Computing Shortest Heterochromatic Monotone Routes

J.M. Díaz-Báñez[1⋆], G. Hernández[2 ⋆⋆], D. Oliveros[3], A. Ramírez-Vigueras[4], J. A. Sellarès[5⋆⋆⋆], J. Urrutia[6†], and I. Ventura[7‡]

[1] Departamento de Matemática Aplicada II, Universidad de Sevilla, Spain
dbanez@us.es
[2] Facultad de Informática, Universidad Politécnica de Madrid, Spain
gregorio@fi.upm.es
[3] Instituto de Matemáticas, Universidad Nacional Autónoma de México, Mexico
dolivero@matem.unam.mx
[4] Instituto de Matemáticas, Universidad Nacional Autónoma de México, Mexico
aramirezv@uxmcc2.iimas.unam.mx
[5] Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain,
sellares@ima.udg.es
[6] Instituto de Matemáticas, Universidad Nacional Autónoma de México, Mexico
urrutia@matem.unam.mx
[7] Departamento de Matemáticas. Universidad de Huelva, Spain
inmaculada.ventura@dmat.uhu.es

Abstract. Given a set of n points on the plane colored with $k \leq n$ colors, the *Trip Planning Problem* asks for the shortest path visiting the k colors. It is a well-known NP-hard problem. We show that under some natural constraints on the path, the problem can be solved in polynomial time.

Keywords: Routing; Heterochromatic Monotone Route; Shortest Path; Computational Geometry.

⋆ Corresponding author. José Miguel Díaz-Báñez. Postal address: Escuela Técnica Superior de Ingenieros. Camino de los Descubrimientos, s/n, 41092, Sevilla, Spain. Research supported by Spanish MEC grant MTM2006-03909.
⋆⋆ Research supported Project CAM P-DPI-000235-0505.
⋆⋆⋆ Research supported by Spanish MEC grant TIN2007-67982-C02-02.
† Research supported in part by Spanish MEC grant MTM2006-03909 and CONACYT of Mexico, Proyecto SEP-2004-Co1-45876.
‡ Research supported by Spanish MEC grant MTM2006-03909.

1 Introduction

Computing shortest paths in numerous geometrical scenarios is a fundamental problem in computational geometry [10]. Perhaps one of the most famous problems is the *Geometric Traveling Salesman Problem*, *GTSP*, where a set P of n points (sites) in general position on the plane is given, and one wants to find a shortest tour that visits all the elements of P. It is well known that the *GTSP* problem is NP-hard [7, 13]. One variation of the *GTSP* is the *Trip Planning Problem* (*TPP*) [9]: Given P, a set of n points on the plane such that each element of P is colored with one of k given colors, $k \leq n$, a starting point S and a destination point E, find a shortest tour that starts at S, ends at E, and visits one point of each color. Such a tour is called *heterochromatic* since each of its elements has a different color. The *TPP* is a generalization of the *GTSP* (when all the elements of P have different colors) and thus it is NP-hard. There are many papers on closely related problems, some of which are known as *one-of-a-set TSP* or *group TSP* where one is given a collection of sets of points and the aim is to visit at least one point from each set; see [10] for a recent overview. Other problems related to ours are the so-called *errand scheduling problem* [14] and the *TSP with neighborhood* [5, 3, 11].

The *Trip Planning Problem* has its origins in many applications such as *route location* (e.g. someone is traveling from one place in town to another, and along the way has to visit various points such as a bank, a supermarket, a newspaper stand, etc.), *advanced internet systems* (such as navigation maps, or task planning in Google), or *computer networks* (e.g. given a computer network and a set of jobs such that each job should be executed only by a specific set of nodes in the network, find the shortest path that visits one node in each category).

In many instances, restrictions on the order in which we want to visit sites impose additional restrictions on the *TPP*. For example, in the first of the above examples, we would like to first visit a bank (to have money to spend), then a newspaper stand, next a supermarket (to buy some items for our visit to the beer garden) and finally the beer garden. We say that a route is *ordered* if we have to visit some sites of our point set according to a specific ordering.

