Report No. 95-199

The Euclidian Traveling Salesman Selection Problem

by

Anja Hamacher and Christoph Moll

1995

in "Proceedings of the SOR '95"

Zentrum für Paralleles Rechnen
Universität zu Köln
Weyertal 80
D-50931 Köln
E–Mail:{hamacher,moll}@zpr.uni-koeln.de

1991 Mathematics Subject Classification: 90B06, 90C27, 68R05, 90C39
Keywords: TSP, branch&bound, Lin–Kernighan, geometry, dynamic programming

The Euclidian Traveling Salesman Selection Problem

Anja Hamacher and Christoph Moll

November 9, 1995

1 Introduction

The traveling salesman problem (TSP) is one of the famous problems of combinatorial optimization[1]. In this paper a generalization of the TSP, the **traveling salesman selection problem** (TSSP) is introduced. The problem is restricted to the Euclidian case where the TSP can be formulated as follows: Given n cities in the plane and their Euclidian distances, the problem is to find the shortest TSP-tour, i.e. a closed path visiting each of the n cities exactly once.

In addition to that the TSSP specifies a number $k < n$ of cities and the shortest TSP–tour through any subset of k cities shall be found. Existing heuristics are based on approximations for the k–Minimal Spanning Tree [1] to find the node cluster containing the shortest k–tour. Unlike many related problems [3, 7], the TSSP does not include a depot that has to be visited by the k–tour.

The main purpose of this paper is the presentation of an exact algorithm for the TSSP. It is based on a geometrical procedure searching for all clusters of about k nodes which can contain the shortest k–tour. These clusters are the start set for a branch&bound procedure that determines the shortest k–tour in each cluster and therefore the exact solution of the TSSP. A dynamic programming approach for calculating shortest t–chains is used to obtain a lower bound for the branch&bound algorithm.

The paper is organized as follows. Section 2 introduces the main definitions. In section 3 a simple start heuristic is proposed to calculate an upper bound for the problem. Section 4 summarizes some improvement heuristics. The clustering procedure is presented in section 5, and section 6 describes the branch&bound algorithm.

[1] A survey is given in [10], other articles treating the TSP are [4, 5, 6, 13]

2 Definitions

Let S be a set of n nodes in the Euclidian plane representing n cities. For a set $R \subseteq S$, $\|S\|$ denotes the cardinality of R.

A *k–tour* T consists of nodes $\mathcal{V}$(T)$\subseteq$ S and edges $\mathcal{E}$(T)$\subseteq$ S $\times$ S with $\|\mathcal{V}(T)\| = k$. Its length $\mathcal{L}$(T) is calculated by $\mathcal{L}(T) = \sum_{(x,y)\in\mathcal{E}(T)} \text{dist}[x,y]$, where *dist[a,b]* denotes the Euclidian distance between the nodes a and b. T' is called a *subtour* of T if $\mathcal{V}$(T') $\subseteq \mathcal{V}$(T) and the common nodes are visited in the same order.

An *n–chain* is a sequence of n edges where successive edges are adjacent, edge and node repetitions are allowed. An *n–circle* is a closed n–chain.

3 Generating a start solution

The clustering algorithm described in section 5 requires an upper bound on the shortest k–tour. Therefore it is useful to have a simple start heuristic to find an approximative solution in a short time.

An obvious approach is to search for an area where k nodes are located closed together. Therefore the node is determined where the sphere around this node containing k nodes is minimal. Then starting from this node one of the insertion heuristics known from the TSP (cp. [6, 13]) can be employed to build the k–tour. This start heuristic can also be applied to the Non–Euclidian TSSP.

Remark 1 *Standard TSP start heuristics are easily transfered [8] and the quality and complexitiy properties for the TSSP correspond to those of the TSP. If the set of nodes from which the k–tour is generated consists of more than k nodes, the farthest insert algorithm has to be slightly modified because the farthest node in general is not a good candidate for building a short k–tour.*

4 Tour improvement heuristics for the TSSP

For the TSP the r–opt heuristic [12] and its generalization, the Lin–Kernighan heuristic [11], are known as tour improvement algorithms. Both heuristics can be applied to the TSSP without any modification. However it is desirable to have an improvement heuristic that could realize a node exchange which inserts new nodes into the k–tour. We modified the Lin–Kernighan heuristic to allow edge exchanges that insert new nodes and delete others from the k–tour as follows.

