Well-solvable special cases of the TSP : a survey

Eindhoven University
of Technology

Department of Mathematics and Computing Science

Memorandum COSOR 96-02

Well-Solvable Special Cases of the TSP: A Survey

R.E. Burkard
V.G. Deĭneko
R. van Dal
J.A.A. van der Veen
G.J. Woeginger

Eindhoven, January 1996
The Netherlands

Well-Solvable Special Cases of the TSP: A Survey *

RAINER E. BURKARD † VLADIMIR G. DEĬNEKO †‡ RENÉ VAN DAL §
JACK A.A. VAN DER VEEN ¶ GERHARD J. WOEGINGER ‖

December 1995

Abstract

The Traveling Salesman Problem belongs to the most important and most investigated problems in combinatorial optimization. Although it is an $\mathcal{NP}$-hard problem, many of its special cases can be solved efficiently. We survey these special cases with emphasis on results obtained during the decade 1985–1995. This survey complements an earlier survey from 1985 compiled by Gilmore, Lawler and Shmoys.

Keywords: Traveling Salesman Problem, Combinatorial optimization, Polynomial time algorithm, Computational complexity.

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

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

‡on leave from Department of Applied Mathematics, Dnepropetrovsk University, Gagarin Av. 72, 320625 Dnepropetrovsk, Ukraine.

§Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR) der Universität Heidelberg, Im Neuenheimer Feld 368, D-69120 Heidelberg, Germany.

¶Nijenrode University, The Netherlands School of Business, Straatweg 25, 3621 BG Breukelen, The Netherlands.

‖Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands. Supported by a research fellowship of the Euler Institute for Discrete Mathematics and its Applications.

Contents

1 Introduction

If a salesman, starting from his home city, has to visit exactly once each city on a given list and in the end has to return again to his home city, it is plausible for him to select the order in which he visits the cities so that the total of the distances traveled in his tour is as small as possible. Clearly, this will save him time and gas. As the salesman tries to find this shortest tour, he faces the so-called *Traveling Salesman Problem* (TSP). In a more mathematical formulation, the TSP can be stated as follows. For a given $n \times n$ distance matrix $C = (c_{ij})$, find a cyclic permutation π of the set $\{1, 2, \ldots, n\}$ that minimizes the function

$$c(\pi) = \sum_{i=1}^{n} c_{i\pi(i)}. \tag{1}$$

The value $c(\pi)$ is usually referred to as the *length* (or *cost* or *weight*) of the permutation π. The TSP is one of the standard problems in combinatorial optimization and has many important applications like routing or production scheduling with job-dependent set-up times. Several applications leading to special cases will be treated in this paper, see in particular Sections 5 and 6. The TSP is also a notoriously hard combinatorial optimization problem, since it is $\mathcal{NP}$-hard. Except implicit enumeration in one way or the other, no other solution method is known for finding an optimal solution. Therefore, there is a special need for good suboptimal solution methods and it is important to investigate special cases which can be solved by polynomial algorithms.

More than ten years ago, Gilmore, Lawler and Shmoys [57] wrote an excellent survey on efficiently solvable special cases of the TSP. The goals of our new survey are threefold. Firstly, the paper [57] strongly stimulated further research on efficiently solvable cases of the TSP, and our first goal was to collect and to summarize the new results obtained since then. Secondly, the paper [57] was written before the times of 'glasnost' and 'perestrojka' and hence without full knowledge of the Soviet literature on this subject. Our new survey provides many references and pointers to Soviet results from the 1960's and the 1970's that were not known in the West before. Our third goal was to present unified approaches to several classes of problems in this area. When ten years ago there were two or three seemingly unrelated results on some special case, it sometimes turned out in the meantime that these 'unrelated' results can be derived from some general theory or general theorem. Summarizing, our new survey does not try to replace, but tries to complement the earlier survey [57].

This survey is organized as follows. Section 2 deals with so-called pyramidal tours. It describes a unified proof technique, the tour improvement technique, for deriving results in this area. As a highlight, a simple and concise proof of Demidenko's celebrated theorem is presented. Section 3 discusses several classes of Euclidean instances of the TSP. It can be shown that special geometric properties of a Euclidean point set lead to polynomial time algorithms for the TSP. Section 4 and Section 5 are closely connected. Section 4 deals with the *theory* of subtour patching on permuted Monge matrices. For the central theorem in this area a simple, geometrically flavored proof is given. Section 5 then *applies* this theory to several special cases. Section 6 collects results on the linear programming relaxation of the TSP, on the graphical TSP, and on special cases of the TSP on distance matrices with concise descriptions.

For further material on the TSP, the reader is referred to the book by Lawler, Lenstra, Rinnooy Kan and Shmoys [70] "The Traveling Salesman Problem – A Guided Tour of Combinatorial Optimization".

We conclude this section with several definitions, preliminaries, and elementary results.

For the TSP, the set of cities will always be denoted by $\{1,\ldots,n\}$, i.e. n will always denote the number of cities.

A *permutation* ϕ is a one-to-one mapping of $I = \{1,2,\ldots,n\}$ onto itself. It can be written as

$$\phi = \begin{pmatrix} 1 & 2 & \cdots & n \\ \phi(1) & \phi(2) & \cdots & \phi(n) \end{pmatrix} = (\phi(1),\phi(2),\ldots,\phi(n))$$

or in compact form in terms of factors. Let $i_1, i_2, \ldots, i_r$ be different elements of $\{1,2,\ldots,n\}$. If $\phi(i_k) = i_{k+1}$ for $k = 1,2,\ldots,r-1$, and $\phi(i_r) = i_1$, then $\langle i_1, i_2, \ldots, i_r\rangle$ is called a *factor* (or *cycle*, or *subtour*) of the permutation ϕ. A factor with $r \geq 2$ will be called a *non-trivial factor*. For example, the permutation

$$\phi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 2 & 5 & 6 & 1 & 4 & 7 \end{pmatrix} = (3,2,5,6,1,4,7) = \langle 1,3,5\rangle\ \langle 2\rangle\ \langle 4,6\rangle\ \langle 7\rangle$$

has two nontrivial factors. In writing down a permutation, usually its trivial factors are omitted. So, a permutation ϕ can be written also as $\phi = \langle 1,3,5\rangle\ \langle 4,6\rangle$. A cyclic permutation or *tour* is a permutation consisting of only one factor. The permutation ε defined by $\varepsilon(i) = i$ for $i = 1,2,\ldots,n$ is called the *identity permutation*. A permutation with a single factor of the form $\langle j,k\rangle$, which just interchanges j and k, is called a *transposition*. A transposition $\langle j,k\rangle$ is called an *adjacent transposition* if $k = j+1$.

If both ϕ and ρ are permutations on $\{1,2,\ldots,n\}$, then their composition (which can be thought of as a multiplication) is denoted by $\phi \circ \rho$ and defined by

$$\phi \circ \rho(i) = \phi(\rho(i)) \text{ for } i = 1,2,\ldots,n.$$

Note that $\phi \circ \rho$ is again a permutation.

The problem of minimizing the function $c(\pi)$ in (1) on the set of all $n!$ permutations is known as the *linear assignment problem*. An optimal solution of the linear assignment problem is called an *optimal assignment* and can be computed in polynomial time (see e.g. Lawler [69]). Clearly, in case an optimal assignment ϕ is a cyclic permutation then ϕ is also an optimal solution for the TSP. This means that the length of an optimal assignment gives a lower bound on the length of an optimal tour. In other words, the linear assignment problem is a relaxation of the TSP.

We close with mentioning two basic properties of the TSP. The first one is that the set of optimal tours for the TSP with distance matrix (c_{ij}) is the same as that for the TSP with distance matrix $(c_{ij}+a_i+b_j)$, where (a_i) and (b_i) are two given vectors. The second basic property is that the length of an optimal tour for the TSP with a distance matrix (c_{ij}) is the same as that for the TSP with the distance matrix $(c_{\sigma(i)\sigma(j)})$, i.e. permuting rows and columns in a distance matrix according to the same permutation σ does not change the length of an optimal tour. An optimal tour for the TSP with $(c_{\sigma(i)\sigma(j)})$ can easily be obtained from an optimal tour for the TSP with (c_{ij}) by renumbering the cities according to σ.

2 Pyramidally solvable TSP cases

A tour $\phi = \langle 1, i_1, i_2, \ldots, i_r, n, j_1, j_2, \ldots, j_{n-r-2}\rangle$ is called a *pyramidal* tour if $i_1 < i_2 < \cdots < i_r$ and $j_1 > j_2 > \cdots > j_{n-r-2}$. In other words, the salesman starts in City 1, then visits some cities in increasing order, reaches City n and returns to City 1 visiting the remaining cities in decreasing order. Pyramidal tours have two nice properties. First, a minimum cost pyramidal tour can be determined in $O(n^2)$ time — although the number of pyramidal tours on n cities is exponential in

n. Hence, the pyramidal tours constitute an exponential size subset of the cyclic permutations over which we can optimize in polynomial time. Secondly, there exist certain combinatorial structures of distance matrices that guarantee the existence of a shortest tour that is pyramidal. Hence, in case the distance matrix possesses these combinatorial structures, the TSP reduces to the problem of finding a shortest pyramidal tour and thus is solvable in polynomial time.

The TSP restricted to a class of matrices $\mathcal{M}$ is called *pyramidally solvable* if for every matrix in $\mathcal{M}$ there is an optimal tour that is pyramidal. In the 1970's and 1980's, quite a few pyramidally solvable TSP classes were detected and investigated. In this section we will review the literature on this subject and provide some 'simple' proofs. All these proofs are derived by a unified proof-technique, called the *tour improvement technique*. We review the following special matrix classes whose combinatorial structure guarantees the existence of a pyramidal optimal tour: Monge matrices, Supnick matrices, Demidenko matrices, Kalmanson matrices, Van der Veen matrices and Generalized Distribution matrices. Moreover, in Section 2.3 and in Section 2.4 we address the question of recognizing these combinatorial structures for given distance matrices. Finally, Section 2.5 deals with pyramidality in the bottleneck version of the TSP.

2.1 Optimization on pyramidal tours

We start with the fundamental result that makes considering pyramidally solvable TSPs interesting. The result is easily proved via a dynamic programming formulation.

Theorem 2.1 (Klyaus [64], Gilmore, Lawler and Shmoys [57])
For any $n \times n$ distance matrix $C = (c_{ij})$, the problem of finding the shortest pyramidal tour with respect to C, i.e. the problem

$$\min \left\{ \sum_{i=1}^{n} c_{i\rho(i)} : \rho \text{ is a pyramidal tour} \right\}$$

is solvable in $O(n^2)$ time. □

Apparently the first paper where pyramidal tours were used for special TSP cases was a paper by Aizenshtat and Kravchuk [5] in 1968. They investigated the TSP on so-called *ordered product matrices* $C = (c_{ij}) = (u_i \cdot v_j)$ where $0 \leq u_1 \leq u_2 \leq \cdots \leq u_n$ and $v_1 \geq v_2 \geq \cdots \geq v_n \geq 0$ hold. Ordered product matrices form a subclass of the *Monge matrices*. An $n \times n$ matrix C is called a *Monge matrix* if

$$c_{ij} + c_{rs} \leq c_{is} + c_{rj} \qquad \text{for all } 1 \leq i < r \leq n,\ 1 \leq j < s \leq n. \tag{2}$$

Gilmore, Lawler and Shmoys [57] call matrices with this property *distribution matrices*, referring to a way to construct such matrices from nonnegative matrices, called density matrices. The term "Monge matrix" was coined by Hoffman [61] who generalized an observation made by Gaspard Monge [75] in 1781 (see also Section 3.1). For more information on Monge matrices, the reader is referred to the survey by Burkard, Klinz and Rudolf [17]. The result by Aizenshtat and Kravchuk [5] is essentially contained in the following theorem.

Theorem 2.2 (Gilmore, Lawler and Shmoys [57], p.101)
If C is a Monge matrix, then there is an optimal tour that is pyramidal. □

The combinatorial structures of Monge matrices allow to improve on the time complexity in Theorem 2.1. Park [87] showed that the TSP restricted to Monge matrices is solvable in $O(n)$ time. This is done by speeding up the dynamic program of Klyaus with the help of matrix search techniques developed by Aggarwal, Klawe, Moran, Shor and Wilber [2] and by Eppstein [45].

2.2 The Tour-Improvement technique for pyramidally solvable TSPs

In this section a unified proof-technique for pyramidally solvable TSPs is presented. This technique was essentially introduced by Van der Veen [117] and successfully applied in e.g. Burkard and Van der Veen [19], Van der Veen, Sierksma and Van Dal [119], Warren [125] and Van der Veen [118].

The idea is as follows. In order to prove that under certain conditions on the distance matrix there exists an optimal tour that is pyramidal, a so-called *tour-improvement technique* (*TI*-technique) is used. Starting from an arbitrary tour τ, a sequence of tours $\tau_1, \tau_2, \ldots, \tau_T$ is constructed, with $\tau_1 = \tau$, such that

$$c(\tau_1) \geq c(\tau_2) \geq \cdots \geq c(\tau_T),$$

where $c(\tau_t)$ denotes the length of the tour τ_t $(t = 1, \ldots, T)$ and $T = T(\tau)$ is the smallest integer such that τ_T is a pyramidal tour. Note that if $T(\tau) < \infty$ for every tour τ, then there is always an optimal tour that is pyramidal. The tour τ_{t+1} will be obtained from τ_t by exchanging a number of arcs. This operation will be called a *transformation*. A transformation will be called *feasible* if the conditions on the distance matrix assure that the total length of the inserted arcs is no longer than the length of the removed arcs.

An index $i \in \{1, \ldots, n\}$ is a *peak* of a permutation τ if $i > \max\{\tau^{-1}(i), \tau(i)\}$ and a valley if $i < \min\{\tau^{-1}(i), \tau(i)\}$. By $P(\tau)$ we denote the set of peaks and by $V(\tau)$ the set of valleys of τ. Clearly, for any permutation τ, $|P(\tau)| = |V(\tau)|$ holds. The permutation τ is a pyramidal tour if and only if $P(\tau) = \{n\}$, or equivalently, if and only if $V(\tau) = \{1\}$. In order to prove that a *TI*-technique terminates after a finite number of steps (i.e. that $T(\tau) < \infty$ for all tours τ) we will use a kind of potential function: The *pyramidality number* $K(\tau)$ of a tour τ is defined by

$$K(\tau) := \left(\sum_{p \in P(\tau)} p \right) - \left(\sum_{v \in V(\tau)} v \right).$$

An alternate way to compute $K(\tau)$ is as follows. Let $k_\tau(i)$, $i = 1, \ldots, n-1$ denote the number of pairs $(u, \tau(u))$ such that

$$\min\{u, \tau(u)\} \leq i < i+1 \leq \max\{u, \tau(u)\}.$$

Note that $k_\tau(i)$ is even and at least two for all $i \in \{1, \ldots, n-1\}$. It follows that

$$K(\tau) = \frac{1}{2} \sum_{i=1}^{n-1} k_\tau(i).$$

For example, the tour $\tau = \langle 1, 4, 5, 2, 6, 3 \rangle$ has $V(\tau) = \{1, 2\}$, $P(\tau) = \{5, 6\}$ and $k_\tau(i) = 2, 4, 4, 4$ and 2 for $i = 1, 2, 3, 4$ and 5, respectively. Hence, $K(\tau) = (5+6) - (1+2) = 8 = (2+4+4+4+2)/2$.

Observe that for every tour τ

$$n - 1 \leq K(\tau) \leq \left\lfloor \frac{n}{2} \right\rfloor \left\lceil \frac{n}{2} \right\rceil$$

holds. The upper bound is sharp for the tours $\langle 1, n, 2, n-1, \ldots \rangle$. Furthermore, note that $K(\tau) = n - 1$ if and only if τ is a pyramidal tour. So, intuitively, the lower the value $K(\tau)$, the closer τ is to pyramidality. From the above remarks it follows that, if it can be shown that $K(\tau_t) > K(\tau_{t+1})$ for all t, then this is sufficient to prove the finiteness of the *TI*-technique.

In the following we will show how to apply the tour improvement technique to classes of symmetric (Section 2.2.1) and classes of asymmetric matrices (Section 2.2.2). In both cases our starting point will be the class of Monge matrices, i.e. it will be shown how the various pyramidally solvable cases relate to the class of Monge matrices. Section 2.2.3 applies the tour improvement technique to the maximization version of the TSP.

2.2.1 Symmetric matrices

We start with considering *Van der Veen matrices*. These are symmetric $n \times n$ matrices $C = (c_{ij})$ which fulfill the property

$$c_{ij} + c_{j+1,l} \leq c_{il} + c_{j,j+1} \qquad \text{for all } 1 \leq i < j < j+1 < l \leq n.$$

Theorem 2.3 (Van der Veen [118])
The TSP restricted to Van der Veen matrices is pyramidally solvable.

Proof sketch. Consider a pair of edges $([a,b];[x,y])$ with distinct $a,b,x,y \in \{1,\ldots,n\}$ such that $a = \min\{a,b,x,y\}$ and $x = \min\{x,y\}$. Such a pair of edges is called a *transformable edge pair* (TEP) of a tour τ if

(i) $[a,b]$ and $[x,y]$ are either both in $\{[i,\tau(i)] : \tau(i) > i\}$ or both in $\{[i,\tau(i)] : \tau(i) < i\}$, and

(ii) either $a < x < b < y$ and $b - x$ is even, or $a < x < y < b$ and $y - x$ is odd.

The TEPs of a tour play an important role in the TI-technique for the class of Van der Veen matrices.

TI-technique
Input: A tour τ.
Output: A pyramidal tour.
Step 0: Set $\tau_1 := \tau$ and $t := 1$.
Step 1: Find a TEP $([a,b];[x;y])$ of τ_t.
If τ_t does not contain a TEP, then STOP:
τ_t is a pyramidal tour.
Step 2: Transformation:
Obtain τ_{t+1} from τ_t by replacing $[a,b]$ and $[x,y]$
by $[a,x]$ and $[b,y]$. Return to Step 1 with $t := t+1$.

By the definition of a TEP, the transformation in Step 2 is feasible. Since $K(\tau_{t+1}) - K(\tau_t) = x - \min\{b,y\} < 0$, the TI-technique ends after a finite number of iterations. It remains to be shown that a tour contains a TEP if and only if it is not pyramidal. If a tour is pyramidal then there is no edge-pair $([a,b];[x,y])$ with $a < x < b$ such that it satisfies the first condition in the definition of a TEP, i.e. if a tour is pyramidal then it does not contain a TEP. Showing that each nonpyramidal tour does contain a TEP is slightly more complicated. The proof of this assertion is based on two observations.

The first observation is the following. Assume that all cities in a tour τ are either a peak or a valley. Let p be the smallest peak, i.e. the city in $P(\tau)$ with smallest index. It follows that $\tau^{-1}(p), \tau(p) \in V(\tau)$, $\tau^{-2}(p), \tau^2(p) \in P(\tau)$, $\max(\tau^{-1}(p), \tau(p)) < p$ and $\min(\tau^{-2}(p), \tau^2(p)) > p$. It can easily be seen that either $([\tau^{-1}(p), p];: [\tau(p), \tau^2(p)])$ or $([\tau^{-2}(p), \tau^{-1}(p)];: [p, \tau(p)])$ is a TEP. Hence, if each city of a tour is either a peak or a valley then the tour contains a TEP.

We now come to the second observation. Let τ be a given nonpyramidal tour. This tour can be divided into $2|P(\tau)|$ paths as follows. There are $|P(\tau)|$ $(= |V(\tau)|)$ paths starting at a city in $V(\tau)$ and ending at a city in $P(\tau)$ and $|P(\tau)|$ paths starting at a city in $P(\tau)$ and ending at a city in $V(\tau)$. Note that such a path has either the structure $[v, i_2, \ldots, i_u, p]$ or the structure $[p, j_2, \ldots, j_w, v]$, where p is a peak, v a valley, $v < i_2 < \cdots < i_u < p$ and $p > j_2 > \cdots > j_w > v$. Consider the subtour obtained from τ by replacing all $2|P(\tau)|$ paths by edges with the same endpoints as the paths. Clearly, each

city that is visited in this subtour is either a valley or a peak. So, by the first observation, this subtour contains a TEP. Let $[a,b]$ and $[x,y]$ be the two edges in the subtour that form a TEP. The second observation is that there are two edges in the paths that correspond to these two TEP-forming edges that form a TEP themselves, i.e. if e.g. there are two paths $[v_1, i_{12}, i_{13}, \ldots, i_{1u}, p_1]$ and $[v_2, i_{22}, i_{23}, \ldots, i_{2w}, p_2]$ in τ and $([v_1,p_1];[v_2,p_2])$ is a TEP in the subtour corresponding to τ, then there is a $k \in \{1,\ldots,u\}$ and a $t \in \{1,\ldots,w\}$ such that $([i_{1k}, i_{1,k+1}];[i_{2t}, i_{2,t+1}])$ is a TEP in τ. From these two observations it follows that each nonpyramidal tour contains a TEP. □

Recent research has been focused on the TSP restricted to symmetric Demidenko matrices that contain the class of symmetric Monge matrices as a subclass. A symmetric matrix $C = (c_{ij})$ is called a *symmetric Demidenko matrix* if

$$c_{ij} + c_{j+1,l} \leq c_{i,j+1} + c_{jl} \qquad \text{for all } 1 \leq i < j < j+1 < l \leq n. \tag{3}$$

Note that the class of Van der Veen matrices neither does contain nor is contained in the class of symmetric Demidenko matrices. However, by definition the symmetric Demidenko matrices are contained in the more general class of *asymmetric* Demidenko matrices (cf. Section 2.2.2 and Theorem 2.8). Thus we may formulate the following proposition.

Proposition 2.4 *The TSP restricted to symmetric Demidenko matrices is pyramidally solvable.* □

A symmetric $n \times n$ matrix $C = (c_{ij})$ is called a *Supnick* matrix if

$$c_{ij} + c_{j+1,l} \leq c_{i,j+1} + c_{jl} \leq c_{il} + c_{j,j+1} \qquad \text{for all } 1 \leq i < j < j+1 < l \leq n \tag{4}$$

It is easy to check that a Supnick matrix is a kind of symmetric Monge matrix with arbitrary diagonal entries (for a more precise relation we refer to Proposition 2.13). Moreover, the class of Supnick matrices is a subclass of both, the class of Demidenko matrices and the class of Van der Veen matrices. It follows that the TSP restricted to Supnick matrices is pyramidally solvable; however, if additional structures are exploited, the following stronger result can be obtained.

Theorem 2.5 (Supnick [109])
The TSP restricted to Supnick matrices is solved by the tour $\langle 1,3,5,7,\ldots,6,4,2\rangle$.

Proof. The following proof was given in Burkard and Van der Veen [19]. By Theorem 2.3 there is an optimal tour that is pyramidal. Using the following TI-technique, it can be shown that any pyramidal tour can be transformed into the tour $\langle 1,3,5,\ldots,4,2\rangle$ without increasing its length.

***TI*-technique Supnick**
Input: A pyramidal tour ρ.
Output: The tour $\langle 1,3,5,\ldots,n,\ldots,4,2\rangle$.
Step 1: Let k be the smallest city $(3 \leq k \leq n-1)$ such that
$\min\{\rho^{-1}(k), \rho(k)\} = k-1$.
If such a city k does not exist then STOP.
Direct the tour such that $\rho^{-1}(k) = k-1$.
Step 2: Transformation:
Update ρ by replacing $[\rho^{-1}(k-2), k-2]$ and $[k-1,k]$
by $[k-2,k]$ and $[\rho^{-1}(k-2), k-1]$. Return to Step 1.

The feasibility of the transformation in Step 2 follows from the observation that $k-2 < k-1 < k < \rho^{-1}(k-2)$ holds and from the second inequality in (4). It is easy to see that after each transformation the tour remains pyramidal. Since in each iteration one city is moved towards its place in the tour $\langle 1,3,5,\ldots,4,2\rangle$, the TI-technique ends after at most n iterations. □

Finally, we turn to the so-called Kalmanson matrices. A symmetric $n \times n$ matrix C is called a *Kalmanson* matrix if it fulfills the *Kalmanson conditions*

$$c_{ij} + c_{k\ell} \leq c_{ik} + c_{j\ell} \qquad \text{for all } 1 \leq i < j < k < \ell \leq n \tag{5}$$

$$c_{i\ell} + c_{jk} \leq c_{ik} + c_{j\ell} \qquad \text{for all } 1 \leq i < j < k < \ell \leq n. \tag{6}$$

Since condition (5) is exactly the symmetric Demidenko condition (3), the Kalmanson matrices are a subclass of the symmetric Demidenko matrices.

