Linear Assignment Problems and Extensions *

Rainer E. Burkard † Eranda Çela †

Abstract

This paper aims at describing the state of the art on linear assignment problems (LAPs). Besides *sum* LAPs it discusses also problems with other objective functions like the *bottleneck* LAP, the *lexicographic* LAP, and the more general *algebraic* LAP. We consider different aspects of assignment problems, starting with the assignment polytope and the relationship between assignment and matching problems, and focusing then on deterministic and randomized algorithms, parallel approaches, and the asymptotic behaviour. Further, we describe different applications of assignment problems, ranging from the well know personnel assignment or assignment of jobs to parallel machines, to less known applications, e.g. tracking of moving objects in the space. Finally, planar and axial three-dimensional assignment problems are considered, and polyhedral results, as well as algorithms for these problems or their special cases are discussed.
The paper will appear in the Handbook of Combinatorial Optimization to be published by Kluwer Academic Publishers, P. Pardalos and D.-Z. Du, eds.

Keywords: assignment, assignment polytope, linear assignment problems, algorithms, asymptotic behavior.

AMS-classification: 90C27, 68Q25, 90C05

Contents

*This research has been supported by the Spezialforschungsbereich F 003 "Optimierung und Kontrolle", Projektbereich Diskrete Optimierung.

†Technische Universität Graz, Institut für Mathematik B, Steyrergasse 30, A-8010 Graz, Austria.

1 Assignments

Assignment problems deal with the question how to assign n items (e.g. jobs) to n machines (or workers) in the best possible way. They consist of two components: the *assignment* as underlying combinatorial structure and an objective function modeling the "best way".

Mathematically an *assignment* is nothing else than a bijective mapping of a finite set into itself, i.e., a *permutation*. Assignments can be modeled and visualized in different ways: every permutation ϕ of the set $N = \{1, \ldots, n\}$ corresponds in a unique way to a *permutation matrix* $X_\phi = (x_{ij})$ with $x_{ij} = 1$ for $j = \phi(i)$ and $x_{ij} = 0$ for $j \neq \phi(i)$. We can view this matrix X_ϕ as adjacency matrix of a bipartite graph $G_\phi = (V, W; E)$, where the vertex sets V and W have n vertices, i.e., $|V| = |W| = n$, and there is an edge $(i, j) \in E$ iff $j = \phi(i)$, cf. Fig. 1.

$$\phi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix}$$

$$X_\phi = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

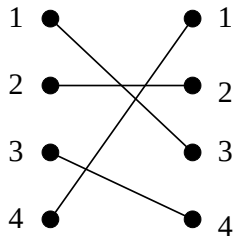


Figure 1: Different representations of assignments

The set of all assignments of n items will be denoted by $\mathcal{S}_n$ and has $n!$ elements. We can describe this set by the following equations called *assignment constraints*

$$\begin{aligned} \sum_{i=1}^{n} x_{ij} &= 1 \qquad \text{for all} \quad j = 1, \ldots, n \\ \sum_{j=1}^{n} x_{ij} &= 1 \qquad \text{for all} \quad i = 1, \ldots, n \\ x_{ij} &\in \{0, 1\} \qquad \text{for all} \quad i, j = 1, \ldots, n \end{aligned} \tag{1}$$

Let A be the coefficient matrix of the system of equations (1). Matrix A is *totally unimodular*, i.e., every square submatrix of A has a determinant of value $+1$, -1 or 0. By replacing the conditions $x_{ij} \in \{0, 1\}$ by $x_{ij} \geq 0$ in (1), we get a *doubly stochastic matrix*. The set of all doubly stochastic matrices forms *the assignment polytope* P_A. Due to a famous result of Birkhoff [36], the assignment polytope P_A is the convex hull of all assignments:

Theorem 1.1 (Birkhoff [36], 1946)
The vertices of the assignment polytope correspond uniquely to permutation matrices.

Differently said every doubly stochastic matrix can be written as convex combination of permutation matrices. Further properties of the assignment polytope are summarized in the following theorem:

Theorem 1.2 (Balinski and Russakoff [22], 1974)
Let P_A be the assignment polytope for assignments on a set N with n elements. Then

1. *dim* $P_A = (n-1)^2$.
2. *Every of the* $n!$ *vertices of* P_A *coincides with* $\sum_{k=0}^{n-1} \binom{n}{k}(n-k-1)!$ *different edges.*
3. *Any two vertices are joined by a path with at most two edges, i.e., the diameter of* P_A *is* 2.
4. P_A *is Hamiltonian.*

A polytope is *Hamiltonian*, if there exists a closed path along its edges which visits every vertex of the polytope just once.

Consider a bipartite graph $G = (V, W; E)$ with vertex sets $V = \{v_1, v_2, \ldots, v_n\}$ and $W = \{w_1, w_2, \ldots, w_n\}$, and edge set E. A subset M of E is called a *matching*, if every vertex of G is incident with at most one edge from M. The cardinality of M is called cardinality of the matching. The *maximum matching problem* asks for a matching with as many edges as possible. A matching M is called a *perfect matching*, if every vertex of G is incident with exactly one edge from M, i.e., $|M| = n$. Evidently, every perfect matching is a maximum matching and corresponds to an assignment. The number of different perfect matchings in a bipartite graph G is given by the *permanent* of the corresponding $n \times n$ adjacency matrix $A = (a_{ij})$ defined by

$$a_{ij} := \begin{cases} 1 & \text{if} \quad (v_i, w_j) \in E \\ 0 & \text{if} \quad (v_i, w_j) \notin E \end{cases}$$

The permanent *per(A)* is defined by

$$\text{per}(A) := \sum_{\phi \in \mathcal{S}_n} a_{1\phi(1)} a_{2\phi(2)} \ldots a_{n\phi(n)}.$$

The numerical evaluation of a permanent (and the determination of the number of different perfect matchings in a graph G) is #P-complete (Valiant [162]). This implies that the determination of the number of different perfect matchings in G is at least as hard as any NP-complete problem. The following theorem due to Hall [99] states a necessary and sufficient condition for the *existence* of a perfect matching in a bipartite graph. Halmos interpreted such a perfect matching as a marriage between ladies in the set V and gentlemen in set W, and due to this interpretation Hall's theorem is known as *Marriage theorem*. For a vertex $v \in V$ let $N(v)$ be the set of its neighbors, i.e., the set of all vertices $w \in W$ which are connected with v by an edge in E. Thus $N(v)$ contains the friends of v. Moreover, for any subset V' of V let $N(V') = \bigcup_{v \in V'} N(v)$.

Theorem 1.3 (Marriage theorem, Hall [99], 1935)
Let $G = (V, W; E)$ *be a bipartite graph.* G *contains an assignment (perfect matching, marriage) if and only if* $|V| = |W|$ *and for all subsets* V' *of* V

$$|V'| \leq |N(V')| \qquad \text{(Hall condition)}$$

This theorem says that by checking the exponentially many subsets V' of V we can decide, whether the graph G has a perfect matching or not. Hopcroft and Karp [103] gave a polynomial-time algorithm to decide this question. They construct a perfect matching, if it exists, in $O(|E|\sqrt{|V|})$ steps. Alt, Blum, Mehlhorn and Paul [9] improved the complexity for dense graphs by a fast adjacency matrix scanning technique and obtain an $O(|V|^{1.5}\sqrt{|E|/\log|V|})$ implementation for the Hopcroft-Karp algorithm.

The decision can also be made faster by a randomized algorithm which is a Monte-Carlo approach. This algorithm may err with an arbitrarily small probability in the case that its output does not confirm the existence of a matching. The randomized algorithm is based on the relationship between the determinant of the Tutte matrix of a graph and the existence of a perfect matching (Tutte [161]). The *Tutte matrix* $A(x) = (x_{ij})$ of an (undirected) graph $G = (V, E)$ is a skew symmetric matrix with variable entries x_{ij}, $x_{ij} = -x_{ji}$, where $x_{ij} = x_{ji} \equiv 0$, if (i, j) is not an edge of G.

Theorem 1.4 (Tutte [161], 1947)
Let $A(x)$ be the Tutte matrix of graph $G = (V, E)$. There exists a perfect matching in G, if and only if the determinant of $A(x)$ is not identically equal to 0.

A randomized algorithm to decide whether G has a perfect matching can be obtained by generating x_{ij} uniformly at random from $\{1, 2, \ldots, 2|E|\}$ and by computing the determinant of the corresponding Tutte matrix $A(x)$, and eventually repeating this process a fixed number of times. The probability of an error can be estimated by applying a result of Schwarz [157] on the probability of hitting roots of a given polynomial. By applying Schwartz's results one gets that the error probability of the above randomized algorithm is equal to $(1/|E|)^r$, where r is the number of repetitions and $|E|$ is the number of edges of G. Since an $n \times n$ determinant can be evaluated in $O(n^{2.376})$ time (see Coppersmith and Vinograd [60]), and by assuming constant time for the generation of a random number, this randomized algorithm is faster than the best existing deterministic algorithm, at least in the case of dense graphs. This procedure does, however, not compute a perfect matching, it just decides - with an arbitrarily small probability of error - whether the given graph has one. To compute a perfect matching in G we may delete edges of G and test whether the remaining graph has a perfect matching. This procedure is repeated until no further edges can be deleted without destroying all perfect matchings in the remaining graph. Obviously, this remaining graph is then a perfect matching. Instead of this naive algorithm Mulmuley, Vazirani and Vazirani [129] propose a parallel algorithm which performs a matrix inversion only *once*. The basic idea relies on assigning random weights to the edges of the graph and on generating the random variables x_{ij} in the Tutte matrix so that the perfect matching with minimum weight is unique. Then, an easy-to-check condition is given for the membership of a certain edge to this minimum weighted perfect matching. This algorithm requires $O(|V|^{3.5}|E|)$ processors to find a perfect matching in $O(\log^2 |V|)$ time.

Cechlárová [57] gives a necessary and sufficient condition that a bipartite graph G contains a *unique* perfect matching. This is the case, if the rows and columns of the adjacency matrix can be permuted by ϕ such that in the permuted matrix $(a_{\phi(i)\phi(j)})$ all 1 entries lie on the main diagonal or below. Cechlárová shows that graphs with a unique perfect matching can be recognized in $O(|E|)$ time and the same time complexity is needed for constructing the unique perfect matching.

The maximum matching problem is particularly easy to solve in so-called *convex bipartite graphs* (Glover [89]). A bipartite graph $G = (V, W; E)$ with vertex sets V and W and edge set E is *convex*, if for all i, k, $i < k$, and j

$$(i, j), (k, j) \in E \Rightarrow \text{all } (i + 1, j), \ldots, (k - 1, j) \in E.$$

Glover described the following simple algorithm for finding a maximum matching in convex bipartite graphs:

1. Compute for every $j \in W$ the label $\pi(j) := \max \{i | (i, j) \in E\}$.

2. Start with the empty matching M.
   For $i = 1, 2, \ldots, |V|$ do:
   If there exists an edge (i, j) with unmatched vertex j, then enlarge the matching by the edge (i, k) with minimum value $\pi(k)$.

A straightforward implementation of this algorithm has complexity $O(n^2)$. Using a method of Gabow and Tarjan [86] the maximum matching can even be found in linear time. Matchings in convex graphs are used in the solution of so-called *terminal assignment problems*. Terminal assignment problems arise in connection with the design of integrated circuits. They can be described as follows: each of n entry terminals positioned on the upper layer of a channel has to be assigned to one of the m exit terminals positioned in the lower layer. The density $d(x)$ of a given assignment at position x is the number of terminals in the upper layer, left to x, which are connected to an exit in the lower layer right to position x, see Fig. 2. The density of an assignment is the maximum of $d(x)$ taken over all positions x. If there are two upper layers, the density is computed for each upper layer separately

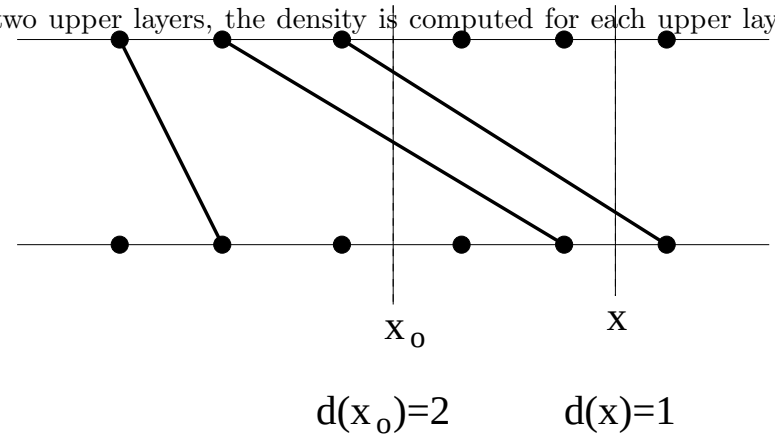


Figure 2: An instance of the terminal assignment problem. The density of this assignment is 2 and is attained at position x_0, $d(x_0) = 2$.

and the total density is the maximum of these two densities. In the terminal assignment problem the terminals situated in two upper layers are to be assigned such that the resulting channel routing problem has minimum total density. Rote and Rendl [155] solve the 2-layer terminal assignment problem by a recursive algorithm in linear time using matchings in convex bipartite graphs.

Flows in networks offer another model to describe assignments. Let G be the bipartite graph introduced before. We embed G in the network $\mathcal{N} = (N, A, c)$ with *node set* N, *arc set* A and *arc capacities* c. The node set N consists of a *source* s, a *sink* t and the vertices of $V \cup W$. The source is connected to every node in V by an arc of capacity 1, every node in W is connected to the sink by an arc of capacity 1, and every edge in E is directed from V to W and supplied with an infinite capacity. A *flow* in network $\mathcal{N}$ is a function $f : A \to \mathbb{R}$ with

$$\sum_{x \in V \cup W} f(x, y) = \sum_{x \in V \cup W} f(y, x)\,, \qquad \text{for all} \quad y \in V \cup W \tag{2}$$

$$0 \le f(x, y) \le c(x, y), \qquad \text{for all} \quad (x, y) \in A \tag{3}$$

Equalities (2) and (3) are called *flow conservation constraints* and *capacity constraints*,

respectively. The *value* $z(f)$ of the flow f is defined as

$$z(f) := \sum_{x \in N} f(s,x)\,.$$

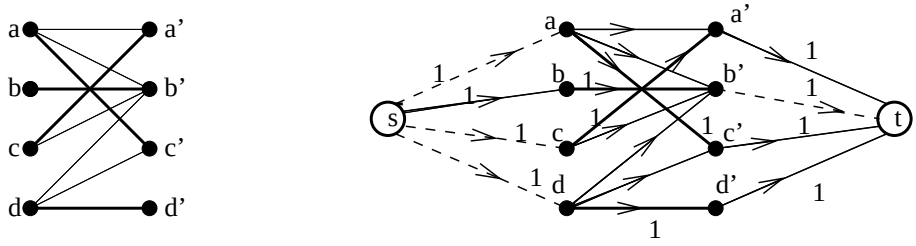


Figure 3: Perfect matching in a bipartite graph and corresponding network flow model. A minimum cut is given by $\{s, b, b'\}$. Th dashed edges contribute to the value of the cut.

The *maximum network flow problem* asks for a flow with maximum value $z(f)$. Obviously, a maximum integral flow in the special network constructed above corresponds to a matching with maximum cardinality. A *cut* in the network $\mathcal{N}$ is a subset C of the node set N with $s \in C$ and $t \notin C$. The value $u(C)$ of cut C is defined as

$$u(C) := \sum_{\substack{x \in C,\, y \notin C \\ (x,y) \in A}} c(x,y)\,.$$

Ford and Fulkerson's famous *Max Flow-Min Cut Theorem* [85] states that the value of a maximum flow equals the minimum value of a cut.

This leads immediately to the matching theorem of König [116]. Given a bipartite graph G, a *vertex cover* (cut) in G is a subset of its vertices such that every edge is incident with at least one vertex in this set.

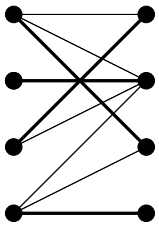
Theorem 1.5 (König's Matching Theorem, 1931)
In a bipartite graph the minimum number of vertices in a vertex cover equals the maximum cardinality of a matching.

Let us now translate this theorem in the language of 0-1 matrices. Given a bipartite graph $G = (V, W; E)$ with $|V| = |W| = n$, we define an $n \times n$ 0-1 matrix $B = (b_{ij})$ as follows

$$b_{ij} := \begin{cases} 0 & \text{if} \quad (i,j) \in E \\ 1 & \text{if} \quad (i,j) \notin E \end{cases}$$

A *zero-cover* is a subset of rows (columns) of matrix B which contains all 0 elements. A row (column) which is an element of a zero-cover is called a *covered row* (*covered column*). Now we get

Theorem 1.6 *There exists an assignment ϕ with $b_{i\phi(i)} = 0$ for all $i = 1, \ldots, n$, if and only if the minimum zero cover has n elements.*



$$\begin{pmatrix} 0 & 0 & 0^* & 1 \\ 1 & 0^* & 1 & 1 \\ 0^* & 0 & 1 & 1 \\ 1 & 0 & 1 & 0^* \end{pmatrix}$$

Figure 4: 0-1 matrix model of a bipartite graph and the corresponding minimum vertex cover which corresponds to the cut in Fig. 3. An assignment is given by the entries marked by $*$.

A maximum matching corresponds uniquely to a maximum flow in the corresponding network $\mathcal{N}$, and therefore to a minimum cut C in this network. From this minimum cut we can construct a zero-cover in the adjacency matrix: if node $i \in V$ of the network does not belong to cut C, then row i is an element of the zero-cover. Analogously, if node $j \in W$ belongs to cut C, then column j is an element of the zero-cover.

2 Linear Sum Assignment Problems (LSAPs)

In this section we deal with the classical *linear sum assignment problem* (LSAP). We first introduce the problem and some alternative interpretations, and then discuss some applications of the LSAP.

Let us return to the original model where n items (e.g. jobs) are to be assigned to n machines (or workers) in the best possible way. Let us assume that the assignment of job i to machine j incurs a cost c_{ij}. In order to complete all jobs in the cheapest possible way, we have to minimize the objective function

$$\sum_{i=1}^{n} c_{i\phi(i)} \,, \tag{4}$$

i.e., we have to find an assignment ϕ^* which minimizes (4). Thus, we get the *linear sum assignment problem* (LSAP) as

$$\min \sum_{i=1}^{n} c_{i\phi(i)} \,. \tag{5}$$

Based on the description (1) of the set of all assignments (see Section 1), the LSAP can also be formulated as a 0-1 linear program:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} c_{ij} x_{ij} \\ & \sum_{i=1}^{n} x_{ij} = 1 && j = 1, \ldots, n \\ & \sum_{j=1}^{n} x_{ij} = 1 && i = 1, \ldots, n \\ & x_{ij} \in \{0, 1\} && i, j = 1, \ldots, n. \end{aligned} \tag{6}$$

Due to Theorem 1.1 we can relax the conditions $x_{ij} \in \{0,1\}$ to $x_{ij} \geq 0$ and solve the corresponding linear program, denoted by LP: any basic solution of this linear program corresponds to a permutation matrix. We will see in Section 3 that LP-based algorithms belong to the most important solution techniques for the LSAP. Often they tackle the dual linear program: Let u_i, $1 \leq i \leq n$, and v_j, $1 \leq j \leq n$, be dual variables. Then the dual of the linear program LP is given by

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j & \\ & u_i + v_j \leq c_{ij} & i,j = 1,\ldots,n \\ & u_i, v_j \ \in \mathbb{R} & i,j = 1,\ldots,n. \end{aligned} \tag{7}$$

Keeping in mind the matching interpretation of assignments, the LSAP can be equivalently formulated as a matching problem: given a bipartite graph $G = (V, W; E)$ with edge weights c_{ij} for all edges $(i,j) \in E$, find a perfect matching with minimum weight:

$$\min \left\{ \sum_{(i,j)\in\mathcal{M}} c(i,j) \colon \mathcal{M} \text{ is a perfect matching} \right\}.$$

The LSAP can be solved efficiently, and the design of efficient solution methods for this problem has been an object of research for many years. There exists an amazing amount of algorithms for the LSAP, ranging from the simple Hungarian method to more recent developments involving sophisticated data structures and including parallel algorithms. (See Section 3 for a more detailed discussion of these topics). There are, however, some simple cases where an optimal solution of the LSAP can be given before hand, even if the coefficients c_{ij} of the cost matrix are not known. These cases are related to *Monge properties* of the cost matrix C.
A matrix $C = (c_{ij})$ is said to be a *Monge matrix* (cf. Hoffman [102]), if it fulfills the following conditions:

$$c_{ij} + c_{kl} \leq c_{il} + c_{kj}\,, \qquad \text{for } 1 \leq i < k \leq n,\ 1 \leq j < l \leq n. \tag{8}$$

It is easy to show that the identical permutation is an optimal solution of an LSAP with a cost matrix which fulfills (8). This remains true, even if condition (8) is relaxed to the so-called *weak Monge property* (cf. Derigs, Goecke and Schrader [69])

$$c_{ii} + c_{kl} \leq c_{il} + c_{ki}\,, \qquad \text{for } 1 \leq i < k \leq n,\ 1 \leq i < l \leq n. \tag{9}$$

Analogously, it can be shown that the permutation ϕ defined by $\phi(i) = n - i + 1$ for all i, is an optimal solution of an LSAP with cost matrix C fulfilling the *inverse Monge property:*

$$c_{ij} + c_{kl} \geq c_{il} + c_{kj} \qquad \text{for } 1 \leq i < k \leq n,\ 1 \leq j < l \leq n. \tag{10}$$

Obviously, Monge properties depend on the proper numbering of the rows and columns of the matrix. A matrix C is called a *permuted Monge matrix*, if there exists a pair of permutations (ϕ, ψ) such that the matrix $C^{(\phi,\psi)} = (c_{\phi(i)\psi(j)})$ obtained from C by permuting its rows by ϕ and its columns by ψ, is a Monge matrix, see Burkard, Klinz and Rudolf [42]. Due to a result of Deĭneko and Filonenko [66] it can be recognized in $O(n^2)$ time, whether an

$n \times n$ matrix C is a permuted Monge matrix. Moreover, the appropriate permutations ϕ, ψ for the rows and the columns can also be found within this time bound. As a consequence, the LSAP with a permuted Monge cost matrix can be solved in $O(n^2)$ time. The reader is referred to Burkard et al. [42] for a detailed discussion of Monge properties, and a description of the algorithm of Deĭneko and Filonenko.
An important special case of the LSAP with permuted Monge cost matrix arises if the cost coefficients have the form

$$c_{ij} = u_i v_j$$

with nonnegative numbers u_i and v_j. Such an LSAP can simply be solved in $O(n \log n)$ time by ordering the elements u_i and v_j.

