SOMETIMES TRAVELLING IS EASY: THE MASTER TOUR PROBLEM*

VLADIMIR G. DEINEKO[†], RÜDIGER RUDOLF[†], AND GERHARD J. WOEGINGER[†]

Abstract. In 1975, Kalmanson proved that if the distance matrix in the travelling salesman problem (TSP) fulfills certain combinatorial conditions (that are nowadays called the *Kalmanson conditions*) then the TSP is solvable in polynomial time [*Canad. J. Math.*, 27 (1995), pp. 1000–1010].

We deal with the problem of deciding, for a given instance of the TSP, whether there is a renumbering of the cities such that the corresponding renumbered distance matrix fulfills the Kalmanson conditions. Two results are derived: first, it is shown that—in case it exists—such a renumbering can be found in polynomial time. Secondly, it is proved that such a renumbering exists if and only if the instance possesses the so-called *master tour* property. A recently posed question by Papadimitriou is thereby answered in the negative.

Key words. travelling salesman problem, Kalmanson condition, master tour, combinatorial optimization

AMS subject classifications. 08C85, 68Q20, 05C38, 68R10, 90C35

PII. S0895480195281878

1. Introduction. The *travelling salesman problem* (TSP) is defined as follows. Given an $n \times n$ distance matrix $C = (c_{ij})$, find a permutation $\pi \in S_n$ that minimizes the sum $\sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)} + c_{\pi(n)\pi(1)}$. In other words, the salesman must visit cities 1 to n in arbitrary order and want to minimize the total travel length. This problem is one of the fundamental problems in combinatorial optimization and known to be NP hard. For more specific information on the TSP, the reader is referred to the book by Lawler et al. [7].

In this paper, we are interested in a special case of the TSP where—due to special combinatorial structures in the distance matrix—the problem is solvable in polynomial time: the case of *Kalmanson* distance matrices. A symmetric $n \times n$ matrix C is called a *Kalmanson* matrix if it fulfills the conditions

$$(1.1) \quad c_{ij} + c_{k\ell} \leq c_{ik} + c_{j\ell} \text{ for all } 1 \leq i < j < k < \ell \leq n,$$

$$(1.2) \quad c_{i\ell} + c_{jk} \leq c_{ik} + c_{j\ell} \text{ for all } 1 \leq i < j < k < \ell \leq n.$$

Note that these conditions do not involve any diagonal entries c_{ii} . Since every city is visited only once, diagonal entries are of no relevance for the TSP and may as well be considered to be “undefined” or zero. Originally, Kalmanson introduced these conditions in order to generalize the concept of *convexity* of finite point sets in the plane: for some convex planar point set, let $p_1, \dots, p_n$ denote its clockwise ordering around the convex hull. Then the Euclidean distance matrix $c_{ij} = d(p_i, p_j)$ fulfills all conditions (1.1) and (1.2) (Proof: In a convex quadrangle, the total length of the diagonals is greater or equal to the total length of two opposite sides.) Moreover, if we “rotate” the ordering by one point, the distance matrix of the resulting *rotated* point

*Received by the editors February 21, 1995; accepted for publication (in revised form) January 6, 1996. This research has been supported by the Spezialforschungsbereich F 003 “Optimierung und Kontrolle,” Projektbereich Diskrete Optimierung.

<http://www.siam.org/journals/sidma/11-1/28187.html>

[†]Institut für Mathematik B, Steyrergasse 30, A-8010 Graz, Austria (deineko@opt.math.tu-graz.ac.at, rudolf@opt.math.tu-graz.ac.at, gwoegi@opt.math.tu-graz.ac.at).

sequence $p_2, p_3, \dots, p_n, p_1$ also is a Kalmanson matrix. It is easy to verify that this “rotation property” does not result from special Euclidean features but solely from inequalities (1.1) and (1.2). Hence, if one removes the first row and first column from a Kalmanson matrix and appends them after the last row and column, the result of this operation is another Kalmanson matrix. Similarly, reversing the ordering of the rows and columns of a Kalmanson matrix will again yield a Kalmanson matrix.

Kalmanson [6] proved that for the TSP with a Kalmanson distance matrix, the identity permutation $\langle 1, 2, 3, \dots, n \rangle$ always constitutes an optimal tour and thus, the TSP is easily solved for this special case. Observe that the length of the optimum TSP tour is not changed when the cities are renumbered, i.e., when the rows and columns of the distance matrix are permuted according to the same permutation. However, such a renumbering will usually destroy the Kalmanson conditions. Intuition tells us that a renumbered instance is still a rather trivial special case of the TSP, since it is just a Kalmanson instance in disguise, but it is by no means obvious how to recognize this disguise. Hence, the problem arises of finding a permutation that transforms the distance matrix back into a Kalmanson matrix.

Another problem related to the TSP is the detection of a master tour, motivated by the following observation. Suppose that all cities in a Euclidean instance of the TSP are the vertices of a convex polygon. Then the optimum tour is not only easy to find (it is the perimeter of the polygon), but the instance also fulfills the much stronger *master tour* property: there is an optimum TSP tour π such that the optimum TSP tour of any subset of cities can be obtained by simply omitting from the tour π the cities that are not in the subset. Such a tour π is called a master tour. The concept of a master tour was first formulated by Papadimitriou [8, 9]. It is easy to prove that deciding whether a given instance of the TSP has the master tour property is in the complexity class $\Sigma_2\mathbf{P}$. Papadimitriou also considered the corresponding decision problem as a “good candidate for a natural $\Sigma_2\mathbf{P}$ -complete problem.” In this paper, we will prove that the following results hold true.

- (1) For a symmetric $n \times n$ matrix C , it can be decided in $O(n^2 \log n)$ time whether C is a permuted Kalmanson matrix.
- (2) A distance matrix allows a master tour if and only if it is a permuted Kalmanson matrix.

Combining results (1) and (2) yields a polynomial-time algorithm for the master tour problem. Hence, unless $\Sigma_2\mathbf{P} = \mathbf{P}$, the conjecture of Papadimitriou is false.

Organization of the paper. Section 2 summarizes elementary definitions and results on permutations and matrices. In section 3, several lemmas on the combinatorial structure of Kalmanson matrices are collected. These lemmas are used in section 4 to derive an $O(n^2 \log n)$ -time algorithm for recognizing permuted $n \times n$ Kalmanson matrices. Section 5 explains the connection between permuted Kalmanson matrices and master tours and shows that a master tour can be detected in polynomial time. Finally, section 6 closes with a short discussion.