Theorem 2.6 (Kalmanson [62])
The TSP restricted to Kalmanson matrices is solved by the tour $\langle 1,2,\ldots,n-1,n\rangle$.

Proof. The following proof has been given in Burkard and Van der Veen [19]. By Proposition 2.4 there exists an optimal tour that is pyramidal. Using the following TI-technique, any pyramidal tour can be transformed into the tour $\langle 1,2,\ldots,n\rangle$ without increasing its length.

TI-**technique Kalmanson**
Input: A pyramidal tour ρ.
Output: The tour $\langle 1,2,\ldots,n-1,n\rangle$.
Step 1: Let k be the smallest city $(3 \leq k \leq n-2)$ such that
$\min\{\rho^{-1}(k), \rho(k)\} \neq k-1$.
If such a city k does not exist then STOP.
Direct the tour such that $\rho(k) < k-1$.
Step 2: Transformation:
Update ρ by replacing $[k, \rho(k)]$ and $[k-1, \rho(k-1)]$ by
$[\rho(k-1), \rho(k)]$ and $[k-1, k]$. Return to Step 1.

The feasibility of the transformation in Step 2 follows from the observation that $\rho(k) < k-1 < k < \rho(k-1)$ and from condition (6). It is easy to see that after each transformation the tour remains pyramidal. Since in each iteration one city is moved towards its position in $\langle 1,2,\ldots,n\rangle$, the TI-technique ends after at most n iterations. □

2.2.2 Asymmetric matrices

In the literature, several classes of asymmetric (or more precise, not necessarily symmetric) matrices have been investigated that (i) contain the class of Monge matrices as a subclass and (ii) for which the TSP is pyramidally solvable: for example, the class of Demidenko matrices and the class of generalized distribution matrices (recall that distribution matrix is just another name for Monge matrix). In this section we deal with these two classes and with some related pyramidally solvable TSP cases.

An $n \times n$ matrix $C = (c_{ij})$ is called a *generalized distribution matrix of type* (I) if

$$c_{ij} + c_{kp} + c_{pq} \leq c_{ip} + c_{pj} + c_{kq} \tag{7}$$

holds for all i,j,k,p,q in

$$S_I = \{i,j,k,p,q \in \{1,\ldots,n\} : i,j,k < p < q; i \neq j; i \neq k\}.$$

Analogously, a matrix is a generalized distribution matrix of type (II), type (III) and type (IV), respectively, if inequality (7) holds for the city sets

$$\begin{aligned} S_{II} &= \{i,j,k,p,q \in \{1,\ldots,n\} : i,j,q < p < k; i \neq j; j \neq q\}, \\ S_{III} &= \{i,j,k,p,q \in \{1,\ldots,n\} : k < p < i,j,q; i \neq j; j \neq q\}, \\ S_{IV} &= \{i,j,k,p,q \in \{1,\ldots,n\} : q < p < i,j,k; i \neq j; i \neq k\}, \end{aligned}$$

respectively. It can be shown that all four classes of generalized distribution matrices contain the Monge matrices as a subclass (Burkard and Van der Veen [19]).

Theorem 2.7 (Burkard and Van der Veen [19])
The TSP restricted to generalized distribution matrices of type (I), type (II), type (III) and type (IV), respectively, is pyramidally solvable.

Proof. Consider generalized distribution matrices of type (I) and the following TI-technique.

TI-technique Generalized Distribution (I)
Input: A tour τ.
Output: A pyramidal tour.
Step 1: Let p be the smallest peak of τ.
If $p = n$ then STOP (τ is pyramidal).
Define $i := \tau^{-1}(p)$ and $j := \tau(p)$.
Step 2: Let k be the smallest city such that $k < p < \tau(k)$.
Define $q := \tau(k)$.
Step 3: Transformation:
Update τ by replacing $[i,p],[p,j]$ and $[k,q]$ by
$[i,j],[k,p]$ and $[p,q]$. Return to Step 1.

Since C is a generalized distribution matrix of type (I) and $i < p$, $j < p$ and $k < p < q$ hold, the transformation in Step 3 is feasible. The technique terminates after at most $O(n^2)$ iterations because the pyramidality number of the tour after the transformation minus the pyramidality number of the tour before the transformation is $\max\{i,j\} - p < 0$. Similar TI-techniques can be stated for the other three types of generalized distribution matrices. Informally, TI-technique Generalized Distribution (I) can be described as: "Remove the smallest peak p from the tour by placing it in an over-going forward arc $[k,q]$". In the TI-techniques for type (II) and type (IV) 'forward arc' is to be replaced by 'backward arc' and for type (III) and type (IV) 'smallest peak' is to be replaced by 'largest valley'. □

An $n \times n$-matrix $C = (c_{ij})$ is called an (asymmetric) *Demidenko matrix* if it satisfies the following four conditions for all cities $i,j,l \in \{1,\ldots,n\}$ with $i < j < j+1 < l$:

$$\begin{aligned} &\text{(i)} && c_{ij} + c_{j,j+1} + c_{j+1,l} \leq c_{i,j+1} + c_{j+1,j} + c_{jl} \\ &\text{(ii)} && c_{ji} + c_{j+1,j} + c_{l,j+1} \leq c_{j+1,i} + c_{j,j+1} + c_{lj} \\ &\text{(iii)} && c_{ij} + c_{l,j+1} \leq c_{i,j+1} + c_{lj} \\ &\text{(iv)} && c_{ji} + c_{j+1,l} \leq c_{j+1,i} + c_{jl}. \end{aligned}$$

The following conditions (8) and (9) are equivalent to the Demidenko conditions (i)–(iv) (see Fazle Baki and Kabadi [46]):

$$c_{ij} + c_{l,l+1} + c_{uv} \leq c_{i,l+1} + c_{uj} + c_{lv} \qquad \text{for all } i \neq j \leq l < l+1 \leq u < v \tag{8}$$

$$c_{ij} + c_{l+1,l} + c_{vu} \leq c_{i,u} + c_{l+1,j} + c_{vl} \qquad \text{for all } i \neq j \leq l < l+1 \leq u < v. \tag{9}$$

(Verify that (8) and (9) with $i < j = l < l+1 = u < v$ implies conditions (i) and (iii), and that (9) and (8) with $j < i = l < l+1 = u < v$ implies (ii) and (iv). On the other hand, combining (iii) and (iv) with (i) and (ii) implies (8) and (9)). Note that if C is symmetric, the conditions (i)–(iv) reduce to just one condition, namely the symmetric Demidenko condition (3). In 1979 Demidenko gave a proof for Theorem 2.8 as part of his Ph.D. thesis. His paper is written in Russian, and the original proof is hardly accessible. Another rather complicated sketch of proof for Theorem 2.8 can be found in Gilmore, Lawler and Shmoys [57] (pp.105–109) together with the desperate question for a more elegant proof. Van der Veen, Sierksma and Van Dal [119] developed a TI-technique for the special case of *strong Demidenko matrices*, where in the conditions (i)–(iv) city $j+1$ is replaced by any k with $j < k < l$. We illustrate once more the strength of the TI-technique by giving the first short and simple proof for Demidenko's theorem.

Theorem 2.8 (Demidenko [40])
The TSP restricted to Demidenko matrices is pyramidally solvable.

Proof. In Step 1 below, we select a pyramidal subtour around the peak $u+1$. This subtour starts in some city i, ends in some city j, and inbetween visits all cities in the set $\{l, l+1, \ldots, u, u+1\}$. The number l is defined as the smallest possible number for which such a pyramidal subtour exists. From this one gets that $i \neq j$ and that $i, j \leq l-1 < l \leq u+1$. Moreover, if one of the cities i and j equals $l-1$ then it must be a valley.

***TI*-Technique Asymmetric Demidenko**
Input: A tour τ.
Output: A pyramidal tour.
Step 1: Let $u+1$ be the smallest peak of τ. If $u+1 = n$ then STOP.
Let $\langle i, \tau(i), \tau^2(i), \ldots, \tau^{u+2-l}(i), j\rangle$ be the pyramidal subtour in τ on the set of indices $\{i, j, l, l+1, \ldots, u, u+1\}$.
If τ contains an arc $[l-1, v]$ with $v > u+1$ then goto Step 2a,
otherwise (τ contains $[v, l-1]$ with $v > u+1$) goto Step 2b.
Step 2a: Transformation I:
Update τ into τ' by replacing the pyramidal subtour and the arc $[l-1, v]$ by $[i, j]$ and $[l-1, l], [l, l+1], \ldots, [u-1, u], [u, u+1]$.
Return to Step 1 with $\tau = \tau'$.
Step 2b: Transformation II:
Update τ into τ' by replacing the pyramidal subtour and the arc $[v, l-1]$ by $[i, j]$ and $[v, u+1], [u+1, u], \ldots, [l+1, l], [l, l-1]$.
Return to Step 1 with $\tau = \tau'$.

We use induction on the cardinality of $\{l, l+1, \ldots, u, u+1\}$ to prove the feasibility of the transformations: For $l = u+1$, (8) and (9) directly yield $c(\tau') \leq c(\tau)$.

Next, consider the case with $l = u$, i.e. the case where the pyramidal subtour is of the form $\langle i, u, u+1, j\rangle$ or $\langle i, u+1, u, j\rangle$. The feasibility of transformation I for $\langle i, u, u+1, j\rangle$ and the feasibility of transformation II for $\langle i, u+1, u, j\rangle$ follows from (8) and (9). Consider now transformation I for

the subtour $\langle i, u+1, u, j\rangle$. (The feasibility of transformation II for $\langle i, u, u+1, j\rangle$ is proved in an analogous way). It is easy to see that in this case

$$c(\tau') - c(\tau) + c_{iu} - c_{iu} + c_{uv} - c_{uv} =$$
$$(c_{ij} + c_{u-1,u} + c_{uv}) - (c_{iu} + c_{uj} + c_{u-1,v}) + (c_{iu} + c_{u,u+1} + c_{u+1,v}) - (c_{i,u+1} + c_{u+1,u} + c_{uv}).$$

It follows from condition (8) that $c(\tau') \leq c(\tau)$, i.e. that transformation I is feasible in this case, too.

Next, suppose that the transformations I and II are feasible for all the sets $\{m, m+1, \ldots, u, u+1\}$ with $l < m \leq u+1$ (inductive assumption). Our goal is to show that they are also feasible for the set $\{l, l+1, \ldots, u, u+1\}$. This will be shown only for transformation I, since the argument for transformation II is analogous. To simplify the further considerations, we use $\Delta(i, j, \{l, l+1, \ldots, u, u+1\}, l-1, v) = c(\tau') - c(\tau)$ as shorthand for the change in the length, when the arcs $[i, j]$ and $[l-1, l], [l, l+1], \ldots, [u-1, u], [u, u+1]$ are incorporated into the tour and the arc $[l-1, v]$ and the pyramidal subtour are removed from the tour. By inductive assumption, $\Delta(i, j, \{m, m+1, \ldots, u, u+1\}, m-1, v) \leq 0$ holds for all $l < m$ and for all $i, j \leq m-1 < m \leq u+1 < v$.

First, consider the case where the pyramidal subtour is of the form $\langle i, l, l+1, \ldots, m-1, \ldots, u+1, \ldots, m, j\rangle$ with $l < m$. Then

$$\begin{aligned}
&\Delta(i, j, \{l, l+1, \ldots, u, u+1\}, l-1, v) = \\
&\quad = c_{ij} + c_{l-1,l} + \sum_{x=l}^{u} c_{x,x+1} + c_{u+1,v} - c_{il} - \sum_{x=l}^{u+1} c_{x\tau(x)} - c_{l-1,v} + c_{mv} - c_{mv} \\
&\quad = c_{ij} + c_{l-1,l} + c_{mv} - c_{il} - c_{l-1,v} - c_{mj} + \sum_{x=m-1}^{u} c_{x,x+1} + \\
&\qquad\qquad + c_{u+1,v} - c_{m-1,\tau(m-1)} - \sum_{x=m+1}^{u+1} c_{x\tau(x)} - c_{mv} \\
&\quad = c_{ij} + c_{l-1,l} + c_{mv} - c_{il} - c_{mj} - c_{l-1,v} + \Delta(m-1, m, \{m+1, \ldots, u+1\}, m, v).
\end{aligned}$$

It follows from (8) and from the inductive assumption that $\Delta(i, j, \{l, l+1, \ldots, u, u+1\}, l-1, v) \leq 0$ holds in this case.

Finally, consider the case where the pyramidal subtour is of the form $\langle i, m, \ldots, u+1, \ldots, m-1, m-2, \ldots, l+1, l, j\rangle$ with $m > l$. Then

$$\begin{aligned}
&\Delta(i, j, \{l, l+1, \ldots, u, u+1\}, l-1, v) + c_{lv} - c_{lv} + c_{il} - c_{il} = \\
&\quad c_{ij} + c_{l-1,l} + c_{lv} - c_{il} - c_{l-1,v} - c_{lj} + \Delta(i, l, \{l+1, l+2, \ldots, u+1\}, l, v).
\end{aligned}$$

Again, (8) and the inductive assumption yield that $\Delta(i, j, \{l, l+1, \ldots, u, u+1\}, l-1, v) \leq 0$. This completes the proof of the theorem. □

In the Soviet literature, there is a large number of papers that report pyramidally solvable TSPs. All these results are summarized in Demidenko's celebrated paper [40] where the following theorem is formulated.

Theorem 2.9 (Demidenko [40])
The following classes of matrices are subclasses of Demidenko matrices:

- (Aizenshtat and Kravchuk, 1968 [5])
  $c_{ij} = a_i \times b_j$ *with* $0 \leq a_1 \leq a_2 \leq \cdots \leq a_n$ *and* $b_1 \geq \cdots \geq b_n \geq 0$.

- (Suprunenko, 1975 [111])
  C is symmetric and $c_{ij} \leq c_{kl}$ for $|i-j| < |k-l|$.

- (Aizenshtat, 1975 [4])
  C with $c_{i,i+1} = c_{i+1,i}$ for $1 \leq i \leq n-1$ and $c_{jl} \geq \max(c_{jk}, c_{kl})$, $c_{lj} \geq \max(c_{lk}, c_{kj})$ for $j < k < l$.

- (Suprunenko, 1976 [112])
  C is symmetric and $c_{kl} \geq c_{il}$ for $k < i \leq l$ and $c_{kl} \geq c_{kj}$ for $k < j \leq l$.

- (Klyaus, 1976 [63])

$$c_{ij} + c_{ji} \geq 0 \qquad c_{ij} + c_{jk} \leq c_{ik} \qquad c_{ji} + c_{kj} \leq c_{ki} \qquad \text{for } i < j < k. \tag{10}$$

- (Demidenko, 1976 [39])
  C is symmetric and $a_{ij} + c_{kl} \leq c_{ik} + c_{jl}$ for $i < j < k < l$. □

Matrices that fulfill the properties in (10) are called *Klyaus* matrices. It is interesting to observe that Klyaus matrices belong to both, the class of Demidenko matrices and the class of the four types of generalized distribution matrices.

We close this section with two more examples of pyramidally solvable asymmetric TSPs. The first example is a non-trivial generalization of Theorem 2.5.

Theorem 2.10 (Demidenko [41])
The TSP restricted to matrices C that fulfill the conditions

$$\begin{aligned}
c_{ij} + c_{i+1,j+1} &\leq c_{i,j+1} + c_{i+1,j} && \text{for all } 1 \leq i \leq n-3,\ i+2 \leq j \leq n-1 \\
c_{ji} + c_{j+1,i+1} &\leq c_{j+1,i} + c_{j,i+1} && \text{for all } 1 \leq i \leq n-3,\ i+2 \leq j \leq n-1 \\
c_{i,i+1} + c_{i+3,i+2} &\leq c_{i,i+2} + c_{i+3,i+1} && \text{for all even } i \text{ with } 2 \leq i \leq n-3 \\
c_{i+1,i} + c_{i+2,i+3} &\leq c_{i+2,i} + c_{i+1,i+3} && \text{for all odd } i \text{ with } 1 \leq i \leq n-3 \\
c_{i,i+1} + c_{i+1,i+2} + c_{i+2,i} &\leq c_{i+1,i} + c_{i+2,i+1} + c_{i,i+2} && \text{for all even } i \text{ with } 2 \leq i \leq n-2 \\
c_{i+1,i} + c_{i+2,i+1} + c_{i,i+2} &\leq c_{i,i+1} + c_{i+1,i+2} + c_{i+2,i} && \text{for all odd } i \text{ with } 1 \leq i \leq n-2
\end{aligned}$$

is solved by the tour $\langle 1, 3, 5, 7, \ldots, 6, 4, 2 \rangle$. □

The second example concerns a class of pyramidally solvable TSP class that can be defined by conditions similar to those defining generalized distribution matrices (for more details, we refer to Van der Veen [117]).

Theorem 2.11 *The TSP restricted to matrices $C = (c_{ij})$ with*

$$\begin{aligned}
c_{ik} + c_{qj} + c_{p,p+1} &\leq c_{i,p+1} + c_{pj} + c_{qk} \\
c_{ki} + c_{jq} + c_{p+1,p} &\leq c_{p+1,i} + c_{jp} + c_{kq}
\end{aligned}$$

for all $i, j, k, p, q \in \{1, \ldots, n\}$ with $i, j, k < p < p+1 \leq q$ and $i \neq k \neq j$ is pyramidally solvable. □

Refinements of the results presented here as well as additional pyramidally solvable cases can be found in Van der Veen [117], Warren [125] and Fazle Baki and Kabadi [46].

2.2.3 Max TSP on Blokh-Gutin matrices

In the maximization version of the TSP, one is interested in finding the *longest* tour. Clearly, by multiplying all entries of the distance matrix by -1, this problem turns back into the standard TSP. A nonnegative matrix $C = (c[i,j])$ is called a Blokh-Gutin matrix if $c[i,j] = 0$ for all i,j with $|i-j| \neq 1$.

Theorem 2.12 (Blokh and Gutin [11])
If the distance matrix is a Blokh-Gutin then there is always a longest tour that is pyramidal.

Proof. Consider the following TI-technique:

TI-Technique Blokh-Gutin

Input: A tour τ.

Output: A pyramidal tour.

Step 1: Let v be the largest valley of τ.

If $v = 1$ the STOP: (τ is pyramidal).

If $\tau(v) < \tau^{-1}(v)$ then go to Step 2a, else go to Step 2b.

Step 2a: Find the smallest city i such that $i < v < \tau(i)$.

(It is easy to see that such a city always exists.)

Let k be the smallest positive integer such that $\tau^k(v) \neq \tau^{k-1}(v) + 1$.

Note that $c[i,\tau(i)] = c[\tau^{-1}(v), v] = c[\tau^{k-1}(v), \tau^k(v)] = 0$.

Transformation:
Update the tour τ by replacing $[i,\tau(i)]$, $[\tau^{-1}(v), v]$ and $[\tau^{k-1}(v), \tau^k(v)]$ by $[i,v]$, $[\tau^{k-1}(v), \tau(i)]$ and $[\tau^{-1}(v), \tau^k(v)]$, respectively. Return to Step 1.

Step 2b: Find the largest city j such that $\tau(j) < v < j$.

(It is easy to see that such a city always exists.)

Let l be the smallest positive integer such that $\tau^{-l}(v) \neq \tau^{-l+1}(v) + 1$.

Note that $c[j,\tau(j)] = c[v,\tau(v)] = c[\tau^{-l}(v), \tau^{-l+1}(v)] = 0$.

Transformation:
Update τ by replacing $[j,\tau(j)]$, $[v,\tau(v)]$ and $[\tau^{-l}(v), \tau^{-l+1}(v)]$ by $[j, \tau^{-l+1}(v)]$, $[v,\tau(j)]$ and $[\tau^{-l}(v), \tau(v)]$, respectively. Return to Step 1.

It is easy to see that the transformations in Step 2a and Step 2b cannot decrease the length of the tour. Moreover, after each transformation the pyramidality number of the tour is decreased, i.e. the above technique ends after a finite number of steps.

2.3 Recognition of specially structured matrices

All the combinatorial structures defined so far would be of little use, if we were not able to recognize them in polynomial time (you cannot exploit properties, if you do not know that they are present). Therefore, this section deals with the *recognition* of these combinatorial structures. In the following we state (without proof) other equivalent characterizations of some of the specially structured matrices introduced so far. For example, a definition for Monge matrices that is equivalent to (2) but simpler to verify is the following. An $n \times n$ matrix C is a Monge matrix if and only if

$$c_{ij} + c_{i+1,j+1} \leq c_{i,j+1} + c_{i+1,j} \qquad \text{for all } 1 \leq i,j \leq n. \tag{11}$$

A symmetric $n \times n$ matrix C is a Demidenko matrix if and only if

$$\max_{1 \leq i \leq j-1} \{c_{ij} - c_{i,j+1}\} \leq \min_{j+2 \leq \ell \leq n} \{c_{j,\ell} - c_{j+1,\ell}\} \quad \text{for all } 2 \leq j \leq n-2. \tag{12}$$

A symmetric $n \times n$ matrix C is a Kalmanson matrix if and only if

$$c_{i,j+1} + c_{i+1,j} \leq c_{ij} + c_{i+1,j+1} \qquad \text{for all } 1 \leq i \leq n-3,\ i+2 \leq j \leq n-1 \tag{13}$$

$$c_{i,1} + c_{i+1,n} \leq c_{in} + c_{i+1,1} \qquad \text{for all } 2 \leq i \leq n-2. \tag{14}$$

Since all these conditions can be verified in $O(n^2)$ time, it can be decided in $O(n^2)$ time whether C is a Monge (Kalmanson, Demidenko) matrix. Burkard and Deĭneko observed a special relationship between Monge and Supnick matrices.

Proposition 2.13 (Burkard and Deĭneko, see Burkard [14])
Let C be a symmetric $n \times n$ matrix. Then the following holds.

(i) If C is a Monge matrix, then C is also a Supnick matrix.

(ii) If C is a Supnick matrix, then the matrix $\widehat{C}$ obtained from C by changing the diagonal entries to $\widehat{c}_{ii} := c_{i-1,i} + c_{i,i+1} - c_{i-1,i+1}$ for $i = 2,\ldots,n-1$, $\widehat{c}_{11} := c_{21} + c_{12} - \widehat{c}_{22}$ and $\widehat{c}_{nn} := c_{n-1,n} - c_{n,n-1} - \widehat{c}_{n-1,n-1}$ is a Monge matrix. □

By Theorem 2.13 the Supnick condition can be verified in $O(n^2)$ time, too.

Next, we turn to so-called *permuted* specially structured matrices. Note that the definitions of Monge, Supnick, Kalmanson, Demidenko, and Van der Veen matrices crucially depend on the numbering of the cities. Hence, the following question arises:

> Given an $n \times n$ distance matrix $C = (c_{ij})$, does there exist a renumbering of the cities, i.e. a permutation σ of the rows and columns of C, such that the resulting matrix $(c_{\sigma(i)\sigma(j)})$ is a Monge (Supnick, Kalmanson, Demidenko, Van der Veen) matrix?

In case the renumbering exists, matrix C is called a *permuted* Monge (Supnick, Kalmanson, Demidenko, Van der Veen) matrix. We are interested in finding the renumbering in polynomial time. Below we shortly report some results on this question.

In 1979, Deĭneko and Filonenko [32] showed that permuted Monge matrices can be recognized in $O(n^2)$ time. This algorithm is also described e.g. in Burkard, Klinz and Rudolf [17]. Although Supnick matrices are closely related to Monge matrices, the recognition of permuted Supnick matrices turned out to be not that easy. Deĭneko, Rudolf and Woeginger [34] constructed an $O(n^2 \log n)$ recognition algorithm for permuted Supnick matrices, thus losing a $\log n$-factor in comparison to Monge matrices. The recognition of permuted Demidenko and permuted Van der Veen matrices is still an open problem. Permuted Kalmanson matrices, which form a subset of the permuted Demidenko matrices, can be recognized in $O(n^2 \log n)$ time, see Deĭneko, Rudolf and Woeginger [35].

2.4 The master tour problem

Permuted Kalmanson matrices have a nice relationship with the so-called master tour problem. A *master tour* π for a set V of cities fulfills the following property. For every $V' \subseteq V$, an optimum traveling salesman tour for V' is obtained by removing from π the cities that are not in V'. Given the distance matrix C for a set of cities, the *master tour problem* is the problem of deciding whether this set of cities possesses a master tour. This problem was first formulated by Papadimitriou [83, 84], who considered it to be a "good candidate for a natural $\Sigma_2\mathbf{P}$-complete problem". By the theorem below, a master tour exists if and only if the distance matrix is a permuted Kalmanson matrix. Hence, the master tour problem is solvable in $O(n^2 \log n)$ time, and the conjecture of Papadimitriou is false (unless $\Sigma_2\mathbf{P}=\mathbf{P}$).

