Journal of Universal Computer Science, vol. 15, no. 5 (2010), 622-652
submitted: 24/4/09, accepted: 12/2/10, appeared: 1/3/10 © *J.UCS*

NP-completeness and FPT Results for Rectilinear Covering Problems

Vladimir Estivill-Castro
(Griffith University, Brisbane, Australia
v.estivill-castro@griffith.edu.au)

Apichat Heednacram
(Griffith University, Brisbane, Australia
apichat.heednacram@student.griffith.edu.au)

Francis Suraweera
(Griffith University, Brisbane, Australia
f.suraweera@griffith.edu.au)

Abstract: This paper discusses three rectilinear (that is, axis-parallel) covering problems in d dimensions and their variants. The first problem is the RECTILINEAR LINE COVER where the inputs are n points in $\mathbb{R}^d$ and a positive integer k, and we are asked to answer if we can cover these n points with at most k lines where these lines are restricted to be axis parallel. We show that this problem has efficient fixed-parameter tractable (FPT) algorithms. The second problem is the RECTILINEAR k-LINKS SPANNING PATH PROBLEM where the inputs are also n points in $\mathbb{R}^d$ and a positive integer k but here we are asked to answer if there is a piecewise linear path through these n points having at most k line-segments (links) where these line-segments are axis-parallel. We prove that this second problem is FPT under the assumption that no two line-segments share the same line. The third problem is the RECTILINEAR HYPERPLANE COVER problem and we are asked to cover a set of n points in d dimensions with k axis-parallel hyperplanes of $d-1$ dimensions. We also demonstrate this has an FPT-algorithm. Previous to the results above, only conjectures were enunciated over several years on the NP-completeness of the RECTILINEAR MINIMUM LINK TRAVELING SALESMAN PROBLEM, the MINIMUM LINK SPANNING PATH PROBLEM and the RECTILINEAR HYPERPLANE COVER. We provide the proof that the RECTILINEAR MINIMUM LINK TRAVELING SALESMAN PROBLEM and the RECTILINEAR MINIMUM LINK SPANNING PATH PROBLEM are NP-complete by a reduction from the ONE-IN-THREE 3-SAT problem. The NP-completeness of the RECTILINEAR HYPERPLANE COVER problem is proved by a reduction from 3-SAT. This suggests dealing with the intractability just discovered with fixed-parameter tractability. Moreover, if we extend our problems to a finite set of orientations, our approach proves these problems remain FPT.

Key Words: Computational Geometry, Restricted Orientations, Parameterized Complexity.
Category: F.2, F.2.2, F.1

1 Introduction

Derick Wood and his colleagues studied many computational geometric problems with restricted orientations [Rawlins and Wood, 1988, 1991, Fink and Wood, 1996, 2004]. Here, we study the LINE COVER and MINIMUM LINK SPANNING PATH PROBLEM problems where a set of k lines is restricted to the lines having a finite number of orientations.

The LINE COVER problem finds the minimum number of straight lines possible to cover a set S of n points in the plane where these lines could be in any orientation. This problem has been known to be NP-hard [Megiddo and Tamir, 1982] for over 20 years (in fact, APX-hard [Kumar et al., 2000]). A *rectilinear line* is an axis-parallel line that is either horizontal or vertical. When the LINE COVER problem is restricted to only rectilinear lines we call this the RECTILINEAR LINE COVER problem. This problem remains very difficult to solve. In fact, Hassin and Megiddo [1991] showed that the problem of finding the minimum number of rectilinear lines required to cover a set of points in 3 dimensions (or higher) is NP-complete but solvable in polynomial time for 2D.

Consider a path that is made up from more than one line-segment, the number of line-segments in a given path is often called the *link length*. A path having a minimum number of line-segments or links is called a *minimum link path* [Wagner, 2006]. The minimum link path is equivalent to a path having the minimum number of bends (or links) and it could be self intersecting. Therefore, the MINIMUM LINK PATH PROBLEM is sometimes referred to as MINIMUM BEND PATH PROBLEM. However, the number of bends in the path is actually one less than the number of line-segments that make up the path. In general, the MINIMUM BEND PATH PROBLEM attempts to find a path between two distinct points that has a minimum number of bends. This problem is only meaningful with obstacles. Note that finding the path between two points is different from finding the path that passes through a set of points. The latter path is called a *spanning path* [Collins, 2004, Bereg et al., 2009] and the problem of finding the spanning path that covers a set of points such that this path has the minimum number of links is known as the MINIMUM LINK SPANNING PATH PROBLEM. This problem is NP-complete by a reduction from EDGE EMBEDDING ON A GRID [Bereg et al., 2009]. An alternate proof of NP-completeness was given by Arkin et al. [2003] who showed that the problem of finding a minimum link tour is NP-complete by a reduction from LINE COVER.

When the MINIMUM LINK SPANNING PATH PROBLEM is restricted to allow only rectilinear lines (axis-parallel lines), we call this the RECTILINEAR MINIMUM LINK SPANNING PATH PROBLEM. It was conjectured in 2007 to be NP-complete, and again the conjecture appeared in 2009 [Bereg et al., 2009]. Bereg et al. [2009] suspected that this problem is NP-complete since Hassin and Megiddo [1991] showed that the RECTILINEAR LINE COVER problem in 3 dimensions (or higher) was NP-complete but they left this issue also open. The problem of traversing a set of points and minimizing the number of links in the tour where the links are restricted to being rectilinear lines is called the RECtilinear Minimum Link Traveling Salesman Problem. Wagner [2006] raised and left open whether this problem is NP-complete or if it is polynomially solvable. Thus, one goal of this paper is to provide the NP-completeness proofs of the Rectilinear Minimum Link Spanning Path Problem and of the Rectilinear Minimum Link Traveling Salesman Problem. A closely related problem is the Rectilinear Hyperplane Cover problem: covering n points in $\mathbb{R}^d$ with the minimum number of rectilinear hyperplanes of dimension $d-1$. The complexity of this problem is polynomial for 2D [Hassin and Megiddo, 1991], but its complexity is unknown even for 3D [Hurtado, 2008]. The problem is known to be NP-complete if we do not require the hyperplanes to be rectilinear [Megiddo and Tamir, 1982]. We first show that the rectilinear version is also NP-complete. Note that the algorithms of Langerman and Morin [Langerman and Morin, 2005] do show that the problem is FPT. However, we provide improved FPT-algorithms that give hope for tractability despite our NP-completeness result.

Minimizing the number of bends in the tour is desirable in applications such as VLSI and the movement of heavy machinery because turns are considered very costly. In the context of VLSI design, the number of bends on a path affects the resistance and hence the accuracy of expected timing and voltage in chips [Lee et al., 1996]. The rectilinear version of the problems received considerable attention during 1990's [Lee et al., 1990, de Berg et al., 1992, Lee et al., 1996] and recently [Collins, 2004, Arkin et al., 2005, Wagner, 2006, Bereg et al., 2009], since much of the interest in the rectilinear setting have been motivated by applications in VLSI. In this setting, paths and tours that self-intersect are feasible solutions.

In this paper, we study the Rectilinear Line Cover, the Rectilinear Minimum Link Spanning Path Problem and the Rectilinear Hyperplane Cover in $d \geq 3$ dimensions.[1] We acknowledge that when $d > 3$ dimensions the problems considered are interesting from a theoretical point of view mainly. Our solutions will be based on parameterized complexity theory [Downey and Fellows, 1999, Niedermeier, 2006, Flum and Grohe, 2006]. In classical complexity theory, NP-completeness is essentially a tag for intractability. However, parameterized complexity theory offers fixed-parameter tractable (FPT) algorithms, which require polynomial time in the size n of the input to find exact answers, although exponential time may be required on a small parameter k.

The paper is organized as follows. In Section 2 we improve the previously known FPT result Langerman and Morin [2005] for the Rectilinear Line Cover in higher dimensions (Theorem 1). In Section 3 we prove that the Line Cover problem when restricted to a finite number of orientations, is FPT (The-

[1] The classical theory of computation as well as the theory of parameterized complexity, focuses on the decision version of these problems. Without further explicit clarification, we will follow this approach as well.

orem 5). In Section 4 we prove that the Rectilinear Minimum Link Traveling Salesman Problem and Rectilinear Minimum Link Spanning Path Problem in 3 dimensions are NP-complete (Theorem 15 and Theorem 16). In Section 5 we then give an FPT algorithm for the latter problem under the assumption that no two line-segments share the same line (Theorem 17). In Section 6 we prove that the Minimum Link Spanning Path Problem, when restricted to a finite number of orientations, is also FPT (Theorem 18). In Section 7 we prove that the Rectilinear Hyperplane Cover problem is NP-complete (Theorem 20). We then give an FPT-algorithm for this problem (Theorem 21). We also prove that the Rectilinear Hyperplane Cover problem, when restricted to a finite number of orientations, is FPT (Theorem 22). Finally, in Section 8 we summarize and conclude with open problems.

2 Rectilinear Line Cover Problem

2.1 Using a Bounded-Search-Tree

Given a set S of $n = |S|$ points in $\mathbb{R}^d$ and a positive integer k, we are asked if it is possible to cover these n points in S with at most k lines where the lines are restricted to be axis parallel. In this section, we will use the bounded-search-tree technique to show that this problem belongs to the class FPT.

First we discuss the problem in 3 dimensions. Consider the search tree in Fig. 1. We choose a point $p \in S$ and we explore the possibilities of a cover at this given point. In a YES-instance, this point must lie on at least one of the lines NS, EW or UD[2], thus we can construct a search tree with fan-out three. The task is to explore exhaustively these three candidate orientations. At each node of the tree, we select a point p not already covered. We expand 3 branches corresponding to the NS-orientation, the EW-orientation and the UD-orientation, respectively and in each brach, we assign one of the three orientations to the point p. The point p is marked as covered in the recursive calls to cover the rest of S. The points that fall into the same line with p are also marked as covered so that we do not consider these points again in the next recursive call. We keep track of how many assignments have been made and we do not assign more than k orientations (k rectilinear lines). That is, every recursive call reduces the value of k by one. Each branch of the tree stops when the upper bound has been reached or when every point is marked as covered. If the algorithm finds a leaf with k lines that cover every point in S, it answers YES. If all leaves result in no cover, then the algorithm answers NO. The algorithm is correct, because if there is a cover with k lines, we can follow the path in the tree to the leaf that agrees with the cover. We examine how each point in the node of the path is covered by the witness cover.

[2] N, S, E, W, U, and D stand for North, South, East, West, Up and Down respectively.

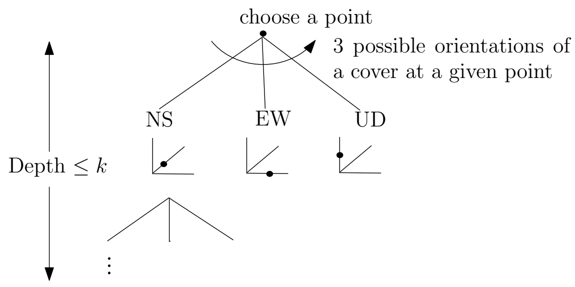


Figure 1: A search tree of the Rectilinear Line Cover in 3D with depth k.

