Technische Universität Graz Institut für Mathematik

The N-line Traveling Salesman Problem

Günter Rote

Report 109 June 1991

Technische Universität Graz, Steyrergasse 30, A-8010 Graz, Austria

Version 2c

A shortened version of this report appeared in *Networks* **22** (1992), 91–108. This report corrects a small error in the corollary in Section 6, and a few references to the literure are extended and updated.

Title page processed by TEX on May 22, 1998

A slightly shortened version of this report appeared in *Networks* **22** (1992), 91–108.

The N-line Traveling Salesman Problem

Günter Rote
Technische Universität Graz
Institut für Mathematik
Steyrergasse 30, A-8010 Graz, Austria
electronic mail: `rote@opt.math.tu-graz.ac.at`
June 1991

Abstract

The special case of the Euclidean Traveling Salesman Problem, where the n given points lie on a small number (N) of parallel lines in the plane, is solved by a dynamic programming approach in time n^N, for fixed N, i. e., in polynomial time. This extends a result of Cutler (1980) for 3 lines. Such problems arise for example in the fabrication of printed circuit boards, where the distance traveled by a laser which drills holes in certain places of the board should be minimized.

The parallelity condition can be relaxed to point sets which lie on N "almost parallel" line segments. We give a characterization of the allowed segment configurations by a set of forbidden subconfigurations.

This work was supported by the Fonds zur Förderung der wissenschaftlichen Forschung, project S32/01.

AMS 1991 categories: Primary: 90C27; Secondary: 68U05

CR categories and subject descriptors: G.2.2 [**discrete mathematics**]: graph theory — *path and circuit problems;* F.2.2 [**analysis of algorithms and problem complexity**]: nonnumerical algorithms and problems — *geometrical problems and computations*

General terms: algorithms, theory

OR/MS index 1978 subject classification: 491 on few parallel lines in the plane, 111 for the N-line Traveling Salesman Problem

IAOR subject classification: Main: network programming; Cross-reference: combinatorial analysis, networks

Keywords: Traveling Salesman Problem, computational geometry

1. Introduction

The *Planar Traveling Salesman Problem* is to find a closed curve through n given points in the plane (a *tour*) having shortest possible length. This problem is known to be NP-complete (Papadimitriou [1977], Itai, Papadimitriou, and Szwarcfiter [1982], cf. also Johnson and Papadimitriou [1985]). Therefore it makes sense to look for classes of the problem where solutions can be found in polynomial time. In the special case which we consider, the points lie on a small number of parallel lines. We present a dynamic programming algorithm for such problems. For a fixed number of lines, the algorithm is polynomial. (The exponent of the polynomial time bound is the number of lines.) Such problems arise in practical applications, for example, in the manufacturing of printed circuit boards and related devices. Nevertheless, since the high running times are rather high, the algorithm seems to be of theoretical interest only.

Our algorithm is an extension of an algorithm by Cutler [1980] for three parallel lines. (For two lines, the problem is trivial.) Cutler also considered the Traveling Salesman *Path* Problem. A similar but easier special case was considered by Ratliff and Rosenthal [1983]: the problem of order-picking in a rectangular warehouse. These authors also used the dynamic programming paradigm for their problem and obtained a linear-time algorithm. Cornuéjols, Fonlupt, and Naddef [1985] extend the result of Ratliff and Rosenthal to the *Steiner Traveling Salesman Problem* for arbitrary undirected series-parallel graphs. In the Steiner Traveling Salesman Problem, we look for a closed path which visits each of a given *subset* of the vertices *at least* once. The algorithm takes linear time.

Another similar algorithm was given in Gilmore, Lawler, and Shmoys [1985], section 15, for the Traveling Salesman Problem with limited bandwidth. That algorithm also has linear running time (for fixed bandwidth).

Cutler's N-line Traveling Salesman Problem has recently been generalized in a different direction by Deĭneko, van Dal, and Rote [1994]. They considered the problem where the given points lie on the boundary of a convex polygon and on one additional line segment inside this polygon. Clearly, this class of problems contains the 3-line Traveling Salesman Problem as a special case. Moreover, they improved the complexity from $O(n^3)$ to $O(n^2)$.[†]

Our condition that the lines are parallel can be relaxed, and the algorithm can be applied to points on "almost parallel" lines, which still have the same combinatorial properties with respect to shortest tours.

In the next section we state the exact condition which the lines must fulfill in order that our algorithm is applicable. A characterization of such sets of line segments by forbidden

[†] A common generalization of the N-line Traveling Salesman Problem and the convex-hull-and-line Traveling Salesman Problem was considered by Deĭneko and Woeginger [1996]: the convex-hull-and-k-line Traveling Salesman Problem, which has k parallel (or "almost parallel", see the following paragraph) line segments inside the convex hull, whose carrying lines intersect the convex hull in two common edges. The time bound is $O(n^{k+2})$, which corresponds to the time bound in this paper.

sub-configurations is deferred to section 6, since it is somewhat peripheral to the main course of the paper. In section 2 we continue by stating one simple but essential lemma that follows from the conditions that we impose on the lines, and we discuss to what extent these conditions are necessary for our algorithm. We also formulate the most elementary facts about the Traveling Salesman Problem, and we define the required notations. In particular, we will define what we mean by a partial solution.

In the third section we derive the main geometric lemma about optimal partial solutions. In section 4, we then formulate a rather standard dynamic programming algorithm, whose correctness is based on that lemma. Finally, in section 5, we analyze the time and space complexity of the algorithm exactly. We can also deal with other metrics than the Euclidean distance. This and other possible extensions and open problems are discussed in the concluding section 7.

2. Elementary facts, definitions, and notations

A *simple* polygon is a polygon in which no two sides have a point in common except for the common endpoint of two adjacent sides.

The following lemma is an immediate consequence of the triangle inequality:

Lemma 1. Unless all points lie on one line the optimal tour is a simple polygon having the given points as vertices. ■

We consider the special case where the points lie on N lines, for a small number $N \geq 2$. We shall assume in the rest of the paper that $N \geq 2$, i. e., not all points lie on a straight line. The algorithm we are going to present works for the case when the lines are parallel, but also more generally when the points lie on N straight line segments fulfilling the following condition:

- No segment is perpendicular to the x-axis.
- For every pair of segments which are not parallel: If we project the two segments and the intersection of their carrying lines onto the x-axis, the projection of the intersection lies always outside the projections of the two segments, never on or between these projections (cf. figure 1).

Of course, the x-axis can be an arbitrary line, but for definiteness, we have chosen this formulation.

This condition allows configurations of segments which are quite non-parallel, such as the one shown in figure 2a (the half-star) or figure 2b (the zigzag). On the other hand, the set of segments shown in figure 3g (the 3-star) or 3e (the triangle) is forbidden. In section 6 we will show that the allowable configurations of segments are precisely those which do not contain a subconfiguration like the ones in figure 3a–g).

This condition has several consequences:

i) The line carrying a segment does not intersect any other segment (cf. figure 3b). In particular the segments themselves do not intersect (cf. figure 3a).

ii) It makes sense to speak of “left” and “right” with respect to points that lie on the same segment.