2. Definitions and preliminaries. In this section, several basic definitions for permutations and matrices are summarized.

For an $n \times n$ matrix C , denote by $I = \{1, \dots, n\}$ the set of rows (columns). A row i *precedes* a row j in C ($i \prec j$ for short), if row i occurs before row j in C . For two sets K_1 and K_2 of rows, we write $K_1 \prec K_2$ if and only if $k_1 \prec k_2$ for all $k_1 \in K_1$ and $k_2 \in K_2$. Let $V = \{v_1, v_2, \dots, v_r\}$ and $W = \{w_1, w_2, \dots, w_s\}$ be two subsets of I . We denote by $C[V, W]$ the $r \times s$ submatrix of C that is obtained by deleting all rows not contained in V and all columns not in W .

For permutations, we adopt the notation $\pi = \langle x_1, x_2, \dots, x_n \rangle$ for “ $\pi(i) = x_i$ for $1 \leq i \leq n$.” The *concatenation* of permutations $\langle x_1, \dots, x_n \rangle$ and $\langle y_1, \dots, y_m \rangle$ is $\langle z_1, \dots, z_{n+m} \rangle$, where $z_i = x_i$ for $1 \leq i \leq n$ and $z_{n+j} = y_j$ for $1 \leq j \leq m$. The identity permutation is denoted by ε , i.e., $\varepsilon(i) = i$ for all $i \in I$. For a permutation ϕ , the permutation ϕ^- defined by $\phi^-(i) = \phi(n - i + 1)$ is called the *reverse permutation* of ϕ . Permutation ϕ is called a *cyclic shift* or a *rotation* if there exists a $k \in I$ such that $\phi = \langle k, k + 1, \dots, n, 1, \dots, k - 1 \rangle$.

By $C_{\phi, \pi}$ we denote the matrix which is obtained from matrix C by permuting its rows according to ϕ and its columns according to π , i.e., $C_{\phi, \pi} = (c_{\phi(i), \pi(j)})$. For $C_{\phi, \phi}$, we usually write C_ϕ . A permutation ϕ is called a *Kalmanson permutation for some matrix C* if C_ϕ is a Kalmanson matrix. A matrix C is called a *permuted Kalmanson matrix* if there exists a Kalmanson permutation for C .

For a partition $V = \langle V_1, \dots, V_v \rangle$ of I into v subsets, the set $\text{STR}(V_1, \dots, V_v)$ is defined to contain all permutations ϕ that fulfill $\phi(v_i) \prec \phi(v_j)$ for all $v_i \in V_i$ and $v_j \in V_j$ with $1 \leq i < j \leq v$. $\text{STR}(V_1, \dots, V_v)$ is called the set of permutations induced by the sequence of *stripes* $V_1, \dots, V_v$. An appropriate data structure for storing, manipulating, and intersecting such sets of permutations are *PQ trees* as introduced by Booth and Lueker [1] (in fact, PQ trees of *height two* suffice to represent these permutations).

PROPOSITION 2.1 (see Booth and Lueker [1]). *For two partitions $\langle U_1, \dots, U_u \rangle$ and $\langle V_1, \dots, V_v \rangle$ of I , the set $\text{STR}(U_1, \dots, U_u) \cap \text{STR}(V_1, \dots, V_v)$ either equals $\text{STR}(W_1, \dots, W_w)$ for an appropriate partition $W = \langle W_1, \dots, W_w \rangle$ of I or it is empty. The partition W can be computed in $O(|I|)$ time.*

An $m \times n$ matrix C is called a *sum-matrix* if there exist numbers $x_1, \dots, x_m$ and $y_1, \dots, y_n$ such that $c_{ij} = x_i + y_j$ for all i and j . Note that this implies $c_{ij} + c_{rs} = c_{is} + c_{rj}$ for $1 \leq i < r \leq m$ and $1 \leq j < s \leq n$ (i.e. in any two by two submatrix, both diagonals have equal sums). For convenience, single rows and columns are also considered to be sum-matrices. Note that every sum-matrix is a Kalmanson matrix and a Contra Monge matrix. An $m \times n$ matrix C is called a *Contra Monge matrix* if $c_{ij} + c_{rs} \geq c_{is} + c_{rj}$ holds for $1 \leq i < r \leq m$ and $1 \leq j < s \leq n$. The combinatorial structure of Contra Monge matrices and of permuted Contra Monge matrices is well understood (see the original paper by Deineko and Filonenko [4] or the survey paper by Burkard, Klinz, and Rudolf [2]). The known main results are summarized in the following proposition.

PROPOSITION 2.2. *Let $X = (x_{ij})$ be an $m \times n$ matrix. Let $\Pi \subseteq S_m \times S_n$ denote the set of all pairs of permutations (π, ϕ) such that $X_{\pi, \phi}$ is a Contra Monge matrix.*

- (i) *Then either Π is the empty set, or there exists an appropriate partition $R_1, \dots, R_r$ of the set R of rows and an appropriate partition $C_1, \dots, C_c$ of the set C of columns of X such that*

$$\Pi = \{(\pi, \phi) | \pi \in \Pi_R, \phi \in \Pi_C\} \cup \{(\pi, \phi) | \pi^- \in \Pi_R, \phi^- \in \Pi_C\}$$

where $\Pi_R = \text{STR}(R_1, \dots, R_r)$ and $\Pi_C = \text{STR}(C_1, \dots, C_c)$.

- (ii) *The partitions $R_1, \dots, R_r$ and $C_1, \dots, C_c$ can be computed in $O(mn + m \log m + n \log n)$ time (in case they exist).*
- (iii) *Every submatrix $X[R_i, C]$ and every submatrix $X[R, C_j]$ is a sum-matrix. These sum-matrices are maximal sum-matrices in X (i.e., neither rows nor columns may be added without destroying the sum-matrix property).*
- (iv) *In case Π is not empty, either $r = c = 1$ holds (and X is a sum-matrix), or the numbers r and c are both at least two (and X is horizontally and vertically divided into several stripes by Π).*

- (v) *Matrix X is a Contra Monge matrix if and only if for all pairs of indices $1 \leq i \leq m-1$ and $1 \leq j \leq n-1$, the inequality*

$$(2.1) \quad x_{ij} + x_{i+1,j+1} \geq x_{i,j+1} + x_{i+1,j}$$

is fulfilled.

