The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Improved Results for MINIMUM CONSTRAINT REMOVAL

Eduard Eiben
Algorithms and Complexity Group,
TU Wien, Vienna, Austria &
Dept. of Informatics, Univ. of Bergen, Norway
eduard.eiben@uib.no

Jonathan Gemmell, Iyad Kanj, Andrew Youngdahl
School of Computing,
DePaul University, Chicago, USA,
{jgemmell,ikanj,ayoungda}@cdm.depaul.edu

Abstract

Given a set of obstacles and two designated points in the plane, the MINIMUM CONSTRAINT REMOVAL problem asks for a minimum number of obstacles that can be removed so that a collision-free path exists between the two designated points. It is a well-studied problem in both robotic motion planning and wireless computing that has been shown to be NP-hard in various settings.

In this work, we extend the study of MINIMUM CONSTRAINT REMOVAL. We start by presenting refined NP-hardness reductions for the two cases: (1) when all the obstacles are axes-parallel rectangles, and (2) when all the obstacles are line segments such that no three intersect at the same point. These results improve on existing results in the literature. As a byproduct of our NP-hardness reductions, we prove that, unless the Exponential-Time Hypothesis (ETH) fails, MINIMUM CONSTRAINT REMOVAL cannot be solved in subexponential time $2^{o(n)}$, where n is the number of obstacles in the instance. This shows that significant improvement on the brute-force $2^{\mathcal{O}(n)}$-time algorithm is unlikely.

We then present a subexponential-time algorithm for instances of MINIMUM CONSTRAINT REMOVAL in which the number of obstacles that overlap at any point is constant; the algorithm runs in time $2^{\mathcal{O}(\sqrt{N})}$, where N is the number of the vertices in the auxiliary graph associated with the instance of the problem. We show that significant improvement on this algorithm is unlikely by showing that, unless ETH fails, MINIMUM CONSTRAINT REMOVAL with bounded overlap number cannot be solved in time $2^{o(\sqrt{N})}$. We describe several exact algorithms and approximation algorithms that leverage heuristics and discuss their performance in an extensive empirical simulation.

Introduction

A fundamental problem in robot motion planning is to move a robot from a starting position to a final position while avoiding collision with a given set of obstacles. This problem is generally referred to as the *piano-mover*'s problem. If a collision-free path does not exist, one naturally seeks a path that collides with the minimum number of obstacles. In this paper, we study a variant of the piano mover's problem, referred to as the MINIMUM CONSTRAINT REMOVAL problem, defined as follows:

MINIMUM CONSTRAINT REMOVAL
Given: A set I of polygonal obstacles in the plane and two designated points s and t.
Goal: Compute a subset of obstacles $S \subseteq I$ of minimum cardinality such that there is an obstacle-free path in the plane between s and t w.r.t. the obstacles in $I \setminus S$.

In addition to its applications in robotics, the problem has been studied extensively, motivated by applications in wireless computing, under the name BARRIER COVERAGE or BARRIER RESILIENCE. In such applications, we are given a field covered by sensors (assumed to be simple geometric shapes), and the goal is to compute a minimum set of sensors that need to fail so that an entity can move undetected between two given sites (Alt et al. 2011; Tseng and Kirkpatrick 2012; Chan and Kirkpatrick 2014; Kumar, Lai, and Arora 2005; Yang 2012).

The MINIMUM CONSTRAINT REMOVAL problem was also formulated as a graph problem (Chan and Kirkpatrick 2014; Hauser 2014). For an instance I of the problem, the auxiliary graph of I, G_I, is defined as follows. Consider the plane subdivision whose regions are determined by the intersections of the obstacles in I. For each region, associate a vertex in G_I representing the set of obstacles intersecting at that region if any, and an empty set otherwise; add an edge between two vertices iff the corresponding regions share an edge. See Figure 1. Clearly, G_I is a plane graph since it is the dual graph of a plane subdivision. The problem then reduces to computing a path in G_I between the vertices corresponding to s and t, such that the total number of obstacles represented by the vertices on this path is minimum.