Theorem 2.14 *For an $n \times n$ symmetric distance matrix C, the permutation $\langle 1, 2, \ldots, n\rangle$ is a master tour if and only if C is a Kalmanson matrix.*

Proof. (Only if): Assume that $\langle 1, 2, \ldots, n\rangle$ 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, we obtain

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

which are exactly the conditions (6) and (5). Hence, C is a Kalmanson matrix.
(If): Let $K = \{x_1, \ldots, x_k\}$ be a subset of $\{1, 2, \ldots, n\}$. The corresponding submatrix of C is again a Kalmanson matrix and by Theorem 2.6 the tour $\langle x_1, \ldots, x_k\rangle$ is an optimal tour for K. Consequently, $\langle 1, 2, \ldots, n\rangle$ is a master tour. □

2.5 The Bottleneck TSP

In the bottleneck version of the TSP, one is interested in finding a tour that minimizes the length of the longest traveled edge, i.e. a tour π that minimizes

$$\max_{1 \leq i \leq n} \left\{ c_{i\pi(i)} \right\}.$$

The result in Theorem 2.1 can be carried over to compute the shortest pyramidal tour with respect to the bottleneck criterion in $O(n^2)$ time. Burkard and Sandholzer [18] identified several conditions on the distance matrix that guarantee the existence of an optimal bottleneck tour that is pyramidal.

Theorem 2.15 (Burkard and Sandholzer [18])
Each of the following conditions on the cost matrix $C = (c_{ij})$ of a bottleneck TSP implies that this problem has an optimal solution which is pyramidal.

(a) $\max\{c_{ir}, c_{js}, c_{st}\} \leq \max\{c_{is}, c_{jt}, c_{sr}\}$ *for* $r \neq i \neq j$ *and* $1 \leq i, j, r < s < t \leq n$.

(b) C *and the transposed matrix* C^t *fulfill* $\max\{c_{ir}, c_{js}, c_{s,s+1}\} \leq \max\{c_{is}, c_{j,s+1}, c_{sr}\}$ *for* $r \neq i \neq j$ *and* $1 \leq i, j, r < s \leq n-1$.

(c) C *and the transposed matrix* C^t *fulfill* $\max\{c_{ir}, c_{s,s+1}, c_{tj}\} \leq \max\{c_{i,s+1}, c_{sj}, c_{tr}\}$ *for* $i \neq r \neq j$ *and* $1 \leq i, j, r < s < s+1 < t \leq n$.

(d) C *is symmetric and fulfills* $\max\{c_{ij}, c_{kl}\} \leq \max\{c_{ik}, c_{jl}\}$ *for* $1 \leq i < j < k < l \leq n$. □

3 Solvable cases of the Euclidean TSP

In this section we consider the *Euclidean TSP*, i.e. the TSP where the cities are represented by points in the two-dimensional Euclidean plane and the distances are measured according to the Euclidean metric. We will write $d(x,y)$ to denote the Euclidean distance between points x and y. This special TSP case is still $\mathcal{NP}$-hard (see e.g. Papadimitriou [82] or Chapter 3 in [70]). However, in the Euclidean case the shortest TSP tour does not intersect itself and thus geometry makes the problem somewhat easier.

Section 3.1 deals with some simple special cases that result from convex sets. Section 3.2 deals with the case where the cities lie on a small number of line segments and Section 3.3 deals with the necklace TSP.

3.1 Special cases related to the convex TSP

In 1956, in one of the first papers on the TSP, Flood [49] observed that "in the Euclidean plane the minimal TSP tour does not intersect itself". This observation is an immediate consequence of the quadrangle inequality (see Monge [75], 1781) which states that in a convex planar quadrangle $p_1p_2p_3p_4$

$$d(p_1,p_2)+d(p_3,p_4) \leq d(p_1,p_3)+d(p_2,p_4) \tag{15}$$

$$d(p_2,p_3)+d(p_1,p_4) \leq d(p_1,p_3)+d(p_2,p_4) \tag{16}$$

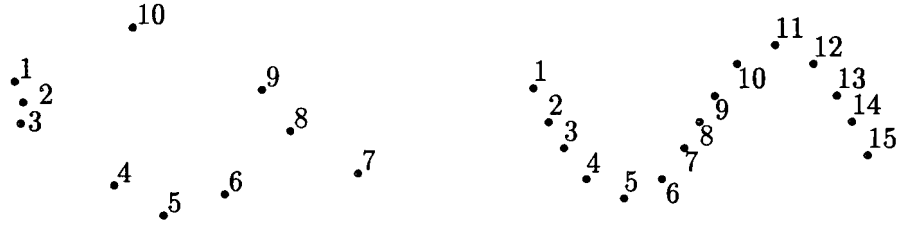
hold. I.e. the total length of the diagonals is at least the total length of two opposite sides (and unless all four points lie on a common line, even strict inequality holds). In case the shortest TSP tour contains two intersecting edges (p_1,p_3) and (p_2,p_4), these edges would constitute the diagonals of a convex quadrangle. One could swap them with an appropriate pair of opposite sides and thus decrease the length of the shortest tour. Flood's observation also implies the following statement.

Lemma 3.1 (Folklore)
If all cities lie on the boundary of a convex polygon, the optimal tour is a cyclic walk along the boundary of the polygon (in clockwise or counterclockwise direction). □

Quintas and Supnick [94] proved an analogous result for the case that the cities lie on a 2-sphere and consecutive cities are connected by minor geodesic arcs. In [93] and [95], Quintas and Supnick derive a complete solution for finding a *longest* traveling salesman tours in case of convex planar Euclidean point sets. Moreover, by applying standard techniques from computational geometry, the cyclic order of n cities on the boundary of a convex polygon can be determined in $O(n \log n)$ time, if their distance matrix C is given. This can be done even without knowing or assigning coordinates to the points.

Kalmanson [62] generalized the purely geometrical concept of convexity and arrived at the algebraic conditions (5) and (6) as stated in Section 2.2.1. Observe that if the cities of a convex point set are numbered along the convex hull, the resulting distance matrix is a Kalmanson matrix. However, there are also non-convex point sets whose distance matrix is a Kalmanson matrix (cf. Figure 1 for an illustration). Kalmanson [62] proved that for the TSP with a Kalmanson distance matrix, the tour $\langle 1,2,3,\ldots,n\rangle$ is always a shortest tour (see Theorem 2.6).

It is also interesting to consider the geometric versions of the other specially structured matrices introduced in Section 2. The right-hand side of Figure 1 gives an illustration for a Euclidean TSP with a Demidenko distance matrix. The (pyramidal) optimal TSP tour is $\langle 1,2,3,4,5,6,7,8,15,14,13,12,11,10,9\rangle$. Note also that the Kalmanson point set depicted in the



A Kalmanson point set A Demidenko point set

Figure 1: Solvable cases of the planar TSP

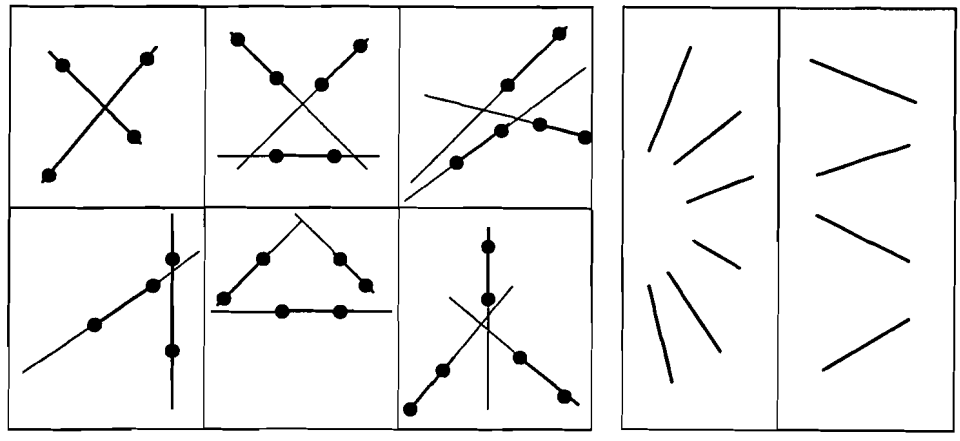
left-hand side of Figure 1 is 'almost convex' and that the numbering follows the 'almost convex hull'.

Deĭneko, Rudolf, Van der Veen and Woeginger [33] dealt with the problem of recognizing special combinatorial structures in the distance matrix of Euclidean point sets. They proved that for the $n \times n$ distance matrix C of a Euclidean point set, it can be decided in $O(n^2)$ time whether C is a permuted Kalmanson matrix and in $O(n^4)$ time whether C is a permuted Demidenko matrix. The combinatorial structure of Euclidean Supnick point sets is rather primitive. In case a Supnick set contains $n \geq 9$ points, all these points must lie on a common straight line (see Quintas and Supnick [96]). This result is based on the observation that no convex set of five points in the Euclidean plane has a Supnick distance matrix and on the fact that every set of $n \geq 9$ points that do not lie on a straight line contains a convex set of five points. Moreover, Quintas and Supnick [96] described point distributions in higher dimensional spaces such that the Supnick conditions are fulfilled for more than 8 points.

3.2 The k-line TSP

This section deals with a special case where the cities lie on k parallel (or almost parallel) lines in the Euclidean plane. Such problems arise for example in the fabrication of printed circuit boards, where a laser drills holes in certain places of the board which usually lie on parallel lines. It is easily seen that minimizing the total distance covered by the laser is equivalent to solving a TSP (cf. Lin and Kernighan [72]).

Cutler [28] developed an $O(n^3)$ time and $O(n^2)$ space dynamic programming algorithm for the case with $k = 3$ parallel lines. Rote [102] generalized Cutler's result and derived an $O(n^k)$ algorithm for an arbitrary but fixed number of $k \geq 4$ lines. This algorithm strongly builds on Flood's result that the shortest tour does not intersect itself. Moreover, Rote observed that the lines need not be exactly parallel but might be "slightly perturbated", i.e. as long as the cities lie on k-line segments which do not contain one of the six forbidden configurations shown in the right-hand side of Figure 2, Rote's algorithm works and finds the optimum tour in $O(n^k)$ time. However, the line segments on which the cities lie are allowed to form 'half-stars' or 'zigzags' as depicted in left part of Figure 2. Such sets of line segments are called *quasi-parallel* line segments.



Forbidden configurations Allowed configurations

Figure 2: Illustrations to the N-line TSP

In 1994, Deĭneko, Van Dal and Rote [37] investigated another related special case, the *convex-hull-and-line* TSP. Here $n - m$ cities lie on the boundary of the convex hull of the n cities, and the remaining m cities lie on a line segment inside of this convex hull (see the left part of Figure 3). Clearly, this is another extension of Cutler's special case (in case the upper chain and the lower chain of the convex hull both degenerate to straight lines, one arrives at the 3-line TSP). Deĭneko, Van Dal and Rote [37] give an $O(nm) = O(n^2)$ time and $O(n)$ space algorithm and thus obtain an improvement in both running time and space requirements over Cutler's algorithm. The algorithm computes a shortest path in a related edge-weighted directed network. The edge-weights in this network arise from Euclidean distances, and it can be shown that they fulfill a certain Monge-like property. This allows to apply the fast search techniques for Monge matrices developed by Aggarwal, Klawe, Moran, Shor and Wilber [2] and thus yields the improved running time.

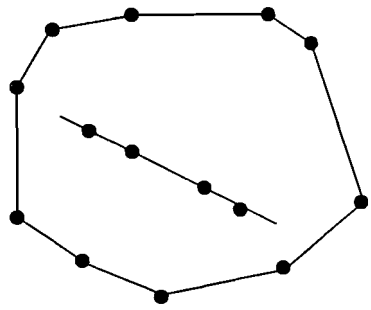
Next, we will describe a more general case of the TSP that generalizes and unifies the three cases detected by Cutler, by Rote, and by Deĭneko, Van Dal and Rote. This case is due to Deĭneko and Woeginger [38]. Assume that the set P of cities is partitioned into k disjoint classes $P_1, \ldots, P_k$, $\cup_{i=1}^{k} P_i = P$. For $1 \leq i \leq k$, define $n_i = |P_i|$. Assume moreover that every class P_i is linearly ordered by the relation $\prec$. The undirected edge between two cities a and b in P is denoted by (a, b). For $a, b \in P$, the edge (a, b) is called a *long-chord* if (i) a and b belong to the same class P_i and (ii) if there exists a city $c \in P$, $a \neq c \neq b$, inbetween them, i.e. with $a \prec c \prec b$ or $b \prec c \prec a$. A set $F = \{(a_1, a_2), (a_3, a_4), \ldots, (a_{2f-1}, a_{2f})\}$ of edges over P is called a *fence* if

(F1) for every ℓ with $1 \leq \ell \leq f$, the cities $a_{2\ell-1}$ and $a_{2\ell}$ belong to distinct classes

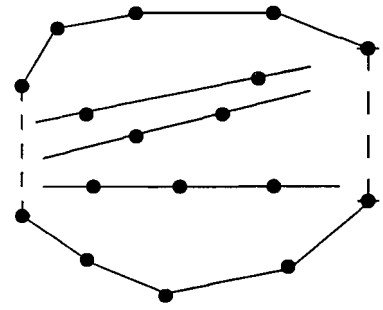
(F2) for every ℓ with $1 \leq \ell \leq f$, there exists a city s, $1 \leq s \leq 2f$, $2\ell - 1 \neq s \neq 2\ell$ that fulfills $a_{2\ell-1} \prec a_s$ or $a_{2\ell} \prec a_s$.

A set E of edges over P is called *fence-free* if it does not contain fences as subsets and it is called *long-chord-free*, if it does not contain long-chords. Observe that long-chord-freeness and fence-freeness are monotone properties of sets of edges and carries over to subsets.

A dynamic programming approach quite similar to that one used by Rote [102] yields the following result.



Convex-hull-and-line TSP



Convex-hull-and-k-line TSP

Figure 3: Combining the convex-hull-and-line and the k-line TSP

Theorem 3.2 (Deĭneko and Woeginger [38])
Let $k \geq 3$ be some fixed integer. Then for any set P of n cities that is partitioned into k linearly ordered classes $P_1, \ldots, P_k$, the shortest long-chord-free and fence-free tour can be computed in $O(\prod_{i=1}^{k} n_i) = O(n^k)$ overall time. □

Now we review the k-line TSP. Without loss of generality assume that all k lines are horizontal. We order the points along every line from left to right, i.e. $a \prec b$ if a lies on the same line and to the left of b. Consider the shortest tour τ. Obviously, τ cannot contain long-chords since then it would intersect itself and contradict Flood [49]. It can also be shown that if a subset of the edges in τ would form a fence, then τ had to intersect itself. Summarizing, the shortest tour is long-chord-free and fence-free and Theorem 3.2 applies. In the convex-hull-and-line TSP, the first class of cities is ordered along the upper chain of the convex hull, the second class is ordered along the line and the third class along the lower chain of the hull. Again, it can be shown that the shortest tour is long-chord-free and fence-free.

The most general Euclidean speciál case around Cutler's result is the following *convex-hull-and-k-line* TSP (see Figure 3). There are $n - m$ points lying on the upper chain and on the lower chain of the convex hull and there are m points lying on k quasi-parallel line segments in the interior of the convex hull. The carrying lines through all these quasi-parallel segments intersect the hull in two common edges. Once more, it can be shown that the shortest TSP tour is fence-free and long-chord-free with respect to the natural ordering along the segments and along the chains of the hull. This results in an $O(n^{k+2})$ algorithm.

One open problem in this area is to find a fast algorithm for the so-called *x-and-y-axes* TSP. In this special case, all cities are situated on the x-axis and on the y-axis of an orthogonal coordinate system for the Euclidean plane. Since this case contains the first one of the forbidden configurations in Figure 2, the methods of this section do not apply.

3.3 The Necklace TSP

This section deals with a special Euclidean TSP case which is reported in Reinhold [99] and Sanders [104]. Assume that there exist n circles around the n cities such that every cycle intersects exactly two adjacent circles (see Figure 4). Then the circles are said to form a *necklace* and

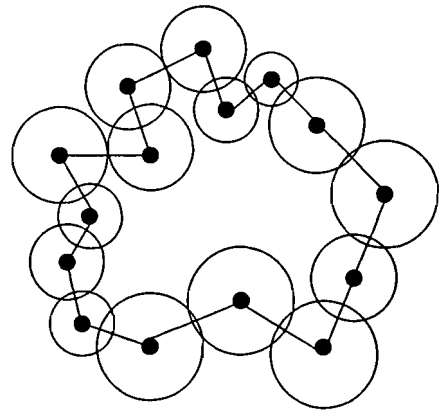


Figure 4: The necklace TSP

the cities are said to fulfill the *necklace condition*. It can be shown that in this case the tour given by the necklace is the unique optimal tour (Sanders [104]).

Rote [101] developed an algorithm which tests whether a necklace tour exists for n given cities $p_1, \ldots, p_n$ in the plane and constructs the tour in $O(n^2 \log n)$ time and $O(n)$ space. The basic idea is to investigate the following transportation problem. Let $a_{ij} = d(p_i, p_j)$ be the Euclidean distance between city p_i and p_j. Then there exists a necklace tour if and only if the capacitated transportation problem

$$\begin{array}{ll} \min & \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_{ij} \\ \text{s.t.} & \begin{cases} \sum_{j=1}^{n} x_{ij} = 2 & \text{for } i = 1, 2, \ldots, n \\ \sum_{i=1}^{n} x_{ij} = 2 & \text{for } j = 1, 2, \ldots, n \\ 0 \leq x_{ij} \leq 1 & \text{for } i, j = 1, 2, \ldots, n \end{cases} \end{array}$$

has a unique optimal solution which is symmetric and corresponds to a tour. Let (x_{ij}^*) be a primal and (u_i^*, v_j^*) be a dual optimal solution of the capacitated transportation problem above. We link two points p_i and p_j by an edge if $x_{ij}^* + x_{ji}^* = 2$. Otherwise, by symmetry of the solution, we have $x_{ij}^* + x_{ji}^* = 0$ and then the two points are not linked. Hence, it is easy to check whether the solution (x_{ij}^*) corresponds to a tour. The radius r_i of the disk around p_i can be obtained from the dual solution by

$$r_i = \frac{1}{2}(u_i^* + v_i^*).$$

Knowing these radii it is easy to test whether the solution is unique or not, since in the case of a *unique* necklace tour the following inequalities must hold:

$$\begin{array}{rl} r_i + r_j \geq c_{ij} & \text{if } \{i, j\} \in T \\ r_i + r_j < c_{ij} & \text{if } \{i, j\} \notin T \\ r_i > 0 & \text{for } 1 \leq i \leq n. \end{array}$$

In order to obtain the low running time for solving the capacitated transportation problem, Rote uses ideas from computational geometry. He shows that the particular transportation problem can

be solved by shortest augmenting path computations in a related sparse graph $G^{(2)}$. The graph $G^{(2)}$ is defined as follows. Let $d^{(2)}(p)$ denote the distance from p to the second nearest neighbor of p in the given set of cities. $G^{(2)}$ has as vertex set the n cities $p_1, \ldots, p_n$. There is an edge between p_i and p_j in $G^{(2)}$ if and only if

$$d^{(2)}(p_i) + d^{(2)}(p_j) \geq a_{ij}.$$

Edelsbrunner, Rote and Welzl [44] proved that a necklace tour is always a subgraph of $G^{(2)}$ and that $G^{(2)}$ has at most $37n$ edges and therefore is sparse.

4 The permuted Monge TSP: Theory

For an $n \times n$ matrix C and two permutations π and ρ, we denote the matrix $(c_{\pi(i),\rho(j)})$ by $C_{\pi,\rho}$. A matrix C is called a *permuted Monge matrix* if there exists a permutation ϕ such that $C_{\varepsilon,\phi}$ is a Monge matrix, where ε denotes the identity permutation. Since for the linear assignment problem on Monge matrices the identity permutation yields an optimum solution (see e.g. [17]), it easily follows that for a permuted Monge matrix C the permutation ϕ is an optimal assignment. Throughout this section, ϕ will be used to denote the optimal assignment of the distance matrix C.

This section deals with the theory of subtour-patching and its application to the TSP on permuted Monge matrices. We will show that permuted Monge matrices always possess optimal tours that are of a very specific form. This leads to several solvable special TSP cases, although the TSP on permuted Monge matrices is in general not polynomially solvable.

The idea of *subtour-patching* is the following. Start with finding an optimal assignment ϕ for the linear assignment problem. If ϕ is a tour, it clearly is a shortest tour and we are done. Otherwise, ϕ consists of several subtours (also called factors). In this case, patch the subtours together to yield a single tour, namely an *optimal* tour. Summarizing, the problem is: "Given an optimal assignment ϕ, find a permutation $\alpha^\star$ such that $\phi \circ \alpha^\star$ is an optimal tour."

The idea of patching the factors of the optimal assignment dates back to (at least) the beginning of the 1960's, see e.g. Szwarc [113]. Apparently, Gilmore and Gomory [56] were the first who used this idea to obtain an efficiently solvable special case.

4.1 The Subtour-patching strategy

Let ϕ be a permutation with two subtours ϕ_1 and ϕ_2 where $i \in \phi_1$ and $j \in \phi_2$ hold. Essentially, the subtour-patching scheme relies on the fact that if we post-multiply ϕ by the transposition $\langle i, j \rangle$, this will result in a permutation where the two subtours ϕ_1 and ϕ_2 are patched together. Hence, the number of subtours of $\phi \circ \langle i, j \rangle$ is one less than the number of subtours of ϕ. A permutation α will be called a *patching permutation* for ϕ, if $\phi \circ \alpha$ is a cyclic permutation. Permutation α will be called an *optimal patching permutation* for ϕ, if $\phi \circ \alpha$ is an optimal tour.

In order to describe some specific patching permutations we will introduce the *patching graph* G_ϕ. Suppose that ϕ consists of $m \geq 2$ factors ϕ_k, i.e. $\phi = \phi_1 \phi_2 \cdots \phi_m$. The *patching graph* $G_\phi = (V, E)$ has for every factor of ϕ a (unique) corresponding vertex in V. Every edge in E corresponds to an adjacent transposition $\langle i, i+1 \rangle$. If city i is in subtour ϕ_j and city $(i+1)$ is in subtour ϕ_k, then the two vertices labeled ϕ_j and ϕ_k are connected by an edge labeled $(i, i+1)$. It is easy to see that every patching graph G_ϕ is a Eulerian multigraph with at most $(n-1)$ edges.

As an example, consider $\phi = \phi_1 \phi_2 \phi_3 \phi_4 = \langle 1, 7 \rangle \langle 2, 5 \rangle \langle 3 \rangle \langle 4, 6 \rangle$. For this permutation, the graph G_ϕ in Figure 5 has four vertices and six edges, where $E = \{(1,2), (2,3), (3,4), (4,5), (5,6), (6,7)\}$.

A permutation that results from multiplying a set of adjacent transpositions that form a spanning tree in the patching graph G_ϕ is called a *tree permutation*. For instance, in the example in Figure 5

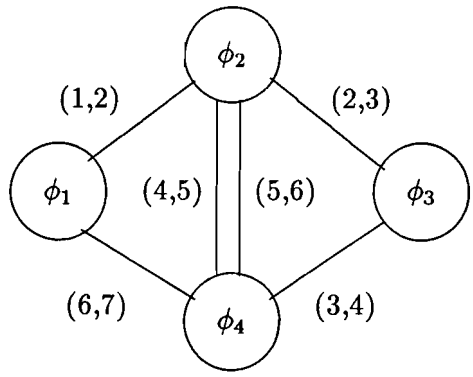


Figure 5: Graph G_ϕ for $\phi = \langle 1,7\rangle\langle 2,5\rangle\langle 3\rangle\langle 4,6\rangle$.

the permutation

$$\langle 1,2\rangle \circ \langle 2,3\rangle \circ \langle 3,4\rangle = \langle 1,2,3,4\rangle = \langle 1,2,3,4\rangle\langle 5\rangle\langle 6\rangle\langle 7\rangle$$

is a tree permutation as the edges corresponding to the transpositions $\langle 1,2\rangle, \langle 2,3\rangle, \langle 3,4\rangle$ form a spanning tree in G_ϕ.

The theory of subtour patching is based on the following result.

Theorem 4.1 (Gilmore and Gomory [56])
Every tree permutation is a patching permutation. □

Note that in Theorem 4.1 there is no condition on the order in which the adjacent transpositions are multiplied. This is important since in general different orders of multiplications will result in different permutations. Moreover, there is a close connection between permutations obtained from multiplying adjacent transpositions and pyramidal tours as is shown in the following proposition.