By $\mathbf{K}$ we denote the set of Kalmanson matrices. Similarly to the above alternate characterization (2.1) of Contra Monge matrices, an alternate characterization of Kalmanson matrices can be given.

PROPOSITION 2.3. *An $n \times n$ symmetric matrix C is a Kalmanson matrix if*

$$(2.2) \quad c_{i,j+1} + c_{i+1,j} \leq c_{ij} + c_{i+1,j+1} \quad \text{for all } 1 \leq i \leq n-3, i+2 \leq j \leq n-1,$$

$$(2.3) \quad c_{i,1} + c_{i+1,n} \leq c_{in} + c_{i+1,1} \quad \text{for all } 2 \leq i \leq n-2.$$

Observe that conditions (2.2) and (2.3) can be verified in $O(n^2)$ time. This yields the following proposition.

PROPOSITION 2.4. *For a symmetric $n \times n$ matrix C , it can be decided in $O(n^2)$ time whether C is a Kalmanson matrix.*

PROPOSITION 2.5. *Let C be an $n \times n$ Kalmanson matrix, let $I' \subseteq I$, and let $\phi \in S_n$ be a cyclic shift. Then (i) $C_{\varepsilon^-} \in \mathbf{K}$, (ii) $C[I', I'] \in \mathbf{K}$, and (iii) $C_\phi \in \mathbf{K}$ hold.*

3. Combinatorial properties of Kalmanson matrices. In this section, we derive several technical lemmas on the combinatorial structure of Kalmanson matrices.

LEMMA 3.1. *Let C be an $n \times n$ symmetric Kalmanson matrix, $2 \leq m \leq n-1$, let $V = \langle 1, 2, \dots, m \rangle$, and let $W = \langle m+1, \dots, n \rangle$. Then $C[V, W]$ is a Contra Monge matrix.*

Proof. This is a consequence of condition (2.2) in Proposition 2.3. $\square$

LEMMA 3.2. *Let C be a symmetric $n \times n$ matrix. Let V and W be a partition of I with $|V| = r \geq 2$, $|W| = s \geq 2$ such that $C[V, W]$ is a sum-matrix. Let $q \in W$ and $p \in V$ be arbitrary. Let $D = C[V \cup \{q\}, V \cup \{q\}]$ and $E = C[\{p\} \cup W, \{p\} \cup W]$. Assume that there is a permutation $\psi = \langle v_1, \dots, v_r, q \rangle$ of $V \cup \{q\}$ and a permutation $\pi = \langle p, w_1, \dots, w_s \rangle$ of $W \cup \{p\}$ such that $D_\psi \in \mathbf{K}$ and $E_\pi \in \mathbf{K}$.*

Under these conditions either $C_\phi \in \mathbf{K}$ for $\phi = \langle v_1, \dots, v_r, w_1, \dots, w_s \rangle$ or there does not exist any permutation $\sigma \in \text{STR}(V, W)$ with $C_\sigma \in \mathbf{K}$.

Proof. We prove that under the conditions in the lemma either the inequality

$$(3.1) \quad c_{v_1 v_r} + c_{w_1 w_s} \leq c_{v_1 w_1} + c_{v_r w_s}$$

is fulfilled and $C_\phi \in \mathbf{K}$ for $\phi = \langle v_1, \dots, v_r, w_1, \dots, w_s \rangle$ or inequality (3.1) is not fulfilled and there does not exist any permutation $\sigma \in \text{STR}(V, W)$ with $C_\sigma \in \mathbf{K}$.

First assume that inequality (3.1) is fulfilled. We prove that $C_\phi \in \mathbf{K}$ according to Proposition 2.3 by verifying conditions (2.2) and (2.3). Consider two indices i and j in C_ϕ , $1 \leq i \leq n-3$, and $i+2 \leq j \leq n-1$. In case $i \prec i+1 \prec j \prec j+1$ are all in V or are all in W , condition (2.2) holds for C_ϕ since $D_\psi \in \mathbf{K}$ and $E_\pi \in \mathbf{K}$. If i and $i+1$ are in V and j and $j+1$ are in W , the four elements $c_{i,j+1}$, $c_{i+1,j}$, c_{ij} , and $c_{i+1,j+1}$ lie in the sum-matrix $C[V, W]$ and thus trivially fulfill (2.2). Next, if i is in V and $i+1 \prec j \prec j+1$ are in W then $i = v_r$ and $i+1 = w_1$ holds. The relations $p \prec w_1 \prec j \prec j+1$ in $E_\pi \in \mathbf{K}$ yield $c_{p,j+1} + c_{w_1,j} \leq c_{pj} + c_{w_1,j+1}$. Since $p, v_r \in V$, $j, j+1 \in W$, and $C[V, W]$ is a sum-matrix, $c_{pj} + c_{v_r,j+1} = c_{p,j+1} + c_{v_r,j}$. Adding this equality to the previous inequality yields (2.2). The last case where $i \prec i+1 \prec j$ are

in V and $j + 1$ is in W is handled symmetrically. Summarizing, (2.2) is true in any case.

Next, consider an index $2 \leq i \leq n - 2$. In case $i \neq v_r$, (2.3) is true since $D_\psi \in \mathbf{K}$ (respectively, $E_\pi \in \mathbf{K}$) holds. In case $i = v_r$, (2.3) is exactly (3.1). Hence, (3.1) implies (2.3), and the first half of the lemma is proven.

To prove the remaining half, assume that for some $\sigma \in \text{STR}(V, W)$, $C_\sigma \in \mathbf{K}$ holds. We show how to derive inequality (3.1) from this. Since V precedes W in σ , only two cases arise:

(i) $v_1 \prec v_r \prec w_1 \prec w_s$ or $v_r \prec v_1 \prec w_s \prec w_1$ in σ . Then condition (1.1) yields (3.1).

(ii) $v_1 \prec v_r \prec w_s \prec w_1$ or $v_r \prec v_1 \prec w_1 \prec w_s$ in σ . Then condition (1.1) yields $c_{v_1 v_r} + c_{w_1 w_s} \leq c_{v_r w_1} + c_{v_1 w_s}$. Since $C[V, W]$ is a sum-matrix, $c_{v_1 w_s} + c_{v_r w_1} = c_{v_1 w_1} + c_{v_r w_s}$. Adding these two inequalities gives (3.1). $\square$

LEMMA 3.3. *Let C be a symmetric $n \times n$ matrix. Let $U_1, \dots, U_m$ be a partition of I such that $C[U_i, I \setminus U_i]$ is a sum-matrix for $1 \leq i \leq m$. Let u_i be an arbitrary element in U_i . Let π_i be a Kalmanson permutation for $C[U_i \cup \{u_{i+1}\}, U_i \cup \{u_{i+1}\}]$ (indices are taken modulo m , i.e., $u_{m+1} = u_1$) that has u_{i+1} as its last element. Let ϕ_i denote the permutation of U_i induced by π_i .*

Under these conditions either $C_\phi \in \mathbf{K}$ where ϕ is the concatenation of $\phi_1, \dots, \phi_m$ or there does not exist any Kalmanson permutation for C in $\text{STR}(U_1, \dots, U_m)$.

