Rectilinear Shortest Path and Rectilinear Minimum Spanning Tree with Neighborhoods

Yann Disser[1], Matúš Mihalák[2], Sandro Montanari[2](✉), and Peter Widmayer[2]

[1] Department of Mathematics, TU Berlin, Berlin, Germany
[2] Department of Computer Science, ETH Zurich, Zurich, Switzerland
sandro.montanari@inf.ethz.ch

Abstract. We consider a setting where we are given a graph $\mathcal{G} = (\mathcal{R}, E)$, where $\mathcal{R} = \{R_1, \ldots, R_n\}$ is a set of polygonal regions in the plane. Placing a point p_i inside each region R_i turns G into an edge-weighted graph $G_{\boldsymbol{p}}$, $\boldsymbol{p} = \{p_1, \ldots, p_n\}$, where the cost of $(R_i, R_j) \in E$ is the distance between p_i and p_j. The *Shortest Path Problem with Neighborhoods* asks, for given R_s and R_t, to find a placement $\boldsymbol{p}$ such that the cost of a resulting shortest *st*-path in $\mathcal{G}_{\boldsymbol{p}}$ is minimum among all graphs $\mathcal{G}_{\boldsymbol{p}}$. The *Minimum Spanning Tree Problem with Neighborhoods* asks to find a placement $\boldsymbol{p}$ such that the cost of a resulting minimum spanning tree is minimum among all graphs $\mathcal{G}_{\boldsymbol{p}}$. We study these problems in the L_1 metric, and show that the shortest path problem with neighborhoods is solvable in polynomial time, whereas the minimum spanning tree problem with neighborhoods is APX-hard, even if the neighborhood regions are segments.

Keywords: Neighborhoods · Minimum spanning tree · Shortest path

1 Introduction

In computational geometry we typically assume to be able to estimate locations of objects as exact points in the plane. In many real world applications, however, obtaining this information without uncertainty might be unrealistic because of noise or uncertain measurements; therefore, standard techniques and algorithms cannot be applied.

A more realistic assumption is to consider, instead of exact points, uncertainty (or *neighborhood*) regions in which we are assured the objects will lie. In this setting, shapes and properties of geometric structures induced by the points (i.e., convex hulls, minimum spanning trees, etc.) will vary with their placements inside the regions. It then becomes crucial to inspect the best and worst possible placements for the considered application. Typically, these are placements minimizing or maximizing a certain cost function of some geometric structure induced by the placement points. In this paper, we study minimum spanning trees and shortest paths under the assumption that the neighborhood regions are rectilinear polygons, and distances are measured in the L_1 metric.

P. Fouilhoux et al. (Eds.): ISCO 2014, LNCS 8596, pp. 208–220, 2014.
DOI: 10.1007/978-3-319-09174-7_18

Several geometric optimization problems have been studied in the setting "with neighborhoods", such as the *Traveling Salesman Problem* [4,5] , the problems of finding a *convex hull* [8] or *enclosing circle* [9], or the *Minimum Spanning Tree Problem* [2,8,11]. In these variants, one searches for a placement of points inside the neighborhood regions such that the resulting cost of the geometric structure induced by the points of the placement is as big or as small as possible. For example, the *Minimum Spanning Tree Problem with Neighborhoods* (MSTN for short) asks for a placement of points, one inside each neighborhood region, such that the cost of a Euclidean minimum spanning tree of these points is smallest among all such placements. Löffler and van Kreveld showed [8] this problem to be NP-hard when the neighborhood regions are squares (not necessarily disjoint). Yang et al. [11] showed that when the neighborhood regions are disjoint unit disks, the problem admits a PTAS (i.e., it can be approximated arbitrarily well). Dorrigiv et al. [2] later proved APX-hardness of MSTN in L_2 metric when the neighborhood regions are disjoint disks. They propose to study the setting where the regions may consist of other shapes, such as line segments or rectangles. Our results show that even if the regions consist of vertically or horizontally aligned segments, the problem remains hard to approximate.

Another problem considered in the setting "with Neighborhoods" is the so-called TouringPolygons. Given a sequence of simple polygons in the plane, a start point s, and a target point t, TouringPolygons is the problem to find a shortest tour that starts at s, visits the polygons in the given order, and ends at t. This problem is solvable in polynomial time whenever the polygons are convex and disjoint [3]. If the polygons are allowed to be non-convex and intersecting, the problem is NP-hard for any metric L_p, $p \geq 1$, unless the polygons are rectilinear regions (not necessarily convex) and distances are measured with the L_1 metric [3]. For several years, the complexity for the case of general non-convex yet disjoint polygons has been open, which motivated the design of approximation algorithms [10]. Recently, Ahadi et al. [1] proved TouringPolygons to be NP-hard for non-convex disjoint polygons for every metric L_p, $p \geq 1$, even for degenerate polygons composed of two line segments joint at a common endpoint whose angles with the x-axis are in $\{0, \pm\pi/4, \pi/2\}$.