Proposition 4.2 *The set of all permutations that can be obtained by multiplying adjacent transpositions* $\langle 1,2\rangle, \langle 2,3\rangle, \ldots, \langle n-1,n\rangle$ *is exactly the set of all pyramidal tours on* $\{1,2,\ldots,n\}$. □

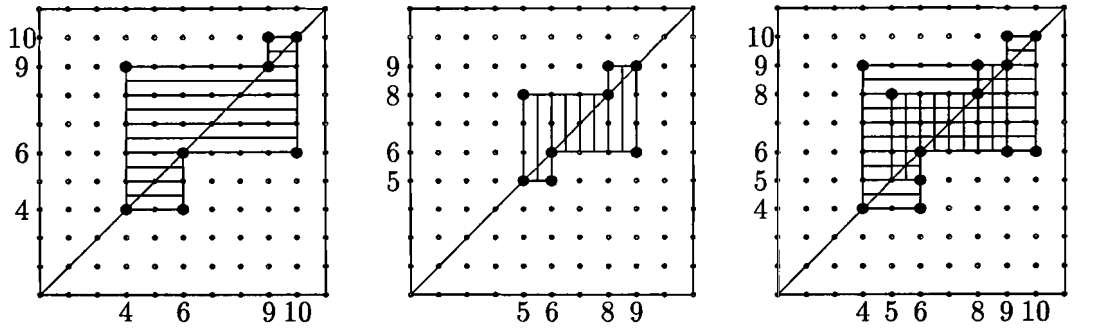
4.1.1 Subtour-patching on permuted Monge matrices

The following result justifies the introduction of the patching graph and the definition of tree permutations.

Theorem 4.3 (Burdyuk and Trofimov [13]; Gilmore, Lawler and Shmoys [57])
Let $C_{\varepsilon,\phi}$ *be a Monge matrix. For any cyclic permutation* τ*, there exists a spanning tree* $T = \{(i_1, i_1+1), \ldots, (i_{m-1}, i_{m-1}+1)\}$ *in the graph* G_ϕ *and a sequence* σ *for multiplying the transpositions corresponding to* T *such that the permutation*

$$\phi_T = \phi \circ \langle i_{\sigma(1)}, i_{\sigma(1)}+1\rangle \circ \cdots \circ \langle i_{\sigma(m-1)}, i_{\sigma(m-1)}+1\rangle$$

is a cyclic permutation with $c(\phi_T) \leq c(\tau)$. □



$A(\langle 4,9,10,6\rangle) \subseteq Q$ $A(\langle 5,8,9,6\rangle) \subseteq Q$ $A(\langle 5,8,9,6\rangle \subseteq A(\langle 4,9,10,6\rangle)$

Figure 6: Illustrations of the definition of regions

This section is mainly devoted to proving the above theorem. We will apply some geometric flavored ideas from Burdyuk and Trofimov [13]. We start with some more definitions and two lemmas.

By deleting vertices from a pyramidal tour we get a so-called *pyramidal subtour*. Let $\lambda = \langle i_1, i_2, \ldots, i_q\rangle$ be a pyramidal subtour with $i_1 = \min\{i_l : l = 1, \ldots, q\}$. By the *region* $A(\lambda)$, we denote the set of points in $\mathbb{R}^2$ that is enclosed by drawing the line pieces between the coordinates

$$(i_1, i_1) \rightarrow (i_1, i_2) \rightarrow (i_2, i_2) \rightarrow (i_2, i_3) \rightarrow \cdots \rightarrow (i_{q-1}, i_q) \rightarrow (i_q, i_q) \rightarrow (i_q, i_1) \rightarrow (i_1, i_1).$$

Furthermore, we say that a pyramidal subtour μ is *contained* in the pyramidal subtour λ if $A(\mu) \subseteq A(\lambda)$. See Figure 6 for an example.

To clarify further observations, we sketch an alternative procedure for constructing the region $A(\lambda)$. The region is constructed by deleting rectangles from the square Q defined by the four points $(1,1)$, $(1,n)$, (n,n), $(n,1)$. We will use the following notations. Let $[i,j]^0$ be the unit square with the vertices (i,j), $(i,j+1)$, $(i+1,j+1)$, and $(i+1,j)$, $[i,i]^+$ the rectangle with the vertices $(1,i)$, $(1,n)$, (i,n), (i,i), and $[i,i]^-$ the rectangle with the vertices $(i,1)$, (i,i), (n,i), $(n,1)$. The region $A(\lambda)$ for $\lambda = \langle i_1, \ldots, i_q\rangle$ can be constructed as follows. For all i with $1 < i \leq i_1$, the rectangles $[i,i]^+$, $[i,i]^-$, and $[i-1,i-1]^0$ are removed from the square Q. For all i with $\max\{i_l : l = 2, \ldots, q\} < i < n$, the rectangles $[i,i]^+$, $[i,i]^-$, and $[i,i]^0$ are removed from the square Q. For all i_k with $i_1 < i_k < \max\{i_l : l = 2, \ldots, q\}$, either $[i_k, i_k]^+$ or $[i_k, i_k]^-$ is deleted from Q. If $i_{k-1} < i_k$, then $[i_k, i_k]^+$ is removed, otherwise $[i_k, i_k]^-$ is removed.

Lemma 4.4 *Let $\lambda = \langle i_1, i_2, \ldots, i_q\rangle$ be a pyramidal subtour. Furthermore, let $\{j, j+1, \ldots, j+t\}$ be a set of cities such that $i_1 \leq j$ and $j + t \leq \max\{i_k : k = 1, \ldots, q\}$. Then there exists a pyramidal subtour μ on the cities $j, j+1, \ldots, j+t$, such that μ is contained in λ.*

Proof. The pyramidal subtour can be determined as follows. Start by taking $\langle i_1, i_2, \ldots, i_q\rangle$ and delete all cities i_k that are not in $\{j, j+1, \ldots, j+t\}$. Note that the remaining subtour is still pyramidal. Now insert all cities from $\{j, j+1, \ldots, j+t\}$ that are not yet contained in such a way into the subtour that the result remains pyramidal (in case all cities were deleted from $\langle i_1, \ldots, i_q\rangle$ during the first step, we simply set $\mu = \langle j, j+1, \ldots, j+t\rangle$). It is easy to verify that the resulting subtour μ is contained in λ. □

Let $D = (d_{ij})$ be a Monge matrix. We define the *area* of the unit square in $\mathbb{R}^2$ that is enclosed by the vertices (k,l), $(k,l+1)$, $(k+1,l+1)$ and $(k+1,l)$ by

$$d_{k,l+1} + d_{k+1,l} - d_{k,l} - d_{k+1,l+1}.$$

Since D is a Monge matrix the area of each unit square is nonnegative. Moreover, we associate an area with the region $A(\lambda)$ of a pyramidal subtour λ. $A(\lambda)$ decomposes into a number of unit squares, and the area $s(A(\lambda))$ is defined as the sum of the areas of all these unit squares. Clearly,

$$A(\lambda) \subseteq A(\mu) \quad \Longrightarrow \quad s(A(\lambda)) \leq s(A(\mu)). \tag{17}$$

The area $s(A(\lambda))$ can be regarded as an algebraic sum of elements d_{ij} where (i,j) are the points enclosed by $A(\lambda)$. It is easy to see that it is not necessary to consider *all* such elements for computing the area. If (i,j) is an inner point of the area (i.e. the vertex (i,j) belongs to four unit squares in $A(\lambda)$), then the algebraic sum of the four corresponding elements d_{ij} equals zero. If a vertex (i,j) lies on the boundary of $A(\lambda)$ and belongs to exactly two unit squares, then the overall contribution of the vertex to the area amounts to zero, too. This implies that only corner points of the region $A(\lambda)$ contribute non-zero values to the sum $s(A(\lambda))$. Furthermore, it is easy to see that for corner points (i_k, i_k) that lie on the main diagonal, the corresponding elements $d_{i_k i_k}$ add up with sign '$-$', and the elements $d_{i_k, i_{k-1}}$ corresponding to the remaining corner points add up with sign '$+$'. Finally, note that the sum of the elements $d_{i_k, i_{k-1}}$ for non-diagonal corner points equals exactly the cost of subtour λ. Summarizing, we have the following result.

Lemma 4.5 *Let $D = (d_{ij})$ be a Monge matrix, and let λ be a permutation with exactly one non-trivial factor, which is pyramidal. Then*

$$d(\lambda) - d(\varepsilon) = \sum_{i=1}^{n} d_{i,\lambda(i)} - \sum_{i=1}^{n} d_{ii} = s(A(\lambda)).$$

□

Now let λ consist of k pyramidal factors $\lambda_1, \lambda_2, \ldots, \lambda_k$. Define the region $A(\lambda)$ as $A(\lambda) = \cup_{i=1}^{k} A(\lambda_i)$ and the area of the region as $s(A(\lambda)) = \sum_{i=1}^{k} s(A(\lambda_k))$. Lemma 4.5 implies that $s(A(\lambda)) = d(\lambda) - d(\varepsilon)$ also holds for this case. This leads to the following corollary.

Corollary 4.6 *Let λ and μ be two permutations that only consist of pyramidal factors and assume $A(\lambda) \subseteq A(\mu)$. Then $d(\lambda) \leq d(\mu)$.* □

We are now ready to prove Theorem 4.3.

Proof of Theorem 4.3. Let τ be an optimal tour. We will construct a tree permutation α such that $c(\phi \circ \alpha) \leq c(\tau)$. Let β be the (unique) permutation such that $\tau = \phi \circ \beta$. Furthermore, let $\beta_1, \ldots, \beta_b$ be the factors of β and $i_{l1}, i_{l2}, \ldots, i_{l,h(l)}$ with $i_{l1} < i_{l2} < \cdots < i_{l,h(l)}$ the cities in the factor β_l, $1 \leq l \leq b$. Let $D = C_{\varepsilon,\phi}$. Then $c(\tau) = c(\phi \circ \beta) = d(\beta)$ and $c(\phi \circ \alpha) = d(\alpha)$. Note that to find a tree permutation α such that $c(\phi \circ \alpha) \leq c(\tau)$ is equivalent to finding a tree permutation α such that $d(\alpha) \leq d(\beta)$.

Observe that without loss of generality we may assume that the factors of β are pyramidal. This follows from the fact that any submatrix of a Monge matrix is also a Monge matrix. Thus, the submatrices corresponding to the cities in each of the factors β_k all fulfill the Monge property. Since by Theorem 2.2 there is an optimal tour that is pyramidal, we may assume that all factors of β are pyramidal.

In order to obtain α, construct a multi-graph $G_{\phi,\beta}$ defined as follows ($G_{\phi,\beta}$ will be a spanning subgraph of G_ϕ). The edge $(i, i+1)$ in G_ϕ corresponding to the transposition $\langle i, i+1\rangle$ is also an edge in $G_{\phi,\beta}$ if there is a factor β_l of β such that $i_{l1} \leq i < i+1 \leq i_{l,h(l)}$.

We claim that $G_{\phi,\beta}$ is connected. Assume the contrary holds. Then the set of vertices of $G_{\phi,\beta}$ can be split into two disjoint nonempty sets V_1 and V_2 such that there is no edge between a vertex in V_1 and vertex in V_2. But this means that there is no factor of β that contains both an index in the set of cities associated with vertex set V_1 as well as an index in the set of cities associated with vertex set V_2. In other words, the two disjoint sets of cities are not patched by the permutation β which contradicts the fact that τ is a cyclic permutation.

Since $G_{\phi,\beta}$ is connected, it contains a spanning tree. Let T be a spanning tree in $G_{\phi,\beta}$. Clearly, T is also a spanning tree in G_ϕ and therefore, any α obtained from multiplying the transpositions associated with the edges of T will be a tree permutation. Let $\alpha'_1, \ldots, \alpha'_a$ be the factors of α and $j_r, j_r+1, \ldots, j_r+t_r$ be the cities in the factor α'_r $(r = 1, \ldots, a)$. From the construction of $G_{\phi,\beta}$ it follows that for all factors α'_r there is a factor β_l such that $i_{l1} \leq j_r < \cdots < j_r + t_r \leq i_{l,h(l)}$. Thus, by Lemma 4.4, we can transform factors α'_r into pyramidal factors α_r such that for all factors α_r there is a factor β_l containing α_r. Let α be a new permutation with pyramidal factors $\alpha_1, \ldots, \alpha_a$. Note that several factors of α can be contained in one factor of β, that a factor of α can be contained in several factors of β and that it is possible that a factor of β does not contain any factor of α. It is possible to assign each factor of α to one factor of β in which it is contained. This yields $A(\alpha) \subseteq A(\beta)$, and therefore $d(\beta) \geq d(\alpha)$. This completes the proof of the theorem. □

Obviously, Theorem 4.3 is a very useful result in the sense that it tells us that we may restrict ourselves to tree permutations consisting of adjacent transpositions instead of considering all possible patching permutations.

As an illustrative example, consider an $n \times n$ product matrix C, where n is even and for $1 \leq i, j \leq n$, $c_{ij} = a_i b_j$ with $a_1 \geq a_2 \geq \cdots \geq a_n$ and

$$b_2 < b_1 < b_4 < b_3 < \cdots < b_{2i} < b_{2i-1} < \cdots < b_n < b_{n-1}.$$

Since every product matrix is a permuted Monge matrix, the above theory applies. For this special case, an optimal assignment is given by $\phi = \langle 1,2\rangle\langle 3,4\rangle \cdots \langle n-1, n\rangle$. Observe that the patching graph G_ϕ is a path, i.e. there is only a single tree. This tree contains the edges $(2,3), (4,5), \ldots, (n-2, n-1)$. Since there is only one tree in G_ϕ, it follows that an optimal tour $\tau^\star$ is given by

$$\tau^\star = \phi \circ \langle 2,3\rangle \circ \langle 4,5\rangle \circ \cdots \circ \langle n-2, n-1\rangle = \langle 1, 2, \ldots, n-2, n, n-1, \ldots, 3, 1\rangle.$$

4.1.2 Spanning trees for the patching graph

In the preceding section, we essentially reduced the TSP on permuted Monge matrices to the problem of finding an "optimal" spanning tree in a patching graph G_ϕ. In this section, we investigate and clarify the notion of what "optimal" actually means in this respect.

Let T be a spanning tree for G_ϕ. We divide the set of edges (i.e. transpositions) of the tree T into a number of t dense, pairwise disjoint *branches* defined as follows:

$$\begin{aligned}
I(i_1, l_1) &= \{(i_1, i_1+1), (i_1+1, i_1+2), \quad \ldots, (i_1+l_1-1, i_1+l_1)\},\\
I(i_2, l_2) &= \{(i_2, i_2+1), (i_2+1, i_2+2), \quad \ldots, (i_2+l_2-1, i_2+l_2)\},\\
&\qquad\qquad \cdots\\
I(i_t, l_t) &= \{(i_t, i_t+1), (i_t+1, i_t+2), \quad \ldots, (i_t+l_t-1, i_t+l_t)\}
\end{aligned}$$

such that $T = I(i_1, l_1) \cup I(i_2, l_2) \cup \cdots \cup I(i_t, l_t)$ and $i_k + l_k < i_{k+1}$ for $1 \leq k \leq t-1$.

Clearly, by multiplying the transpositions of T in arbitrary order we obtain t subcycles. Each subcycle is a pyramidal subtour and corresponds to a set $I(i_k, l_k)$, $1 \leq k \leq t$. It is well-known and easy to verify (see e.g. [57], Theorem 14) that

$$c(\phi \circ \alpha) - c(\phi) = \sum_{i=1}^{t} (c(\phi \circ \alpha_i) - c(\phi)) \tag{18}$$

holds for any permutation α consisting of t disjoint cycles $\alpha_1, \alpha_2, \ldots, \alpha_t$. Now for a given tree T, we define a patching permutation α_T corresponding to T as a permutation that minimizes (18). Such a permutation can be constructed by finding an optimal pyramidal subtour for each dense branch $I(i,l)$.

Define the *branch weight* $w(I(i,l))$ of the branch $I(i,l)$ by

$$w_{i,i+l-1} = w(I(i,l)) = c(\phi \circ \alpha_i^*) - c(\phi),$$

where α_i^* is an optimal pyramidal subtour corresponding to the branch $I(i,l)$. In particular,

$$w_{ii} = c(\phi \circ \langle i, i+1 \rangle) - c(\phi) = c_{i\phi(i+1)} + c_{i+1\phi(i)} - c_{i\phi(i)} + c_{i+1\phi(i+1)}$$

holds. Define the branch weight (*b-weight*) $w(T)$ of a tree T as the sum of the branch weights of all tree branches. It follows from the definitions of α_T and $w(T)$ that $c(\phi \circ \alpha_T) = c(\phi) + w(T)$.

Proposition 4.7 *A patching permutation α_T is an optimal patching permutation if and only if a corresponding tree T has a minimal b-weight $w(T)$.* □

By applying Park's algorithm [87] for finding an optimal pyramidal tour, all the weights w_{ij} for $i = 1, \ldots, n-1$ and $j = i, \ldots, n-1$ can be found in $O(n^2)$ overall time. This time complexity follows from the fact that in Park's algorithm, by computing the value of an optimal tour on the set $\{1, \ldots, n\}$ in $O(n)$ time, we can compute at the same time the values of the optimal tours for all sets $\{i, \ldots, n\}$ for $i = 1, \ldots, n-1$.

The weights (w_{ij}) can also be interpreted as areas of regions as defined above. This interpretation clarifies some useful properties of b-weights as shown in the following lemma.

Lemma 4.8 *If $C_{\varepsilon,\phi}$ a Monge matrix, then*

(i) $w_{ij} = s(A(\lambda_{ij}))$, *where λ_{ij} is an optimal pyramidal tour on the set $\{i, i+1, \ldots, j+1\}$ for $1 \leq i \leq n-1$ and $i \leq j \leq n-1$;*

(ii) $w_{ik} + w_{k+1,j} \leq w_{ij}$ *for* $1 \leq i \leq n-1$ *and* $i \leq k < j \leq n-1$.

(iii) $w_{ij} \geq \sum_{k=i}^{j} w_{kk}$ *for* $1 \leq i \leq j \leq n$.

Proof. (i) follows immediately from the definition of the area of regions and of w_{ij}.
(ii) Let $\lambda_{i,j}$ be an optimal pyramidal tour on the set $\{i, i+1, \ldots, k, k+1, \ldots, j+1\}$. Furthermore, let $\mu_{i,k}$ and $\mu_{k+1,j}$ be reductions of $\lambda_{i,j}$ to the sets $\{i, i+1, \ldots, k+1\}$ and $\{k+1, k+2, \ldots, j, j+1\}$, respectively. From the procedure for constructing regions we obtain

$$w_{ij} = s(A(\lambda_{ij})) \geq s(A(\mu_{ik})) + s(A(\mu_{k+1,j})) \geq w_{ik} + w_{k+1,j}.$$

(iii) This is an immediate consequence of (ii). □

Summarizing, the TSP restricted to permuted Monge matrices reduces to the following *minimum spanning tree problem with branches* (which we will denote by B-MST).

Let $G = (V, E)$ be a Eulerian multigraph with $E = \{1, 2, \ldots, |E|\}$ where the edge sequence $1, \ldots, |E|$ forms a Eulerian path. Let w be a weight function that assigns to every interval $[i, \ldots, j]$, $1 \leq i \leq j \leq |E|$, a nonnegative integer w_{ij} and let w fulfill the property

$$w_{ij} \geq w_{ik} + w_{k+1,j} \qquad \text{for all } i \leq k < j. \tag{19}$$

An interval $[i, \ldots, j]$ is called a *branch* with respect to some set $E' \subseteq E$ if it is a maximal subinterval of E' (i.e. $[i, \ldots, j] \subseteq E'$, $i-1 \notin E'$ and $j+1 \notin E'$). For $E' \subseteq E$, let $Br(E')$ denote the set of all branches with respect to E'. The *branch weight* (or *b-weight*) of $E' \subseteq E$ is defined by

$$w(E') = \sum_{[i,\ldots,j] \in Br(E')} w_{ij}.$$

The *branch weight (b-weight) of a spanning tree* $T = (V, E_T)$ of G is $w(E_T)$. The problem B-MST consists in finding a spanning tree of G with minimum b-weight.

This problem is investigated in detail in Section 4.2.

4.1.3 Generalizations of the patching scheme

A larger class of matrices. As mentioned earlier, the idea of using geometrical arguments in the proof of Theorem 4.3 is originally due to Burdyuk and Trofimov [13]. However, the proof presented above differs from [13] in several aspects. For example, it allows to generalize the statement of the theorem to a more general class of matrices than Monge matrices. Observe that by the way we constructed the regions, only rectangles with a very restricted structure can be removed from the region. These rectangles are: $[i, i]^0$ for $1 \leq i \leq n-1$ and the rectangles with corner points (j, j), (i, j), (i, k), (j, k) respectively (j, j), (k, j), (k, i), (j, i) for $i < j < k$. Hence, the implication (17) is valid if the following set of inequalities is fulfilled.

$$d_{i,i+1} + d_{i+1,i} + d_{ii} + d_{i+1,i+1} \geq 0 \qquad \text{for } 1 \leq i \leq n-1 \tag{20}$$

$$d_{kj} + d_{ji} - d_{jj} - d_{ki} \geq 0 \qquad \text{for } 1 \leq i < j < k \leq n \tag{21}$$

$$d_{ij} + d_{jk} - d_{ik} - d_{jj} \geq 0 \qquad \text{for } 1 \leq i < j < k \leq n \tag{22}$$

It is easy to see that a Monge matrix $D = (d_{ij})$ satisfies conditions (20)–(22). However, (20)–(22) describe a larger class of matrices, e.g. observe that the matrix

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

is not a Monge matrix, but fulfills (20)–(22).

It can be shown that if a matrix D fulfills properties (20)–(22), then it belongs to the class of Demidenko matrices as well as to the class of so-called weak Monge matrices. A matrix $D = (d_{ij})$ is a *weak Monge matrix* (see [17]) if it satisfies the property $d_{ii} + d_{jl} \leq d_{il} + d_{ji}$ for all $1 \leq i \leq j, l \leq n$. Derigs, Goecke and Schrader [43] observed that for weak Monge matrices (exactly as for standard Monge matrices) the identity permutation constitutes an optimal solution for the linear assignment problem.

Since for the TSP on a Demidenko matrix, an optimal tour can be found as an optimal pyramidal tour (see Theorem 2.8), Theorem 4.3 carries over to matrices C such that the matrix $D = C_{\epsilon,\phi}$ has

the properties (20)–(22). The problem of recognizing such matrices is still open. A first step in this direction has been done by Čechlárová and Szabó [21] who presented a polynomial time algorithm for recognizing permuted weak Monge matrices.

The TSP with p traveling salesmen. In the *TSP with p traveling salesmen* (pTSP), the goal is to find for a given $n \times n$ distance matrix $C = (c_{ij})$, a permutation π of the set $\{1, 2, \ldots, n\}$ with at most p factors that minimizes the function

$$c(\pi) = \sum_{i=1}^{n} c_{i\pi(i)}.$$

Hence, p salesmen must visit the cities 1 to n in arbitrary order and they want to minimize their total travel length. It was observed by Burdyuk and Deĭneko [12] that Theorem 4.3 can be reformulated and applied to the pTSP, too. It can be shown that if an optimal assignment ϕ has more than p cycles, then for every permutation τ with no more than p factors, there exists a forest F with p components in G_ϕ and a permutation α that results from multiplying the transpositions corresponding to F, such that $\phi \circ \alpha$ has p cycles and such that $c(\phi \circ \alpha) \leq c(\tau)$. Hence, the pTSP reduces to the problem of finding a forest $F^\star$ on p components with minimum sum of branch weights.

Related results. Another way to generalize the Gilmore-Gomory subtour-patching scheme is to consider also non-adjacent transpositions (see [57], pp. 109–113). Much remains to be done, since so far there is just a small number of results in this direction.

Lawler's special TSP case [68] on *right-upper-triangular* matrices can be interpreted via the subtour-patching scheme. A matrix $C = (c_{ij})$ is right-upper-triangular if $c_{ij} = 0$ for all $j \geq i$. In this special case, an optimal tour τ_{opt} can be constructed from an optimal assignment ϕ such that $c(\tau_{opt}) = c(\phi)$ holds.

Burkard and Van der Veen [19] consider the TSP on a symmetric nonnegative *left-upper-triangular* matrix (i.e. a symmetric matrix $C = (c_{ij})$ where $c_{ij} = 0$ for all i and j with $i+j \geq n+1$). An optimal tour in this case is constructed from an optimal assignment in a related matrix. Again, given an optimal assignment, an optimal tour can be constructed without increasing the cost.

Another result in this direction is presented by Kollias, Manolopoulos and Papadimitriou [65]. They investigate a TSP on a permuted right-upper-triangular Monge matrix with an additional column and show how to solve it in $O(n \log n)$ time. Note that the additional column usually destroys the Monge structure of the distance matrix.