Proof. The proof is done by induction on the number t of stripes U_i with cardinality at least two. If $t = 0$, the statement trivially holds. Otherwise if $t \geq 1$, we may assume without loss of generality that $|U_1| \geq 2$. Moreover, we assume that there exists a Kalmanson permutation for C in $\text{STR}(U_1, \dots, U_m)$, since otherwise there is nothing to show.

Set $W = U_2 \cup \dots \cup U_m$ and consider the sets U'_i where $U'_1 = \{u_1\}$ and $U'_i = U_i$ for $2 \leq i \leq m$. By the induction assumption, the concatenation of $\langle u_1 \rangle, \phi_2, \dots, \phi_m$ is a Kalmanson permutation for $C[\{u_1\} \cup W, \{u_1\} \cup W]$. Set $V = U_1$. By the conditions of the lemma, the concatenation π_1 of ϕ_1 and $\langle u_2 \rangle$ is a Kalmanson permutation for $C[V \cup \{u_2\}, V \cup \{u_2\}]$. Moreover, the matrix $C[V, W]$ is a sum-matrix. Summarizing, all conditions for applying Lemma 3.2 with $p = u_1$ and $q = u_2$ are fulfilled. The statement in Lemma 3.2 yields that the concatenation ϕ is indeed a Kalmanson permutation and the inductive proof is complete. $\square$

The following notation is convenient. For two rows i and j of a matrix C , define the set

$$(3.2) \quad \mathcal{M}(i, j) = \{k \in I \setminus \{i, j\} \mid c_{ik} - c_{jk} = \min_{\ell \neq i, j} \{c_{i\ell} - c_{j\ell}\}\}.$$

Note that $C[\{i, j\}, \mathcal{M}(i, j)]$ is a sum matrix. In case $|\mathcal{M}(i, j)| = n - 2$ holds, $c_{ik} - c_{jk} = \text{const}$ for all $k \in I \setminus \{i, j\}$. Such a pair of rows is called *equivalent*, and this is denoted by $i \sim j$. We also define that every row is equivalent to itself.

LEMMA 3.4. *For any symmetric $n \times n$ matrix C , the relation $\sim$ is an equivalence relation.*

Proof. By definition, the relation $\sim$ is symmetric and reflexive. To prove that $\sim$ is transitive, consider $i, j_1, j_2 \in I$ with $j_1 \sim i$ and $i \sim j_2$. The goal is to show that $j_1 \sim j_2$, i.e., to show that for any $k, \ell \in I$ with $\{j_1, j_2\} \cap \{k, \ell\} = \emptyset$ the equality $(*)$ $c_{j_1 k} - c_{j_2 k} = c_{j_1 \ell} - c_{j_2 \ell}$ holds. If $i \notin \{k, \ell\}$ holds, we use $j_1 \sim i$ and $j_2 \sim i$: subtract the equalities $c_{ik} - c_{j_1 k} = c_{i\ell} - c_{j_1 \ell}$ and $c_{ik} - c_{j_2 k} = c_{i\ell} - c_{j_2 \ell}$ from each other, and derive $(*)$. Otherwise, assume that $k = i$. Use $i \sim j_2$ to obtain $c_{ij_1} - c_{j_2 j_1} = c_{i\ell} - c_{j_2 \ell}$

and use $i \sim j_1$ to obtain $c_{ij_2} - c_{j_1j_2} = c_{i\ell} - c_{j_1\ell}$. Subtracting these equations yields (*). $\square$

LEMMA 3.5. *Let C be a symmetric $n \times n$ matrix. If $1 \sim i$ for all $i \in I$, then $C \in \mathbf{K}$.*

Proof. By Lemma 3.4 above, all inequalities (1.1) and (1.2) are fulfilled with equality. $\square$

LEMMA 3.6. *Let C be a symmetric $n \times n$ Kalmanson matrix. Let i and j be two rows of C with $i \prec j$, let $K_1 = \mathcal{M}(i, j) \cup \{i\}$, and let $K_2 = I \setminus K_1$. Then there exists a cyclic shift ϕ such that $C_\phi \in \mathbf{K}$ and $K_1 \prec K_2$ in C_ϕ .*

Proof. By definition, $i \in K_1$ and $j \in K_2$. Consider any $k \in \mathcal{M}(i, j)$. Then $c_{ik} - c_{jk} = c_{i\ell} - c_{j\ell}$ for all $\ell \in K_1 \setminus \{i\}$ and $c_{ik} - c_{jk} < c_{i\ell} - c_{j\ell}$ for all $\ell \in K_2 \setminus \{j\}$. We distinguish the following three cases on the relative positions of i, j , and k in C .

- (i) $k \prec i \prec j$. Then condition (1.2) yields $c_{ip} - c_{jp} \leq c_{ik} - c_{jk}$ for any p with $k \prec p \prec i \prec j$. Hence, $p \in K_1$ for all $p \in I$ with $k \prec p \prec i$.
- (ii) $i \prec k \prec j$. Analogously to the argument in (i), condition (1.1) implies $p \in K_1$ for all $p \in I$ with $i \prec p \prec k$.
- (iii) $i \prec j \prec k$. Analogously to the argument in (i), conditions (1.2) and (1.2) imply $p \in K_1$ for all $p \in I$ with $p \prec i$ or $k \prec p$.