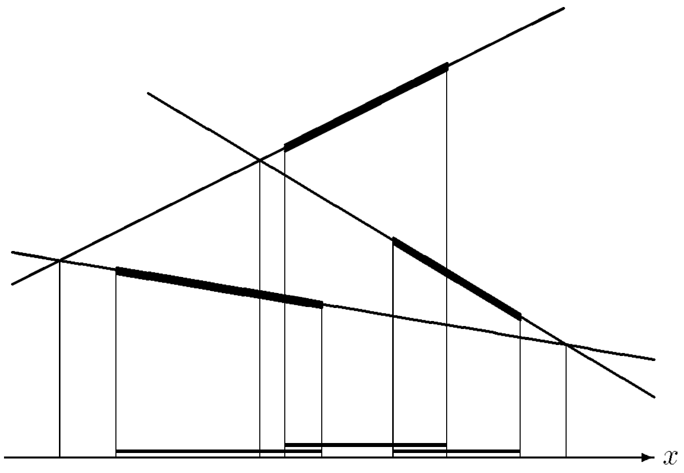

Figure 1.
Three line segments which fulfill
the condition for allowed segments

iii) There is a linear ordering of the segments from top to bottom.

Henceforth we shall always assume that the lines are numbered from 1 to N from top to bottom, and that the points on each line are numbered from left to right, from 1 to n_i.

We are goint to exploit one main property that follows from these assumptions, which we formulated in the following lemma. (The notation is already determined by the way how we are going to use the lemma.)

Lemma 2. *Let P be a simple polygon whose vertices are points of the given set. Let l be the bottom-most line that contains a vertex of the polygon, and let k be the top-most line that contains a vertex of the polygon (see figure 4a.) Assume that P touches each of the lines k and l in more than one point, and let $X_{k'}$ and $B(k)$ denote the left-most and right-most point of the polygon on line k, and similarly, define the points $X_{l'}$ and $B(l)$ on line l. Then the cyclic sequence of these points around the polygon cannot be $X_{k'}$, $B(k)$, $X_{l'}$, $B(l)$ (or the reverse order).*

Proof: Assume that the cyclic order were as given above. Then the simple polygon P together with parts of the lines k and l cound be extended to a planar embedding of the complete bipartite graph $K_{3,3}$, as shown in figure 4b.[‡] ■

We cannot relax the condition for allowed segment configurations, because then we would not have the ordering of the lines from top to bottom and the consistent ordering of the points from left to right, and it is not obvious how to even formulate an analog of the above properties or of lemma 2.

‡ A different proof of this lemma uses an argument of de Bruijn [1955] about sum of the internal angles of P. (De Bruijn's original argument considered only parallel segments.)

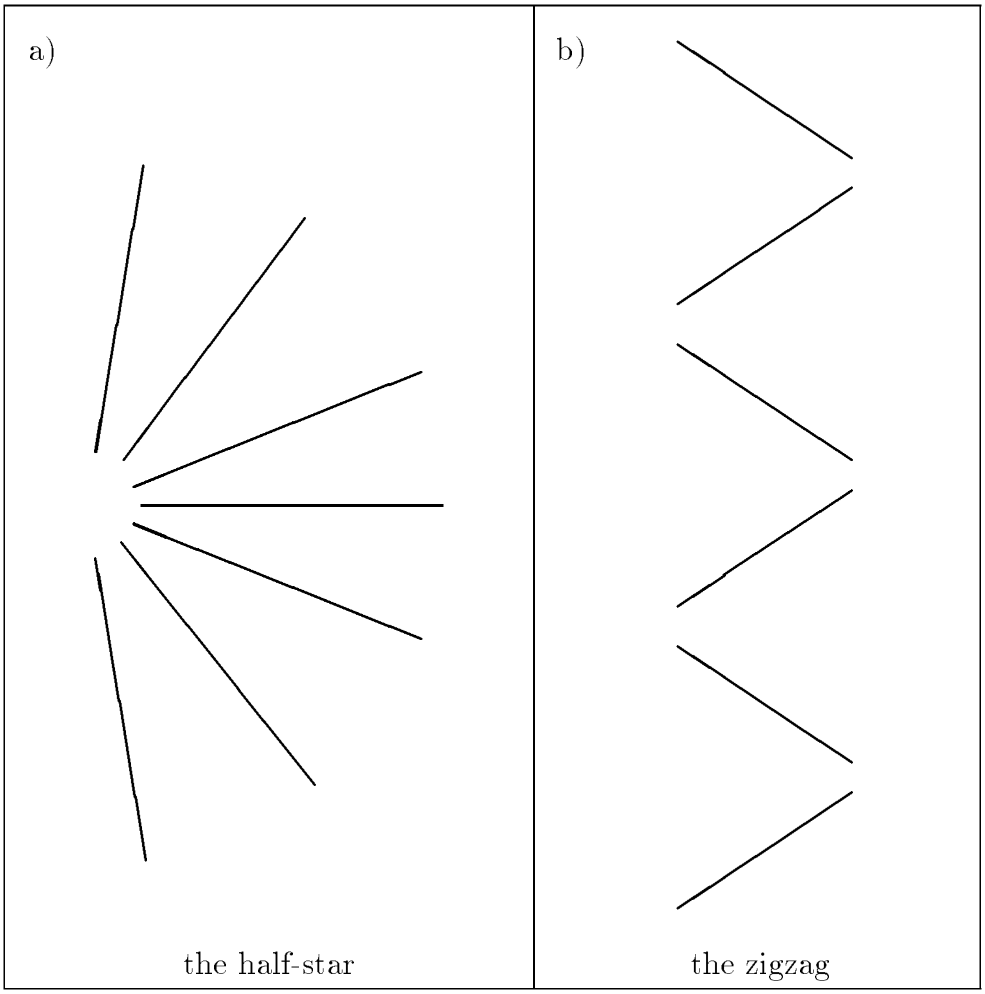

Figure 2. Allowed configurations

By a *path* on a set of points we mean an open polygonal line whose vertices are points of the set.

The distance between two points X and Y is denoted by $\mathsf{dist}(X, Y)$.

We are going to solve the Traveling Salesman Problem "from left to right", i. e., we will consider partial solutions defined on subsets of points, and we want to extend these solutions by adding more and more points on the right. The subsets of points which we consider will now be defined.

Let $(k_1, \ldots, k_N)$ be an N-tuple of integers, $0 \le k_i \le n_i$. $P(k_1, \ldots, k_N)$ is the set consisting of the first k_i points of every line. With respect to this sequence, the k_i-th point on line i (it exists only for $k_i > 0$) is called the i-th *boundary point*, denoted by $B(i)$. If $k_i \ge 2$, the $(k_i - 1)$-st point is denoted by $\overline{B}(i)$. Thus, $P(k_1, \ldots, k_N)$ contains the boundary points and the points to their left.