The basic idea of the Lin–Kernighan heuristic for the TSP remains unchanged: a sequence of edge exchanges leads to a shorter tour. The insertion of an edge adjacent to a non–tour node requires the deletion of another node from the tour. The heuristic deletes the node which permits the highest saving value. A detailed description of the modified heuristic is given in [8].

5 Constructing the clusters

This section describes a geometrical procedure for generating clusters of about k nodes containing short k–tours. The resulting sets of nodes are the start sets for the further described branch&bound algorithm. For a tour T a pair of nodes with maximum distance in the tour is called a diameter of T. The idea of the algorithm is to classify the k–tours by their diameter (a, b) and the node c that together with a and b builds the longest 3–subtour of the tour. By exploiting some geometrical and combinatorial arguments the algorithm determines all set of nodes that can contain k–tours shorter than a given upper bound $\overline{U}$.

Definition 1 *(pastille $P_{a,b}$)*
Let $a, b \in S, a \neq b$. We define a **pastille spanned by a and b** *by:*

$$P_{a,b} := \{x \in S \mid (dist\,[x,a] \leq dist\,[a,b]) \text{ and } (dist\,[x,b] \leq dist\,[a,b])\}. \qquad (1)$$

A tour T **belongs to the pastille** $P_{a,b}$ *iff* $(a, b \in \mathcal{V}(T))$ *and* $(\mathcal{V}(T) \subseteq P_{a,b})$.

A pastille $P_{a,b}$ consists of all nodes that could belong to a tour with diameter (a, b). Each k–tour T in S belongs to at least one pastille $P_{a,b}$, in particular the $P_{a,b}$ where (a, b) is a diameter of T. The nodes of $P_{a,b}$ are located in the intersection of the two circles around a and b with radius dist$[a, b]$.

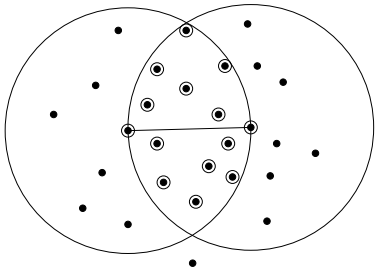

Figure 1: geometrical representation of a pastille

The following properties are used in the algorithm to find candidates among the pastilles for clusters of about k nodes containing short k–tours.

- Since a tour consists of pairwise distinct nodes, there must be at least k nodes in $P_{a,b}$ and every pastille containing fewer than k nodes can be neglected.
- If the nodes $c \in P_{a,b} \setminus \{a, b\}$ are sorted in ascending order of the lengths $l_c := \mathcal{L}(a - b - c - a)$ in the sequence $[c_i]$, a k–tour T in $P_{a,b}$ consists of a, b, and $k - 2$ additional nodes of the ordered sequence $c_1 ... c_{||P_{a,b}||-2}$. At least one of the nodes $c_{k-2} ... c_{||P_{a,b}||-2}$ must belong to T. Therefore $l_{c_{k-2}}$ is a lower bound for any k–tour in $P_{a,b}$.

- Each pastille with $l_{c_{k-2}} > \overline{U}$ contains only k–tours longer than the upper bound $\overline{U}$.

These properties base on the triangle inequality that holds in the Euclidian plane. In the following only pastilles with at least k nodes and $l_{c_{k-2}} \leq \overline{U}$ are dealt with.

Definition 2 *(triple $T_{a,b,c}$)*
Let $a, b \in S$, $a \neq b$. Let $P_{a,b}$ be a pastille. Let $c \in P_{a,b} \setminus \{a, b\}$.
We define a **triple spanned by a, b and c** *by:*

$$T_{a,b,c} := \{x \in P_{a,b} \mid (l_x \leq l_c) \text{ and } (dist[x,c] \leq dist[a,b])\}. \tag{2}$$

A tour T **belongs to the triple** *$T_{a,b,c}$ iff $(a, b, c \in \mathcal{V}(T))$ and $(\mathcal{V}(T) \subseteq T_{a,b,c})$.*