Summarizing, we conclude that there exist two elements r and s such that either $K_1 = \{r, \dots, i, \dots, s\}$ or $K_2 = \{s + 1, \dots, j, \dots, r - 1\}$. By Proposition 2.5(iii), every cyclic shift of C again yields a Kalmanson matrix. Hence, choosing $\phi = \langle r, \dots, s, \dots, n, 1, \dots, r - 1 \rangle$ or choosing $\phi = \langle r, \dots, n, 1, \dots, s, s + 1, \dots, r - 1 \rangle$ completes the argument. $\square$

4. Recognition of permuted Kalmanson matrices. This section shows how to recognize permuted Kalmanson matrices in polynomial time. The recognition algorithm is described in two steps: first we give a rough outline of the algorithm in subsection 4.1. We sketch a divide and conquer approach that is based on the lemmas derived in the preceding section. Then in subsection 4.2, we describe a fast implementation of the algorithm that runs in $O(n^2 \log n)$ time.

4.1. Outline of the algorithm. Given an $n \times n$ matrix C , we want to decide whether there exists a permutation σ such that $C_\sigma \in \mathbf{K}$ and we want to compute σ in case it exists. Our solution algorithm follows a divide and conquer strategy. The main goal is to find in polynomial time $D(n)$ a so-called *nice bipartition* of the set I of rows, i.e., a bipartition into two sets V and W that satisfies the following three properties.

(N1) $|V|, |W| \geq 2$.

(N2) $C[V, W]$ is a sum-matrix.

(N3) The matrix C is a permuted Kalmanson matrix if and only if there exists a permutation $\sigma \in \text{STR}(V, W)$ with $C_\sigma \in \mathbf{K}$.

If we have found some nice bipartition, we choose rows $q \in W$ and $p \in V$ and recursively compute Kalmanson permutations ψ and π for the two matrices $C[V \cup \{q\}, V \cup \{q\}]$ and $C[\{p\} \cup W, \{p\} \cup W]$. According to Lemma 3.2 and property (N3) above, either the concatenation of ψ and π is a Kalmanson permutation for C , or C cannot be a permuted Kalmanson matrix. By Proposition 2.4, it can be decided in $O(n^2)$ time whether the concatenation of ψ and π indeed yields a Kalmanson permutation. Summarizing, this results in a recursive algorithm with time complexity

$$(4.1) \quad T(n) \leq \max_{2 \leq k \leq n-2} \{T(k+1) + T(n-k+1)\} + D(n) + O(n^2).$$

1. Find a row k with $k \not\sim 1$.
If k does not exist: $\implies C$ itself is Kalmanson matrix. Stop.
2. From k , define an initial partition of I with two stripes K_1 and K_2 .
3. If all stripes in the current partition of I have cardinality one:
 $\implies$ Only one potential Kalmanson permutation left. Stop.
4. Rotate the current partition such that the first stripe has cardinality at least two.
5. If the first stripe in the current partition of I together with its complement forms a nice bipartition: $\implies$ Nice bipartition found. Stop.
6. Refine the partition by applying Proposition 2.2 to the submatrix whose row set is the first stripe and whose column set is the complement of the first stripe: $\implies$ Refinement of partition found. Goto 3.

FIG. 4.1. A high-level description of how to find a nice bipartition.

It is easy to verify that $T(n) = O(nD(n) + n^3)$ and hence, the algorithm runs in polynomial time. It remains to explain how to find a nice bipartition (a high-level pseudocode description of this procedure is given in Figure 4.1).

First, we find a row k that is *not* equivalent to row 1 (if such a row k does not exist, it follows from Lemma 3.5 that the identity permutation is a Kalmanson permutation). Compute $\mathcal{M}(1, k)$ and define the sets $K_1 = \mathcal{M}(1, k) \cup \{1\}$ and $K_2 = I \setminus K_1$. Since $1 \not\sim k$, $|K_1|, |K_2| \geq 2$ holds. By Lemma 3.6, it is sufficient to deal with permutations ϕ for which $K_1 \prec K_2$ holds, i.e., with permutations $\phi \in \text{STR}(K_1, K_2)$. Now, if $C[K_1, K_2]$ is a sum-matrix, K_1 and K_2 form a nice bipartition and we are done. Otherwise, by Lemma 3.1, it is necessary to deal with permutations ϕ for which the matrix $C_\phi[K_1, K_2]$ is a Contra Monge matrix. According to Proposition 2.2, these permutations can be described by $\phi \in \text{STR}_1 \cup \text{STR}_1^*$ where $\text{STR}_1 = \text{STR}(K_{11}, \dots, K_{1r}, K_{21}, \dots, K_{2s})$ and $\text{STR}_1^* = \text{STR}(K_{1r}, \dots, K_{11}, K_{2s}, \dots, K_{21})$ for appropriate partitions $K_{11}, \dots, K_{1r}$ of K_1 and $K_{21}, \dots, K_{2s}$ of K_2 . It is easy to see that by rotation and reversion, every $\phi^* \in \text{STR}_1^*$ can be transformed into some $\phi \in \text{STR}_1$. By Proposition 2.5 we conclude that in case C can be permuted into a Kalmanson matrix; this can also be reached by some $\phi \in \text{STR}_1$ and thus it is sufficient to consider permutations in STR_1 .

In case all stripes in STR_1 have cardinality one, there remains just a single potential Kalmanson permutation (and it can be checked in $O(n^2)$ time whether the permutation indeed is a Kalmanson permutation). Otherwise, there is some stripe K_{ij} of cardinality at least two. We rotate the sequence of stripes in STR_1 in such a way that K_{ij} becomes the first stripe and rename the stripes into $L_1, \dots, L_\ell$ with $L_1 = K_{ij}$. In case $C[L_1, I \setminus L_1]$ is a sum-matrix, $V = L_1$ and $W = I \setminus L_1$ form a nice bipartition. Otherwise, we observe that any Kalmanson permutation in $\text{STR}(L_1, \dots, L_\ell)$ must transform $C[L_1, I \setminus L_1]$ into a Contra Monge matrix. According to Proposition 2.2, we compute an appropriate partition of L_1 and an appropriate partition of $I \setminus L_1$ that encodes all permutations that transform $C[L_1, I \setminus L_1]$ into a Contra Monge matrix. This either results in a refinement of the stripes in $L_1, \dots, L_\ell$ (cf. Proposition 2.1) or the set of potential permutations becomes empty (and C is not a permuted Kalmanson matrix).

This procedure is repeated over and over again as long as there are stripes of cardinality at least two. Either we find a nice bipartition of I , or we may refine the stripes, or eventually all stripes are of cardinality one. Since the stripes can be refined

at most $(n - 1)$ times, a conservative estimation yields $D(n) = O(n^3)$. According to the above arguments, the recognition algorithm runs in polynomial time $O(n^4)$.