Theorem 2.1 (Hardy, Littlewood, and Pólya [101], 1952)
Let $0 \leq u_1 \leq \ldots \leq u_n$ *and* $0 \leq v_1 \leq \ldots \leq v_n$. *Then for any permutation* ϕ

$$\sum_{i=1}^{n} u_i v_{n+1-i} \leq \sum_{i=1}^{n} u_i v_{\phi(i)} \leq \sum_{i=1}^{n} u_i v_i.$$

Let us now discuss some applications of the linear sum assignment problem. LSAPs occur quite frequently as subproblems in more involved applied combinatorial optimization problems like the quadratic assignment problem, traveling salesman problems or vehicle routing problems. Classical direct applications include the assignment of personnel (cf. e.g. Machol [122], Ewashko and Dudding [78]), and the assignment of jobs to parallel machines, but LSAPs have quite a number of further interesting applications. Neng [130] modeled the optimal engine scheduling in railway systems as an assignment problem. Further applications in the area of vehicle and crew scheduling are described by Carraresi and Gallo [49]. Another application concerns problems in telecommunication, in particular in earth - satellite systems. Usually the data to be remitted are buffered in the ground stations and are sent in very short data bursts to the satellite. There they are received by transponders and then, are transmitted again to the receiving earth station. A transponder just connects one sending station with a receiving station. For a fixed time interval of variable length λ_k the n sending stations are connected with the receiving stations via the n transponders onboard the satellite, i.e., a certain switch mode is applied. Mathematically, a *switch mode* P_k corresponds to a permutation matrix. Then the connections onboard the satellite are changed simultaneously and in the next time interval new pairs of sending and receiving stations are connected via transponders, i.e., a new switch mode is applied. Given an $(n \times n)$ traffic matrix $T = (t_{ij})$, where t_{ij} describes the amount of information to be remitted from the i-th sending station to the j-th receiving station, we have to determine the switch modes P_k, $k = 1, 2, \ldots$, and the nonnegative lengths λ_k of the corresponding time slots during which the switch modes P_k are applied, such that all data are remitted in the shortest possible time. Summarizing, our problem consists of finding suited switch modes and corresponding lengths for the time slots during which the switch modes are applied. This leads to the following model:

$$\begin{array}{lll} \min & \sum_k \lambda_k & \\ s.t. & & \\ & \sum_k \lambda_k p_{ij}^{(k)} \geq t_{ij}\,, & \text{for } 1 \leq i, j \leq n \\ & \lambda_k \geq 0\,, & \text{for all } k \end{array} \tag{11}$$

Since here the time is split among the participating stations, all of them having access to the satellite, such a traffic model is called time-division-multiple-access (TDMA) model. The problem can be transformed to an LSAP, if a system of $2n$ fixed switch modes $P_1, \ldots, P_n, Q_1, \ldots, Q_n$ is installed onboard the satellite, where

$$\sum_k P_k = \sum_l Q_l = \mathbf{1}$$

and $\mathbf{1}$ is the matrix with 1-entries only. For details see Burkard [38].

Brogan [37] describes assignment problems in connection with locating objects in space. Let us consider n objects which are detected by two sensors at geographically different sites. Each sensor measures the angle under which the object can be seen, i.e., it provides n lines, on which the objects lie. The location of every object is found by intersecting the appropriate lines. The pairing of the lines is modeled as follows: let c_{ij} be the smallest distance between the i-th line from sensor 1 and the j-th line from sensor 2. Due to small errors during the measurements, c_{ij} might even be greater than 0 if some object is determined by lines i and j. Solving an LSAP with cost matrix $C = (c_{ij})$ leads to very good results in practice. A similar technique can be used for tracking n moving objects (e.g. missiles) in space. If their locations at two different times t_1 and t_2 are known, we compute the (squared) Euclidean distances between any pair of old and new locations and solve the corresponding assignment problem in order to match the points in the right way.

For some further applications of assignment problems, including the optimal depletion of inventory, see Ahuja, Magnanti, Orlin and Reddy [2].

There are also other possibilities for the choice of the objective function in assignment problems. For example, if c_{ij} is the time needed for machine j to complete job i and all machines run in parallel, the shortest completion time for all jobs is given by $\max_{1 \le i \le n} \{c_{i\phi(i)}\}$. This leads to the *linear bottleneck assignment problem* (LBAP) which will be treated in Section 5. Other objective functions will be discussed in Section 6.

3 Algorithms for the LSAP

A large number of algorithms, sequential and parallel, have been developed for the linear sum assignment problem (LSAP). They range from primal-dual combinatorial algorithms, to simplex-like methods, cost operation algorithms, forest algorithms, and relaxation approaches. The worst-case complexity of the best sequential algorithms for the LSAP is $O(n^3)$, where n is the size of the problem. There is a number of survey papers and books on algorithms, e.g. Derigs [68], Martello and Toth [126], Bertsekas [30], Akgül [6], and the book on the first DIMACS challenge edited by Johnson and McGeoch [106]. Among papers reporting on computational experience we mention Derigs [68], Carpaneto, Martello and Toth [51], Carpaneto and Toth [56], Lee and Orlin [121], and some of the papers in [106].

Most sequential algorithms for the LSAP can be classified into primal dual algorithms and simplex-based algorithms. Primal-dual algorithms work with a pair consisting of an infeasible primal and a feasible dual solution which fulfill the complementarity slackness conditions. Such algorithms update the solutions iteratively until the primal solution becomes feasible, while keeping the complementary slackness conditions fulfilled. At this point the primal solution is also optimal, according to duality theory.

Simplex-based algorithms are special implementations of the primal and the dual simplex algorithm for linear programming. These algorithms can be applied to solve the LSAP

due to Theorem 1.1. Although simplex algorithms are not polynomial in general, there are special polynomial implementations of them in the case of the LSAP.
More recently, an interior point algorithm has been applied to the LSAP, see Ramakrishnan, Karmarkar, and Kamath [152]. This algorithm produces promising results, in particular when applied to large size instances.

From the computational point of view very large scale dense assignment problems with about 10^6 nodes can be solved within a couple of minutes by sequential algorithms, see [121].

A number of parallel approaches developed for the LSAP show that the speed up achieved by such algorithms is limited by the sparsity of the cost matrices and/or the decreasing load across the iterations, see e.g. Balas, Miller, Pekny, and Toth [14], Bertsekas and Castañon [31, 32], Castañon, Smith, and Wilson [53], as well as Wein and Zenios [167, 168]. More information on these topics is given in Section 3.4.

3.1 Primal-Dual Algorithms

Primal-dual algorithms work with a pair of an unfeasible primal solution x_{ij}, $x_{ij} \in \{0, 1\}$, $1 \leq i, j \leq n$, and a feasible dual solution u_i, v_j, $1 \leq i, j \leq n$, which fulfill the complementary slackness conditions:

$$x_{ij}(c_{ij} - u_i - v_j) = 0\,, \quad \text{for } 1 \leq i, j \leq n \tag{12}$$

We denote $\bar{c}_{ij} := c_{ij} - u_i - v_j$ and call $\bar{c}_{ij}$ *reduced costs* with respect to the dual solution u_i, v_j. This pair of solutions is updated iteratively by means of so-called *admissible transformations* of the costs matrix C. Also the process of determining the first pair of primal and dual solutions can be seen as an admissible transformation. A transformation T of a matrix $C = (c_{ij})$ to a matrix $\tilde{C} = (\tilde{c}_{ij})$ is called admissible if for each permutation ϕ of $1, 2, \ldots, n$ the following equality holds:

$$\sum_{i=1}^{n} c_{i\phi(i)} = \sum_{i=1}^{n} \tilde{c}_{i\phi(i)} + z(T)\,,$$

where $z(T)$ is a constant which depends only on the transformation T. Clearly, an admissible transformation does not change the order of the permutations (feasible solutions of the LSAP) with respect to the value of the objective function. Hence the LSAPs with cost matrices C and $\tilde{C}$ are equivalent. We may assume without loss of generality that the entries of C are nonnegative. (Otherwise we could add a large enough constant to all entries of C. This results in just adding a constant to the objective function, and hence, does not change the optimal solution of the problem but only its optimal value.)
If all entries $\tilde{c}_{ij}$ of matrix $\tilde{C}$ are nonnegative and there exists a permutation ϕ^* such that $\tilde{c}_{i\phi^*(i)} = 0$, then ϕ^* is an optimal solution of the original LSAP with cost matrix C. The optimal value is equal to $z(T)$, where T is the transformation from C to $\tilde{C}$.

Primal-dual algorithms differ on 1) the way they obtain their first pair of a primal and a dual solution fulfilling the complementary slackness conditions, and 2) the implementation of the update of dual variables.

The Hungarian method

The first polynomial-time primal-dual algorithm for the LSAP is the Hungarian method due to Kuhn [118]. In the Hungarian method a starting dual solution is obtained by so-called

row and column reductions:

$$u_i = \min\{c_{ij} : 1 \le j \le n\}, \quad \text{for } 1 \le i \le n,$$

and then

$$v_j = \min\{c_{ij} - u_i : 1 \le i \le n\} \quad \text{for } 1 \le j \le n\,.$$

An infeasible primal start solution is obtained by finding a matching $\mathcal{M}$ of maximal cardinality in the graph $\bar{G} = (V, W; \bar{E})$, where $V = W = \{1, 2, \ldots, n\}$, and $\bar{E} = \{(i,j) : \bar{c}_{ij} = c_{ij} - u_i - v_j = 0\}$. Then we set $x_{ij} = 1$ if (i,j) is an edge of the matching $\mathcal{M}$, and $x_{ij} = 0$, otherwise. Clearly, the primal solution x_{ij}, $1 \le i, j \le n$, and the dual solution u_i, v_j, $1 \le i, j \le n$, fulfill the complementarity slackness conditions.
According to Theorem 1.5, the matching $\mathcal{M}$ corresponds to a minimal vertex cover of graph $\bar{G}$, or equivalently to a minimal zero-cover of the matrix. Let I be the set of row indices of uncovered rows and let J contain the indices of uncovered columns. Then the dual variables are updated by setting

$$u_i := \begin{cases} u_i - \delta & \text{if } i \in I \\ u_i & \text{otherwise} \end{cases} \qquad v_j := \begin{cases} v_j & \text{if } j \in J \\ v_j + \delta & \text{otherwise} \end{cases},$$

where δ is defined as minimum of the currently uncovered reduced costs, i.e., $\delta = \min\{\bar{c}_{ij} : i \in I, j \in J\}$. This corresponds to an admissible transformation T given by

$$\tilde{c}_{ij} := \begin{cases} c_{ij} - \delta & \text{if } i \in I, j \in J \\ c_{ij} + \delta & \text{if } i \notin I, j \notin J \\ c_{ij} & \text{otherwise} \end{cases}. \tag{13}$$

Its constant $z(T)$ is equal to $\delta(|I| + |J| - n)$. This transformation preserves the nonnegativity of the reduced cost matrix (or equivalently, the feasibility of the dual solution). According to Theorem 1.6, the constant $z(T)$ equals zero ($|I| + |J| = n$), if and only if there is a permutation ϕ such that $c_{i\phi(i)} = 0$, for $1 \le i \le n$. In the latter case we have a feasible primal solution, and hence also an optimal one.

One possibility to implement the Hungarian method so that it achieves a worst case time complexity of $O(n^3)$ is by means of so-called *alternating trees*. Such a tree contains *matched edges* - which are edges from $\mathcal{M}$, and *free edges* which do not belong to $\mathcal{M}$. Analogously, *free nodes* are nodes which are adjacent only to free edges, and *matched nodes* are adjacent to one matched edge.

Definition 3.1 *An alternating tree is a directed tree rooted at some free node $r \in V$, where all arcs are directed away from the root and*

- *all matched edges are directed from W to V and all free edges are directed from V to W,*
- *the root r is the only free node,*
- *all leafs are nodes from V.*

Starting from a maximal matching in graph $\bar{G}$ the algorithm builds an alternating tree in $\bar{G}$. As soon as this tree reaches a leaf $v \in V$ which is adjacent to some free node in W, there is an *alternating path* between the free nodes v and w, i.e., there is a path joining v

and w in G whose edges are alternatively free and matched edges. Such a path is called an *augmenting path*, because by exchanging the free and matched edges in it we get a new maximal matching in $\bar{G}$, whose cardinality is larger than that of the previous maximal matching. In this case we can increase the number of primal variables x_{ij} which are equal to 1, and thus *improve* the primal solution without changing the dual one. There may be at most n such augmentations because the cardinality of a matching cannot exceed n. After having performed all possible augmentation steps we get finally a *maximal alternating tree*, i.e., an alternating tree where all leafs have cardinality equal to 1 in $\bar{G}$. At this point, the dual variables - and with them the reduced costs and the graph $\bar{G}$ - are updated by an admissible transformation. Thus some new edges are added to $\bar{G}$, i.e., additional reduced costs equal to 0 arise. Now the existing alternating tree can be updated with respect to the new graph $\bar{G}$. It is possible to implement the update of dual variables so that all due updates between two consecutive augmentations are performed by applying $O(n^2)$ elementary operations. Since there may be at most n augmentations, this leads to an $O(n^3)$ algorithm.

Shortest augmenting path algorithms

Although the Hungarian algorithm described above grows an alternating tree to augment the primal solution, in principle only alternating augmenting paths are used. This observation is the conceptual basis of numerous *shortest augmenting path algorithms* for the LSAP, which belong to the class of primal-dual methods and can also be implemented in $O(n^3)$ steps. Given a pair of a primal solution x_{ij}, $1 \leq i,j \leq n$, and a dual feasible solution u_i, v_j, $1 \leq i,j \leq n$, which fulfill the complementarity slackness conditions, together with the reduced costs $\bar{c}_{ij}$, construct a weighted directed bipartite graph $\tilde{G} = (V, W; \tilde{E})$ with arc set $\tilde{E} = D \cup R$. Here $D = \{(i,j)\colon (i,j) \in E,\ x_{ij} = 0\}$ is the set of *forward arcs*, and $R = \{(j,i)\colon (i,j) \in \bar{E},\ x_{ij} = 1\}$ is the set of *backward arcs*. The weights of the backward arcs are set equal to 0, whereas the weights of the forward arcs are set equal to the corresponding reduced costs. We select a free node r in V and solve the single-source shortest path problem, i.e., compute the shortest paths from r to all nodes of $\tilde{G}$. The shortest among all paths from r to some free node in W is used to augment the current primal solution by swapping the free and matched edges. The primal and dual solutions as well as the reduced costs are updated similarly as in the case of the Hungarian method. It can be shown that each (unfeasible) primal solution (or equivalently, each matching in $\bar{G}$) constructed in the course of the algorithm has minimum weight among all solutions of the same cardinality, see e.g. Lawler [119], Derigs [68]. Hence, after n augmentations we obtain an optimal primal solution. The single-source shortest paths can be computed in $O(n^2)$ time in a graph with nonnegative weights, e.g. by applying the Dijkstra algorithm implemented with Fibonacci heaps, see Fredman and Tarjan [80]. This implies that the overall complexity of augmenting path algorithms amounts to $O(n^3)$.

The shortest path computation may stop as soon as the first path joining r with some free node in W has been computed.

Basically shortest augmenting path algorithms for the LSAP may differ in the way they determine a starting pair of primal and dual solutions, and by the subroutine they use for computing the shortest paths. (The reader is referred to Cherkassky, Goldberg, and Radzik [59] for a review on shortest paths algorithms. For developments on data structures used in shortest path algorithms the reader is referred to Fredman and Tarjan [80], and to Ahuja, Mehlhorn, Orlin, and Tarjan [3].) Most of the existing algorithms use the Dijkstra

algorithm for the shortest path computations, e.g. Carpaneto and Toth [56], Jonker and Volgenant [107, 108], and Volgenant [164].
Several authors try to speed up the shortest path computations by *thinning out* the underlying bipartite graph, e.g. Carraresi and Sodini [52], Glover, Glover, and Klingman [90]. In these algorithms only "short" edges of the graph $\bar{G}$ are considered in the shortest path computations. A *threshold value* which is updated after each augmentation is used to determine the *short* edges. Therefore these algorithms are sometimes called *threshold algorithms*. Their worst case complexity is also $O(n^3)$.
Some shortest augmenting path algorithms apply specific implementations of shortest paths computations especially suited for sparse matrices, see [56], and [164]. To apply this technique the cost matrix is thinned-out by some heuristic. If no feasible primal solution is found for the sparse problem, then the cost matrix is enlarged by re-adding some of the "canceled" entries.

The auction algorithm

Another primal-dual algorithm, called *the auction algorithm*, has been proposed by Bertsekas [28]. The auction algorithm is similar to the Hungarian method to the extent that the updates of primal and dual solutions can be modeled by admissible transformations. However, whereas in the Hungarian method the value of the dual objective function increases after each such an update, in auction algorithms it is only guaranteed that this value does not decrease. Another difference is that in the Hungarian algorithm a matched vertex of V or W remains matched for the rest of the algorithm, while this property only holds for vertices of V in the case of the auction algorithm. Some vertex of W matched in a certain iteration of the auction algorithm may become free in some further iteration. The original auction algorithm was developed for maximization problems, where we try to maximize the objective function of the LSAP under the assignment constraints. In the minimization case the auction algorithm associates each vertex of W with an item to be sold, and each vertex of V with a customer. c_{ij} is the cost of item j for customer i, the dual variable v_j is the profit on item j, and the difference $c_{ij} - v_j$ is customer's i cost margin relative to the profit on item j. The algorithm starts with a pair of an (unfeasible) primal solution x_{ij} (e.g. $x_{ij} = 0$ for all i, j) and a feasible dual solution u_i, v_j. This pair of solutions fulfills the complementarity slackness conditions (12), and is updated iteratively by maintaining (12). The update of the current x_{ij}, u_i, v_j, to obtain the subsequent pair of solutions x'_{ij}, u'_i, v'_j, is made by choosing a free customer $\bar{i}$ ($\bar{i} \in V$), and by determining the smallest and second smallest cost margin $\bar{u}$ and $\tilde{u}$, respectively:

$$\bar{u} = \min\{c_{\bar{i}j} - v_j : 1 \le j \le n\}, \quad \bar{j} = \operatorname{argmin}\{c_{\bar{i}j} - v_j : 1 \le j \le n\},$$

$$\tilde{u} = \min\{c_{\bar{i}j} - v_j : 1 \le j \le n, j \ne \bar{j}\}.$$

Then the algorithm distinguishes two cases: Case 1, where $\bar{u} < \tilde{u}$, or $\bar{u} = \tilde{u}$ and $\bar{j}$ is free, and Case 2, where $\bar{u} = \tilde{u}$ and $\bar{j}$ is matched. In the first case the algorithm assigns $\bar{i}$ to $\bar{j}$ ($x'_{\bar{i}\bar{j}} = 1$) and updates the dual variables $u_{\bar{i}}$ and $v_{\bar{j}}$ so as to preserve dual feasibility: $u'_{\bar{i}} = \tilde{u}$ and $v'_{\bar{j}} = v_{\bar{j}} + (\bar{u} - \tilde{u})$. If $\bar{j}$ was not a free vertex, i.e., there was some $\hat{i}$ such that $x_{\hat{i}\bar{j}} = 1$, the new primal solution sets $x'_{\hat{i}\bar{j}} = 0$. In the second case, where $\bar{u} = \tilde{u}$ and there is an index $\hat{i}$ such that $x_{\hat{i}\bar{j}} = 1$, the algorithm sets $u_{\bar{i}} := \bar{u}$ and performs a labeling procedure to either update the dual variables so as to assign $\bar{j}$ to $\bar{i}$ or to get an augmenting path starting from $\bar{i}$ and ending at some free vertex in W. In the latter case we would have an update of the pair

of primal and dual solutions as in the case of the Hungarian method (see [28] for a complete description). This auction algorithm is pseudopolynomial and runs in $O(n^2(n+R))$ time, where $R = \max\{|c_{ij}|: 1 \leq i,j \leq n\}$. The similarity with the Hungarian method allows the combination of both algorithms as proposed in [28]. The combined algorithm starts by applying the auction algorithm, counts the iterations which do not lead to an increase of the number of items assigned to customers, and switches to the Hungarian method as soon as the counter exceeds an appropriately defined parameter. It can be shown that the worst case complexity of this combination is $O(n^3)$.