In this paper we study a generalization of TouringPolygons relaxing the requirement to visit all polygons in a natural way. In addition to the set of polygons, we are given a set of *allowed traversals* represented as a graph defined on the polygonal regions. Given a start and a target polygon, we search for a path of minimum length traversing the polygons accordingly with the edges of the underlying graph. We call this the *Shortest Path Problem with Neighborhoods*, SPN for short. In general the problem remains NP-hard, since TouringPolygons is a special case where the graph is the path induced by the order in which the polygons need to be visited. Our results show that SPN is solvable in polynomial time for the L_1 metric in case the polygons are rectilinear regions not necessarily convex.

2 Shortest Path with Neighborhoods

Let $\mathcal{G} = (\mathcal{R}, E)$ be a directed graph defined over a set of non-overlapping rectilinear polygons $\mathcal{R} = \{R_1, \dots, R_n\}$. Placing a point p_i inside each R_i turns $\mathcal{G}$ into an edge-weighted graph $\mathcal{G}_{\boldsymbol{p}}, \boldsymbol{p} = \{p_1, \dots, p_n\}$, where the cost of $(R_i, R_j) \in E$ is the L_1 distance between p_i and p_j. Given a pair $s, t \in \{1, \dots, n\}$, the Shortest Path Problem with Neighborhoods, or SPN, asks for a placement $\boldsymbol{p}$ such that the cost of a shortest path between R_s and R_t in $\mathcal{G}_{\boldsymbol{p}}$ is smallest among all possible placements. We call such a placement an *optimum SPN placement.*

The SPN problem can be solved trivially if $(R_s, R_t) \in E$. In this case, a shortest st-path is the edge (R_s, R_t), and an optimum placement minimizes the length of this edge. If $(R_s, R_t) \notin E$, it is not clear a priori what sequence of rectangles constitutes a shortest st-path, for an optimum placement $\boldsymbol{p}$. Note that, if we know which regions constitute a shortest st-path, and in which order, the problem becomes the Touring Polygons Problem.

Given a finite set of points $P \subset \mathbb{R}^2$, the *Hanan grid* of P is induced by imposing horizontal and vertical lines through the points in P. In the following, we show that an optimum SPN placement lies on the intersections of the lines of the Hanan grid induced by the corners of the regions in $\mathcal{R}$. Based on this, we provide an algorithm that computes an optimum SPN placement in time $\mathcal{O}(n^2k^2 \log nk + mnk^3)$, where $n = |\mathcal{R}|$, $m = |E|$, and k is the maximum number of corners of a region of $\mathcal{R}$. This result is a generalization of the polynomial-time algorithm for TouringPolygons with rectilinear regions.

2.1 Properties of Optimum Solutions

Since distances between points are measured in L_1 metric and the neighborhood regions are axis-parallel, one of the most trivial approaches is to consider as possible placement points only the corners of regions in $\mathcal{R}$. It is however easy to construct instances where every optimum placement contains at least one point that is not a corner of a region in $\mathcal{R}$. We do not have to consider many more points other than the corners of the regions, though. Lemma 1 (below) shows that there always exists an optimum SPN placement where all points are points of the Hanan grid induced by the corners of the regions in $\mathcal{R}$, lying on the perimeters of those regions. To prove this, we use a property of the L_1 metric defined in terms of bounding boxes of points in $\mathbb{R}^2$. Given $x, y \in \mathbb{R}^2$, the *bounding box* B_{xy} is the smallest axis-parallel rectangle containing x and y.

Proposition 1. *For every $x, y, z \in \mathbb{R}^2$,*

$$\begin{aligned} z \in \mathsf{B}_{xy} &\iff \|xy\| = \|xz\| + \|zy\| \\ z \notin \mathsf{B}_{xy} &\iff \|xy\| < \|xz\| + \|zy\|. \end{aligned}$$

Lemma 1. *There exists an optimum placement $\boldsymbol{p}$ such that every $p_i \in \boldsymbol{p}$ lies on the perimeter of R_i and is a grid point of the Hanan grid induced by the corners of the regions in $\mathcal{R}$.*

Proof. Let $\boldsymbol{p}$ be an optimum placement and P a shortest st-path in $\mathcal{G}_{\boldsymbol{p}}$. We show how to move points in $\boldsymbol{p}$ not satisfying the lemma to points of the Hanan grid on the perimeter of the regions in a way such that the resulting placement is still optimum. We distinguish between regions on (visited by) P and not on P.

