International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:1, No:11, 2007

Geometric Data Structures and Their Selected Applications

Miloš Šeda

***Abstract*—**Finding the shortest path between two positions is a fundamental problem in transportation, routing, and communications applications. In robot motion planning, the robot should pass around the obstacles touching none of them, i.e. the goal is to find a collision-free path from a starting to a target position. This task has many specific formulations depending on the shape of obstacles, allowable directions of movements, knowledge of the scene, etc. Research of path planning has yielded many fundamentally different approaches to its solution, mainly based on various decomposition and roadmap methods. In this paper, we show a possible use of visibility graphs in point-to-point motion planning in the Euclidean plane and an alternative approach using Voronoi diagrams that decreases the probability of collisions with obstacles. The second application area, investigated here, is focused on problems of finding minimal networks connecting a set of given points in the plane using either only straight connections between pairs of points (minimum spanning tree) or allowing the addition of auxiliary points to the set to obtain shorter spanning networks (minimum Steiner tree).

***Keywords*—**motion planning, spanning tree, Steiner tree, Delaunay triangulation, Voronoi diagram.

I. Introduction

IN recent years a number of new data structures and algorithmic techniques have been developed that have improved and simplified many of the previous approaches used in network optimisation [1], robot motion planning [2], etc. Geometric data structures defined in computational geometry have a surprising variety of uses [3]-[7].

Computational geometry emerged from the field of algorithm design and analysis in the late 1970s. It has many application domains including computer graphics, geographic information systems (GIS), robotics, and others in which geometric algorithms play a fundamental role. Computational geometry deals with specific geometric data structures, the most important ones being Voronoi diagrams, Delaunay triangulation, visibility graph and convex hull.

Before we study examples of their applications, we will introduce them and summarise the basic definitions.

Manuscript received February 15, 2006. This work was supported in part by the Ministry of Education, Youth and Sports of the Czech Republic under research plan MSM 0021630518 "Simulation Modelling of Mechatronic Systems".

Miloš Šeda works in the Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2896/2, CZ 616 69 Brno, Czech Republic (phone: +420-54114 3332; fax: +420-54114 2330; e-mail: seda@fme.vutbr.cz).

II. Basic Notions

A Voronoi diagram of a set of points (called *sites*) in the Euclidean plane is a collection of regions that divide up the plane. Each region corresponds to one of the sites and all the points in one region are closer to the site representing the region than to any other site.

More formally [3]-[6]:

Definition 1 Let P be a set of n points in the plane. For two distinct sites $p_i, p_j \in P$, the *dominance* of p_i over p_j is defined as the subset of the plane that is at least as close to p_i as to p_j. Formally,

$$\mathrm{dom}(p_i, p_j)=\{x\in\Re^2 \mid d(x, p_i)\le d(x, p_j)\}, \quad (1)$$

where d denotes the Euclidean distance. □

Clearly, $\mathrm{dom}(p_i, p_j)$ is a closed half-plane bounded by the perpendicular bisector of p_i and p_j.

Definition 2 *Voronoi region* (or *Voronoi polytope*, *Voronoi cell*, *Voronoi face*, *Dirichlet polygon*, *Thiessen polygon*) of a site $p_i\in P$ is a close or open area $V(p_i)$ of points in the plane such that $p_i\in V(p_i)$ for each p_i, and any point $x\in V(p_i)$ is at least as close to p_i as to any other sites in P (i.e. $V(p_i)$ is the area lying in all of the dominances of p_i over the remaining sites in P).

Formally,

$$V(P_i)=\left\{x\in\Re^2 \mid d(x,p_i)\le d(x,q):\forall q\in(P-\{p_i\})\right\}= \\ =\bigcap_{q\in P-\{p_i\}}\mathrm{dom}(p_i,q) \quad (2)$$

□

Since the Voronoi regions are formed by intersecting $n-1$ half planes, they are convex polygons. Thus the boundary of a region consists of at most $n-1$ *edges* (maximal open straight-line segments) and *vertices* (their endpoints). Points on the boundary of $V(p_i)$ and $V(p_j)$ are equidistant to p_i and p_j.