Bertsekas and Eckstein [35] modified the above described auction algorithm to obtain a polynomial-time algorithm with worst case complexity $O(n^3 \log(nR))$, where R is defined as above. The algorithm works with a pair of a primal and a dual solution which fulfills the *ϵ-complementarity conditions*

$$x_{ij}(c_{ij} - u_i - v_j) \geq -\epsilon \,, \text{for } 1 \leq i,j \leq n.$$

In terms of the ϵ-complementarity conditions an ϵ-relaxed problem is defined: It consists of finding a pair of feasible primal and dual solutions of the given LSAP which fulfills the ϵ-complementarity conditions. It can be shown that an optimal solution of the $1/n$-relaxed problem ($\epsilon = \frac{1}{n}$) is also an optimal solution of the initial problem. This fact suggests to apply so-called *scaling techniques*, i.e., to solve initially the ϵ-relaxed problem for a large ϵ, and then to decrease ϵ gradually until it reaches $1/n$.
When ϵ is decreased to ϵ' the optimal solution of the previous ϵ-relaxed problem is updated with "few" efforts to obtain the optimal solution of the ϵ'-relaxed problem. Bertsekas et al. [35] implement this ϵ-scaling in terms of a scaling of the costs. All costs c_{ij} are multiplied by $n+1$ and a 1-relaxed problem ($\epsilon = 1$) is solved. If x_{ij} and u_i, v_j are optimal primal and dual solutions of the 1-relaxed problem with costs $c'_{ij} = (n+1)c_{ij}$, then x_{ij} and $\frac{u_i}{n+1}$, $\frac{v_j}{n+1}$, is an optimal pair of primal and dual solutions for the $\frac{1}{n}$-relaxed problem with costs c_{ij}. Consequently, x_{ij} is an optimal solution of the LSAP with costs c_{ij}. The 1-relaxed subproblem with costs c'_{ij} is solved by solving $M = O(log(nR))$ subproblems consecutively.
The m-th subproblem is a 1-relaxed LSAP with costs given by $\frac{c'_{ij}}{2^{M-m}}$, and is solved by an algorithm similar to the original auction algorithm [28]. For a free node $\bar{i} \in V$, the values $\bar{u}$, $\tilde{u}$, and the index $\bar{j} \in W$ are determined as in the original auction algorithm, and two analogous cases are distinguished. The dual variable corresponding to $\bar{i}$ is set to $u_{\bar{i}} := \tilde{u} + \frac{\epsilon}{2}$, and the dual variable corresponding to $\bar{j}$ is set to $v_{\bar{j}} := v_{\bar{j}} + (\bar{u} - \tilde{u}) - \frac{\epsilon}{2}$.
Bertsekas et al. [35] call *bidding* the update of $u_{\bar{i}}$. The version where *one* node bids at a time is called Gauss–Seidel version of the auction algorithm and is better suited for sequential implementations (see [35] for more information). Another version, where *all* free customers bid simultaneously is called Jacobi version of the auction algorithm and is more appropriate for parallel implementations, see Bertsekas [29].

Pseudoflow algorithms

Other primal-dual algorithms for the LSAP are based on the formulation of the LSAP as a minimum cost flow problem (see Section 1). A *pseudoflow* is a flow which fulfills the capacity constraints (3), but does not necessarily fulfill the flow conservation constraints (2). Pseudoflow algorithms work with ϵ-relaxations of the minimum cost flow problem where the flow conservation constraints are violated by at most ϵ. They iteratively transform the starting solution, which is a pseudoflow, into an optimal solution for the ϵ-relaxed

problem. Then, ϵ is decreased and the procedure is repeated until ϵ reaches $\frac{1}{n}$. At that point an optimal solution of the ϵ-relaxed flow problem is also an optimal solution for the original flow problem, and hence for the LSAP. The general idea is quite similar to the auction algorithm of Bertsekas et al. [35] which involves ϵ-complementarity conditions. One difference is that pseudoflow algorithms apply ϵ scaling, whereas the auction algorithm in [35] applies cost scaling. Another difference concerns the transformation of a pseudoflow into an optimal solution of the ϵ-relaxed flow problem, which is done by the push-relabel method described by different authors, e.g. Goldberg and Tarjan [96], Cherkassky and Goldberg [58]. If ϵ is divided by $\delta \geq 2$ in each iteration, it can be shown that the push relabel algorithm terminates after $O(\log_\delta(nR))$ iterations, where R is defined as in the previous section. Different pseudoflow algorithms for the LSAP have been developed by Orlin and Ahuja [133], Goldberg, Plotkin and Vaidya [95], and Goldberg and Kennedy [94]. These algorithms are particularly efficient for sparse problems: the worst-case complexity of the first two algorithms is $O(\sqrt{n}m\log(nR))$, where m is the number of finite entries of the cost matrix C. (Infinite entries correspond to non-existing edges in the matching model.)

3.2 Simplex-Based Algorithms

Simplex-based algorithms for the LSAP are specific implementations of the network simplex algorithm. The latter is a specialization of the simplex method for linear programming to network problems and is due to Dantzig, see e.g. [65]. The specialization relies on exploiting the combinatorial structure of network problems to perform more efficient pivots acting on trees rather than on the coefficient matrix. A number of such algorithms for the LSAP, and more generally for the transportation problem of whom the LSAP is a special case, have been developed in the early 1970s, see e.g. Glover, Karney and Klingman [91], Glover and Klingman [92]. It is well known that there is a one-to-one correspondence between primal (integer) basic solutions of the LSAP and spanning trees of the bipartite graph G related to assignment problems as described in Section 1. Moreover, given the spanning tree, the values of the corresponding dual variables fulfilling the complementarity slackness conditions are uniquely determined, as soon as the value of one of those variables is fixed (arbitrarily). Every primal feasible basic solution is highly degenerate because it contains $2n-1$ variables and $n-1$ of them are equal to 0. Hence degeneracy poses a problem, and the first simplex-based algorithms for the LSAP were exponential. The first steps towards the design of polynomial-time simplex-based algorithms were made by introducing the concept of so-called *strongly feasible trees*, due to Cunningham [62, 63] and Barr, Glover, and Klingman [26].

Primal simplex-based algorithms

Primal simplex-based algorithms work with a primal feasible basic solution which corresponds to a spanning tree T in G. n of the x variables corresponding to edges of the tree are equal to 1, the other $n-1$ edges of the tree correspond to x variables equal to 0. Given the primal variables x_{ij}, the dual variables u_i, v_j are uniquely determined by setting one of them arbitrarily and computing the others such that the complementarity conditions are fulfilled, i.e., $\bar{c}_{ij} = c_{ij} - u_i - v_j = 0$ for all i, j such that $(i,j) \in T$. If the reduced costs corresponding to the edges of the co-tree $T^c = E \setminus T$ are nonnegative, the current basis T is optimal. Otherwise, there is an edge $(i,j) = e \in T^C$ with $\bar{c}_{ij} < 0$. Adding this edge to T creates a unique cycle $C(T,e)$. The subsequent basis T' is given by $T' := (T \setminus \{f\}) \cup \{e\}$, where $f = (i',j')$ is some edge in $C(T,e)$, with $x_{i'j'} = \min\{x_{ij} \colon (i,j) \in C(T,e)\}$.

Barr et al. [26] and Cunningham [62, 63] introduced a special type of spanning trees in G, called *strongly feasible trees*. They showed that the network simplex method for the LSAP can be implemented so as to visit only bases which correspond to strongly feasible trees. Strongly feasible trees are defined similarly as alternating trees in the Hungarian method.

Definition 3.2 *A strongly feasible tree is a rooted directed tree in G with root in V and arcs directed away from the root, such that for all arcs (i,j) with $i \in V$ and $j \in W$, $x_{ij} = 1$, and for all arcs (i,j) with $i \in W$ and $j \in V$, $x_{ij} = 0$.*

Strongly feasible trees have a number of useful properties with respect to the primal network simplex algorithm. Before we describe them, let us say that $(i,j) = f \in T^C$ is a *forward* (*backward*) arc if $i \in V$ ($j \in W$) lies on the directed path joining the root r of T with $j \in W$ ($i \in V$). An arc f which is neither backward nor forward is called a *cross* arc.

Proposition 3.3 (Barr et al. [26], 1977, Cunningham [62, 63], 1976, 1979) *Let T be a strongly feasible tree for an LSAP instance, and let T^C be its corresponding co-tree. Then*

1. *The root has degree 1, every other node from V has degree 2.*
2. *If the edges $e = (i,j)$ and $f = (i',j')$, $i, i' \in V$, $j, j' \in W$, satisfy the properties $e \in T^C$, $f \in c(T,e)$, and $i \equiv i'$ then $(T \setminus \{f\}) \cup \{e\}$ is again a strongly feasible tree.*
3. *For an $e \in T^C$ the pivot by the pair (e,f) defined as above is non-degenerate iff e is a forward arc.*

Cunningham showed that the primal network simplex algorithm for the LSAP cannot perform more than $O(n^2)$ degenerate pivots at some vertex of the assignment polytope, if strongly feasible trees and a specific pivot rule are employed. By combining these ideas with Dantzig's pivot rule (the new basic variable is that with the most negative reduced cost), Hung [104] obtained an algorithm which performs at most $O(n^3 log\Delta)$ pivots, where Δ is the difference between the objective function value of the starting solution with the optimal one. Orlin [132] reduced the number of pivots to $O(n^3 \log n)$ for the primal network simplex algorithm with Dantzig's rule. He exploits the fact that employing strongly feasible trees is equivalent to running the standard network simplex algorithm with some lexicographic pivoting rule for an appropriately perturbed problem. (This fact was also observed by Cunningham [62].) By scaling the coefficients of the perturbed problem, an upper bound of $O(n^2 \log \Delta)$ on the number of pivots is obtained. In the case of LSAPs can be shown that there exists an equivalent problem with cost coefficients bounded by $4^n(n!)^2$, and hence $\Delta = O(n \log n)$.
Another primal simplex algorithm which uses a pivot rule similar to Dantzig's was proposed by Ahuja and Orlin [4]. Assuming that the cost coefficients are integral, the algorithm performs at most $O(n^2 \log R)$ pivots and has a worst case complexity of $O(n^3 \log R)$, where R is defined as $R = \max\{|c_{ij}|: 1 \leq i,j \leq n\}$. The pivot rule involves scaling, and a pivot edge is chosen among those edges (i,j) for which $\bar{c}_{ij} \leq -0.5\epsilon$. If at any stage in the algorithm there does not exist a pair (i,j) which fulfills $\bar{c}_{ij} \leq -0.5\epsilon$, then ϵ is divided by 2. This process is continued until ϵ becomes less than 1. At this point an optimal solution is reached.

Based on strongly feasible trees Roohy-Laleh [154] gave a strongly polynomial primal network-simplex algorithm which performs at most $O(n^3)$ pivots and has a worst-case complexity of $O(n^5)$.

Akgül [7] designed a primal simplex algorithm which performs $O(n^2)$ pivots and has an overall worst case complexity of $O(n^3)$. Also this algorithm uses strongly feasible trees as basic solutions and Dantzig's pivot rule to choose the variable which enters the basis. A special feature of the algorithm is that it solves an increasing sequence of LSAPs which are subproblems of the original one, in the sense that the edge sets of the corresponding graphs are subsets of the edge set E for the original problem. The edge set associated with a certain subproblem is obtained by adding some edges incident to a single node to the edge set associated with the previous subproblem. The last subproblem in the sequence is the original LSAP. The author investigates the structure and properties of so-called *cut sets* which are sets of vertices whose dual variables are changed during the solution of a subproblem. It can be shown that the cut sets are disjoint, and that edges whose heads lie in the cut set are dual feasible for all subgraphs considered afterwards. By exploiting these properties it is shown that the optimal solution of the $(i+1)$-st subproblem can be found by performing at most $k+2$ pivots starting from an optimal solution of the i-th subproblem.

Dual simplex-based algorithms

A spanning tree T of the graph G, besides determining uniquely an integer valued basic solution for the LSAP, determines also a basic solution of the dual (7) of the LSAP. This is done by setting the value of one of the variables arbitrarily, say $u_1 = 0$, and the other variables so as to fulfill $u_i + v_j = c_{ij}$ for all i, j such that $(i, j) \in T$. This dual solution is unique up to the arbitrary setting of one of the variables. Moreover, this basic dual solution is feasible iff $\bar{c}_{ij} = c_{ij} - u_i - v_j \geq 0$. Clearly, if both primal and dual solutions determined by T are feasible, then both of them are optimal.
Dual simplex methods for the LSAP work with a tree T which defines a dual feasible basic solution and keep transforming T iteratively until the corresponding primal solution is feasible, and hence, optimal. The transformation is done by pivoting on an edge $e = (k, l)$ of $E \setminus T$ for which the corresponding primal constraint is violated, i.e., $x_{kl} \leq -1$. This edge is added to the current tree T and one of the edges, say f, contained in the cycle $C(T, e)$ is removed from T to form a new tree $T' = (T \setminus \{f\}) \cup \{e\}$. In order to keep dual feasibility the chosen edge $f = (i, j)$ should fulfill $i \in T^{(k)}$, $j \in T^{(l)}$, where $T^{(k)}$, $T^{(l)}$ are the subtrees of T which arise when the edge $e = (k, l)$ is removed from T, and contain k and l, respectively.

Different rules for the choice of the edges e and f yield different dual simplex algorithms. Many of these algorithms make use of so-called *signatures*, introduced by Balinski [18]. Therefore, such algorithms are also called *signature methods*.

Definition 3.4 *Consider an LSAP and a spanning tree T in the graph $G = (V, W; E)$ related to the LSAP. The* row *(*column*)* signature *of T is the vector of the degrees in T of the vertices in V (W) for some arbitrary but fixed order of those vertices.*

It can be shown that a spanning tree T which defines a dual feasible basis and whose row signature contains only one 1 and the other entries equal to 2 is an optimal basis. Consequently, the primal solution defined as follows is optimal. Start with x_{kl}=1 for $k \in V$ being the node of degree 1 and $(k, l) \in T$. Set the other variables x_{ij} such that the assignment constraints are fulfilled and $x_{ij} = 0$ for $(i, j) \notin T$. Thus the LSAP is reduced to finding a dual feasible tree with row (column) signature $(1, 2, 2, \ldots, 2)$. Balinski [18, 19] proposes two algorithms to find such a tree T. Both algorithms start with a tree with row signature $(n, 1, \ldots, 1)$ and dual variables $u_1 = 0$, $v_j = c_{1j}$ for $1 \leq j \leq n$, and $u_i =$

$\min_j\{c_{ij} - v_j\}$, for $2 \leq i \leq n$. Such a tree is often called a *Balinski tree*. In the first algorithm [18] the choice of the *pivoting edges* e and f is guided only by the signatures, trying to decrease the number of 1-s in the signature. This makes the algorithm a non-genuine simplex method because it may pivot on some edge $e = (k, l)$ with $x_{kl} = 0$. Moreover, the algorithm does not stop when an optimal solution with signature different from $(1, 2, \ldots, 2)$ is found. The algorithm performs at most $O(n^2)$ operations to decrease the number of 1-s in the signature by 1. Hence its worst case complexity is $O(n^3)$.
Goldfarb [97] proposed a similar algorithm based on signatures which solves a sequence of subproblems of the given LSAP. As in the algorithm of Akgül [7] the subproblems are LSAPs with cost matrices being submatrices of $C = (c_{ij})$. The cost matrix of the k-th subproblem is defined by the first k rows and the first k columns of C. Balinski's algorithm is used to solve the k-th subproblem, i.e., to produce a tree corresponding to a dual feasible basis with signature $(2, \ldots, 2, 1)$. After the transition to the $(k + 1)$-st subproblem, i.e., after the introduction of the $(k + 1)$-st row and column of matrix C, the signature either remains optimal or has the form $(3, 2, \ldots, 2, 1, 1)$. In the latter case at most $k - 1$ pivots are needed to produce a tree with signature $(2, \ldots, 2, 1)$ for the $(k+1)$-st problem. Clearly, this algorithm is not a genuine simplex dual algorithm for the same reasons as Balinski's algorithm.
The second algorithm of Balinski [19] employs another pivot rule which is a combination of the rule based on signatures used in [18] and the classical pivot on an edge $e = (i, j)$ with the most negative x_{ij}. This rule allows the algorithm to visit only so-called *strongly dual feasible trees*. These trees are defined in terms of so-called odd and even arcs. Consider a tree T corresponding to a dual feasible solution and root it at the first vertex in V (according to some fixed arbitrary order) and direct all edges away from the root. An arc $(i, j) \in T$ with $i \in V$, $j \in W$, is called *odd arc* and an arc (j, i) with $j \in W$ and $i \in V$ is called *even arc*. A dual feasible tree is strongly dual feasible if $x_{ij} \geq 1$ for all even arcs $(i, j) \in T$, and $x_{ij} \leq 0$ for all odd arcs $(i, j) \in T$ which do not join $1 = i \in V$ with a node j of degree 1 in W. The algorithm proposed in [19] pivots only on edges (i, j) for which $x_{ij} < 0$. Hence, this algorithm is a genuine dual simplex method, sometimes also called *competitive simplex algorithm*. The number of pivots is at most $(n - 1)(n - 2)/2$, thus yielding a worst case time complexity of $O(n^3)$.
Kleinschmidt, Lee, and Schannath [114] have shown that this algorithm of Balinski [19] is equivalent to an algorithm proposed earlier by Hung and Rom [105].

Akgül [5] proposed an algorithm similar to Goldfarb's. The two main differences are: 1) Akgül's algorithm works with *strongly* dual feasible trees, i.e., it finds a strongly dual feasible tree for the k-th problem and transforms it into a strongly dual feasible tree for the $(k + 1)$-st problem, and 2) the subproblems considered by Akgül are transportation problems. The algorithm performs at most $O(n^2)$ pivots and has a worst case complexity of $O(n^4)$.
A similar algorithm was developed by Paparrizos [136]. The main difference to Akgül's algorithm is that here the subproblems are determined with the help of a Balinski tree. They are, however, solved in the same way as in [5].

Two other signature-guided algorithms were given by Paparrizos [135] and Akgül and Ekin [8]. Paparrizos' algorithm is similar to Balinski's competitive simplex algorithm [19]. The main difference concerns the dual feasibility which is not an invariant of Paparrizos' algorithm. The algorithm starts with a Balinski tree. Then it performs pivots which may produce infeasible dual solutions. The pivots are grouped in stages and each stage

starts (and stops) with a strongly dual feasible tree. The last tree of the last stage is also primal feasible and hence optimal. The algorithm visits only so-called *strong trees* which are obtained from strongly feasible trees by dropping the feasibility requirement. The algorithm performs $O(n^2)$ pivots but spends as much time to determine the pair of edges involved in a pivot. Thus the overall complexity is $O(n^4)$. Akgül and Ekin [8] give an efficient implementation of Paparrizos' algorithm, which maintains and updates a forest rather than a single tree. In this way only dual feasible trees are visited. The number of pivots is again $O(n^2)$ but the overall complexity becomes $O(n^3)$.
Later, Paparrizos [137] designed a *purely infeasible* simplex algorithm which performs $O(n^2)$ pivots and has worst case complexity of $O(n^3)$. This algorithm is similar to the algorithm presented in [135] since it starts again with a Balinski tree and visits only strong trees. Also the edge inserted to a the current strong tree during a pivot operation is determined as in [135]. The edge which is removed from the tree is determined, however, by a different rule. In contrast with the previous algorithm proposed in [135], this algorithm is purely infeasible in the sense that it restores dual feasibility only when it stops with an optimal solution.