4.2. Implementation of the algorithm. In this subsection, we explain how to implement the divide and conquer algorithm described in the preceding section in $O(n^2 \log n)$ time. Our main tools are advanced data structures (PQ-trees and union-find structures) and a slight modification of the divide step. Let us start with three simple but important statements.

- All through the algorithm we will derive and exploit sufficient conditions on the matrix for being Kalmanson. These conditions will restrict and cut down the set of potentially feasible permutations. We will not verify at every single step whether we are indeed dealing with Kalmanson permutations (this would be too time consuming). Hence, the output of the algorithm will be some $\sigma \in S_n$ with the following property: “In case C is a permuted Kalmanson matrix, then $C_\sigma \in \mathbf{K}$.” (Cf. Lemma 4.1). The verification of whether C_σ is indeed in $\mathbf{K}$ is postponed to a single $O(n^2)$ check in the end, *after* the algorithm.

- All sets of permutations induced by stripes are stored in PQ-trees of height two (as already stated in section 2). For a set of permutations Π over I stored in a PQ-tree of constant height and a subset $J \subseteq I$, the following operation can be performed in $O(|J|)$ time: “Restructure the PQ-tree in such a way that the restructured PQ-tree stores exactly those permutations $\pi \in \Pi$ in which the objects in J are consecutive” (cf. Booth and Lueker [1]). Note that this operation might lead to an empty set of permutations.

- The algorithm represents its knowledge on the equivalence of rows in a union-find data structure in order to answer in constant time questions of the form “Are the rows i and j already known to be equivalent?”. In case it receives new information on the equivalence of two rows i and j , the corresponding two equivalence classes have to be combined into a single class. We choose an implementation of this data structure that supports the FIND operation in constant time and that supports the UNION operation in time that is linear in the size of the merged classes (this can be done, for example, via pointers from every element to the name of the corresponding class; see Cormen, Leiserson, and Rivest [3]).

Next, we give a precise low-level description of the algorithm. The algorithm performs the following five steps (S0)–(S4).

(S0) If $n \leq 3$, output any permutation $\sigma \in S_n$.

(S1) Row 1 is compared to rows k , $2 \leq k \leq n$ until some row k is found that is not equivalent to 1 or all rows are known to be equivalent to row 1. If $1 \not\sim k$, the algorithm moves on to (S2). If all rows are found to be equivalent to row 1, the algorithm outputs the identity permutation and stops.

Observing that any $n \times n$ matrix with $n \leq 3$ is a Kalmanson matrix justifies step (S0). In step (S1), a comparison between rows 1 and k is performed as follows: first, the algorithm checks whether row 1 is already known to be equivalent to k (from some higher level of the recursion). In case it is, the algorithm immediately moves on to the next row. Otherwise, it scans row k in linear time. If it turns out that $1 \sim k$, this information is handed over to the union-find structure and the next row is investigated.

The following step (S2) is a kind of initialization for step (S3).

(S2) Let $k \not\sim 1$ be the result of (S1). Compute $\mathcal{M}(1, k)$ and define the sets $K_1 = \mathcal{M}(1, k) \cup \{1\}$ and $K_2 = I \setminus K_1$. Compute for $C[K_1, K_2]$ the partitions $K_{11}, \dots, K_{1r}$ of K_1 and $K_{21}, \dots, K_{2s}$ of K_2 that encode all permutations that

FIG. 4.2. Illustration for step (S3) of the algorithm: the cross-hatched region is matrix D ; the last column of D is q .

transform $C[K_1, K_2]$ into a Contra Monge matrix (this is done according to Proposition 2.2).

Set $\text{STR}_0 = \text{STR}(K_1, K_2)$ and $\text{STR}_1 = \text{STR}(K_{11}, \dots, K_{1r}, K_{21}, \dots, K_{2s})$.

By the discussion in subsection 4.1, it is sufficient to search for Kalmanson permutations in STR_1 . Note that from the submatrix $C[K_1, K_2]$, we will not receive any further information on refining the stripes: by Proposition 2.2(iii) for every stripe K_{1i} , the matrix $C[K_{1i}, K_2]$ is a sum-matrix and for every stripe K_{2j} , the matrix $C[K_1, K_{2j}]$ is a sum-matrix. However, we may receive information from $C[K_{1i}, K_1]$ and $C[K_{2j}, K_2]$.

In the following “refinement step” (S3), a sequence of sets of permutations $\text{STR}_1, \text{STR}_2, \dots$ is constructed with $\text{STR}_i = \text{STR}(Q_1^{(i)}, \dots, Q_{m_i}^{(i)})$. The partition in STR_{i+1} is always a refinement of the partition in STR_i . The refinement procedure stops as soon as, for every stripe $Q = Q_j^{(i)}$, the submatrix $C[Q, I \setminus Q]$ is a sum-matrix.

(S3) Starting with $j = 1$, perform the following refinement procedure for every stripe $Q = Q_j^{(i)}$ in STR_i : rotate STR_i in such a way that Q becomes the first stripe in STR_i . Let Q^* be the *mother stripe* of Q in STR_{i-1} , i.e., the stripe that contains set Q . Let q be any column that is *not* in Q^* . In case $Q \neq Q^*$ holds, consider the matrix $D = C[Q, (Q^* \setminus Q) \cup \{q\}]$ and compute according to Proposition 2.2 the partition of the rows and columns of D that implicitly describe all permutations that transform D into a Contra Monge matrix. If $Q = Q^*$ holds, consider the next set $Q_j^{(i)}$.

The set STR_{i+1} results from refining STR_i according to all these new partitions. Step (S3) is repeated until $\text{STR}_{i+1} = \text{STR}_i$, i.e., the partition has not been refined.

Consider the matrices $E = [Q, I \setminus Q]$ and $F = [Q, I \setminus Q^*]$ (cf. Figure 4.2). Since Q was obtained as a stripe from Q^* , Proposition 2.2(iii) ensures that the matrix F is a sum-matrix. By Lemma 3.1, we may restrict our attention to permutations that transform matrix E into a Contra Monge matrix. E essentially decomposes into D and F (where D and F have the common column q). Proposition 2.2 states that in Contra Monge matrices, sum-submatrices may as well be represented by a single column (in our case by column q). This simplifies matrix E down to matrix D .

If no more refinement of a stripe Q is possible, it follows from Proposition 2.2 that

the corresponding matrix D is a sum-matrix. Since F is a sum-matrix, too, and the sum-matrices D and F have a common column q , this implies that the matrix E itself is a sum-matrix. Hence at the end of (S3), all matrices $C[Q, I \setminus Q]$ are sum-matrices.