Definition 3 A *Voronoi diagram* (or *Voronoi tessellation*) for a given set $P=\{p_1, p_2, \dots, p_n\}$ of points (or sites) is a polygonal partition of the plane into Voronoi regions $V(p_1),V(p_2), \dots, V(p_n)$. The vertices of polygons $V(p_i)$ are called the *vertices of the Voronoi diagram*, and their edges are called the *edges of the Voronoi diagram*. A Voronoi diagram is called *degenerate* if four or more of its Voronoi edges have a common endpoint. □

Clearly, each edge of the Voronoi diagram belongs to just two Voronoi regions and

$$V(P) = \bigcup_{p_i \in P} V(P_i) \quad (3)$$

Definition 4 A *triangulation* T is a collection of N triangles satisfying the following requirements:

(i) The interiors of the triangles are pairwise disjoint.
(ii) Each edge of a triangle in T is either the common edge of two triangles in T or else it is on the boundary of the union D of all the triangles.
(iii) D is homeomorphic to a square (this requirement rules out holes, pinchpoints where just two triangles meet in a single point, and disjoint sets of triangles).

□

Definition 5 The graph $D(P)$ on P with an edge (p_i , p_j) if $V(p_i)$ and $V(p_j)$ share a common side is called a *Delaunay triangulation*. □

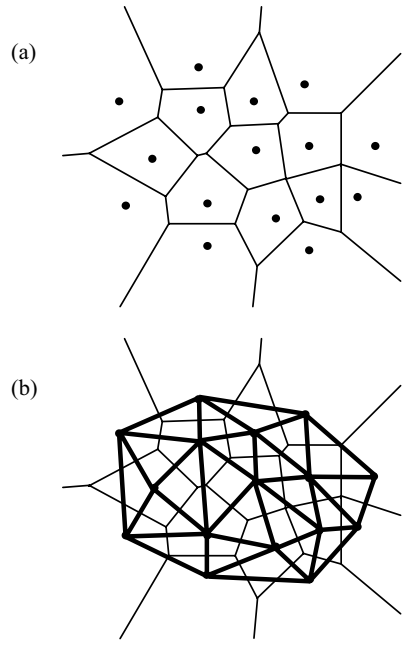

Fig. 1 (a) A set of points and its Voronoi diagram; (b) Delaunay triangulation (bold)

Definition 6 Let $P=\{p_1, p_2, \dots , p_n\}$ be a set of n distinct points and $O=\{O_1, O_2, \dots , O_m\}$ be a set of m closed regions that represent a set of obstacles that are neither transparent nor traversable and do not overlap. Two vertices that can see each other are called (*mutually*) *visible*, and the segment connecting them is called a *visibility edge*. □

Note that endpoints of the same obstacle edge always see each other.

Definition 7 Let $O=\{O_1, O_2, \dots , O_m\}$be a set of m obstacles, S be a set of their vertices and p_{start} and p_{target} be the starting and target positions. A *visibility graph* is a graph $G=(V,E)$ whose set of vertices V is given by $S \cup p_{start}$ and p_{target} and the set of edges E is given by the visibility edges on V. □

First summarise the time complexity of algorithms for constructing Voronoi diagrams and visibility graphs as described in the literature.

The most efficient algorithms for constructing Voronoi diagrams, *randomised incremental algorithm*, *divide and conquer* and *plane sweep algorithm*, have the same time complexity $O(n \log n)$where n is the number of given points.

The visibility graph of a set of disjoint polygonal obstacles with k edges in total can be computed in $O(k^2 \log k)$ [4].