A point in $\boldsymbol{p}$ of a region not on P not satisfying the lemma can be trivially moved to an arbitrary corner of that region. Since the cost of P in the resulting placement is the same as in $\mathcal{G}_{\boldsymbol{p}}$, the resulting placement is still optimum.

We first show how to move points of regions on P not satisfying the lemma to the perimeter in a way such that the resulting placement is still optimum. Then, we show that every remaining point still not satisfying the lemma can be moved to a Hanan grid point on the perimeter of its region.

Note that p_s of R_s lies on its perimeter, otherwise we can obtain a better placement by moving it to a point on the perimeter of R_s closest to the point in the successor of R_s on P. The same argument holds by simmetry for p_t.

Let $R_j \notin \{R_s, R_t\}$ be a region on P, and consider $p_i, p_k \in \boldsymbol{p}$, where R_i is the predecessor of R_j on P and R_k is its successor. Consider the bounding box $\mathsf{B}_{p_i p_j}$, and let p_c be a point on the perimeter of R_j contained in B_{ij}. By Proposition 1 and triangle inequality, we have

$$\|p_i p_j\| + \|p_j p_k\| = \|p_i p_c\| + \|p_c p_j\| + \|p_j p_k\| \geq \|p_i p_c\| + \|p_c p_k\|.$$

Thus, moving p_j to p_c does not increase the cost of P. The resulting placement is still optimum, and p_j now lies on the perimeter of R_j. We can apply this operation to every point in the interior of its region.

We now show how to move points in $\boldsymbol{p}$ to Hanan grid points on the perimeters of their regions in such a way that the resulting placement is still optimum. By the above, we can assume each point of $\boldsymbol{p}$ to be lying on the perimeter its region, and that only points of regions on P may not be grid points.

Let $p_j = (x_j, y_j) \in \boldsymbol{p}$ be a point on the perimeter of R_j not on the Hanan grid. Since R_j is axis-parallel, p_j lies on a line of the grid. Thus, either x_j is the x-coordinate of a grid point, or y_j is the y-coordinate of a grid point. We consider only the latter case; the former is symmetric.

Let x_l be the largest x-coordinate of a grid point lying to the left of p_j, and x_r be the smallest x-coordinate of a grid point lying to the right of p_j. We define the set $\{(x, y) \in \mathbb{R}^2 \mid x_l < x < x_r\}$ as the *vertical stripe* of p_j.

Consider a sequence $R_i, \ldots, R_k$ of consecutive regions on P of maximal length such that R_j is in the sequence, and every point in $\boldsymbol{p}$ of a region in the sequence lies in the vertical stripe of p_j. None of the points in the sequence is a grid point; however, the y-coordinate of all such points are y-coordinates of grid points. We first consider the case where $R_i \neq R_s$ and $R_k \neq R_t$.

Let $R_{i'}$ be the predecessor of R_i on P, and $R_{k'}$ be the successor of R_k on P. If $p_{i'}$ lies to the left of the vertical stripe of p_j, we move every point $p_i, \ldots, p_k$ horizontally to the x-coordinate x_l. Otherwise, we move them horizontally to x_r. Figure 1 illustrates an example of such a moving. The cost difference of P before and after moving the points can be expressed as

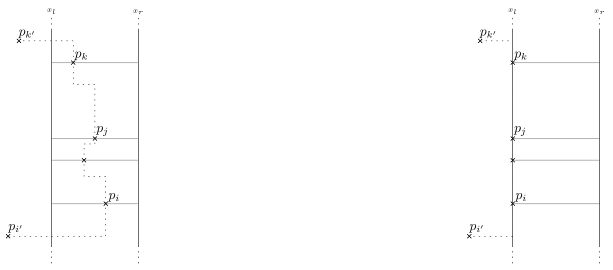

Fig. 1. Moving points in a vertical stripe.

$$\sum_{(R_a,R_b)\in P'} \|p_a,p_b\| - \|p'_a,p'_b\|, \tag{1}$$

where P' is the sub-path between $R_{i'}$ and $R_{k'}$, and p'_a (resp. p'_b) is the new location of p_a (p_b). Since points are only moved horizontally, their y-differences do not change. Thus, (1) can be rewritten as

$$\sum_{(R_a,R_b)\in P'} |x_a - x_b| - |x'_a - x'_b|. \tag{2}$$

Before moving them, all points $p_i, \dots, p_k$ are contained in the vertical stripe of p_j; therefore the cost of P' before the moving is at least $|x_{i'} - x_i| + |x_{k'} - x_k|$. After the moving, the x-coordinates of all $p_i, \dots, p_k$ become $x' \in \{x_l, x_r\}$. Thus, (2) is at least

$$|x_{i'} - x_i| + |x_{k'} - x_k| - |x_{i'} - x'| - |x_{k'} - x'|. \tag{3}$$

If $p_{i'}$ and $p_{k'}$ lie on the same side of the vertical stripe of p_j, the new coordinate x' is closer to both $x_{i'}$ and $x_{k'}$. If $p_{i'}$ and $p_{k'}$ lie on different sides of the vertical stripe, then $|x_{i'} - x'| + |x_{k'} - x'| = |x_{i'} - x_{k'}|$. In both cases, (3) is positive; that is, the cost of P does not increase and the new placement is still optimum.