Note that some of the derived refinements may contradict each other: we receive constraints (i.e., subsets of I that must be consecutive) from every single stripe Q . The constraints arise from the rows and from the columns of the Contra Monge matrices and they also concern the other stripes within Q^* . Another type of contradiction arises if the Contra Monge conditions force q to become an interior column of D . Hence, if some constraint cannot be fed into the PQ-tree, there does not exist a consistent refinement for Q^* and matrix C cannot be a permuted Kalmanson matrix. In this case, the algorithm returns any permutation and stops.

(S4) Recursion. Let $\text{STR}_i = \text{STR}(Q_1^{(i)}, \dots, Q_m^{(i)})$ be the resulting set of permutations as derived in step (S3). For every stripe $Q_j^{(i)}$, select any row q_{j+1} from $Q_{j+1}^{(i)}$. Set $X_j = Q_j^{(i)} \cup \{q_{j+1}\}$ and determine the restriction of the union-find structure to the elements in X_j . Compute recursively a Kalmanson permutation π_j for $C[X_j, X_j]$ under this union-find structure. Afterwards, rotate π_j such that q_{j+1} becomes the last element and remove q_{j+1} from the rotated permutation. This yields permutation σ_j .

The output consists of the concatenation of the permutations $\sigma_1, \sigma_2, \dots, \sigma_m$ in exactly this order.

Step (S4) essentially applies Lemma 3.3 to the stripes in STR_i .

LEMMA 4.1. *Either matrix C is not a permuted Kalmanson matrix, or $C_\sigma \in \mathbb{K}$ holds for the output permutation $\sigma \in S_n$ of the above algorithm.*

The correctness of this statement follows from the lemmas derived in section 3. It remains to investigate the time complexity of the algorithm. In doing this, it is convenient to make a separate analysis for the main part of the algorithm and a separate analysis for the UNION operations.

We first investigate the main part of the algorithm. Let $\text{STR}_i = \text{STR}(Q_1^{(i)}, \dots, Q_m^{(i)})$ be the final set of permutations computed in step (S3) and let a_j denote the cardinality of $Q_j^{(i)}$. Clearly, $a_j \leq n - 2$ for all j and $\sum_{j=1}^m a_j = n$. Define $S_a = \sum_{j=1}^m a_j^2$. For every j , the algorithm treats the submatrix corresponding to stripe $Q_j^{(i)}$ recursively. The remaining area of size $n^2 - S_a$ is covered by small, almost disjoint submatrices (they are disjoint with the exception of the negligible columns used to represent the sum-submatrices). Such a covering submatrix with dimensions $x \times y$ is handled in $O(xy + x \log x + y \log y)$ time according to Proposition 2.2. Hence, handling a matrix of area A is done in $O(A \log A)$ time and this complexity is superlinear in the concerned area. Thus, the total cost for handling all these matrices is at most $O((n^2 - S_a) \log(n^2 - S_a))$. Storing, refining, and modifying the partitions with the help of PQ-trees costs time that is linear in the size of the concerned set (i.e., proportional to the sidelengths of the submatrices) and thus is dominated by the cost for handling the submatrices. The overall cost for the FIND operations in step (S1) is $O(n)$. Summarizing, the time $T(n)$ for treating a matrix of sidelength n obeys

$$(4.2) \quad T(n) \leq \max_{a_j} \left\{ \sum_{j=1}^m T(a_j + 1) + c_1 \left(n^2 - \sum_{j=1}^m a_j^2 \right) \log_2 \left(n^2 - \sum_{j=1}^m a_j^2 \right) + c_2 n \right\},$$

where the maximum is taken over all m -tuples of integers a_j with $1 \leq a_j \leq n - 2$ and $\sum_{j=1}^m a_j = n$, and where c_1 and c_2 are appropriate positive constants.

LEMMA 4.2. $T(n) = O(n^2 \log n)$.

Proof. Define $c_3 = \max\{20c_1, 20c_2, T(2), T(3)\}$. We prove by induction that for all $n \geq 2$, $T(n) \leq c_3(n-1)^2 \log_2 n$ holds. By the definition of c_3 , this inequality holds true for $n = 2$ and $n = 3$. Next consider some fixed $n \geq 4$ and consider an m -tuple of integers a_j that maximizes the expression in the right-hand side of (4.2). Observe that $S_a = \sum_{j=1}^m a_j^2 \leq n^2 - 4n + 8$ holds, and use the inductive assumption to derive that $\sum_{j=1}^m T(a_j + 1) \leq c_3 S_a \log_2 n$. Hence, $T(n) \leq c_3 S_a \log_2 n + c_1(n^2 - S_a) \log_2(n^2) + c_2 n$, and the lemma follows. $\square$

LEMMA 4.3. *The total time needed for performing all UNION operations is bounded by $O(n^2)$.*

Proof. The union-find structure is only used in step (S1). Since the cost for the FIND operations was already investigated above, it remains to analyze the UNION operations. We represent the recursive process in the standard way by a tree: the root of the tree represents the original problem. The sons of a vertex v in the tree represent the subproblems originating in step (S4) when the problem corresponding to v is treated. Every vertex v is labeled by two numbers a_v and b_v , where a_v equals the number of rows (i.e., the *size*) of the corresponding subproblem and b_v denotes the number of UNION operations that result from treating the subproblem. Clearly, the a -label of the root equals n , and $b_v \leq a_v$ holds for every vertex v .

Since every UNION operation decreases the number of equivalence classes by one and since in every leaf of the tree there remains at least one equivalence class, on every branch going from some leaf up to the root, the overall sum of all values b_v is at most $n - 1$. Moreover, the sum of the a -labels of all sons of vertex v is bounded by a_v plus the number of sons of v . With this, the total sum of the a -labels of all leaves is $O(n)$. The overall cost of all UNION operations is $O(\sum a_v b_v)$ where the sum is taken over all vertices in the tree. This sum may be bounded from above by another sum that runs over all leaves and adds up the a -label of the leaves times the overall sum of all b -labels on the corresponding path leading from the leaf up to the root. By the above inequalities, this second sum is dominated by $O(n^2)$. This completes the proof of the lemma. $\square$

THEOREM 4.4. *For a symmetric $n \times n$ matrix C , it can be decided in $O(n^2 \log n)$ time whether C is a permuted Kalmanson matrix.*

Proof. From Lemma 4.1, we get the correctness and from Lemmata 4.2 and 4.3, we get the time complexity of the algorithm. In the end, we permute C according to the output permutation σ and check whether the permuted matrix C_σ indeed is Kalmanson. This is done in $O(n^2)$ time as described in Proposition 2.4. $\square$

5. Master tours in polynomial time. A *master tour* π for a set V of cities fulfills the following property: for every $V' \subseteq V$, an optimum travelling salesman tour for V' is obtained from π by removing from it the cities that are not in V' . Given the distance matrix C for a set of cities, the *master tour problem* consists in deciding whether this set of cities possesses a master tour. In this section, we prove that the master tour problem is closely related to permuted Kalmanson matrices and hence solvable in polynomial time.