Remark 2 *The location of the nodes of $T_{a,b,c}$ can be described analogous to the representation of the pastilles. All nodes $x \in T_{a,b,c}$ are located in the ellipse with focal points a and b through c. Note that l_x corresponds to the sum of distances from node x to the focal points a and b (up to the constant term $dist[a,b]$) and that all interior points y of the ellipse satisfy $l_y \leq l_c$ because c is a point on the border of the ellipse. The starting point of the representation is the pastille $P_{a,b}$. Then the ellipse with focal points a and b through c is constructed. Since $\forall x \in T_{a,b,c} : dist[x,c] \leq dist[a,b]$, the nodes are also located in the circle with center c and radius $dist[a,b]$. Thus the nodes of $T_{a,b,c}$ can be found in the intersection of the two circles of $P_{a,b}$, the ellipse, and the circle described above.*

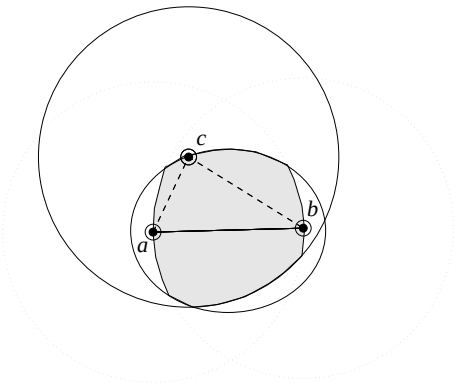

Figure 2: geometrical representation of a triple

The following algorithm determines triples consisting of about k nodes and a 3–tour $a - b - c - a$. These triples contain all k–tours shorter than an upper bound $\overline{U}$, for example determined by the start heuristic.

Algorithm 1 *:*
Let $\overline{U}$ be an upper bound for the shortest k–tour in S.

1. *Determine $P_{a,b}$ for all pair of nodes $(a,b) \in S \times S$ with $2 \cdot dist[a,b] \leq \overline{U}$.*

If $\|P_{a,b}\| \geq k$:

2. *Construct the sequence $[c_i]$ as described above.*

If $l_{c_{k-2}} \leq \overline{U}$:

3. *Determine $\overline{i} := \max\{i \mid l_{c_i} \leq \overline{U}\}$.*

4. *Determine the triple T_{a,b,c_i} for each $i \in \{k-2, ..., \overline{i}\}$.*

If $\|T_{a,b,c_i}\| \geq k$:

5. *Search for the shortest k–tour in T_{a,b,c_i}.*

For all $i > \overline{i}$ the 3–tour T=$a - b - c_i - a$ has a length $l_{c_i} > \overline{U}$. Thus all tours in T_{a,b,c_i} are longer than $\overline{U}$ because they contain T as subtour. For those c_i with $i < k-2$ the triples T_{a,b,c_i} contain at most $k-1$ nodes so that no k–tour belongs to T_{a,b,c_i}. The constructed triples consist of a 3–tour and of all nodes that could be inserted to construct a k–tour shorter than $\overline{U}$. The 3–tour can be used as a start tour for an insert–heuristic to get an approximative solution or for the branch&bound algorithm to obtain an exact solution.

6 A branch&bound algorithm for solving the TSSP

The root of the branch&bound(b&b) tree is formed by one of the triples generated by the geometric construction of algorithm 1. The lower bound is calculated by a dynamic programming approach determining the shortest k–circle.

6.1 Branching in the TSSP

The basic idea of branching in the TSSP is the insertion of a selected node at every "possible" position.
A **b&b node** consists of

- an m–tour T ($m \leq k$),
- a global upper bound $\overline{U}$ and a lower bound $\underline{L}$ (cp. section 6.2) for all k–tours that can be constructed from this b&b node,
- a set $M \subseteq S$ of insertable nodes with:

1. $\|M\| \geq k - m$, otherwise there are not enough nodes to build a k–tour.
2. For each node there must be a position in T where it could be inserted validly, i.e. the resulting tour length must remain less than $\overline{U}$.

If the set M consists of exactly $k - m$ nodes the following branching steps can be simplified. Each node fitting at only one position is inserted exactly there because this must be the position where it appears in the shortest k–tour containing T. For this argument the triangle inequality is essential.