3.3 Other Algorithms

A primal algorithm of Balinski and Gomory [20]

The algorithm of Balinski and Gomory is probably the oldest primal algorithm for the LSAP. It works with a perfect matching $\mathcal{M}$ in the graph G (i.e., a primal feasible solution x_{ij}, $1 \leq i, j \leq n$) and dual variables u_i, v_j, $1 \leq i, j \leq n$, which fulfill the complementary slackness conditions. The primal and dual solutions are updated iteratively by maintaining thereby the complementarity slackness conditions. Let $\tilde{E} = \{(i,j) \colon \bar{c}_{ij} = c_{ij} - u_i - v_j \geq 0\}$. If $\tilde{E} = E$ than the primal solution $\mathcal{M}$ and the dual solution u_i, v_j, $1 \leq i, j \leq n$, are optimal. If there is an edge $e = (i,j) \notin \tilde{E}$, then there exists an edge $(r,j) \in \mathcal{M}$. The algorithm builds an alternating tree T in the subgraph $(V, W; \tilde{E})$ of G and grows it by updating the dual variables similarly as in the primal-dual methods. Then, the matching is changed if there is a cycle in $T \cup \{e\}$. This procedure is repeated until $E = \tilde{E}$. A characteristic feature of the algorithm is that $\tilde{E}$ grows monotonically in the course of the algorithm, i.e., if an edge becomes dual feasible at a certain iteration it remains feasible in the subsequent iterations until the algorithm stops. An efficient version of this algorithm with time complexity $O(n^3)$ has been described by Cunningham and Marsh [64].

Cost Operator Algorithms

Two other primal algorithms for the transportation problem, and hence also for the LSAP, are the cost operator algorithms developed by Srinivasan and Thompson [158]. In contrast to the algorithm of Balinski and Gomory these algorithms work with primal basic solutions. Both algorithms start with a dual feasible solution u_i, v_j, $1 \leq i, j \leq n$, found by the usual row/column reductions: compute the column minima, subtract them from the columns, then compute the row minima and subtract them from the rows. Then both algorithms construct a primal feasible basic solution x_{ij} (x_{ij} are reals between 0 and 1), and introduce new costs: $\tilde{c}_{ij} = u_i + v_j$ for all i, j such that x_{ij} is in the basic solution, and $\tilde{c}_{ij} = c_{ij}$, otherwise. Thus, x_{ij} and u_i, v_j form a pair of feasible primal and dual solutions for the LSAP with cost matrix $(\tilde{c}_{ij})$. This pair of solutions fulfills the complementarity slackness conditions. Hence both solutions are optimal. Next both algorithms change step by step

the costs $\tilde{c}_{ij}$ back to c_{ij}. In each step, the primal and dual solutions are updated so as to remain optimal for the problem considered at that step. One algorithm employs the *cell cost operator* and changes c_{ij} for *one* pair (i, j) at a step, whereas the other algorithm employs the *area cost operator* and changes the costs c_{ij} for a set of pairs (i, j) at a step. Both algorithms maintain strongly feasible trees for all intermediate problems, as pointed out by Akgül [6]. The cell operator algorithm performs one dual pivot (changing the dual variables) and one primal pivot (changing the basic solution) at each step. Nevertheless this is not a purely pivotal algorithm because the primal and the dual solution do not always fulfill the complementary slackness conditions. Each step performs $O(n)$ pivots. The total number of pivots is $O(n^2)$.
The area cost operator uses similar dual and primal pivots and tries to restore as many c_{ij} as possible, at a step. This algorithm performs also $O(n^2)$ pivots. Akgül [6] conjectures that both algorithms can be implemented so as to achieve a time complexity of $O(n^3)$.

A recursive algorithm

Thompson [160] proposed an $O(n^3)$ algorithm which solves a sequence of transportation problems the last of which is the given LSAP. Given an LSAP of size n with cost matrix $C = (c_{ij})$, the algorithm introduces an $(n+1)$-st dummy vertex in V (dummy row in C) and an $(n + 1)$-st dummy vertex in W (dummy columns in C). All edges adjacent to dummy nodes have costs equal to 0, $c_{n+1,j} = c_{i,n+1} = 0$, $1 \leq i, j \leq n + 1$. Obviously, the problem of minimizing $\sum_{i,j=1}^{n+1} c_{ij}x_{ij}$ subject to $\sum_j x_{ij} = 1$, if $i \leq n$, and $\sum_j x_{ij} = 0$, if $i = n + 1$, $\sum_i x_{ij} = 1$, if $j \leq n$, and $\sum_i x_{ij} = 0$, if $j = n + 1$, is equivalent to the original LSAP.
The first subproblem solved is the trivial transportation problem defined by the last row of the extended matrix C with supply equal to n at vertex $i = n + 1$ in V, demand equal to 1 for all vertices $j \leq n$ in W, and demand equal to 0 for $j = n + 1$. The k-th problem is a transportation problem defined by the $k - 1$ first rows and the $(n + 1)$-st row of the extended matrix C, where the supply of the first $k - 1$ vertices in V is equal to 1 and that of the $(n + 1)$-st vertex is equal to $n - k$. The demands of vertices in W are as for the first subproblem, and they remain the same for all subproblems. The relaxation algorithm proceeds as follows: it produces a primal optimal solution for the current problem and uses it to find an optimal dual solution for the current problem. This is used to produce a primal optimal solution of the next problem in sequence, and so on. A specific feature of the algorithm is that the dual variables change monotonically: the u variables do not increase, whereas the v variables do not decrease. Another feature is that every primal or dual feasible solution found for some subproblem is also optimal. The algorithm can be implemented to achieve an $O(n^3)$ worst case complexity. For more information we refer to [160].

Forest algorithms

We saw above that primal (dual) simplex-based algorithms for the LSAP work with primal (dual) feasible bases which correspond to spanning trees in bipartite graphs.
Forest algorithms work with forests in G instead of trees. Feasible (primal or dual) forests are defined analogously as feasible (primal or dual) trees. A forest is primal feasible, if there exists a primal feasible solution with $x_{ij} = 0$ for all i, j such that (i, j) is not an edge of the forest. Analogously, a forest is dual feasible, if there exists a dual feasible solution u_i, v_j, such that $\bar{c}_{ij} = c_{ij} - u_i - v_j = 0$, whenever (i, j) is an edge of the forest. Unlike trees, forests do not determine (primal or dual) bases of the problem.

A primal-dual forest algorithm, or more precisely a forest implementation of the Hungarian method, has been described by Akgül in [6]. The algorithm iteratively solves a shortest path problem in the graph G associated with the assignment problem, and equipped with the reduced costs $\bar{c}_{ij}$ as edge lengths. This is done by growing a forest of alternating trees rooted at a free node in V each and by exploiting the relationship between alternating trees and strongly feasible trees. As soon as the leafs of an alternating tree are joined by edges with free nodes in W, a strongly feasible tree arises. The algorithm maintains a set of alternating trees called *planted forest*, and a set of strongly feasible trees called *matched forest*. Each augmentation transforms an alternating tree in a strongly feasible tree, until the planted forest becomes empty. The augmentations employ labeling and update of dual variables like all primal-dual algorithms in general, and the Hungarian method in particular.

An $O(n^3)$ forest algorithm is proposed by Achatz, Kleinschmidt, and Paparrizos [1] and is a more efficient variant of the algorithm proposed by Paparrizos in [137]. The algorithm starts with some basic dual feasible tree rooted at some arbitrary node in V, and constructs a forest by deleting all edges (i, j) where $i \in V$, $j \in W$, and i is the father of j with respect to the usual orientation of the tree which directs all edges away from the root. This yields a spanning forest which is dual feasible. All nodes in W are considered as roots of the connected components to which they belong. Further, the algorithm works with *superforests* which are forests where every connected component contains one root, and nodes from W, which are not roots, have degree equal to 2. The starting forest constructed as above is a *superforest*. Each superforest is divided into a *surplus forest* and a *deficit forest*. The surplus forest contains all connected components whose roots have degree larger than 1. The deficit forest is the complement of the surplus forest with respect to the considered superforest. The algorithm constructs a sequence of dual feasible superforests by applying a special pivoting strategy. It stops when a superforest with an empty surplus forest is found. It can be shown that this yields a dual optimal solution of the considered LSAP.

3.4 On Parallel Algorithms

Since the late 1980s a number of parallel algorithms for the LSAP has been proposed. They are parallelized versions of primal-dual algorithms based on shortest path computations, due to Kennington and Wang [113], Zaki [170], Balas, Miller, Pekny, and Toth [14], Bertsekas and Castañon [32, 33], parallelized versions of the auction algorithm, due to Zaki [170], Wein and Zenios [167, 168], Bertsekas and Castañon [31], and parallelizations of primal simplex-based methods, due to Miller, Pekny, and Thompson [128], Peters [138], Barr and Hickman [27]. For a good review on parallel algorithms for the LSAP and network flow problems in general the reader is referred to Bertsekas, Castañon, Eckstein, and Zenios [34].

Parallel implementations of primal-dual shortest path algorithms

The primal-dual parallel algorithms based on shortest path computations can be classified into *single node algorithms* and *multi-node algorithms*. Single node algorithms compute a shortest path tree from a *single* free source node in parallel, i.e., by parallelly scanning the arcs connected to a single node. Most of these algorithms are obtained as implementations of the algorithm of Jonker and Volgenant [107]. As shown in [113, 170], considerable speedups can be obtained in the case of dense assignment problems. Kennington et al. [113] obtain a speedup of 3–4 for a number of processors between 3 and 8. For sparse problems the

speedups decrease, since the amount of work which can be performed in parallel decreases. Thus, the effectiveness of these algorithms is limited by the density of the input network. Computational results, e.g. Castañon, Smith, and Wilson [53], show that *single instruction stream multiple data stream* (SIMD) shared memory massively parallel architectures are well suited for implementing single node parallel algorithms to solve dense problems.

Multi-node approaches compute multiple augmenting paths starting from different free nodes, and perform multiple augmentations and updates of variables in parallel. Balas et al. [14] propose a synchronous algorithm of this type which employs a coordination protocol to combine all augmentations and changes of the dual variables so as to maintain feasibility and keep the complementarity slackness conditions fulfilled. Bertsekas et al. [32] extend this protocol to obtain asynchronous algorithms. The coordination in the asynchronous algorithm is realized by maintaining a master copy of a pair of primal and dual solutions which can be accessed by all processors. The latter compute augmenting paths and variable updates based on the copy of the primal and dual solution they have, and then check whether the update is feasible (the master copy may have been updated in the meanwhile). If the update is feasible, the master copy is updated. The master copy is locked during the updates. Computational experience with these algorithms shows that their principal limitation is the decreasing amount of parallel work with the number of iterations. Initially many augmenting paths can be found so that many processors can be used effectively. Then the load decreases as the number of iterations increases. When comparing synchronous and asynchronous implementations Bertsekas et al. [32] show that for sparse problems asynchronous algorithms may be outperformed by synchronous algorithms. This is basically due to the considerable coordination overhead of the asynchronous algorithms.

Parallel implementations of the auction algorithm

Recall that the auction algorithm computes in each iteration the smallest and the second smallest cost margin with respect to the profit for each free customer. Different implementations may compute these differences (also called *bids* in the maximization version of the LSAP) for all free customers - resulting in the so-called *Jacobi methods*, for a single free customer - resulting in so-called *Gauss-Seidel methods*, or for a subset of free customers - resulting in so-called *block Gauss–Seidel methods*. Here again, we have single node and multi-node algorithms. The multi-node algorithms compute the bids for different customers in parallel, and this is appropriate for Gauss-Seidel and block Gauss-Seidel methods. Single node algorithms compute only one bid at a time but perform this computation in parallel. There are also hybrid approaches where multiple bids are computed in parallel and the computation of each bid is done in parallel by several processors. Hybrid approaches yield the maximum speedup, if the number of processors assigned to different tasks is chosen properly. Bertsekas and Castañon [31] have proposed a hybrid implementation of the auction algorithm which converges to an optimal solution even in the case of a fully asynchronous implementation.
Computational results of Phillips and Zenios [141] and Wein and Zenios [167] show that massively parallel architectures are appropriate for the implementation of the above mentioned hybrid approach. The obtained speedup is often larger than that achieved by primal-dual shortest path algorithms. Bertsekas and Castañon [31] show that their hybrid approach can be implemented efficiently on SIMD architectures. These authors also compare synchronous versus asynchronous algorithms and show that in the case of auction algorithms, asynchronous implementations really pay off.

Parallel implementations of primal simplex-based algorithms

Most of the computation load of primal simplex-based algorithms concerns the computation of the new edge to be included in the basic solution. It would be desirable to include an edge which violates dual feasibility most, but this requires the computation of the reduced costs for all edges. Many algorithms compute instead a list of candidate edges to enter the basis and choose among them the edge which violates dual feasibility at most.
Most of parallel primal simplex-based algorithms compute this list of candidates in parallel. One possibility is to assign to each processor a set of rows to investigate. Each processor chooses a pivot row. These rows form a list of candidate edges for pivoting. Then, another processor performs pivot steps sequentially. This yields a synchronous algorithm. For the transportation problem such an algorithm has been proposed by Miller at al. [128]. The computational results of Miller et al. show that such procedures yield a good speedup (3–7.5, if 14 processors are employed), when applied to dense problems and implemented on a small-scale coarse-grain parallel system with distributed memory.
Another approach would be to search for edges to enter the basis in parallel, while pivot operations are taking place, resulting in an asynchronous algorithm. Such an algorithm was proposed by Peters [138]. The convergence of the algorithm is guaranteed by the so-called *pivot processor* which performs a pivot only after having checked whether the pivot is feasible. It is necessary to check feasibility because the processors searching for pivot edges do their job while pivots are taking place, and hence, their computations may be based on out-of-date information. Computational experience on a shared memory machine shows again a good speedup which increases with the size of the problem. Peter's results also show that it pays off to increase the number of processors when the size of the problems increases. The time needed to perform a pivot operation induces, however, limitations to the number of processors which may be used efficiently.
A refinement of the algorithm of Peters was proposed by Barr and Hickman [27]. These authors give an implementation of the algorithm of Barr, Glover, and Klingman [26]. This implementation does not distinguish between search and pivot processors, all processors may perform both tasks. The algorithm maintains a queue of tasks and each processors refers to the queue to receive its task. Also the pivot operations are divided into so-called *basis update* and *flow update* operations, and different groups of operations are performed by different processors in parallel. Computational results show that this algorithm is faster than that of Peters. This is mainly due to the efficient use of the processors through the task assignment approach, but also to the splitting of a long pivot task among two different processors.

4 Asymptotic Behavior and Probabilistic Analysis

When dealing with the asymptotic behavior of the LSAP, it is always assumed that the cost coefficients c_{ij} are independent random variables with a common prespecified distribution. The main question concerns the behavior of the expected optimal value of the problem as its size tends to infinity.
Let us first consider the question whether a random bipartite graph admits a perfect matching. A class of bipartite random graphs $(V, W; E)$ can be described by specifying the number of vertices and the probabilities for the existence of each edge. The existence of a perfect matching in such a graph is intuitively related to the expected number of edges. Erdös and Rényi [74] have shown that $O(n \log n)$ edges do not suffice for the existence of a perfect

matching. More precisely, for each $n \in \mathbb{N}$ the authors introduce a class of random bipartite graphs $\mathcal{G}(n) = (V(n), W(n); E(n))$ with $|V(n)| = |W(n)| = n$ and $|E(n)| = O(n \log n)$, and show that the probability that a graph chosen uniformly at random from $\mathcal{G}(n)$ contains a perfect matching tends to 0 as n tends to infinity. The reason for this behavior is the growing probability of the existence of isolated vertices in such graphs as n tends to infinity.

Another scenario is obtained, if a probabilistic model is chosen which avoids isolated vertices. Consider the class $\mathcal{G}(n, d)$ of *directed* bipartite graphs with n vertices in each part, where each vertex has out-degree d, and denote by $P(n, d)$ the probability that a graph chosen uniformly at random from $\mathcal{G}(n, d)$ contains a perfect matching. (Matchings and perfect matchings in directed bipartite graphs are defined as matchings and perfect matchings of the graphs obtained from the digraphs by removing the orientation of the edges, respectively.) Walkup [165] shows that $P(n, 1)$ tends to 0, but for all $d \geq 2$, $P(n, d)$ tends to 1 as n approaches infinity. More precisely

$$P(n,1) \leq 3n^{1/2}(\frac{2}{e})^n\,, \quad P(n,2) \geq 1 - \frac{1}{5n}\,, \quad \text{and}$$

$$P(n,d) \geq 1 - \frac{1}{122}(\frac{d}{n})^{(d+1)(d-2)} \text{ for all } d \geq 3$$

Notice that due to the marriage theorem the existence of a perfect matching in an undirected bipartite graph regular of degree d is trivial.

The above lower bounds on $P(n, d)$ are used by Walkup [166] to show that 3 is an upper bound on the expected optimal value of the LSAP in the case that the cost coefficients c_{ij} are independent random variables uniformly distributed on $[0, 1]$. The idea is to consider sparse subgraphs of the graph G related to the LSAP as described in Section 1. The considered subgraphs are regular of degree d and have edges with small weights. Choosing d edges with the smallest weights could create problems since the chosen edges would not be independent. To avoid this difficulty Walkup replaces each edge by two directed edges with opposite directions, and chooses their weights from distributions such that the minimum of the two weights still has uniform distribution on $[0, 1]$. More precisely, c_{ij} can be obtained as $\min\{a_{ij}, b_{ij}\}$, where a_{ij}, b_{ij} are independent random variables with a common distribution function F given by $F(\lambda) = 1 - (1 - \lambda)^{0.5}$. Then, with the help of a_{ij} and b_{ij} graphs from the classes $\mathcal{G}(n, d)$, $1 \leq d \leq n$, are associated with the $n \times n$ cost matrix $C = (c_{ij})$, and the existence results described above are applied to evaluate the expected value of $\min_\phi \sum_i c_{i\phi(i)}$ by means of conditional probabilities.

Four years later the upper bound of 3 was improved to 2 by Karp [111]. The proofs for both bounds are non-constructive and do not lead to heuristics which produce assignments with expected optimal value within the given bounds.

Lower bounds for the expected optimal value of the LSAP with independent and uniformly distributed costs c_{ij} on $[0, 1]$ are given by Lazarus [120]. The author exploits the weak duality and evaluates the expected value of the dual objective function $\sum_i u_i + \sum_j v_j$ achieved after the first iteration of the Hungarian method (row and column reductions, see also Section 3.1). By computations involving first order statistics it is shown that the expected value of $\sum_i u_i + \sum_j v_j$ - which is a lower bound for the expected optimal value of the LSAP - is of order $1 + \frac{1}{e} + \frac{\log n}{n}$. This yields a bound of 1.368. Moreover, Lazarus evaluates the maximum number of 0-s lying in different rows and columns of the reduced cost matrix after row and column reductions (or, equivalently, after the first iteration of the Hungarian method). This evaluation implies that the probability of finding an optimal assignment after row/column reductions tends to 0 as n tends to infinity.

The lower bound was improved by Goemans and Kodilian [93] to 1.44, and then to 1.51 by Olin [131]. 1.51 is currently the best known value. Goemans and Kodilian propose a dual heuristic for the LSAP and show that the expected objective function value yielded by this heuristic is larger than 1.44. The heuristic starts with the dual solution obtained by applying row/column reductions as in the first iteration of the Hungarian method. Then, this solution is improved by elementary operations in the fashion of the Hungarian method. The analysis of the heuristic builds upon the analysis of Lazarus and essentially exploits the fact that the solution obtained after the first iteration of the Hungarian method is with high probability highly structured.
Also the best known bound given by Olin [131] is obtained from a feasible solution of the dual. Again, Olin starts with the solution of the dual obtained by row and column reductions, after the first iteration of the Hungarian method. Then, this solution is improved by adding the second smallest element of each row to all elements in that row and by subtracting to each column the largest among the terms added to the elements of that column. This transformation leads to a new feasible dual solution which yields an expected value of the objective function equal to 1.47. By applying then an analogous transformation starting from the columns of the reduced cost matrix, the bound of 1.51 is obtained.

