Math. Program., Ser. A 87: 519–542 (2000)

Digital Object Identifier (DOI) 10.1007/s101070000135

Vladimir G. Deĭneko · Gerhard J. Woeginger

A study of exponential neighborhoods for the Travelling Salesman Problem and for the Quadratic Assignment Problem

Received: September 5, 1997 / Accepted: November 15, 1999
Published online February 23, 2000 –

Abstract. This paper deals with *exponential neighborhoods* for combinatorial optimization problems. Exponential neighborhoods are large sets of feasible solutions whose size grows exponentially with the input length. We are especially interested in exponential neighborhoods over which the TSP (respectively, the QAP) can be solved in polynomial time, and we investigate combinatorial and algorithmical questions related to such neighborhoods.

First, we perform a careful study of exponential neighborhoods for the TSP. We investigate neighborhoods that can be defined in a simple way via assignments, matchings in bipartite graphs, partial orders, trees and other combinatorial structures. We identify several properties of these combinatorial structures that lead to polynomial time optimization algorithms, and we also provide variants that slightly violate these properties and lead to NP-complete optimization problems. Whereas it is relatively easy to find exponential neighborhoods over which the TSP can be solved in polynomial time, the corresponding situation for the QAP looks pretty hopeless: *Every* exponential neighborhood that is considered in this paper *provably* leads to an NP-complete optimization problem for the QAP.

Key words. neighborhood – local search – search problem – Travelling Salesman Problem – Quadratic Assignment Problem – polynomial time algorithm – NP-completeness – combinatorial optimization

1. Introduction

In this paper, we deal with *optimization problems* of the following form: An underlying *cost function* $\textsc{Cost} : S_n \rightarrow \mathbb{R}$ assigns to every permutation a non-negative cost (throughout the paper, we will denote by S_n the set of all permutations of $\{1, 2, \ldots, n\}$). The goal of the optimization problem is to identify a permutation π at which the function $\textsc{Cost}(\cdot)$ takes its minimum. Two well-known and fundamental problems of this type in combinatorial optimization are the *Travelling Salesman Problem* (TSP) and the *Quadratic Assignment Problem* (QAP).

In the TSP, the objective is to find for a given $n \times n$ distance matrix $C = (c_{ij})$ a permutation $\pi \in S_n$ that minimizes the cost function

$$\textsc{Tsp}(C, \pi) \doteq \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)} + c_{\pi(n)\pi(1)}. \tag{1}$$

V.G. Deĭneko: Warwick Business School, The University of Warwick, Coventry CV4 7AL, United Kingdom, e-mail: orsvd@wbs.warwick.ac.uk

G.J. Woeginger: TU Graz, Institut für Mathematik, Steyrergasse 30, A-8010 Graz, Austria, e-mail: gwoegi@opt.math.tu-graz.ac.at.
Supported by the START program Y43-MAT of the Austrian Ministry of Science

Mathematics Subject Classification (1991): 90C27, 90C35, 68Q25

When dealing with the TSP, the permutations over $\{1, 2, \ldots, n\}$ will also be called *tours*, the elements of $\{1, 2, \ldots, n\}$ will be called *cities*, and two consecutive cities $\pi(i)$ and $\pi(i+1)$ in the tour will be said to form an *edge* of the tour. In other words, the minimization of (1) means that the travelling salesman has to visit the cities 1 to n in some order, that in the end he has to return to the city from which he had started, and that he has to minimize the total travel length while doing this.

In the QAP in Koopmans-Beckmann form (Koopmans and Beckmann [22]), the objective is to find for two given $n \times n$ matrices with real entries $A = (a_{ij})$ and $B = (b_{ij})$, $1 \leq i, j \leq n$, a permutation $\pi \in S_n$ that minimizes the cost function

$$\text{QAP}(A, B, \pi) \doteq \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij}. \tag{2}$$

We refer the reader to the book by Lawler, Lenstra, Rinnooy Kan and Shmoys [25] for more information on the TSP, and to the survey papers by Lawler [23,24], by Burkard [5], and by Pardalos, Rendl, and Wolkowicz [31] for more information on the QAP.

Neighborhoods. A *neighborhood* $\mathcal{N}$ is a sequence $\langle \mathcal{N}_n \rangle_{n \geq 1}$ of sets of permutations with $\mathcal{N}_n \subseteq S_n$. A neighborhood $\mathcal{N}$ assigns to every permutation $\pi \in S_n$ the corresponding neighborhood $\mathcal{N}_n(\pi) = \{\pi \circ \phi \mid \phi \in \mathcal{N}_n\}$. Note that $\mathcal{N}_n = \mathcal{N}_n(\text{id}_n)$ holds, where id_n is the identity permutation in S_n. As an example, consider the K-OPT neighborhood that is the classical and probably best investigated neighborhood for the TSP: The set $\text{K-OPT}_n \subseteq S_n$ consists of all tours that differ in at most k edges from the identity permutation id_n. Consequently, for any $\pi \in S_n$ the neighborhood $\text{K-OPT}_n(\pi)$ consists of all tours that differ in at most k edges from π.

This is the time for our first disclaimer: The above definition of neighborhood does by no means cover *all* concepts of a neighborhood that have been investigated in the literature. The main drawback is that in our definition, a neighborhood is a strictly static concept that only depends on the current permutation π. In the literature, however, neighborhoods are often dynamic structures that are time-dependent (e.g. Simulated Annealing depends on the current temperature respectively current time), history dependent (e.g. Tabu Search depends on the constructed tabu list) or data-depend (e.g. subtour patching depends on the optimal assignment). Nevertheless, we believe that our definition of neighborhood already captures a good deal of the difficulty and of the complexity of this concept.

Local search. For combinatorial optimization problems of the type defined above, one can build around each neighborhood $\mathcal{N}$ a corresponding *local search* procedure as follows:

```
1  Initialize π
2  Loop
3      Compute a permutation π_new ∈ N(π) that minimizes COST(π_new)
4      If COST(π_new) = COST(π) then
5              Output π and stop
6      Else set π := π_new and continue
7  Forever
```

Since there is only a finite number of possible permutations in S_n, the procedure will eventually halt. Of course, the critical part of this generic local search procedure is line 3: How do we find the permutation π_{new}? If the size of $\mathcal{N}(\pi)$ is relatively small (i.e. constant or polynomial in the input size), then we can simply search through $\mathcal{N}(\pi)$ by enumerating all of its elements. On the other hand, if the size of $\mathcal{N}(\pi)$ indeed is small, then every iteration of the loop only considers a small number of potential solutions, and this may lead to a very long sequence of search steps. Hence, it seems natural to require that the neighborhood $\mathcal{N}$ fulfills the following two properties:

(N1) The neighborhood $\mathcal{N}$ is exponentially large, i.e. the cardinalities $|\mathcal{N}_n|$ grow exponentially with n.

(N2) One can compute in polynomial time for every permutation $\pi \in S_n$, a permutation in the neighborhood $\mathcal{N}_n(\pi)$ that minimizes the function COST(·).

If a neighborhood fulfills (N1) then we will call it an *exponential neighborhood*, and if it fulfills (N2) then we will say that it *can be searched in polynomial time with respect to function* COST(·). In this paper we investigate a variety of combinatorial and algorithmical questions that are related to neighborhoods fulfilling properties (N1) and (N2) for the TSP and for the QAP.

Now is the time for our second disclaimer: We do *not* claim that every useful local search algorithm has to be based on a neighborhood that fulfills the properties (N1) and (N2). We only state that these properties are natural and hence are worth investigating. In fact, most of the neighborhoods that have been studied till now and that are successfully used in practice do *not fulfill* the property (N1), and most of them do not exploit property (N2). Consider e.g. the K-OPT neighborhood that has been mentioned before. The K-OPT neighborhood was introduced by Croes [10] and by Lin [26], and its variants for $k = 2$ and $k = 3$, 2-OPT and 3-OPT, perform perfectly well in practical applications. However, the K-OPT neighborhood only contains $O(n^k)$ permutations, and hence its size is small and violates property (N1). Moreover, in practical applications one in general does not search for the *best* neighbor in K-OPT with minimum cost, but only for some *better* neighbor in K-OPT with smaller cost. Hence, property (N2) is not strictly needed in this approach.

Comparison of neighborhoods. In 1981, Sarvanov and Doroshko [34] designed a simple neighborhood called ASSIGN that contains $(n/2)!$ elements and that can be searched with respect to the TSP in $O(n^3)$ time. Let us compare neighborhood ASSIGN to the 3-OPT-neigborhood: Searching both neighborhoods with respect to the TSP takes roughly $O(n^3)$ time. However, ASSIGN contains $(n/2)!$ elements, whereas 3-OPT only contains $O(n^3)$ elements. Hence, a local search procedure based on 3-OPT would have to go through an exponential number of steps, just to compare the current permutation against the same number of permutations as a local search procedure based on neighborhood ASSIGN does in a single step! (However, we would not advice the reader to incorporate this neighborhood in its current form into a local search procedure: it would always get stuck in a local optimum after the first step).

In this paper, we define and study several neighborhoods of a similar flavor that fulfill property (N1) and property (N2) for the TSP. We are interested in identifying the borderline between neighborhoods that can and neighborhoods that can not be searched

in polynomial time. We look at different ways for defining neighborhoods with nice combinatorial structure via assignments, matchings in bipartite graphs, partial orders, and trees. Moreover, we will provide some weak evidence, that unless $P = NP$ the QAP does not possess neighborhoods that simultaneously fulfill (N1) and (N2).

Organization of the paper. Section 2 presents a short discussion of some results on local search for the TSP from the literature. Section 3 introduces some notation and basic definitions. Section 4 deals with exponential neighborhoods for the TSP (the proofs of all theorems in this section can be found in Appendix A) and Sect. 5 deals with exponential neighborhoods for the QAP (the proofs of the theorems are in Appendix B). The discussion in Sect. 6 closes the paper.