When we look at a tour restricted to a subset $P(k_1, \ldots, k_N)$ of points, we will not get a tour but rather a collection of paths. We want to extend such a partial tour by adding points and edges on the right. Thus, we have to focus our interest on the interface between

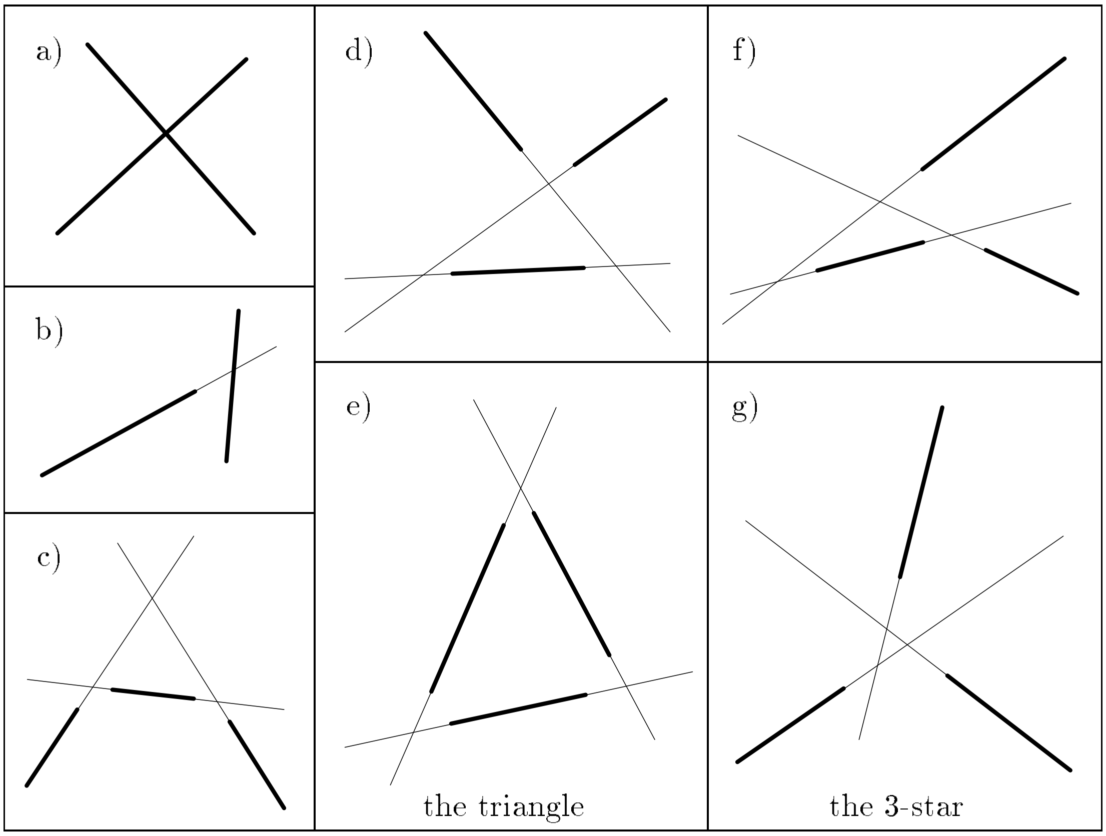

Figure 3. Forbidden configurations

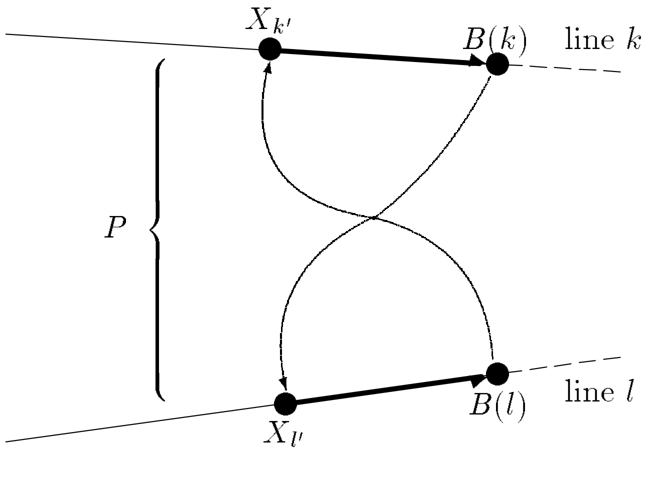

Figure 4a. The polygon P of lemma 2, and the self-intersection in lemma 3

the partial tour on $P(k_1, \ldots, k_N)$ and the rest of the tour. We shall see that we mainly have to consider the situation near the boundary points, but we have to take care that we don't add edges which connect two endpoints of the same path, forming a subtour which cannot be extended to a complete tour. Thus, in order to know which edges can be added,

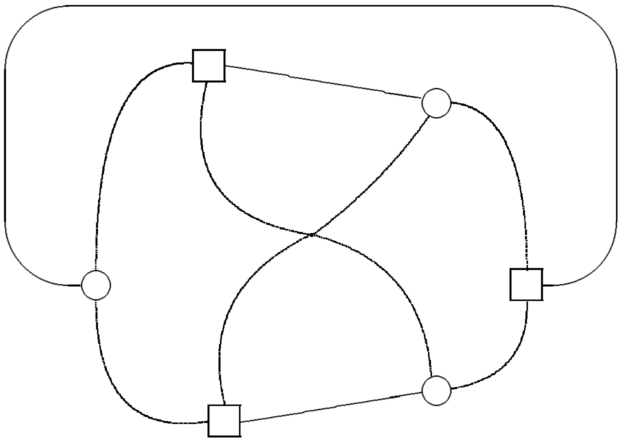

Figure 4b. Extension of the simple polygon P to an embedding of $K_{3,3}$

we have to classify the partial solutions according to the way how the paths connect the boundary points.

Let $S = \big\{\{i_1, i'_1\}, \{i_2, i'_2\}, \ldots, \{i_m, i'_m\}\big\}$ be a collection of disjoint two-element subsets of $\{1, \ldots, N\}$.

For non-empty S we define:

$M(S; k_1, \ldots, k_N) =$
the set of all collections $T = \{T_1, \ldots, T_m\}$ of m paths on the set $P(k_1, \ldots, k_N)$, where the j-th path extends between $B(i_j)$ and $B(i'_j)$, for $j = 1, \ldots, m$, and every point in $P(k_1, \ldots, k_N)$ is contained in a path.

In the case where some of the required $B(i_j)$ (or $B(i'_j)$) do not exist because $k_{i_j} = 0$, (or $k_{i'_j} = 0$) $M(S; k_1, \ldots, k_N)$ is empty.

For $S = \emptyset$ we define:

$M(\emptyset; n_1, \ldots, n_N) =$ the set of all tours through the given points.

$M(\emptyset; 0, \ldots, 0) =$ the set consisting only of the "empty tour".

The *length* of T, denoted by $\mathsf{length}(T)$, is the sum of the lengths of its paths.

The elements of the sets $M(S; k_1, \ldots, k_N)$ are called *partial tours.* We call a partial tour *cross-free* if it consists only of paths that do not intersect themselves or each other or if it is a tour which is a simple polygon or if it is the "empty tour". (A turn by 180° in a path of T counts as self-intersection.) Lemma 1 could now be rephrased as saying that an optimal tour is cross-free.