The case where $R_i = R_s$ follows trivially from above, because we can define $p_{i'}$ to be the point p_i itself. In this way, the distance between p_i and $p_{i'}$ is always 0, and the direction of the moving depends only on the position of $p_{k'}$. The same holds also for the remaining cases. □

2.2 Algorithm

We now present an algorithm that computes an optimum SPN placement by exploiting the structural properties of optimal placements established in Lemma 1. To do so, we create an auxiliary graph from $\mathcal{R}$ and E with the property that a shortest path between two designated vertices of this graph yields a minimum SPN placement. Such a path can be found using standard shortest path techniques, such as Dijkstra's algorithm. The auxiliary graph $\mathcal{D} = (V_\mathcal{D}, E_\mathcal{D})$ is defined as follows. There is a vertex in $V_\mathcal{D}$ for every point on the perimeter of a region that is also a point of the Hanan grid induced by the corners of the regions in $\mathcal{R}$, and two additional vertices v_s and v_t. In the following, we say "a vertex v of region R_i" to indicate a vertex corresponding to a point of R_i. There is an edge in $E_{\mathcal{D}}$ from v_s to every vertex of R_s, and from every vertex of R_t to v_t. Also, let u be a vertex of R_i and R_j be a region such that $(R_i, R_j) \in E$. For every segment composing the perimeter of R_j, there is an edge in $E_{\mathcal{D}}$ from u to its closest vertex on that segment. Furthermore, there is an edge in $E_{\mathcal{D}}$ from u to the next vertex along the perimeter of R_i, in both directions. We assign a cost to an edge $(u, v) \in E_{\mathcal{D}}$ equal to 0 if either $u = v_s$ or $v = v_t$, and equal to $\|uv\|$ otherwise. The following theorem shows that a shortest path between v_s and v_t in $\mathcal{D}$ yields an optimum SPN placement.

Theorem 1. *Given a shortest path $P_{\mathcal{D}}$ from v_s to v_t in $\mathcal{D}$, let $\boldsymbol{p}$ be a placement as follows. For each region $R_i \in \mathcal{R}$, if R_i has vertices on $P_{\mathcal{D}}$, $\boldsymbol{p}$ contains the first of them. Otherwise, $\boldsymbol{p}$ contains one of its corners chosen arbitrarily. The placement $\boldsymbol{p}$ is an optimum SPN placement.*

Proof. Consider the vertices on $P_{\mathcal{D}}$ chosen as points of $\boldsymbol{p}$ in the order as they appear on $P_{\mathcal{D}}$. Since the regions of these points are connected in $\mathcal{G}$, $P_{\mathcal{D}}$ corresponds to an *st*-path P in $\mathcal{G}$. By triangle inequality, the cost of P in $\mathcal{G}_{\boldsymbol{p}}$ is at most the cost of $P_{\mathcal{D}}$. For the sake of contradiction, suppose there exists an optimum placement $\boldsymbol{q}$ and a shortest *st*-path Q in $\mathcal{G}_{\boldsymbol{q}}$ with cost smaller than the cost of P in $\mathcal{G}_{\boldsymbol{p}}$. Without loss of generality, we can assume the points in $\boldsymbol{q}$ to satisfy Lemma 1. Thus, every point of $\boldsymbol{q}$ corresponds to a vertex of $\mathcal{D}$. We construct a path $Q_{\mathcal{D}}$ from v_s to v_t in $\mathcal{D}$ as follows. The first edge is (v_s, q_s); after that, for every edge (R_i, R_j) on Q, consider the points $q_i, q_j \in \boldsymbol{q}$, the bounding box $\mathsf{B}_{q_i q_j}$, and the at most two segments on the perimeter of R_j on which q_j lies. By construction, q_i is connected in $\mathcal{D}$ to a vertex on both segments; let v be one of them chosen arbitrarily such that $v \in \mathsf{B}_{q_i q_j}$. We add to $Q_{\mathcal{D}}$ the path that from q_i goes to v, and follows the perimeter of R_j to q_j. By Proposition 1, the cost of this path is equal to $\|q_i q_j\|$. The last edge on $Q_{\mathcal{D}}$ is (q_t, v_t). To see that the cost of $Q_{\mathcal{D}}$ is equal to the cost of Q in $\mathcal{G}_{\boldsymbol{q}}$ it is sufficient to notice that the first and the last edge of $Q_{\mathcal{D}}$ have cost 0 and, for every edge (R_i, R_j) on Q, the sub-path from q_i to q_j in $Q_{\mathcal{D}}$ has cost equal to $\|q_i q_j\|$. This results in a contradiction, because we have then found a path from v_s to v_t with cost smaller than $P_{\mathcal{D}}$. □