2. Some well-known results on local search for the TSP

Extensive simulations by Johnson and McGeoch [19] suggest that 2-OPT and 3-OPT are the champion local search algorithms for the Euclidean TSP. In the Euclidean plane with the Euclidean metric, the expected number of steps to reach a local minimum for the 2-OPT-neighborhood is polynomially bounded by $O(n^{10} \log n)$ as shown by Chandra, Karloff and Tovey [8] (cf. also Kern [20]). However, on randomly generated non-Euclidean instances K-OPT in general performs poorly. Moreover, Papadimitriou and Steiglitz [29] design specific instances that have a single optimum tour but exponentially many tours that are local optimums with respect to 2-OPT and whose length is an exponential factor away from the optimum.

Papadimitriou and Steiglitz [28] show that unless $P = NP$, no local search algorithm that fulfills property (N2) for the TSP can be exact (a local search algorithm is called *exact* if every local optimum is also a global optimum). For an extensive discussion of local search algorithms, see Chapt. 19 in the book by Papadimitriou and Steiglitz [30], the case study by Johnson and McGeoch [19], and the book by Reinelt [32]. Finally, we want to mention that the overall shortest TSP tour can be found in exponential-time $O(n^2 2^n)$ by the well-known dynamic programming algorithm of Held and Karp [17].

3. Basic definitions

The set of all permutations over $\{1, 2, \ldots, n\}$ is denoted by S_n. With a slight abuse of notation, we will sometimes denote *any* sequence of pairwise distinct cities over $\{1, 2, \ldots, n\}$ as a permutation. For a permutation π, we adopt the notation $\pi = \langle x_1, x_2, \ldots, x_n \rangle$ as abbreviation for "$\pi(i) = x_i$ for $1 \leq i \leq n$". The *identity permutation* $\langle 1, 2, 3 \ldots, n \rangle$ in S_n is denoted by id_n. For every permutation π, we define its *reverse* permutation π^- by $\pi^-(i) = \pi(n - i + 1)$ for $1 \leq i \leq n$. A permutation π is called a *cyclic shift* or a *rotation* if there exists an index $k \in \{1, \ldots, n\}$ such that $\pi = \langle k, k+1, \ldots, n, 1, \ldots, k-1 \rangle$ holds.

For two permutations $\pi_1 = \langle x_1, \ldots, x_n \rangle$ and $\pi_2 = \langle y_1, \ldots, y_m \rangle$, their *concatenation* $\pi_1 \star \pi_2$ is the permutation $\langle z_1, \ldots, z_{n+m} \rangle$, where $z_i = x_i$ for $1 \leq i \leq n$ and $z_{n+j} = y_j$ for $1 \leq j \leq m$. For two sets Π_1 and Π_2 of permutations, we define

$$\Pi_1 \star \Pi_2 \;=\; \{\pi_1 \star \pi_2 \mid \pi_1 \in \Pi_1,\ \pi_2 \in \Pi_2\}\,. \tag{3}$$

Clearly, the operation '$\star$' is an associative operation on sets of permutations. The *composition* $\pi \circ \phi$ of two permutations $\pi, \psi \in S_n$ is defined via $(\pi \circ \phi)(i) = \pi(\phi(i))$.

For a neighborhood $\mathcal{N}_n(\pi)$ of a permutation $\pi \in S_n$, we will often simply write "$\mathcal{N}(\pi)$" and will omit the index, if the dimension is clear from the context. Moreover, we will define most of our neighborhoods directly for all permutations $\pi \in S_n$ and not take the roundabout route to define it via the identity permutation.

To keep the notation as simple as possible, we will often use expressions of the form $(n/2)!$, or $\sqrt{n}$ as integer expressions. In doing this, we assume that the occurring fractions are always rounded up or rounded down to integers in an appropriate way.

4. Exponential neighborhoods: the TSP

This section discusses combinatorial aspects and complexity aspects of optimizing over strongly structured exponential neighborhoods for the TSP. Essentially, we will deal with three groups of neighborhoods: Neighborhoods that are based on assignments and matchings (Sect. 4.1), neighborhoods that are based on partially ordered sets (Sect. 4.2), and neighborhoods that are based on tree structures (Sect. 4.3). Moreover, Sect. 4.4 discusses some other, 'unclassified' approaches to exponential neighborhoods, and Sect. 4.5 draws some conclusions and poses open problems.

The proofs of all theorems are to be found in Appendix A.

4.1. Neighborhoods that are based on assignments and matchings

Let us start our investigations with a simple neighborhood called ASSIGN that has been introduced in 1981 by Sarvanov and Doroshko [34]. Formally,

$$\text{ASSIGN}_n = \{\phi \in S_n \mid \phi(2i-1) = 2i-1 \text{ for all } i = 1, \ldots, \lceil \tfrac{n}{2} \rceil\}. \tag{4}$$

In other words, a permutation ϕ is in ASSIGN(π) if and only if one can obtain ϕ from π by fixing all cities in the odd positions at their places, then removing the cities in the even positions and finally reinserting them in arbitrary order into the empty positions. It can easily be seen that computing the cheapest tour in ASSIGN(π) corresponds to solving a standard assignment problem on an $\frac{n}{2} \times \frac{n}{2}$ cost matrix. Since this assignment problem can be solved in $O(n^3)$ time by standard methods (see e.g. Papadimitriou and Steiglitz [30]), we arrive at the following result.

Proposition 1. *(Sarvanov and Doroshko [34])*
The neighborhood ASSIGN_n *contains* $(n/2)!$ *permutations and can be searched in* $O(n^3)$ *time with respect to the TSP.*

□

There are several natural generalizations of neighborhood ASSIGN that immediately come to one's mind:

- E.g. one may consider the set of permutations that result from arbitrarily permuting the cities within the even places and simultaneously permuting the cities within the odd places. This yields the neighborhood ASSIGN-EO that contains $(n/2)!^2$ permutations, a much larger neighborhood than ASSIGN (the "EO" in ASSIGN-EO stands for EVEN and for ODD).
- Another generalization arises from dividing the cities into three classes, instead of only two classes. Hence, let us define that a permutation ϕ is in the ASSIGN-MOD3-neighborhood of π, if and only if π and ϕ agree in the positions whose numbers are divisible by 3, whereas the cities in the positions that are $\equiv 1 (\mathrm{mod}\ 3)$ are permuted arbitrarily within their places, and the cities in the positions that are $\equiv 2 (\mathrm{mod}\ 3)$ are also permuted arbitrarily within their places. ASSIGN-MOD3 contains $(n/3)!^2$ permutations.
- Finally, we define that a permutation ϕ is in the ASSIGN-2/3-neighborhood of π, if and only if π and ϕ agree in the positions that are divisible by 3, whereas the cities in the positions that are not divisible by 3 may be permuted arbitrarily. ASSIGN-2/3 contains $(2n/3)!$ permutations.

The three generalizations ASSIGN-EO, ASSIGN-MOD3, and ASSIGN-2/3 all have much larger cardinalities than neighborhood ASSIGN and hence are a clear improvement over ASSIGN with respect to property (N1). However, unless $P = NP$ holds, they can not fulfill property (N2) with respect to the TSP as the following theorem demonstrates.

Theorem 1. *It is NP-hard to minimize for a given input matrix $C = (c_{ij})$ the function* $\mathrm{TSP}(C, \pi)$ *over the set of*

(i) *permutations* $\pi \in$ ASSIGN-EO,
(ii) *permutations* $\pi \in$ ASSIGN-MOD3,
(iii) *permutations* $\pi \in$ ASSIGN-2/3.

□

An observation of Gutin [16] yields the following slight improvement over neighborhood ASSIGN: For $\pi \in S_n$, call the first $n/2 + \sqrt{n/8}$ cities in π the *fixed* cities and call the last $n/2 - \sqrt{n/8}$ cities the *moving* cities. A permutation ϕ is in ASSIGN-GUTIN(π) if and only if one can get ϕ from π by first removing the moving cities from π and then reinserting them arbitrarily between the fixed cities in such a way that between any pair of consecutive fixed cities, at most one moving city is inserted. Again we observe that computing the cheapest travelling salesman tour in ASSIGN-GUTIN(π) can be done in $O(n^3)$ time by solving an assignment problem. Simple calculations reveal that the size of neighborhood ASSIGN-GUTIN is $\Omega((n/2)!(1+\varepsilon)^{\sqrt{n}})$, where $\varepsilon > 0$ is some small, positive real that does not depend on n.

Another way of representing neighborhoods that is closely related to assignments, is the representation via matchings in a bipartite graph: Let $H = (I \cup I', F)$ be a bipartite graph with $F \subseteq I \times I'$ where $I = \{1, \ldots, n\}$ corresponds to the set of cities and where $I' = \{1', \ldots, n'\}$ corresponds to the set of positions in some tour. For $i \in I$, we denote by $\Gamma_H(i)$ the set of nodes in I' that are adjacent to i. We define the neighborhood

$$\mathrm{MATCHING}_H = \{\phi \in S_n \mid (\phi(i), i') \in F \text{ for all } i = 1, \ldots, n\}. \tag{5}$$

In other words, the positions in $\Gamma_H(i)$ tell the potential positions of city i in a neighboring permutation. E.g. in this model, the neighborhood ASSIGN may be represented via a bipartite graph that contains all the edges (i, i') with odd i and all the edges (i, j') with even i and j. See Fig. 1 for an illustration.

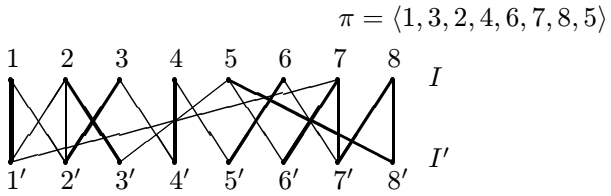

Fig. 1. A bipartite graph $H = (I \cup I', F)$. The bold solid lines form a matching that encodes the depicted permutation π in MATCHING_H

Define the *extension* ext(i) of a node $i \in I$ as the value $\max\{j | j' \in \Gamma_H(i)\} - \min\{j | j' \in \Gamma_H(i)\}$, and define the extension ext(H) of the graph H as $\max_{i \in I} \text{ext}(i)$. Define the *out-degree* out-deg(H) of the graph H as the maximum $|\Gamma_H(i)|$ over all $i \in I$. Balas and Simonetti [3] proved the following result with the help of dynamic programming.