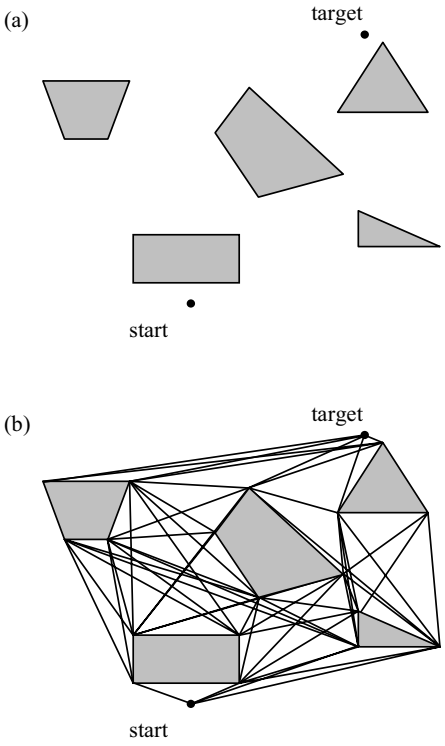

Fig. 2 (a) A set of polygonal obstacles and starting and target position; (b) visibility graph for (a)

III. Shortest Path in the Plane with Obstacles

The shortest path between two points in the plane with polygonal obstacles can be easily solved in the corresponding visibility graph by the Dijkstra algorithm. Using a binary heap implementation, its time complexity is given by $O(|E| \log |V|)$, where E is the set of edges and V is the set of vertices. Fig. 3 shows the shortest path between the starting and target positions using the visibility graph in Fig. 2.

IV. Robot Motion Planning

The algorithm, described in the previous section, can also be used for solving the motion planning problem for a point

robot. This approach can be adapted for the case of robot with a polygonal projection into the plane. We "add" the size of robot to the obstacles using Minkowski sums [4] and thus the robot is reduced to a point.

Now we will describe another approach that does not guarantee finding the shortest path but takes into consideration motion safety in the sense of minimising possible collisions with obstacles.

Consider a disc-shaped robot as in Fig. 4 (a). If a set of obstacles is only by points, these point obstacles can be considered as sites of a Voronoi diagram and the robot can use for its tour the shortest path along the Voronoi diagram edges that represent passable channels among obstacles. This allows us to reduce the robot motion problem to a graph search problem again. Fig. 4 (a) demonstrates this approach. However, in practice, obstacles very often have more general shapes than point ones and thus we must generalise the algorithm for constructing Voronoi diagrams. We take the vertices of polygonal obstacles to be point obstacles, then compute the corresponding Voronoi diagram and, finally, remove from the diagram all the edges intersecting the obstacles. An example of Voronoi diagram for a plane with polygonal obstacles is shown in Fig. 4 (b).

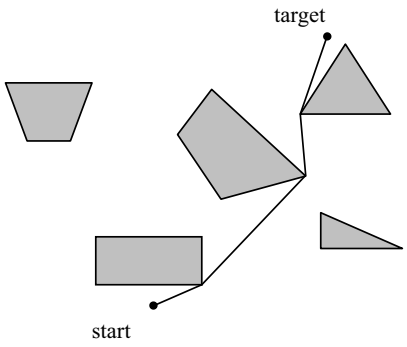

Fig. 3 The shortest path for the visibility graph in Fig. 2

If we use the rectilinear metric for a Voronoi diagram, then, due to the rectilinearity, each straight-line segment of a bisector in the now rectilinear Voronoi diagram will be either horizontal, vertical, or inclined at 45° or 135° to the positive direction of the x-axis [8], [9]. This finding suggests using the rectilinear Voronoi diagram for the 8-directional motion planning. This approach avoids all the drawbacks of classical plane decomposition methods (combinatorial explosion, low boundaries for grid representation and generating many infeasible solutions).

V. Euclidean Minimum Spanning Tree

Definition 8 Let $G = (V,E)$ be a connected undirected graph in which each edge e is assigned a real non-negative weight (or cost) $w(e)$.

(i) A spanning tree T is an acyclic connected subgraph of G containing every vertex of V.