The above theorem shows how to construct an optimum SPN placement once a shortest path between v_s and v_t in $\mathcal{D}$ is known. Since edge costs in $\mathcal{D}$ are greater or equal than 0, we can find such a path with Dijkstra's algorithm in time $\mathcal{O}(|V_{\mathcal{D}}| \log |V_{\mathcal{D}}| + |E_{\mathcal{D}}|)$. The sizes of $V_{\mathcal{D}}$ and $E_{\mathcal{D}}$ depend on the number of points on the perimeters of the regions that are the grid points of the Hanan grid induced by the corners of $\mathcal{R}$. To evaluate this number, consider a line of the Hanan grid. Each time this line intersects (cut in two nonempty parts) an orthogonal segment on the perimeter of a region, an additional vertex is introduced. Conversely, each segment of the perimeter of a region can in the worst case be intersected by every grid line orthogonal to it. If k is the maximum number of corners of a region in $\mathcal{R}$ (and therefore on the number of segments of its perimeter), and $|\mathcal{R}| = n$, the number of grid lines is $\mathcal{O}(nk)$. Thus, the number of grid points lying on the perimeter of one region is $\mathcal{O}(nk^2)$, and the size of $V_{\mathcal{D}}$ is $\mathcal{O}(n^2k^2)$. To evaluate the size of $E_{\mathcal{D}}$, consider an edge $(R_i, R_j) \in E$ and a vertex v of R_i. By construction, there is an edge from v to a vertex on each of the at most k segments on the perimeter of R_j. Furthermore, v is connected to at most two vertices on the perimeter of R_i. If we have $|E| = m$ edges and $\mathcal{O}(nk^2)$ vertices in each region, the size of $E_{\mathcal{D}}$ is $\mathcal{O}(mnk^3)$. Thus, computing a shortest path from v_s to v_t in $\mathcal{D}$ with Dijkstra's algorithm takes time $\mathcal{O}(n^2k^2 \log nk + mnk^3)$.

3 Minimum Spanning Tree with Neighborhoods

In the Minimum Spanning Tree Problem with Neighborhoods, or MSTN, we are given a set of regions $\mathcal{R} = \{R_1, \ldots, R_n\}$ and an underlying graph $\mathcal{G} = (\mathcal{R}, E)$. The problem asks for a placement $\boldsymbol{p}$ such that the cost of a minimum spanning tree in $\mathcal{G}_{\boldsymbol{p}}$ is smallest among all possible placements.

It is known [2] that, if distances are measured in L_2 norm and the neighborhood regions are disks, the MSTN problem does not admit an FPTAS unless $\mathsf{P} = \mathsf{NP}$. We will adapt their proof and show that MSTN does not admit an FPTAS for the L_1 metric even for non-overlapping axis-parallel segments.

The reduction is from the planar 3-SAT problem. Planar 3-SAT is a variant of 3-SAT where the graph associated with the formula is planar. The graph contains a vertex for each variable and each clause, and there is an edge from a variable to a clause if the clause contains a literal of that variable. Planar 3-SAT was shown to be NP-hard by a reduction from the standard 3-SAT problem [7]. Furthermore, it was shown that in the plane embedding used in the reduction there always exists a so-called *spinal path* passing through every vertex corresponding to a variable without crossing any edge of the graph. Knuth and Raghunathan [6] observed that there always is a simple embedding where the variables are arranged on a straight line (the spinal path), and the clauses are drawn as three legged segments completely above or below them, in a way such that none of the legs cross each other. Figure 2 shows an example of such an embedding.

The reduction starts from a plane embedding of an instance of planar 3-SAT and constructs an instance of MSTN such that a solution to the latter indicates whether the former is satisfiable. First, we define three types of gadgets: a gadget for each variable, a gadget for each clause, and a gadget for the spinal path. Then, we show how to replace each variable, clause and the spinal path with a

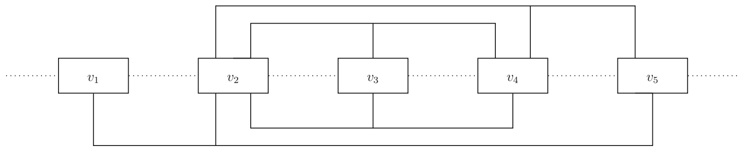

Fig. 2. A planar 3-SAT instance on 5 variables. Dashed lines are parts of the spinal path, solid lines are clauses.

corresponding gadget resulting in an instance of MSTN. From our construction, it will be easy to see that the size of the resulting MSTN instance is polynomially bounded. Finally, we provide two threshold values t_1 and t_2, with $t_1 < t_2$, and we prove that an optimum solution of the constructed MSTN instance has a cost smaller than t_1 if and only if the initial 3-SAT formula is satisfiable. If the formula is not satisfiable, the cost of an optimum solution of the MSTN instance is at least t_2. This proves that MSTN does not admit an FPTAS unless $\mathsf{P} = \mathsf{NP}$.