Proposition 2. *(Balas and Simonetti [3])*
For a bipartite graph H with extension $\text{ext}(H) = d$*, the neighborhood* MATCHING_H *can be searched in $O(4^d n)$ time with respect to the TSP.*

□

Hence, if the extension of the graph H is small, i.e. $d = O(\log n)$, then the neighborhood MATCHING_H is easy to search. Perhaps somewhat surprisingly, an analogous result does *not* hold for graphs H with small out-degree:

Theorem 2. *It is NP-hard to minimize for a given input matrix $C = (c_{ij})$ and for a given bipartite graph H with* $\text{out-deg}(H) = 2$*, the function* $\text{TSP}(C, \pi)$ *over the set of all permutations* $\pi \in \text{MATCHING}_H$.

□

4.2. Neighborhoods that are based on partial orders

In this section, we consider a *partial order* $\preceq$ on the set $I = \{1, \ldots, n\}$ of cities. Two cities i and j are called *comparable* if $i \preceq j$ or $j \preceq i$ holds, and they are called *incomparable*, otherwise. An *anti-chain* of a partial order is a set of pairwise incomparable elements. The *Dilworth number* of a partial order is the cardinality of its largest anti-chain. A permutation $\pi \in S_n$ is a *linear extension* of the order $\preceq$ if and only if $i \preceq j$ implies $\pi(i) \leq \pi(j)$ for all $1 \leq i, j \leq n$. By $\text{LINEXT}(\preceq)$ we denote the set of all linear extensions of the order $\preceq$.

Theorem 3. *For a given partial order $\preceq$ which is defined on the set of cities and which has Dilworth number $d \geq 2$, and for a given distance matrix $C = (c_{ij})$, the function* $\textsc{Tsp}(C, \pi)$ *can be minimized over the permutations $\pi \in \textsc{LinExt}(\preceq)$ in $O(n^d)$ time.*

□

Balas and Simonetti [3] deal with a special case of finding the shortest tour in a neighborhood $\textsc{LinExt}(\preceq)$: In their special case, the partial order is based on a parameter k, and $i \preceq j$ holds for two cities i and j if and only if $i + k \leq j$. It is easy to see that the Dilworth number of the Balas-Simonetti order equals k, and so our Theorem 3 implies the existence of an $O(n^k)$ solution algorithm. However, besides having a small Dilworth number, the Balas-Simonetti order also fulfills other strong combinatorial properties: It only has $O(2^k n)$ distinct anti-chains, and these anti-chains can be found and enumerated efficiently, without much additional overhead. A dynamic programming approach that is based on these anti-chains yields an $O(k^2 2^k n)$ solution algorithm.

In the literature (see e.g. the survey article by Möhring [27]), one can find many special classes of computationally tractable partial orders: interval orders, series-parallel orders, N-free orders, orders of bounded height, two-dimensional orders, orders of bounded dimension, etc. We note that all these classes contain the empty partial order as a special case. Since the set of linear extensions of the empty partial order equals S_n, these classes in general will not lead to neighborhoods that can be searched in polynomial time with respect to the TSP.

4.3. Neighborhoods that are based on tree structures

A *permutation tree* T over the universal set of cities $I = \{1, \ldots, n\}$ is a rooted, ordered tree whose *leaves* are pairwise distinct elements of I. The set of leaves in T is denoted by $\textsc{leaf}(T)$. Every *interior* node has at least two sons; an interior node v with d sons is labeled with a non-empty set $\Psi(v) \subseteq S_d$ of permutations. With every permutation tree T, we associate a set $\textsc{Tree}(T)$ of permutations of $\textsc{leaf}(T)$ as follows: If T consists of only a single leaf $u \in I$, then $\textsc{Tree}(T) = \langle u \rangle$. Otherwise, let $v_1, \ldots, v_d$ denote the sons of the root r of T, ordered from left to right, and let T_i denote the maximal subtrees rooted at v_i, $1 \leq i \leq d$. Define

$$\textsc{Tree}(T) = \bigcup_{\psi \in \Psi(v)} \textsc{Tree}(T_{\psi(1)}) \star \textsc{Tree}(T_{\psi(2)}) \star \cdots \star \textsc{Tree}(T_{\psi(d)}). \qquad (6)$$

A well-known special case of permutation trees are the *PQ-trees* introduced by Booth and Lueker [4]: A PQ-tree is a permutation tree whose interior nodes all are either *P-nodes* or *Q-nodes*; for every P-node v with d sons, $\Psi(v) = S_d$ holds, and for every Q-node v with d sons, $\Psi(v) = \{\mathrm{id}_d, \mathrm{id}_d^-\}$ holds. Burkard, Deĭneko, Woeginger [6] investigated the problem of minimizing the function $\textsc{Tsp}(C, \pi)$ over the permutations represented by a given PQ-tree. A simple modification of their approach yields the following theorem.

Theorem 4. *For a permutation tree T with $f = \max_{v \in T} |\Psi(v)|$, and for a distance matrix $C = (c_{ij})$, the shortest TSP tour for matrix C contained in* $\textsc{Tree}(T)$ *can be computed in $O(fn^4)$ overall time.*

□

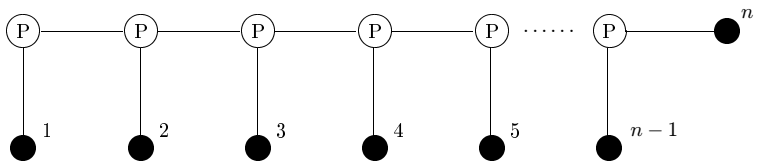

Fig. 2. A permutation tree that represents the set of pyramidal tours in S_n. The root is the leftmost P-node. All interior nodes v have $\Psi(v) = S_2$

The classical example for a neighborhood that can be represented via permutation trees is the neighborhood PYRAMID that is formed by the *pyramidal permutations* (cf. e.g. Gilmore, Lawler, and Shmoys [14]).

$$\text{PYRAMID}_n = \{\phi = \langle i_1, i_2, \ldots, i_k, n, j_1, \ldots, j_{n-k-1}\rangle \mid k \geq 0,\ i_1 < i_2 < \cdots < i_k,\ j_1 > j_2 > \cdots > j_{n-k-1}\}.$$

In other words, a permutation is pyramidal if it first goes through the cities in increasing order until city n is reached, and then goes through the remaining cities in decreasing order. For example, the permutation $\langle 1, 2, 5, 7, 8, 9, 6, 4, 3\rangle$ is pyramidal, whereas the permutation $\langle 1, 8, 3, 4, 2, 5, 6, 7\rangle$ is not. It is well-known that the neighborhood PYRAMID_n contains 2^{n-1} permutations and that minimizing the function $\text{TSP}(C, \pi)$ over the set PYRAMID_n can be done in $O(n^2)$ time by dynamic programming (Klyaus [21] or Gilmore, Lawler, and Shmoys [14]).

In the setting of permutation trees, we observe that the neighborhood PYRAMID can be represented through a permutation tree of the form as depicted in Fig. 2.

Apparently, Sarvanov and Doroshko [33] were the first to apply the set of pyramidal permutations as an exponential neighborhood in a local search algorithm for the TSP. Carlier and Villon [7] investigated the neighborhood PYRAMID-CV that consists of all permutations of the form $\pi \circ \phi$ where π is a pyramidal permutation and where ϕ is a rotation. In their computational experiments on the resulting local search algorithm, Carlier and Villon observed that their heuristic performs much better than 2-OPT, and that it even is competitive with K-OPT: Every local optimum for neighborhood PYRAMID-CV is also a local optimum for 2-OPT. Moreover, our computational experiments seem to indicate that most of the times when the local search heuristic gets stuck in a local optimum, 3-OPT is not able to improve on this local optimum. The following theorem lends further support to these observations.

Theorem 5. *For symmetric distance matrices, every permutation $\pi \in S_n$ fulfills the equation*

$$\frac{\mid \text{PYRAMID-CV}(\pi) \cap \text{3-OPT}(\pi) \mid}{\mid \text{3-OPT}(\pi) \mid} = \frac{3}{4} + o\left(\frac{1}{n}\right) \tag{7}$$

In other words, neighborhood PYRAMID-CV *covers at least 75% of the permutations contained in neighborhood* 3-OPT.

□

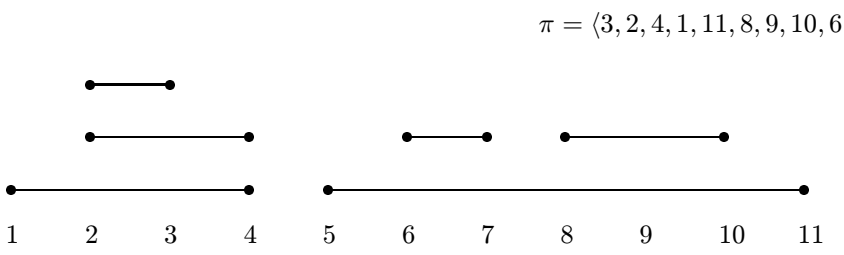

Fig. 3. Reversing all the intervals yields the twisted permutation

Another example for a neighborhood that can be represented via permutation trees are the *twisted sequences*. Aurenhammer [2] calls a permutation π a twisted sequence if and only if there exists a permutation tree T such that (i) $\pi \in \text{TREE}(T)$, (ii) $\text{id} \in \text{TREE}(T)$, and (iii) every interior node v with d sons has $\Psi(v) = \{\text{id}_d, \overline{\text{id}_d}\}$; note that this definition does not depend on a specific permutation tree T. Another, equivalent way of defining twisted sequences is as follows: Start with the identity permutation $\langle 1, 2, \ldots, n\rangle$ and choose a set of intervals over $[1, \ldots, n]$ such that for every pair of intervals either one of them contains the other one, or the two intervals are disjoint. Then reverse (= twist) for every interval the order of its elements. A permutation is a twisted sequence if and only if it can be derived from the identity permutation via such a reversal process. Observe that e.g. $\langle 3, 2, 4, 1\rangle$ is a twisted sequence, whereas $\langle 1, 3, 5, 2, 4, 6\rangle$ is not. See Fig. 3 for another illustration.