Figure 5a shows a cross-free element T of $M(\big\{\{2,4\}, \{3,5\}\big\}; 3, 5, 4, 4, 4, 0)$. Figure 5b shows that a set $M(S; k_1, \ldots, k_N)$ can contain no cross-free element at all. Thus, we do not require the paths in a solution to be disjoint.

The notation $S - \{i_1, j_1\} + \{i_2, j_2\}$ is an abbreviation for $\big(S \setminus \big\{\{i_1, j_1\}\big\}\big) \cup \big\{\{i_2, j_2\}\big\}$.

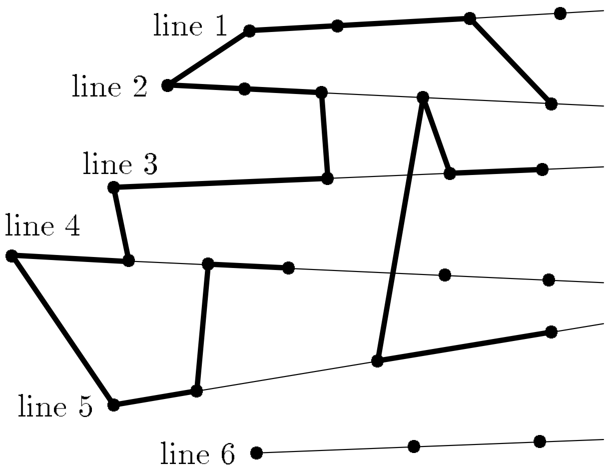

Figure 5a.
A cross-free partial solution of
$M(\{\{2,4\},\{3,5\}\};3,5,4,4,4,0)$

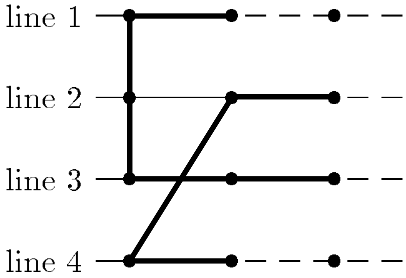

Figure 5b.
$M(\{\{1,3\},\{2,4\}\};2,3,3,2)$
contains no partial
solutions without crossings.

3. Properties of partial solutions

The following lemma states that an optimal partial tour in each set $M(S;k_1,\ldots,k_N)$ must contain some edge which is "close to the boundary" of the vertices in $P(k_1,\ldots,k_N)$. This will allow us to select the optimal partial tour in each set $M(S;k_1,\ldots,k_N)$ from a set of few candidates which consist of a smaller optimal partial tour with an additional edge.

Lemma 3.

A) *If T is a cross-free element of $M(S;k_1,\ldots,k_N)$ and $(k_1,\ldots,k_N)\neq(0,\ldots,0)$ then T contains one of the following edges:*

(i) *an edge $(B(i),B(j))$, for some i,j with $1 \le i < j \le N$, and $k_i,k_j > 0$, (a "boundary edge")*

or

(ii) *an edge $(B(i),\overline{B}(i))$, for some i with $1\le i\le N$, where i is contained in a pair of S.*

B) *A cross-free element of $M(\emptyset;n_1,\ldots,n_N)$ contains an edge $(B(i),B(j))$, where $1\le i<j\le N$.*

Example: The set of paths shown in figure 5a contains a boundary edge between lines 1 and 2, and 3 edges of type (ii) on lines 3, 4, and 5.

Proof:

Since part B) is a special case of part A), we have to prove only the first part. We prove it by contradiction, assuming that there is no edge of type (i) or (ii).

If i is contained in a pair of S the edge of T leaving $B(i)$ must go to a different line, otherwise it would be of type (ii). If i is not contained in a pair of S and $k_i>0$ then there must be two edges in T incident with $B(i)$, but only one of them can go to a point on the same line, otherwise T would not be cross-free. Summarizing, we can say that for every line i with $k_i > 0$ there is an edge from the boundary point $B(i)$ to some point X_i on some other line $f(i) \neq i$ with $k_{f(i)} > 0$. Moreover, X_i is not a boundary point ($X_i \neq B(f(i))$ since we assumed that there is no boundary edge in T.

Now we construct the following polygonal trajectory, which consists alternately of two types of edges (see figure 6):

- of pieces of the N line segments ("segment edges"), and
- of edges of T ("across edges").

We choose some arbitrary i_0 with $k_{i_0} > 0$, and we start at the point X_{i_0} on line $i_1 := f(i_0)$. Then we go to $B(i_1)$ along the line i_1. Then we take the edge of T to X_{i_1} and set $i_2 := f(i_1)$. Then we go to $B(i_2)$, then to X_{i_2} on line $i_3 := f(i_2)$, and so on.

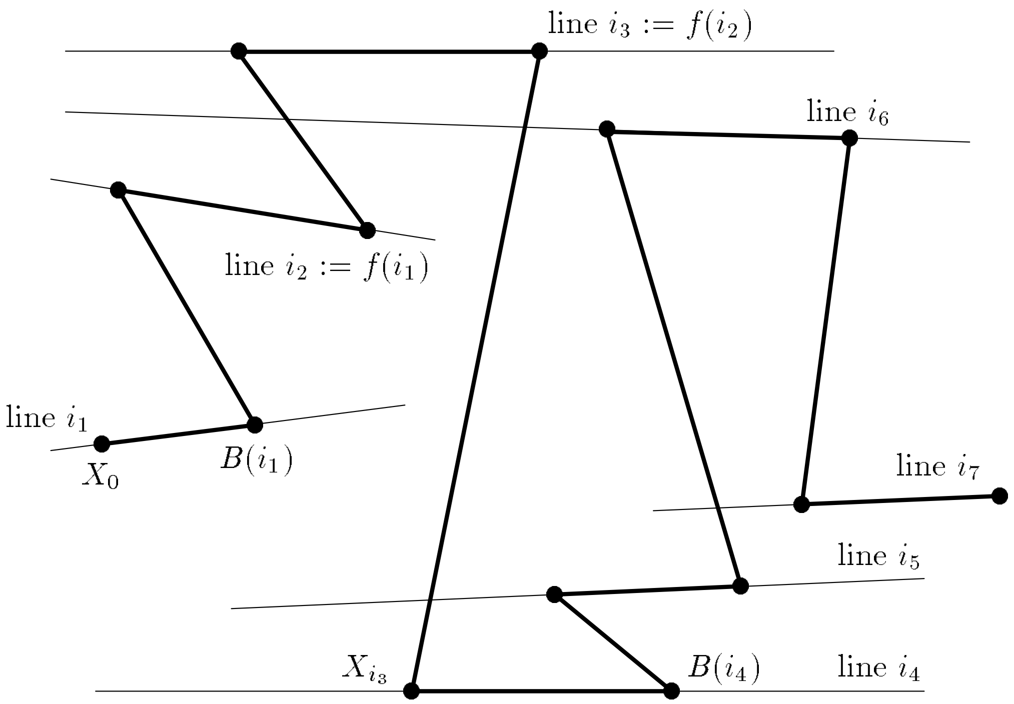

Figure 6. The polygonal trajectory

Since there is only a finite number of points the trajectory must intersect itself. Let Q be the first point where this happens. We claim that the simple polygon P formed by the trajectory between its first and second visit to Q consists alternately of pieces of segments and pieces of edges of T. Since this is true by construction for the whole trajectory, we just have to check the vicinity of Q: If the trajectory leaves Q for the first time along a "segment edge" and enters Q for the second time along an "across edge", or vice versa, the claim is true. Thus we only have to deal with the remaining two cases:

- The trajectory leaves Q for the first time along an "across edge" $(B(i), X_i)$ of T, and it enters Q for the second time along another "across edge" $(B(j), X_j)$ of T. The only possiblility how this could be consistent with the cross-free property of T is that Q is a common endpoint of the two edges. Q cannot be X_i because the trajectory has to *leave* Q via the edge $(B(i), X_i)$, and similarly, Q cannot be $B(j)$ because the trajectory *enters* Q via the edge $(B(j), X_j)$. The only remaining possibility, $Q = X_j = B(i)$, is excluded because a point X_j is never a boundary point.
- The trajectory leaves Q for the first time along a "segment edge" $(X_{i_k}, B(i_{k+1}))$, and it enters Q for the second time along a "segment edge" $(X_{i_l}, B(i_{l+1}))$. Two different line segments never cross, and therefore the two edges must lie on the same line $i_{k+1} = i_{l+1}$. In this case, the point Q, which equals X_{i_k}, is redundant as a vertex of P, and the claim is again true.

Now we can apply lemma 2: Let $k = f(k')$ and $l = f(l')$ be the top-most and bottom-most lines that are used by P (figure 4a). As one traverses P in the direction of the trajectory, one visits $B(k)$ immediately after $X_{k'}$ and after a while $X_{l'}$ and immediately afterwards $B(l)$, and then after a while one returns to $X_{k'}$. But this contradicts lemma 2. (If Q lies on line k or on line l, Q might have to take the place of $X_{k'}$ or $X_{l'}$ in this argument.) ■

When specialized to the case $N = 3$, lemma 3B is Cutler's "Triangle Theorem", and lemma 3A corresponds to his "Σ Theorem" (Cutler [1980], section 4).

Lemma 4.

A) *Let T be a shortest cross-free element of $M(S; k_1, \ldots, k_N)$ $(S \neq \emptyset)$. Then at least one of the following four cases holds (see figure 7):*

(i) *(For some $i \neq j$, T contains an edge $(B(i), B(j))$; $k_i, k_j > 0$:)*

a) $\{i, j\} \in S$:

$$\mathsf{length}(T) = \mathsf{dist}(B(i), B(j)) + \mathsf{length}(T'),$$

where $T' \in M(S - \{i, j\}\,; k_1, \ldots, k_i - 1, \ldots, k_j - 1, \ldots, k_N)$.

b) $\{i, q\} \in S$ *for some* q, *and* j *is not contained in a pair of* S:

$$\mathsf{length}(T) = \mathsf{dist}(B(i), B(j)) + \mathsf{length}(T'),$$

where $T' \in M(S - \{i, q\} + \{j, q\}\,; k_1, \ldots, k_i - 1, \ldots, k_N)$.

c) neither i *nor* j *is contained in a pair of* S:

$$\mathsf{length}(T) = \mathsf{dist}(B(i), B(j)) + \mathsf{length}(T'),$$

where $T' \in M(S - \{p, q\} + \{i, p\} + \{j, q\}\,; k_1, \ldots, k_N)$ *and* $\{p, q\}$ *is some arbitrary pair of* S.

(ii) *(For some i, T contains an edge $(B(i), \overline{B}(i))$; $k_i \geq 2$, and i occurs in S:)*

d) $$\mathsf{length}(T) = \mathsf{dist}(B(i), \overline{B}(i)) + \mathsf{length}(T'),$$

where $T' \in M(S; k_1, \ldots, k_i - 1, \ldots, k_N)$.

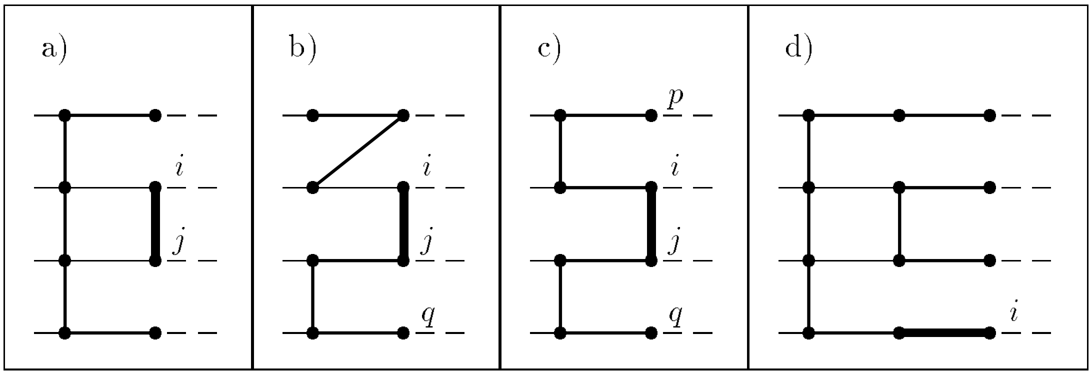

Figure 7. The four possible cases of lemma 4

B) *Let T be a shortest simple polygon in $M(\emptyset; n_1, \ldots, n_N)$. Then for some $i < j$, T contains an edge $(B(i), B(j))$, and we have:*

$$\mathsf{length}(T) = \mathsf{dist}(B(i), B(j)) + \mathsf{length}(T'),$$
$$\text{where } T' \in M(\{\{i, j\}\}; n_1, \ldots, n_N).$$

In all cases, T' is a cross-free element of the respective set.

Proof: The lemma follows by considering in each case the structure of the partial solution T' which is obtained from T by removing the edge whose existence is implied by the previous lemma. In case (i), the case that i and j are in different pairs of S cannot occur. ■

4. The algorithm

We can now set up a dynamic programming recursion corresponding to the equations in lemma 4 in a straightforward way, involving variables $d(S; k_1, \ldots, k_N)$ corresponding to the sets $M(S; k_1, \ldots, k_N)$.

The starting value of the recursion is $d(\emptyset; 0, \ldots, 0) := 0$.