An important tool in the definitions of the gadgets is a so-called wire. A *wire* is a set of points (i.e., regions) placed in close succession, so that any minimum spanning tree (for any placement) will contain the edges connecting the points. To ensure this, it is sufficient to place two consecutive points in a wire at a suitably small distance. Since the edges between consecutive points in wires do not form a cycle, any minimum spanning tree in any placement will contain the edge connecting them. However, this suitably small distance must still be large enough to guarantee that a wire can be realized with a polynomial number of points. Since in the following construction the smallest non-zero distance between any two regions (other than those for the wires) is at least $d/2$, for a constant $d := 0.25$, a suitably small value for the points of a wire is, for example, $d/4$.

3.1 Reduction Gadgets

Variable Gadget. For each variable there are $k = 6c + 6$ segments of length α, where c is the maximum number of clauses in which the variable appears as a literal that are completely above or below the variable vertex in the embedding. Note that $k \geq 12$, because a variable appears at least once in a clause. In the following we specify the value of the parameter α more precisely, and we show it to be polynomial in the number of clauses and variables.

As illustrated in Fig. 3, the segments are placed along the perimeter of a rectangle with sides of length $3c\alpha + d$ and $3\alpha + d$. In its interior we place a wire for every two segments consecutive in clockwise order. Each of these wires ends on the line bisecting the angle formed by the corresponding segments; for parallel segments, the endpoint is at distance d from their common point. For perpendicular segments, the endpoint is at distance d from the intersection of the lines passing through the segments. We connect these wires in the bounded region in a tree-like structure as in Fig. 3. We call this arrangement of wires in the internal region a *k-tree.*

A placement of points inside a variable gadget is called a *configuration* if, for every two consecutive segments in clockwise order, the placement contains either their two closest points or their two farthest points. For a variable gadget there exist exactly two different configurations. To see this, consider two consecutive segments in a variable gadget and a configuration placement. If the placement contains their two closest points, we can place points in the remaining segments in exactly one way in order to obtain a configuration. Similarly, if the placement contains their two farthest points, we have exactly one way to place points in the remaining segments. We associate these two possible configurations with the two assignments to the variable.

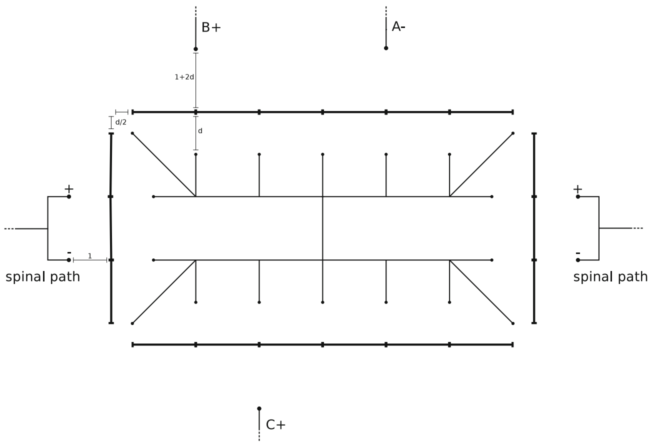

Fig. 3. A variable gadget with $k = 18$. The variable appears in A with negative sign and in B, C with positive sign. Thick lines are segments, the rest wires.

Clause Gadget. Clause gadgets are composed of at most three wires meeting at a single point following the embedding. As in Fig. 3, each wire of a clause gadget approaches the common point of two adjacent horizontal segments of a variable gadget. Clauses that are located above the spinal path in the rectilinear embedding approach variable gadgets from above, while clauses that in the rectilinear embedding are located below the spinal path approach variable gadgets from below. Furthermore, clause wires approach a variable gadget in the same clockwise order as the edges connecting the variable vertex to the corresponding clauses in the rectilinear embedding.

A clause wire terminates at distance $1 + 2d$ from the common point of the approached segments along the vertical line passing through it. The approached segments are chosen such that their common point is contained in a configuration satisfying the clause. That is, an edge with cost $1 + 2d$ connects the clause wire to the segments in a configuration placement satisfying the clause.