We denote by $\text{TWISTED} \subseteq S_n$ the set that consists of all twisted sequences. It is easy to see that for every n, the neighborhood TWISTED_n contains at least $\Omega(2^n)$ and at most $O(6^n)$ permutations. Moreover, twisted sequences can be recognized in $O(n)$ time (Aurenhammer [2]).

Theorem 6. *For a given distance matrix $C = (c_{ij})$, the function $\text{TSP}(C, \pi)$ can be minimized over the permutations $\pi \in \text{TWISTED}$ in polynomial time $O(n^7)$.*

□

4.4. Other neighborhoods for the TSP

In this section we discuss several approaches to exponential neighborhoods that did not fit into the framework of the preceding three sections: a geometric approach via Jordan curves, a graph theoretic approach via Hamiltonian cycles in edge-weighted graphs, and a combinatorial approach via transpositions.

Let us turn to so-called *Jordan sequences* or *Jordan permutations*, a geometrically motivated concept. A permutation $\pi = \langle x_1, \ldots, x_n\rangle \in S_n$ is called a Jordan permutation if and only if there exists a simple, non-selfintersecting curve in the Euclidean plane (i.e. a Jordan curve) that (i) goes through the points $(x_1, 0), (x_2, 0), \ldots, (x_n, 0)$ in precisely this order and (ii) does not intersect the x-axis in any other points. See Fig. 4 for an illustration.

This type of permutation can be recognized in $O(n)$ time, see Hoffmann, Mehlhorn, Rosenstiehl, and Tarjan [18]. We denote by $\text{JORDAN} \subseteq S_n$ the set that consists of all

Jordan permutations. For every n, the neighborhood JORDAN_n contains at least $\Omega(2^n)$ and at most $O(8^n)$ permutations [18].

Open Problem 1. *What is the computational complexity of minimizing for a given input matrix $C = (c_{ij})$ the function $\text{TSP}(C, \pi)$ over the set of permutations $\pi \in \text{JORDAN}$?*

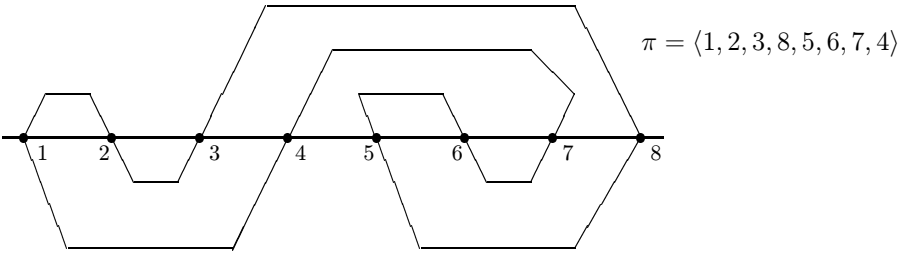

Fig. 4. A Jordan curve that intersects the x-axis in 8 points and one of the corresponding Jordan permutations

Another approach to strongly structured exponential neighborhoods for the TSP is via weighted Hamiltonian cycles in graphs. Let $\mathcal{H}$ be a class of specially structured graphs on which the weighted Hamiltonian cycle problem is solvable in polynomial time. For a graph $H \in \mathcal{H}$ on n vertices that are numbered $1, \ldots, n$, let $\text{HAMCYC}(H) \subseteq S_n$ denote the set of all permutations $\pi \in S_n$ for which the sequence $\pi(1), \pi(2), \ldots, \pi(n)$ forms a Hamiltonian cycle in H.

There are quite a few such graph classes $\mathcal{H}$ that contain graphs with an exponential number of Hamiltonian cycles. Cornuejols, Naddef, and Pulleyblank [9] show that a minimum weight Hamiltonian cycle in a Halin graph can be computed in polynomial time. Glover and Punnen [15] construct a class of graphs on n vertices for which one can compute in $O(n)$ time a weighted Hamiltonian cycle whose weight is less or equal to the weight of $\Theta(12^{n/3})$ other Hamiltonian cycles in the graph. The results of Fonlupt and Nachef [12] can be applied in a similar way to yield exponential neighborhoods.

Finally, we come to sets of permutations that are defined via *transpositions*. It is well-known that every permutation $\pi \in S_n$ can be factored into a sequence of cycles of length two, so-called transpositions. This means that $\pi = (i_\ell, j_\ell)(i_{\ell-1}, j_{\ell-1}) \cdots (i_1, j_1)$ holds with the following interpretation: If one starts with the identity permutation id_n, and first swaps element i_1 with element j_1, then swaps the elements i_2 and j_2, and so on, then one will finally end up with permutation π. The sequence in which these swaps are performed is essential, and this factorization is not unique. However, for every such factorization of π into transpositions, the number of transpositions in the factorization has the same parity. The set $A_n \subset S_n$ consists of the permutations that can be factored into an even number of transpositions. It is easy to see that $|A_n| = \frac{1}{2}n!$ holds, and thus A_n forms an exponential neighborhood. Of course, it is NP-hard to search A_n with respect to the TSP.

4.5. Some conclusions on neighborhoods for the TSP

To summarize, in the preceding sections we have constructed quite a few exponential neighborhood that can be searched in polynomial time with respect to the TSP. Most of these neighborhoods had sizes $\Theta(\gamma^n)$ for some $\gamma > 1$, and just a few of them (variants of neighborhood ASSIGN) were of larger size. Our results seem to indicate that it will be difficult to obtain a substantial improvement over neighborhood ASSIGN. Hence, we formulate the following two open questions.

Open Problem 2. *Does there exist an exponential neighborhood $\mathcal{N}$ that can be searched in polynomial time with respect to the TSP and that fulfills for all sufficiently large n*

(a) $|\mathcal{N}_n| \geq (\alpha n)!$ *for some fixed* $\alpha > \frac{1}{2}$*?*

(b) $|\mathcal{N}_n| \geq \beta \cdot n!$ *for some fixed* $\beta > 0$*?*

We conjecture that the first question has answer YES, and that the second question has answer NO (of course, under the assumption that $P \neq NP$).

Theorem 7. *Let $\mathcal{N}$ be an exponential neighborhood that can be searched in time $f(n)$ with respect to the TSP. Then $|\mathcal{N}_n| \leq (\frac{2}{n} f(n))^n$ holds for all n.*

□

An immediate consequence of this theorem is that any exponential neighborhood that fulfills the conditions in Open Problem 2.(a) and (b) cannot be searched in linear time $O(n)$ with respect to the TSP. This also answers a question of Gutin [16].

5. Exponential neighborhoods: the QAP

In this section, we go once again through the list of neighborhoods that were defined in the preceding section for the TSP, and we discuss and investigate them with respect to the QAP. The outcome of these investigations is disastrous: Unless $P = NP$, all known exponential neighborhoods do not fulfill property (N2), i.e. can not be searched in polynomial time with respect to the QAP. The proofs of the theorems can be found in Appendix B.

First, let us recall that the TSP is a special case of the QAP. Hence, all NP-completeness results stated in Sect. 4 immediately carry over to the QAP. Next, let us introduce yet another neighborhood, called TWIN, that will turn out to be useful in this section. The neighborhood TWIN is only defined for even numbers n.

$$\text{TWIN}_n = \{\pi \in S_n \mid \{\pi(2i-1), \pi(2i)\} = \{2i-1, 2i\} \text{ for } i = 1, \ldots, n/2\}. \quad (8)$$

Whereas it is easy to search TWIN in $O(n)$ time with respect to the TSP, we have the following fundamental negative result for the QAP.

Theorem 8. *The problem of minimizing the function* QAP(A, B, π) *over all permutations $\pi \in$* TWIN *for given input matrices $A = (a_{ij})$ and $B = (b_{ij})$ is NP-hard, even if A and B both are 0-1-matrices.*

□

Let us start our discussion with the neighborhoods based on assignments and matchings as introduced in Sect. 4.1. In this area, the known polynomial time results for the TSP are the result on neighborhood ASSIGN in Proposition 1, and the result on neighborhood MATCHING$_H$ with bipartite graphs of constant extension in Proposition 2. First, the proof of Theorem 9 below shows that neighborhood ASSIGN cannot be searched in polynomial time with respect to the QAP. Secondly, for bipartite graphs H with $\text{ext}(H) = 1$, the neighborhood MATCHING$_H$ is uninteresting since it contains at most one permutation. Finally, searching the neighborhood MATCHING$_H$ with respect to the QAP is an NP-complete problem for bipartite graphs H with $\text{ext}(H) = 2$: The neighborhood TWIN can be represented as MATCHING$_H$ with a graph H with $\text{ext}(H) = 2$.

Theorem 9. *It is NP-hard to minimize for given input matrices $A = (a_{ij})$ and $B = (b_{ij})$ the function* QAP(A, B, π) *over the permutations $\pi \in$* ASSIGN.

□

Next, let us turn to neighborhoods based on partial orders as discussed in Sect. 4.2. The only polynomial time result that we had for the TSP in this area is the result on neighborhood LINEXT$(\preceq)$ for partial orders with bounded Dilworth number in Theorem 3. However, partial orders $\preceq$ with Dilworth number 1 are total orders, and for them the neighborhood LINEXT$(\preceq)$ only contains a single permutation. On the other hand, the neighborhood TWIN can be represented as LINEXT$(\preceq)$ of a partial order $\preceq$ with Dilworth number $d = 2$: Let $i \preceq j$ if and only if $\lfloor i/2 \rfloor \leq \lfloor j/2 \rfloor$. Hence by Theorem 8, searching the neighborhood LINEXT$(\preceq)$ for partial orders of bounded Dilworth number ≥ 2 with respect to the QAP is an NP-complete problem.

Finally, we come to neighborhoods that are based on tree structures (cf. Sect. 4.3). In this area, we had four polynomial time results for the TSP: for the neighborhood TREE(T) if $\max_{v \in T} |\Psi(v)|$ is polynomially bounded in n (Theorem 4), for the neighborhood PYRAMID, for the neighborhood PYRAMID-CV, and for the neighborhood TWISTED (Theorem 6).