MINIMUM CONSTRAINT REMOVAL was studied by many researchers in Wireless Computing, AI, and Computational Geometry (Alt et al. 2011; Tseng and Kirkpatrick 2012; Chan and Kirkpatrick 2014; Kumar, Lai, and Arora 2005; Yang 2012; Erickson and LaValle 2013; Hauser 2014). Alt *et al.* (Alt et al. 2011) showed that the problem is NP-hard when the obstacles are line segments such that no three intersect at the same point. Independently, Yang (Yang 2012), in his Ph.D. dissertation, showed the NP-hardness of the problem when the obstacles are line segments. This result was refined independently by Tseng and Kirkpatrick (Tseng and Kirkpatrick 2012) who showed that the problem is NP-hard even when the obstacles are line segments of unit length. The more general graph problem

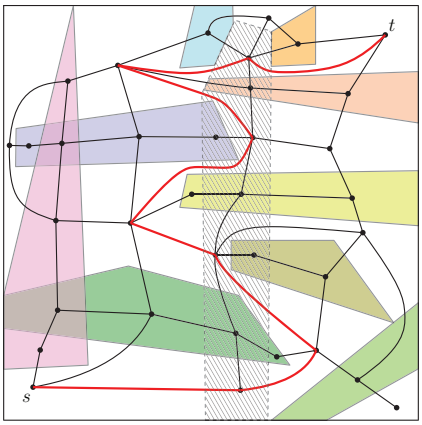

Figure 1: Illustration of the regions determined by a set of obstacles (placed within a bounding box) and its auxiliary graph, with a highlighted optimal path crossing one obstacle.

was considered by several researchers, including (Chan and Kirkpatrick 2014; Hauser 2014), and it is well known to be NP-hard (for instance, see (Hauser 2014) for a proof). Hauser (Hauser 2014) implemented and tested several algorithms for MINIMUM CONSTRAINT REMOVAL.

In this paper, we continue the study of the MINIMUM CONSTRAINT REMOVAL problem. We first consider the complexity of the problem, and show the following:

(1) MINIMUM CONSTRAINT REMOVAL is NP-hard even if all the obstacles are axes-parallel rectangles.

(2) MINIMUM CONSTRAINT REMOVAL is NP-hard even if all the obstacles are line segments such that no three intersect at the same point.

The results in (1) and (2) refine and improve the earlier work on the problem. More specifically, the result in (1) answers an open question posed in (Erickson and LaValle 2013). Even though the result in (2) was obtained earlier by Alt *et al.* (Alt et al. 2011), the NP-hardness reduction we use to prove (2) is more refined than the reduction used in (Alt et al. 2011). In particular, the reduction we use implies the ETH results in (3) and (5) below, and those cannot follow from the reduction in (Alt et al. 2011); this is because our reduction results in a linear number of obstacles, as opposed to the quadratic number of obstacles resulting from their reduction. As a byproduct of the reductions used to derive the above NP-hardness results, we obtain the following:

(3) Unless the Exponential-Time Hypothesis (ETH) fails, MINIMUM CONSTRAINT REMOVAL cannot be solved in subexponential time $2^{o(n)}$, where n is the number of obstacles in the instance.

The result in (3) shows that significant improvement on the $2^{\mathcal{O}(n)}$-time brute-force algorithm is unlikely, as ETH is a standard hypothesis for proving lower bounds (Aghighi et al. 2016; de Haan, Kanj, and Szeider 2015; Lokshtanov, Marx, and Saurabh 2011), which states that the satisfiability of k-CNF formulas (for $k \geq 3$) is not solvable in subexponential-time $2^{o(n)}$, where n is the number of variables in the formula.

We then design algorithms for the NP-hard restriction of CONSTRAINT REMOVAL to instances in which no more than a constant number b of obstacles overlap at the same point, denoted b-OVERLAP MINIMUM CONSTRAINT REMOVAL, for any integer-constant $b \geq 2$. We show that:

(4) There is a subexponential-time algorithm for b-OVERLAP MINIMUM CONSTRAINT REMOVAL that runs in time $2^{\mathcal{O}(\sqrt{N})}$, where N is the number of the vertices in the auxiliary graph associated with the instance; and

(5) unless ETH fails, b-OVERLAP MINIMUM CONSTRAINT REMOVAL cannot be solved in time $2^{o(\sqrt{N})}$.

The result in (4) gives a subexponential-time algorithm for b-OVERLAP MINIMUM CONSTRAINT REMOVAL w.r.t. the number of obstacles n, for instances in which the number of regions N, equivalently vertices in G_I, is $o(n^2)$.