Spinal Path Gadget. The spinal path gadget consists of wires following the embedding of the planar 3-SAT instance. As in Fig. 3, the spinal path gadget approaches every variable gadget twice, once from the left and once from the right. For each side, the spinal path wire is split in two parts, each approaching two adjacent vertical segments. The point at which a part terminates is located at distance 1 from the common point of the approached segments along the horizontal line passing through it.

3.2 The Reduction

Given a rectilinear embedding of an instance of planar 3-SAT, we create an instance of MSTN and provide two threshold values t_1, t_2, with $t_1 < t_2$. We show that if the 3-SAT instance is satisfiable, then there is a placement with a minimum spanning tree of cost at most t_1, and if the 3-SAT instance is unsatisfiable, then the cost of a minimum spanning tree for any placement is at leat t_2.

Theorem 2. *MSTN with L_1 metric and axis-parallel segments is* APX*-hard.*

Proof. To create an instance of MSTN, replace in the given embedding every variable, clause, and the spinal path with a gadget as explained above. The wires forming the spinal path, the m clause gadgets and the k-trees in the internal region of each variable gadget have a fixed cost in every MST, denoted as c_{wires}. The remaining cost of the spanning tree is given by connecting the segments of the variable gadgets to the k-trees and the spinal path and clause wires.

Suppose there exists a satisfying assignment. Then, we place points in each variable gadget in a configuration according to its value in the assignment. We provide an upper bound t_1 on the cost of a minimum spanning tree in this placement by constructing a spanning tree and evaluating its cost. For each pair of consecutive segments having their closest points in the placement, the spanning tree connects them to the k-tree of the corresponding variable with cost d. If there is a total of K segments among all variable gadgets, the spanning tree requires a cost of $(K/2)d$ to connect all of them to the k-trees (note that K is even). For each clause gadget, consider a variable satisfying it in the assignment. We connect the corresponding endpoint of the clause wire to one of the segments it approaches with an edge with cost $1 + 2d$. Overall, the cost for connecting all the clause wires to the tree is $m(1 + 2d)$. For each part of the spinal path gadget approaching a variable gadget, exactly one of its endpoint approaches a point of the placement. The spanning tree contains the $2n$ edges of cost 1 connecting them. Overall, the cost of an optimum MSTN solution in case a satisfying assignment exists is therefore at most

$$t_1 := c_{wires} + (K/2)d + (1 + 2d)m + 2n.$$

If there is no satisfying assignment, we show that the cost of an optimum MSTN solution is at least $t_2 := t_1+d$. To see this, consider an optimum placement where every point in a segment is one of its extreme points. The existence of such an optimum placement is guaranteed by the fact that wires approaching variable gadgets and wires of the k-trees terminate either to the left or to the right of horizontal segments, and above or below vertical segments.

We first provide an upper bound on the minimum spanning tree cost for such a placement. By constructing a spanning tree and evaluating its cost in the placement. Then, we use this upper bound to show that, in every minimum spanning tree, clause wires are connected to the tree either with an edge from one of the endpoints to one of the approached segments, or with an edge from one of its endpoints to the approached k-tree endpoint. Finally, we show that the cost of any optimum MSTN solution is at least t_2.

The spanning tree contains the wires composing the spinal path, the clauses, and all k-trees. Every segment in a variable gadget is connected to an endpoint of the k-tree with an edge of cost d. Every part of a wire of the spinal path approaching a variable gadget is connected to one of the approached segments by one of its endpoints with an edge with cost 1. Similarly, an endpoint of each clause wire chosen arbitrarily is connected to a k-tree of a variable appearing in that clause with an edge with cost $1 + 3d$. The cost of such a spanning tree is

$$c_{wires} + 2n + m(1 + 3d) + Kd. \tag{4}$$

We now prove that, in every minimum spanning tree, each clause is connected to it either with an edge from one of its endpoint to an approached segment, or with an edge from one of its endpoints to the corresponding k-tree endpoint. Suppose this is not the case, and there is a clause whose endpoint in the MST is connected neither to an approached segment, nor to the corresponding k-tree endpoint. By construction, the next closest object is located at distance at least α, where α is the above defined length of the segments of the variable gadgets. Since the spinal path, clauses and k-trees wires are part of every MST, setting

$$\alpha := 2n + m(1 + 3d) + Kd + 1$$

we get a contradiction, because the cost of a minimum spanning tree would then be greater than (4). Thus, in every minimum spanning tree every clause is connected via one of its endpoints to one of the approached regions.

Finally, we show that if the formula is not satisfiable, any optimum MSTN solution has cost greater than t_2. Clearly, we cannot provide a configuration for each variable gadget such that every clause where that variable appears can be connected to it with an edge with cost $1 + 2d$, otherwise the formula would be satisfiable. Therefore, in an optimum solution, either at least one variable gadget is not set in a configuration, or every variable gadget is in a configuration and for at least one clause no wire endpoint approaches a point of the placement.