First, let us discuss neighborhoods that are based on permutation trees. Permutation trees T for which $|\Psi(v)| = 1$ for all $v \in T$ lead to uninteresting neighborhoods TREE(T) that only contain a single permutation. On the other hand, the neighborhood TWIN may be represented via a permutation tree T for which $|\Psi(v)| \leq 2$ holds for all $v \in T$. Hence, there is no hope for getting non-trivial positive results on the QAP via permutation trees. Theorem 10 shows that also the neighborhoods PYRAMID, PYRAMID-CV, and TWISTED can not help in reaching this goal. Also the neighborhood JORDAN whose status with respect to the TSP remained unsettled (cf. Open Problem 1) is NP-hard to search with respect to the QAP.

Theorem 10. *It is NP-hard to minimize for given input matrices $A = (a_{ij})$ and $B = (b_{ij})$ the function* QAP(A, B, π) *over*

(i) the permutations $\pi \in$ PYRAMID,
(ii) the permutations $\pi \in$ PYRAMID-CV,
(iii) the permutations $\pi \in$ TWISTED,
(iv) the permutations $\pi \in$ JORDAN.

□

To summarize the results of this section, we notice that unless $P = NP$, all exponential neighborhoods treated in this paper do not fulfill property (N2) with respect to the QAP. Hence, the following conjecture seems to be reasonable.

Conjecture 1. *Under the assumption $P \neq NP$: For every neighborhood $\mathcal{N}$ that can be searched in polynomial time with respect to the QAP, there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that $|\mathcal{N}_n| = O(p(n))$.*

6. Discussion

In this paper we considered exponential neighborhoods for the TSP and for the QAP. For the TSP, we investigated a variety of neighborhoods that can be defined in a simple way via assignments, matchings in bipartite graphs, partial orders, trees and other combinatorial structures, and we were able to identify several properties of these combinatorial structures that lead to polynomial time optimization algorithms. For the QAP, we encountered quite a different situation: Every exponential neighborhood that we considered in this paper provably leads to an NP-complete optimization problem for the QAP. In fact, it is rather easy to come up with other exponential neighborhoods over which the QAP cannot be solved efficiently (this part of our work, however, will remain unpublished). These investigations naturally lead us to Conjecture 1. Settling Conjecture 1 and Problem 2 are also the main questions that are left open in this paper.

Most of our polynomial time results should be considered merely as first versions or as sketches of polynomial time algorithms. We were mainly interested in deriving polynomial time results and not in getting a polynomial running time with a small exponent. Currently, we are performing large scale computational experiments on which we will report elsewhere.

A question that we did not attack is how to actually measure the quality of a neighborhood for a local search procedure. Exponential size of a neighborhood clearly is *not* sufficient to guarantee a highly efficient local search algorithm. Another important parameter is e.g. the average time of going from one permutation to another permutation within the local search (cf. Tovey [35]). Overall, we feel that we only scratched the surface of this area, and we hope that this paper will be the starting point of a systematic theoretical study of exponential neighborhoods.

Acknowledgements. We thank Eranda Çela, Richard Congram, Bettina Klinz, Chris Potts, and Steef van de Velde for discussions, for helpful comments, and for pointing out several minor mistakes in an earlier version of this paper.

A. Appendix: Proofs of the theorems on the TSP

Proof of Theorem 1(i). The proof is done by a simple reduction from the standard travelling salesman problem. In the decision version of the standard TSP, the input consists of a non-negative $n \times n$ distance matrix $D = (d_{ij})$ together with an integer bound d^*. The question is whether there exists a permutation $\phi \in S_n$ such that $\textsc{Tsp}(D, \phi) \leq d^*$. We construct a $2n \times 2n$ matrix C as follows:

- For all $1 \leq i \leq n$, $c_{2i-1,2i} = 0$
- For all $1 \leq i \leq n$ and all $\ell \neq 2i$, $c_{2i-1,\ell} = \infty$
- For all $1 \leq i \leq n$, $c_{2i,2i-1} = \infty$
- For all $1 \leq i \neq j \leq n$, $c_{2i,2j-1} = d_{ij}$
- For all $1 \leq i \neq j \leq n$, $c_{2i,2j} = \infty$

We claim that there exists a permutation $\phi \in S_n$ such that $\text{TSP}(D, \phi) \leq d^*$ if and only if there exists a permutation $\pi \in$ ASSIGN-EO such that $\text{TSP}(C, \pi) \leq d^*$:

First, assume that there exists a permutation $\pi \in$ ASSIGN-EO such that $\text{TSP}(C, \pi) \leq d^*$. In π, the only possible successor for a city $2i-1$ is the city $2i$. From this it can be seen that the tour ψ defined as $\langle \pi(2)/2, \pi(4)/2, \ldots, \pi(2n-2)/2, \pi(2n)/2 \rangle$ constitutes a travelling salesman tour of cost $\leq d^*$ for D. Vice versa, if $\text{TSP}(D, \phi) \leq d^*$ then the permutation $\psi = \langle 2\pi(1)-1, 2\pi(1), 2\pi(2)-1, 2\pi(2), \ldots, 2\pi(n)-1, 2\pi(n) \rangle$ yields $\text{TSP}(C, \pi) \leq d^*$.

□ □

Proof of Theorem 1(ii) and (iii). We only give the proof for statement (iii). The proof for statement (ii) will follow from the observation that in the instance constructed below, all feasible permutations $\pi \in$ ASSIGN-2/3 with $\text{TSP}(C, \pi) = 0$ also belong to ASSIGN-MOD3.

The proof is done by a reduction from PARTITION INTO TRIANGLES (cf. Garey and Johnson [13]) which is defined as follows. The input consists of a tripartite undirected graph $G = (U \cup V \cup W, E)$ with tripartition $U = \{u_1, \ldots, u_k\}$, $V = \{v_1, \ldots, v_k\}$, and $W = \{w_1, \ldots, w_k\}$, and with $E \subseteq (U \times V) \cup (V \times W) \cup (W \times U)$. The question is whether there exists a partition of the node set $U \cup V \cup W$ into triples (u, v, w) such that (u, v), (v, w) and (w, u) all belong to E. PARTITION INTO TRIANGLES is NP-complete.

Set $n = 6k$ and construct an $n \times n$ distance matrix C as follows. For every $1 \leq i \leq k$, the cities $6i-3$ and $6i$ correspond to node u_i, the city $6i-2$ corresponds to node v_i, the city $6i-1$ corresponds to node w_i, and the cities $6i-4$ and $6i-5$ are dummy cities. The distances are defined as follows (the indices are taken modulo $6k$, i.e. city $6k+1$ equals city 1 and so on):

- For all $1 \leq i \leq k$, $c_{6i-6,6i-5} = c_{6i-5,6i-4} = c_{6i-4,6i-3} = 0$. The distances between cities $6i-5$ and $6i-4$ and all other cities are 1. Moreover, the distances from $6i-6$ to all other cities are 1.
- For all $1 \leq i \leq k$, $c_{6i-3,6j-2} = 0$ if and only if in G there is an edge between u_i and v_j. The distances from city $6i-3$ to all other cities are 1.
- For all $1 \leq i \leq k$, $c_{6i-2,6j-1} = 0$ if and only if in G there is an edge between v_i and w_j. The distances from city $6i-2$ to all other cities are 1.
- For all $1 \leq i \leq k$, $c_{6i-1,6j} = 0$ if and only if in G there is an edge between w_i and u_j. The distances from city $6i-1$ to all other cities are 1.

It can be shown that there exists a permutation $\pi \in$ ASSIGN-2/3 with $\text{TSP}(C, \pi) = 0$ if and only if the instance of PARTITION INTO TRIANGLES has answer "Yes".

□ □

Proof of Theorem 2. The proof is done by a reduction from the NP-complete MAX CUT problem in cubic graphs (cf. Garey and Johnson [13]). This version of the MAX CUT problem takes as input an undirected simple graph $G = (V, E)$ where every node has degree three, together with an integer q. The goal is to decide whether there exists a partition $V = V_1 \cup V_2$ of the nodes such that at least q of the edges in E have their endpoints in different parts of the partition. Note that in the cubic graph G, $3|V| = 2|E|$ holds.

For every node v and for every incident edge $e = (v, w) \in E$, we introduce two corresponding cities that are denoted by $X_1(e, v)$ and $X_2(e, v)$. For every node $v \in V$ with incident edges e_1, e_2, and e_3, we introduce three positions $P(e_1, v)$, $P(e_2, v)$, $P(e_3, v)$ and three positions $Q_1(v)$, $Q_2(v)$, $Q_3(v)$ that may only be occupied by the six cities that correspond to this node. Then the two matchings in the cycle

$$X_1(e_1, v) - P(e_1, v) - X_2(e_1, v) - Q_1(v) - X_1(e_2, v) - P(e_2, v) - \\ -X_2(e_2, v) - Q_2(v) - X_1(e_3, v) - P(e_3, v) - X_2(e_3, v) - Q_3(v) - X_1(e_1, v)$$

encode the two feasible assignments between the cities corresponding to v and the positions corresponding to v (actually this cycle is part of the bipartite graph H that we are going to define below). Moreover, there will be $3|E|$ *dummy cities* and $3|E|$ *dummy positions*. Every dummy city can be assigned to a unique corresponding dummy position. The introduced cities form the set I of the bipartite graph H, and the introduced positions form the set I'. The edge set F contains only edges that arise from the cycles defined above, and edges that connect dummy cities to their unique dummy positions.

Next, we specify the exact ordering of the positions. In this ordering, first there comes an alternating sequence of $3|V|$ dummy positions and the $3|V|$ positions $Q_\ell(v)$ with $1 \le \ell \le 3$ and $v \in V$. Then there comes an alternating sequence of the remaining $|E|$ dummy positions and $|E|$ pairs of positions $P(*, *)$: For every edge $e = (v, w) \in E$, the sequence contains a corresponding pair of positions that consists of the two positions $P(e, v)$ and $P(e, u)$ next to each other. Finally, we specify the distance matrix C that describes the distances of the cities to each other.