Finally, in many instances we also need to ask for a path to have some desirable properties such as little turning or backtracking; i.e. we would like to have *monotone paths with respect to a direction* θ. Monotonicity has been widely used in modeling transportation and robotic problems in which non-momotonic paths increase or make calculation of the paths very expensive [1, 4]. Furthermore, monotone pieces reflect the monotonicity implicit in many data sets [15].

Let P be a point set with n elements in general position on the plane such that each element of P is colored with one of k colors, $k \geq 4$. Denote by n_i the number of points of color i, $i = 1, \ldots, k$; $n_1 + n_2 + \ldots + n_k = n$. From now on we will assume that we are dealing with *monotone ordered chains*, and that for each chromatic class i, its elements are labelled $\{p_1^i, \ldots, p_{n_i}^i\}$ such that if $l < m$, then the x-coordinate of p_l^i is smaller than or equal to that of p_m^i. We will also assume that our ordered paths must visit the color classes in the order $1, \ldots, k$.

A polygonal chain C with vertices $p_1, \ldots, p_s$ is a sequence of vertices and line segments $\{p_1, \ell_1, \ldots, \ell_{s-1}, p_s\}$ such that ℓ_i is the line segment connecting p_i to p_{i+1}, $i = 1, \ldots, s-1$. C is called x-monotone if any vertical line intersects C in a connected set. Similarly, a polygonal C is monotone with respect to a direction $\theta \in [0, \pi)$ if any line orthogonal to θ intersects C in a connected set. The length of C is the sum of the lengths of $\ell_i, \ldots, \ell_{s-1}$. In this paper we consider the following variants of the *TPP*:

MOOP (Monotone Ordered Oriented Polygonal Chain Problem): Given a k-colored point set P of n points in the plane in general position, find the shortest x-monotone path, if it exists, that visits a point of each color class in order $1, \ldots, k$.

MOP (Monotone Ordered Polygonal Chain Problem): Given a k-colored point set P of n points in the plane in general position, find a direction $\theta \in [0, \pi)$ such that the optimal path for the problem **MOOP** with orientation θ is the shortest among all the possible θ-monotone routes for all $\theta \in [0, \pi)$.

Note that under above restrictions (order and monotony), a solution for these problems may not exist. For example, Fig. 1 shows a configuration of points for which there is no monotone polygonal (in any orientation) that visits the four categories in the prefixed order (*cross, square, disk, circle*).

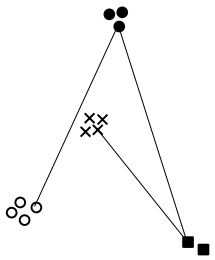

Fig. 1. There is no heterochromatic ordered monotone path for the order (*cross, square, disk, circle*).

In Section 2 of this paper we present an efficient $O(n \log^2 n)$-time and $O(n)$-space algorithm for **MOOP**. Next, in Section 3, we show how to modify our algorithm so that it solves the **MOP** in $O(n^3 \log n)$-time and $O(n^2)$-space. Section 4 concludes the paper with some remarks and future work.

2 Monotone, Oriented and Ordered Polygonal Chain

In this section we shall consider x-monotone polygonal chains. We say that a point p is to the left (resp. right) of a point q if the x-coordinate of p is smaller (resp. greater) than that of q. The distance between p and q will be denoted as $d(p, q)$. A polygonal chain C will be denoted by the sequence $p_i, \ldots, p_s$ of its vertices, ordered according to the order of their occurrence along C.