4.2 Specially structured patching graphs

In 1980, Sarvanov [108] proved that the TSP on product matrices is $\mathcal{NP}$-hard (cf. Theorem 5.2 and Section 5.2). Recall that a product matrix C is of the form $c_{ij} = a_i b_j$ with real numbers a_i and b_j, $1 \leq i, j \leq n$. Since the product matrices form a subclass of the permuted Monge matrices, Sarvanov's result implies that the TSP on permuted Monge matrices is $\mathcal{NP}$-hard in general. In Section 4.1.2 the TSP restricted to permuted Monge matrices was essentially reduced to problem B-MST. So, Sarvanov's result implies that problem B-MST is in general $\mathcal{NP}$-hard. Burkard, Deĭneko and Woeginger [16] proved this more directly and obtained an even stronger results. They showed that the problem B-MST is $\mathcal{NP}$-hard even for *planar* Eulerian multigraphs (the transformation is from the $\mathcal{NP}$-hard Hamiltonian path problem in planar bipartite digraphs with degree bound two, cf. Plesnik [90]).

This section deals with well-solved special cases of the TSP arising from certain conditions on the patching graph G_ϕ. A graph is called a *multi*path (-cycle, -star, -tree, respectively) if the underlying

simple graph is a path (cycle, star, tree). It turns out that if the patching graph has a sufficiently simple structure (multipath, multicycle, multistar, multitree), then the problem B-MST of finding a spanning tree of G_ϕ with minimum b-weight is solvable in polynomial time. By the discussion in Section 4.1.2, this is equivalent to solving the TSP for the corresponding permuted Monge matrix.

4.2.1 Multipaths and multicycles

A matrix $C = (c_{ij})$ is called an *anti-Monge* matrix if $c_{ij} + c_{kl} \geq c_{i,l} + c_{jk}$ for all $1 \leq i < k \leq n$ and $1 \leq j < l \leq n$. Observe that C is anti-Monge if and only if $-C$ is Monge.

It is easy to see that for an anti-Monge matrix C, an optimal assignment is given by the permutation ϕ^*, defined by $\phi^*(i) = n + 1 - i$ for $1 \leq i \leq n$. The corresponding patching graph G_{ϕ^*} is a multipath of special structure, i.e. there are $m = \lfloor (n+1)/2 \rfloor$ vertices corresponding to the m factors of ϕ^*. Let $V = \{1, 2 \ldots, m\}$. Then each pair of vertices k and $k+1$ with $1 \leq k \leq m-1$ is connected by two edges which correspond to the two transpositions $\langle k, k+1 \rangle$ and $\langle n-k, n-k+1 \rangle$.

Recall that the weights of the edges in the patching graph are defined by

$$\begin{aligned} w_{kk} &= c_{k,\phi^*(k+1)} + c_{k+1,\phi^*(k)} - c_{k,\phi^*(k)} - c_{k+1,\phi^*(k+1)} \\ &= c_{k,n-k} + c_{k+1,n+1-k} - c_{k,n+1-k} - c_{k+1,n-k} \qquad \text{and} \\ w_{n-k,n-k} &= c_{n-k,k} + c_{n-k+1,k+1} - c_{n-k,k+1} - c_{n-k+1,k}. \end{aligned}$$

Now consider the special case of the TSP on symmetric anti-Monge matrices. It is easy to see that in this case symmetry yields $w_{k,k} = w_{n-k,n-k}$ for $1 \leq k \leq m-1$. Since by (19) the branch weights are super-additive, there are only two potential candidates for a spanning tree with minimum branch weight namely

$$\begin{aligned} T_1 &= \{\langle 1,2 \rangle, \langle n-2, n-1 \rangle, \langle 3,4 \rangle, \langle n-4, n-3 \rangle, \ldots\} \text{ and} \\ T_2 &= \{\langle n-1, n \rangle, \langle 2,3 \rangle, \langle n-3, n-2 \rangle, \langle 4,5 \rangle, \ldots\}. \end{aligned}$$

The permutations $\tau_1 = \langle 1, n-1, 3, n-3, \ldots, n-2, 2, n \rangle$ and $\tau_2 = \langle 1, n, 2, n-2, 4, \ldots, n-3, 3, n-1 \rangle$ that correspond to T_1 and T_2 respectively, both yield optimal tours for the symmetric anti-Monge TSP. Clearly, τ_1 and τ_2 describe the same tour, only in two directions.

This result has already been proved in 1957 by Supnick [109], but by a different reasoning. Interestingly, the result was rediscovered twice, once in 1971 by Rubinshtein [103] and once in 1987 by Michalski [74].

Aizenshtat and Kravchuk [5, 6] considered the special case of the TSP on symmetric product matrices C (see Theorem 5.3(ii)). Aizenshtat and Maksimovich [7] generalized their approach to the general (not necessarily symmetric) anti-Monge TSP. These papers did not use the subtour patching technique, but a quite complicated characterization of the structure of an optimal tour. Based on this characterization, a dynamic programming algorithm was developed for computing the shortest tour. The underlying set of permutations contains exactly those permutations that can be obtained by postmultiplying ϕ^* with transpositions from T_1 and T_2.

In Aizenshtat [3], Demidenko and Metelsky [42] and Kunzevich [66, 67], various special cases of the TSP with permuted product matrices were considered. Using the terminology presented here, all these cases can be characterized as cases where the patching graph is a multipath with an additionally restricted special structure. Sarvanov [107] and Deĭneko [31] were the first who used a patching graph in order to characterize solvable cases of the TSP. They derived algorithms with $O(n^3)$ time complexity for the case where the patching graph is an arbitrary multipath (see Figure 7 for an illustration). In these papers the problem of finding a spanning tree with minimum

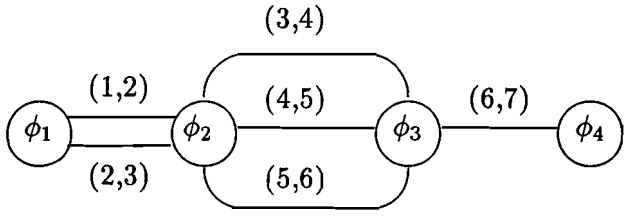


Figure 7: For $\phi = \phi_1\phi_2\phi_3\phi_4 = \langle 2\rangle\langle 1,3,5\rangle\langle 4,6\rangle\langle 7\rangle$, the graph G_ϕ is a multipath.

branch weight is reduced to the problem of finding a specially structured minimum length path in a related network. Two years later, Gaikov [54] reduced the time complexity of Sarvanov's algorithm down to $O(mn)$ where m is the number of factors in an optimal assignment and n is the number of cities. Finally, Burkard and Deĭneko [15] derived a linear time algorithm for this special case of the permuted Monge TSP.

Theorem 4.9 (Burkard and Deĭneko [15])
Let $C_{\varepsilon,\phi}$ be a Monge matrix. Assume that the permutation ϕ is given a priori and let ϕ consist of m factors. Then the following statements hold.

(i) If the patching graph G_ϕ is a multipath, then the TSP with distance matrix C can be solved in $O(n)$ time.

(ii) If the patching graph G_ϕ is a multicycle, then the TSP with distance matrix C can be solved in $O(nm)$ time. □

4.2.2 Multistars and multitrees

Now we return to the problem of finding a spanning tree with minimum branch weight (problem B-MST introduced in Section 4.1.2). In this section, we will consider the most general known solvable case of this problem which includes the case of a multipath (see Theorem 4.9(ii)) as special case, namely the case where the patching graph G_ϕ is a multitree. The results in this section are based on Gaikov's paper [54]. We start with an auxiliary problem.

Gaikov's Covering Problem. Let $H = (N, A)$ be a directed multigraph. Some of the arcs in A are grouped into so-called twin-pairs. A *twin-pair* consists of two arcs a_i and a_j where $a_i = (v_s, v_t)$ and $a_j = (v_t, v_s)$ holds for some $v_s, v_t \in N$, $v_s \neq v_t$. Obviously, every arc belongs to at most one twin-pair. Arcs that are not part of any twin pair are called *single arcs*. For $B \subseteq A$, let $Twin(B)$ denote the set of all twin-pairs contained in B, let $Sing(B)$ denote the set of all non-loop single arcs in B (these arcs may have a twin-brother in A but not in B), and let $Loop(B)$ denote the set of all loops in B.

With every arc a in A, we associate a non-negative weight $g(a)$ and for every twin-pair $(a_i, a_j) \in Twin(A)$, there is a non-negative weight $g(a_i, a_j)$. The *Gaikov weight* (or *g-weight*) of a subset $B \subseteq A$ is defined by

$$g(B) = \left(\sum_{(a_i,a_j)\in Twin(B)} g(a_i, a_j) \right) + \left(\sum_{a_k \in Sing(B)} g(a_k) \right) + \left(\sum_{a_l \in Loop(B)} g(a_l) \right).$$

We say that every arc $(v_s, v_t) \in A$ *covers* its target-vertex v_t. A set $B \subseteq A$ is an *exact cover* for the directed graph H, if every vertex in N is covered by exactly one arc in B. It is easy to see that H possesses an exact cover if and only if each vertex in N has in-degree at least one.

By formulating the problem as a weighted matching problem in a related undirected graph and by applying the weighted matching algorithm of Gabov [52], one derives the following result.

Lemma 4.10 *Let $H = (N, A)$ be a directed multigraph where every vertex has in-degree at least one, and let g be a weight function on A and $Twin(A)$. Then an exact cover B^* for H with minimum Gaikov weight can be computed in $O(|A|^2 \log |A|)$ time.* □

Gaikov considered graphs with a special weight function, namely, with a function that has the following property

$$g(a_i) + g(a_j) \leq g(a_i, a_j) \qquad \text{for all } (a_i, a_j) \in Twin(A). \tag{23}$$

In this case the time complexity stated in the previous lemma can be reduced.

Theorem 4.11 (Gaikov [54])
Let $H = (N, A)$ be a directed graph where every vertex has in-degree at least one, and let g be a weight function on A and $Twin(A)$ that fulfills condition (23). Then an exact cover B^* for H with minimum Gaikov weight can be computed in $O(|N|^2 + |A|)$ time. □

A proof of this theorem can also be found in Burkard, Deĭneko and Woeginger [16].

Gaikov's Algorithm for Multistars. We now consider the problem of finding a spanning tree with minimum branch weight on a *multistar*. For example, the patching graph G_ϕ for the permutation $\phi = \langle 1,3,5,7\rangle\langle 2\rangle\langle 4\rangle\langle 6\rangle\langle 8\rangle$ is a multistar whose central vertex corresponds to the cycle $\langle 1,3,5,7\rangle$, see Figure 8.

Gaikov's main idea is to translate the problem of computing a spanning tree with minimum b-weight into an equivalent problem of computing an exact cover with minimum g-weight in a related directed multigraph.

Theorem 4.12 *Let $G = (V, E)$ be a Eulerian multistar with $E = \{1, 2, \ldots, |E|\}$ such that the edge sequence $1, \ldots, |E|$ forms a Eulerian path. Let w be a weight function on the intervals $[i, \ldots, j]$, $1 \leq i \leq j \leq |E|$, that fulfills property (19). Then a spanning tree for G with minimum branch weight can be computed in $O(|V|^2 + |E|)$ time.*

Proof. Let $v_0 \in V$ denote the central vertex of the multistar (the vertex that is incident to all edges), and let $v_1, \ldots, v_m$ be an enumeration of the other vertices in V. It is easy to see that for multistars, every branch of a spanning tree contains at most two edges. We call two consecutive edges $i = [v_0, v_s]$ and $i+1 = [v_0, v_t]$ in E a *critical pair*, if $v_s \neq v_t$ holds (and hence, they are potential candidates for forming a branch in some spanning tree). Since $1, \ldots, n$ forms a Eulerian path, every edge participates in at most one critical pair. Edges that are not part of any critical pair are called *non-critical edges*.

The construction of the related directed multigraph $H = (V, A)$ is done as follows. The vertex set N of H equals $\{v_1, \ldots, v_m\}$. For every non-critical edge $i = [v_0, v_t]$ in E, we introduce a loop $\ell = (v_t, v_t)$ in A that is incident to vertex v_t and has weight $g(\ell) = w_{tt}$. For every critical pair of edges $i = [v_0, v_s]$ and $i+1 = [v_0, v_t]$ in E, we introduce a twin-pair a_i and a_{i+1} of arcs in A by $a_i = (v_t, v_s)$ with weight $g(a_i) = w_{ii}$ and $a_{i+1} = (v_s, v_t)$ with weight $g(a_{i+1}) = w_{i+1,i+1}$. The weight $g(a_i, a_{i+1})$ is set to $w_{i,i+1}$.

It is easy to see that G has a spanning tree with branch weight W if and only if H has an exact cover with Gaikov weight W. There is a one-to-one correspondence between edges in G and arcs in H such that an edge connects some vertex v_t to v_0 in G if and only if the corresponding arc covers vertex v_t in H. Applying Theorem 4.11 completes the proof. □

As an example, consider the distance matrix (c_{ij}) with $c_{ij} = x_i y_j$ for $i, j = 1, \ldots, 8$, where $x = (1,2,3,4,5,6,7,8)$ and $y = (1,8,10,6,7,2,4,0)$. An optimal assignment ϕ with cost $c(\phi) = 110$ can be found by sorting the vector y, i.e. $\phi = \langle 1,3,5,7\rangle\langle 2\rangle\langle 4\rangle\langle 6\rangle\langle 8\rangle$. The graph H is computed according to Theorem 4.12 (see Figure 8). The weights of the arcs and twin-arcs are defined as follows:

$$
\begin{aligned}
&g(a_1) := w(1,2) = w_{11} = 2,\\
&g(a_2) := w(2,3) = w_{22} = 1, \quad g(a_3) := w(3,4) = w_{33} = 1, \quad g(a_2, a_3) := w_{23} = 3,\\
&g(a_4) := w(3,4) = w_{33} = 2, \quad g(a_5) := w(4,5) = w_{44} = 2, \quad g(a_4, a_5) := w_{45} = 6,\\
&g(a_6) := w(6,7) = w_{66} = 1, \quad g(a_7) := w(7,8) = w_{77} = 1, \quad g(a_6, a_7) := w_{67} = 3.
\end{aligned}
$$

An exact cover B^* with minimum Gaikov weight for the graph H consists of arcs a_1, a_3, a_5 and a_7 and its weight equals 6. This yields a patching permutation $\alpha = \langle 1,2\rangle \circ \langle 3,4\rangle \circ \langle 5,6\rangle \circ \langle 7,8\rangle$. Therefore, an optimal tour τ can be found by multiplying ϕ and α. Hence, $\tau = \langle 1,2,3,4,5,6,7,8\rangle$ and $c(\tau) = 116$.

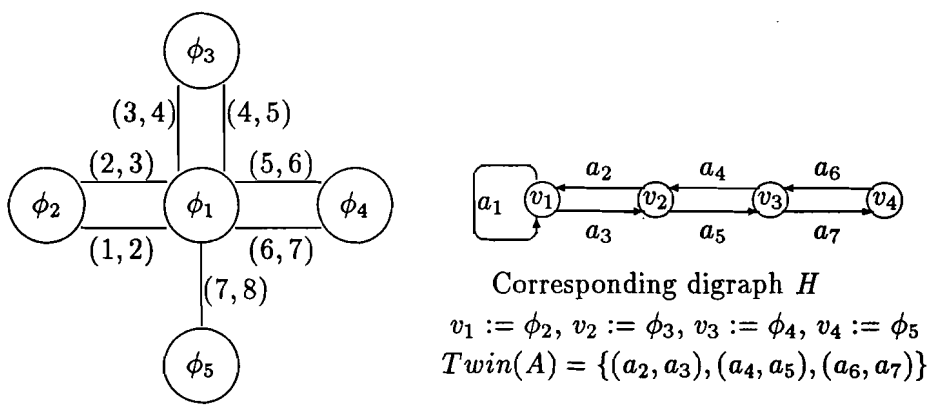


Figure 8: Graph G_ϕ and digraph H for $\phi = \langle 1,3,5,7\rangle\langle 2\rangle\langle 4\rangle\langle 6\rangle\langle 8\rangle$

A Fast Algorithm for Multitrees. Gaikov [54] used the algorithm for multistars as a subroutine for constructing a dynamic programming algorithm for the case where the patching graph is a multitree. The time complexity of Gaikov's algorithm is $O(nm^5)$, where m is the number of factors in an optimal assignment. By devising an improved dynamic programming scheme and by carrying out a very careful time analysis, Burkard, Deĭneko and Woeginger [16] derived an algorithm with time complexity $O(nm + m^3)$.

Theorem 4.13 (Burkard, Deĭneko and Woeginger [16])
Let $G = (V, E)$ be a Eulerian multitree with branch weights that fulfill condition (19). Then a spanning tree for G with minimum branch weight can be computed in $O(|V|^3 + |V||E|)$ time. □

5 The permuted Monge TSP: Applications

We already mentioned that by Sarvanov's result [108] the TSP on permuted Monge matrices is in general $\mathcal{NP}$-hard. In this section we will report several positive results on the permuted Monge TSP. We describe the heuristic *"Subtour-patch"* that is build on the ideas developed in Section 4 and performs quite well on permuted Monge matrices. Moreover, we deal with special subclasses of permuted Monge matrices for which the TSP can be solved in polynomial time.

5.1 The Heuristic "Subtour-patch"

The heuristic to be presented in this section is essentially based on the theory of subtour patching and on the property of the weights (w_{ij}) which is formulated in Lemma 4.8(iii).

We define the *weight* $p(i)$ of an edge $(i, i+1)$ in G_ϕ by

$$p(i) = w_{ii} = c_{i\phi(i+1)} + c_{i+1\phi(i)} - c_{i\phi(i)} - c_{i+1\phi(i+1)}.$$

Let T be a spanning tree of G_ϕ. The *weight* of T is the sum of the weights $p(i)$ of its edges and is denoted by $P(T)$. Now let $T_{\min}$ be a minimum spanning tree in G_ϕ with $P(T_{\min}) = \min_T P(T)$. By Lemma 4.8(iii), for every spanning tree T its branch weight $w(T)$ fulfills $w(T) \geq P(T_{\min})$. Let $\tau_{patch} := \phi \circ \alpha_{T_{\min}}$ be a tour corresponding to $T_{\min}$ that is optimal in the sense that it results from finding an optimal pyramidal subtour for every branch in $T_{\min}$. In other words, τ_{patch} is the heuristic tour determined by the following procedure.

Heuristic "Subtour-patch"
Input: A permuted Monge matrix.
Output: A cyclic permutation τ.
Step 1: Determine an optimal assignment ϕ.
If ϕ is cyclic then $\tau := \phi$ is an optimal tour. Stop.
Otherwise continue with Step 2.
Step 2: Construct the weighted assignment graph G_ϕ.
Step 3: Determine a minimal spanning tree $T_{\min}$ in G_ϕ.
Step 4: Find from the transpositions corresponding to the edges in $T_{\min}$ the optimal patching permutation $\alpha_{T_{\min}}$ by finding an optimal pyramidal tour for every branch in $T_{\min}$.
Step 5: Construct the tour $\tau := \tau_{patch} = \phi \circ \alpha_{T_{\min}}$.

Let τ_{opt} be an optimal tour. From the earlier observations we have

$$c(\phi) + P(T_{\min}) \leq c(\tau_{opt}) \leq c(\tau_{patch}) = c(\phi) + w(T_{\min}).$$

Note that if $w(T_{\min}) = c(\tau_{patch}) - c(\phi) = P(T_{\min})$ holds, then τ_{patch} is an optimal tour. Otherwise, τ_{patch} is a suboptimal permutation that fulfills $c(\tau_{patch}) \leq (1+\varepsilon)c(\tau_{opt})$ with

$$\varepsilon = \frac{w(T_{\min}) - P(T_{\min})}{c(\phi) + P(T_{\min})}.$$

Heuristic Subtour-patch has some nice properties which will be discussed in the following sections. For example, it will be shown that this heuristic solves the TSP restricted to several subclasses of the permuted Monge matrices to optimality. However, we first demonstrate a 'weakness' of the heuristic.

Proposition 5.1 *Heuristic Subtour-patch does not have a finite worst case ratio.*

Proof. Consider the following example. For $x \geq 0$, let $D = C_{\varepsilon,\phi}$ be the following Monge matrix

$$\begin{pmatrix} 0 & 1 & 1 & x+1 \\ 1 & 0 & 0 & x \\ 1 & 0 & 0 & 0 \\ x+1 & x & 0 & 0 \end{pmatrix}$$

with optimal assignment $\phi = \langle 1,3\rangle\langle 2\rangle\langle 4\rangle$. The patching graph G_ϕ contains three edges $(1,2)$, $(2,3)$ and $(3,4)$ with weights $p(1,2) = 2$, $p(2,3) = 0$ and $p(3,4) = 0$, respectively. There are only two spanning trees: $T_1 = \{(1,2),(3,4)\}$ with weight $P(T_1) = 2$ and $T_2 = \{(2,3),(3,4)\}$ with weight $P(T_2) = 0$. Heuristic Subtour-patch takes the tree T_2 and constructs a solution $\tau_{patch} = \phi \circ \langle 2,3\rangle \circ \langle 2,4\rangle$ with cost $c(\tau_{patch}) = x$, whereas tree T_1 leads to a tour τ with $c(\tau) = 2$. Since x can become arbitrarily large, the argument is complete. □

5.2 The TSP on product matrices

We start with Sarvanov's classical result on the product TSP.

Theorem 5.2 (Sarvanov [108])
The TSP on product matrices is $\mathcal{NP}$-hard. □

Apparently, the first paper dealing with the TSP restricted to product matrices as a special TSP case is the paper by Suprunenko [110] from 1968 (see Theorem 5.3(i) below). This paper inspired a sequence of results by Belorussian mathematicians, many of them mentioned in this survey.

Despite the $\mathcal{NP}$-hardness of the general case, the TSP restricted to *symmetric* product matrices is easily solvable. If $C = (c_{ij})$ is a symmetric product matrix then $c_{ij} = a_i b_j = a_j b_i = c_{ji}$. Hence $a_i/b_i = a_j/b_j$ holds for $1 \leq i,j \leq n$. We define $\lambda = a_1/b_1 = \cdots = a_n/b_n$. It follows that $b_i = \lambda a_i$ for $1 \leq i \leq n$. Since we may assume without loss of generality that $a_1 \leq \cdots \leq a_n$ it follows that either $b_1 \geq \cdots \geq b_n$ (if $\lambda < 0$) or $b_1 \leq \cdots \leq b_n$ (if $\lambda \geq 0$).

Theorem 5.3

(i) (Suprunenko [110])
The TSP restricted to non-positive symmetric product matrices is solved by the permutation $\langle 1,3,5,\ldots,n,\ldots,4,2\rangle$.

(ii) (Aizenshtat and Kravchuk [5, 6])
The TSP restricted to non-negative symmetric product matrices is solved by the permutation $\langle 1,n,2,n-2,4,n-4,\ldots,5,n-3,3,n-1\rangle$. □

An application of this problem in statistics is given by Hallin, Melard and Milhaud [60] where the problem of minimizing the autocorrelation coefficient for a given time series is investigated. The paper [60] confirms the results by Aizenshtat and Kravchuk.

Next we describe an application of the product TSP that has been reported in Plante, Lowe and Chandrasekaran [89] (see also Plante [88]). A rotor in a gas turbine engine is driven at very high velocity by the gas flow through the turbine. The efficiency of the turbine is influenced by the distribution of the gas over the rotor blades. The more 'uniform' the distribution of the gas

flow, the more efficient the engine. A nozzle guide vane assembly (*assembly*, for short) is located immediately upstream (relative to the direction of the gas flow) of the rotor. One of the purposes of this assembly is to distribute the gases that drive the rotor. An assembly is composed of several nozzle guide vanes (hereafter called *vanes*). The gases flow through the areas between adjacent vanes.

When a gas turbine engine is overhauled at a maintenance facility, it must often be entirely disassembled to permit detailed inspection of the individual components. Upon disassembly of the engine, some vanes might be discarded, if severely worn, others might be resurfaced. Thus, prior to reassembly, the mechanic has available a collection of vanes that often have characteristics quite different from the vanes that were removed from the engine. The task of the mechanic is to find a sequence of the vanes to be placed in the assembly in order to obtain uniformity of flow areas between adjacent vanes, and hence uniformity of the gas flow distribution.