(ii) A minimum spanning tree (MSpT) is a spanning tree of minimum weight

$$w(T) = \sum_{e \in E', E' \subseteq E} w(e) \qquad (4)$$

□

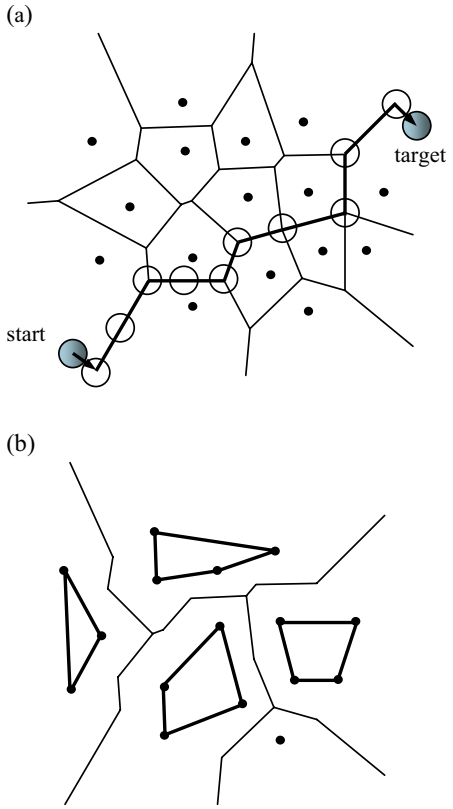

Fig. 4 (a) Motion planning in a scene with point obstacles; (b)Voronoi diagram for polygonal obstacles

A frequent practical problem is one of constructing a minimum spanning tree in the Euclidean plane (EMSpT). This problem can be easily solved by well-known Jarník's (Prim's), Kruskal's or Borůvka's polynomial algorithms for the Minimum Spanning Tree Problem in graphs (MSpTG) when we construct from the given set of points a complete graph whose edges are represented by straight lines between each pair of points and their weights correspond to Euclidean distances of these points. All these algorithms have the same asymptotic running time $O((|V|+|E|) \log |V|)$ for a graph $G=(V,E)$ [10], but, unfortunately, the time complexity of the complete graph construction is higher, it equals $O(|V|^2)$ and therefore the total running time of the algorithm is $O(|V|^2)$.

Another approach to solving the EMSpT problem is based on the concept of the Delaunay triangulation. We refer to the fact that when searching for a current edge of minimal weight

in minimum spanning tree algorithms, it suffices to look over the edges of the Delaunay triangulation. It results from the following assertions [6], [11].

Lemma 1 *Let $P = P' \cup P''$ be an arbitrary partition of a finite set P of points of the plane, and assume that e is the shortest segment joining the sets P' and P''. Then e is an edge of the Delaunay triangulation for the set P.*

Theorem 2 *Let P be an arbitrary finite set of points of a plane, and assume that T is some minimum tree spanning P. Then the edges of T are edges of the Delaunay triangulation D(P) for the set P.*

Proof. Each step of Jarník's algorithm for constructing an MST spanning P consists of constructing a tree T spanning $T(P) \subset P$, and adding an edge e that joins a vertex in $T(P)$ with one in $P-T(P)$ such that the length of e is the least possible among the lengths of the edges of this type. •

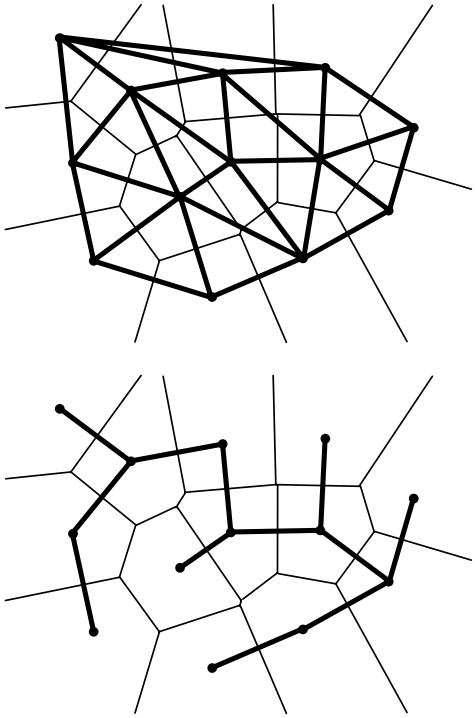

Fig. 5 Voronoi diagram, Delaunay triangulation and Euclidean mimimum spanning tree