Therefore, the depth of the tree is at most k. Note that the fan-out is a constant $d = 3$ and the depth of the tree is bounded by the parameter k and not the input size n. The number of nodes in the tree is bounded by 3^k, thus we obtain an FPT-algorithm because its time complexity is linear in the size of the input, although it is exponential in the parameter.

Note that when the problem is moved to d dimensions, the search tree will have to explore d candidate orientations at the given point p. Consider a set C of candidate orientations with the following members $\{x_1, x_2, x_3, \ldots, x_d\}$ where $x_i \in C$ is a straight line parallel to the i-th axis. The task is again to select a point p for which a cover orientation x_i for $1 \leq i \leq d$ has not been determined, and assign one of the d orientations to the point p and continue as in the 3 dimensional case. Recall that the Rectilinear Line Cover in 2 dimensions is polynomially solvable but in 3 dimensions (or higher) this problem is NP-complete [Hassin and Megiddo, 1991].

Theorem 1. *The* Rectilinear Line Cover *in $d \geq 3$ dimensions can be solved in $O(d^k n)$ time.*

Proof. The work at the leaves and the nodes is linear in n and the number of nodes is $O(d^k)$. The total time complexity is then $O(d^k n)$. The problem is FPT because it can be decided in running time $f(k) \cdot n^{O(1)}$. □

Note that the FPT-algorithms of Langerman and Morin [2005] are generic enough to the Rectilinear Line Cover problem but their complexities do not improve. Their first algorithm uses a bounded-search-tree and requires $O(k^{dk} n)$ time while the other uses kernelization and requires $O(k^{d(k+1)} + n^{d+1})$ time. Our FPT-algorithm offers better complexity than the FPT-algorithms of Langerman and Morin when specialized to the rectilinear case. Next we will exploit the kernelization approach to solve the Rectilinear Line Cover problem and reduce the time complexity even further.

2.2 Using Kernelization

This approach reduces the problem to a smaller one using reduction rules. The rules are applied repeatedly until none of the rules applies. If the result is a problem of size no longer dependent on n, but only on k, then the problem is kernelizable because the kernel can be solved by exhaustive search in time that depends only on the parameter (even if it is exponential time). We now present our reduction rules for the RECTILINEAR LINE COVER problem.

If k is large enough, we could use one rectilinear line for each point in S.

Reduction Rule 1 *If $k \geq n$, then the answer is yes.*

Consider the simplest case when $k = 1$, we would need to have all n points in S in a line.

Reduction Rule 2 *If $k = 1$, then the answer is yes if and only if all points in S lie on only one rectilinear line, otherwise the answer is no.*

If the input S has repeated points, then we can remove duplicated points.

Reduction Rule 3 *If S has repeated points, then S can be simplified to a set with no repetitions and the answer for the simplified set is an answer for the original input [Langerman and Morin, 2005].*

If a rectilinear line covers a lot of points, it must be used in the cover (specifically, we need at least $k + 1$ points).

Reduction Rule 4 *If there is a rectilinear line with $k + 1$ or more co-linear points, place the line through them in the cover and remove them from further consideration. If there is a line with $k + 1$ or more co-linear points and the line is not rectilinear, then the answer is automatically no.*

Similar to the rule for arbitrary lines [Langerman and Morin, 2005], the above rule is correct because if the rectilinear line through the $k + 1$ different points was not in the cover then we would need more than k rectilinear lines to cover just these points. Thus, if the set accepts a cover with k lines, any rectilinear line with $k+1$ or more points must be in the cover. If the line through the $k+1$ points is not rectilinear, then we would also need more than k rectilinear lines to cover just these points. Since we do not accept the non-rectilinear line into the cover and we do not accept more than k rectilinear lines, the answer here is immediately no.

The reduction rules above can be formulated into a kernel lemma. The result below leads to a quadratic size kernel.

Lemma 2. *If there is a subset S_0 of S such that $|S_0| \geq k^2 + 1$, and the largest number of co-linear points of S_0 on a rectilinear line is at most k, then the answer is no.*

Proof. Similar to the rule for arbitrary lines [Grantson and Levcopoulos, 2006], if all rectilinear lines covered at most k points, any cover of S_0 with k lines would cover at most k^2 points. This means we cannot cover S_0, let alone S with k lines. □

Next, we let L_2 be the set of all rectilinear lines that cover at least 2 points in S, that is, a rectilinear line l is in L_2 if and only if there are 2 or more co-linear points of S in l. We let $cover(L_2)$ be all points in S covered by a line in L_2. With this notation, we now describe two more reduction rules.

Reduction Rule 5 *Let p be a point in $S \setminus cover(L_2)$, i.e. $p \in S$, but $p \notin cover(L_2)$. Then, the original instance has a yes answer if and only if the instance $S \setminus \{p\}$ with parameter $k-1$ has a yes answer.*

This reduction rule essentially says that if we can find a point that is never covered by a rectilinear line that covers 2 or more points, then we can reduce to a smaller problem that ignores this point and uses one less line.

Proof. Since p is the only point in a rectilinear line, any cover with $k-1$ lines of $S \setminus \{p\}$ can be made a cover of $S \cup \{p\}$ with one more rectilinear line. Also a rectilinear cover C of S must use a rectilinear line l_p over p but $C \setminus \{p\}$ has $k-1$ rectilinear lines and covers $S \setminus \{p\}$. □

Reduction Rule 6 *Let q be a point in $cover(L_2)$. Suppose that there is no other line in L_2 besides l_q that also covers q (i.e. $\{l \in L_2 | l \text{ covers } q\} = \{l_q\}$). Then, the original instance has a yes answer if and only if the instance $S \setminus cover(l_q)$ with parameter $k-1$ has a yes answer.*

This reduction rule is similar to the previous rule. That is, if we can find a point that can only be covered by one line and not by any other lines (even though this line is in L_2), then we can reduce to a smaller problem that ignores the points that lie on this line and reduce our budget by one.

Proof. If a cover C did not use l_q, it must use the other rectilinear line $l_i \notin L_2$ to cover q. The line $l_i \in C$ can cover only the point q, thus $C \cup \{l_i\} \setminus \{l_q\}$ is a cover of the same cardinality that uses l_q. □

In general, the more reduction rules we can apply, the smaller the kernel is. Although the size of our kernel remains quadratic in theory, in practical circumstances the incorporation of these additional rules can result in a kernel that has size significantly fewer than k^2 points (or the size even goes down to zero) [Estivill-Castro et al., 2009]. We apply the above reduction rules in the preprocessing phase. When we apply these rules one after another until none of the rules applies, we have a problem kernel whose size is determined theoretically by Reduction Rule 4.

Lemma 3. *Any instance (S, k) of the of the* RECTILINEAR LINE COVER *problem can be reduced to a problem kernel S' of size $O(k^2)$.*

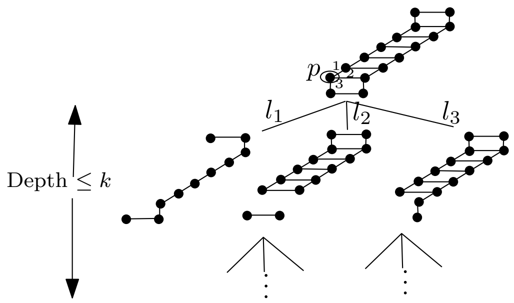


Figure 2: Example of a ladder instance ($n = 16, k = 4$) where the bounded-search-tree is used in conjunction with Reduction Rule 6.

Proof. Let S' be the set of points after we cannot repeat the removal of points covered by a rectilinear line covering $k+1$ or more points (Reduction Rule 4). If we repeated the removal more than k times, we know that it is a NO-instance. If we repeated no more than k times and it is a YES-instance, every line covers at most k points; thus, any cover with k lines would cover at most k^2 points. $\square$

Now, we can invoke the bounded-search-tree approach in the previous section to solve the kernel. Recall that this algorithm runs in $O(d^k n)$ time. However, the exponential term now is no longer multiplied by n since the input instance will be the reduced instance (the kernel). Therefore, we can improve the exponential term to $(d^k k^2)$ time and obtain a better algorithm. However, we will introduce an alternative approach to solve the kernel using the bounded-search-tree technique in conjunction with the application of Reduction Rule 6. Consider an alternative bounded-search-tree that focuses on the set of candidate lines $L_p \subset L_2$ that go through a given point $p \in S'$ where S' is the kernel and $|S'| \leq k^2$. Note that the number of axis-parallel lines that go through any point in $\mathbb{R}^d$ is at most d. Therefore, the branching of the search tree is bounded by the value of d. Our strategy is to choose a point $p \in S'$ which is covered by a set of axis-parallel lines $L_p \subset L_2$, and then we test each of the candidate lines in L_p. The number of recursive calls in each level of the tree is controlled by $|L_p| \leq d$. At each node of the tree, we also invoke Reduction Rule 6. If the rule applies, this will further reduce the size of the instance before we move to the next level of the tree. We illustrate the idea in Fig. 2. There are two main tasks to be performed at each node of the tree.

1. We pick a point p, and test the candidate lines that go through p. These are the lines l_1, l_2 and l_3 in Fig. 2. If we pick l_1, we mark all the points that lie on l_1 including p as covered in a recursive call to cover the rest of $S' \setminus cover(l_1)$. We remove l_1 from L_2 and remove the covered points from $cover(L_2)$.

2. We check if Reduction Rule 6 can be applied on the instance $S' \setminus cover(l_i)$.

The rule can be exhaustively applied no more than k times if the instance is a YES-instance. Although this technique appears to increase the cost to the algorithm, it can reduce the size of the instance and thus could essentially reduce the depth of the tree. The best case scenario is shown in the left branch of Fig. 2 where the rule removes all the remaining candidate lines and the tree stops at the first level of the tree. The worst case is when the rule cannot be applied at any level of the tree. However, this is not exactly true because at the $(k-1)$-th level, there will be at most 2 lines left to cover. When one of the lines is taken, Reduction Rule 6 will automatically remove the remaining line and there is no need to proceed to the k-th level. Hence, effectively the depth of the search tree is at most $k-1$.

We repeat the two tasks above in each recursive call until there are no points left to cover or the tree has reached the upper bound which is the depth of $k-1$. Here if every point in S' is covered, we answer yes. Otherwise, we answer no.

In addition, we can incorporate a greedy strategy like sorting heavy-lines (lines that cover many points) that go through p to speed up the process. Alternatively, we may use a hashing scheme to avoid exploring the same candidate line in the search tree twice. These strategies may be of interest for practical implementation and experimentation of expected-case performance and we do not elaborate on the idea here. We now conclude our result as follows.