Before describing our method, we present the following properties which are exploited by our algorithm:

Lemma 1. *Let C be an x-monotone polygonal chain with vertices $p^1_{i_1}, p^2_{i_2}, \ldots, p^{k-1}_{i_{k-1}}, p^k_{i_k}$ of minimum length. Then $p^k_{i_k}$ is the closest point of color k to the right of $p^{k-1}_{i_{k-1}}$.*

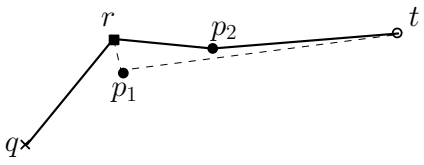

Fig. 2. p_2 belongs to the solution but it is not the closest disk to the right of r.

The proof of this lemma is straightforward and is left to the reader.

We remark that Lemma 1 does not necessarily hold for points $p_{i_j}^j$, $j < k$, as shown in Figure 2. Nevertheless Lemma 1 is important, since as we will see shortly, it allows us to find an optimal solution to the **MOOP** using dynamic programming.

Lemma 2. *Let C be an x-monotone polygonal chain with vertices $p_{i_1}^1, p_{i_2}^2, \ldots, p_{i_{k-1}}^{k-1}, p_{i_k}^k$ of minimum length. Then the heterochromatic x-monotone ordered polygonal chain of minimum length that visits colors $1, 2, \ldots, j$, $j \leq k$, and ends at $p_{i_j}^j$ has vertices $p_{i_1}^1, \ldots, p_{i_j}^j$.*

Proof. The proof is by contradiction. Suppose there exists another polygonal chain C' that visits colors $1, \ldots, j$ and ends at $p_{i_j}^j$. Let $q_{r_1}^1, \ldots, q_{r_{j-1}}^{j-1}, p_{i_j}^j$ be the vertex set of C'. If the length of C' is less than the length of the subpath of C containing $p_{i_1}^1, \ldots p_{i_j}^j$, then the length of the path with vertex set $q_{r_1}^1, \ldots, q_{r_{j-1}}^{j-1}, p_{i_j}^j, \ldots, p_{i_k}^k$ is less than the length of C, which is a contradiction.

Lemma 2 and Lemma 1 suggest the use of dynamic programming and planar point location to solve the **MOOP**. The main idea is as follows:

For each p_r^j, $1 < j \leq k$, we calculate l_r^j, the length of the shortest x-monotone route visiting colors $1, \ldots, j$ and ending at p_r^j. Let $p_{i_1}^1, \ldots, p_{ij-1}^{j-1}, p_r^j$ be the vertices of such a path. Then we create a pointer, $prev(p_{i_j}^j)$, pointing to $p_{i_{j-1}}^{j-1}$. Given the set of values $\{l_i^j, \ldots, l_{i_j}^j\}$, we can calculate, by Lemma 1, the label of a point of color $j+1$ to the right of p_r^j, say p_s^{j+1}, by using the following formula:

$$l_s^{j+1} = \min_r \{d(p_s^{j+1}, p_r^j) + l_r^j\}.$$

This expression suggests the use of additively weighted Voronoi (AW-Voronoi) diagrams, by using for each point p_r^i the weight l_r^i. We remark that in the AW-Voronoi each point p has a weight $w(p)$ assigned to it, and it is characterized by the weighted distance $d(p, p_i) + w(p)$ [12]. For our purposes, $w(p_r^j)$ will be l_r^j. Thus we propose a method consisting of $k-1$ steps denoted by $STAGE(i)$, $i = 2, \ldots, k$. At each stage, we calculate the labels $l_1^i, \ldots, l_{n_i}^i$ using $l_1^{i-1}, \ldots, l_{n_{i-1}}^{i-1}$ for all the points of color i and we calculate the labels and the pointers of points of color i. Initially the weight l_r^1 of all points of color 1 is zero. We now describe the algorithm in detail:

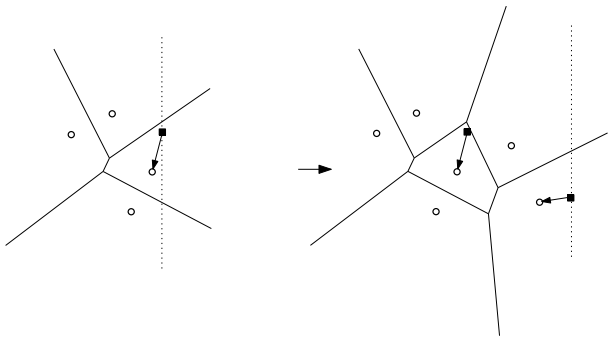

Fig. 3. Sweeping and point location query.