Finally, we implement several exact and approximation algorithms and provide extensive empirical simulation results to compare their performance for varying sizes of n. Heuristics are proposed to improve performance.

Hardness Results

In this section, we consider the decision version of MINIMUM CONSTRAINT REMOVAL, denoted CONSTRAINT REMOVAL, in which we are given a set I of obstacles, two points s and t, and $k \in \mathbb{N}$, and we need to decide if there is an s-t path that intersects at most k obstacles in I. First, we start by showing that CONSTRAINT REMOVAL remains NP-hard even when the obstacles are axes-parallel rectangles; this answers an open question in (Erickson and LaValle 2013). Second, we show how our reduction can be modified to yield a reduction in which the obstacles are line segments such that no three of them intersect at the same point; this corrects the claim in (Erickson and LaValle 2013) stating that MINIMUM CONSTRAINT REMOVAL becomes trivial when the obstacles intersect only pairwise. As we mentioned in the previous section, the NP-hardness result for the line segments case was obtained earlier (Alt et al. 2011); however, the NP-hardness reduction we use is more refined.

To obtain the hardness results, we reduce from an NP-hard restriction of the MAXIMUM NEGATIVE 2-SATISFIABILITY problem. An instance of MAXIMUM NEGATIVE 2-SATISFIABILITY is given as a pair (F, m'), where $m' \in \mathbb{N}$ and F is a Boolean formula on n variables and m clauses. The question is to decide whether F has a satisfying assignment that satisfies at least m' clauses. The NP-hard restriction of MAXIMUM NEGATIVE 2-SATISFIABILITY we use, denoted R-MN2Sat, satisfies the following properties: (1) Each clause in the formula F is either a unit clause containing a positive literal $\{x_i\}$, or a binary clause containing two negative literals; (2) the unit clauses in F are precisely the clauses $\{x_i\}$, for each variable x_i in F; and (3) the number of both positive and negative occurrences of each variable is at most 4. A consequence of (3) is that the number of clauses m in F is at most $n + 3n/2 \leq 3n$. It can be easily shown that R-MN2Sat is NP-hard via a straightforward reduction from INDEPENDENT SET on graphs of maximum degree at most 3. It is well known, and follows from (Johnson and Szegedy 1999), that unless ETH fails, INDEPEN-

DENT SET on graphs of maximum degree at most 3 is not solvable in time $2^{o(n)}$, where n is the number of vertices in the graph. It follows from the reduction from INDEPENDENT SET to R-MN2Sat that, unless ETH fails, R-MN2SAT is not solvable in time $2^{o(n)}$. This result will be employed to derive lower-bound results on the subexponential-time complexity of MINIMUM CONSTRAINT REMOVAL. We start with this simple observation:

Observation 1. *Any truth assignment to an instance F of R-MN2Sat that satisfies m' clauses can be transformed into an assignment that satisfies at least m' clauses in F including all binary clauses.*

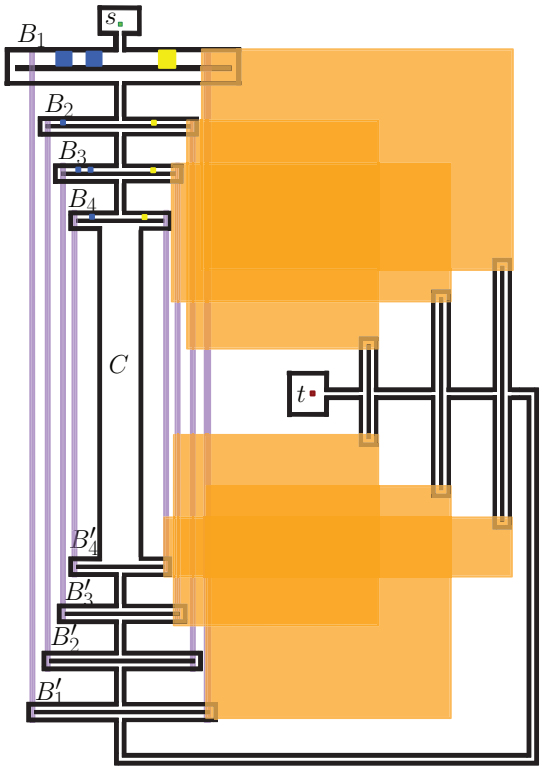

Figure 2: Illustration for the proof of Theorem 2 for $F = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge (\bar{x}_1 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee \bar{x}_1)$.