Lemma 4. *The* Rectilinear Line Cover *in* $d \geq 3$ *dimensions can be decided in* $O(n^2d^2 + d^{k-1}k^2)$ *time.*

Proof. Reduction Rule 1 can be performed in constant time. Reduction Rule 2 can be applied in $O(nd)$ time. Reduction Rule 3 can be checked in $O(n^2d)$ time. Reduction Rule 4 can be applied if using a naive approach in $O(n^2d^2)$ time by computing $O(dn)$ rectilinear lines and checking if these lines cover at least $k+1$ points. Hence, the preprocessing phase can be carried out in $O(n^2d^2)$ time. The depth of the new bounded-search-tree is at most $k-1$ and the fan-out is at most d. The number of nodes in the tree is $O(d^{k-1})$. The work at each node is to remove at most k lines each having at most k points and update which points are covered by which lines (i.e. update a set L_2). Note that the set L_2 can be computed in the preprocessing phase which is already bounded by the term $O(n^2d^2)$. To update L_2, we require k^2 time to update what points are covered by which remaining lines. Therefore, the overall work of the bounded-search-tree is $O(d^{k-1}k^2)$ time and the total time complexity is $O(n^2d^2 + d^{k-1}k^2)$. □

3 Line Cover Restricted to a Finite Number of Orientations

We now extend our result by considering the Line Cover problem where a set of k covering lines is restricted to the lines having a finite number of orientations,

m. This is the case where these k lines must have their orientations taken from a given finite set $R = \{r_1, r_2, \ldots, r_m\}$ where $|R| = m$ [Güting, 1983, Rawlins and Wood, 1991].

We can construct the search tree with m branches similar to the one in Fig. 1 where each branch represents a cover orientation that is a straight line with a slope r_i, for $1 \leq i \leq m$. We then explore the possibilities of a cover at a given point $p \in S$ as we did before in the previous section. Since any point must lie on at least one of the m orientations (slopes), we select a point p for which a cover orientation has not been determined, and assign one of the m orientations to the point p. The point p along with the points that fall into the same cover with p are marked as covered in the recursive call. After k recursive calls, we have a leaf where we either have a cover or not. If all leaves do not lead to a cover, this is a NO-instance. A leaf with a cover provides the certificate of a YES-instance. The depth of the tree is at most k. The number of nodes in the tree is bounded by m^k. Thus we obtain an FPT-algorithm because its time complexity is polynomial in the size of the input n, although it is exponential in the parameter k.

Theorem 5. *The* LINE COVER *problem when restricted to the lines having a finite number $m \geq 2$ of orientations is FPT.*

4 NP-Completeness Results

We will first prove that the RECTILINEAR MINIMUM LINK TRAVELING SALESMAN PROBLEM in 3 dimensions is NP-complete by a reduction from ONE-IN-THREE 3-SAT, a known NP-complete problem. It is easy to show that the RECTILINEAR MINIMUM LINK TRAVELING SALESMAN PROBLEM belongs to NP. The verification algorithm checks whether a given set of $|S|$ points in $\mathbb{R}^3$ can constitute a tour with at most k links and every link in the tour is axis-parallel. This can be performed in polynomial time.

Now, we show that the ONE-IN-THREE 3-SAT problem can be reduced in polynomial time to RECTILINEAR MINIMUM LINK TRAVELING SALESMAN PROBLEM. The ONE-IN-THREE 3-SAT problem is to decide whether there exists a satisfying assignment to the variables so that each clause has exactly one true literal and thus exactly two false literals. In contrast, the 3-SAT problem only requires that every clause has at least one true literal. The ONE-IN-THREE 3-SAT problem remains NP-complete even when no clause contains a negated literal [Garey and Johnson, 1979, LO4]. Hence, to simplify our reduction, we will assume that no literals are negated in our problem.

Suppose that we are given an instance of ONE-IN-THREE 3-SAT with a set of m clauses $E_j = w'_j \vee w''_j \vee w'''_j$ for $j = 1, 2, \ldots, m$ where $\{w'_j, w''_j, w'''_j\} \subset \{u_1, \bar{u}_1, \ldots, u_{n'}, \bar{u}_{n'}\}$. From the instance $E_1 \wedge \ldots \wedge E_m$ of ONE-IN-THREE 3-SAT we construct an instance of RECTILINEAR MINIMUM LINK TRAVELING SALESMAN PROBLEM by gadget design. We represent each variable u_i by a set S_i of

$4400m$ points, for $i = 1, 2, \ldots, n'$. We represent each clause E_j with three points p'_j, p''_j, p'''_j, for $j = 1, 2, \ldots, m$. We let $P = \{p'_1, p''_1, p'''_1, \ldots, p'_m, p''_m, p'''_m\}$ be the set of points representing all clauses. The instance of Rectilinear Minimum Link Traveling Salesman Problem will have a set $S = P \cup \bigcup_{i=1}^{n'} S_i$ where $|S| = 4400mn' + 3m$. The set S_i representing u_i will be made of a gadget. Such a gadget will be forced to use $22m$ different covering line-segments parallel to one of the axes in $\mathbb{R}^3$. Because of this, rather than describing the points in S_i we will be describing these line-segments. The value $4400m$ is chosen so that there are many points in the covering line-segments that a rectilinear tour with minimum number of bends will be forced to use each of these line-segments.

The plan for the proof is to show that there exists a satisfying assignment for $E_1 \wedge \ldots \wedge E_m$ if and only if a set of $|S|$ points in 3D can be covered optimally by a rectilinear tour that has k links where k will be set to $34mn' + 2n' - 2$. The tour will be made up of two types of line-segments, the covering line-segments we alluded before which cover $\bigcup_{i=1}^{n'} S_i$ and line-segments that do not cover points in $\bigcup_{i=1}^{n'} S_i$. The second type of line-segments joins the first type of line-segments together. We will design the gadget (and the set S_i) representing u_i so that there are two ways of covering the points in S_i. These two ways will be used to represent a variable being assigned the value "true" or "false". To present our 3D gadget we require some notation.

Definition 6. The three possible types of line-segments parallel to an axis in $\mathbb{R}^3$ are called **forms** and are denoted by $line(x, y, *), line(x, *, z)$ and $line(*, y, z)$.

Thus, a point with a coordinate (x_0, y_0, z_0) can be covered only by $line(x_0, y_0, *)$, $line(x_0, *, z_0)$ or $line(*, y_0, z_0)$.

Definition 7. We denote the three axis-parallel planes with fixed x-value, fixed y-value, and fixed z-value as $\Pi(x, *, *)$, $\Pi(*, y, *)$, and $\Pi(*, *, z)$, respectively.

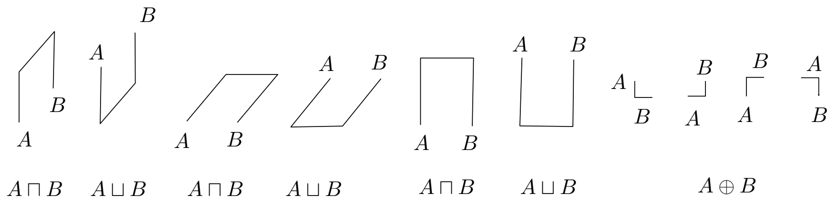
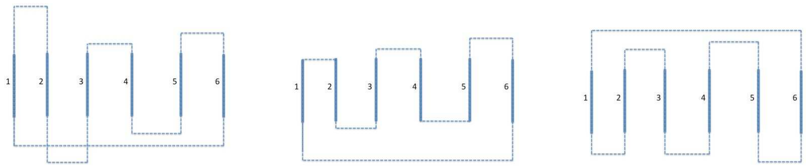


Figure 3: Definitions when joining two axis-parallel line-segments.

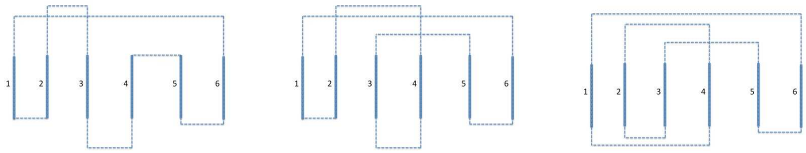
Definition 8. Given two line-segments of the form $A = (x_1, y_1, *)$ and $B = (x_1, y_2, *)$ that are in the plane $\Pi(x_1, *, *)$ we write $A \sqcap B$ when they are joined by one line-segment and two bends on the high z-values (the joining line-segment has the form $(x_1, *, z_\sqcap)$ and the z-value $z_\sqcap$ is larger than any z-value in A or B). We denote $A \sqcup B$ when they are joined by one line-segment $(x_1, *, z_\sqcup)$ and two



(a) A member of the equivalence class $line_1 \sqcap line_2$, $line_2 \sqcup line_3$, $line_3 \sqcap line_4$, $line_4 \sqcup line_5$, $line_5 \sqcap line_6$, $line_6 \sqcup line_1$.

(b) Another member of the equivalence class $line_1 \sqcap line_2$, $line_2 \sqcup line_3$, $line_3 \sqcap line_4$, $line_4 \sqcup line_5$, $line_5 \sqcap line_6$, $line_6 \sqcup line_1$.

(c) A member of the equivalence class $line_1 \sqcup line_2$, $line_2 \sqcap line_3$, $line_3 \sqcup line_4$, $line_4 \sqcap line_5$, $line_5 \sqcup line_6$, $line_6 \sqcap line_1$.



(d) Another member of the equivalence class $line_1 \sqcup line_2$, $line_2 \sqcap line_3$, $line_3 \sqcup line_4$, $line_4 \sqcap line_5$, $line_5 \sqcup line_6$, $line_6 \sqcap line_1$.

(e) It is not a member of the other equivalence classes but it respects where extensions are possible.

(f) It is not a member of the other equivalence classes and it enables other extensions.

Figure 4: Different tours with the same number of bends cover 6 vertical line-segments in 2D.

bends, and the z-value $z_\sqcup$ is smaller than any z-value in A or B (see Fig. 3). Similarly, given two line-segments of the form $A = (x_1, *, z_1)$ and $B = (x_2, *, z_1)$ that are in the plane $\Pi(*, *, z_1)$, we write $A \sqcap B$ when they are joined by one line-segment $(*, y_\sqcap, z_1)$ and two bends, and $y_\sqcap$ is larger than any y-value in A or B. While $A \sqcup B$ is when the line-segment $(*, y_\sqcup, z_1)$ joining them has $y_\sqcup$ smaller than any y-value in A or B. For the third option the definition is analogous. Finally, $A \oplus B$ is when the line-segments A and B are extended and joined with one bend.

Definition 9. If two rectilinear tours cover a set $T = \{l_1, \ldots, l_t\}$ of line-segments in the same order (or exactly in reverse order) with the same number of bends, we say they are equivalent.