Algorithm MOOP: For each $2 \leq i \leq k$ in $STAGE(i)$, we have a list of points of color $(i-1)$ with its corresponding pointers and labels, and we do the following:

If all points of color i are located to the left of all points of color $i-1$, the algorithm terminates with "no solution." Otherwise, we make a line sweep from left to right with a vertical line stopping at all the

points of color i. At each one, we calculate the closest point of color $i-1$ to its left. Fig. 3 shows the line sweep for a unweighted Voronoi diagram. This process is performed by using the dynamic sweep line technique of [6] to dynamically calculate the additively weighted Voronoi diagram and the data structure presented in [2] to solve point location queries. The method of [2] only requires the underlying graph associated to a planar subdivision to be connected. Let us observe that although the data structure in [8] is more efficient, it works when each face in the planar subdivision is a monotone polygon. Unfortunately, the weighted Voronoi diagram is not in general a monotone subdivision of the plane.

During the execution of $STAGE(i)$, each time the sweep line stops at a point p_r^i, using a data structure as in [2], we obtain the Voronoi region (of the Voronoi diagram of the set of points of color $i-1$ to the left of p_r^i) containing the query point p_r^i. Next, we assign to p_r^i the weight given by the label $l_r^i = d(p_r^i, p_k^{i-1}) + l_k^{i-1}$ and the prefix $prev(p_r^i) = p_k^{i-1}$, where p_k^{i-1} is the site of the above Voronoi region. Observe that none of the points of color i that are to the left of all points of color $i-1$ are considered in the process.

At the end of $STAGE(k)$, we obtain at most n_k labels and we select the one with minimum weight. The optimal polygonal chain can be recalculated by using the pointers and the labels from right to left.

Complexity Analysis: Every stage of the algorithm **MOOP** is based on two fundamental tools. The first is the line sweep technique of [6] to compute additively weighted Voronoi diagrams. It can be done in $O(n_{i-1} \log n_{i-1})$ time and $O(n_{i-1})$ space, $i = 2, \ldots, k$. The second is the point-location data structure for a dynamic planar subdivision of [2]. For each point of color i, p^i, the closest point of color $i-1$ to the left of p^i, is obtained in $O(\log^2(n_{i-1}))$ time, using $O(n_{i-1})$ space and $O(\log n_{i-1})$ update time.

The overall time is $O(n_1 \log n_1 + n_2 \log^2 n_1) + \ldots + O(n_{k-1} \log n_{k-1} + n_k \log^2 n_{k-1}) \leq O(n \log^2 n)$. Thus we have:

Theorem 1. *The **MOOP** problem can be solved in $O(n \log^2 n)$ time and $O(n)$ space.*

3 Monotone Ordered Polygonal Chain

In this section, we generalize the x-monotone version of the **MOOP** to a more general problem in which the monotonicity orientation is not fixed. Thus we are looking for a direction θ that allows us to construct the shortest heterochromatic monotone polygonal chain of a given point set with respect to θ over all possible values of θ. First, we show a reduction of the **MOP** to a quadratic number of instances of the oriented **MOOP** .

Let $\theta \in [0, \pi)$ be a fixed angle and $\ell(\theta)$ the line through the origin in the direction given by θ. We say that a polygonal chain C with vertices $p^1, p^2, \ldots, p^k$ is *monotone with respect to* θ if the projections $p'_1, p'_2, \ldots, p'_k$ of the points $p_1, p_2, \ldots, p_k$ on $\ell(\theta)$ fall precisely in the order $p'_1, p'_2, \ldots, p'_k$. If we rotate the line $l(\theta)$ through the origin (with θ between 0 and π), the order of the projected points on $l(\theta)$ changes a quadratic number of times. We call these changes *events*. In fact, we have an event when the line passing through any two points p^i and p^j of C is orthogonal to $l(\theta)$, and we denote this event by (p^i, p^j).