Let $S = \{\{i_1, i'_1\}, \{i_2, i'_2\}, \ldots, \{i_m, i'_m\}\}$ and let $l_1, \ldots, l_{N-2m}$ be the indices that do not occur in a pair of S, in some fixed order. Then:

If $S \neq \emptyset$ and $k_{i_j} = 0$ or $k_{i'_j} = 0$ for some $1 \leq j \leq m$, then we set:

$$d(S; k_1, \ldots, k_N) := \infty.$$

Otherwise we set:

$$d(S; k_1, \ldots, k_N) := \min\{\Delta_a, \Delta_{b1}, \Delta_{b2}, \Delta_c, \Delta_{d1}, \Delta_{d2}\},$$

where

$$\Delta_a := \min\{\, \mathsf{dist}(B(i_j), B(i'_j)) + d(S - \{i_j, i'_j\}; k_1, \ldots, k_{i_j}-1, \ldots, k_{i'_j}-1, \ldots, k_N) \mid 1 \leq j \leq m \,\}$$

$$\Delta_{b1} := \min\{\, \mathsf{dist}(B(i'_j), B(l_s)) + d(S - \{i_j, i'_j\} + \{i_j, l_s\}; k_1, \ldots, k_{i'_j}-1, \ldots, k_N) \mid 1 \leq j \leq m,\ 1 \leq s \leq N-2m \,\}$$

$$\Delta_{b2} := \min\{\, \mathsf{dist}(B(i_j), B(l_s)) + d(S - \{i_j, i'_j\} + \{i'_j, l_s\}; k_1, \ldots, k_{i_j}-1, \ldots, k_N) \mid 1 \leq j \leq m,\ 1 \leq s \leq N-2m \,\}$$

$$\Delta_c := \min\{\, \mathsf{dist}(B(l_s), B(l_t)) + d(S - \{i_j, i'_j\} + \{i_j, l_s\} + \{i'_j, l_t\}; k_1, \ldots, k_N) \mid 1 \leq s, t \leq N-2m,\ s \neq t,\ 1 \leq j \leq m \,\}$$

$$\Delta_{d1} := \min\{\, \mathsf{dist}(B(i_j), \overline{B}(i_j)) + d(S; k_1, \ldots, k_{i_j}-1, \ldots, k_N) \mid 1 \leq j \leq m,\ k_{i_j} \geq 2 \,\}$$

$$\Delta_{d2} := \min\{\, \mathsf{dist}(B(i'_j), \overline{B}(i'_j)) + d(S; k_1, \ldots, k_{i'_j}-1, \ldots, k_N) \mid 1 \leq j \leq m,\ k_{i'_j} \geq 2 \,\}$$

(The minimum of an empty set is always taken to be ∞. This ensures that $d(\emptyset; k_1, \ldots, k_N)$ is ∞ unless $k = (0, \ldots, 0)$.)

Finally, as an exception to the above rule:

$$d(\emptyset; n_1, \ldots, n_N) := \min\{\, \mathsf{dist}(B(i), B(j)) + d(\{\{i, j\}\}; n_1, \ldots, n_N) \mid 1 \leq i < j \leq N \,\}.$$

We have to establish some order in which these recursions can be computed. We could for example compute the $d(S; k_1, \ldots, k_N)$ in increasing lexicographic order of $(k_1, \ldots, k_N)$. For equal $(k_1, \ldots, k_N)$, the values with larger cardinality of S are computed first. Another possibility would be to compute them in increasing order of $\sum k_i - |S|$, which is equal to the number of edges in the elements of $M(S; k_1, \ldots, k_N)$.

Example: $N = 5$; (a_{ij} denotes the j-th point on line i):

$$
d(\{\{1,3\}\};1,3,5,0,7) = \min \left\{ \begin{array}{ll}
\mathsf{dist}(a_{11},a_{35}) + d(\{\};0,3,4,0,7), & \text{(a)} \\
\mathsf{dist}(a_{11},a_{23}) + d(\{\{2,3\}\};0,3,5,0,7), & \text{(b)} \\
\mathsf{dist}(a_{11},a_{57}) + d(\{\{3,5\}\};0,3,5,0,7), & \text{(b)} \\
\mathsf{dist}(a_{23},a_{35}) + d(\{\{2,3\}\};1,3,4,0,7), & \text{(b)} \\
\mathsf{dist}(a_{35},a_{57}) + d(\{\{3,5\}\};1,3,4,0,7), & \text{(b)} \\
\mathsf{dist}(a_{23},a_{57}) + d(\{\{1,2\},\{3,5\}\};1,3,5,0,7), & \text{(c)} \\
\mathsf{dist}(a_{23},a_{57}) + d(\{\{1,5\},\{2,3\}\};1,3,5,0,7), & \text{(c)} \\
\mathsf{dist}(a_{35},a_{34}) + d(\{\{1,3\}\};1,3,4,0,7) & \text{(d)}
\end{array} \right\}
$$

The first line, corresponding to case (a), can be omitted in this case since $M(\emptyset;0,4,5,0,7)$ is empty and $d(\emptyset;0,4,5,0,7)=\infty$. ■

Lemma 5. *If $M(S;k_1,\ldots,k_N)\neq\emptyset$, then:*

$$
\begin{array}{r}\textit{length of the}\\ \textit{shortest element in}\\ M(S;k_1,\ldots,k_N)\end{array} \;\leq\; d(S;k_1,\ldots,k_N) \;\leq\; \begin{array}{l}\textit{length of the shortest}\\ \textit{cross-free element in}\\ M(S;k_1,\ldots,k_N)\end{array}
$$

Proof:

The left inequality is true since $d(S;k_1,\ldots,k_N)$ is always the length of some element from $M(S;k_1,\ldots,k_N)$, and the right inequality follows from lemma 4 by induction on the recursion. ■

Since for $M(\emptyset;n_1,\ldots,n_N)$ the left and right sides of lemma 5 are equal (by lemma 1) we get

Theorem 1. *$d(\emptyset;n_1,\ldots,n_N)$ is the length of the shortest tour.* ■

Remark: Some impossible cases could be excluded from the recursion. For example, if $|S|=1$ then case (a) need not be considered (like in the previous example) except at the very beginning. However, this would not reduce the running time substantially except for very small N.

5. Complexity analysis

Let us now analyze the complexity of the algorithm, both as regards space and time. Let P^N denote the number of different sets S, i. e., the number of sets of disjoint unordered pairs of $\{1,\ldots,N\}$. If we regard the pairs in such a set S as the cycles of a permutation, it is easy to see that P^N equals the number of permutations of N elements whose square is the identity, or equivalently, which are equal to their own inverse.

The numbers P^N can be computed by the recursion:

$$P^{N+1} = P^N + NP^{N-1} \qquad (N \geq 1) \tag{1}$$

from the start values $P^0 = P^1 = 1$. This recursion was given in Rothe [1800], p. 282. It can be proved by splitting the P^{N+1} sets S into those where the element $N+1$ is not contained in a pair and into those where it forms a pair with one of the N other elements. (The very same recursion, but with different start values, occurs in Gilmore, Lawler, and Shmoys [1985], p. 138, where it describes the number of equivalence classes of partial tours for the bandwidth-constrained Traveling Salesman Problem.)

The following table shows the first few values of P^N: (The meaning of the last column will be explained later.)

N	P^N	$\mathsf{TIME}'(N)$
1	1	0
2	2	3
3	4	15
4	10	72
5	26	300
6	76	1,290
7	232	5,418
8	764	23,520
9	2,620	102,672
10	9,496	461,700
11	35,696	2,107,380
12	140,152	9,876,768
13	568,504	47,127,600

Chowla, Herstein, and Moore [1951] proved the following asymptotic expression for P^N:

$$P^N \sim \left(\frac{N}{e}\right)^{N/2} e^{\sqrt{N}} \frac{1}{\sqrt[4]{4e}}.$$

Very roughly, P^N is $\sqrt{N!}$. A more exact approximation with additional terms of higher order was proved by Moser and Wyman [1955] (cf. also Knuth [1973], pp. 65–67).