Fig. 4 illustrates the notion that two parallel line-segments l_i and l_j can be covered with two bends in two ways: $l_i \sqcap l_j$ and $l_i \sqcup l_j$ and the line-segment in between is free to shift by extending the tour's line-segments that cover l_i and l_j. Fig. 4a and Fig. 4b are in the same equivalence class since $l_i \sqcap l_j$ and $l_i \sqcup l_j$ alternate for the same line-segments and enable us to extend the tour by shifting the horizontal line-segments in the gap of $line_2$-$line_3$ and the gap of $line_4$-$line_5$ downwards. This may suggest this equivalence class as the "true" setting of the gadget. While Fig. 4c and Fig. 4d enable the extension downwards in the gap of $line_1$-$line_2$, the gap of $line_3$-$line_4$, and the gap of $line_5$-$line_6$ as the false setting. However, Fig. 4e shows a different equivalence class that respects the same extensions downwards. Moreover, Fig. 4f shows even one more equivalence class and this one mixes the possible extensions. Our gadgets will be placed in 3D because we can alternate the planes where parallel line-segments are placed and in this way, restrict the equivalence classes of rectilinear tours with minimum bends to two equivalence classes. We call the 3D gadgets, "true-false" gadgets.

4.1 A Building Block

First we illustrate the true-false gadgets for a variable u_i when $m = 1$ (see Fig. 5). We call the gadget when $m = 1$, a *building block* because we will repeat the structure of this gadget m times for a variable participating in m clauses.

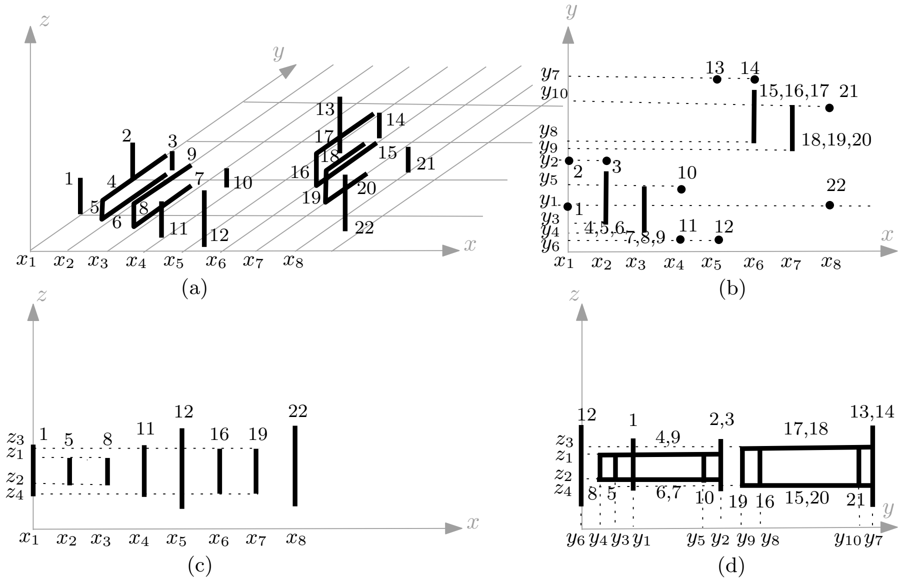


Figure 5: The structure of S_i (for $m = 1$) represents a variable u_i, (a) the building block in 3 dimensions, (b)-(d) the three different projections.

The building block of gadgets has the following properties.

1. All points lie within 8 different x-values $\{x_1, \ldots, x_8\}$, 10 different y-values $\{y_1, \ldots, y_{10}\}$ and 4 different z-values $\{z_1, \ldots, z_4\}$.

2. In each line-segment illustrated with a solid line in Fig. 5, we place 200 points equally spaced along the segment. There are many points in the line-segments so that a path with minimum number of bends will be forced to use these line-segments.

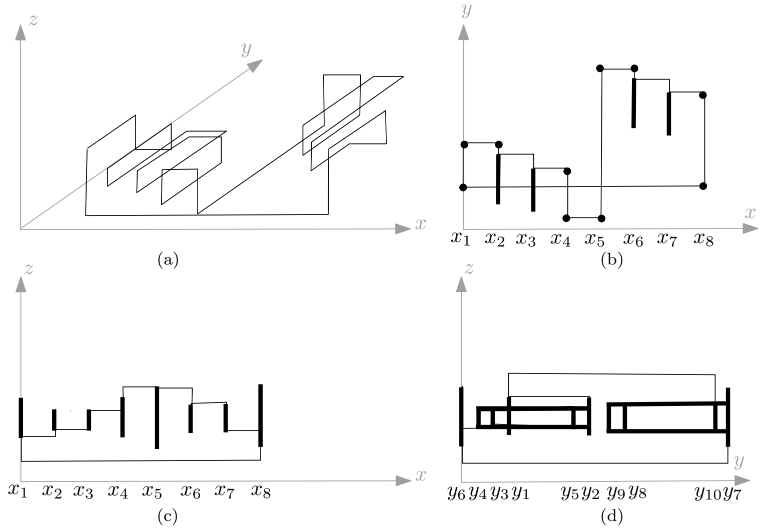


Figure 6: A tour for setting the gadget to true ($m = 1$) and its three projections.

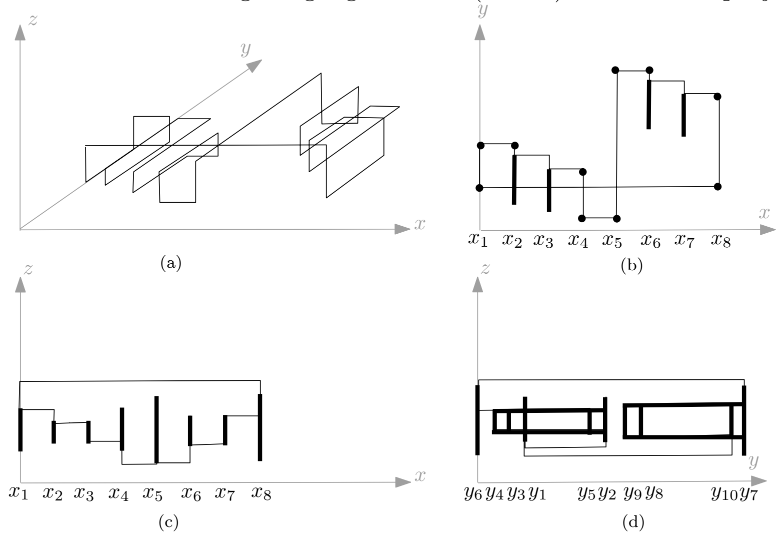


Figure 7: A tour for setting the gadget to false ($m = 1$) and its three projections.

3. The maximum number of line-segments of the same form in the same axis-parallel plane is two.

4. The set B_j of 4400 points can be covered by 14 line-segments of the form $line(x, y, *)$ and 8 line-segments of the form $line(x, *, z)$ where

$$B_j \subseteq \bigcup_{s=1}^{22} cover(\text{line-segment}_s).$$

Table 1 shows line-segment$_s$ for $s = 1, 2, \ldots, 22$. Note that in Fig. 5 these line-segments are labelled as $1, 2, \ldots, 22$.

Table 1: A building block B_j can be covered by 22 line-segments.

line-segment$_s$($s = 1, \ldots, 11$)	line-segment$_s$($s = 12, \ldots, 22$)
line-segment$_1$ $= line(x_1, y_1, *)$	line-segment$_{12}$ $= line(x_5, y_6, *)$
line-segment$_2$ $= line(x_1, y_2, *)$	line-segment$_{13}$ $= line(x_5, y_7, *)$
line-segment$_3$ $= line(x_2, y_2, *)$	line-segment$_{14}$ $= line(x_6, y_7, *)$
line-segment$_4$ $= line(x_2, *, z_1)$	line-segment$_{15}$ $= line(x_6, *, z_4)$
line-segment$_5$ $= line(x_2, y_{3,*})$	line-segment$_{16}$ $= line(x_6, y_8, *)$
line-segment$_6$ $= line(x_2, *, z_2)$	line-segment$_{17}$ $= line(x_6, *, z_3)$
line-segment$_7$ $= line(x_3, *, z_2)$	line-segment$_{18}$ $= line(x_7, *, z_3)$
line-segment$_8$ $= line(x_3, y_{4,*})$	line-segment$_{19}$ $= line(x_7, y_9, *)$
line-segment$_9$ $= line(x_3, *, z_1)$	line-segment$_{20}$ $= line(x_7, *, z_4)$
line-segment$_{10}$ $= line(x_4, y_5, *)$	line-segment$_{21}$ $= line(x_8, y_{10}, *)$
line-segment$_{11}$ $= line(x_4, y_6, *)$	line-segment$_{22}$ $= line(x_8, y_1, *)$

The building block of all gadgets for all variables will be placed so that

1. The lengths of line-segment$_1$ and line-segment$_2$ are both longer than the lengths of line-segment$_3$, line-segment$_5$, line-segment$_8$ and line-segment$_{10}$.

2. The y-coordinate of line-segment$_{10}$ in B_j is smaller than the y-coordinate of line-segment$_2$.

3. The sequence of line-segments starting with line-segment$_{12}$ til line-segment$_{22}$ follows the same pattern of the first 11 segments. However, we expand this part of the building block in the z-direction, equally on both sides, by 10% and we place them in increasing x and y values as shown in Fig. 5(c). The length of every line-segment being expanded is essentially 1.2 times longer than the length of the previous 11 line-segments. This requirement will prevent any possible interference between lines of different forms.

4. Any two line-segments of the form $line(x, y, *)$ in the same plane $\Pi(x, *, *)$ in B_j do not coincide in such a plane with those line-segments of this or any other gadget.

5. Any two line-segments of the form $line(x, *, z)$ in the same plane $\Pi(*, *, z)$ in B_j do not coincide in such a plane with those line-segments of this or any other gadget.

We will show that there are exactly two equivalence classes of covering the building block in Fig. 5 and both require at least 34 bends. To prove this claim, we require the following observations.

Observation 1 *All line-segments in the building block are in one of two forms, namely,* $line(x, y, *)$ *or* $line(x, *, z)$.

Observation 2 *In the building block, any two line-segments of the same form and in a common axis-parallel plane can be joined with two bends (and no fewer). That is,*

- $line(x_1, y_1, *)$ *and* $line(x_1, y_2, *)$ *with* $y_1 \neq y_2$ *or*
- $line(x_1, y_1, *)$ *and* $line(x_2, y_1, *)$ *with* $x_1 \neq x_2$ *or*
- $line(x_1, *, z_1)$ *and* $line(x_1, *, z_2)$ *with* $z_1 \neq z_2$ *or*
- $line(x_1, *, z_1)$ *and* $line(x_2, *, z_1)$ *with* $x_1 \neq x_2$ *or*

can be joined in a rectilinear path of two bends.