Summarizing, the gap between the lower and the upper bound on the expected optimal value of the LSAP is still large. In the case of independent costs c_{ij} uniformly distributed in $[0,1]$, it is believed that the expected value is close to 1.6 as observed by Donath [72] in his experiments, or more precisely, equal to $\frac{\pi^2}{6} - o(1)(\frac{\pi^2}{6} - 1.645)$ as suggested by Mezard and Parisi [127].

Frenk, Houweninge, and Rinnooy Kan [81] as well as Olin [131] have studied the more general situation of an arbitrary distribution function for the cost elements c_{ij}. Frenk et al. analyze the asymptotic behavior of the first order statistics in the case of distribution functions F defined on $(-\infty, +\infty)$ ($\lim_{n\to\infty} F^{-1}(1/n) = -\infty$, F^{-1} being the inverse of function F) which fulfill the conditions

$$\int_{-\infty}^{+\infty} |x|F(x)dx < \infty \quad \text{and} \quad \liminf_{x\to+\infty} \frac{F(-x)}{F(-ax)} > 1 \quad \text{for some } a > 1\,,$$

as well as for functions F defined on $(0,\infty)$ ($\lim_{n\to\infty} F^{-1}(1/n) = 0$). The estimates on the first order statistics are then used to evaluate the expected optimal value of the LSAP along the ideas of Walkup [166]. In the case of functions F defined on $(-\infty, +\infty)$ the authors show the following relation for the expectation EZ_n of the optimal value Z_n of the LSAP of size n

$$\left(1 - \frac{3}{2\sqrt{e}}\right)^2 \leq \liminf_{n\to\infty} \frac{EZ_n}{nF^{-1}(1/n)} \leq \limsup_{n\to\infty} \frac{EZ_n}{nF^{-1}(1/n)} < \infty\,. \tag{14}$$

In the case that F is defined on $(0,\infty)$ an analogous result holds with a lower bound equal to 0 on the left hand-side of (14). In this second case, Olin [131] imposes further conditions on F and derives specific bounds which generalize those of Walkup and Lazarus. More precisely, if (i) F admits a continuous density function which is strictly positive in a neighborhood of the origin, (ii) F has finite expected value, and (iii) $F^{-1}(0^+) := \lim_{y\to 0^+} F^{-1}(y)$ exists, then

$$(1 + e^{-1})F^{-1}(0^+) \leq \liminf_{n\to\infty} EZ_n \leq \limsup_{n\to\infty} EZ_n \leq 3F^{-1}(0^+)\,.$$

It seems difficult to derive analogous results which hold almost surely under the above mild conditions on F. However, for special distributions e.g. the uniform distribution,

almost sure results can be derived. Olin shows that in this case almost surely the lower bound equals $1 + e^{-1}$ and the upper bound equals 4. This result generalizes to distribution functions F defined on $(0, +\infty)$ and fulfilling the stronger set of conditions posed by Olin. In this case the following inequalities hold with probability 1:

$$\frac{1 + e^{-1}}{f(0^+)} \leq \liminf_{n\to\infty} Z_n \leq \limsup_{n\to\infty} Z_n \leq \frac{4}{f(0^+)} .$$

Let us now turn to the probabilistic analysis of heuristics for the LSAP. One of the first results in this area was obtained by Karp [110] and provides an exact algorithm for the LSAP with expected running time $O(n^2 \log n)$ in the case of independent costs c_{ij} uniformly distributed on $[0, 1]$. The proposed algorithm - which is faster than the best known sequential algorithm - is a special implementation of the standard shortest augmenting path algorithm for the assignment problem (see Section 3). It uses priority queues to compute a shortest augmenting path in $O(n^2 \log n)$ time and has thus a worst case time complexity of $O(n^3 \log n)$. The algorithm tries to reduce the number of insertions in the priority queue at the cost of a slight increase of the number of deletions. This is done by so-called *surrogate items*. Each surrogate item replaces a large number of regular items of the priority queue, and is used instead of them. In the case of the specific distribution of the costs c_{ij} the surrogate items do indeed represent and replace many regular items, reducing the expected overall number of operations associated with the queue to $O(n^2)$. Considering that each of the queue-operations takes $O(\log n)$ times and that the time for all other operations is $O(n^2)$, we obtain the above mentioned time complexity.

Other results in this area concern *heuristics* and provide worst case bounds and/or average case bounds in the case that the costs c_{ij} are independently and unformly distributed on $[0, 1]$. Avis and Devroye [11] analyze a heuristic for the LSAP proposed by Kurzberg [117]. This heuristic decomposes a large problem of size $n = mk$, into k^2 problems of size $m \times m$ each, solves the smaller problems, and combines their solutions to obtain a solution for the original problem. In its space-optimal version ($k = \sqrt{n}$) the heuristic takes $O(n)$ space and $O(n^{2.5})$ time. In its time-optimal version ($k = n^{3/4}$) it takes $O(n^{2.25})$ time and $O(n^{1.5})$ space. It is not difficult to show that the worst case ratio of this heuristic is ∞. (By applying this heuristic to maximize the objective function of the LSAP, one can obtain a worst case ratio of $\min\{m, k\}$.) However, in the case of cost coefficients c_{ij} uniformly distributed on $[0, 1]$, the expected value of the solution produced by the heuristic gets smaller than or equal to $k/2$ times the expected optimal value, as n tends to infinity.

The first heuristic which produced a solution with expected value within a constant factor from the optimum value, was proposed by Avis and Lai [12]. This is an $O(n^2)$ algorithm which provides a solution within a factor of 4 from the optimal one (for $n \geq 11$), in the case of the specific distribution of the costs c_{ij}. The heuristic elaborates the idea of Walkup [166] to work with a sparse subgraph of the given graph. The sparse subgraph considered by the authors has $10n$ "cheap" edges and contains a perfect matching with high probability. If necessary, the largest matching contained in this graph is completed to a perfect matching of the original graph in a greedy way. The good behavior of the heuristic relies on the fact that the sparse subgraph contains with high probability a cheap perfect matching, i.e., a good solution of the LSAP instance.

Karp, Rinnooy Kan, and Vohra [112] derive a better heuristic which runs in $O(n \log n)$ time (expected time $O(n)$) and provides a solution of the LSAP whose value is smaller than $3 + O(n^{-a})$, for some $a > 0$, with probability $1 - O(n^{-a})$. The basic idea is similar to that

of Avis and Lai: Construct a "cheap" sparse subgraph of the given graph and show that it contains a perfect matching with high probability. Again, if the subgraph does not contain a perfect matching a solution for the original LSAP instance is determined in a greedy way. The subgraph is introduced in terms of so-called *random 2-out* bipartite graphs. Such a graph contains a perfect matching with probability $1 - O(n^{-a})$ and the matching can be found in $O(n \log n)$ time. Further, it is shown how to construct a random 2-out bipartite subgraph with cheap edges for the given LSAP instance. The expected value of a solution obtained either as a perfect matching in this subgraph, or by applying a greedy approach, if the former does not exist, equals $3 + O(n^{-a})$.

5 Bottleneck Assignment Problems

As mentioned in Section 2 linear bottleneck assignment problems (LBAP)

$$\min_{\phi} \max_{1 \le i \le n} c_{i\phi(i)} \tag{15}$$

occur in connection with assigning jobs to parallel machines so as to minimize the latest completion time. The first authors who considered bottleneck assignment problems were Fulkerson, Glicksberg and Gross [84]. Gross [98] proved the following duality result for LBAPs which gave rise to the theory of blocking systems, see Edmonds and Fulkerson [73].

Theorem 5.1 (Gross [98], 1959)
Let $N = \{1, 2, \ldots, n\}$ and let $\mathcal{S}_n$ be the set of all permutations ϕ of N. Then, for an arbitrary $n \times n$ matrix $C = (c_{ij})$ of real numbers

$$\min_{\phi \in \mathcal{S}_n} \max_{i \in N} c_{i\phi(i)} = \max_{\substack{A,B \subseteq N \\ |A|+|B|=n+1}} \min_{i \in A, j \in B} c_{ij}. \tag{16}$$

So-called *threshold algorithms* play an important role to solve LBAPs. A threshold algorithm alternates between two phases. In the first phase a cost element c^*_{ij} - the *threshold value* - is chosen and a matrix $\bar{C}$ is defined by

$$\bar{c}_{ij} := \begin{cases} 1 & \text{if} \quad c_{ij} > c^*_{ij} \\ 0 & \text{if} \quad c_{ij} \le c^*_{ij} \end{cases}$$

In the second phase it is checked, whether the bipartite graph with adjacency matrix $\bar{C}$ contains a perfect matching or not. The smallest value c^*_{ij} for which the corresponding bipartite graph contains a perfect matching, is the optimum value of the LBAP (15).
There are several ways to implement such a threshold algorithm. One possibility is to order the cost elements increasingly and to apply a binary search, in the first phase. This leads to an $O(T(n) \log n)$ algorithm, where $T(n)$ is the time complexity for checking the existence of a perfect matching. Concerning $T(n)$ see the remarks after Theorem 1.3 in Section 1.
Another possibility is to mimic the Hungarian method. We start with

$$c^* := \max_{i,j} (\min_i c_{ij}, \min_j c_{ij}) \tag{17}$$

and grow bottleneck augmenting paths as long as the matrix $\bar{C}$ does not contain an assignment with objective function value equal to 0, see Page [134], Garfinkel [88], Derigs and Zimmermann [70], Carpaneto and Toth [55] and Derigs [67]. FORTRAN codes for

this method can be found in the book by Burkard and Derigs [39] and in Carpaneto and Toth [55]. The implementations differ in the determination of a starting solution and in the applied data structures. One of the most efficient implementations is described in Derigs [67]. A similar approach using the framework of strong spanning trees (see also Section 3 and Balinski and Gonzalez [21]) is given by Armstrong and Jin [10]. Their algorithm can be implemented within a running time of $O(mn + n^2 \log n)$, where m is the number of finite entries in matrix C, i.e., the number of edges of the bipartite graph G associated with the assignment problem.

An algorithm with the currently best time complexity is obtained by combining the threshold approach with augmenting paths. This idea goes back to Gabow and Tarjan [87], who gave an algorithm for LBAP with worst case complexity $O(m\sqrt{n \log n})$. For dense graphs this bound has been further improved by Punnen and Nair [150]. According to these authors we first solve

$$\min_{M \in F^*} \max_{(i,j) \in M} c_{ij} \,, \tag{18}$$

where F^* is the set of all matchings in G which differ from a maximum matching by at most $n\sqrt{n/m}$ edges. Combining the maximum matching algorithm of Alt et al. [9] (see Section 1) with binary search, this problem can be solved in $O(n^{1.5}\sqrt{m})$ time. Then this solution is extended to a optimal solution of the bottleneck assignment problem by growing at most $n\sqrt{n/m}$ augmenting paths. Every augmenting path can be completed in $O(m)$ time, see e.g. Tarjan [159]. Thus the overall complexity of the algorithm becomes $O(n\sqrt{nm})$.

Pferschy [139] describes an algorithm with expected running time $O(n^2)$, i.e., linear on m in the case that the graph G is dense. In a first step the given graph is thinned out by considering only the $2n \log n$ cheapest edges. This can be done in $O(n^2)$ time by using a linear selection algorithm for finding the $2n \log n$-smallest edge. In the second step the LBAP on the sparse subgraph is solved by using the method of Gabow and Tarjan. This yields $O((n \log n)^{3/2})$ additional elementary operations. Finally the solution is completed to a perfect matching of the full graph by applying again the algorithm of Gabow and Tarjan. But this completion step is only necessary with a low probability, namely with a probability less than $O(1/\sqrt{n \log n})$. Thus, the expected time needed by the completion is $O(n^2)$, and we get an overall expected running time of $O(n^2)$. This algorithm provides not only a good bound in terms of complexity, but is also very efficient and simple to use.

In a detailed study Pferschy [140] compared different (deterministic) solution procedures for LBAPs up to $n = 2000$. He observed that the fastest algorithm with respect to the overall running time consists of computing a lower bound for the optimal solution value, e.g. by (17), then finding a maximum matching (using the Hopcroft-Karp procedure) in the subgraph consisting of the edges with weight less or equal to this bound, and finally augmenting this matching. Moreover, Pferschy noticed that the selection of a sparse subgraph takes a considerable amount of time, whereas not much computational efforts are needed to solve the LBAP in sparse subgraphs.

There are again some simple cases when an optimal solution of the LBAP can be given before hand. As in Section 2 these cases are related to Monge properties of the cost matrix C.

Matrix $C = (c_{ij})$ is said to be a *bottleneck Monge matrix* (cf. Burkard et al. [42]), if it fulfills the following conditions:

$$\max\{c_{ij}, c_{kl}\} \leq \max\{c_{il}, c_{kj}\} \,, \qquad \text{for } 1 \leq i < k \leq n,\ 1 \leq j < l \leq n. \tag{19}$$

It is easy to show that the identical permutation is an optimal solution of an LBAP with a bottleneck Monge cost matrix. An analogous result holds for inverse bottleneck Monge matrices defined by

$$\max\{c_{ij}, c_{kl}\} \geq \max\{c_{il}, c_{kj}\}, \qquad \text{for } 1 \leq i < k \leq n,\ 1 \leq j < l \leq n. \tag{20}$$

LBAPs with an inverse bottleneck Monge cost matrix are solved by the permutation $\phi(i) = n - i + 1$, for all i. The recognition of permuted bottleneck Monge matrices, however, is more difficult than solving the bottleneck assignment problem directly, see Klinz, Rudolf, and Woeginger [115], and Burkard et al. [42].

In Pferschy [139] the asymptotic behavior of the LBAP is studied. Pferschy shows that the expected value of the optimal solution of an LBAP with independently and identically distributed costs tends towards the infimum of the cost range as the size of the problem tends to infinity. He derives explicit lower and upper bounds for the optimum value of an LBAP of size n.

Besides the classical application in scheduling theory bottleneck assignment problems have also been used in connection with time slot assignment problems (see the description of the telecommunication problem in the earth-satellite systems, Section 2). One variant of the time slot assignment problem is to decompose the given traffic matrix in a weighted sum of at most n permutation matrices such that $T \leq \sum_{k=1}^{n} \lambda_k P_k,\ \lambda_k \geq 0$ holds and $\sum_{k=1}^{n} \lambda_k$ is minimum. Balas and Landweer [13] propose a heuristic for this NP-hard problem: solve a series of bottleneck assignment problems in order to determine the weights λ_k.

A multi-level bottleneck assignment problem has been considered by Carraresi and Gallo [50] in connection with the bus drivers' rostering problem: Daily shifts are to be assigned to bus drivers. To each shift a weight is assigned. This weight represents the cost of the shift for the drivers, e.g., length of the shift, lateness, time to reach the starting point etc. The problem of finding an equal balance of the shifts leads to minimizing the total weight of the shifts assigned to a single driver. For solving this NP-hard optimization problem the authors propose to solve a sequence of bottleneck assignment problems.

Lexicographic bottleneck assignment problems (LexBAP) have been considered by Burkard and Rendl [43]. Let $C = (c_{ij})$ be a real $n \times n$ matrix and let ϕ be a permutation of the set $\{1, 2, \ldots, n\}$. Now let c_ϕ be the vector where the n cost coefficients $c_{i,\phi(i)}$, $1 \leq i \leq n$, are ordered decreasingly. The lexicographic bottleneck assignment problem asks for a solution ϕ^* with

$$c_{\phi^*} = \text{lexmin}_{\phi \in \mathcal{S}_n} c_\phi.$$

The authors propose two solution approaches. The first one scales the cost coefficients and transforms the LexBAP in a sum assignment problem. (A similar transformation of bottleneck assignment problems in sum assignment problems had already been proposed by Page [134]). In a second approach the LexBAP is iteratively solved by a series of bottleneck and sum assignment problems. If the LexBAP has k distinct cost coefficients and we denote the computational complexity of the sum assignment problem by $T(n)$, then LexBAP can be solved in $O(T(n)n^2 \log k)$ time. Bardadym [25] suggested the following improved algorithm. Order the k distinct cost elements decreasingly and represent any cost coefficient c_{ij} by a vector $d_{ij} = (d_{ij}^1, d_{ij}^2, \ldots, d_{ij}^k)$ with k elements, where $d_{ij}^r = 1$, if c_{ij} is the $r-$largest cost coefficient, and $d_{ij}^r = 0$, otherwise. In this model the lexicographic bottleneck assignment problem becomes a lexicographic vector assignment problem which fits in the framework of algebraic assignment problems (see Section 6). The solution of this problem involves $O(kT(n))$ elementary operations.

Balanced assignment problems have been considered by Martello, Pulleyblank, Toth, and de Werra [125]. Given a real $n \times n$ matrix $C = (c_{ij})$, the balanced assignment problem can be formulated as

$$\min_{\phi} \left[\max_{i} c_{i\phi i} - \min_{i} c_{i\phi i} \right].$$

For solving this problem the authors sort the entries c_{ij} non-decreasingly and propose the following $O(n^4)$ procedure:

1. Solve the corresponding bottleneck assignment problem. Let ϕ^* be its optimal solution.

2. Define
$$l := \min_{i} c_{i\phi^*(i)}, \;\; u := \max_{i} c_{i\phi^*(i)}.$$
If $l = u$, stop. A balanced solution has been found. Otherwise go to Step 3.

3. Delete in C all elements $\leq l$ and $> u$ and grow augmenting paths.
If there exists a perfect matching, set ϕ^* equal to that solution and go to Step 2. If no solution exists, then go to Step 4.

4. If u is already the maximum element of matrix C, stop. The present solution is optimal. Otherwise increase u to the next larger element and return to Step 3.

6 Assignment Problems with Other Objective Functions

As pointed out by Burkard, Hahn, and Zimmermann [41], sum and bottleneck assignment problems are just special cases of a more general problem, the so-called *algebraic assignment problem* which can also be solved efficiently.

Let $(H, *, \prec)$ be a totally ordered commutative semigroup with composition $*$ and order relation $\prec$. The algebraic assignment problem (AAP) with cost coefficients $c_{ij} \in H$ can be formulated as

$$\min_{\phi \in \mathcal{S}_n} \; c_{1\phi(1)} * c_{2\phi(2)} * \cdots * c_{n\phi(n)} \,. \tag{21}$$

For deriving efficient solution procedures we require some additional properties for $(H, *, \prec)$. We assume that the order relation is compatible with the composition, i.e.,

$$a \prec b \Rightarrow a * c \prec b * c \quad \text{for } a, b, c \in H \,,$$

and that H is a so-called *d-monoid*, i.e.,

$$a \prec b \;\Rightarrow\; \text{there exists } c \in H \text{ such that } a * c = b.$$

As we will see below, under these conditions the AAP can be solved in $O(n^4)$ time. If we additionally require that the *weak cancellation rule*

$$\text{if } \; a * c = b * c, \; \text{then either } a = b \text{ or } a * c = b$$

applies, the complexity can be improved to $O(n^3)$, see Burkard and Zimmermann [46]. For further results in this direction, consult the survey on algebraic optimization by Burkard and Zimmermann [47].

Special models for d-monoids are:

- $H = \mathbb{R}$ with addition as composition and the usual (or inverse) order relation. This model leads to sum assignment problems (with a minimization or a maximization objective function).
- H is the set of extended real numbers $\bar{\mathbb{R}}$ (including $-\infty$) with the usual order relation. The composition is defined by $a * b := \max\ (a, b)$. This model leads to bottleneck assignment problems.
- $H = \mathbb{R}^n$, the composition is the vector addition and the order relation is the lexicographical order. This leads to lexicographical sum assignment problems.
- $H = \mathbb{R} \times \bar{\mathbb{R}}$, $\prec$ is the lexicographical order and the composition is defined by

$$(a, b) * (c, d) := \begin{cases} (a, b) & \text{if} \quad a \geq c \\ (a, b + d) & \text{if} \quad a = c \end{cases}$$

  This leads to the so-called *time-cost assignment problem*, where a duration measure t_{ij} and a cost c_{ij} are assigned under these conditions to each pair (i, j). Among all assignments which minimize the overall duration we seek for one which minimizes the overall cost.
- $H = [0, 1]$, the order relation is the usual "greater than or equal to" order $\geq$, and the composition is the multiplication of real numbers. This model leads to reliability assignment problems: $\max \Pi_{i=1}^{n} a_{i\phi(i)}$ over all permutations ϕ of $1, 2, \ldots, n$.

Algebraic assignment problems can be solved by performing *admissible transformations*. Let us denote

$$z[C, \phi] := c_{1,\phi(1)} * c_{2,\phi(2)} * \cdots * c_{n,\phi(n)}\,.$$

Definition 6.1 *(Admissible transformations)*
A transformation T of the $n \times n$ matrix $C = (c_{ij})$ to the matrix $\bar{C} = (\bar{c}_{ij})$ is called admissible *with* index $z(T)$*, if*

$$z[C, \phi] = z(T) * z[\bar{C}, \phi]$$

for all $\phi \in \mathcal{S}_n$.