The storage requirement for the algorithm is now

$$P^N \cdot \prod_{i=1}^{N} (n_i + 1) \,.$$

Thus, for fixed N, the storage requirement is

$$O\left(\prod_{i=1}^{N} (n_i + 1)\right) = O\left(\prod_{i=1}^{N} n_i\right) = O(n^N),$$

if we denote the total number of points by $n = \sum_{i=1}^{N} n_i$.

If n is fixed then the maximum of the left side in the above equation is achieved for $n_i = n/N$; thus, using the inequality $n_i + 1 \le 2n_i$, the constant of the O-notation in the expression $O(n^N)$ is at most

$$\left(\frac{2}{N}\right)^N \cdot P^N = \left(\frac{2}{N}\right)^N \left(\frac{N}{e}\right)^{N/2} e^{\sqrt{N}} \frac{1}{\sqrt[4]{4e}} = O\left(\frac{1}{(N/4)^{N/2}} \cdot \frac{1}{e^{N/2-\sqrt{N}}}\right),$$

which decreases quite fast as N increases.

For analyzing the time complexity, let us now establish the complexity of one step of the recursion: If S contains m pairs, then there are at most m, $2m(N-2m)$, $(N-2m)(N-2m-1)m$, and $2m$ terms corresponding to cases (a), (b), (c), and (d), respectively. Therefore, the total number of terms which are necessary for computing $d(S; k_1, \ldots, k_N)$ is the sum of these four expressions, which is $m(3 + N + N^2) + m^2(-4N - 2) + 4m^3$.

Let P_m^N denote the number of sets S containing exactly m disjoint unordered pairs of $\{1, \ldots, N\}$. It is equal to the number of permutations with m cycles of length 2 and $N-2m$ cycles of length 1. Therefore, we have

$$P_m^N = \frac{N!}{2^m m!\,(N-2m)!}\,, \qquad \text{for } 0 \le m \le N/2. \tag{2}$$

Neglecting the boundary cases, the time complexity is thus

$$\mathsf{TIME}(N) = \mathsf{TIME}'(N) \cdot \left(\prod_{i=1}^{N} (n_i + 1)\right),$$

where $\mathsf{TIME}'(N)$, the time for evaluating $d(S; k_1, \ldots, k_N)$ for all sets S for some fixed N-tuple $(k_1, \ldots, k_N)$, is given as follows:

$$\mathsf{TIME}'(N) = \sum_m P_m^N \cdot \left(m(3 + N + N^2) + m^2(-4N - 2) + 4m^3 \right). \tag{3}$$

(By observing that $P^N_m = 0$ for $m < 0$ or $m > N/2$, we may simplify matters by letting the summation index m vary over all integers.)

The terms involving the variable m in this sum can be eliminated by using the following formulas, which follow directly from (2):

$$\begin{aligned} P^N_m\, m &= \frac{N(N-1)}{2}\, P^{N-2}_{m-1} \\ P^N_m\, m(m-1) &= \frac{N(N-1)(N-2)(N-3)}{4}\, P^{N-4}_{m-2} \\ P^N_m\, m(m-1)(m-2) &= \frac{N(N-1)(N-2)(N-3)(N-4)(N-5)}{8}\, P^{N-6}_{m-3}. \end{aligned}$$

After this we can carry out the summation over m, using the identity $P^N = \sum_m P^N_m$, and express $\mathsf{TIME}'(N)$ in terms of N, P^{N-2}, P^{N-3}, ..., and P^{N-6}. Repeated application of the recursion (1) leads then to the following short expression:

$$\mathsf{TIME}'(N) = \frac{N(N-1)}{2}\left(P^N + P^{N-2}\right).$$

(The calculation is carried out in detail in the original version of this technical report Rote [1988].) The values of $\mathsf{TIME}'(N)$ are tabulated in the above table.

Thus, for fixed N, the time complexity is again

$$O\left(\prod_{i=1}^{N}(n_i+1)\right) = O\left(\prod_{i=1}^{N} n_i\right) = O(n^N).$$

Arguing as in the case of the storage requirement, we find that the constant of the O-notation in the expression $O(n^N)$ is at most

$$\begin{aligned} \left(\frac{2}{N}\right)^N \cdot \binom{N}{2}\left(P^N + P^{N+2}\right) &= O\left(\left(\frac{2}{N}\right)^N N^2 \left(\frac{N}{e}\right)^{N/2} e^{\sqrt{N}}\right) \\ &= O\left(\frac{1}{(N/4)^{N/2-2}} \cdot \frac{1}{e^{N/2-\sqrt{N}}}\right). \end{aligned}$$

We summarize our results in the following

Theorem 2. *The N-line Traveling Salesman Problem with n_1, n_2, ..., n_N points on the lines 1, 2, ..., N can be solved in space*

$$O\left(P^N \cdot \prod_{i=1}^{N}(n_i+1)\right)$$

and time

$$O\left(P^N \cdot N^2 \prod_{i=1}^{N}(n_i+1)\right),$$

where the numbers P^N are defined by the recursion (1). For fixed N, and for a total number of n points, the space and time complexities are thus

$$O(n^N).$$

■

6. Characterization of "quasi-parallel" line segments

In section 2, the following property was required of a set of line segments in order that our algorithm could be applied:

> No segment is perpendicular to the x-axis, and for any two segments which are not parallel the projection of the intersection of the two lines carrying the segments onto the x-axis lies always outside the projections of the two segments.

We call a set of line segments *quasi-parallel*, if they can be rotated in such a way that this property is fulfilled, in other words, if an appropriate x-axis can be found. In this section, we give a characterization of this property.

First we have to introduce some terminology:

By an *orientation* of a line we mean an assignment of the label "left" to one direction of the line and the label "right" to the other direction. An *oriented line* is a line together with an orientation. When we draw an oriented line the orientation will be indicated by an arrow pointing in the "right" direction.

By an *oriented segment* we mean a segment a with an orientation of the line $g(a)$ carrying the segment.

When we project an oriented line on another line which is not perpendicular to it we get a *corresponding orientation* of the second line in a natural way.

Now we call two oriented segments a and b *oriented consistently* if the intersection of the carrying lines $g(a)$ and $g(b)$ either lies both on the left side of a on $g(a)$ and on the left side of b on $g(b)$ or on the right side of a resp. b on both $g(a)$ and $g(b)$. (In case a and b are parallel, the orientations have to be the same in order to be consistent.)

We can now rephrase the above definition of quasi-parallelness in the terminology just introduced as follows:

> There is an oriented line l (this line corresponds to the x-axis in the previous formulation), not perpendicular to any segment, such that for the corresponding orientations of the segments obtained by projection from the orientation of l, any pair of non-parallel segments a and b is oriented consistently.

Theorem 3. *Let a set of at least 3 segments be given, and let g_0 be any fixed segment of this set. Then the set of segments is quasi-parallel if and only if every subset of three segments containing g_0 is quasi-parallel.*

Proof: First of all, it is clear that consequence (i) of section 2 holds, i. e., the line carrying a segment does not intersect any other segment, and in particular, the segments themselves do not intersect.