Observation 3 *In the building block, any two line-segments of mixed form on different axis-parallel planes such that there is an axis-parallel plane that contains one line-segment and intersects the other line-segment in an interior point, can be joined with three bends (and no fewer). That is,*

- $line(x_1, y_1, *)$ *and* $line(x_2, *, z_2)$ *where* z_2 *is chosen such that* (x_1, y_1, z_2) *is an interior point in* $line(x_1, y_1, *)$*, can be joined with three bends.*
- $line(x_1, y_1, *)$ *and* $line(x_2, *, z_2)$ *where* y_1 *is chosen such that* (x_2, y_1, z_2) *is an interior point in* $line(x_2, *, z_2)$*, can be joined with three bends.*
- $line(x_1, y_1, *)$ *and* $line(x_2, *, z_2)$ *where the values* z_2 *and* y_1 *are chosen such that* (x_1, y_1, z_2) *is an interior point in* $line(x_1, y_1, *)$*, and* (x_2, y_1, z_2) *is an interior point in* $line(x_2, *, z_2)$*, can be joined with four bends.*

Observation 4 *In the building block, any two line-segments of mixed form on different axis-parallel planes such that there is an axis-parallel plane that contains one line-segment and intersects the other line-segment in an exterior point, can be joined with two bends (an no fewer). That is,* $line(x_1, y_1, *)$ *and* $line(x_2, *, z_2)$ *where* z_2 *and* y_1 *are chosen such that* (x_1, y_1, z_2) *is an exterior point in* $line(x_1, y_1, *)$*, and* (x_2, y_1, z_2) *is an exterior point in* $line(x_2, *, z_2)$*, can be joined with two bends.*

Observation 5 *In the building block, any two line-segments of mixed form on the same axis-parallel plane where a line of one line-segment intersects the other line-segment in an interior point, can be joined with three bends (and no fewer). That is,*

- $line(x_1, y_1, *)$ *and* $line(x_1, *, z_2)$ *where* z_2 *is chosen such that* (x_1, y_1, z_2) *is an interior point in* $line(x_1, y_1, *)$*, or*
- $line(x_1, y_1, *)$ *and* $line(x_1, *, z_2)$ *where* y_1 *is chosen such that* (x_1, y_1, z_2) *is an interior point in* $line(x_1, *, z_2)$*, or*
- $line(x_1, y_1, *)$ *and* $line(x_1, *, z_2)$ *where* (x_1, y_1, z_2) *is a point of line-segment intersection*

can be joined in a rectilinear path of three bends (and no fewer).

Observation 6 *In the building block, any two line-segments of mixed form on the same axis-parallel plane where lines hosting these two segments intersects in an exterior point of the segments, can be joined with one bend. That is,* $line(x_1, y_1, *)$ *and* $line(x_1, *, z_2)$ *can be extended to meet at a point* (x_1, y_1, z_2) *that is exterior to* $line(x_1, y_1, *)$ *and* $line(x_1, *, z_2)$*.*

Lemma 10. *A building block as shown in Fig. 5 can be covered by a rectilinear tour with* 34 *bends and no fewer in only two equivalent classes as Fig. 6 and Fig. 7.*

Proof. First, we prove 34 bends are necessary. We know from the previously described six observations that any pair of line-segments in the building block can be joined with one bend, two bends, three bends, or four bends. One-bend is the best we can do and four-bends is the worst case of connecting any pairs of line-segments. We will show that an optimal tour uses only one-bend type of connections and two-bends type of connections but does not use three-bends and four-bends type of connections. If we could use one bend for every pair of line-segments, we would have an optimal tour with 22 bends because there are 22 line-segments in the building block. However, the best we can do is to use one-bend type of connections (Observation 6) at the 10 locations in the building block (see Fig. 8). Therefore, any optimal tour can have at most 10 bends of a connection-type described by Observation 6. The structure of the building block is then reduced to 12 components. The next best thing that we can do is to join any pair of these components with two bends. This would result in at least another 24 bends. Hence, an optimal tour covering the building block in Fig. 5 requires 34 bends or more. Note that if a tour uses three-bends and four-bends type of connections, we would definitely have more than 34 bends in the tour.

The fact that 34 bends are sufficient is shown by any of the tours in Fig. 6 or Fig. 7.

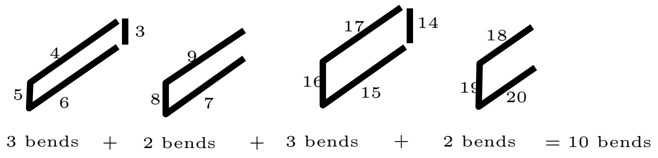


Figure 8: Ten bends that use a connection-type described by Observation 6.

We now show these are the only two possible equivalence classes. According to Observation 2 and Observation 4, we have two options when joining any pair of line-segments with two bends. That is, we have an option to join them at the top or to join them at the bottom. These two options give rise to exactly the two equivalent classes as per Fig. 6 and Fig. 7. Disobeying this pattern would require a three-bends connection or a four-bends connection and consequently will result in more than 34 bends in the tour. □

4.2 A Complete Gadget

The structure of the 22 line-segments in a building block is repeated. Our aim is to generate a complete gadget that holds 22 covering line-segments per clause or $22m$ covering line-segments overall (see Fig. 9). A complete gadget has $S_i = \bigcup_{j=1}^{m} B_j$ for $j = 1, 2, \ldots, m$. The set S_i can be covered by axis-parallel line-segments that have a cyclic structure as follows.

$$S_i = cover(\{line(x_1, y_1, *), line(x_1, y_2, *), line(x_2, y_2, *), line(x_2, *, z_1), line(x_2, y_{3,*}), line(x_2, *, z_2), line(x_3, *, z_2), line(x_3, y_{4,*}), line(x_3, *, z_1), line(x_4, y_5, *), line(x_4, y_6, *), line(x_5, y_6, *), \ldots, line(x_{8m}, y_{10m}, *), line(x_{8m}, y_1, *), line(x_1, y_1, *)\}).$$

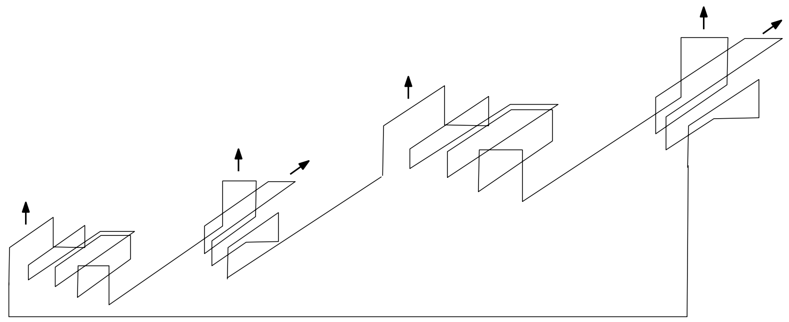


Figure 9: A complete gadget when a variable is set to true ($m = 2$).

The last line-segment of B_j and the first line-segment of $B_{(j \ mod \ m)+1}$ are always on the same plane $\Pi(*, y, *)$. For every 11 line-segments in the gadget, we still expand the shape of the gadget in z-direction, equally on both sides, by 10%. Therefore, a building block of the first clause is smaller than the block of the second clause, a building block of the second clause is smaller than the block of third clause and so on.

Finally, we put n' gadgets that represents n' variables on different x, y, and z-coordinates such that no two covering line-segments of different variables are on the same axis-parallel plane. We call two parallel line-segments of the form $line(x, *, z)$ and the line-segment $line(x, y, *)$ that connects them, a “C” shape. For example, line-segment$_4$, line-segment$_5$, and line-segment$_6$ in Fig. 5 form a “C” shape. We enforce the following relationship between the “C” shape and the line-segments of different building blocks of the form $line(x, y, *)$ in front of it: the z-coordinates of every line-segment in the “C” shape must be between the smallest and largest z-coordinates of the line-segment $line(x, y, *)$ that has larger y-coordinates than the “C” shape. To maintain this relationship across all gadgets, we make sure that the n' gadgets are different in size. That is, in a clause the gadget of a variable that participates as the first literal is slightly bigger than the one that participates as the second literal, and the gadget of a variable that participates as the second literal is slightly bigger than the one that participates as the third literal. Again, this requirement is to prevent any possible interference between lines of different gadgets.

The structures of a tour T_i for setting the gadget to "true" (see Fig. 6) and a tour F_i for setting the gadget to "false" (see Fig. 7) are described next.

$$\begin{aligned}
T_i = \{&line(x_1, y_1, *) \sqcap line(x_1, y_2, *) \sqcup line(x_2, y_2, *) \oplus line(x_2, *, z_1) \oplus \\
&line(x_2, y_{3,*}) \oplus line(x_2, *, z_2) \sqcap line(x_3, *, z_2) \oplus line(x_3, y_{4,*}) \oplus \\
&line(x_3, *, z_1) \sqcap line(x_4, y_5, *) \sqcup line(x_4, y_6, *) \sqcap line(x_5, y_6, *), \ldots, \\
&line(x_{8m}, y_{10m}, *) \sqcap line(x_{8m}, y_1, *) \sqcup line(x_1, y_1, *)\} \text{ and} \\
F_i = &line(x_1, y_1, *) \sqcup line(x_1, y_2, *) \sqcap line(x_2, y_2, *) \oplus line(x_2, *, z_2) \oplus \\
&line(x_2, y_{3,*}) \oplus line(x_2, *, z_1) \sqcap line(x_3, *, z_1) \oplus line(x_3, y_{4,*}) \oplus \\
&line(x_3, *, z_2) \sqcup line(x_4, y_5, *) \sqcap line(x_4, y_6, *) \sqcup line(x_5, y_6, *), \ldots, \\
&line(x_{8m}, y_{10m}, *) \sqcup line(x_{8m}, y_1, *) \sqcap line(x_1, y_1, *)\}.
\end{aligned}$$

If we assign "true" to the variable u_i, then we cover the set S_i by the tour of T_i. On the other hand, if we assign "false" to the variable u_i, then we cover the set S_i by the tour of F_i. In each of these two configurations, the rectilinear tour has $34m$ bends, which is the minimum number of bends any tour can do.

It is important to note that the arrows in Fig. 9 indicates the paths in true-gadgets that have the flexibility to be extended to cover points of clauses (discussed later in the next section). In false-gadgets, all arrows will be pointing in the opposite direction.

4.3 Gadget Assignment

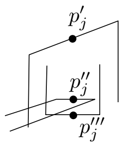
A complete gadget is built in such a way that it can be covered by a tour in two ways that give the minimum number of bends to cover the points in S_i. One of the ways represents the variable being assigned the value "true", and the other way represents the variable being assigned the value "false". The minimum number of bends for each complete gadget is set to $34m$ bends. That is, each gadget consists of $34m$ line-segments (i.e., $22m$ covering line-segments plus $12m$ joining line-segments that do not cover points in $\bigcup_{i=1}^{n'} S_i$).