We say that two angles θ and θ' are *equivalent* if the order of the corresponding projections does not change. This defines a partition of $[0, \pi)$ into a set of intervals in which any two angles in the same interval are equivalent. By processing the algorithm **MOOP** for an orientation in each of the above intervals, we obtain an algorithm that solves the problem **MOP** in $O(n^3 \log^2 n)$ time and $O(n^2)$ space.

Let $C^\theta = \{p_{i_1}^1, \ldots, p_{i_k}^k\}$ be a shortest heterochromatic monotone (with respect to θ) ordered polygonal chain that visits the chromatic classes in the order $1, \ldots, k$. We now propose a data structure for solving the **MOOP** problem with respect to θ efficiently by using the information accumulated from any event to the next.

Lemma 3. *Let $C^\theta = \{p^1_{i_1}, \dots p^j_{i_j}, \dots, p^k_{i_k}\}$. Then the shortest heterochromatic θ-monotone ordered polygonal chain that visits colors $1, \dots, j$ in that order and ends at $p^j_{i_j}$ has vertices $p^1_{i_1}, \dots, p^j_{i_j}$. Moreover the shortest heterochromatic θ-monotone ordered polygonal chain that visits colors $i, i+1, \dots, k$ in this order and starts at $p^j_{i_j}$ has vertices $p^j_{i_j}, \dots, p^{k-1}_{i_{k-1}}, p^k_{i_k}$.*

Proof. The first part follows from Lemma 2. The proof of the second part is by contradiction. Suppose that there exists another θ-monotone polygonal chain C' with vertices $\{p^j_{i_j}, q^{i+1}, \dots, q^k\}$ that visits colors $i, i+1, \dots, k$ in this order. If the length of C is less than the length of the chain with vertices $\{p^j_{i_j}, \dots, p^{k-1}_{i_{k-1}}, p^k_{i_k}\}$, then the path with vertices $\{p^1_{i_1}, \dots p^j_{i_j}, q^{i+1}, \dots, q^k\}$ is shorter than C', which is a contradiction.

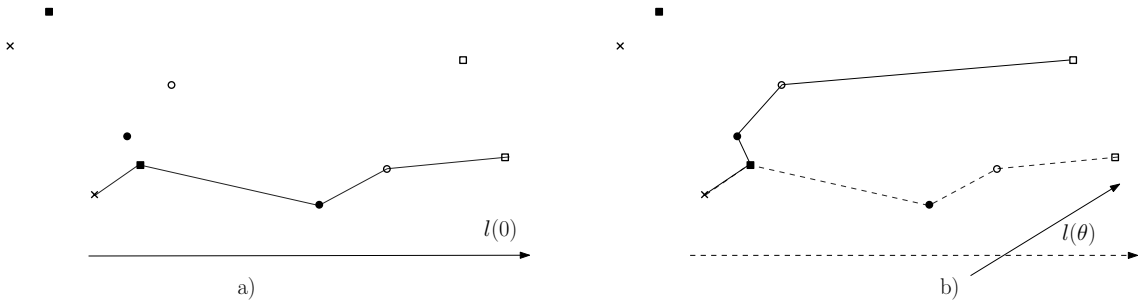

Fig. 4. a) The solution for $\theta = 0$ and order (*cross, square, disk, circle, box*). b) The solution after the first event.