Using these results, we can construct the minimum spanning tree in the Euclidean plane as follows: First, we construct a Voronoi diagram from the given set of n points (in $O(n \log n)$ time), then the corresponding Delaunay triangulation (in $O(n)$ time) and finally the EMST (in $O(n)$ time [12]). Therefore, the Delaunay based approach is more efficient than the one based on constructing the complete graph consuming $O(n^2)$ time.

Let us note that there are algorithms for constructing the Delaunay triangulation directly without precomputing the Voronoi diagram [5].

By traversing the Euclidean minimum spanning tree twice (as in Fig. 6) we produce a Euclidean travelling salesman tour trough a given set of points. In [6], it is proved that its length is less than twice the length of the exact solution.

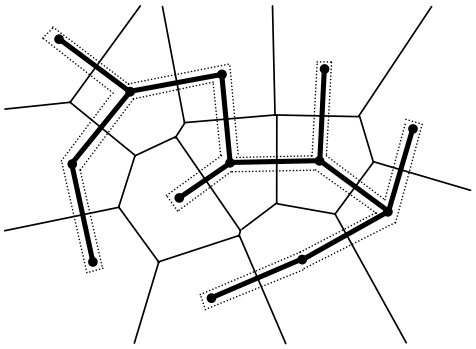

Fig. 6 An approximation of the Euclidean travelling salesman problem.

VI. Euclidean Minimum Steiner Tree

The *Euclidean Steiner tree problem* is given by a set of fixed points $V=\{v_1,v_2, \ldots ,v_n\}$ in the Euclidean plane, called terminals, and asks for the shortest straight-line spanning V. The solution takes the form of a tree, called a *Euclidean Steiner minimum tree* (EStMT). Contrary to the *minimum spanning tree problem*, connections in EStMT's are not required to be between the terminals only. Additional intersections, called *Steiner points*, can be introduced to obtain shorter spanning networks. However, the Steiner tree problem is NP-hard and therefore it cannot be solved exactly for larger instances by polynomial algorithms as minimum spanning tree problem. Therefore, heuristic or approximation methods must be used instead.

Definition 9 The *Euclidean Steiner ratio*, denoted ρ_E, is the supremum over the set of all the ratios of the length of the Euclidean minimum spanning tree $w(\mathrm{EMSpT}(V))$ to the length of the Euclidean Steiner minimum tree $w(\mathrm{EStMT}(V))$,

$$\rho_E = \sup_{V \subset \Re^2} \frac{w(\mathrm{EMSpT}(V))}{w(\mathrm{EStMT}(V))} \tag{5}$$

□

Theorem 3 The Steiner ratio ρ_E for the Euclidean problem satisfies the following formula [13]:

$$\rho_E = \frac{2}{\sqrt{3}} \approx 1.1547 \tag{6}$$

•

This means that the EMSpT length does not exceed that of an EStMT by more than 15.47% (the average excessive length is of course smaller). Therefore, the Euclidean minimum

spanning tree can be used as an approximation of minimum Steiner tree and, naturally, becomes the standard to which other approximation algorithms or heuristics are compared.

If $|V|=3$ we can directly construct the EStMT as follows: Let $V=\{a,b,c\}$.

1. If one of the angles of Δabc is at least 120°, then the EStMT consists of simply the two edges subtending the obtuse angle.
2. If all internal angles of Δabc are less than 120°, then we draw an equilateral triangle Δabc and circumscribe a circle around this triangle. The Steiner point s is given by the intersection of line cd with the circle (see Fig. 7 (a)). It can be shown that the total length of segments as, bs, cs is equal to the length of segment cd, which is known as the *Simpson line* for the FST over terminals a, b, c.