Lemma 11. *A complete gadget can be covered by a rectilinear tour with $34m$ bends and no fewer in the "true" configuration or the "false" configuration, where m is the number of clauses.*

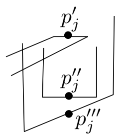
Proof. First we establish necessity. A complete gadget has $S_i = \bigcup_{j=1}^{m} B_j$ for $j = 1, 2, \ldots, m$. Since each building block is replicated in the same way, there are $10m$ locations where we can use one-bend type of connections. The structure of a complete gadget is then reduced to $12m$ components. If we could join any pair of these components with two bends, this would result in at least another $24m$ bends. Hence, an optimal tour covering a complete gadget requires $34m$ bends and no fewer.

To establish sufficiency consider Fig. 9. This shows how the path pattern of Fig. 6 extends for $m > 1$.

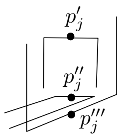
That these are the only two equivalence classes, is established as follows. According to Observation 2 and Observation 4, we have two options when joining two parallel line-segments A and B in the same axis-parallel plane with two bends. That is, we can join them as $A \sqcap B$ or $A \sqcup B$. These two options corresponds to the "true" configuration and the "false" configuration. A tour that deviates from these two configuration would require a three-bends or a four-bends connection somewhere and consequently will result in more than $34m$ bends in the tour. $\square$



(a) u_1 = "true", u_2 = "false", u_3 = "false"



(b) u_1 = "false", u_2 = "true", u_3 = "false"



(c) u_1 = "false", u_2 = "false", u_3 = "true"

Figure 10: Three cases where the clause $E_j = u_1 \vee u_2 \vee u_3$ is satisfiable and p'_j, p''_j and p'''_j is covered.

For every clause, if it is a YES-instance, then there are three patterns; true-false-false, false-true-false, and false-false-true. This is equivalent to choosing exactly one of the three variables that participate in a clause, setting it to true and placing the other two as false. Formally, we represent each clause E_j by three points $p'_j = (a_j, b_j, c'_j)$, $p''_j = (a_j, b_j, c''_j)$ and $p'''_j = (a_j, b_j, c'''_j)$ in a line $(a_j, b_j, *)$ where $c'_j > c''_j > c'''_j$ and $j = 1, 2, \ldots, m$. The coordinates of all these points will be pairwise distinct. The gadget passing through the point p'_j is the one set to true. The gadget passing through the points p''_j and p'''_j will be the gadget set to false. Essentially, the points p'_j, p''_j and p'''_j are covered by the extendible paths (see Fig. 9) of the three gadgets participating in a clause. The line-segments of these three gadgets are positioned with respect to the the line $(a_j, b_j, *)$ as follows.

1. The line $(a_j, b_j, *)$ is placed on the plane $\Pi(x_{8j-7}, *, *)$ between two line-segments $line(x_{8j-7}, y_{10j-9}, *)$ and $line(x_{8j-7}, y_{10j-8}, *)$ of a gadget that participates as the first literal. Here, we set $x_{8j-7} = a_j$, $y_{10j-9} < b_j < y_{10j-8}$, $c'_j \geq z_\sqcap$, and $c'''_j \leq z_\sqcup$.
2. The line $(a_j, b_j, *)$ is placed between the top plane $\Pi(*, *, z_{4j-1})$ and the bottom plane $\Pi(*, *, z_{4j})$. The plane $\Pi(*, *, z_{4j-1})$ has two line-segments: $line(x_{8j-2}, *, z_{4j-1})$ and $line(x_{8j-1}, *, z_{4j-1})$. The plane $\Pi(*, *, z_{4j})$ has two line-segments: $line(x_{8j-2}, *, z_{4j})$ and $line(x_{8j-1}, *, z_{4j})$. Here, we set $z_{4j-1} = c'_j$, $z_{4j} = c''_j$, , $x_{8j-2} < a_j < x_{8j-1}$ and $b_j \geq y_\sqcap$.
3. The line $(a_j, b_j, *)$ is placed on the plane $\Pi(*, y_{10j-3}, *)$ between two line-segments $line(x_{8j-3}, y_{10j-3}, *)$ and $line(x_{8j-2}, y_{10j-3}, *)$ of a gadget that participates as the third literal. Here, we set $y_{10j-3} = b_j$, $x_{8j-3} < a_j < x_{8j-2}$, $c'_j \geq z_\sqcap$, and $c''_j \leq z_\sqcup$.

If the clause E_j is satisfiable, then one path from true-gadget must cover p'_j and two paths from false-gadgets must cover p''_j and p'''_j (see Fig. 10).

Lemma 12. *To maintain the minimum number of bends, a complete gadget must be covered only in the "true" configuration or the "false" configuration but cannot be covered by both configurations.*

Proof. A tour that switches the setting of a gadget from "true" to "false" would require three-bends or four-bends type of connections. This contradicts the optimal tour described in Lemma 10 and Lemma 11 in which the tour uses only one-bend and two-bends type of connections. □

Lemma 13. *A tour, once it enters a complete gadget, maintains its equivalent classes (therefore, all have the same number of bends) as per Definition 9, and it only changes in extensions that cover points of clauses.*

Proof. Since all the gadgets are different in size and we place n' gadgets on different x, y, and z-coordinates such that no two covering line-segments of different gadgets are on the same axis-parallel plane, the same argument applies as per Lemma 11. If the tour leaves the gadget (not to cover points of clauses), then at least three additional bends are required and we would need more than $34m$ bends to cover the points in S_i. Note that a clause was mapped to three collinear points. If a path that covers a gadget passes through more than one of these three points, it would be incurring on a line $(a_j, b_j, *)$ that is additional to the axis-parallel line-segments it is already covering. By Observation 1 to Observation 6, this implies at least one more bend. □

4.4 A Complete Tour

We have discussed a tour that covers S_i and there are n' of these tours, each having $34m$ bends. We now explain how to obtain a single tour that covers $\bigcup_{i=1}^{n'} S_i$. Two tours of S_i can be merged into one tour with the minimum number of additional bends. Merging any two tours of S_i requires a removal of no more than one line-segment from each tour (4 bends fewer) but we have to add two line-segments for connecting any two line-segments from different axis-parallel planes (6 bends more). Thus, in total we require 2 additional bends for merging any two tours of S_i. We illustrate this merging in Fig. 11 where we place two tours of S_i far away from each other for easy viewing. Note that the line-segments that we removed above cannot be the covering line-segments of S_i and cannot be the line-segments that are set to cover points of clauses. Therefore, we set the following two options. We either remove a line-segment joining $line(x_{8m}, y_{10m}, *)$ with $line(x_{8m}, y_1, *)$ in the tour, or remove a line-segment joining $line(x_{8m}, y_1, *)$ with $line(x_1, y_1, *)$. We repeat the merging for $n'-1$ times to get a single tour covering $\bigcup_{i=1}^{n'} S_i$. The number of bends in a complete tour is therefore $34mn' + 2(n'-1) = 34mn' + 2n' - 2$. The number of links in a complete tour is also $34mn' + 2n' - 2$.

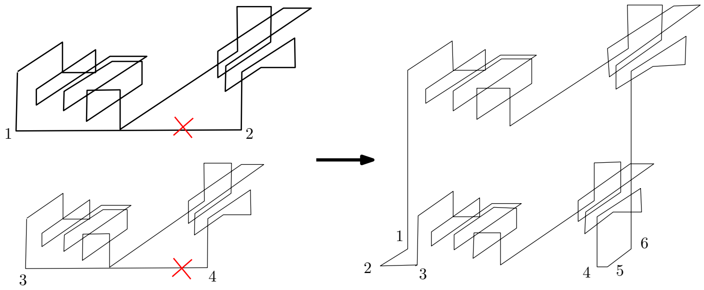


Figure 11: Merging any two tours of S_i requires 2 additional bends.

Lemma 14. *There exists a satisfying assignment for $E_1 \wedge \ldots \wedge E_m$ if and only if a set of $|S|$ points in 3D can be covered optimally by a rectilinear tour that has $34mn' + 2n' - 2$ links.*

Proof. If a clause is satisfiable, then the three points p'_j, p''_j and p'''_j that represent the clause are covered by three different line-segments. The point p'_j is covered by a line-segment arising from a true-gadget while the points p''_j and p'''_j are covered by two different line-segments arising from two different false-gadgets. Thus, there are no additional bends incurred to cover points in P. Then, the number of bends in the tour is the minimum number of bends to cover points in $\bigcup_{i=1}^{n'} S_i$, which is $34mn' + 2n' - 2$ bends or $34mn' + 2n' - 2$ links. On the other hand, if a clause is not satisfiable (i.e., a clause has one false literal or all equal literals), then at least one of the three points that represent the clause would not be covered in the tour and to include these points in the tour we would need more than $34mn' + 2n' - 2$ links. Conversely, if a tour with $34mn' + 2n' - 2$ links covers $\bigcup_{i=1}^{n'} S_i$ and it also covers P that represents the points of clauses $E_1, E_2 \dots, E_m$, then $E_1 \wedge \dots \wedge E_m$ is clearly satisfiable. $\square$

As a consequence of Lemma 14, we obtain the following result.

Theorem 15. *The decision version of* RECTILINEAR MINIMUM LINK TRAVELING SALESMAN PROBLEM *in 3 dimensions is NP-complete.*

Similarly, we prove the NP-completeness of RECTILINEAR MINIMUM LINK SPANNING PATH PROBLEM by a reduction from the ONE-IN-THREE 3-SAT. We follow the same line of argument as above except that we do not connect the first line-segment to the last line-segment in order to complete the tour.

Theorem 16. *The decision version of* RECTILINEAR MINIMUM LINK SPANNING PATH PROBLEM *in 3 dimensions is NP-complete.*

Since Theorem 16 is true in 3 dimensions, the problem is also NP-complete for $d > 3$ dimensions.

5 Rectilinear Spanning Path Problem

In this section, we prove that the problem associated with Theorem 16 belongs to the class FPT under the assumption that no two line-segments share the same line. By revisiting the NP-completeness proof above, we find that this restricted problem remains NP-complete. This is important because it means that even with this restriction the problem has no polynomial-time exact algorithm, unless P=NP. Now, we specify the problem in the format of parameterized complexity as follows. Given a set S of n points in $\mathbb{R}^d$, and a positive integer k, we are asked if there is a piecewise linear path through these n points in S having at most k line-segments (links) where every line-segment in the path is axis-parallel.

We follow the standard convention of the problem, that is to restrict the turns in the path to $90°$ turns. We decide a problem instance for the rectilinear spanning path with at most k links in two phases. First, we compute candidate sets of k lines that cover all the n points in S. We assume that one line-segment in the spanning path covers all the points on the same line[3], so once we have

[3] This is a constraint of the problem. In unrestricted case several line-segments could cover points on the same line, we leave this case as an open problem.

the candidate set of k covering lines, we also have the candidate set of k line-segments. Second, we test if the candidate set can constitute a spanning path based on these k line-segments in the set.