Now fix any orientation of g_0. Then there is only one possible orientation of every other segment which is consistent with the orientation of g_0.

We have to show two things:

(a) There is an oriented line l such that the orientations thus constructed correspond by projection to the orientation of l.

(b) Every pair of non-parallel segments a and b (different from g_0) is also oriented consistently.

For proving (b), let's assume that two segments a and b are not oriented consistently. Then the orientation of b resulting from the orientation of g_0 is different from the orientation of b resulting from the orientation of a resulting from the orientation of g_0; thus, even for the 3-set $\{g_0, a, b\}$, it would be impossible to orient the segments consistently.

Now we still have to prove (a). For each segment the possible oriented directions of l form an open half-circle (cf. figure 8). We know that the half-circle belonging to g_0 has a non-empty intersection with every two other half-circles. We intersect the other half-circles with the half-circle belonging to g_0 and consider only the results, as parts of the half-circle belonging to g_0, or, equivalently, as intervals. We know that any two of these intervals have a non-empty intersection. From this it follows, by Helly's Theorem, that the intersection of all intervals is non-empty. Therefore there is a possible orientation of l yielding the constructed orientations of the segments. ■

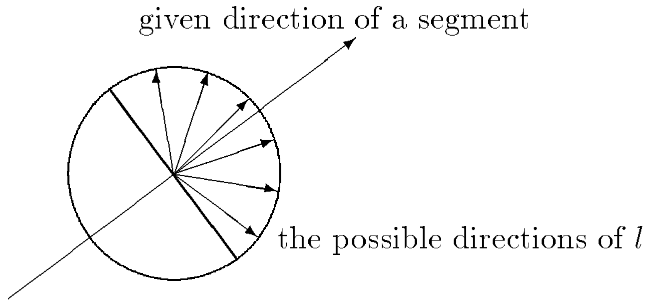

Figure 8.
The possible directions of l for
a given direction of a segment

An $O(N^2)$-algorithm for testing whether N line segments are quasi-parallel follows in a straightforward manner. It is possible to reduce this time to $O(N \log N)$, but since this is by far surpassed by the complexity of solving the problem, this is not so interesting.

By a simple case analysis, one can determine all configurations of three or fewer line segments which are not quasi-parallel, and thus one obtains the following corollary:

Corollary. *A set of segments is quasi-parallel if and only if it does not contain a subset of segments which looks like one of the seven* types of configurations in figure 3.* ■

---

* *In the original version of this paper I had only six types of configurations. I thank Gerhard Woeginger for pointing out that the configuration in figure 3c was missing.*

7. Conclusion

From a higher standpoint our approach to the N-line Traveling Salesman Problem can be viewed in the following way: A partial tour was defined to be a certain subset of the edges of a tour. We have grouped the partial tours into equivalence classes with respect to the edge sets which can be added to complete the tour: Two partial tours T_1 and T_2 are equivalent if, for all sets T', $T_1 \cup T'$ is a tour if and only if $T_2 \cup T'$ is a tour. This allowed us to forget all partial solutions except the best one in each equivalence class. With little effort, we have now been able to compute the best partial solution in each equivalence class in a systematic way.

Exactly the same paradigm has been followed by Ratliff and Rosenthal [1983] in their solution of the order-picking problem, and by Gilmore, Lawler, and Shmoys [1985] for the bandwidth-limited problem.

It is in principle not difficult to extend this approach to the Taveling Salesman *Path* problem, which requires to find a shortest, not necessarily closed curve containing the given set of points. It is necessary to modify the notion of a partial solution and to establish a corresponding analog of lemma 3.

Our algorithm also applies to Traveling Salesman Problems in the plane with other metrics than the Euclidean distance, for example the L_1 metric (Manhattan metric) and the L_∞ metric (maximum metric). These metrics are important for the class of manufacturing problems that were mentioned in the introduction. In fact, the only property of the Euclidean distance that we have used is the triangle inequality, which was necessary for the cross-free property of lemma 1. It is clear that there is always an optimal tour such that the corresponding polygon is a simple polygon, for any symmetric distance function that fulfills the triangle inequality. Thus, the only thing in the algorithm that has to be changed is the computation of $\mathsf{dist}(X, Y)$.

A question which we have not pursued so far is the following: Can our results be applied to construct heuristic algorithms in cases when there is no set of few parallel straight lines containing the points? One might only require that the points lie in the vicinity of the lines; or one might reduce the number of partial solutions by excluding "unlikely" sets $P(S; k_1, \ldots, k_N)$, whose boundary points are distributed in a wide range between the left-most and the right-most extremes, thus speeding up the algorithm.

8. References

S. Chowla, I. N. Herstein, and W. K. Moore [1951]
On recursions connected with symmetric groups I, *Canad. J. Math.* **3**, 328–334.

N. G. de Bruijn [1955]
Solution to exercises 17 and 18 (in Dutch), in: *Wiskundige opgaven met de oplossingen* **20**, pp. 19–20.

G. Cornuéjols, J. Fonlupt, and D. Naddef [1985]
The Traveling Salesman Problem on a graph and some related integer polyhedra, *Mathematical Programming* **33**, 1–27.

M. Cutler [1980]
Efficient special case algorithms for the N-line planar Traveling Salesman Problem, *Networks* **10**, 183–195.

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

V. G. Deĭneko and G. Woeginger [1996]
The convex-hull-and-k-lines traveling salesman problem, *Information Processing Letters* **59**, 259–301.

P. C. Gilmore, E. L. Lawler, and D. B. Shmoys [1985]
"Well-solved special cases", chapter 4 of the book *The Traveling Salesman Problem*, ed. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Wiley & Sons, pp. 87–143.

A. Itai, C. H. Papadimitriou, and J. Szwarcfiter [1982]
Hamiltonian paths in grid graphs, *SIAM J. Computing* **11**, 676–686.

D. S. Johnson and C. H. Papadimitriou [1985]
"Computational complexity", chapter 3 of the book *The Traveling Salesman Problem*, ed. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Wiley & Sons, pp. 37–85.

D. E. Knuth [1973]
The Art of Computer Programming, vol. 3: Sorting and Searching, Addison-Wesley.

L. Moser and M. Wyman [1955]
On solutions of $x^d = 1$ in symmetric groups, *Canad. J. Math.* **7**, 159–168.

C. H. Papadimitriou [1977]
The Euclidean TSP is NP-complete, *Theoret. Comput. Sci.* **4**, 237-244.

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

G. Rote [1988]
The N-line Traveling Salesman Problem, Bericht 1988-109, Technische Universität Graz, Institut für Mathematik, January 1988.

H. A. Rothe [1800]
Über Permutationen, in Beziehung auf die Stellen ihrer Elemente. Anwendung der daraus abgeleiteten Sätze auf das Eliminationsproblem, in: *Sammlung combinatorisch-analytischer Abhandlungen, 2. Sammlung*, ed. C. F. Hindenburg. Leipzig, bei Gerhart Fleischer dem Jüngeren, pp. 263–305.