From Fig. 7 (b) it can be easily derived that the Euclidean Steiner minimum trees for points lying at the vertices of equilateral triangles (Fig. 7 (b)) are $2/\sqrt{3}$-times shorter than their corresponding Euclidean minimum spanning trees and represent the best improvements with respect to the minimum spanning trees. All angles in an equilateral triangle are equal to 60° and, obviously, the minimal angle in such a triangle is maximised. Since, by [4], any Delaunay triangulation of V maximises the minimal angle over all the triangulations of V, we begin with the Delaunay triangulation $DT(V)$.

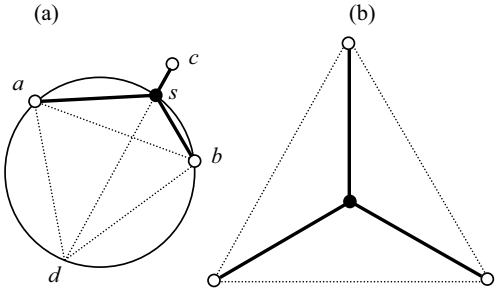

Fig. 7 (a) A 3-point EStMT algorithm; (b) the EStMT for 3 points at vertices of an equilateral triangle.

As mentioned above, we can use $DT(V)$ for the EMSpT enumeration. But we can achieve a better initial approximation if we first replace all the triangles that have each angle less than 120° by the EStMT's of the triangle vertices and, after this, find EMSpT. The length of the graph resulting from this procedure can be still decreased by repeated selections of the smallest angle based on the fact that the smaller the angle is, the better local improvement is reached. Of course this angle must be less than 120°. The replacements of such subgraphs by their EStMT's are the same as before.

The angle between a pair of edges meeting at an end point can be easily determined from elementary geometry. If α is an angle to be calculated and the Euclidean distances of the end points defining the lengths of the triangle edges are denoted by a, b, c, as in Fig. 7, we have:

$$a^2 = b^2 + c^2 - 2bc \cos\alpha \Rightarrow$$
$$\Rightarrow \alpha = \arccos\frac{b^2 + c^2 - a^2}{2bc} \quad (7)$$

After previous considerations, the algorithm can be described as follows:

1. Find the Delaunay triangulation for a given set V.
2. Replace all triangles that have each angle less than 120° by the EStMT's of triangle vertices.
3. Determine the Euclidean minimum spanning tree EMSpT for a graph found in Step 2.
4. **For each** edge connecting points x, y **do**
   a. Find the edge $\{y,z\}$ that meets $\{x,y\}$ at the smallest angle.
   b. **If** this angle is less than 120° **then**
      i. Place a new Steiner point s_n on top of y.
      ii. Remove the edges $\{x,y\}$ and $\{y,z\}$. These edges will no longer be considered for the loop of Step 4.
      iii. Add the edges $\{x,s_n\}$, $\{y,s_n\}$ and $\{z,s_n\}$.
5. Remove all Steiner points of degree 1 along with their incident edges. □

Step 4 is a slightly modified step from an insertion heuristic described in [14]. It systematically inserts Steiner points between the edges of the current graph that meet at angles less than 120 degrees. However, because of the first three steps in our algorithm, we ignore whether the vertices connecting edges being terminals or not. The final optimisation in Step 5 is inspired by a pre-processing rule known from the Steiner tree problem in graphs.

Comparing it to [14], we see that it contains an additional step computing Delaunay triangulation and using the fact that EStMTs for triangles can be determined very easily. This improves the behaviour of the heuristic and the results achieved are near-optimal [15].