Assume that n vanes have to be placed in a circular assembly. The assembly has n equidistant slots. Each vane has to be placed in one of these slots. It is assumed that each vane fits in each slot. A placement of the vanes in the assembly can be described by a tour $\tau = \langle i_1, i_2, \ldots, i_n \rangle$ which is to be interpreted as follows. The i_k-th vane is placed directly on the left (in a clockwise way) of the i_{k+1}-th vane ($k = 1, \ldots, n-1$) and the i_n-th vane is placed directly on the left of the i_1-th vane. The characteristics of the i-th vane ($i = 1, \ldots, n$) are given by two nonnegative real numbers a_i and b_i. If the i-th vane is placed directly on the left of the j-th vane then the size of the flow area between these two vanes is given by $f[i,j] = a_i + b_j$ for $i, j = 1, \ldots, n$. Note that the total flow area F does not depend on the placement of the vanes because for all possible placements τ its value is

$$F = \sum_{i=1}^{n} f[i, \tau(i)] = \sum_{i=1}^{n} (a_i + b_{\tau(i)}) = \sum_{i=1}^{n} (a_i + b_i).$$

However, since the efficiency of the turbine engine is determined by the uniformity of the distribution of the gas through the assembly, the objective is to find a placement of vanes such that the dispersion of the sizes of the flow areas is minimized. If the dispersion is measured by the variance, the problem is to find a cyclic permutation τ such that

$$c(\tau) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{n} \sum_{j=1}^{n} f[j, \tau(j)] - f[i, \tau(i)] \right)^2$$

is minimized, where $f[i, \tau(i)] = a_i + b_{\tau(i)}$. By using this equality we obtain

$$c(\tau) = \frac{1}{n} \sum_{i=1}^{n} (a_i^2 + b_i^2) - \frac{F^2}{n^2} + \frac{2}{n} \sum_{i=1}^{n} a_i b_{\tau(i)}.$$

Clearly, the first two terms do not depend on τ. So, the nozzle guide vane placement problem can be formulated as a product TSP, i.e. the problem of finding a cyclic permutation τ that minimizes $\sum_{i=1}^{n} a_i b_{\tau(i)}$, where a_i and b_i are given nonnegative reals, $i = 1, \ldots, n$.

Another application of the product TSP was recently discussed by Ferreira and Guimaraes [47]. They considered a problem to balance the rest periods inbetween regular duties of busdrivers in a rotating duty-schedule (rota). In this application, a_i (b_i respectively) denotes the time in a day after (before) duty i. So, the length of the rest period between duty i and duty j is given by $a_i + b_j$. The problem is to determine a rota (which can be modeled as a tour) which minimizes the dispersion of the rest periods. By similar arguments as above, it can been seen that this problem can be reformulated as a product TSP.

Next, we will argue that heuristic Subtour-patch can be used as a very 'good' heuristic for the product TSP. It will be shown that heuristic Subtour-patch applied to product matrices does have a finite worst-case ratio.

Theorem 5.4 *Let $C = (a_i b_j)$ be a product matrix with $a_i \geq 0$ and $b_j \geq 0$ for $i, j = 1, 2, \ldots, n$. Furthermore, let τ_{opt} be an optimal tour and let τ_{patch} be the tour obtained by heuristic Subtour-patch. Then $c(\tau_{patch}) \leq 2c(\tau_{opt})$.*

Proof. The theorem follows from the following chain of inequalities

$$c(\phi) + P(T_{\min}) \leq c(\tau_{opt}) \leq c(\tau_{patch}) \leq 2\left(c(\phi) + P(T_{\min})\right).$$

What is left to be proved is the rightmost inequality

$$c(\tau_{patch}) \leq 2\left(c(\phi) + P(T_{\min})\right). \tag{24}$$

To simplify the presentation, we prove the inequality only for the case where the tree $T_{\min}$ equals $\{(1,2),(2,3),\ldots,(n-1,n)\}$, which means that $a_1 \leq a_2 \leq \cdots \leq a_n$ and $b_1 \geq b_2 \geq \cdots \geq b_n$ (in the general case, the argument has to be applied to every branch of the tree). Note that this implies that matrix C itself is a Monge matrix. Then the cost of the optimal assignment ϕ is $c(\phi) = \sum_{i=1}^{n} a_i b_i$ and the weight of the tree T is

$$P(T) = \sum_{i=1}^{n-1} w_{ii} = \sum_{i=1}^{n-1} \left(c_{i,i+1} + c_{i+1,i} - c_{ii} - c_{i+1,i+1}\right).$$

Since τ_{patch} is an optimal pyramidal tour in this case, $c(\tau_{patch}) \leq c(\tau_0)$ holds for the pyramidal tour $\tau_0 = \langle 1, n, n-1, n-2, \ldots, 3, 2\rangle$. Hence, in order to establish (24), it is sufficient to prove that $c(\tau_0) \leq 2(c(\phi) + P(T_{\min}))$. Putting everything together, it has to be shown that

$$a_1 b_n + \sum_{i=2}^{n} a_i b_{i-1} \leq 2\left(\sum_{i=1}^{n-1} a_i b_{i+1} + \sum_{i=2}^{n} a_i b_{i-1} - \sum_{i=2}^{n-1} a_i b_i\right),$$

or equivalently, that

$$a_1 b_n + 2\sum_{i=2}^{n-1} a_i b_i \leq 2\sum_{i=1}^{n-1} a_i b_{i+1} + \sum_{i=2}^{n} a_i b_{i-1}. \tag{25}$$

Since the matrix under consideration is a Monge matrix, we have

$$\sum_{i=1}^{n-2} (a_i b_i + a_{i+1} b_{i+1}) \leq \sum_{i=1}^{n-2} (a_i b_{i+1} + a_{i+1} b_i). \tag{26}$$

Furthermore, from $0 \leq b_2 \leq b_1$ and $0 \leq a_{n-1} \leq a_n$ we obtain

$$a_2 b_2 + a_{n-1} b_{n-1} \leq a_2 b_1 + a_n b_{n-1}. \tag{27}$$

Finally, since $a_1 b_n$ is the smallest entry in the matrix, we certainly have

$$a_1 b_n \leq 2a_1 b_2 + 2a_{n-1} b_n + \sum_{i=2}^{n-2} a_i b_{i+1}. \tag{28}$$

Adding inequalities (26), (27) and (28) gives the desired result (25). □

There are also some results known on the asymptotic behavior of the heuristic Subtour-patch.

Theorem 5.5 (Sarvanov [106])
Let a_i and b_i be integers with $a_1 < a_2 < \cdots < a_n$ and $b_{\phi(1)} > b_{\phi(2)} > \cdots > b_{\phi(n)}$. Moreover, let

$$u = \max_{i,j} \left\{ \frac{a_{i+1} - a_i}{a_{j+1} - a_j}, \frac{b_{\phi(i)} - b_{\phi(i+1)}}{b_{\phi(j)} - b_{\phi(j+1)}} \right\}$$

and $p = n/2$. Then on the $n \times n$ product matrix $C = (a_i b_j)$, heuristic Subtour-patch gives a suboptimal solution τ_{patch} with

$$\frac{c(\tau_{patch})}{c(\tau_{opt})} \leq 1 + 2\frac{u^{\frac{2}{3}}}{(p-2)^{\frac{4}{3}}} \left(\sqrt[3]{\frac{9}{16}} - \sqrt[3]{\frac{1}{u(p-2)}} \right)^{-1} + \frac{6u}{(p-1)^2}.$$

Hence, $c(\tau_{patch}) \leq \left(1 + d_0 \frac{u}{p^2}\right) c(\tau_{opt})$ for an appropriate real constant $d_0 > 0$. □

Theorem 5.5 implies that if $\frac{u}{n^2} \to 0$ as n tends to infinity, then $\frac{c(\tau_{patch})}{c(\tau_{opt})} \to 1$. A related result was derived by Danilovich and Deĭneko [29].

Theorem 5.6 (Danilovich and Deĭneko [29])
Let $d > 0$, $a_1 \geq 0$, $b_{\phi(n)} \geq 0$. Furthermore, let $u_i = a_{i+1} - a_i \geq d > 0$ and $v_i = b_{\phi(i)} - b_{\phi(i+1)} \geq d > 0$ for $1 \leq i \leq n-1$, and define

$$M = \max_i \{u_i v_{i+1},\ u_{i+1} v_i\}.$$

Then on the $n \times n$ product matrix $C = (a_i b_j)$, heuristic Subtour-patch finds a suboptimal solution τ_{patch} with

$$\frac{c(\tau_{patch}) - c(\tau_{opt})}{c(\tau_{opt})} \leq \frac{6M}{d^2(n-2)^2}.$$

□

In Plante, Lowe and Chandrasekaran [89] and Van Acken, Molenkamp and Van Dal [1] two alternative heuristics for the product TSP are studied. In [1] it is shown that all three heuristics perform very well on randomly generated product matrices and that none of the heuristics is superior to the other. However, it has been shown in [1] that, unlike Subtour-patch, the other two heuristics do not have a finite worst case ratio for product matrices.

5.3 The TSP with Gilmore-Gomory matrices

Apparently, the first paper in which subtour patching was used for efficiently solving a special case of the TSP was the paper by Gilmore and Gomory [56]. They considered the following problem. Assume that there are n jobs $J_1, \ldots, J_n$ ready for heat-treatment in a furnace. Job J_i, $1 \leq i \leq n$, has to be placed in the furnace at temperature b_i, and after some prescribed variations in temperature (over which we have no control), it is taken out of the furnace at temperature a_i. If job J_j immediately follows job J_i, the temperature must be changed from a_i to b_j. The costs for heating (cooling, respectively) the furnace by one degree when the current temperature is x degrees is given by $h_1(x)$ ($h_2(x)$ respectively), where h_1 and h_2 are given density functions. Hence, if job J_j is placed in the furnace directly after job J_i the inbetween costs are given by

$$c_{ij} = \begin{cases} \int_{a_i}^{b_j} h_1(x)dx \text{ if } b_j \geq a_i, \\ \int_{b_j}^{a_i} h_2(x)dx \text{ if } b_j < a_i. \end{cases} \tag{29}$$

The functions h_1 and h_2 need not be strictly nonnegative, but it will be assumed that $h_1(x)+h_2(x) \geq 0$ for all x, which implies that there is no gain from unnecessary heating and cooling the furnace. The objective is to find a sequence of jobs such that the total cost is minimal. By assuming that the furnace is at temperature b_0 at the start and has to be at temperature a_0 at the end of the sequence, this problem can be formulated as an $(n+1)$-city TSP by introducing a dummy job J_0. In their paper [56], Gilmore and Gomory showed that this special case is solvable in $O(n \log n)$ time. The algorithm essentially is a realization of the idea of subtour-patching.

A matrix $C = (c_{ij})$ with a structure like (29) will be called a *Gilmore-Gomory matrix.* Note that if we permute the rows of C in such a way that $b_1 \leq \cdots \leq b_n$ and the columns such that $a_1 \leq \cdots \leq a_n$, the resulting matrix is a Monge matrix, i.e. Gilmore-Gomory matrices form a subclass of the permuted Monge matrices (see also [57] or [17]). Hence, for Gilmore-Gomory matrices the assignment problem is solvable in $O(n \log n)$ time by sorting. Moreover, for the Gilmore-Gomory TSP the patching permutation $\alpha^\star$ can be found in $O(n)$ time. Summarizing, the subtour-patching algorithm on Gilmore-Gomory matrices takes only $O(n \log n)$ time.

It can also be shown that heuristic Subtour-patch solves this problem to optimality. In order to prove this, it would be sufficient to show that $w(T_{\min}) = P(T_{\min})$ for a minimal spanning tree $T_{\min}$ in the patching graph G_ϕ. However, a much stronger property holds.

Lemma 5.7 *If the distance matrix C is a Gilmore-Gomory matrix, then $w(T) = P(T)$ for every spanning tree T in G_ϕ.*

Proof. The geometrical interpretation presented in Section 4.1 is used in the following arguments. The only difference is that now the points with coordinates (a_i, b_j) for $i, j = 1, \ldots, n$ are considered instead of points with coordinates (i, j) as in the previous constructions. To simplify notations, suppose that $a_1 < a_2 < \cdots < a_n$ and $b_{\phi(1)} < b_{\phi(2)} < \cdots < b_{\phi(n)}$, where ϕ is an optimal assignment. In order to keep the notation consistent with the previous section, the term 'point (i, j)' will be used for the point with coordinates $(a_i, b_{\phi(j)})$.
It is easy to show that for a Gilmore-Gomory matrix the area associated with a rectangle is non-zero if and only if the bisector $x = y$ intersects the rectangle. In Figure 9 these rectangles are dashed. It follows that for every i at least one of the areas associated with the two rectangles $[i, i]^+$ and $[i, i]^-$ equals zero. Hence, for every branch $(j, j+1, \ldots, k-1, k)$ a pyramidal subtour λ on the set $\{j, j+1, \ldots, k\}$ and a corresponding region $A(\lambda)$ can be constructed, such that the area $s(A(\lambda))$ equals the sum of areas of diagonal rectangles. It is sufficient to delete for every branch one of the rectangles with non-zero area (in case such one exists). For the example in Figure 9, rectangles $[2, 2]^-$, $[3, 3]^-$ and $[4, 4]^+$ have to be removed when constructing a pyramidal subtour for a branch containing subbranch $\{1, 2, 3, 4, 5\}$. Hence, $S(A(\lambda)) = w_{i,k-1} = \sum_{j=i}^{k-1} w_{jj}$ for all i and k and therefore $w(T) = P(T)$. □

So, the Gilmore-Gomory TSP is a special case of the TSP that can be solved to optimality by heuristic Subtour-patch. It is an open question to characterize all such cases by identifying algebraic properties for the distance matrix. Chandrasekaran [22] gives a polynomial time recognition algorithm for permuted Gilmore-Gomory matrices (but without providing a convincing proof of correctness).

The No-wait Flow-shop Scheduling problem and TSPs on large matrices. Below we describe a problem that can be reformulated as a TSP restricted to so-called *large* matrices, i.e. matrices $C = (c_{ij})$ such that $c_{ij} = \max\{a_i, b_j\}$ where (a_i) and (b_i) are two given vectors.

Assume that n jobs $J_1, \ldots, J_n$ are to be processed without interruption on m machines $M_1, \ldots, M_m$ in this order (hence, this is an instance of the *flow-shop problem*). Each machine

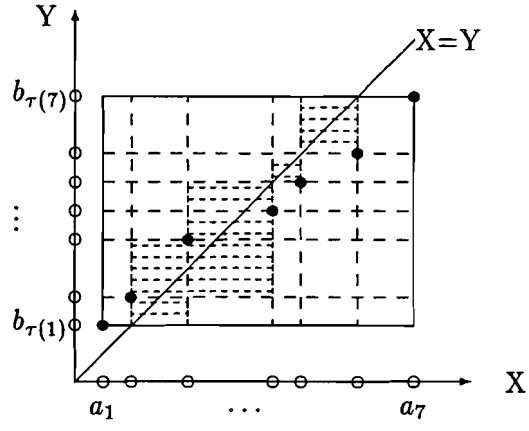


Figure 9: Illustration to the Gilmore-Gomory TSP.

can only process one job at a time and immediately after a job is finished on machine M_k it has to be started on machine M_{k+1} (this is the *no-wait* or *zero-storage* condition). An instance of the problem is given by an $n \times m$ nonnegative matrix (p_{ik}) where p_{ik} denotes the processing time of job J_i on machine M_k. A feasible solution (a schedule) is given by a permutation π of the jobs where $J_{\pi(1)}$ is the first job that is to be processed, $J_{\pi(2)}$ the second and so on. Note that it follows from the no-wait condition that the order in which the jobs are processed on a machine is the same for all machines. The objective is to minimize the makespan, i.e. to find a schedule such that the last job that is processed is finished on the last machine as soon as possible. This problem is called the *No-Wait Flow-Shop Scheduling Problem* (NWFSSP). It is well-known that the NWFSSP is $\mathcal{NP}$-hard if m is part of the input (see Lenstra, Rinnooy Kan and Brucker [71]) and for fixed $m \geq 3$ (see Papadimitriou and Kanellakis [85] for the case $m \geq 4$, and Röck [100] for the case $m = 3$). For fixed $m = 2$ this problem is solvable in $O(n \log n)$ time. In order to show this, we reformulate the n-job NWFSSP as an $(n+1)$-city TSP (this reformulation is due to Reddi and Ramamoorthy [98] and Wismer [126]).

Hence, assume $m = 2$. Let π be a schedule and let $F_{\pi(j)}$ be the completion time of job J_j on the second machine. It is easy to see that if job J_j is scheduled directly after job J_i, then the difference in completion times is given by $F_{\pi(j)} - F_{\pi(i)} = \max\{p_{j2}, p_{j1} + p_{j2} - p_{i2}\}$. In order to transform a schedule into a tour, we introduce a dummy job J_0 with $p_{0k} = 0$ for $k = 1, 2$. A schedule ϕ corresponds to the tour $\tau = \langle 0, \pi(1), \ldots, \pi(n) \rangle$. The distance matrix $C = (c_{ij})$ of the TSP is given by

$$c_{ij} = \max\{p_{j2},\, p_{j1} + p_{j2} - p_{i2}\} = p_{j2} - p_{i2} + \max\{p_{i2}, p_{j1}\}.$$

So, C is a so-called pseudo-large matrix. A matrix (c_{ij}) is called *a pseudo large matrix* if there are vectors p, q, a and b such that $c_{ij} = p_i + q_j + \max\{a_i, b_j\}$ (see Van Dal [114]). Note that the length of the tour τ is equal to the completion time of the last job of schedule π, i.e.

$$c(\tau) = \sum_{i=0}^{n} d_{i,\tau(i)} = F_{\pi(1)} + \sum (F_{\pi(i+1)} - F_{\pi(i)}) = F_{\pi(n)}.$$

We conclude that the NWFSSP with two machines is equivalent to the TSP restricted to large matrices. It is easy to show that the large matrices are a special case of the Gilmore-Gomory

matrices by setting $h_1(x) = 1$ and $h_2(x) = 0$ for all x. Summarizing, the NWFSSP for two machines is solvable in $O(n \log n)$ time.

In Van der Veen and Van Dal [120] it is shown that the m-machine NWFSSP where the processing times are fixed on all except two machines can also be solved in polynomial time because the distance matrix of the corresponding TSP is pseudo large.

Another solvable case of the NWFSSP was studied by Arora and Rana [8]. They considered the NWFSSP restricted to so-called semi-ordered processing time matrices. The processing time matrix (p_{ik}) is called *semi-ordered* if there is an ordering σ of the jobs such that $p_{\sigma(i)k} \leq p_{\sigma(i+1)k}$ for $i = 1, \ldots, n-1$ and $k = 1, \ldots, m$. Such matrices appear in real-life situations where jobs differ greatly in complexity or in the number of units or parts they consist of, and the per unit processing time on each of the machines is constant. Arora and Rana [8] showed that the distance matrix of the TSP corresponding to a semi-ordered processing time matrix $(p_{\sigma(i)k})$ is a Monge matrix. Consequently, the NWFSSP restricted to semi-ordered processing time matrices is solvable in $O(n)$ time.

5.4 A Generalization of the Small TSP

A matrix $C = (c_{ij})$ is called a *small matrix* if there exist numbers a_i and b_i, $1 \leq i \leq n$, such that $c_{ij} = \min\{a_i, b_j\}$ holds for $1 \leq i, j \leq n$. The small TSP (i.e. the TSP restricted to small matrices) is known to be solvable in $O(n)$ time. This result is due to Gabovich [53] and can also be found in Gilmore, Lawler and Shmoys [57] and in Van Dal, Van der Veen and Sierksma [116]. Volodarskij, Gabovich and Zakharin [124] used small matrices in order to generate test problems for the TSP. Moreover, they proved a theorem that characterizes potential values for the cost of an optimal tour for the small TSP.

Observe that if we multiply a small matrix by -1 then we get a Gilmore-Gomory matrix. This suggests to consider cost matrices (c_{ij}) with $c_{ij} = -g_{ij}$, where (g_{ij}) is a Gilmore-Gomory matrix. These matrices form yet another class of permuted Monge matrices. The TSP on such matrices was first investigated by Deĭneko and Trofimov [36] who derived generalizations of the results in [124]. The following theorem characterizes the structure of the patching graph G_ϕ for this special case of the permuted Monge TSP.

Theorem 5.8 *Let $C = (c_{kl})$ be a matrix where $(-c_{kl})$ is a Gilmore-Gomory matrix, let ϕ be an optimal assignment for C and let G_ϕ be the corresponding patching graph. Furthermore, let (w_{ij}) denote the corresponding branch weights, and let the weight $p(i) = w_{i,i}$ denote the weight associated with the edge $(i, i+1)$ in G_ϕ. Then,*

- *There exists at most one edge $(i', i'+1)$ with weight $p_{i'} > 0$.*
- *There exists at most one branch with two edges $\{(j, j+1), (j+1, j+2)\}$ and $w_{j,j+1} > w_{jj} + w_{j+1,j+1}$. In case such a branch exists, the following holds.*
  - *If there exists an edge $(i', i'+1)$ with non-zero weight, then $i' \in \{j, j+1\}$.*
  - *Only branches containing the subbranch $\{(j, j+1), (j+1, j+2)\}$ have non-zero branch weight. In this case, their branch weight equals $w_{j,j+1}$.*

Proof. Let the Gilmore-Gomory matrix $-c_{ij}$ be defined as introduced in Section 5.3 by two functions h_1 and h_2 and by numbers a_i and b_i. To simplify the presentation, suppose that $a_1 < a_2 < \cdots < a_n$ and $b_{\phi(1)} > b_{\phi(2)} > \cdots > b_{\phi(n)}$, where ϕ is an optimal assignment for C.

Once more, the proof is done with the help of the geometrical interpretation (see Figure 10). The term 'point (i,j)' is used for the point with coordinates $(a_i, b_{\phi(j)})$. The points (i,i) for $i = 1,\ldots,n$ are depicted by filled circles in Figure 10.

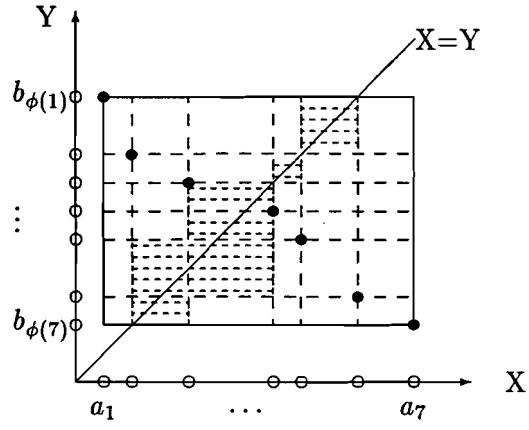


Figure 10: Illustration to the generalization of the small TSP.

Analogously to the Gilmore-Gomory case, the area associated with a rectangle is non-zero if and only if the bisector $x = y$ intersects the rectangle. In Figure 10 these rectangles are the dashed rectangles. So, if $a_{i+1} < b_{\phi(j+1)}$ or $a_i > b_{\phi(j)}$, then $s([i,j]^0) = 0$ must hold. Therefore only the following six cases may arise.

1. $a_n < b_{\phi(n)}$ or $a_1 > b_{\phi(1)}$.
In this case $s([i,j]^0) = 0$ for all rectangles $[i,j]^0$. Hence, all weights of all edges and all branch weights of all branches are equal to zero.

2. There exists an index i^* such that $a_{i^*} > b_{\phi(i^*)}$, $a_{i^*} \geq b_{\phi(i^*+1)}$, but $a_{i^*+1} \geq b_{\phi(i^*)}$.
In this case, the bisector intersects the rectangle $[i^*,i^*]^0$ and therefore $s([i^*,i^*]^0) > 0$ holds. Moreover, all rectangles with non-zero areas lie in the two rectangles $[i^*,i^*]^-$ and $[i^*+1,i^*+1]^+$ in this case. These two rectangles can be removed while constructing a pyramidal tour. Hence, only the edge (i^*,i^*) has non-zero weight. (Note that a rectangle $[i,i]^+$ in Figure 10 has point (i,i) as right upper corner, whereas in the illustrations for the Gilmore-Gomory case such a rectangle has point (i,i) as a lower left corner). This case is illustrated in Figure 10.

3. There exists an index i^* such that $a_{i^*} \leq b_{\phi(i^*)}$, $a_{i^*+1} \leq b_{\phi(i^*)}$, but $a_{i^*} \geq b_{\phi(i^*+1)}$.
This case is symmetric to Case 2. The only difference is that now the rectangles $[i^*,i^*]^+$ and $[i^*+1,i^*+1]^-$ have to be removed.

4. There exists an index i^* such that $a_{i^*} = b_{i^*}$.
In this case all rectangles with non-zero areas lie in the rectangles $[i^*,i^*]^+$ and $[i^*,i^*]^-$. Therefore, all weights of all edges are zero and only branches that contain the two edges (i^*-1,i^*) and (i^*,i^*+1) have non-zero weight. This weight then equals w_{i^*-1,i^*}.