6.2 Bounding in the TSSP

The b&b algorithm needs an upper and a lower bound to decide when to cut off the branches. The upper bound is determined by the shortest k–tour yet found. For a given m–tour T $= u_1 - ... - u_m (-u_{m+1} = u_1)$ the lower bound is determined by the shortest k–circle containing T as a subtour. Each edge $[u_i,u_{i+1}]$ of T is replaced by a t–chain from u_i to u_{i+1}. The following algorithm represents a dynamic programming approach for calculating the shortest t–chains between two nodes i and j recursively:

Definition 3 *SC(i,j,t) := length of the shortest t–chain from i to j*

Algorithm 2 *:*

$$SC(i,j,1) := dist[i,j] \ \forall i,j = 1..n$$
$$SC(i,j,t) := \min_{h \in \{1..n\} \setminus \{i,j\}} \{SC(i,h,t\text{-}1) + dist[h,j]\} \ \forall i,j = 1..n,\ 2 \leq t \leq K(i,j)$$

The parameter $K(i,j)$ is given as follows:

$$K(i,j) := \| \{x \in M \mid \mathcal{L}(T) + \text{dist}[i,x] + \text{dist}[x,j] - \text{dist}[i,j] \leq \overline{U}\} \|.$$

$K(i,j)$ determines the maximum number of nodes that can be inserted between i and j to build a k–tour shorter than $\overline{U}$.

To get the shortest k–circle containing an m–tour T, all possibilities of combining chains of various length to a k–circle have to be considered. These possibilities are given by all combinations of m positive numbers s_i with $\sum_{i=1..m} s_i = k$.

For most of the triples determined by algorithm 1 the first lower bound calculated in the b&b procedure is greater than the actual best solution, so that only for a small number of triples the branching procedure has to be started. Detailed computational results are presented in [8].

References

[1] Baruch Awerbuch, Yossi Azar, Avrim Blum, Santosh Vempala, *Improved approximation guarantees for minimum–weight k–trees and prize–collecting salesmen,* Technical Report CMU–CS–94–173, School of Computer Science, Carnegie Mellon Univ., Pittsburgh 1994

[2] Bronstein–Semendjajew, *Taschenbuch der Mathematik,* Verlag Harri Deutsch, Thun und Frankfurt/Main, p 273

[3] Steven E. Butt, Tom C. Cavalier, *A Heuristic for the Multiple Tour Maximum Collection Problem,* IMSE Working Paper 91-150

[4] Nicos Christofides, *The Travelling Salesman Problem,* Report 77–11, Department of Management Science, Imperial College, London

[5] Domschke, *Logistik: Transport (Bd.1), Rundreisen und Touren (Bd.2),* R. Oldenburg Verlag München Wien

[6] B. Golden, L. Bodin, T. Doyle, W. Stewart Jr., *Approximate Traveling Salesman Algorithms,* Operations Research 28 (1980), pp 694–711

[7] B. Golden, T.L. Magnanti, H.Q. Nguyen, *Implementing Vehicle Routing Algorithms,* Networks 7 (1977), pp 113–148

[8] Anja Hamacher, *Numerische Behandlung des "Traveling Salesman"–Auswahl–Problemes,* Diplomarbeit, Mathematisches Institut, Universität zu Köln 1992/93

[9] M. Held, R. Karp, *The Traveling Salesman Problem and Minimum Spanning Trees (part 1 and 2),* Operations Research 18 (1970), pp 1138–1162 and Mathematical Programming 1 (1971), pp 6–25

[10] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan, D.B. Shmoys, *The Traveling Salesman Problem,* Wiley–Interscience Series in Discrete Mathematics

[11] S. Lin, B.W. Kernighan, *An Effective Heuristic Algorithm for the Traveling–Salesman Problem,* Operations Research 21 (1973), pp 498–516

[12] Shen Lin, *Computer Solutions of the Traveling Salesman Problem,* The Bell System Technical Journal (Dezember 1965), pp 2245–2269

[13] Daniel J. Rosenkrantz, Richard E. Stearns, Philip M. Lewis, *An Analysis of several Heuristics for the Traveling Salesman Problem,* SIAM Journal Comput., Vol.6 (1977), pp 563–581

[14] Ton Volgenant, Roy Jonker, *A branch and bound algorithm for the symmetric traveling salesman problem based on the 1–tree relaxation,* European Journal of Operational Research 9 (1982) pp 83–89