Theorem 4 *The proposed algorithm runs in $O(n^3)$ time.*

Proof. If $|V|=n$, then the Delaunay triangulation can be constructed in $O(n \log n)$ time [4]. The expected number of triangles created is at most $9n+1$ [4] and thus Step 2 needs $O(n)$ time. Step 3 needs $O(n \log n)$ time [3], [4], [16], time complexity of Step 4 is $O(n^3)$ and, finally, Step 5 is done in $O(n)$ time. Since Step 4 is the most expensive, it gives the time complexity of the algorithm. •

VII. Conclusion

In this paper, two approaches for planning trajectories were discussed. Our considerations were restricted to known two-dimensional scenes with point and polygonal obstacles. Since traditional potential field methods and cell decomposition methods have many drawbacks such as convergence to local minima, and combinatorial explosion, or generating infeasible solutions, we focused on roadmap methods using computational geometry data structures. It can be said that, for scenes with point, straight line and polygonal obstacles, the

simplest way of finding optimal trajectories is to compute ordinary Voronoi diagrams for vertices of obstacles and then remove the Voronoi diagram edges that intersect the obstacles. While the application of visibility graphs guarantees finding the shortest paths between the starting and target positions, Voronoi diagrams guarantee increased safety of robot movements because the generated collision-free trajectory goes midway between pairs of obstacles.

The algorithm for finding Euclidean minimum spanning tree introduced in section V decreases time complexity in comparison with the classical complete graph-based approach. Finally, it can be shown that the algorithm from section VI achieves near-optimal solutions in a reasonable amount of time, more precisely, for benchmarks of up to 100 points, less than in tens of seconds and, in the case of 500 points, in 10 minutes.

REFERENCES

[1] D.-Z. Du, J.M. Smith, and J.H. Rubinstein, *Advances in Steiner Trees*. Dordrecht: Kluwer Academic Publishers, 2000.
[2] S.M. LaValle, *Planning Algorithms*. Cambridge: University Press, 2006.
[3] F. Aurenhammer, "Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345-405, 1991.
[4] M. de Berg, M., M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. Berlin: Springer-Verlag, 2000.
[5] D.A. Du and F.K. Hwang (eds.), *Euclidean Geometry and Computers*. Singapore: World Scientific Publishing ,1992.
[6] A.O. Ivanov and A.A. Tuzhilin, *Minimal Networks. The Steiner Tree Problem and its Generalizations*. Boca Raton: CRC Press, 1994.
[7] A. Okabe, B. Boots, K. Sugihara, and S.N. Chiu, *Spatial Tessellations and Applications of Voronoi Diagrams*. New York: John Wiley & Sons, 2000.
[8] S. Guha and I. Suzuki, "Proximity Problems for Points on a Rectilinear Plane with Rectangular Obstacles," *Algorithmica*, vol. 17, pp. 281-307, 1997.
[9] M. Šeda, "Rectilinear Voronoi Diagram-Based Motion Planning in the Plane with Obstacles," *Elektronika* (Poland), no. 8-9, pp. 24-26, 2004.
[10] D.M. Mount, „Design and Analysis of Computer Algorithms," Lecture Notes, University of Maryland, College Park, 1999, 131 pp.
[11] F.K. Hwang, D.S. Richards and P. Winter, *The Steiner Tree Problem*. Amsterdam: North-Holland, 1992.
[12] D. Cheriton and R.E. Tarjan, "Finding Minimum Spanning Trees," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 724-742, 1976.
[13] D.-Z. Du and F.K. Hwang, "A Proof of the Gilbert-Pollak Conjecture on the Steiner Ratio," *Algorithmica*, vol. 7, pp. 121-135, 1992.
[14] D.R. Dreyer and M.L. Overton, "Two Heuristics for Euclidean Steiner Tree Problem," *Journal on Global Optimization*, vol. 13, pp. 95-106, 1998.
[15] M. Šeda, "Solving the Euclidean Steiner Tree Problem Using Delaunay Triangulation," *WSEAS Transactions on Computers*, vol. 4, no. 6, pp. 471-476, 2005.
[16] M.I. Shamos and D. Hoey, "Closest Point Problems," in *Proc. 16th Annual Symposium on Foundations of Computer Science FOCS '75*, Berkeley, 1975, pp. 151-162.