- For every edge $e = (v, w) \in E$, the two cities $X_1(e, v)$ and $X_1(e, u)$ are at distance 1 to each other, and also the two cities $X_2(e, v)$ and $X_2(e, u)$ are at distance 1 to each other.
- All other distances are 0; note that especially every dummy city is at distance 0 to all other cities.

This completes the description of the TSP instance and of the bipartite graph H. The two possible matchings between the cities corresponding to a node v and the positions corresponding to v encode whether node v is put into node set V_1 or into node set V_2. It can be shown that the instance of MAX CUT has answer "Yes" if and only if there exists a permutation $\pi \in \text{MATCHING}_H$ such that $\text{TSP}(C, \pi)$ takes a value of at most $|E| - q$.

□ □

Proof of Theorem 3. Throughout the proof of this theorem, we consider the Dilworth number d of the partial order to be a constant. By the theorem of Dilworth [11], the set

I of cities can be partitioned into d linearly ordered chains $I_1, \ldots, I_d$. Moreover, such a chain partition can be computed in time $O(n^d)$. For two cities $i \neq j$ with $i \preceq j$, we say that city i is the *direct predecessor* of city j in a chain partition, if (i) cities i and j belong to the same chain I_ℓ and if (ii) there does not exist a city $k \in I_\ell$, $i \neq k \neq j$ with $i \preceq k \preceq j$.

We generate the acyclic, weighted, directed graph $G = (V, A)$ where every node is a $(d+1)$-tuple from

$$I \cup \{\emptyset\} \times I_1 \cup \{\emptyset\} \times I_2 \cup \{\emptyset\} \times \cdots \times I_d \cup \{\emptyset\}, \tag{9}$$

such that the first component of the $(d+1)$-tuple also occurs at a (unique) other coordinate. This first component is called the *head* of the node, the other d components are called the *body components* of the node. Hence, graph G contains $O(n^d)$ nodes. There is an arc from a source node to a target node if and only if

- the two nodes have distinct heads,
- the two nodes agree in exactly $d-1$ of the body components, but not in that body component that corresponds to the chain I_ℓ containing the head of the target node,
- in this ℓth body component, the component of the target node equals the head of the target node, and the corresponding component of the source node is the direct predecessor of the head of the target node.

The length of this arc equals the distance from the head of the source node to the head of the target node. Clearly, the number of arcs in G is $O(n^d)$. Intuitively speaking, every node in this graph encodes the starting segment of some permutation in LINEXT($\preceq$): The head describes the last city visited by this starting segment, and the body component for chain I_ℓ gives the last city in chain I_ℓ that has been visited by the starting segment. If a source node is connected to a target node by an arc, this means that the starting segment that corresponds to the target node can be obtained from the starting segment that corresponds to the source node by adding one additional city. If the position corresponding to some chain I_ℓ equals "$\emptyset$", then the starting segment has not yet visited any city from this chain.

The shortest TSP tour in LINEXT($\preceq$) decomposes into a Hamiltonian path that starts in a city x without predecessor in the partial order and ends in a city y without successor, and into the edge (y, x). With this it is clear that in order to solve this version of the TSP, one only has to perform a shortest path computation in the graph G for every pair of cities x without predecessor and y without successor. Since there are only $O(d^2)$ such pairs and since every shortest path can be computed in $O(n^d)$ time, this leads to the desired $O(n^d)$ time solution algorithm.

□ □

Proof of Theorem 4. The node set of the permutation tree T is denoted by V; hence, V consists of n leaves and of $O(n)$ interior nodes. For $v \in V$, denote by $T(v)$ the maximal subtree rooted at node v. For a leaf a in $T(v)$, we denote by $T(v, a)$ the tree $T(w)$ where w is the unique son of v with a in LEAF($T(w)$). For an interior node v, two leaves a and b in LEAF($T(v)$) are said to be *separated by* v, if and only if they are in two distinct subtrees rooted at sons of v. We denote this by $(a, b) \in$ SEP(v). For the

sake of completeness, we also define that for every leaf v, $(v, v) \in \text{SEP}(v)$ holds. It can easily be seen that the overall number of triples (v, a, b) with $(a, b) \in \text{SEP}(v)$ equals n^2 (cf. [6]).

For any interior node v and for any two leaves a and b in $T(v)$ with $(a, b) \in \text{SEP}(v)$, define two values $X[v; a, b]$ and $Y[v; a, b]$ as follows: $X[v; a, b]$ is the length of the shortest Hamiltonian path in $\text{TREE}(T(v))$ that starts in a and ends in b while obeying all restrictions imposed by the permutation tree. $Y[v; a, b]$ is the length of the shortest Hamiltonian path through the cities in $\text{LEAF}(T(v, a)) \cup \{b\}$ that starts in a and ends in b and whose initial part belongs to $\text{TREE}(T(v, a))$. If v is a leaf, define $X[v; v, v] = 0$ and $Y[v; v, v] = 0$. Our goal is to compute all values $X[v; *, *]$ and $Y[v; *, *]$. All computations are done in a bottom-up fashion, starting at the leaves v of T and moving up towards the root. When we are dealing with a father, for all of its sons w the values $X[w; *, *]$ have already been computed.

We only need to consider the case where v is an interior node with sons $v_1, \ldots, v_d$ ordered from left to right. For $a \in \text{LEAF}(T(v))$ define v_i to be the root of the subtree $T(v, a)$. Then the equation

$$Y[v; a, b] \;=\; \min \left\{ \; X[v_i; a, z] + c_{z,b} \;\mid\; z \in \text{LEAF}(T(v_i)) \; \right\} \tag{10}$$

holds. Next, let us compute $X[v; a, b]$. Consider some fixed permutation $\psi \in \Psi(v)$ with $a \in \text{LEAF}(T(v_{\psi(1)}))$ and $b \in \text{LEAF}(T(v_{\psi(d)}))$, and for $2 \le k \le d$, consider arbitrary nodes $z_k \in \text{LEAF}(T(v_{\psi(k)}))$. Then

$$X[v; a, b] \;\le\; Y[v; a, z_2] + \sum_{k=2}^{d-1} Y[v; z_k, z_{k+1}] + X[v_{\psi(d)}; z_d, b] \tag{11}$$

holds, and in fact $X[v; a, b]$ equals the minimum of the righthand side taken over all such choices for ψ and $z_2, \ldots, z_d$. For every fixed ψ, computing the corresponding minimum over choices for $z_2, \ldots, z_d$ can be done as a shortest path computation in an appropriate underlying graph with $O(n)$ nodes and $O(n^2)$ edges. This can be done by standard techniques in $O(n^2)$ time, and hence, the total time for computing $X[v; a, b]$ is $O(|\Psi(v)|n^2)$. Since there are only $O(n^2)$ values $X[v; a, b]$ and $Y[v; a, b]$ that have to be computed, the overall time for computing all these values is $O(\, fn^4)$ where $f = \max_{v \in T} |\Psi(v)|$.

Finally, note that the shortest TSP tour in $\text{TREE}(T)$ decomposes into a single edge $(1, x)$ and a shortest Hamiltonian path through $\{1, \ldots, n\}$ that starts in city x and ends in city 1, for an appropriate city x. Hence, if r is the root of T then the length of this shortest tour equals $\min_x \left\{ c_{1,x} + X[r; x, 1] \right\}$.

□ □

Proof of Theorem 5. The proof is done by carefully counting the tours in $3\text{-OPT}(\pi)$. Since we are dealing with symmetric distance matrices, we will not distinguish between a permutation ϕ and its reverse permutation. Let t_0, t_2, and t_3 denote the number of tours in $3\text{-OPT}(\pi)$ that differ from π in 0, 2 and 3 edges, respectively, and let p_0, p_2, and p_3, respectively, denote the number of such tours that are in $\text{PYRAMID-CV}(\pi)$. Trivially, $t_0 = p_0 = 1$ holds. If a tour ϕ differs from π in exactly two edges, it results from π by removing two *non-adjacent* edges and reconnecting the resulting parts in the only

possible way. From this and from the fact that every such tour is in PYRAMID-CV(π), it follows that $t_2 = p_2 = \frac{1}{2}n(n-3)$ holds.

Next, let us discuss the tours that differ from π in exactly three edges. The removal of three edges from π decomposes the tour into three paths X, Y, and Z such that π is a cyclic shift of $X \star Y \star Z$. Since we do not distinguish between permutations and their reverse permutations, we may assume without loss of generality that any permutation 3-OPT(π) that results from the paths X, Y, and Z and that differs from π in exactly three edges starts with the path X (after an appropriate cyclic shift). Hence, there are only four such tours in 3-OPT(π):

$$\phi_1 = X \star Z^- \star Y, \quad \phi_2 = X \star Z \star Y^-, \quad \phi_3 = X \star Z \star Y, \quad \phi_4 = X \star Y^- \star Z^-. \quad (12)$$

(We do not consider the remaining four tours $X \star Y \star Z$, $X \star Y \star Z^-$, $X \star Y^- \star Z$, and $X \star Z^- \star Y^-$ since they differ from π in at most two edges). Next, distinguish three cases. (C1). The case where two or more of the paths X, Y, and Z consist of a single city is meaningless, as in this case the tours ϕ_i differ from π in at most two edges. (C2). In case exactly one of the three paths consists of a single city, say path X, then ϕ_1 and ϕ_2 differ from π in only two edges, and ϕ_3 is the reverse of ϕ_4. (C3). If all three paths contain at least two cities, then the four permutations ϕ_1, ϕ_2, ϕ_3 and ϕ_4 are pairwise distinct. Finally, since there are $n(n-4)$ possibilities for case (C2) and $\frac{1}{6}n(n-4)(n-5)$ possibilities for case (C3), we get that $t_3 = \frac{1}{3}n(n-4)(2n-7)$.