A composition of admissible transformations S and T with indices $z(S)$ and $z(T)$, respectively, is again an admissible transformation with index $z(S) * z(T)$. Now we get the following optimality criterion:

Theorem 6.2 *Let $T : C \mapsto \bar{C}$ be an admissible transformation such that there exists a permutation $\hat{\phi}$ with the following properties:*

1. $z(T) * \bar{c}_{ij} \geq z(T)$
2. $z[\bar{C}, \hat{\phi}] * z(T) = z(T)$.

Then $\hat{\phi}$ is an optimal assignment with value $z(T)$.

Proof. Let ϕ be an arbitrary permutation. According to Definition 6.1 and properties (1) and (2) of the proposition above we get:

$$z[C, \phi] = z(T) * z[\bar{C}, \phi] \geq z(T) = z(T) * z[\bar{C}, \hat{\phi}] = z[C, \hat{\phi}]$$

Therefore $\hat{\phi}$ is optimal.
Admissible transformations for assignment problems are described by the following theorem:

Theorem 6.3 (Admissible transformations for assignment problems [41], 1977)
Let $I, J \subseteq \{1, 2, \ldots, n\}$, $m := |I| + |J| - n \geq 1$, and $c := \min\{c_{ij} : i \in I, j \in J\}$. Then the transformation $C \mapsto \bar{C}$ defined by

$$\begin{aligned} \bar{c}_{ij} * c &= c_{ij}\,, && \text{for } i \in I, j \in J \\ \bar{c}_{ij} &= c_{ij} * c\,, && \text{for } i \notin I, j \notin J \\ \bar{c}_{ij} &= c_{ij}\,, && \text{otherwise} \end{aligned}$$

is admissible with $z(T) = c * c * \cdots * c$, where the expression on the right hand-side contains m factors.

For solving an algebraic assignment problem we can use now the following procedure:

1. Perform *row reductions*, i.e., perform admissible transformations with $I = \{i\}, J = \{1, 2, \ldots, n\}$. Start with $i = 1$ and let $z := z(T)$ be the corresponding index. Continue with $i = 2, \ldots, n$ and update $z := z * z(T)$. Afterwards all elements are "nonnegative", namely $\bar{c}_{ij} * z \geq z$.

2. Perform *column reductions*, i.e., perform admissible transformations with $J = \{j\}, I = \{1, 2, \ldots, n\},$, for $j = 1, 2, \ldots, n$. Afterwards every row and column contains at least one *generalized 0-element*, i.e., an element which fulfills

$$z * \bar{c}_{ij} = z\,.$$

3. Determine a maximum matching in subgraph of the bipartite graph associated with the assignment problem, which contains only the edges corresponding to the generalized 0-elements, (cf. Theorem 1.6).

4. If the maximum matching is perfect, then stop: the optimal solution is given by this matching and z is the optimal value of the objective function.
Otherwise, go to Step 5.

5. Determine a minimum cover of the "0-elements". This cover yields the new index sets I and J. I contains the indices of the uncovered rows, J contains the indices of uncovered columns.

6. Perform an admissible transformation determined by the new index sets I and J as in Theorem 6.3, update $z := z * z(T)$, and go to Step 3.

It is rather straightforward to show that this algorithm yields after at most $n^2 - 2n + 3$ admissible transformations an optimal solution of the algebraic assignment problem. If the composition $*$ is specialized to "+", it becomes a variant of the Hungarian method. If the composition is specialized to the max operation, then we obtain the bottleneck assignment problem and the above algorithm is a variant of the threshold method.

Analogously to (sum) Monge and bottleneck Monge matrices, we can also define Monge matrices in the case that c_{ij} are taken from the ordered semigroup H, cf. Burkard et al. [42]. The AAP with an algebraic Monge cost matrix is again solved to optimality by the identity permutation. The problem of recognizing permuted algebraic Monge matrices is rather subtle. Woeginger [169] has shown that this problem is NP-hard, if $n \geq 3$ and the ordered

semigroup fulfills no additional property. It becomes polynomially solvable, if e.g. a weak cancellation rule of the form

$$a * c = b * c \Rightarrow a = b \quad \text{or} \quad a * c = c\,, \qquad \text{for} \quad a, b, c \in H$$

is fulfilled. For details see Burkard et al. [42].

7 Available computer codes and test instances

FORTRAN listings of a primal-dual algorithm for the LSAP, based on shortest path computations done by a version of Dijkstra's algorithm, can be found in the book by Burkard and Derigs [39]. In that book there is also a code of an algorithm for the linear bottleneck assignment problem (LBAP). This algorithm is obtained as an adaptation of the above mentioned LSAP algorithm to the LBAP, and is described in detail by Derigs and Zimmerman [70]. The differences between the two algorithms are discussed by Burkard and Zimmermann [46].

Source codes of another primal-dual algorithm for the LSAP can be downloaded from `http://207.158.230.188/assignment.html`. One can choose among a C++, a PASCAL, and a FORTRAN implementation of the algorithm of Jonker and Volgenant [108] for dense LSAPs, and among a FORTRAN and a PASCAL implementation for sparse LSAPs. There are also PASCAL and FORTRAN codes of an adaptation of the above mentioned algorithm to the LBAP. All codes are available as `zip` files.

A (compressed) FORTRAN source file - called `548.Z` - of an implementation of the Hungarian algorithm, due to Carpaneto and Toth [54], can be downloaded from
`ftp://netlib.att.com` in `/netlib/toms`. Other listings of FORTRAN codes for the LSAP can be found in Carpaneto, Martello, and Toth [51]. These codes are also available from the floppy disc included in the book [51].

The C code (compressed tar) of an efficient implementation of the scaling push-relabel algorithm of Goldberg and Kennedy [94] for the LSAP can be downloaded from Goldberg's network optimization library at `http://www.neci.nj.nec.com/homepages/avg/soft.html`.

Finally, listings of 5 FORTRAN codes of auction algorithms for the LSAP can be found in Bertsekas' homepage at `http://web.mit.edu/dimitrib/www/auction.txt`.
The codes are: `AUCTION` - a forward auction algorithm for symmetric assignment problems, i.e., problems where the cost matrix is symmetric,
`AUCTION_FLP` - a version of `AUCTION` which emploies floating point arithmetic, `AUCTION_AS` - an auction algorithm for the asymmetric LSAP, `AUCTION_FR` - a forward-reverse auction algorithm for symmetric LSAPs, and `NAUCTION_SP` - a combination of the naive auction algorithm with sequential shortest paths methods.

Test instances of the LSAP can be downloaded as `ascii` files from the homepage of the OR-Library maintained by J. Beasley at `http://mscmga.ms.ic.ac.uk/pub`.
Other test instances can be obtained as `ascii` or uuencoded files from the ELIB library at
`ftp://ftp.zib.de/pub/Packages/mp-testdata/assign/index.html`.

Let us notice that due to the equivalent formulation of the LSAP as a minimum cost flow problem, algorithms developed for the later can also be applied to the LSAP. However, such algorithms are not supposed to exploit the specific features of the LSAP, and

hence may not be competitive with algorithms developed especially for the LSAP. Besides Goldberg's network optimization library and Bertsekas' homepage, other codes for network optimization can be found in `Netlib` at

`http://www.OpsResearch.com/OR-Links/index.html`.

Furthermore, `C` codes (compressed tar files) of implementations of the primal and the dual network simplex algorithm, due to Löbel, can be obtained through

`http://www.zib.de/Optimization/index.de.html`.

8 Multidimensional Assignment Problems

8.1 General Remarks and Applications

Multi-dimensional (sometimes referred as *multi-index*) assignment problems (MAP) are natural extensions of the linear assignment problem. They have been considered for the first time by Pierskalla [143]. The most prominent representatives of this class are axial and planar 3-dimensional assignment problems, which are treated in the subsections below. The general formulation of the MAP is

$$
\begin{aligned}
\min \quad & \sum_{i_1=1}^{n} \cdots \sum_{i_N=1}^{n} c_{i_1 \cdots i_d} x_{i_1 \cdots i_d} \\
\text{s.t.} \quad & \sum_{i_2=1}^{n} \cdots \sum_{i_d=1}^{n} x_{i_1 \cdots i_d} = 1, \quad i_1 = 1, \ldots, n, \\
& \sum_{i_1=1}^{n} \cdots \sum_{i_{k-1}=1}^{n} \sum_{i_{k+1}=1}^{n} \cdots \sum_{i_d=1}^{n} x_{i_1 \cdots i_d} = 1, \\
& \quad \text{for } \; k = 2, \ldots, d-1, \;\; \text{and } i_k = 1, 2, \ldots, n, \\
& \sum_{i_1=1}^{n} \cdots \sum_{i_{d-1}=1}^{n} x_{i_1 \cdots i_d} = 1, \quad i_d = 1, \ldots, n, \\
& x_{i_1 \cdots i_d} \in \{0, 1\} \;\; \text{for } \; 1 \le i_1, i_2, \ldots i_d \le n,
\end{aligned}
\tag{22}
$$

with n^d cost coefficients $c_{i_1 \cdots i_d}$.
Differently stated the MAP asks for $d-1$ permutations $\phi_1, \phi_2, \ldots, \phi_{d-1}$ such that the objective function becomes minimum:

$$
\min_{\phi_1, \phi_2, \ldots, \phi_{d-1}} \; \sum_{i=1}^{n} c_{i \phi_1(i) \phi_2(i) \ldots \phi_{d-1}(i)} \, .
$$

The multidimensional assignment problem is NP-hard in general, but in the case that the array of the cost coefficients is a Monge array (see [42]), it is solved by the identical permutations $\phi_i = \mathrm{id}$, for $i = 1, 2, \ldots, d-1$. A slightly weaker condition is considered in Fortin and Rudolf [79]. Burkard, Rudolf and Woeginger [45] have shown, however, that the MAP remains NP-hard for $d \ge 3$, if the cost array is inverse Monge. This implies that *maximizing* the objective function with cost elements drawn from a Monge array is NP-hard.
In terms of graphs a multidimensional assignment problem can be described as follows: Let a complete d-partite graph $G = (V_1, V_2, \ldots, V_d; E)$ with vertex sets V_i, $|V_i| = n$, $i = 1, 2, \ldots, d$, and edge set E be given. A subset X of $V = \bigcup_{i=1}^{d} V_i$ is a *clique*, if it meets every set V_i in exactly one vertex. A d-dimensional assignment is a partition of V into n

pairwise disjoint cliques. If c is a real valued cost function defined on the set of cliques of $G = (V_1, V_2, \ldots, V_d; E)$, the d-dimensional assignment problem asks for a d-dimensional assignment of minimum cost.

Bandelt, Crama, and Spieksma [24] consider cases where the costs c of a clique are not arbitrary, but given as a function of elementary costs attached to the edges of the complete d-partite graph. The costs of a clique considered in [24] are sum costs (sum of the lengths of edges induced by the clique), star costs (minimum length of a spanning star), tour costs (minimum cost of a traveling salesman tour in the clique), and tree costs (minimum cost of a spanning tree in the clique). The authors derive worst case bounds on the ratio between the costs of solutions obtained by applying simple heuristics and the costs of optimal solutions for these special MAPs.

Multidimensional assignment problems in their general form have recently found some applications as a means to solve data association problems. More specifically, the central problem in any multi-target tracking and multi-sensor surveillance is the data association problem of partitioning the observations into tracks and false alarms in real time. General classes of these problems can be formulated as multidimensional assignment problems. In the following we briefly describe such a formulation for a multi-target tracking problem. The reader is referred to Poore, Rijavec, Liggins, and Vannicola [147] for more details and an alternative derivation of the multidimensional assignment formulation. In a target tracking process we collect reports from different measurements performed at discrete moments in time denoted by $1, 2, \ldots, n$. Let us denote by $Z(k)$ a data set of M_k reports (dummy or missing reports are included) obtained by measurements at time k, and let Z^n denote the cumulative set of $Z(k)$, $k = 1, 2, \ldots, n$,

$$Z(k) = \{z_{i_k}^k\}_{i_k=0}^{M_k}, \qquad Z^n = \{Z(1), Z(2), \ldots, Z(n)\}.$$

The index $i_k = 0$ is reserved for missing reports. We are looking for a feasible partition γ of the data set Z^n into data tracks corresponding to different objects and false alarms, which leads to a best possible reproduction of the real life situation. A feasible partition $\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_{n(\gamma)}\}$ consists of $n(\gamma)$ tracks γ_j, $1 \le j \le n(\gamma)$ of reports. Each track γ_j of a partition γ is given as $\gamma_j = \{z_{i_1}^1, z_{i_2}^2, \ldots, z_{i_n}^n\}$ and is defined in terms of the properties it fulfills. First the different tracks may have only missing or dummy reports in common, and no track consists of only dummy reports:

$$\gamma_i \cap \gamma_j \subset \{z_0^1, z_0^2, \ldots, z_0^n\}, \quad \text{for } i \neq j, \text{ and } \gamma_j \neq \{z_0^1, z_0^2, \ldots, z_0^n\}, \quad \text{for all } j$$

Further, the union of all tracks together with the track consisting of only dummy reports covers the cumulative set of data Z^n, and each track contains at least one report from the set $Z(k)$, $1 \le k \le n$:

$$Z^n = \left[\cup_{j=1}^{n(\gamma)} \gamma_j\right] \cup \{z_0^1, z_0^2, \ldots, z_0^n\}, \qquad \gamma_j \cap Z(k) \neq \emptyset, \text{ for all } j, k.$$

Now we can use 0-1 variables $x_{i_1 i_2 \ldots i_n}$ to characterize a feasible partition γ, where

$$x_{i_1 i_2 \ldots i_n} = \begin{cases} 1 & \text{if } \{z_{i_1}^1, z_{i_2}^2, \ldots, z_{i_n}^n\} \in \gamma \\ 0 & \text{otherwise} \end{cases}$$

and the following equalities are fulfilled

$$\sum_{i_1, i_2, \ldots, i_{k-1}, i_{k+1}, \ldots, i_n}^{M_1, M_2, \ldots, M_{k-1}, M_{k+1}, \ldots, M_n} x_{i_1 i_2 \ldots i_n} = 1, \quad \text{for } k = 1, 2, \ldots, n.$$

It is clear that these equalities are similar to those in the formulation (22) of the MAP given earlier in this section. In the special case where all M_i have a common value we obtain the constraints of (22).
Let us turn now to the modeling of the objective function. Let γ_0 be a reference partition, e.g. the partition consisting of all false reports, and let $P(\gamma|Z^n)$ be the probability of a partition γ given the data set Z^n. Without going into details - for which the user is referred to [147] - notice that the maximization of $P(\gamma|Z^n)/P(\gamma_0|Z^n)$ over all feasible partitions γ would be a reasonable objective function from the physical point of view. Under natural probabilistic assumptions on the space of feasible partitions it can be assumed that

$$\frac{P(\gamma|Z^n)}{P(\gamma_o|Z^n)} = L_\gamma = \prod_{\{z^1_{i_1}, z^2_{i_2}, \ldots, z^n_{i_n}\} \in \gamma} L_{i_1 i_2 \ldots i_n} \,,$$

where each *likelihood ratio* $L_{i_1 i_2 \ldots i_n} \equiv L(z^1_{i_1}, z^2_{i_2}, \ldots, z^n_{i_n})$ depends only on the sequence of reports $\{z^1_{i_1}, z^2_{i_2}, \ldots, z^n_{i_n}\}$ and not on the partition in which this sequence occurs. (The reader is referred to Poore and Rijavec [146] for a complete list of assumptions under which this decomposition is allowed.) By introducing $c_{i_1 i_2 \ldots i_n} := -\ln L_{i_1 i_2 \ldots i_n}$ for all $i_1, i_2, \ldots, i_n$ we get

$$-ln \left[\frac{P(\gamma|Z^n)}{P(\gamma_0|Z^n)} \right] = \sum_{\{z^1_{i_1}, z^2_{i_2}, \ldots, z^n_{i_n}\} \in \gamma} c_{i_1 i_2 \ldots i_n} \,. \tag{23}$$

Then the maximization of $P(\gamma|Z^n)/P(\gamma_0|Z^n)$ is equivalent to the minimization of the sum in the right hand-side of (23).

For other applications of MAPs in target tracking problems, like track initiation and track maintenance, or multi-sensor tracking, the reader is referred to Poore [144] and the references therein. Some Lagrangian relaxation algorithms for the MAP have been developed by Poore and Rijavec [146] and Poore and Robertson [148]. These algorithms relax one or more constraints and derive a set of good Lagrangean multipliers, e.g. by standard non-smooth optimization techniques, in a first phase. In a second phase they involve a so-called *recovery* procedure to construct a good feasible solution of the original problem starting from an optimal (or good) solution of the relaxed problem with values of Lagrangean multipliers determined in the first phase. A numerical study of data association problems arising in multi-target and multi-sensor tracking is given in Poore [145].

Pusztaszeri, Rensing, and Liebling [149] describe another interesting MAP which arises in the context of tracking elementary particles. By solving a five-dimensional assignment problem, they reconstruct tracks generated by charged elementary particles produced by the Large Electron-Positron Collider (LEP) at CERN institute.

8.2 Axial 3-Dimensional Assignment Problems

Consider n^3 cost coefficients c_{ijk}. The *axial 3-dimensional assignment problem* (3-DAP) can then be stated as

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} x_{ijk} \\ \text{s.t.} \quad & \sum_{j=1}^{n} \sum_{k=1}^{n} x_{ijk} = 1 \,, \quad i = 1, 2, \ldots, n \,, \end{aligned}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1, \quad i = 1,2,\ldots,n\,, \tag{24}$$
$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk} = 1, \quad i = 1,2,\ldots,n\,,$$
$$x_{ijk} \in \{0,1\} \;\; \forall \quad 1 \le i,j,k \le n\,.$$

The name stems from the model which assigns the 1-s on the right hand-side to distinct positions at the axes of a 3-dimensional array. The sum over the corresponding "flat" in the array must equal the amount assigned to the position on the axis, see Fig. 5. We can

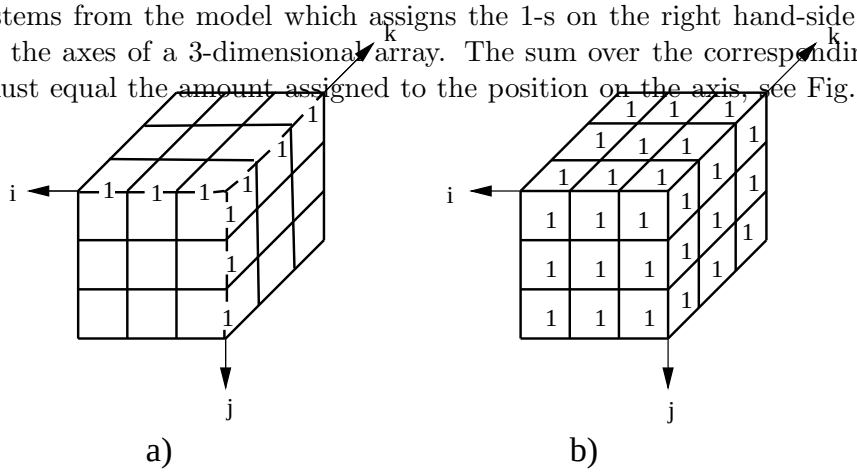


Figure 5: a) A geometric representation of the axial 3-dimensional assignment problem for $n = 3$. b) A geometric representation of the planar 3-dimensional assignment problem for $n = 3$.

think of c_{ijk} as the cost of assigning job j to be performed by worker i in machine k. It follows that $x_{ijk} = 1$, if job j is assigned to worker i in machine k, and $x_{ijk} = 0$, otherwise. A feasible solution of the above problem will be a three dimensional *permutation array*, i.e., a 0-1 array that satisfies the above assignment constraints.
Equivalently, a 3-DAP can be described with the help of two permutations ϕ and ψ

$$\min_{\phi,\psi\in\mathcal{S}_n} \sum_{i=1}^{n} c_{i\phi(i)\psi(i)}. \tag{25}$$

Thus this problem has $(n!)^2$ feasible solutions. Karp [109] showed that the 3-DAP is NP-hard in contrast to the sum linear assignment problem (LSAP).

In terms of graph theory, the 3-DAP can be stated as follows: Given a complete 3-partite graph $K_{n,n,n}$ with disjoint vertex sets U,V, and W, find a subset A of n triangles, $A \subseteq U \times V \times W$, such that every element of $U \cup V \cup W$ occurs in exactly one triangle of A and the total cost $\sum_{(i,j,k)\in A} c_{ijk}$ is minimized.