THEOREM 5.1. *For an $n \times n$ symmetric distance matrix C , the permutation $\langle 1, 2, \dots, n \rangle$ is a master tour if and only if C is a Kalmanson matrix.*

Proof. (Only if): Assume that $\langle 1, 2, \dots, n \rangle$ is a master tour for the distance matrix C . Then by definition, for each subset of four cities $\{i, j, k, \ell\}$ with $1 \leq i < j < k < \ell \leq n$, the tour $\langle i, j, k, \ell \rangle$ is an optimal TSP tour. Since C is symmetric, there are only three combinatorially different tours through those cities: (i) $\langle i, j, k, \ell \rangle$, (ii) $\langle i, j, \ell, k \rangle$

and (iii) $\langle i, k, j, \ell \rangle$. The optimality of tour (i) implies that $c_{ij} + c_{jk} + c_{k\ell} + c_{\ell i} \leq c_{ij} + c_{j\ell} + c_{\ell k} + c_{ki}$ and $c_{ij} + c_{jk} + c_{k\ell} + c_{\ell i} \leq c_{ik} + c_{kj} + c_{j\ell} + c_{\ell i}$. By exploiting the symmetry of C and simplifying, the above inequalities turn into

$$(5.1) \quad c_{jk} + c_{i\ell} \leq c_{ik} + c_{j\ell} \quad \text{and} \quad c_{ij} + c_{k\ell} \leq c_{ik} + c_{j\ell},$$

which are exactly the conditions (1.2) and (1.1). Hence, C is a Kalmanson matrix.

(If): Let $K = \{x_1, \dots, x_k\}$ be a subsequence of $\langle 1, 2, \dots, n \rangle$. Then by Proposition 2.5(ii), the matrix $C[K, K]$ is again a Kalmanson matrix and by Kalmanson's result [6] the tour $\langle x_1, \dots, x_k \rangle$ is an optimal tour for K . Consequently, $\langle 1, 2, \dots, n \rangle$ is a master tour. $\square$

THEOREM 5.2. *For a symmetric $n \times n$ matrix C , it can be decided in $O(n^2 \log n)$ time whether C possesses a master tour.*

Proof. By Theorem 5.1, a symmetric distance matrix has a master tour if and only if it is a permuted Kalmanson matrix. By Theorem 4.4, permuted Kalmanson matrices can be recognized in $O(n^2 \log n)$ time. $\square$

6. Discussion. In this paper we have developed an algorithm for recognizing permuted $n \times n$ Kalmanson matrices in $O(n^2 \log n)$ time and showed that this problem is equivalent to detecting master tours. Since the input is of size n^2 , the derived time complexity is close to optimal. Two questions remain open.

(1) We would like to know whether the $\log n$ factor in the time complexity can be shaved off in the *random access machine* model of computation.

(2) The second question concerns characterizing all Kalmanson permutations for some given input matrix C . Our algorithm just outputs a single Kalmanson permutation. However, we would like to have a complete and concise description of *all* Kalmanson permutation similar to the concise description of all Contra Monge permutations in Proposition 2.2. One of the main obstacles in deriving such a description is that we do not fully understand the structure of equivalent columns. For example, it is *not true* that equivalent columns must stick together in Kalmanson permutations. Consider the following two matrices.

$$A = \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 1 \\ 0 & 0 & * & 0 \\ 0 & 1 & 0 & * \end{pmatrix} \quad A_\sigma = \begin{pmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 1 \\ 0 & 0 & 1 & * \end{pmatrix}$$

Matrix A is a Kalmanson matrix where rows 1 and 3 are equivalent. However, its permutation A_σ is not a Kalmanson matrix and it is easy to check that *no* permutation of A which makes rows 1 and 3 neighboring rows yields a Kalmanson matrix.

Note added in proof. In a recent paper presented at the 1996 European Symposium on Algorithms, Christopher, Farach, and Trick answered both of the above questions in the positive. They derived an $O(n^2)$ recognition algorithm and a simple characterization of the set of all Kalmanson permutations that is based on PQ-trees.

Acknowledgment. We would like to thank Bettina Klinz for a careful reading of the paper and for many helpful comments.

REFERENCES

- [1] K. S. BOOTH AND G. S. LUEKER, *Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms*, J. Comput. System Sci., 13 (1976), pp. 335–379.

- [2] R. E. BURKARD, B. KLINZ, AND R. RUDOLF, *Perspectives of Monge properties in optimization*, Discrete Appl. Math., 70 (1996), pp. 95–161.
- [3] T. H. CORMEN, C. E. LEISERSON, AND R. L. RIVEST, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.
- [4] V. G. DEĬNEKO AND V. L. FILONENKO, *On the Reconstruction of Specially Structured Matrices*, Aktualnyje Problemy EVM i programirovaniye, Dnepropetrovsk, DGU, 1979 (in Russian).
- [5] P. C. GILMORE, E. L. LAWLER, AND D. B. SHMOYS, *Well-solved special cases*, in The Travelling Salesman Problem, John Wiley, Chichester, 1985, Chapter 4, pp. 87–143.
- [6] K. KALMANSON, *Edgeconvex circuits and the travelling salesman problem*, Canad. J. Math., 27 (1975), pp. 1000–1010.
- [7] E. L. LAWLER, J. K. LENSTRA, A. H. G. RINNOOY KAN, AND D. B. SHMOYS, *The Travelling Salesman Problem*, John Wiley, Chichester, 1985.
- [8] C. H. PAPADIMITRIOU, *Lecture on computational complexity at the Maastricht Summer School on Combinatorial Optimization*, Maastricht, the Netherlands, August, 1993.
- [9] C. H. PAPADIMITRIOU, *Computational Complexity*, Addison–Wesley, Reading, MA, 1994.