What about p_3? First, observe that every tour that results from case (C2) is in PYRAMID-CV(π). Observe that $\pi = Y \star Z \star X$ (remember that PYRAMID-CV looks at all cyclic shifts) and consider the cyclic shift $\phi_1' = Y \star X \star Z^-$ of ϕ_1. With this, it is easy to see that ϕ_1 is contained in PYRAMID-CV(π). By similar arguments, one gets that also ϕ_2 and ϕ_4 are contained in PYRAMID-CV(π). Summarizing, this then yields $p_3 \geq \frac{1}{2}n(n-4)(n-3)$ and the statement of the theorem follows from a simple calculation.

□□

Proof of Theorem 6. This proof is done by straightforward dynamic programming. For every i, j with $1 \leq i \leq j \leq n$ and for every a, b with $i \leq a, b \leq j$, we introduce the value $X[i, j; a, b]$ as follows: $X[i, j; a, b]$ is the length of the shortest Hamiltonian path through the cities in $\{i, i+1, \ldots, j\}$ that starts in a, that ends in b, and that forms a twisted sequence for the cities in $\{i, i+1, \ldots, j\}$. Our goal is to compute all such values $X[i, j; a, b]$. These values are computed in the order of increasing value of $j - i$.

Clearly, we may set $X[i, i; i, i] := 0$, $X[i, i+1; i, i+1] := c_{i,i+1}$, $X[i, i+1; i+1, i] := c_{i+1,i}$ for $1 \leq i \leq n-1$. Moreover, all other values $X[i, j; a, b]$ with $a = b$ are set to ∞. For $j \geq i+2$ and $a < b$, we compute

$$X[i, j; a, b] = \min_{k,e,f} \{\, X[i, k; a, e] + X[k+1, j; f, b] + c_{e,f} \mid a \leq k \leq b-1,\ i \leq e \leq k,\ k+1 \leq f \leq j \,\}, \quad (13)$$

and for $j \geq i+2$ and $a > b$, we compute in a symmetric way

$$X[i, j; a, b] = \min_{k,e,f} \{\, X[k+1, j; a, e] + X[i, k; f, b] + c_{e,f} \mid b \leq k \leq a-1,\ k+1 \leq e \leq j,\ i \leq f \leq k \,\}. \quad (14)$$

The intuition behind these formulas is as follows: In every interval representation of a twisted sequence for $\{i, i+1, \ldots, j\}$, there must exist a k, $i \le k \le j-1$, such that either none of the intervals does contain $\{k, k+1\}$ or only the interval $[i, j]$ contains $\{k, k+1\}$. The formulas (13) and (14) essentially check every possibility for this value k.

Since there are $O(n^4)$ values $X[i, j; a, b]$ that are to be computed, and since every value is computed in $O(n^3)$ time via (13) and (14), the overall computation time is $O(n^7)$. Finally, note that the shortest TSP tour for $\{1, \ldots, n\}$ in TWISTED decomposes into a single edge $(x, 1)$ and into a shortest Hamiltonian path through $\{1, \ldots, n\}$ that starts in city 1 and ends in city x, for an appropriate city x. Hence, the length of this shortest tour equals $\min_x \{c_{1,x} + X[1, n; x, 1]\}$.

□ □

Proof of Theorem 7. Let $\mathcal{N}$ be an exponential neighborhood that can be searched in time $f(n)$ with respect to the TSP. Then while searching through $\mathcal{N}_n$, the search algorithm considers at most $f(n)$ of the $\Theta(n^2)$ distances between the n cities. Construct an undirected graph G whose vertices are the cities and that contains exactly those $O(f(n))$ edges that are considered by the search algorithm. Let $\deg(i)$ denote the degree of city i in G.

Clearly, the number of Hamiltonian cycles in G is an upper bound on $|\mathcal{N}_n|$. In a Hamiltonian cycle, there are at most $\deg(i)$ possibilities for the successor city of city i. Hence we get that

$$|\mathcal{N}_n| \;\le\; \prod_{i=1}^{n} \deg(i) \;\le\; \left(\frac{1}{n} \sum_{i=1}^{n} \deg(i) \right)^n \;\le\; \left(\frac{2}{n} f(n) \right)^n, \tag{15}$$

where we applied the arithmetic-geometric mean inequality. This proves the theorem.

□ □

B. Appendix: Proofs of the theorems on the QAP

Proof of Theorem 8. The proof is done by a reduction from the NP-complete MAX CUT problem (cf. Garey and Johnson [13]). The MAX CUT problem takes as input an undirected simple graph $G = (V, E)$ and a number q, and asks whether there exists a partition $V = V_1 \cup V_2$ of the nodes such that at least q of the edges in E have their endpoints in different parts of the partition.

Let $k = |V| + |E|$ and set $n = 2k$. The matrices A and B are both symmetric $n \times n$ matrices with zeroes on the main diagonal, and thus it is sufficient to specify the entries above their main diagonals. The first $2|V|$ rows (and by symmetry, columns) in matrices A and B correspond to the nodes of G, and the last $2|E|$ rows and columns correspond to the edges in G. More precisely, for $1 \le i \le |V|$, the two rows $2i-1$ and $2i$ correspond to the ith node, and for $1 \le j \le |E|$, the two rows $2|V|+2j-1$ and $2|V|+2j$ correspond to the jth edge. In the definition of A and B, we will put copies of the 2×2 matrices

$$M_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

at the crossings of the two rows $2i-1$ and $2i$ with the two columns $2|V|+2j-1$ and $2|V|+2j$, in case the ith node is incident to the jth edge. With this, matrix A is defined as follows: For every pair (i, j) such that the ith node is incident to the jth edge, put the 2×2 matrix M_1 at the crossing of the two rows $2i-1$ and $2i$ with the two columns $2|V|+2j-1$ and $2|V|+2j$. All other entries above the main diagonal of A are set to 0. Matrix B is defined as follows: For every edge, put the matrix M_1 at one crossing, and the matrix M_2 at the other crossing of the two columns corresponding to this edge with the rows corresponding to the two incident nodes. All other entries above the main diagonal of B are set to 0.

We show that the instance of MAX CUT has answer "Yes" if and only if there exists a permutation $\pi \in$ TWIN such that $\text{QAP}(A, B, \pi)$ takes a value of at most $2|E|-2q$. First, assume that the instance of MAX CUT has answer "Yes", i.e. that there exists a partition $V = V_1 \cup V_2$ of the nodes such that at least q of the edges in E have their endpoints in different parts of the partition. Define the permutation π as follows: If the ith node is in V_1, then define $\pi(2i-1) = 2i-1$ and $\pi(2i) = 2i$, and if the $i$the node is in V_2, then define $\pi(2i-1) = 2i$ and $\pi(2i) = 2i-1$. If the jth edge is in the cut, then define $\pi(2|V|+2j-1)$ and $\pi(2|V|+2j)$ in such a way that these two columns contribute 0 to the objective function. If the jth edge is not in the cut, then define $\pi(2|V|+2j-1) = 2|V|+2j-1$ and $\pi(2|V|+2j) = 2|V|+2j$; the contribution of these two columns to the objective function will be 2. It can be verified that $\pi \in$ TWIN and that $\text{QAP}(A, B, \pi)$ is at most $2|E|-2q$. Next, assume that $\text{QAP}(A, B, \pi) \leq 2|E|-2q$ for some $\pi \in$ TWIN. Define a partition of $V = V_1 \cup V_2$ such that the ith node is in V_1 if $\pi(2i) = 2i$, and otherwise in V_2. It can be verified that the cut defined by this partition is crossed by at least q edges in E.

□ □

Proof of Theorem 9. The proof is done by a straightforward reduction from the standard QAP. Let the $n \times n$ matrices A and B form an input for the standard quadratic assignment problem. Define $2n \times 2n$ matrices A' and B' as follows: For $1 \leq i, j \leq n$, define $a'_{2i,2j} = a_{ij}$ and $b'_{2i,2j} = b_{ij}$. For $1 \leq i, j \leq 2n$, define $a'_{ij} = b'_{ij} = 0$ if i or j is odd. It is easy to prove that the minimum of the function $\text{QAP}(A, B, \pi)$ over all permutations $\pi \in S_n$ equals the minimum of the function $\text{QAP}(A', B', \pi')$ over the permutations $\pi' \in$ ASSIGN.

□ □

Proof of Theorem 10(i) and (ii). We only give the proof for statement (i). Correctness of statement (ii) easily follows along the same construction. The proof of (i) is done by a reduction from the NP-complete EQUIPARTITION problem (cf. Garey and Johnson [13]). The EQUIPARTITION problem takes as input $2k$ positive integers $x_1, x_2, \ldots, x_{2k}$ and asks whether there exists a set $I \subseteq \{1, 2, \ldots, 2k\}, |I| = k$, such that $\sum_{i \in I} x_i = \sum_{i \notin I} x_i$. Set $n = 2k$ and define the $n \times n$ matrix $A = (a_{ij})$ by $a_{ij} = x_i \cdot x_j$ for $1 \leq i, j \leq 2k$. Define matrix B by $b_{ij} = 1$ for $1 \leq i, j \leq k$ and $k+1 \leq i, j \leq 2k$, and by $b_{ij} = -1$, otherwise. We show that the instance of EQUIPARTITION has answer "Yes" if and only if there exists $\pi \in$ PYRAMID such that $\text{QAP}(A, B, \pi)$ takes a non-positive value.

First, assume that there exists a permutation $\pi \in$ PYRAMID such that QAP(A, B, π) take a value ≤ 0. Then the equation

$$0 \;\geq\; \text{QAP}(A, B, \pi) \;=\; \sum_{i=1}^{n}\sum_{j=1}^{n} x_{\pi(i)}x_{\pi(j)}b_{ij} \;=\; \left(\sum_{i:\pi(i)\leq k} x_{\pi(i)} - \sum_{i:\pi(i)>k} x_{\pi(i)}\right)^2 \tag{16}$$

holds, and hence the index set $I = \{\pi(i) : \pi(i) \leq k\}$ yields a solution of the EQUIPARTITION instance. Next, assume that the instance of EQUIPARTITION has answer "Yes", i.e. that there exists a set $I \subseteq \{1, 2, \ldots, 2k\}$, $|I| = k$, such that $\sum_{i\in I} x_i = \sum_{i\notin I} x_i$. By symmetry we may assume that $1 \in I$. Define the permutation $\pi = \langle y_1, \ldots, y_k, z_1, \ldots, z_k\rangle$ where the y_i are the elements of I put into increasing order, and where the z_i are the elements of $\{1, 2, \ldots, 2k\} \setminus I$ put into decreasing order. Then $\pi \in$ PYRAMID and QAP$(A, B, \pi) = 0$.

