in 1978 by Cunningham and Marsh [208] by generalizing the Klein idea.

The first *dual* (non-simplex) *algorithm* for LSAP appeared in the already mentioned 1969 paper by Dinic and Kronrod [235], discussed in Section 4.3. This method is also the basis of the dual algorithm presented in 1980 by Hung and Rom [382], in which a series of relaxed problems (where constraints (4.3) are disregarded) is solved by updating the current solution through shortest paths until the solution becomes feasible for LSAP. This algorithm too has time complexity $O(n^3)$. In 1981 Bertsekas [86] proposed a dual algorithm having pseudo-polynomial time complexity but high average efficiency in practice (the *auction algorithm*). Polynomial-time auction algorithms were later obtained by Bertsekas and Eckstein [92] and by Orlin and Ahuja [515], who combined auction and a particular scaling technique (known as *ε-relaxation*). The time complexity of the latter algorithm is $O(\sqrt{n}\ m \log(n\mathcal{C}))$, equal to that of the primal-dual scaling algorithm by Gabow and Tarjan [297]. The same time complexity characterizes the computationally very effective algorithms developed in the mid-1990s by Goldberg and Kennedy [328], who adopted a scaling technique (*pseudoflow*) originally developed for min-cost flow problems. The most famous *dual simplex algorithms* for LSAP are the so-called *signature* methods, proposed in the mid-1980s by Balinski [62] and Goldfarb [334]. These algorithms have $O(n^3)$ time complexity, and it can be shown that they are substantially equivalent to the dual (non-simplex) approach by Hung and Rom [382]. Dual methods are discussed in Section 4.6.

The latest relevant theoretical result for LSAP was obtained in the new millennium by Kao, Lam, Sung, and Ting [403] who closed a long standing gap between the time complexity of LSAP and that of the maximum cardinality matching problem. Their result is discussed in Section 4.7.

The 1980s saw the diffusion of parallel computers. In the following years many sequential methods for LSAP (especially auction, shortest path, and primal simplex algorithms) have been parallelized and computationally tested on parallel machines. We describe these topics in Section 4.11.

Starting in the late 1970s, many books and surveys on LSAP have been proposed in the literature. The first survey was presented by Burkard [128], who included a summary of results on the structure of the associated polytope. The book by Burkard and Derigs [145] considers various assignment-type problems and includes, among others, a Fortran program implementing a variant of the shortest augmenting path algorithm proposed by Tomizawa [640] (see Sections 4.4.1 and 4.9). Derigs [226] presented an extensive survey on the shortest augmenting path technique (see Section 4.4), discussing and relating to it all classical algorithms and examining the results of an extensive computational experience, performed over 14 Fortran codes. Martello and Toth [481] reviewed LSAP and other linear assignment-type problems and analyzed the performance of different algorithms through computational

experiments. The book by Bertsekas [88] on relaxation and auction techniques (see Section 4.6.3) includes several implementations of algorithms for LSAP and the corresponding Fortran listings (also downloadable from the internet). The survey by Akgül [18] analyzes the main solution approaches and discusses their relationships. The volume edited by Johnson and McGeoch [390] includes several papers on implementations of algorithms for LSAP proposed at the first *DIMACS Implementation Challenge*. A specialized survey on the probabilistic analysis of simple online and offline heuristic algorithms for LSAP can be found in Faigle [264].

In 1997 Dell'Amico and Martello [219] presented an annotated bibliography, with special attention to results obtained in the 1980s and the 1990s. The extensive survey by Burkard and Çela [138] gives the state-of-the-art on LSAP and other linear assignment problems with other objective functions like the algebraic, bottleneck, balanced, axial, and planar assignment problems (see Chapters 6 and 10). Dell'Amico and Toth [220] presented extensive computational experiments with the eight most popular computer codes for dense instances of LSAP. Burkard [132] surveyed recent developments in the fields of bipartite matchings (see Chapter 3), LSAP, and quadratic assignment problems (see Chapter 7). Linear and non-linear assignment problems are discussed in a recent survey by Pentico [541].

Chapters on LSAP can also be found, e.g., in Murty [507, Chapter 3], Ahuja, Magnanti, and Orlin [11, Chapter 12], Jungnickel [399, Chapter 13], Korte and Vygen [428, Chapter 11], and Schrijver [599, Chapters 17 and 18].

## 4.2 Primal-dual algorithms

The genesis of these methods was reported by the author, H. W. Kuhn [441]. In 1953, while reading the classical graph theory book by König [426], *Theorie der Endlichen und Unendlichen Graphen*, he encountered an efficient algorithm for determining the maximum cardinality matching of a bipartite graph. This problem is equivalent to an LSAP with 0-1 costs, and a footnote in König's book pointed to a paper by Egerváry [253] (in Hungarian) for a generalization of the result to general cost matrices. Kuhn then spent two weeks translating Egerváry's paper, with the aid of a Hungarian dictionary, finding a method to reduce a general LSAP to a finite number of LSAPs with 0-1 costs. (The translated paper can be found in Kuhn [439].) The combination of Egerváry's reduction and König's maximum cardinality matching algorithm produced an efficient algorithm for solving LSAP [438, 440] that Kuhn called the "Hungarian method" in honor of these two Hungarian mathematicians. These two famous papers were recently reprinted in the 50th anniversary of the original publication, together with an editorial note from H. W. Kuhn [442, 443].

(We mention here that Egerváry's paper [253] also contains a second nice result that directly provides an efficient algorithm for solving the preemptive open shop scheduling problem. This algorithm was independently rediscovered many times in the following years. See Section 3.8.2 and Dell'Amico and Martello [217] for more details.)

### 4.2.1 The Hungarian algorithm

The Hungarian algorithm is recognized as a predecessor of the *primal-dual method* for linear programming, designed one year later by Dantzig, Ford, and Fulkerson [212]. It starts with a feasible dual solution $u, v$ satisfying (4.7) and a partial primal solution (in which less