Lemma 3 implies that if $p^j_{i_j}$ lies in the optimal polygonal chain, then the route can be calculated recursively by using a linear number of pointers pointing to both the left and the right. Our algorithm does not start at the beginning for each angle-event. Instead, we perform a rotational sweep with $\ell(\theta)$, maintaining the following data structure:

Data structure: The sweep status is maintained in an array of size $O(n^2)$. Initialize $\theta = 0$. Apply the algorithm **MOOP** twice, from left to right and from right to left, and attach the following information to each point p^j_r of color j:

- two labels, $l^-(p^j_r)$ and $l^+(p^j_r)$; the length of the optimal route with colors $1, 2, \dots, j$ ending at p^j_r and the length of the optimal route with colors $j, j+1, \dots, k$ starting at p^j_r, respectively;
- two ordered lists, $list(p^j_r, left)$ (resp. $list(p^j_r, right)$) containing the points of color $j-1$ (resp. $j+1$) located to the left (resp. right) of p^j_r. The lists are stored in increasing order w.r.t. θ and to the labels $l^-(p^{j-1}_s)$ (resp. $l^+(p^{j+1}_s)$);
- two pointers, $prev(p^j_r)$ and $next(p^j_r)$. Pointer $prev(p^j_r)$ points to the element in $list(p^j_r, left)$ closest to p^j_r, that is the point of color $j-1$ to the left of p^j_r that minimizes $d(p^j_r, p^{j-1}_s) + l^-(p^{j-1}_s)$ over all points p^{j-1}_s of color $j-1$ to the left of p^j_r. Pointer $next(p^j_r)$ points to the closest element in the list $list(p^j_r, right)$, that is the point of color $i+1$ to the left of p^j_r that minimizes $d(p^j_r, p^{j+1}_s) + l^+(p^{j+1}_s)$ over all points p^{j+1}_s of color $j+1$ to the right of p^j_r.

The array is initialized for $\theta = 0$ using the algorithm for the **MOOP** problem presented in the previous section. Next, a linear scan in the array can be used to set all the pointers attached to the elements in the array. Each point has attached to it two lists of linear complexity; thus the overall complexity of the data structure is $O(n^2)$.

Processing: While increasing θ, we maintain the above data structure and the pair (θ, l_θ), where l_θ is the length of the optimal route for the angle θ. We start with $(0, l_0)$. Each time a new event is reached,

say (p_r^j, p_s^t), we insert or delete a value into the corresponding ordered lists of the points involved, say p_r^j and p_s^t, and update the corresponding pointers. This can be done in $O(\log n)$ time. After this event, the relative positions of other points do not change but their lists may be modified; see Fig. 4. Then, to update the overall data structure, we need to update the lists $list(p_s^m, left)$ (and/or $list(p_s^m, right)$) and the pointers for the other points of color m, $m \neq j, t$ that can be affected. As a consequence, the overall data structure can be maintained in $O(n \log n)$ time per event. At the end of the sweep, we obtain the optimal orientation θ^* and the corresponding length l_{θ^*}. At this point we have the data structure updated for the last event. The optimal route can be recalculated by using the algorithm **MOOP** for $\theta = \theta^*$. We have obtained the result, which can be stated:

Theorem 2. *The **MOP** problem can be solved in $O(n^3 \log n)$ time and $O(n^2)$ space.*

Remarks: The non-oriented version of the problem can easily be solved in $O(n \log n)$ time for two and three colors. In fact for two colors, a polygonal chain with two and three vertices is always monotone in some direction, and thus for two colors, say 1 and 2, the problem reduces to finding, for each point of color 1, its closest neighbor of color 2. For three colors, say 1, 2 and 3, the problem reduces to that of finding, for each point of color 2, its closest neighbors of colors 1 and 3. Other advantages of using the data structures proposed in this section lie in their implementation; updating the array of the data structure is easy, while the implementation of the algorithm **MOOP** requires a more complicated data structure management [6, 2].

4 Conclusion and future work

This paper formulates and solves a monotone variation of the *Trip Planning Problem* with restrictions on the order in which a route visits the color classes of our point sets. Our methods exploit the spatial information of the points and can be adapted to other non-Euclidean metrics.