□□

Proof of Theorem 10(iii). This proof is done by modifying the proof of Theorem 8 in a straightforward way. First we observe that TWIN $\subseteq$ TWISTED holds. Next, we consider the symmetric $n \times n$ matrices A and B that are constructed in this NP-completeness proof: Both are 0-1 matrices with zeroes on the main diagonal, and hence $0 \leq$ QAP$(A, B, \pi) \leq n^2 - n$ holds. We define matrices A' and B' by changing the main diagonals of A and B in the following way: $a'_{2i-1,2i-1} = a'_{2i,2i} = n2^i$ and $b'_{2i-1,2i-1} = b'_{2i,2i} = n2^{n-i}$. It is easy to verify that for any $\pi' \in$ TWIN, the contribution of the main diagonal entries to the objective function QAP(A', B', π') equals $n^3 2^n$, whereas for any $\pi' \notin$ TWIN, the contribution of the main diagonal entries will be at least $n^3 2^n + n^2$. From this we conclude that the problem of minimizing QAP(A, B, π) over the permutations $\pi \in$ TWIN and the problem of minimizing QAP(A', B', π') over the permutations $\pi' \in$ TWISTED both take their minima at the same permutations.

□□

Proof of Theorem 10(iv). Let us introduce a subset AUX of the Jordan permutations over $\{1, 2, \ldots, 6k\}$. A permutation π is in AUX$_{6k}$ if and only if

$$\begin{aligned} \{\pi(3i-2),\ \pi(3i)\} = \{3i-2,\ 3i\} \qquad & \text{for } i = 1, \ldots, 2k, \text{ and} \\ \pi(3i-1) = 3i-1 \qquad & \text{for } i = 1, \ldots, 2k. \end{aligned}$$

In other words, every triple $3i-2, 3i-1, 3i$ of cities must occur at the positions $3i-2$, $3i-1$, and $3i$ of the permutation, and it must either occur in precisely this ordering or in the reverse ordering. It is not hard to see that AUX$_{6k} \subset$ JORDAN$_{6k}$ holds.

For the NP-completeness proof, we modify the proof of Theorem 8 in a similar way as we did in the proof of Theorem 10(iii). Again, we start with the symmetric $n \times n$ matrices A and B that are constructed in the proof of Theorem 8. Without loss of generality, we assume that n is divisible by 4, i.e. $n = 4k$. Since A and B are 0-1 matrices with zeroes on the main diagonal, $0 \leq$ QAP$(A, B, \pi) \leq n^2 - n$ holds. We construct $6k \times 6k$ matrices A' and B' as follows: We insert between any pair $2i-1$ and $2i$ of rows (respectively, of columns) a dummy row (respectively, a dummy column)

that with the exception of the diagonal entry entirely consists of 0-entries. The main diagonal entries of A' and B' are defined by $a'_{3i-2,3i-2} = a'_{3i-1,3i-1} = a'_{3i,3i} = 3^i$ and $b'_{3i-2,3i-2} = b'_{3i-1,3i-1} = b'_{3i,3i} = n3^{n-i}$, for $i = 1, \ldots, 2k$. It can be verified that for any $\pi' \in \text{AUX}$, the contribution of the main diagonal entries to the objective function $\text{QAP}(A', B', \pi')$ equals $n^3 3^n$, whereas for any $\pi' \in \text{JORDAN} \setminus \text{AUX}$, the contribution of the main diagonal entries will be at least $n^3 3^n + n^2$. With this, the problem of solving the QAP for A' and B' over the permutations in JORDAN boils down to the NP-complete problem of solving the QAP for A and B over the permutations in TWIN.

□ □

References

1. Aarts, E., Lenstra, J.K. (eds.) (1997): Local Search in Combinatorial Optimization. John Wiley, New York
2. Aurenhammer, F. (1988): On-line sorting of twisted sequences in linear time. BIT **28**, 194–204
3. Balas, E., Simonetti, N. (1996): Linear time dynamic programming algorithms for some new classes of restricted travelling salesman problems. Management Science Research Report # MSSR-617, Carnegie Mellon University, July 1996. A preliminary version appeared in Proceedings of IPCO V, LNCS **1084**, pp. 316–329. Springer Verlag
4. Booth, K.S., Lueker, G.S. (1976): Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms. J. Comput. Syst. Sci. **13**, 335–379
5. Burkard, R.E. (1990): Locations with spatial interactions: the quadratic assignment problem. In: Discrete Location Theory, Chapt. 9 (Mirchandani, P.B., Francis, R.L. (eds.)), pp. 387–437. John Wiley, New York
6. Burkard, R.E., Deĭneko, V.G., Woeginger, G.J. (1998): The Travelling Salesman and the PQ-Tree. Math. Oper. Res. **23**, 613–623. A preliminary version appeared in Proceedings of IPCO V, LNCS **1084**, pp. 490–504. Springer Verlag, 1996
7. Carlier, J., Villon, P. (1990): A new heuristic for the travelling salesman problem. RAIRO – Oper. Res. **24**, 245–253
8. Chandra, B., Karloff, H., Tovey, C. (1994): New results on the old k-opt algorithm for the TSP. In: Proceedings of 5th ACM-SIAM Symposium on Discrete Algorithms, pp. 150–159
9. Cornuejols, G., Naddef, G.D., Pulleyblank, W.R. (1983): Halin graphs and the travelling salesman problem. Math. Program. **26**, 287–294
10. Croes, G.A. (1958): A method for solving travelling-salesman problems. Oper. Res. **6**, 791–812
11. Dilworth, R.P. (1950): A decomposition theorem for partially ordered sets. Ann. Math. **51**, 161–166
12. Fonlupt, J., Nachef, A. (1993): Dynamic programming and the graphical travelling salesman problem. J. Assoc. Comput. Mach. **40**, 1165–1187
13. Garey, M.R., Johnson, D.S. (1979): Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco
14. Gilmore, P.C., Lawler, E.L., Shmoys, D.B. (1985): Well-solved special cases. Chapt. 4 in [25], pp. 87–143
15. Glover, F., Punnen, A.P. (1997): The travelling salesman problem: new solvable cases and linkages with the development of approximation algorithms. J. Oper. Res. Soc. **48**, 502–510
16. Gutin, G. (1997): Exponential neighborhoods local search for the travelling salesman problem. Research Report TR/2/97, Brunel University, Uxbridge, February 1997
17. Held, M., Karp, R.M. (1962): A dynamic programming approach to sequencing problems. J. SIAM **10**, 196–210
18. Hoffmann, K., Mehlhorn, K., Rosenstiehl, P., Tarjan, R.E. (1986): Sorting Jordan sequences in linear time using level-linked search trees. Information and Control **68**, 170–184
19. Johnson, D.S., McGeoch, L.A. (1997): The travelling salesman problem: a case study. Chapt. 8 in [1], pp. 215–310
20. Kern, W. (1989): A probabilistic analysis of the switching algorithm for the Euclidean TSP. Math. Program. **44**, 213–219
21. Klyaus, P.S. (1976): The structure of the optimal solution of certain classes of travelling salesman problems (in Russian). Vestsi Akad. Nauk BSSR, Phys. Math. Sci., Minsk, pp. 95–98
22. Koopmans, T.C., Beckmann, M.J. (1957): Assignment problems and the location of economic activities, Econometrica **25**, 53–76
23. Lawler, E.L. (1963): The quadratic assignment problem. Manage. Sci. **9**, 586–599

24. Lawler, E.L. (1975): The quadratic assignment problem: a brief review. In: Combinatorial Programming: Methods and Applications (Roy, B. (ed.)), Dordrecht, Holland, pp. 351–360
25. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (eds.) (1985): The travelling salesman problem. John Wiley, Chichester
26. Lin, S. (1965): Computer solutions to the travelling salesman problem. Bell System Tech. J. **44**, 2245–2269
27. Möhring, R.H. (1989): Computationally tractable classes of ordered sets. In: Algorithms and Order (Rival, I. (ed.)). Kluwer Academic Publishers, pp. 105–193
28. Papadimitriou, C.H., Steiglitz, K. (1977): On the complexity of local search for the travelling salesman problem. SIAM J. Comput. **6**, 76–83
29. Papadimitriou, C.H., Steiglitz, K. (1978): Some examples of difficult travelling salesman problems. Oper. Res. **26**, 434–443
30. Papadimitriou, C.H., Steiglitz, K. (1982): Combinatorial optimization: algorithms and complexity. Prentice Hall, New Jersey
31. Pardalos, P., Rendl, F., Wolkowicz, H. (1994): The Quadratic Assignment Problem: A Survey and Recent Developments. In: Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, (Pardalos, P., Wolkowicz, H. (eds.)), DIMACS Series in Discrete Mathematics and Theoretical Computer Science **16**, 1–42
32. Reinelt, G. (1994): The travelling salesman problem: Computational solutions for TSP applications. LNCS **840**, Springer Verlag, Berlin
33. Sarvanov, V.I., Doroshko, N.N. (1981): The approximate solution of the travelling salesman problem by a local algorithm that searches neighborhoods of exponential cardinality in quadratic time (in Russian). Software: Algorithms and Programs **31**, Mathematical Institute of the Belorussian Academy of Sciences, Minsk, pp. 8–11
34. Sarvanov, V.I., Doroshko, N.N. (1981): The approximate solution of the travelling salesman problem by a local algorithm with scanning neighborhoods of factorial cardinality in cubic time (in Russian). Software: Algorithms and Programs **31**, Mathematical Institute of the Belorussian Academy of Sciences, Minsk, pp. 11–13
35. Tovey, C.A. (1997): Local improvement on discrete structures. Chapt. 3 in [1], pp. 57–89