Euler [75] started the investigation of the axial 3-index assignment polytope (i.e., the convex hull of feasible solutions to problem (24)) by considering the role of odd cycles for a class of facets of this polytope. Independently Balas and Saltzman [16] investigate in detail the polyhedral structure of the three-index assignment polytope, and show that this polytope has dimension n^3-3n+2. They describe an $O(n^4)$ separation algorithm for facets induced by certain cliques. Balas and Qi [15] continue the above work and present $O(n^3)$ separation algorithms for earlier identified classes of facets as well as for a new class of facets. (Note that since there are n^3 variables these are linear-time algorithms). Qi, Balas,

and Gwan [151] base their work on linear-time separation algorithms for different facets and construct a polyhedral procedure for solving the 3-DAP.
Pierskalla [142] was the first, who proposed a heuristic for solving 3-DAP. As for algorithms to solve the 3-DAP to optimality, branch and bound is a classical one. Most of the algorithms split the current problem into two subproblems by fixing one variable x_{ijk} to 1 and to 0, respectively. In this way the size of the first subproblem decreases. Balas and Saltzman [17] have introduced a branching strategy that exploits the structure of the problem and allows to fix several variables at each branching node. Burkard and Rudolf [44] experiment with different branch and bound schemes for the 3-DAP. In particular, they investigate branching rules stemming from the polyhedral description of the underlying polytope, see Balas and Saltzman [16].
Hansen and Kaufman [100] describe a primal-dual method similar to the Hungarian method for linear assignment problems. In the case of 3-DAPs, hypergraphs have to be considered instead of bipartite graphs. First as many 0-elements as possible are generated among the cost coefficients by generalized row- and column reductions. Then a covering problem is solved for these 0-elements. If the covering number is smaller than n, further 0-elements are generated by means of an admissible transformation similar to those in Section 6. If the covering number equals n, the corresponding stability problem is solved. Notice that for hypergraphs the minimum number of (hyper-)edges in a cover is in general strictly larger than the cardinality of a stable set, and this is an inherent difficulty for this problem class. If the stability number equals n, an optimal solution has been found, otherwise a branching is performed. Note the similarity of this method with the approach in Section 6 for solving algebraic assignment problems. Due to this similarity the same algorithm solves the *"algebraic"* version of the 3-DAP with cost coefficients drawn from a d-monoid. The subproblems of finding a minimum cover and a maximum stable set are in this case, however, NP-hard.

Admissible transformations for 3-DAP are described by the following theorem due to Burkard and Fröhlich [40]. Let us first introduce some notation. Let $N := \{1, 2, \ldots, n\}$, I,J,$K \subseteq N$ and denote $\bar{I} := N \setminus I$ $\bar{J} = N \setminus J$, $\bar{K} = N \setminus K$. With these definitions we get

Theorem 8.1 (Admissible transformations for the 3-DAP [40], 1980
Let a 3-DAP with cost coefficients c_{ijk}, $i, j, k = 1, 2. \ldots, n$, be given. For $I, J, K \subseteq N$ with $m := n - (|I| + |J| + |K|) \geq 1$ and

$$c := \min\{c_{ijk} \colon (i, j, k) \in \bar{I} \times \bar{J} \times \bar{K}\}$$

we define

$$\begin{aligned}
\bar{c}_{ijk} &:= c_{ijk} - c\,, && (i, j, k) \in \bar{I} \times \bar{J} \times \bar{K} \\
\bar{c}_{ijk} &:= c_{ijk} + c\,, && (i, j, k) \in (\bar{I} \times J \times K) \cup (I \times \bar{J} \times K) \cup (I \times J \times \bar{K}) \\
\bar{c}_{ijk} &:= c_{ijk} + 2c\,, && (i, j, k) \in I \times J \times K \\
\bar{c}_{ijk} &:= c_{ijk}\,, && \textit{otherwise}
\end{aligned}$$

Then, for any feasible solution ϕ, ψ of the 3-DAP we have

$$\sum_{i=1}^{n} c_{i\phi(i)\psi(i)} = \sum_{i=1}^{n} \bar{c}_{i\phi(i)\psi(i)} + mc.$$

Clearly, row and column reductions are special cases of the admissible transformations described by the above theorem.

Strong lower bounds are essential for a branch and bound procedure. One of the approaches used to compute lower bounds for the 3-DAP is Lagrangean relaxation. By taking two blocks of the constraints of a 3-DAP into the objective function via Lagrangean multipliers, we get a Lagrangian relaxation of (24) (cf. [40]):

$$L(\pi,\epsilon) := \min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(c_{ijk}+\pi_j+\epsilon_i)x_{ijk} - \sum_{j=1}^{n}\pi_j - \sum_{i=1}^{n}\epsilon_i\right\}$$

such that

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk} &= 1\,, \quad k = 1,2,\ldots,n\\ x_{ijk} &\in \{0,1\}\,, \qquad 1 \le i,j,k \le n\\ \pi \in \mathbb{R}^n, \epsilon &\in \mathbb{R}^n\,.\end{aligned}$$

$L(\pi,\epsilon)$ is a concave function, and a subgradient method can be used to find its maximum. Let r be a counter of iterations. Start with $\pi^r := \epsilon^r := 0$ for $r = 0$. In each iteration r use a greedy algorithm to minimize $L(\pi^r,\epsilon^r)$. Let x^r_{ijk} be the corresponding optimal solution. Define $v^r_{i_0} := |\{x^r_{i_0,j,k}\colon x_{i_0,j,k} = 1\}|-1$ for $i_0 = 1,2,\ldots,n$, and $w^r_{j_0} := |\{x^r_{i,j_0,k}\colon x_{i,j_0,k} = 1\}|-1$ for $j_0 = 1,2,\ldots,n$. If $v^r = w^r = (0,0,\ldots,0)$, then the maximum is reached. Otherwise π and ϵ are updated by setting

$$\pi^{r+1} := \pi^r + \lambda_r w^r, \quad \epsilon^{r+1} := \epsilon^r + \lambda_r v^r\,,$$

where λ is a suitable step length λ. This procedure is repeated a prespecified number of iterations.

Another subgradient procedure for solving a Lagrangean relaxation of the 3-DAP together with computational considerations has been described by Frieze and Yadegar [83]. Burkard and Rudolf [44] report on satisfactory computational results obtained by an algorithm which uses the classical branching rule combined with a reduction step in every node of the search tree. The lower bound computation is done by applying the above described subgradient optimization procedure. Another branch and bound algorithm proposed by Balas and Saltzman [17] solves another Lagrangean relaxation to compute the lower bounds. This relaxation incorporates a class of facet inequalities and is solved by a modified subgradient procedure. The upper bounds are computed by a primal heuristic based on the principle of minimizing maximum regret, plus a variable depth interchange phase. A novel branching strategy fixes several variables at each node and reduces the size of the branching tree. The authors report on satisfactory numerical results.

There exists a number of polynomially solvable special cases of the 3-DAP. As mentioned in Section 8.1, the 3-DAP becomes polynomially solvable, if the cost coefficients are taken from a 3-dimensional Monge array (see [42]). Burkard, Rudolf, and Woeginger [45] investigate 3-DAPs with decomposable cost coefficients, where $c_{ijk} = u_i v_j w_k$ and u_i, v_j, and w_k are nonnegative. It is shown that the maximization version of this problem is polynomially solvable, whereas the minimization is in general NP-hard. Moreover, several polynomially solvable special cases of the minimization problem are identified.

Crama and Spieksma [61] consider special cases of the graph theoretic formulation of the 3-DAP. They assume that the lengths of the edges of the underlying 3-partite graph fulfill

the triangle inequality, and that the cost of a triangle is defined either as sum of the lengths of its sides (problem $T\Delta$) or as sum of the lengths of the two shorter sides (problem $S\Delta$). The authors prove that the corresponding 3-DAPs are NP-hard. They design, however, heuristics which always deliver feasible solutions whose corresponding value of the objective function is within 3/2 (4/3) from the optimal value, in the case of $T\Delta$ ($S\Delta$). of the optimal cost. Computational experiments show that the performance of these heuristics is very good on randomly generated instances.

8.3 Planar 3-Dimensional Assignment Problems

Let n^3 cost coefficients c_{ijk} be given. *Planar 3-dimensional assignment problems* (3-PAP) have the following form.

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}x_{ijk} \\
\text{s.t.} \quad & \\
& \sum_{i=1}^{n} x_{ijk} = 1\,, \quad j,k = 1,2,\ldots,n\,, \\
& \sum_{j=1}^{n} x_{ijk} = 1\,, \quad i,k = 1,2,\ldots,n\,, \\
& \sum_{k=1}^{n} x_{ijk} = 1\,, \quad i,j = 1,2,\ldots,n\,, \\
& x_{ijk} \in \{0,1\} \quad i,j,k = 1,2,\ldots,n\,.
\end{aligned}
\tag{26}
$$

For a geometric interpretation of the 3-PAP see Fig. 5. Every "flat" in the three-dimensional array x_{ijk} must contain a (2-dimensional) assignment. Thus the feasible solutions of the 3-PAP correspond to *Latin squares*. Fig. 6 shows a feasible solution for a 3-PAP with $n = 3$: number 1 represents the assignment in the lowest horizontal flat, number 2 shows the assignment in the medium flat, and 3 represents the assignment in the upper flat. Due to this interpretation, the number of feasible solutions of a 3-PAP of size n equals the number of Latin squares of order n, and hence increases very fast. The number of different Latin squares of order d equals $d!(d-1)!T(d)$, where $T(d)$ is the number of reduced Latin squares. There is not much known about $T(d)$, the largest value known so far seems to be for $d = 9$. $T(9)$ =377,597,570,964,258,816 due to Bammel and Rothstein [23]. An explicit formula for the number of Latin squares of order d can be found in Shao and Wei [156], but this formula is difficult to evaluate.

Frieze [82] has shown that the 3-PAP is NP-hard. A description of the polyhedral structure of the 3-PAP polytope has been given by Euler, Burkard, and Grommes [76].

3-PAPs have interesting applications in time tabling problems. A recent study on time tables and related polyhedra can be found in Euler and Verge [77].

There are not many algorithms known for the 3-PAP. The first branch and bound algorithm goes back to Vlach [163]. Vlach computes the lower bounds by applying (generalized) row and column reductions similar to those used in the case of 3-DAPs. Another branch and bound procedure for the planar 3-index assignment problem has been described by Magos and Miliotis [124]. To our knowledge this is the only paper reporting on computational experience with an algorithm which solves the 3-PAP to optimality. The authors use a so-called relaxation heuristic to compute upper bounds by relaxing the third group of

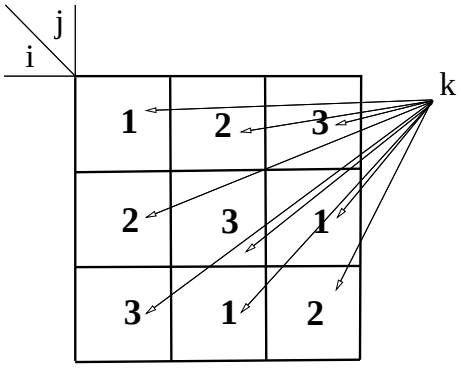


Figure 6: A latin square representing a feasible solution of the planar 3-dimensional assignment problem of size $n = 3$.

constraints in (26). The remaining problem can be decomposed into n LSAPs, one for each fixed value of the index k. These LSAPs are solved consecutively, and after solving each of them a number of variables are fixed so as to guarantee feasibility for the original problem. The so-found solutions are further improved by a local improvement procedure based on the relationship between 3-PAPs and Latin squares. More precisely, the current solution is presented as a Latin square and a cell of the Latin square is selected. The contents of this cell is changed according to some prespecified rule. Then, the contents of other cells of the Latin square is changed so as to obtain a new Latin square corresponding to a better feasible solution of the 3-PAP.
The lower bounds are computed by a heuristic to solve the dual and - in a small number of branch and bound nodes - also by solving a Lagrangean relaxation of the 3-PAP. The heuristic for the dual is based on row and column reductions in the fashion of the Hungarian method. The Lagrangean relaxation is obtained by relaxing the third group of constraints in (26). Good values for the Lagrangean multipliers are found by standard subgradient optimization techniques (see also Camerini, Fratta, and Maffioli [48]). For each set of fixed multipliers the relaxed problem can again be decomposed in n LSAPs, and is solved similarly as in the case of the relaxation heuristic. The algorithm uses depth first search and the following branching rule. Fix the indices i and j and let Q be a set of indices k such that $\{(i, j, k): k \in Q\}$ is the set of free variables. The set Q_1 contains about half as many elements as Q, namely the indices k for those free variables with smaller reduced costs. Then the current node is branched into two new nodes obtained by imposing $\sum_{k \in Q_1} x_{ijk} = 0$ and $\sum_{k \in Q \setminus Q_1} x_{ijk} = 0$.
A nice feature of this branch and bound scheme is that the lower bound is additive, and the lower bounding procedure works with the reduced costs instead of the original costs of the problem. Numerical experiments with instances of size up to 9 show that this really helps. As reported in [124], all instances of size 9 are solved in less than 4 minutes of CPU time.

The neighborhood structure involved in the improvement procedure mentioned above is also used by Magos [123] in a tabu search algorithm. A move in the neighborhood of some Latin square is completely determined by changing the contents a certain cell (i, j). This affects at least 4 other cells and at most $2n$ of them (n is the size of the problem). These cells have to be adapted accordingly. There are n^2 cells in total and there are $n - 1$ possibilities to change the contents of each of them. Consequently, the neighborhood size is between $\frac{n(n-1)}{2}$ and $\frac{n^2(n-1)}{4}$. There are two nice properties of this neighborhood structure. First, for each move the change of the objective function can be computed in linear time ($O(n)$). Secondly, not all moves have to be evaluated in each iteration: moves which put a certain

element in a certain cell imply the same change in the objective function, independently from the solution to which they are applied. The other tabu search elements involved in this algorithm, e.g. tabu size, tabu and aspiration criteria, stopping criterion etc., are designed and implemented in a standard way. The numerical results show that the algorithm provides a good trade-off between computation time and solution quality for 3-PAP instances of size up to 14.

References

[1] H. Achatz, P. Kleinschmidt, and K. Paparrizos, A dual forest algorithm for the assignment problem, in *Applied Geometry and Discrete Mathematics*, P. Gritzmann and B. Sturmfels, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **4**, AMS, Providence, RI, 1991, pp. 1–11.

[2] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, and M. R. Reddy, Applications of network optimization, in *Network Models - Handbooks of Operations Research and Management Science* **7**), M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, eds., Elsevier, Amsterdam, 1995, pp.1–83.

[3] R. K. Ahuja, K. Mehlhorn, J. B. Orlin, and R. E. Tarjan, Faster algorithms for the shortest path problem, *Journal of the ACM* **37**, 1990, 213–223.

[4] R. K. Ahuja and J. B. Orlin, The scaling network simplex algorithm, *Operations Research* **40**, *Suppl. No. 1*, 1992, S5–S13.

[5] M. Akgül, A sequential dual simplex algorithm for the linear assignment problem, *Operations Research Letters* **7**, 1988, 155-158.

[6] M. Akgül, The linear assignment problem, in *Combinatorial Optimization*, M. Akgül and S. Tufecki, eds., Springer Verlag, Berlin, 1992, pp. 85–122.

[7] M. Akgül, A genuinely polynomial primal simplex algorithm for the assignment problem, *Discrete Applied Mathematics* **45**, 1993, 93–115.

[8] M. Akgül and O. Ekin, A dual feasible forest algorithm for the assignment problem, *RAIRO Operations Research* **25**, 1991, 403–411.

[9] H. Alt, N. Blum, K. Mehlhorn, and M. Paul, Computing maximum cardinality matching in time $O(n^{1.5}\sqrt{m/\log n})$, *Information Process. Letters* **37**, 1991, 237–240.

[10] R. D. Armstrong and J. Zhiying, Solving linear bottleneck assignment problems via strong spanning trees, *Operations Research Letters* **12**, 1992, 179–180.

[11] D. Avis and L. Devroye, An analysis of a decomposition heuristic for the assignment problem, *Operations Research Letters* **3**, 1985, 279–283.

[12] D. Avis and C. W. Lai, The probabilistic analysis of a heuristic for the assignment problem, *SIAM Journal on Computing* **17**, 1988, 732–741.

[13] E. Balas and P. R. Landweer, Traffic assignment in communications satellites, *Operations Research Letters* **2**, 1983, 141–147.

[14] E. Balas, D. Miller, J. Pekny, and P. Toth, A parallel shortest path algorithm for the assignment problem, *Journal of the ACM* **38**, 1991, 985–1004.

[15] E. Balas and L. Qi, Linear-time separation algorithms for the three-index assignment polytope, *Discrete Applied Mathematics* **43**, 1993, 1–12.

[16] E. Balas and M. J. Saltzman, Facets of the three-index assignment polytope, *Discrete Applied Mathematics* **23**, 1989, 201–229.

[17] E. Balas and M. J. Saltzman, An algorithm for the three-index assignment problem, *Operations Research* **39**, 1991, 150–161.

[18] M. L. Balinski, Signature methods for the assignment problem, *Operations Research* **33**, 1985, 527–537.

[19] M. L. Balinski, A competitive (dual) simplex method for the assignment problem, *Mathematical Programming* **34**, 1986, 125–141.

[20] M. L. Balinski and R. E. Gomory, A primal method for the assignment and transportation problems, *Management Science* **10**, 1964, 578–593.

[21] M. L. Balinski and J. Gonzalez, Maximum matchings in bipartite graphs via strong spanning trees, *Networks* **21**, 1991, 165–179.

[22] M. L. Balinski and A. Russakoff, On the assignment polytope, *SIAM Review* **16**, 1974, 516–525.

[23] S. E. Bammel and J. Rothstein, The number of 9×9 Latin squares, *Discrete Mathematics* **11**, 1975, 93–95.

[24] H.-J. Bandelt, Y. Crama, and F. C. R. Spieksma, Approximation algorithms for multidimensional assignment problems with decomposable costs, *Discrete Applied Mathematics* **49**, 1994, 25–50.

[25] V. A. Bardadym, Modifications of general lexicographic bottleneck optimization problems, *Proceedings of the 5-th Twente Workshop on Graphs and Combinatorial Optimization*, U. Faigle and C. Hoede, eds., 1997, University of Twente, Enschede, The Netherlands, 27–30.

[26] R. S. Barr, F. Glover, and D. Klingman, The alternating basis algorithm for assignment problems, *Mathematical Programming* **13**, 1977, 1–13.

[27] R. S. Barr and B. L. Hickman, A new parallel network simplex algorithm and implementation for large time-critical problems, Technical Report, Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX, 1990.

[28] D. P. Bertsekas, A new algorithm for the assignment problem, *Mathematical Programming* **21**, 1981, 152–171.

[29] D. P. Bertsekas, The auction algorithm: A distributed relaxation method for the assignment problem, *Annals of Operations Research* **14**, 1988, 105–123.

[30] D. P. Bertsekas, *Linear Network Optimization: Algorithms and codes*, MIT Press, Cambridge, MA, 1991.

[31] D. P. Bertsekas and D. A. Castañon, Parallel synchronous and asynchronous implementations of the auction algorithm, *Parallel Computing* **17**, 1991, 707–732.

[32] D. P. Bertsekas and D. A. Castañon, Parallel asynchronous Hungarian methods for the assignment problem, *ORSA Journal on Computing* **5**, 1993, 661–674.

[33] D. P. Bertsekas and D. A. Castañon, Parallel primal-dual methods to the minimum cost network flow problem, *Computational Optimization and Applications* **2**, 1993, 319–338.

[34] D. P. Bertsekas, D. A. Castañon, J. Eckstein, and S. Zenios, Parallel computing in network optimization, in *Network Models - Handbooks in Operations Research and Management Science*, Vol. **7**, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, eds., Elsevier, Amsterdam, The Netherlands, 1995, pp. 330–399.

[35] D. P. Bertsekas and J. Eckstein, Dual coordinate step methods for linear network flow problems, *Mathematical Programming* **42**, 1988, 203–243.

[36] G. Birkhoff, Tres observaciones sobre el algebra lineal, *Rev. univ. nac. Tucumán (A)* **5**, 1946, 147–151.

[37] W. L. Brogan, Algorithm for ranked assignments with applications to multiobject tracking, *Journal of Guidance* **12**, 1989, 357–364.

[38] R. E. Burkard, Time-slot assignment for TDMA-systems, *Computing* **35**, 1985, 99–112.

[39] R. E. Burkard and U. Derigs, *Assignment and Matching Problems: Solution Methods with FORTRAN Programs*, Springer, Berlin, 1980.