We can compute the candidate sets of k lines that cover all the n points in S using the algorithm that solves the Rectilinear Line Cover in d dimensions. Recall that this can be done in $O(d^k n)$ using the bounded-search-tree technique. However, the bounded-search-tree by itself is not to establish the solution, but at the leaves we check if the candidate lines from the tree can be completed into a spanning path with no more than k links. Each of the candidate set of lines in a leaf of the search tree in Fig. 1 results in a candidate set of line-segments since we can simply connect the extreme points in $cover(l)$ for each l in the k lines of the candidate set to get the candidate line-segments. Now we can explore exhaustively if they conform to the required solution of spanning path. In the worst case, the k line-segments can be organized into $k!$ orders in a possible spanning path. For any given order, there are at most 2^k possible ways of connecting these k line-segments as a path because we can connect the current segment with the next segment in the sequence by two possible end-points. This is necessary because the two segments could have different orientations, and we could save an extra line-segment if we connect the two segments at the correct end-point. This means a total of $(k!)(2^k)$ tests. In a simplest case, the extension of the two segments is enough to make a turn, therefore no extra line-segment is required. In the worst case, it requires at most $d+1$ additional line-segments in order to connect two consecutive line-segments. In the construction of the rectilinear spanning path, these additional line-segments can be added in $O(kd)$ time. Note that if the total number of line-segments in the final rectilinear spanning path exceeds k, we immediately answer no.

Theorem 17. *The* Rectilinear k-Links Spanning Path Problem *in d dimensions is fixed-parameter tractable (FPT) under the assumption that no two line-segments share the same line.*

Proof. The search tree for the first phase has at most $O(d^k)$ leaves and $O(d^{k-1})$ internal nodes. The work at each internal node can be performed in time linear in n. The dominant work is the computation at the leaves of the tree. Here we perform the tests whether we have the spanning path that covers all the points in S and has at most k links. This results in time bounded by

$$\begin{aligned} O(d^k(k!)(2^k)(kd+n)) &= O((2d)^k\sqrt{2\pi k}(k/e)^k(kd+n)) \\ &= O((0.74dk)^k(kd+n)\sqrt{k}). \end{aligned}$$

The time complexity is exponential in the parameter but polynomial in the size of the input. Thus, the problem is FPT. □

6 Spanning Path Restricted to a Finite Number of Orientations

We now consider the MINIMUM LINK SPANNING PATH PROBLEM where the set of k links is restricted to the lines having a finite number m of orientations. Note that the statements that follow hold without specifying the dimensions, but in this case, the orientations are vectors based at the origin. That is, in this case the k line-segments must have their orientations taken from a given finite set $R = \{r_1, r_2, \ldots, r_m\}$ where $|R| = m$. This also implies that the possible angles between turns at linking points belong to a finite set. We decide a problem in two phases. First, we compute candidate sets of k lines that cover all the n points in S. We assume that one line-segment in the spanning path covers all the points on the same line. Second, we test if the candidate set can constitute a spanning path based on these k lines that have a finite number of orientations defined in R.

The first phase is done using the algorithm in Section 2.1. Recall that this algorithm outputs a set of leaves, and for each leaf a set of k lines covering S that is restricted to the lines having orientations in R. The algorithm achieves this in $O(m^k n)$ time. The second phase is done in a way similar to that of the previous section. Each leaf is evaluated to decide if it can be made into a path. That is, an exhaustive search with the total of $(k!)(2^k)$ possible constructions of a path from the cover. In each test, $O(km)$ additional lines may be added in a construction of a path. The total time complexity is $O((0.74mk)^k(km+n)\sqrt{k})$. Thus we obtain an FPT-algorithm because its time complexity is polynomial in the size of the input, although it is exponential in the parameter.

Theorem 18. *The* RECTILINEAR k-LINKS SPANNING PATH PROBLEM *when restricted to the lines having a finite number $m \geq 2$ of orientations and no two line-segments share the same line, is FPT.*

7 Rectilinear Hyperplane Cover Problem

We prove that the problem of covering a set of $|S|$ points in 3D with k axis-parallel planes of 2D is NP-complete. The method of our proof is inspired by the NP-completeness proof of the RECTILINEAR LINE COVER problem in 3D [Hassin and Megiddo, 1991]. First, our problem is trivially in NP. The verification algorithm checks whether each point in the given set S is covered by at least one of the k planes and every plane is axis-parallel. This can be performed in polynomial time.

To prove NP-hardness we present a reduction from 3-SAT (3-satisfiability), a classical NP-complete problem. Our reduction is by gadget construction. Assume that we are given an instance of 3-SAT with a set of m clauses $E_j = w'_j \vee w''_j \vee w'''_j$ for $j = 1, 2, \ldots, m$ where $\{w'_j, w''_j, w'''_j\} \subset \{u_1, \bar{u}_1, \ldots, u_{n'}, \bar{u}_{n'}\}$. The

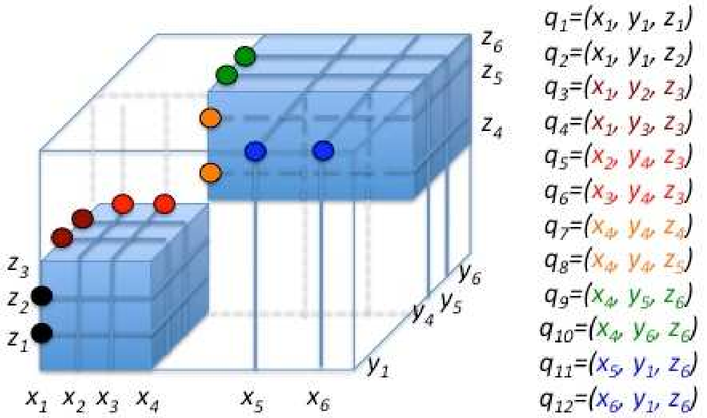
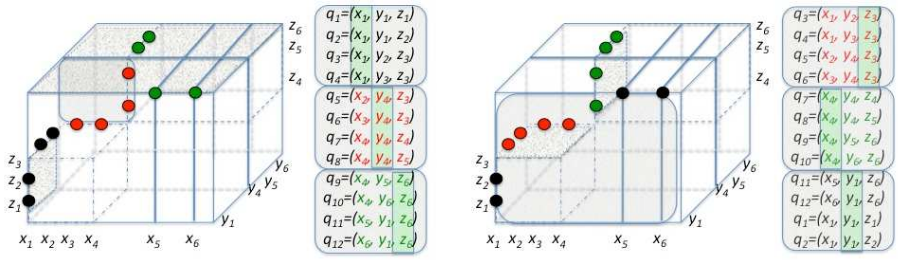


Figure 12: The structure of S_i that represents each variable u_i is guided by $2m$ cubes that are the building blocks of the gadget (illustration with $m = 1, \pi = 3$).

instance of 3-SAT has n' variables and we will represent each variable u_i for $i = 1, 2, \ldots, n'$ by a set S_i of $12m$ points. We represent each clause E_j with a point p_j. The corresponding instance of Rectilinear Hyperplane Cover will have a set $S = \{p_1, p_2, \ldots, p_m\} \cup \bigcup_{i=1}^{n'} S_i$ where $|S| = 12mn' + m = n$. The set S_i represents u_i by a gadget which will be covered optimally $\pi = 3m$ different axis-parallel planes. There are n' gadgets in total. Since the gadgets ensure that all occurrences of a variable across the formula have a consistent setting, we call these gadgets, "true-false" gadgets. First we construct the true-false gadgets for each variable u_i (see Fig. 12).

Each gadget is build from $2m$ axis-parallel cubes as reference. The cubes are aligned one after the other with the bottom-left-front corner of the next cube coinciding with the top-right-back corner of the previous one. In each of these $2m$ cubes there are 6 points (except on the last one, where there are four). Two points are in the edge that meets the front face and the left face. Two points are in the edge that meets the top face and the left face. And two points are in the edge that meets the top face and the back face. For the last cube 4 are like the first 4 of all other cubes; however, the last two are in the edge that meets the top face of this cube and the front face of the fist cube. Therefore, for each gadget, the set S_i has a cyclic structure. The x-coordinate has a fixed value x_1 for the first 4 points, then a different value x_2 in the 5-th point and another different value x_3 in the 6-th point. Then, this pattern of 4 equal, and 2 distinct is repeated, with x_4 in the next 4 points, a new value x_5 in the 11-th point and x_6 in the 12-th point. The pattern of 4 equal values and 2 different is also repeated in the y-coordinate and the z-coordinate, but in the z-th coordinate the 4 equal values start from the 3-rd to the 6-th point, and for the y-coordinate from the 5-th to the 8-th point (in all coordinates the pattern of 4 equal values, one different and



(a) Cover for “true”. (b) Cover for “false”.

Figure 13: The optimal covers for the illustration of $m = 1$ from Fig. 12.

one different wraps around modulo 12) Thus, S_i looks as follows.

$$\begin{aligned}S_i = \{&(x_1, y_1, z_1), (x_1, y_1, z_2), (x_1, y_2, z_3),\\ &(x_1, y_3, z_3), (x_2, y_4, z_3), (x_3, y_4, z_3),\\ &(x_4, y_4, z_4), \ldots, (x_{2\pi-2}, y_{2\pi-1}, z_{2\pi}),\\ &(x_{2\pi-2}, y_{2\pi}, z_{2\pi}), (x_{2\pi-1}, y_1, z_{i2\pi}),\\ &(x_{i2\pi}, y_1, z_{2\pi})\}.\end{aligned}$$

We denote the plane perpendicular to the x axis, that intersects x at x_0 and is parallel to the plane of the y and z axes by $\Pi(x_0, *, *)$. Similarly, we represent the plane perpendicular to the y axis, that intersects y at y_0 and is parallel to the plane of the x and z axes by $\Pi(*, y_0, *)$. The plane perpendicular to the z axis, that intersects z at z_0 and is parallel to the plane of the x and y axes is called $\Pi(*, *, z_0)$. Now, to cover the set S_i it requires π distinct axis-parallel planes. The gadget is built in such a way that there are exactly two sets of π planes that cover S_i (see Fig. 13). These are

$$\begin{aligned}T_i = \{&\Pi(x_1, *, *), \Pi(*, y_4, *), \Pi(*, *, z_6), \ldots,\\ &\Pi(*, y_{2\pi-2}, *), \Pi(*, *, z_{2\pi})\}\end{aligned}$$

and

$$\begin{aligned}F_i = \{&\Pi(*, *, z_3), \Pi(x_4, *, *), \Pi(*, y_7, *), \ldots,\\ &\Pi(x_{2\pi-2}, *, *), \Pi(*, y_1, *)\}.\end{aligned}$$

In each of these two configurations, each plane covers 4 points, which is the maximum any plane can cover. The two sets, T_i, F_i, each consists of π axis-parallel planes and each covers S_i, for $i = 1, 2, \ldots, n'$. If we assign “true” to the variable u_i, then we cover the set S_i by the planes of T_i. On the other hand, if we assign “false” to the variable u_i, then we cover the set S_i by the planes of F_i.