5. There exists an index i^* such that $a_{i^*} \leq b_{\phi(i^*)}$, $a_{i^*} \geq b_{\phi(i^*+1)}$ and $a_{i^*+1} \geq b_{\phi(i^*)}$.
In this case there is only one edge (i^*,i^*+1) with non-zero weight and all branches that contain the edges (i^*-1,i^*) and (i^*,i^*+1) have non-zero weight which is equal to w_{i^*-1,i^*}.

6. There exists an index (i^*) such that $a_{i^*} \geq b_{\phi(i^*)}$, $a_{i^*} \leq b_{\phi(i^*-1)}$ and $a_{i^*-1} \leq b_{\phi(i^*)}$.
This case is treated symmetrically to Case 5. □

Now consider two examples. As a first example, we solve the TSP with distance matrix $(c_{ij}) =$

$(\min\{a_i, b_j\})$ where $(a_i) = (1,2,3,5)$ and $(b_i) = (4,5,6,2)$, see Figure 11(i). An optimal assignment for this case is $\phi = \langle 1,3\rangle\langle 2\rangle\langle 4\rangle$. The graph G_ϕ contains the three edges $(1,2), (2,3)$ and $(3,4)$ with weights $p(1,2) = 0, p(2,3) = 0$ and $p(3,4) = 1$ and two spanning trees $T_1 = \{(1,2),(3,4)\}$ and $T_2 = \{(2,3),(3,4)\}$ with $P(T_1) = P(T_2) = 1$. But $w(T_2) > w(T_1) = P(T_1)$ and, therefore, an optimal solution can only be obtained by using T_1, i.e.

$$\tau_{opt} = \phi \circ \langle 1,2\rangle \circ \langle 3,4\rangle = \langle 1,2,3,4\rangle.$$

As a second example, we solve the TSP with distance matrix $(c_{ij}) = (\min\{a_i, b_j\})$ where $(a_i) = (3,4,5,6)$ and $(b_i) = (2,4,3,5)$, see Figure 11(ii). An optimal assignment for this case is $\phi = \langle 1,4\rangle\langle 2\rangle\langle 3\rangle$. Graph G_ϕ contains three edges $(1,2), (2,3)$ and $(3,4)$ with the weights $p(1,2) = p(2,3) = p(3,4) = 0$ and three spanning trees $T_1 = \{(1,2),(2,3)\}$, $T_2 = \{(2,3),(3,4)\}$ and $T_3 = \{(1,2),(3,4)\}$ with $P(T_1) = P(T_2) = P(T_3) = 0$. But $w(T_1) > w(T_2) = w(T_3)$ and, therefore, an optimal solution can only be obtained by using T_2 or T_3, i.e.

$$\tau_{opt1} = \phi \circ \langle 1,2\rangle \circ \langle 3,4\rangle = \langle 1,2,4,3\rangle$$

or

$$\tau_{opt2.1} = \phi \circ \langle 2,3\rangle \circ \langle 3,4\rangle = \langle 1,4,2,3\rangle$$

and

$$\tau_{opt2.2} = \phi \circ \langle 3,4\rangle \circ \langle 2,3\rangle = \langle 1,4,3,2\rangle.$$

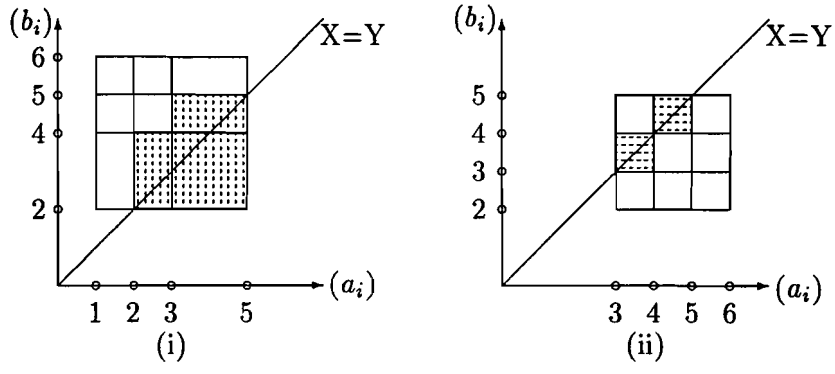


Figure 11: Illustrations to the examples.

It follows from the proof of Theorem 5.8 that heuristic Subtour-patch fails to find an optimal solution if one of the Cases 4, 5 or 6 occurs. The simplest way to avoid this trap is adding the following step to heuristic Subtour-patch inbetween Step 3 and Step 4.

Heuristic Subtour-patch: additional step

Step 3.1: Find in $T_{\min}$ a branch $\{(i^*-1, i^*), (i^*, i^*+1)\}$ with maximal b-weight w_{i^*-1,i^*}. If $w_{i^*-1,i^*} > 0$, try to find a spanning tree T that does not use both edges (i^*-1, i^*) and (i^*, i^*+1). In case T exists, set $T_{\min} := T$.

It is easy to verify that this modification of the heuristic solves the generalized small TSP to optimality.

6 Various special cases

In this section well-solvable cases of relaxations of the TSP are considered as well as the TSP for classes of special matrices (graphs) not considered in the preceding sections. Section 6.1 is devoted to well-solvable cases of the linear programming relaxation of the TSP. In Section 6.2 we consider well-solvable cases of the so-called graphical TSP, a relaxation of the TSP obtained by relaxing the constraint of the classical TSP stating that every city is visited exactly once and every edge is used at most once. In Section 6.3 the TSP for circulant matrices is considered. Finally, in Section 6.4 we discuss the TSP for matrices whose description can be given in a concise way.

6.1 Well-solvable cases of the LP relaxation

In this section well-solvable cases of the linear programming (LP) relaxation of the TSP are considered. We will restrict ourselves to the symmetric case.

The TSP can be formulated as an integer linear programming problem as follows. Let n be the number of cities and let x_{ij} be a binary variable indicating whether or not the salesman travels directly either from i to j or from j to i. Furthermore, let c_{ij} be the distance between i and j, with $i < j$ $(i, j = 1, \ldots, n)$. Then the (symmetric) TSP is the problem to minimize

$$\sum_{i=1}^{n} \sum_{j>i} c_{ij} x_{ij} \tag{30}$$

subject to

$$\sum_{i<j} x_{ij} + \sum_{i>j} x_{ji} = 2, \qquad j = 2, \ldots, n, \tag{31}$$

$$\sum_{i \in S} \sum_{j \in S, j>i} x_{ij} \leq |S| - 1, \text{ for all } S \subset \{1, \ldots, n\},\ S \neq \emptyset, \tag{32}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n,\ i < j. \tag{33}$$

If we replace the 0–1 restrictions (33) by the so-called trivial inequalities

$$0 \leq x_{ij} \leq 1,\ i, j = 1, \ldots, n,\ i < j \tag{34}$$

then the problem of minimizing (30) subject to the degree restrictions (31), the subtour elimination constraints (32) and the trivial inequalities (34) is called the linear programming (LP) relaxation of the TSP or, also, *the subtour relaxation problem.*

Note that the subtour relaxation problem has an integer optimal solution if and only if the optimal values of the TSP and the subtour relaxation problem coincide. Moreover, the subtour relaxation problem can be solved in polynomial time, because the facet identification problem for subtour elimination constraints, i.e. the problem to decide which subtour elimination is violated by the current solution, can be solved in polynomial time, despite the fact that their number is exponential (see Padberg and Grötschel [80]).

A natural question to ask is to characterize those classes of instances of the TSP for which the TSP subtour relaxation problem has an integer optimal solution. Obviously, the TSP for these classes can be solved in polynomial time. However, Chvátal [23], has proven that the problem of recognizing instances of the TSP for which the subtour relaxation problem has an integer optimal solution, is $\mathcal{NP}$-hard by reducing to it the problem of recognizing those cubic 3-edge-connected graphs which are Hamiltonian. Consequently, perhaps the most that we can realistically hope for

is to find several classes of instances of the TSP for which the subtour relaxation problem has an integer optimal solution. We will consider a few of these classes in the following.

One such class we have already met. It can be shown that the TSP subtour relaxation problem has an integer optimal solution in case there exists a necklace tour (see the discussion in Section 3.3).

Papadimitriou and Steiglitz [86] have constructed instances of symmetric TSPs with the property that edge exchange (or local search) algorithms are ineffective for their solution. These instances have exponentially many suboptimal solutions of arbitrarily large cost. Once we reach a suboptimal solution, we have to perform in the worst case exponentially many iterations of an edge exchange algorithm to reach the optimal solution. Papadimitriou and Steiglitz called these instances *traps* for the TSP, i.e. instances of the TSP that are hard to solve.

Although edge exchange algorithms are ineffective for the solution of the traps, Padberg and Sung [81] have shown that the traps can be solved in polynomial time by solving a relaxation of the TSP consisting of the degree equations (31) and the trivial inequalities (34), and appended with some suitable subtour elimination constraints. Therefore, the traps are an example of a class of instances of the TSP for which the optimal value of the subtour relaxation problem equals the optimal value of the TSP, i.e. the traps form a well-solvable special case of the TSP.

In the above we discussed the (symmetric) TSP defined on the complete graph K_n. One may as well study the TSP on an arbitrary graph $G = (V, E)$ and its associated polytope

$$Q_T(G) = \lim conv\left\{x^I \in \mathbb{R}^E : x^I \text{ is the characteristic vector of a tour } T \text{ in } G\right\}.$$

For special classes of graphs G it might be easy to give a complete description of $Q_T(G)$. Such investigations may lead to the discovery of new special classes of graphs for which the TSP is solvable in polynomial time. For instance, Cornuéjols, Naddef and Pulleyblank [25, 26] (see also Gilmore, Lawler and Shmoys [57]) have studied the class of graphs G such that $Q_T(G)$ is given by the trivial inequalities, the degree equations and one equation for each 3-edge cut-set, where an *edge cut-set* defined by $W \subseteq V$ is the set of edges $\{u, v\} \in E$ with $u \in W$ and $v \in V - W$. This class contains $K_{3,3}$, wheels, Halin graphs and some other graphs.

6.2 Special cases of the graphical TSP

In this section we discuss well-solvable cases of the graphical TSP. The *graphical TSP* is a relaxation of the TSP in which it is allowed to visit each city more than once and to use each edge more than once. However, we will first consider the so-called Steiner TSP, a related problem.

Let $G = (V, E)$ be a graph and $P \subseteq V$. The elements of $V \setminus P$ are called *Steiner points*. A *Steiner tour* of G is a closed walk that visits each vertex of P at least once. Therefore, there are two differences between a Steiner tour and a traveling salesman tour. First, in a Steiner tour the Steiner points do not have to be visited, and secondly, a Steiner tour may contain some vertices (and edges) more than once. The *Steiner TSP* is the problem of finding a minimum length Steiner tour in G.

A real-life problem that can be formulated as a Steiner TSP is the *order-picking problem* which is defined as follows. Suppose the sales department of a firm receives an order, i.e. a list of items demanded, from a customer. A vehicle leaves the loading dock to collect or to pick up the items in the order and to transport them back to the loading dock. We assume that the capacity of the vehicle is sufficient to pick up all the items in the order. The objective is to minimize the total distance traversed by the vehicle.

This problem is easily recognized as a TSP. Therefore, for the order-picking problem to be efficiently solvable we have to assume some structure on the underlying graph of the warehouse.

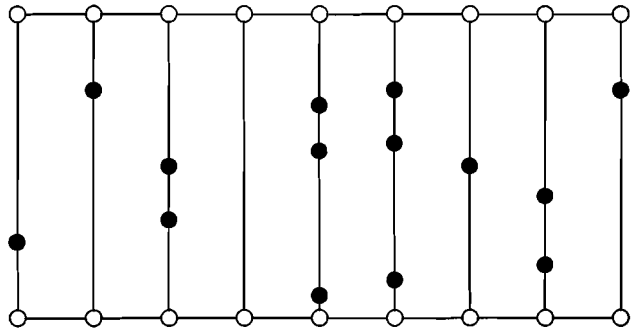


Figure 12: Graph of a rectangular warehouse.

An example of such a graph is the graph of a rectangular warehouse (see Figure 12), which is the warehouse configuration most frequently encountered in practice. The order-picking problem in a rectangular warehouse will be formulated as a Steiner TSP. The vertices in P represent the locations of the items in a particular order stored along the aisles in the warehouse and the location of the loading dock, and the vertices in $V \setminus P$, i.e. the Steiner points, represent the intersections of aisles and crossovers (in Figure 12, these points are denoted by $\bullet$ and $\circ$, respectively). The distance between any pair of vertices u and v is the length of a shortest path between u and v.

Ratliff and Rosenthal [97] gave an $O(m)$ time dynamic programming algorithm for solving the order-picking problem in a rectangular warehouse, where m is the number of aisles. This problem was also solved (independently, but four years later) by Carlier and Villon [20].

The graph in Figure 12 is an example of a series-parallel graph, defined as follows. Starting from the graph consisting of two vertices joined by a single edge, a *series-parallel graph* can be constructed by applying recursively a *series operation*, i.e. replacing an edge $\{u, v\}$ by two edges $\{u, w\}$ and $\{w, v\}$ where w is a new vertex, or a *parallel operation*, i.e. duplicating an edge. Cornuéjols, Fonlupt and Naddef [24] showed that Ratliff and Rosenthal's algorithm can be extended to solve the Steiner TSP in all series-parallel graphs. Van Dal [114] considered the Steiner TSP for two non-series-parallel graphs and showed that, due to the sparsity of these graphs, these problems can also be solved using the same dynamic programming algorithm. Additional classes of graphs for which a similar approach works are given in Fonlupt and Nachef [50].

If $P = V$ then the Steiner TSP is called the graphical Traveling Salesman Problem. That is, in the graphical TSP all vertices have to be visited, but it is still allowed to use some vertices (and edges) more than once. The description of the graphical Traveling Salesman polyhedron, i.e. the convex hull of the (nonnegative integer) incidence vectors associated with the Steiner tours of G, has been the subject of extensive research; see e.g. Cornuéjols, Fonlupt and Naddef [24], Fleischmann [48], Naddef and Rinaldi [77], [78], [79], and Naddef [76]. One important reason for the interest in the graphical Traveling Salesman polyhedron is the fact that it contains the symmetric Traveling Salesman polytope as a face.

However, in contrast with the (classical) TSP, Fonlupt and Naddef [51] showed that it is possible to characterize those graphs for which an optimal solution to the subtour relaxation problem of the

graphical TSP, i.e. minimizing

$$\sum_{i=1}^{n}\sum_{j>i} c_{ij}x_{ij}$$

subject to

$$x_{ij} \geq 0,\ i,j = 1,\ldots,n,\ i < j,$$

and

$$\sum_{i\in S}\sum_{j\notin S, j>i} x_{ij} + \sum_{i\notin S}\sum_{j\in S, j>i} x_{ij} \geq 2, \text{ for all } S \subset \{1,\ldots,n\},\ S \neq \emptyset,$$

is integer. The authors call these graphs TSP-perfect graphs. The characterization is given both in terms of excluded minors and in terms of (like in the definition of a series-parallel graph) constructing a TSP-perfect graph by repeated application of two operations.

6.3 The Circulant TSP

In this section we consider the TSP for circulant matrices. An $n \times n$ matrix A is called a *circulant* matrix if it is of the form

$$\begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-2} & a_{n-1} \\ a_{n-1} & a_0 & a_1 & \cdots & a_{n-3} & a_{n-2} \\ a_{n-2} & a_{n-1} & a_0 & \cdots & a_{n-4} & a_{n-3} \\ \vdots & & & & & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_{n-1} & a_0 \end{pmatrix},$$

i.e. for each $i,j = 1,\ldots,n$ and $k = 0,1,\ldots,n-1$, all elements (i,j) with $j - i = k \pmod n$ have the same value a_k. These elements form the so-called k-th *stripe* of A. Obviously, a circulant matrix is determined by its first row (or column). Circulant matrices and their properties have been studied extensively in Davis [30].

Let $n, c_1, c_2, \ldots, c_m$ be integers with $0 < c_1 < c_2 < \cdots < c_m < n$ and let $N = \{1,2,\ldots,n\}$. A directed graph (digraph) $D = (N, A)$ is called a *circulant digraph* if for every arc $(i,j) \in A$ there is a k, $1 \leq k \leq m$, such that $j - i = c_k \pmod n$. In other words, a circulant digraph is a digraph with a circulant adjacency matrix. A circulant (di)graph will be denoted by $C_n(c_1, c_2, \ldots, c_m)$. In this section a clear distinction will be made between (undirected) graphs and digraphs. By the notation (di)graph, both graph and digraph are meant. Note that for a circulant graph there is a c_j with $c_j = n - c_i \pmod n$ for every c_i, $i,j = 1,2,\ldots,m$.

Several authors have formulated conditions for connectivity and Hamiltonicity of circulant (di)graphs. Garfinkel [55] proved that the number of directed cycles associated with the k-th stripe of an $n \times n$ circulant matrix is given by $\gcd(k,n)$. Boesch and Tindell [10] proved that the circulant graph $C_n(c_1, c_2, \ldots, c_m)$ is connected if and only if

$$\gcd(n, c_1, c_2, \ldots, c_m) = 1$$

and conjectured that all connected circulant graphs are Hamiltonian. This conjecture has been proven by Burkard and Sandholzer [18].

The nearest neighbor heuristic solves the problem of finding a shortest Hamiltonian *path* in circulant (di)graphs, see e.g. Gilmore, Lawler and Shmoys [57]. However, although there exists a characterization for circulant graphs being Hamiltonian, even the TSP for symmetric circulant matrices is still an open problem. (In case of the bottleneck objective, however, the problem

is efficiently solvable, see Burkard and Sandholzer [18]). Medova-Dempster [73] investigated the complexity of the TSP for asymmetric circulant matrices and conjectured that this problem is $\mathcal{NP}$-hard. Further, she gave a simple branch-and-bound algorithm and showed that its time complexity is exponential. Van der Veen [117] described two heuristics for the TSP restricted to symmetric circulant matrices. One of these two heuristics is based on the above condition for Hamiltonicity of a circulant graph and has a worst case ratio of two, while the other is based on the nearest neighbor heuristic which, as mentioned above, solves the Hamiltonian path problem for circulant matrices. Further, he shows that at least one of the two heuristics gives an optimal solution for quite a broad class of symmetric circulant matrices. Computational experiments show that the combined heuristic outputs near-optimal tours.

Van Doorn [123] derived an explicit expression for the connectivity of circulant digraphs. However, the problem of characterizing the circulant digraphs that are Hamiltonian is also still an open problem. Yang, Burkard, Çela and Woeginger [127] gave necessary and sufficient conditions for circulant digraphs with $m = 2$ for being Hamiltonian by proving the following theorem.

Theorem 6.1 (Yang, Burkard, Çela and Woeginger [127])
The circulant digraph $C_n(c_1, c_2)$ is Hamiltonian if and only if $\gcd(n, c_1, c_2) = 1$ *and there exist integers $p, q \geq 0$, with $p + q \geq 1$ such that $(p+q)|(c_2 - c_1)$ and* $\gcd(n, pc_1 + qc_2) = p + q$.
Given the three integers n, c_1 and c_2, the above condition can be verified in $O(\log^4 n)$ time. □

Circulant matrices form a subclass of the class of Toeplitz matrices. A matrix (c_{ij}) is called a *Toeplitz matrix* if $c_{ij} = c_{i,j-1}$ for all i and j. Note that a Toeplitz matrix contains $2n-1$ independent elements, which occur in the first row and column. The TSP restricted to (symmetric) Toeplitz matrices is still open. Hamiltonian properties of the associated Toeplitz graph are studied in Van Dal, Tijssen, Van der Veen, Zamfirescu and Zamfirescu [115].

6.4 The TSP on matrices with concise descriptions

In general, an $n \times n$ distance matrix is specified by listing all of its n^2 entries. This section deals with matrices that can be described in a more concise way by only $O(n)$ numbers together with some rule how to construct the matrix from these $O(n)$ numbers. Several examples for this type of matrix have already been treated earlier in this survey, e.g. product matrices (described by the $2n$ numbers a_i and b_i and the rule $c_{ij} = a_i b_j$), small matrices (described by the $2n$ numbers a_i and b_i and the rule $c_{ij} = \min\{a_i, b_j\}$) and large matrices (described by the $2n$ numbers a_i and b_i and the rule $c_{ij} = \max\{a_i, b_j\}$).

The first result we will discuss here deals with the so-called *many visits TSP* as introduced by Cosmadakis and Papadimitriou [27]. Here we are given K cities (where K is some small constant number), the $K \times K$ distance matrix D for these K cities and K integers $n_1, \ldots, n_K$, where $\sum_{k=1}^{K} n_k = n$. We are asked to find the shortest walk that visits the first city n_1 times, the second city n_2 times, and so on (we are allowed to visit city i twice in a row, but this costs us d_{ii}).

The many-visits TSP also arises in scheduling theory, where the cities correspond to tasks that have to be executed on a single machine. The distance c_{ij} reflects the change-over time if task j immediately follows task i and the objective is to minimize makespan, i.e. to minimize the total change-over time. In certain applications, each task belongs to one of K types, and tasks of the same type have identical characteristics. One example of such a problem is known as the *Aircraft Sequencing Problem* (ASP), see e.g. Psaraftis [92]. Assume there are n airplanes waiting for permission to land on a single runway. The airplanes can be divided into K categories, e.g. regular, private, and military airplanes. Safety regulations require a certain time-gap between the landing

of two airplanes. This time-gap depends on the types of the two airplanes. Now assume that there are n_k airplanes of type K pending for landing ($k = 1, \ldots, K$). The ASP is the problem of finding a sequence in which the $n = n_1 + \ldots + n_K$ will land in order to minimize the total amount of time.

Cosmadakis and Papadimitriou [27] gave an $O(e(K) \log n)$ time algorithm for this problem where $e(K)$ is a moderately growing exponential function of K (note that for fixed K this implies that the many-visits TSP is polynomially solvable). This is done by decomposing the many-visits TSP into two problems: a *transportation problem*, whose solution guarantees that each city i is visited and departed from exactly n_k times, and a *connectivity problem*, which forbids 'subtours' in the solution. Due to the special combinatorial structure of the many-visits TSP, the connectivity problem can be solved efficiently. An alternative solution method based on integer programming yielding the same computational complexity has been given in Van der Veen and Zhang [122].

Van der Veen, Woeginger and Zhang [121] investigate the so-called *K-template* TSP. An instance of this problem is fully described by the numbers n and K with $1 \leq K \leq n$, by numbers a_i and b_i, $1 \leq i \leq n$, and by a partition $N_1, \ldots, N_K$, $\cup_{k=1}^{K} N_i = \{1, \ldots, n\}$ of the cities into K *groups*. The distance between two cities i and j is defined by

$$c_{ij} = \begin{cases} a_i & \text{if } i \text{ and } j \text{ are from the same group} \\ b_j & \text{if } i \text{ and } j \text{ are from different groups.} \end{cases}$$

The motivation for considering the K-template TSP comes from scheduling. Assume, there are n jobs, $J_1, \ldots, J_n$, to be processed without interruption on a single machine. In order to process a job, a certain additional resource is required, e.g. a fixture or a *template*. In total there are K different templates ($K \leq n$) each with a unique form. It is assumed that each template can contain only one job at a time and that jobs can only be processed in one (job-specific) template. So, the jobs can be partitioned according to the template they require. Furthermore, there is a change-over time between the processing of two jobs. This change-over time is determined by the following two rules. (R1) After a job has been processed, it has to remain in its template for a certain time period to allow some after-processing, e.g. cooling. For job J_i this time period is denoted by a_i, $1 \leq i \leq n$. However, it is not necessary that the template remains placed on the machine. If a different template is to be used, the template (with the job still in it) can be removed from the machine and the after-processing can take place off-line. (R2) If the same template is not used for the following job, then a set-up time is needed. The length of this set-up time depends on the job that is processed next. For job J_i this set-up time is denoted by b_i, $1 \leq i \leq n$.

In order to get an expression for the change-over time, suppose that job J_j is scheduled directly after job J_i, and that $i \in N_p$ and $j \in N_q$. If both jobs require the same template (i.e. $p = q$) then the change-over time is given by a_i because the after-processing of job J_i is done on-line, so only after that time job J_j can be placed in the template for processing. If J_j requires another template than J_i (i.e. $p \neq q$) then the change-over time is given by b_j because in this case the template with J_i is can be removed from the machine (the after processing takes place off-line) and the machine has to be set up in order to process job J_j. Clearly, the problem of minimizing the total change-over time corresponds to solving a K-template TSP. In [121], an $O(n \log n)$ time solution algorithm for the K-template TSP is derived.