In the former case, the cost of a minimum spanning tree is at least $c_{wires} + 2n + (K/2)d + j(1 + 2d) + (m - j)\delta$, where j is the number of clauses that can be satisfied by the assignment corresponding to the configuration and δ is the minimum cost necessary to connect a clause that is not satisfied by the assignment. By the above, we know that in any minimum spanning tree a clause wire is connected to one of the approached segments or the corresponding k-tree endpoint. Since every variable gadget is in a configuration, the smallest distance δ between a non satisfied clause wire and a point in the placement is at least $1 + 3d$. Thus, any optimum MSTN solution where every variable gadget is set in a configuration results in a minimum spanning tree with cost at least t_2.

In the latter case, there exists at least one variable gadget that is not in a configuration. Let then a be the overall number of segments for which the point in the placement is not the closest or the farthest to the point in one of the consecutive segments. Note that a is even, therefore $a \geq 2$, and the cost of a spanning tree is at least

$$c_{wires} + 2n + \frac{(K + a)}{2}d + m(1 + 2d) \geq t_2.$$

Suppose now that there exists an FPTAS for MSTN. Given an instance of planar 3-SAT, we construct the gadget presented above and calculate t_1.

We then set a parameter $\epsilon < d/t_1$, so a $(1+\epsilon)$-approximate solution to the MSTN problem would tell us whether the cost of the corresponding optimum solution is smaller than t_1 or greater than t_2, and thus, whether there exists a satisfying assignment for the planar 3-SAT instance. □

4 Conclusions

We considered the Shortest Path Problem and the Minimum Spanning Tree Problem with Neighborhoods in the L_1 metric and showed that the former can be solved efficiently if the neighborhood regions are rectilinear polygons not necessarily convex, while the latter does not admit a PTAS unless $\mathsf{P} = \mathsf{NP}$ even if the regions are axis-parallel segments. An interesting open problem is to consider variants of SPN and MSTN where the goal is to find placements *maximizing* the cost of shortest paths and minimum spanning tree, respectively.

Acknowledgments. This work was supported by the EU FP7/2007-2013 (DG CONNECT.H5-Smart Cities and Sustainability), under grant agreement no. 288094 (project eCOMPASS) and by the Alexander von Humboldt-Foundation.

References

1. Ahadi, A., Mozafari, A., Zarei, A.: Touring disjoint polygons problem is NP-hard. In: Widmayer, P., Xu, Y., Zhu, B. (eds.) COCOA 2013. LNCS, vol. 8287, pp. 351–360. Springer, Heidelberg (2013)
2. Dorrigiv, R., Fraser, R., He, M., Kamali, S., Kawamura, A., López-Ortiz, A., Seco, D.: On minimum-and maximum-weight minimum spanning trees with neighborhoods. In: Erlebach, T., Persiano, G. (eds.) WAOA 2012. LNCS, vol. 7846, pp. 93–106. Springer, Heidelberg (2013)
3. Dror, M., Efrat, A., Lubiw, A., Mitchell, J.S.B.: Touring a sequence of polygons. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC), pp. 473–482 (2003)
4. Dumitrescu, A., Mitchell, J.S.B.: Approximation algorithms for TSP with neighborhoods in the plane. In: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete algorithms (SODA), pp. 38–46 (2001)
5. Elbassioni, K.M., Fishkin, A.V., Mustafa, N.H., Sitters, R.A.: Approximation algorithms for euclidean group TSP. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1115–1126. Springer, Heidelberg (2005)
6. Knuth, D., Raghunathan, A.: The problem of compatible representatives. SIAM J. Discrete Math. **5**(3), 422–427 (1992)
7. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. **11**(2), 329–343 (1982)
8. Löffler, M., Kreveld, M.: Largest and smallest convex hulls for imprecise points. Algorithmica **56**(2), 235–269 (2010)
9. Löffler, M., van Kreveld, M.J.: Largest bounding box, smallest diameter, and related problems on imprecise points. Comput. Geom. **43**(4), 419–433 (2010)

10. Pan, X., Li, F., Klette, R.: Approximate shortest path algorithms for sequences of pairwise disjoint simple polygons. In: Proceedings of the 22nd Canadian Conference on Computational Geometry (CCCG), pp. 175–178 (2010)
11. Yang, Y., Lin, M., Xu, J., Xie, Y.: minimum spanning tree with neighborhoods. In: Kao, M.-Y., Li, X.-Y. (eds.) AAIM 2007. LNCS, vol. 4508, pp. 306–316. Springer, Heidelberg (2007)