We represent each clause E_j by a single point $p_j = (a_j, b_j, c_j)$ for $j = 1, 2, \ldots, m$. The coordinates of all the points is S_i are all pairwise distinct from those of $S_{i'}$, for $i \neq i'$. The only constraints on the coordinates of the points in the set S_i are relative to the clauses where the variable represented in the gadget participates. The idea to set up the gadget to have planes that can stretch to the points of the clauses E_j where it participates. To enable at least one of the planes in the true-false gadgets to covers the point $p_j = (a_j, b_j, c_j)$ representing clause E_j where it participates we proceed as follows. If $u_i = w'_j$ (it participates as the first literal and does so positively), then in S_i we set $x_{6j-5} = a_j$. This means a setting to "true" will make the $2j-1$ cube have its left face as a plane with value a_j and cover p_j. But if $\bar{u}_i = w'_j$ (it is the first literal in the clause but participates negated), then in S_i we set $x_{6j-2} = a_j$. Now the plane is the right face of cube $2j-1$ (which is the left face of cube $2j$). If $u_i = w''_j$ (participates in the second literal), then in S_i we set $y_{6j-2} = b_j$. If $\bar{u}_i = w''_j$, then in S_i we set $y_{6j-5} = b_j$. If $u_i = w'''_j$ (the third literal), then in S_i we set $z_{6j} = c_j$. If $\bar{u}_i = w'''_j$, then in S_i we set $z_{6j-3} = c_j$ (see Fig. 14).

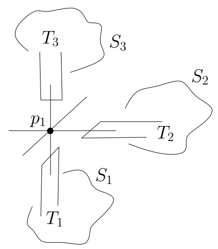


Figure 14: Example where the clause $E_1 = u_1 \vee u_2 \vee u_3$ is satisfiable and p_1 is covered.

Lemma 19. *There exists a satisfying assignment for $E_1 \wedge \ldots \wedge E_m$ if and only if a set of $|S|$ points in 3D can be covered by $k = \pi n'$ axis-parallel 2D-planes.*

Proof. Suppose $E_1 \wedge \ldots \wedge E_m$ is satisfiable. By assigning planes to n' gadgets as per a satisfying assignment, we can cover $\bigcup_{i=1}^{n'} S_i$ with $3mn'$ axis-parallel planes. Moreover, these planes can cover the points in $\{p_1, p_2, \ldots, p_m\}$. That is, there are no additional planes needed for covering p_j. Therefore, if $E_1 \wedge \ldots \wedge E_m$ is satisfiable, then the set $\{p_1, p_2, \ldots, p_m\} \cup \bigcup_{i=1}^{n'} S_i$ can be covered by $\pi n'$ axis-parallel planes.

Conversely, if the $\pi n'$ axis-parallel planes cover $\bigcup_{i=1}^{n'} S_i$ and these planes also cover $p_1, p_2, \ldots, p_m$, then, first of all they need to cover the gadgets as consistent assignments for each gadget. But then, as the points p_j represents the clause $E_1, E_2 \ldots, E_m$ respectively, then $E_1 \wedge \ldots \wedge E_m$ is clearly satisfiable. $\square$

As a consequence of Lemma 19, we obtain the following result.

Theorem 20. *The decision version of covering a set of points in 3D with axis-parallel planes of 2D is NP-complete.*

Since Theorem 20 is true in 3 dimensions, the problem is also NP-complete for $d > 3$ dimensions.

7.1 Using a Bounded-Search-Tree

Given a set S of n points in $\mathbb{R}^d$ and a positive integer k, we are asked if it is possible to cover these n points in d dimensions with k axis-parallel hyperplanes of $d-1$ dimensions. This problem is FPT when parameterized by the number of dimensions, d, and the size of the solution, k. Using the bounded-search-tree technique, the proof goes as follows. Consider the problem in 3D first. For each point p in 3-dimensional space there are three possibilities of being covered, namely, 1) a plane that is orthogonal to height, 2) a plane that is orthogonal to depth, and 3) a plane that is orthogonal to width. Besides the point p, other points on the same plane with p are also marked as covered in the recursive call. The search tree has a branching factor of three and if we use k as the depth, that is, we assigned no more than k different axis-parallel planes, then the tree has 3^k leaves. If all leaves do not make a cover, then the answer is NO. Otherwise, we have a cover and the instance is YES.

In d dimensions, each hyperplane is orthogonal to one of the dimensions, so the branching factor becomes d while the depth is still k. The running time of the algorithm is then $O(d^k n)$.

Theorem 21. *The problem of covering a set of n points in $d \geq 3$ dimensions with no more than k axis-parallel hyperplanes of $d-1$ dimensions can be solved in $O(d^k n)$ time.*

Note that the algorithms of Langerman and Morin [2005] can be adapted to solve the Hyperplane Cover in both the general and the rectilinear case but their complexity remains the same. Their bounded-search-tree algorithm that runs in $O(k^{dk} n)$ time is clearly FPT with respect to k and d. However, the kernelization algorithm that runs in $O(k^{d(k+1)} + n^{d+1})$ time is FPT for the parameter k but not for the parameter d. When the problem is specialized to the rectilinear case, our FPT-algorithm above offers a better complexity.

We now extend our result by considering the problem of covering a set of points in d dimensions with k hyperplanes of $d-1$ dimension where a set of these k hyperplanes is restricted to the hyperplanes having a finite number of orientations, ϕ. In other words, these k hyperplanes must have their orientations taken from a given finite set $\mathcal{H}^{d-1} = \{\Pi_1, \Pi_2, \ldots, \Pi_\phi\}$. We can construct the search tree with ϕ branches similar to the early discussion in Section 7.1 and the depth is still k. Thus, we obtain another FPT-algorithm since the time complexity of our algorithm is polynomial in the size of the input n, although it is exponential in the parameter k.

Theorem 22. *The problem of covering a set of n points in $d \geq 3$ dimensions with no more than k hyperplanes of $d-1$ dimensions such that these hyperplanes have a finite number $\phi \geq 2$ of orientations, can be solved in $O(\phi^k n)$ time.*

8 Conclusions

We provided 6 FPT results and 3 NP-completeness results related to the problem of covering points in rectilinear domain in higher dimensions and in the settings where the covering objects are restricted to have a finite number of orientations. Although we proved that the Rectilinear Minimum Link Traveling Salesman Problem and Rectilinear Minimum Link Spanning Path Problem in 3 dimensions are NP-complete, the hardness of both problems in the planar case (2 dimensions) remains an open problem. In addition, we suspect that more problems in connection to the Traveling Salesman Problem such as finding a tour that minimizes the distance as well as the number of links in the plane or in higher dimensions belong to FPT. Also left open is removing the condition that no two line-segments share the same line in Theorem 17.

Acknowledgments

We thank the anonymous reviewers for valuable remarks that improved the original version of this paper.

References

[Arkin, E., Bender, M., Demaine, E., Fekete, S., Mitchell, J. and Sethia, S. 2005], 'Optimal covering tours with turn costs', *SIAM J. of Computing* **35**, 531–566.

[Arkin, E., Mitchell, J. and Piatko, C. 2003], 'Minimum-link watchman tours', *Information Processing Letters* **86(4)**, 203–207.

[Bereg, S., Bose, P., Dumitrescu, A., Hurtado, F. and Valtr, P. 2009], 'Traversing a set of points with a minimum number of turns', *Discrete and Computational Geometry* **41**, 513–532.

[Collins, M. 2004], 'Covering a set of points with a minimum number of turns', *International J. Computational Geometry and Applications* **14(1-2)**, 105–114.

[de Berg, M., van Kreveld, M., Nilsson, B. and Overmars, M. 1992], 'Shortest path queries in rectilinear worlds', *International Journal Computational Geometry Applications* **2(3)**, 287–309.

[Downey, R. and Fellows, M. 1999], *Parameterized Complexity*, Monographs in Computer Science, Springer, New York.

[Estivill-Castro, V., Heednacram, A. and Suraweera, F. 2009], 'Reduction rules deliver efficient FPT-algorithms for covering points with lines', *ACM Journal of Experimental Algorithmics* **14**, 1.7–1.26.

[Fink, E. and Wood, D. 1996], 'Fundamentals of restricted-orientation convexity', *Information Sciences* **92(1-4)**, 175–196.

[Fink, E. and Wood, D. 2004], *Restricted-Orientation Convexity*, Monographs in Theoretical Computer Science. An EATCS Series, Springer, New York.

[Flum, J. and Grohe, M. 2006], *Parameterized Complexity Theory*, Texts in Theoretical Computer Science, Springer, Berlin.

[Garey, M. and Johnson, G. 1979], *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, USA.

[Grantson, M. and Levcopoulos, C. 2006], Covering a set of points with a minimum number of lines, *in* 'Algorithms and Complexity, 6th Italian Conference, CIAC', Vol. 3998 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 6–17.

[Güting, H. 1983], 'Stabbing c-oriented polygons', *Information Processing Letters* **16**, 35–40.

[Hassin, R. and Megiddo, N. 1991], 'Approximation algorithms for hitting objects with straight lines', *Discrete Applied Mathematics* **30(1)**, 29–42.

[Hurtado, F. 2008]. Private communication.

[Kumar, V., Arya, S. and Ramesh, H. 2000], Hardness of set cover with intersection 1, *in* '27th International Colloquium on Automata, Languages and Programming, ICALP', Vol. 1853 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 624–635.

[Langerman, S. and Morin, P. 2005], 'Cover things with things', *Discrete and Computational Geometry* **33(4)**, 717–729.

[Lee, D., Chen, T. and Yang, C. 1990], Shortest rectilinear paths among weighted obstacles, *in* '6th ACM Symposium on Computational Geometry (SCG 90)', ACM, NY, pp. 301–310.

[Lee, D., Yang, C. and Wong, C. 1996], 'Rectilinear paths among rectilinear obstacles', *Discrete Applied Mathematics* **70(3)**, 185–215.

[Megiddo, N. and Tamir, A. 1982], 'On the complexity of locating linear facilities in the plane', *Operations Research Letters* **1(5)**, 194–197.

[Niedermeier, R. 2006], *Invitation to Fixed-Parameter Algorithms*, Oxford Lecture Series in Mathematics and its Applications 31, Oxford University Press, New York.

[Rawlins, G. and Wood, D. 1988], Computational geometry with restricted orientations, *in* '13th IFIP Conference on System Modelling and Optimization', Vol. 113 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 375–384.

[Rawlins, G. and Wood, D. 1991], 'Restricted-oriented convex sets', *Information Sciences* **54**, 263–281.

[Wagner, D. 2006], Path Planning Algorithms under the Link-Distance Metric, PhD thesis, Dartmouth College, Hanover, NH, USA.