Other variants that arise by removing one of the restrictions can be solved with our approach, as well. The first is the problem of finding the shortest heterochromatic monotone polygonal chain (not ordered). In this case, a simple algorithm is to consider all permutations of k colors and then to apply our method for each order. Obviously, it is efficient only for small values of k. Another problem is to find the heterochromatic ordered route of minimum length. If the monotonicity restriction is removed, the problem can be solved by using an incremental point location query strategy in $O(n \log n)$ time. In fact, in this case we do not use the dynamic point location structure because it is not necessary to consider a line sweep. We summarize the results of these problems in Table 1.

Monotone	Fixed Order	Complexity of **MOOP**, **MOP**
✓	✓	$O(n \log^2 n), O(n^3 \log n)$
–	✓	$O(n\ log\ n)$
✓	–	$k!(O(n \log^2 n))$, $k!(O(n^3 \log n))$
–	–	NP-hard

Table 1. Summary of results.

Another interesting problem arises when an x-monotone path does not exist. In this case, we would like to find a non-monotone path that minimizes both the length and the number of times it backtracks with respect to the x-axis.

Acknowledgements

The problems considered in this paper were partially solved at the First Workshop of Combinatorial and Computational Geometry that took place at the Mathematics Research Center (CIMAT) in Guanajuato, Mexico, December 2006. The authors would like to thank the organizers and the rest of participants for their comments and support.

References

1. E.M. Arkin, R. Connelly, J.S. Mitchell. On monotone paths among obstacles with applications to planning assemblies, Proceedings of the Fifth Annual Symposium on Computational Geometry, Saarbruchen, West Germany, (1989) 334–343.
2. S.W. Cheng, R. Janardan. New results on dynamic planar point location. *SIAM J. Comput.* **21** (1992), 972–999.
3. M. de Berg, J. Gudmundsson, M.J. Katz, C. Levcopoulos, M.H. Overmars, A.F van der Stappen. TSP with neighborhoods of varying size. *Journal of Algorithms* **57** (2005) 22–36.
4. J.M Díaz-Báñez, F Gómez, F Hurtado. Approximation of point sets by 1-corner polygonal chains, *INFORMS Journal on Computing* **12** (2000) 317–323.
5. K Elbassioni, A.V Fishkin, N.H. Mustafa, R. Sitters. Approximation algorithms for Euclidean group TSP. *In Proc. 32nd Internat. Colloq. Automata Lang. Prog.* (2005) 1115–1126.
6. S.J. Fortune. A sweepline algorithm for Voronoi diagrams, *Algorithmica* **2** (1987) 153–174.
7. M.R. Garey, D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.
8. M.T. Goodrich, R. Tamassia. Dynamic trees and dynamic point location, *SIAM J. Comput.* **28**(2) (1998) 612–636.
9. F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, S.H. Teng. On trip planning queries in spatial databases. In *Proc. of the 9th International Symposium on Spatial and Temporal Databases: SSTD05* (2005) 273–290.
10. J.S.B. Mitchell. Shortest paths and networks, In Goodman, J.E., O'Rourke, J. eds. *Handbook of Discrete and Computational Geometry*, CRC Press LLC, (1997) 445–466.
11. J.S.B. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane, *Proc. ACM–SIAM Symp. Discrete Algorithms (SODA 2007)*, 11–18.
12. A. Okabe, B. Boots, K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams* New York: Wiley, 1992.
13. C.H. Papadimitriou. The euclidean traveling salesman problem is NP-complete, *Theor. Comput. Sci.* **4** (1977) 237–244.
14. P. Slavík. The errand scheduling problem. Technical report, March 14 1997. Technical Report, SUNY, Buffalo, USA.
15. Y. Yan, D. Lemire, M. Brooks. Monotone pieces analysis for qualitative modeling, *Proceedings ECAI 2004 MONET Workshop on Model-Based System*, Valencia, Spain, 2004.