[40] R. E. Burkard and K. Fröhlich, Some remarks on 3-dimensional assignment problems, *Methods of Operations Research* **36**, 1980, 31–36.

[41] R. E. Burkard, W. Hahn, and U. Zimmermann, An algebraic approach to assignment problems, *Mathematical Programming* **12**, 1977, 318–327.

[42] R. E. Burkard, B. Klinz, and R. Rudolf, Perspectives of Monge properties in optimization, *Discrete Applied Mathematics* **70**, 1996, 95–161.

[43] R. E. Burkard and F. Rendl, Lexicographic bottleneck problems, *Operations Research Letters* **10**, 1991, 303–308.

[44] R. E. Burkard and R. Rudolf, Computational investigations on 3-dimensional axial assignment problems, *Belgian J. of Operations Research* **32**, 1993, 85–98.

[45] R. E. Burkard, R. Rudolf, and G. J. Woeginger, Three dimensional axial assignment problems with decomposable cost coefficients, *Discrete Applied Mathematics* **65**, 1996, 123–169.

[46] R. E. Burkard and U. Zimmermann, Weakly admissible transformations for solving algebraic assignment and transportation problems, *Mathematical Programming Study* **12**, 1980, 1–18.

[47] R. E. Burkard and U. Zimmermann, Combinatorial optimization in linearly ordered semimodules: a survey, in *Modern Applied Mathematics*, B. Korte ed., North Holland, Amsterdam, 1982, pp. 392–436.

[48] P. Camerini, L. Fratta, and F. Maffioli, On improving relaxation methods by modified gradient techniques, *Mathematical Programming Study* **3**, 1975, 26–34.

[49] P. Carraresi and G. Gallo, Network models for vehicle and crew scheduling, *European Journal of Operational Research* **16**, 1984, 139–151.

[50] P. Carraresi and G. Gallo, A multi-level bottleneck assignment approach to the bus drivers' rostering problem, *European Journal of Operational Research* **16**, 1984, 163–173.

[51] G. Carpaneto, S. Martello, and P. Toth, Algorithms and codes for the assignment problem, *Annals of Operations Research* **13**, 1988, 193–223.

[52] P. Carraresi and C. Sodini, An efficient algorithm for the bipartite matching problem, *European Journal of Operational Research* **23**, 1986, 86–93.

[53] D. A. Castañon, B. Smith, and A. Wilson, Performance of parallel assignment algorithms on different multiprocessor architectures, Technical Report TP-1245, ALPHATECH, Inc., Burlington, Mass.

[54] G. Carpaneto and P. Toth, Solution of the assignment problem, *ACM Transactions on Mathematical Software* **6**, 1980, 104–11.

[55] G. Carpaneto and P. Toth, Algorithm for the solution of the bottleneck assignment problem, *Computing* **27**, 1981, 179–187.

[56] G. Carpaneto and P. Toth, Primal-dual algorithms for the assignment problem, *Discrete Applied Mathematics* **18**, 1987, 137–153.

[57] K. Cechlárová, The uniquely solvable bipartite matching problem, *Operations Research Letters* **10**, 1991, 221–224.

[58] B. V. Cherkassky and A. V. Goldberg, On implementing push-relabel methods for the maximum flow problem, *Algorithmica* **19**, 1997, 390–410.

[59] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, Shortest paths algorithms: theory and experimental evaluation, *Mathematical Programming* **73**, 1996, 129–174.

[60] D. Coppersmith and S. Vinograd, Matrix multiplication via arithmetic progressions, *Journal of Symbolic Computing* **9**, 1990, 251–280.

[61] Y. Crama and F. C. R. Spieksma, Approximation algorithms for three-dimensional assignment problems with triangle inequalities, *European Journal of Operational Research* **60**, 1992, 273–279.

[62] W. H. Cunningham, A network simplex method, *Mathematical Programming* **11**, 1976, 105–116.

[63] W. H. Cunningham, Theoretical properties of the network simplex method, *Mathematics of Operations Research* **4**, 1979, 196–208.

[64] W. H. Cunningham and A. B. Marsh, A primal algorithm for optimum matching, *Mathematical Programming Study* **8**, 1978, 50–72.

[65] G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.

[66] V. G. Deĭneko and V. L. Filonenko, On the reconstruction of specially structured matrices, *Aktualnyje Problemy EVM i Programmirovanije*, Dnjepropetrovsk, DGU, 1979, (in Russian).

[67] U. Derigs, Alternate strategies for solving bottleneck assignment problems - analysis and computational results, *Computing* **33**, 1984, 95–106.

[68] The shortest augmenting path for solving assignment problems - motivation and computational experience, in *Algorithms and Software for Optimization- Part I*, *Annals of Operations Research* **4**, C. L. Monma ed., Baltzer, Basel, 1985, 57–102.

[69] U. Derigs, O. Goecke, and R. Schrader, Monge sequences and a simple assignment algorithm, *Discrete Applied Mathematics* **15**, 1986, 241–248.

[70] U. Derigs and U. Zimmermann, An augmenting path method for solving linear bottleneck assignment problems, *Computing* **19**, 1978, 285–295.

[71] E. W. Dijkstra, A note on two problems in connection with graphs, *Numerische Mathematik* **1**, 1959, 269–271.

[72] W. E. Donath, Algorithms and average-value bounds for assignment problems, *IBM Journal on Research Development* **13**, 1969, 380–386.

[73] J. Edmonds and D. R. Fulkerson, Bottleneck extrema, *Journal of Combinatorial Theory* **8**, 1970, 299-306.

[74] P. Erdös and A. Rényi, On random matrices, *Pub. Math. Inst. Hung. Acad. of Sciences* **8A**, 1963, 455-461.

[75] R. Euler, Odd cycles and a class of facets of the axial 3-index assignment polytope, *Applicationes mathematicae (Zastowania Matematyki)* **19**, 1987, 375–386.

[76] R. Euler, R. E. Burkard, and R. Grommes, On Latin squares and the facial structure of related polytopes, *Discrete Mathematics* **62**, 1986, 155–181.

[77] R. Euler and H. Le Verge, Time-tables, polyhedra and the greedy algorithm, *Discrete Applied Mathematics* **65**, 1996, 207–221.

[78] T. A. Ewashko and R. C. Dudding, Application of Kuhn's Hungarian assignment algorithm to posting servicemen, *Operations Research* **19**, 1971, 991.

[79] D. Fortin and R. Rudolf, Weak algebraic Monge arrays, to appear in *Discrete Mathematics*, 1998.

[80] M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *Journal of the ACM* **34**, 1987, 596–615.

[81] J. B. G. Frenk, M. van Houweninge, and A. H. G. Rinnooy Kan, Order statistics and the linear assignment problem, *Computing* **39**, 1987, 165–174.

[82] A. M. Frieze, Complexity of a 3-dimensional assignment problem, *European Journal of Operational Research* **13**, 1983, 161–164.

[83] A. M. Frieze and L. Yadegar, An algorithm for solving 3-dimensional assignment problems with application to scheduling in a teaching practice, *Journal of the Operational Research Society* **32**, 1981, 989–995.

[84] R. Fulkerson, I. Glicksberg, and O. Gross, A production line assignment problem, Technical Report RM–1102, The Rand Corporation, Sta. Monica, CA, 1953.

[85] L. R. Ford and D. R. Fulkerson, Maximal flow through a network, *Canadian Journal of Mathematics* **8**, 1956, 399–404.

[86] H. N. Gabow and R. E. Tarjan, A linear time algorithm for a special case of set union, *Journal of Computer and System Sciences* **30**, 1985, 209–221.

[87] H. N. Gabow and R. E. Tarjan, Algorithms for two bottleneck optimization problems, *Journal of Algorithms* **9**, 1988, 411–417.

[88] R. Garfinkel, An improved algorithm for the bottleneck assignment problem, *Operations Research* **19**, 1971, 1747–1751.

[89] F. Glover, Maximum matching in a convex bipartite graph, *Naval Research Logistics Quarterly* **14**, 1967, 313–316.

[90] F. Glover, R. Glover, and D. Klingman, Threshold assignment algorithm, *Mathematical Programming Study* **26**, 1986, 12–37.

[91] F. Glover, D. Karney, and D. Klingman, Implementation and computational study on start procedures and basis change criteria for a primal network code, *Networks 4*, 1974, 191–212.

[92] F. Glover and D. Klingman, Improved labeling of L.P. bases in networks, Research report CS 218, Center for Cybernetic Studies, University of Texas, Austin, TX, 1974.

[93] M. X. Goemans and M. Kodilian, A lower bound on the expected value of an optimal assignment, *Mathematics of Operations Research* **18**, 1993, 267–274.

[94] A. V. Goldberg and R. Kennedy, An efficient cost scaling algorithm for the assignment problem, *Mathematical Programming* **75**, 1995, 153–177.

[95] A. V. Goldberg, S. A. Plotkin, and P. Vaidya, Sublinear-time parallel algorithms for matching and related problems, *Journal of Algorithms* **14**, 1993, 180–213.

[96] A. V. Goldberg and R. E. Tarjan, Finding minimum-cost circulations by successive approximation, *Mathematics of Operations Research* **15**, 1990, 430–466.

[97] D. Goldfarb, Efficient dual simplex methods for the assignment problem, *Mathematical Programming* **33**, 1985, 187–203.

[98] O. Gross, The bottleneck assignment problem, Technical Report P–1630, The Rand Corporation, Sta. Monica, CA, 1959.

[99] Ph. Hall, On representatives of subsets, *Journal of the London Mathematical Society* **10**, 1935, 26–30.

[100] P. Hansen and L. Kaufman, A primal-dual algorithm for the three-dimensional assignment problem, *Cahiers du CERO* **15**, 1973, 327–336.

[101] G. H. Hardy, J. E. Littlewood, and G. Pólya, *Inequalities*, Cambridge University Press, London and New York, 1952.

[102] A. J. Hoffman, On simple linear programming problems, in *Convexity*, *Proceedings of Symposia in Pure Mathematics* **7**, V. Klee ed., AMS, Providence, RI, 1963, 317–327.

[103] J. E. Hopcroft and R. M. Karp, An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs, *SIAM Journal on Computing* **2**, 1973, 225–231.

[104] M. S. Hung, A polynomial simplex method for the assignment problem, *Operations Research* **31**, 1983, 595–600.

[105] M. S. Hung and W. D. Rom, Solving the assignment problem by relaxation, *Operations Research* **28**, 1980, 969–982.

[106] D. S. Johnson and C. C. McGeoch, eds., *Network Flows and Matching - First DIMACS Implementation Challenge*, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **12**, AMS, Providence, RI, 1993.

[107] R. Jonker and A. Volgenant, Improving the Hungarian assignment algorithm, *Operations Research Letters* **5**, 1986, 171–175.

[108] R. Jonker and A. Volgenant, A shortest augmenting path algorithm for dense and sparse linear assignment problems, *Computing* **38**, 1987, 325–340.

[109] R. M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972, 85–103.

[110] R. M. Karp, An algorithm to solve the $m \times n$ assignment problem in expected time $O(mn \log n)$, *Networks* **10**, 1980, 143–152.

[111] R. M. Karp, An upper bound on the expected cost of an optimal assignment, in *Discrete Algorithms and Complexity*, Academic Press, Boston, 1987, 1–4.

[112] R. M. Karp, A. H. G. Rinnooy Kan, and R. V. Vohra, Average case analysis of a heuristic for the assignment problem, *Mathematics of Operations Research* **19**, 1994, 513–522.

[113] J. Kennington and Z. Wang, Solving dense assignment problems on a shared memory multiprocessor, Report 88-OR-16, Department of Operations Research and Applied Science, Souther Methodist University, Dallas, TX, 1998.

[114] P. Kleinschmidt, C. W. Lee, and H. Schannath, Transportation problem which can be solved by the use of Hirsch paths for the dual problems, *Mathematical Programming* **37**, 1987, 153–168.

[115] B. Klinz, R. Rudolf and G. J. Woeginger, On the recognition of permuted bottleneck Monge matrices, *Discrete Applied Mathematics* **60**, 1995, 223–248.

[116] D. König, Graphok és matrixok, *Mat. Fiz. Lapok* **38**, 1931, 116–119.

[117] J. M. Kurzberg, On approximation methods for the assignment problem, *Journal of the ACM* **9**, 1962, 419–439.

[118] H. W. Kuhn, The Hungarian method for the assignment and transportation problems, *Naval Research Logistics Quarterly* **2**, 1955, 83–97.

[119] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, New York, 1976.

[120] A. J. Lazarus, Certain expected values in the random assignment problem, *Operations Research Letters* **14**, 1993, 207–214.

[121] Y. Lee and J. B. Orlin, On very large scale assignment problems, in *Large Scale Optimization: State of the Art*, W. W. Hager, D. W. Hearn, and P. M. Pardalos, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994, pp. 206–244.

[122] R. E. Machol, An application of the assignment problem, *Operations Research* **18**, 1970, 745–746.

[123] D. Magos, Tabu search for the planar three-index assignment problem, *Journal of Global Optimization* **8**, 1996, 35–48.

[124] D. Magos and P. Miliotis, An algorithm for the planar three-index assignment problem, *European Journal of Operational Research* **77**, 1994, 141-153.

[125] S. Martello, W. R. Pulleyblank, P. Toth, and D. de Werra, Balanced optimization problems, *Operations Research Letters* **3**, 1984, 275–278.

[126] S. Martello and P. Toth, Linear assignment problems, in *Surveys in Combinatorial Optimization*, *Annals of Discrete Mathematics* **31**, S. Martello, G. Laporte, M. Minoux, and C. Ribeiro, eds., North-Holland, Amsterdam, 1987, pp. 259–282.

[127] M. Mézard and G. Parisi, On the solution of the random link matching problems, *Journal de Physique* **48**, 1987, 1451–1459.

[128] D. Miller, J. Pekny, and G. L. Thompson, Solution of large dense transportation problems using a parallel primal algorithm, *Operations Research Letters* **9**, 1990, 319–324.

[129] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani, Matching is as easy as matrix inversion, *Combinatorica* **7**, 1087, 105–113.

[130] B. Neng, Zur Erstellung von optimalen Triebfahrzeugplänen, *Zeitschrift für Operations Research* **25**, 1981, B159–B185.

[131] B. Olin, Asymptotic Properties of Random Assignment Problems, Ph.D. Thesis, Division of Optimization and Systems Theory, Department of Mathematics, Royal Institute of Technology, Stockholm, 1992.

[132] J. B. Orlin, On the simplex algorithm for networks and generalized networks, *Mathematical Programming Studies* **24**, 1985, 166–178.

[133] J. B. Orlin and R. K. Ahuja, New scaling algorithms for the assignment and minimum cycle mean problems, *Mathematical Programming* **54**, 1992, 41–56.

[134] E. S. Page, A note on assignment problems, *Computer Journal* **6**, 1963, 241–243.

[135] K. Paparrizos, A non-dual signature method for the assignment problem and a generalization of the dual simplex method for the transportation problem, *RAIRO Operations Research* **22**, 1988, 269–289.

[136] K. Paparrizos, A relaxation column signature method for assignment problems, *European Journal of Operational Research* **50**, 1991, 211–219.

[137] K. Paparrizos, An infeasible (exterior point) simplex algorithm for assignment problems, *Mathematical Programming* **51**, 1991, 45–54.

[138] J. Peters, The network simplex method on a multiprocessor, *Networks* **20**, 1990, 845–859.

[139] U. Pferschy, The random linear bottleneck assignment problem, *RAIRO Operations Research* **30**, 1996, 127–142.

[140] U. Pferschy, Solution methods and computational investigations for the linear bottleneck assignment problem, *Computing* **59**, 1997, 237–258.

[141] C. Phillips and S. Zenios, Experiences with large scale network optimization on the connection machine, in *The Impact of Recent Computing Advances on Operations Research*, *Operations Research Series* **9**, Elsevier, 1989, pp. 169–180.

[142] W. P. Pierskalla, The tri-substitution method for the three-multidimensional assignment problem, *Canadian ORS Journal* **5**, 1967, 71–81.

[143] W. P. Pierskalla, The multidimensional assignment problem. *Operations Research* **16**, 1968, 422–431.

[144] A. B. Poore, Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking, *Computation Optimization and Application* **3**, 1994, 27–54.

[145] A. B. Poore, A numerical study of some data association problems arising in multitarget tracking, in *Large Scale Optimization: State of the Art*, W. W. Hager, D. W. Hearn, and P. M. Pardalos, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994, pp. 339–361.

[146] A. B. Poore and N. Rijavec, Partitioning multiple data sets: multidimensional assignments and Lagrangian relaxation, in *Quadratic assignment and related problems*, P. M. Pardalos and H. Wolkowicz, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **16**, AMS, Providence, RI, 1994, pp. 317–342.

[147] A. B. Poore, N. Rijavec, M. Liggins, and V. Vannicola, Data association problems posed as multidimensional assignment problems: problem formulation, in *Signal and Data Processing of Small Targets*, O. E. Drummond ed., SPIE, Bellingham, WA, 1993, pp. 552–561.

[148] A. B. Poore and A. J. Robertson III, A new Lagrangean relaxation based algorithm for a class of multidimensional assignment problems, *Computational Optimization and Applications* **8**, 1997, 129–150.

[149] J. Pusztaszeri, P. E. Rensing, and T. M. Liebling, Tracking elementary particles near their primary vertex: a combinatorial approach, *Journal of Global Optimization* **16**, 1995, 422–431.

[150] A. P. Punnen and K. P. K. Nair, Improved complexity bound for the maximum cardinality bottleneck bipartite matching problem, *Discrete Applied Mathematics* **55**, 1994, 91–93.

[151] L. Qi, E. Balas, and G. Gwan, A new facet class and a polyhedral method for the three-index assignment problem, in *Advances in Optimization*, D.-Z. Du ed., Kluwer Academic Publishers, 1994, pp. 256–274.

[152] K. G. Ramakrishnan, N. K. Karmarkar, and A. P. Kamath, An approximate dual projective algorithm for solving assignment problems, in *Network flows and matching–First DIMACS Implementation Challenge*, D. S. Johnson and C. C. McGeoch, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **12**, AMS, Providence, RI, 1993, pp. 431–449.

[153] F. Rendl, On the complexity of decomposing matrices arising in satellite communication, *Operations Research Letters* **4**, 1985, 5–8.

[154] E. Roohy-Laleh, Improvements to the theoretical efficiency of the network simplex method, Ph.D. Thesis, Carleton University, Ottawa, 1980.

[155] G. Rote and F. Rendl, Minimizing the density in lay-out design, *Operations Research Letters* **5**, 1986, 111–118.

[156] J. Shao and W. Wei, A formula for the number of Latin squares, *Discrete Mathematics* **110**, 1992, 293–296.

[157] J. T. Schwarz, Fast probabilistic algorithms for verification of polynomial identities, *Journal of the ACM* **27**, 1980, 701–717.

[158] V. Srinivasan and G. L. Thompson, Cost operator algorithms for the transportation problem, *Mathematical Programming* **12**, 1977, 372–391.

[159] R. E. Tarjan, Data Structures and Network Algorithms, SIAM, Philadelphia, PA, 1983.

[160] G. L. Thompson, A recursive method for solving assignment problems, in *Studies on Graphs and Discrete Programming*, *Annals of Discrete Mathematics* **11**, P. Hansen ed., North Holland, Amsterdam, 1981, 319–343.

[161] W. T. Tutte, The factorization of linear graphs, *Journal of the London Mathematical Society* **22**, 1947, 107–111.

[162] L. G. Valiant, The complexity of computing the permanent, *Theoretical Computer Science* **8**, 1979, 189–201.

[163] M. Vlach, Branch and bound method for the three-index assignment problem, *Ekonomicko-Matematicky Obzor* **12**, 1967, 181–191.

[164] A. Volgenant, Linear and semi-assignment problems: A core oriented approach, *Computers of Operations Research* **23**, 1996, 917–932.

[165] D. W. Walkup, On the expected value of a random assignment problem, *SIAM Journal on Computing* **8**, 1979, 440–442.

[166] D. W. Walkup, Matching in random regular bipartite digraphs, *Discrete Mathematics* **31**, 1980, 59–64.

[167] J. Wein and S. Zenios, Massively parallel auction algorithms for the assignment problem, *Proceedings of the 3-rd Symposium on the Frontiers of Massively Parallel Computations*, 1990, pp. 90–99.

[168] J. Wein and S. Zenios, On the massively parallel solution of the assignment problem, *Journal of the Parallel and Distributed Computing* **13**, 1991, 221–236.

[169] G. J. Woeginger, private communication.

[170] H. Zaki, A comparison of two algorithms for the assignment problem, Technical Report ORL 90-002, Department of Mechanical and industrial Engineering, University of Illinois, Champaign-Urbana, IL, 1990.