Yet another application of a solvable TSP in this category occurs in *printed circuit board* (PCB) design. Assume that n parts are to be inserted by a robot arm at specified points on a PCB. Each of the parts belongs to one of K part-types. All parts of a given type are stored in a bin. The K bins are located along one side of the PCB. Furthermore, it is assumed that the robot arm can only move sequentially in either horizontal or vertical direction, so that the robot arm travel time from

one location to another is proportional to the rectilinear distance between the two locations. The *Robot Tour Problem* (RTP) is to find an optimal sequence of insertions of the parts in the PCB, given the locations of the n insertion points and the locations of the K bins. See Hamacher [59] for more details and extensions.

For fixed locations of the bins, the RTP can be reformulated as a TSP on the n insertion points. Obviously, the distance from one insertion point to another equals the distance from the first insertion point to the bin containing the part to be inserted in the second insertion point plus the distance from this bin to the second insertion point. Note that the second part of this distance is fixed, in the sense that independent of the sequence this distance has to be traveled anyhow. However, the first part is sequence dependent. Since all bins are located at the same side of the PCB and since distances are measured rectilinearly, the only part of this first part of the distance that is really sequence dependent is the distance from the projection of the insertion point on the side of the PCB where the bins are located to the next bin. Let N_k be the set of all parts of type k ($k = 1, \ldots, K$). Note that if a part from N_k is to be inserted after a given part i the sequence-dependent 'distance' c_{ij} to travel from i to $j \in N_k$ has the same value for all $j \in N_k$. It follows that the distance matrix $C = (c_{ij})$ of the TSP modeling the RTP is given by $c_{ij} = a_i^k$ if $j \in N_k$ for all $i, j \in \{1, \ldots, n\}$, where $a^1, \ldots, a^K$ are given n-dimensional vectors. A polynomial time algorithm for solving this RTP has been given in Ball and Magazine [9]

In the TSP restricted to *2-norm matrices*, the distance matrix $C = (c_{ij})$ is specified by the numbers a_i and b_i, $1 \leq i \leq n$, and by the rule $c_{ij} = |a_i - b_j|^2$. In this case the length of a tour τ is given by

$$c(\tau) = \sum_{i=1}^{n} (a_i - b_{\tau(i)})^2 = \sum_{i=1}^{n} (a_i^2 + b_i^2) - 2 \sum_{i=1}^{n} a_i \cdot b_{\tau(i)}.$$

So the problem of finding a minimum length tour is equivalent to the product TSP which is $\mathcal{NP}$-hard, see Theorem 5.2. This is in contrast with the TSP restricted to *1-norm matrices* C which are given by $c_{ij} = |a_i - b_j|$ for all i and j. It is easy to see that a 1-norm matrix is a Gilmore-Gomory matrix, hence the 1-norm TSP is solvable in $O(n \log n)$ time.

Finally, we mention the TSP for Brownian matrices. An $n \times n$ matrix $C = (c_{ij})$ is called a *Brownian matrix* if there are two vectors $a, b \in \mathbb{R}^n$ such that

$$c_{ij} = \begin{cases} a_i & \text{if } i < j \\ b_j & \text{otherwise} \end{cases}$$

Burkard and Van der Veen [19] proved that the TSP for Brownian matrices can be solved in $O(n \log n)$ time.

Acknowledgment. We thank Eranda Çela, Vitali Demidenko, Bettina Klinz, Nikolai Metelsky, Günter Rote and Gerard Sierksma for helpful discussions and for providing pointers to the literature.

References

[1] G.A. Van Acken, J.B. Molenkamp and R. Van Dal, Simple patch: A new heuristic for the product traveling salesman, Report 93-85, Technical University Delft, Faculty of Technical Mathematics, 1993.

[2] A. Aggarwal, M.M. Klawe, S. Moran, P. Shor and R. Wilber, Geometric applications of a matrix-searching algorithm, *Algorithmica* **2**, 1987, 195–208.

[3] V.S. Aizenshtat, The reduction theorem in the problem of minimizing a linear form on the set of cyclic permutations, *Dokl. Akad. Nauk BSSR* **8**, 1970, 685–688, (in Russian).

[4] V.S. Aizenshtat, Talk at the 4th Republican Conference of Belorussian Mathematicians, 1975, Minsk, Belorussia, (in Russian).

[5] V.S. Aizenshtat and D.N. Kravchuk, Algorithms for finding the extremum of a linear form on the set of all cycles in special cases, *Dokl. Akad. Nauk BSSR* **12**, 1968, 401–404, (in Russian).

[6] V.S. Aizenshtat and D.N. Kravchuk, On the minimum of a linear form on the set of all cycles of the symmetric group S_n, *Kibernetika*, Issue 2, 1968, 64–66, (in Russian), English translation in *Cybernetics* **4**, 1968.

[7] V.S. Aizenshtat and E.P. Maksimovich, Certain classes of traveling salesman problems, *Kibernetika*, Issue 4, 1978, 80–83, (in Russian), English translation in *Cybernetics* **14**, 1979, 565–569.

[8] R.K Arora and S.P. Rana, Scheduling in a semi-ordered flow shop without intermediate queues, *AIIE Transactions* **12**, 1980, 263–272.

[9] M.O. Ball and M.J. Magazine, Sequencing of insertions in printed circuit board assembly, *Operations Research* **36**, 1988, 192–201.

[10] F. Boesch and R. Tindell, Circulants and their connectivities, *Journal of Graph Theory* **8**, 1984, 487–49.

[11] D. Blokh and G. Gutin, Maximizing traveling salesman problem for special matrices, *Discrete Applied Mathematics* **56**, 1995, 83–86.

[12] V.Y. Burdyuk and V.G. Deĭneko, On solvable cases of the multi-traveling salesmen problem, *Teoriya optimalnych reshenij*, Kiev, 1979, 3–9, (in Russian).

[13] V.Y. Burdyuk and V.N. Trofimov, Generalization of the results of Gilmore and Gomory on the solution of the traveling salesman problem, *Izw. Akad. Nauk SSSR, Tech. Kibernet.*, Issue 3, 1976, 16–22, (in Russian), English translation in *Engineering Cybernetics* **14**, 1976, 12–18.

[14] R.E. Burkard, Special cases of traveling salesman problems and heuristics, *Acta Mathematicae Applicatae Sinica* **6**, 1990, 273–288.

[15] R.E. Burkard and V.G. Deĭneko, Polynomially solvable cases of the traveling salesman problem and a new exponential neighborhood, *Computing* **54**, 1995, 191–211.

[16] R.E. Burkard, V.G. Deĭneko and G.J. Woeginger, The traveling salesman problem on permuted Monge matrices, Technical Report, 1995, Institut für Mathematik, TU Graz, Austria.

[17] R.E. Burkard, B. Klinz and R. Rudolf, Perspectives of Monge Properties in Optimization, to appear in *Discrete Applied Mathematics.*

[18] R.E. Burkard and W. Sandholzer, Efficiently solvable special cases of bottleneck traveling salesman problems, *Discrete Applied Mathematics* **32**, 1991, 61–67.

[19] R.E. Burkard and J.A.A. Van der Veen, Universal conditions for algebraic traveling salesman problems to be efficiently solvable, *Optimization* **22**, 1991, 787–814.

[20] J. Carlier and P. Villon, A well solved case of the traveling salesman problem, Research Report, Université de Technologie de Compiègne, 1987.

[21] K. Čechlárová and P. Szabó, On the Monge property of matrices, *Discrete Mathematics* **81**, 1990, 123–128.

[22] R. Chandrasekaran, Recognition of Gilmore-Gomory traveling salesman problem, *Discrete Applied Mathematics* **14**, 1986, 231–238.

[23] V. Chvátal, A note on the traveling salesman problem, *Operations Research Letters* **8**, 1989, 77–78.

[24] G. Cornuéjols, J. Fonlupt and D. Naddef, The traveling salesman problem on a graph and some related integer polyhedra, *Mathematical Programming* **33**, 1985, 1–27.

[25] G. Cornuéjols, D. Naddef and W.R. Pulleyblank, Halin graphs and the traveling salesman problem, *Mathematical Programming* **26**, 1983, 287–294.

[26] G. Cornuéjols, D. Naddef and W.R. Pulleyblank, The traveling salesman problem in graphs with 3-edge cutsets, *Journal of the ACM* **32**, 1985, 383–410.

[27] S.C. Cosmadakis and C.H. Papadimitriou, The traveling salesman problem with many visits to few cities, *SIAM J. Computing* **13**, 1984, 99–108.

[28] M. Cutler, Efficient special case algorithms for the N-line planar traveling salesman problem, *Networks* **10**, 1980, 183–195.

[29] I.S. Danilovich and V.G. Deĭneko, On the estimation of accuracy of an approximation algorithm, *Metody resheniya nelinejnych zadach i obrabotki dannych*, Dnepropetrovsk, 1983, 84–85, (in Russian).

[30] Ph.J. Davis, *Circulant matrices*, John Wiley and Sons, New York, 1979.

[31] V.G. Deĭneko, Applying dynamic programming to solving a special multi-traveling salesmen problem, *Issledovanie operaziy i ASU* **14**, Kiev, 1979, 47–50, (in Russian).

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

[33] V. Deĭneko, R. Rudolf, J.A.A. Van der Veen and G.J. Woeginger, Three Easy Special Cases of the Euclidean Traveling Salesman Problem, SFB Report 17, 1995, Institut für Mathematik, TU Graz, Austria.

[34] V.G. Deĭneko, R. Rudolf and G.J. Woeginger, On the Recognition of Permuted Supnick and Incomplete Monge Matrices, to appear in *Acta Informatica.*

[35] V. Deĭneko, R. Rudolf and G.J. Woeginger, Sometimes Traveling is Easy: The Master Tour Problem, SFB Report 15, 1994, Institut für Mathematik, TU Graz, Austria.

[36] V.G. Deĭneko and V.N. Trofimov, On possible values of the function in the special traveling salesman problem, *Kibernetika*, Issue 6, 1981, 135–136, (in Russian).

[37] V. Deĭneko, R. Van Dal and G. Rote, The convex-hull-and-line traveling salesman problem: a solvable case, *Information Processing Letters* **51**, 1994, 141–148.

[38] V. Deĭneko and G.J. Woeginger, Long-Chord- and Fence-Free Tours for the traveling salesman problem, SFB Report 46, 1995, Institut für Mathematik, TU Graz, Austria.

[39] V.M. Demidenko, A special case of traveling salesman problems, *Izv. Akad. Nauk. BSSR, Ser. Fiz.-mat. Nauk* **5**, 1976, 28–32, (in Russian).

[40] V.M. Demidenko, The traveling salesman problem with asymmetric matrices, *Izv. Akad. Nauk. BSSR, Ser. Fiz.-mat. Nauk* **1**, 1979, 29–35, (in Russian).

[41] V.M. Demidenko, An extension of Supnick conditions for the asymmetric traveling salesman problem and its application, manuscript, Belorussian Academy of Sciences, 1995.

[42] V.M. Demidenko and N.N. Metelsky, On the problem of minimizing a linear form on the set of all cycles, Talk on the 3th Allunion conference on problems of theoretical cybernetics, Novosibirsk, 1974, (in Russian).

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

[44] H. Edelsbrunner, G. Rote and E. Welzl, Testing the necklace condition for shortest tours and optimal factors in the plane, *Theoretical Computer Science* **66**, 1989, 157–180.

[45] D. Eppstein, Sequence comparison with mixed convex and concave costs, *Journal of Algorithms* **11**, 1990, 85–101.

[46] Md. Fazle Baki and S.N. Kabadi, Some sufficient conditions for pyramidal optimal traveling salesman tour, Working paper #95-021, Faculty of Administration, University of New Brunswick, 1995.

[47] J.V. Ferreira and R.C. Guimaraes, A traveling salesman problem for the sequencing of duties in bus crew rotas, *Journal of the Operational Research Society* **46**, 1995, 415–426.

[48] B. Fleischmann, A new class of cutting planes for the symmetric traveling salesman problem, *Mathematical Programming* **40**, 1988, 225–246.

[49] M.M. Flood, The traveling salesman problem, *Operations Research* **4**, 1956, 61–75.

[50] J. Fonlupt and A. Nachef, Dynamic programming and the graphical traveling salesman problem, *Journal of the ACM* **40**, 1993, 1165-1187.

[51] J. Fonlupt and D. Naddef, The traveling salesman problem in graphs with some excluded minors, *Mathematical Programming* **53**, 1992, 147–172.

[52] H.N. Gabov, Data structures for weighted matching and nearest common ancestors with linking, *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, Philadelphia, 1990, 434–443.

[53] E.Y. Gabovich, The small traveling salesman problem, *Trudy Vychisl. Centra Tartu. Gos. Univ.* **19**, 1970, 17–51, (in Russian).

[54] N.E. Gaikov, On the minimization of a linear form on cycles, Minsk, 1980, Manuscript presented by *Vesti Akad. Nauk BSSR, Ser. Fiz. mat. Nauk* **4**, 128, deposited in VINITI (AISTI), Moscow, No.479-80, (in Russian).

[55] R.S. Garfinkel, Minimizing wallpaper waste, part 1: A class of traveling salesman problems, *Operations Research* **25**, 1977, 741–751.

[56] P.C. Gilmore and R.E. Gomory, Sequencing a one state-variable machine: a solvable case of the traveling salesman problem, *Operations Research* **12**, 1964, 655–679.

[57] P.C. Gilmore, E.L. Lawler and D.B. Shmoys, Well-solved special cases, Chapter 4 in [70], 87–143.

[58] F. Glover and A.P. Punnen, The traveling salesman problem: New solvable cases and linkages with the development of approximation algorithms, manuscript, University of Colorado, Boulder, 1995.

[59] H.W. Hamacher, Combinatorial optimization problems motivated by robotic assembly problems, in: M. Akgul (Ed.), Combinatorial optimization, *NATO ASI series* F82, 1992, 187–198.

[60] M. Hallin, G. Melard and X. Milhaud, Permutational extreme values of autocorrelation coefficients and a Pitman test against serial dependence, *Annals of Statistics* **20**, 1992, 523–534.

[61] A.J. Hoffman, On simple linear programming problems, *Convexity, Proc. Symposia in Pure Mathematics*, AMS, Providence, RI, 1961, 317–327.

[62] K. Kalmanson, Edgeconvex circuits and the traveling salesman problem, *Canadian Journal of Mathematics* **27**, 1975, 1000–1010.

[63] P.S. Klyaus, The structure of the optimal solutions of some classes of the traveling salesman problem, *Izv. Akad. Nauk. BSSR, Ser. Fiz.-mat. Nauk* **6**, 1976, 95–98, (in Russian).

[64] P.S. Klyaus, Generation of testproblems for the traveling salesman problem, *Preprint Inst. Mat. Akad. Nauk. BSSR* **N.16**, Minsk, 1976.

[65] J.G. Kollias, Y. Manolopoulos and C.H. Papadimitriou, The optimum execution order of queries in linear storage, *Information Processing Letters* **36**, 1990, 141–145.

[66] I.M. Kunzevich, On the minimization of the value of a linear form on the set of all cycles of the length n. Part 1, *Izv. Akad. Nauk. BSSR, Ser. Fiz.-mat. Nauk* **2**, 1968, 25–30, (in Russian).

[67] I.M. Kunzevich, On the minimization of the value of a linear form on the set of all cycles of the length n. Part 2, *Izv. Akad. Nauk. BSSR, Ser. Fiz.-mat. Nauk* **3**, 1968, 12–21, (in Russian).

[68] E.L. Lawler, A solvable case of the traveling salesman problem, *Mathematical Programming* **1**, 1971, 267–269.

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

[70] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, *The Traveling Salesman Problem*, Wiley, Chichester, 1985.

[71] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problems, *Annals of Discrete Mathematics* **1**, 1977,343–362.

[72] S. Lin and B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, *Operations Research* **21**, 1973, 498–516.

[73] E.A. Medova-Dempster, The algebraic approach to combinatorial optimization, Ph.D. thesis, Technical University of Nova Scotia, 1990.

[74] M. Michalski, On a class of polynomially solvable traveling salesman problems, *Zastosowania Matematyki* **19**, 1987, 531–539.

[75] G. Monge, Mémoires sur la théorie des déblais et des remblais, in *Histoire de l'Academie Royale des Science, Année M. DCCLXXXI, avec les Mémoires de Mathématique et de Physique, pour la même Année, Tirés des Registres de cette Académie*, Paris, 1781, 666–704.

[76] D. Naddef, The binested inequalities for the symmetric traveling salesman polytope, to appear in *Mathematics of Operations Research.*

[77] D. Naddef and G. Rinaldi, The symmetric traveling salesman polytope and its graphical relaxation: Composition of valid inequalities, *Mathematical Programming* **51**, 1991, 359–400.

[78] D. Naddef and G. Rinaldi, The crown inequalities for the symmetric traveling salesman polytope, *Mathematics of Operations Research* **17**, 1992, 308–326.

[79] D. Naddef and G. Rinaldi, The graphical relaxation: A new framework for the symmetric traveling salesman polytope, *Mathematical Programming* **58**, 1993, 53–88.

[80] M.W. Padberg and M. Grötschel, Polyhedral computations, Chapter 9 in [70], 307–360.

[81] M.W. Padberg and T.-Y. Sung, A polynomial-time solution to Papadimitriou and Steiglitz's 'Traps', *Operations Research Letters* **7**, 1988, 117–125.

[82] C.H. Papadimitriou, The Euclidean traveling salesman problem is $\mathcal{NP}$-complete, *Theoretical Computer Science* **4**, 1977, 237–244.

[83] C.H. Papadimitriou, Lecture at the Maastricht Summerschool on Combinatorial Optimization, August 1993.

[84] C.H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

[85] C.H. Papadimitriou and P.C. Kanellakis, Flow-shop scheduling with limited temporary storage, *Journal of the ACM* **27**, 1980, 533–549.

[86] C.H. Papadimitriou and K. Steiglitz, Some examples of difficult traveling salesman problems, *Operations Research* **26**, 1978, 434–443.

[87] J.K. Park, A special case of the n-vertex traveling salesman problem that can be solved in $O(n)$ time, *Information Processing Letters* **40**, 1991, 247–254.

[88] R.D. Plante, The nozzle guide vane problem, *Operations Research* **36**, 1988, 18–33.

[89] R.D. Plante, T.J. Lowe and R. Chandrasekaran, The product traveling salesman problem: An application and solution heuristic, *Operations Research* **35**, 1987, 772–783.

[90] J. Plesnik, The $\mathcal{NP}$-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two, *Information Processing Letters* **8**, 1979, 199–201.

[91] F.P. Preparata and M.I. Shamos, *Computational Geometry – An Introduction*, Springer Verlag, New York, 1985.

[92] H.N. Psaraftis, A dynamic programming approach for sequencing groups of identical jobs, *Operations Research* **28**, 1980, 1347–1359.

[93] L.V. Quintas and F. Supnick, Extreme Hamiltonian Circuits: Resolution of the convex-odd case, *Proc. Amer. Math. Soc.* **15**, 1964, 454–459.

[94] L.V. Quintas and F. Supnick, On some properties of shortest Hamiltonian circuits, *American Mathematical Monthly* **72**, 1965, 977–980.

[95] L.V. Quintas and F. Supnick, Extreme Hamiltonian Circuits: Resolution of the convex-even case, *Proc. Amer. Math. Soc.* **16**, 1965, 1058–1061.

[96] L.V. Quintas and F. Supnick, Extrema in Space-Time, *Canadian Journal of Mathematics* **18**, 1966, 678–691.

[97] H.D. Ratliff and A.S. Rosenthal, Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem, *Operations Research* **31**, 1983, 507–521.

[98] S.S. Reddi and C.V. Ramamoorthy, On the flowshop sequencing problem with no-wait in process, *Journal of the Operational Research Society* **23**, 1972, 323-331.

[99] A.G. Reinhold, Some results on minimal covertex polygons, Manuscript written under the supervision of Fred Supnick, City College of New York, 1965.

[100] H. Röck, The three-machine no-wait flow-shop is NP-complete, *Journal of the ACM* **31**, 1984, 336–345.

[101] G. Rote, Two solvable cases of the traveling salesman problem, Ph.D. thesis, Technical University Graz, Graz (Austria), 1988.

[102] G. Rote, The N-line traveling salesman problem, *Networks* **22**, 1992, 91–108.

[103] M.I. Rubinshtein, On the symmetric traveling salesman problem, *Automation and Remote Control* **32**, 1971, 1453–1460 (translated from *Avtomatika i Telemekhanika*).

[104] D. Sanders, On extreme circuits, Ph.D. thesis, City University of New York, 1968.

[105] V.I. Sarvanov, On the minimization of a linear form on the set of all n-elements cycles, *Izv. Akad. Nauk. BSSR, Ser. Fiz.-mat. Nauk* **4**, 1976, 17–21, (in Russian).

[106] V.I. Sarvanov, Approximate solution of the problem of minimization of a linear form on the set of cyclic permutations, *Izv. Akad. Nauk. BSSR, Ser. Fiz.-mat. Nauk* **6**, 1977, 5–10, (in Russian).

[107] V.I. Sarvanov, On quadratic choice problems, Ph.D. thesis, Inst. Mat. AN BSSR, 1978, Minsk, (in Russian).

[108] V.I. Sarvanov, On the complexity of minimizing a linear form on a set of cyclic permutations, *Dokl. AN SSSR* **253**, 1980, 533–535, (in Russian), English Translation in *Soviet Math. Dokl.* **22**, 118–120.

[109] F. Supnick, Extreme Hamiltonian lines, *Annals of Math.* **66**, 1957, 179–201.

[110] D.A. Suprunenko, On the value of a linear form on a set of permutations, *Kibernetika*, Issue 2, 1968, 59–63, (in Russian), English translation in *Cybernetics* **4**, 1968, 48–51.

[111] D.A. Suprunenko, The traveling salesman problem, *Kibernetika*, Issue 5, 1975, 121–124, (in Russian), English translation in *Cybernetics* **11**, 1975, 799–803.

[112] D.A. Suprunenko, On the minimum of the value $v(A,t)$ on the set of cycles, *Preprint Inst. Mat. Akad. Nauk. BSSR* **N.14**, Minsk, 1976, (in Russian).

[113] W. Szwarc, The approximate solution of the traveling salesman problem, *Mathematica (RPR)* **1**, 1961, 183–191.

[114] R. Van Dal, Special cases of the traveling salesman problem, Ph.D. thesis, University of Groningen, The Netherlands, 1992.

[115] R. Van Dal, G. Tijssen, J.A.A. Van der Veen, C. Zamfirescu and T. Zamfirescu, Hamiltonian properties of Toeplitz graphs, to appear in *Discrete Mathematics*.

[116] R. Van Dal, J.A.A. Van der Veen and G. Sierksma, Small and large TSP: Two polynomial solvable cases of the traveling salesman problem, *European Journal of Operational Research* **69**, 1993, 107-120.

[117] J.A.A. Van der Veen, Solvable cases of the traveling salesman problem with various objective functions, Ph.D. thesis, University of Groningen, The Netherlands, 1992.

[118] J.A.A. Van der Veen, A new class of pyramidally solvable symmetric traveling salesman problems, *SIAM J. Discr. Math.* **7**, 1994, 585–592.

[119] J.A.A. Van der Veen, G. Sierksma and R. Van Dal, Pyramidal tours and the traveling salesman problem, *European Journal of Operational Research* **52**, 1991, 90–102.

[120] J.A.A. Van der Veen and R. Van Dal, Solvable cases of the no-wait flow-shop scheduling problem, *Journal of the Operational Research Society* **42**, 1991, 971–980.

[121] J.A.A. Van der Veen, G.J. Woeginger and S. Zhang, Sequencing jobs that require a common resource on a single machine: A solvable case of the TSP, SFB Report 21, 1995, Institut für Mathematik, TU Graz, Austria.

[122] J.A.A. Van der Veen and S. Zhang, Low complexity algorithms for sequencing jobs with a fixed number of job-classes, Nijenrode working paper series 1, 1995.

[123] E.A. Van Doorn, Connectivity of circulant digraphs, *Journal of Graph Theory* **10**, 1986, 9–14.

[124] Y.M. Volodarskij, E.Y. Gabovich and A.Y. Zacharin, A solvable case in discrete programming, *Izv. Akad. Nauk. SSSR, Ser. Techn. Kibernetika* **1**, 1976, 34–44, (in Russian), English translation in *Engineering Cybernetics* **14**, 1977, 23–32.

[125] R.H. Warren, Pyramidal tours for the traveling salesman problem, *Optimization* **29**, 1994, 81–87.

[126] D.A. Wismer, Solution of the flowshop scheduling problem with no intermediate queues, *Operations Research* **20**, 1972, 689–697.

[127] Q.F. Yang, R.E. Burkard, E. Çela and G.J. Woeginger, Hamiltonian cycles in circulant digraphs with two stripes, submitted to *Discrete Mathematics*.