Axes-parallel Rectangles

Let RECTANGLE-CONSTRAINT REMOVAL be the restriction of CONSTRAINT REMOVAL to instances in which each obstacle is an axes-parallel rectangle.

Theorem 2. RECTANGLE-CONSTRAINT REMOVAL *is* NP*-hard.*

Proof. We give a polynomial-time reduction from R-MN2Sat to the decision version of RECTANGLE-CONSTRAINT REMOVAL. We highlight the main differences between the current reduction and that in (Erickson and LaValle 2013). See Figure 2 for illustration.

Let (F, m') be an instance of R-MN2Sat, where F has n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. Let $m_1 = n$ be the number of unit clauses in F, and $m_2 = m - n$ be the number of remaining (binary) clauses. To construct the instance (I, k) of RECTANGLE-CONSTRAINT REMOVAL, as in (Erickson and LaValle 2013), we start by enforcing a schema that any valid path must be confined within. We first outline a rectilinear region $\mathcal{R}$, whose boundary is shown in black in Figure 2. This region consists of an open rectangular region containing the starting point s (green); a rectilinear corridor $\mathcal{C}$ that runs vertically and then makes three left turns; a set of rectangular-like "boxes" intersecting $\mathcal{C}$ in their middle, each enclosing a black rectangle; and an open rectangular region containing the destination point t (burgundy).

For each variable x_i, we associate two boxes, B_i and B'_i, of the same shape and size, laid out horizontally and symmetrically on opposite ends of the vertical part of corridor $\mathcal{C}$ (before $\mathcal{C}$ turns). We call the B_i's and B'_i's *variable-boxes*. The B_i's appear on the top and run in decreasing horizontal dimension, and the B'_i's appear on the bottom and run in increasing horizontal dimension. The order in which the B_i's appear is the opposite order of the B'_i's. Each B_i (resp. B'_i) contains a black rectangle inside, subdividing it into an upper and a lower part, and creating a left passage and a right passage from its upper to its lower part. For each binary clause C_j, we associate a box B_{C_j} that we call a *clause-box*; the clause-boxes are placed towards the end of $\mathcal{C}$, just before the destination point t, and are laid out vertically.

To place the obstacles of I in $\mathcal{R}$, we start by fixing two integer-constants: $c_1 > 4$ and $c_2 > c_1 \cdot n + 3m$. To simplify the presentation of the proof, we will assign (in the proof) integer weights to certain obstacles to indicate an overlaying of the weight-many distinct identical-shape obstacles. Intuitively, the obstacles will be placed so that the portion of the path traversing the variable-boxes corresponds to an assignment of the variables in F, where traversing the left (resp. right) side of box B_i corresponds to assigning x_i to TRUE (resp. FALSE); the other portion of the path, traversing the clause-boxes, can be done in such a way that if a literal has been assigned TRUE by the first portion of the path, the clauses containing the literal can be traversed at no additional cost. To confine the path to the interior of region $\mathcal{R}$, we form the boundary of $\mathcal{R}$ by placing axes-parallel rectangular obstacles (shown in the same black color as the boundary), each of weight c_2, along this boundary so that they only (pairwise) overlap (in small squares) to form the corners of $\mathcal{R}$; this ensures that the cost of crossing the boundary of $\mathcal{R}$ exceeds the required budget k, which will be specified shortly. Similarly, each black rectangle outlined inside a box in $\mathcal{R}$, whose role is to block the direct passage of a path from one box to the next without setting the truth assignment of the variable associated with the first box, is formed using an axes-parallel rectangular obstacle of weight c_2; this way, the desired path cannot intersect any of these internal rectangles. We refer to the obstacles of weight c_2 as *heavy* obstacles.

For each binary clause C_j in F, we arbitrarily order the two (negative) literals in C_j as first and second. For each literal $\overline{x_i}$ and clause C_j such that $\overline{x_i}$ is the first (resp. second) literal in C_j, we create an axes-parallel rectangular obstacle of weight 1 (orange) that intersects the right side of box B_i (resp. B'_i) including the internal rectangle, *without*

intersecting any other variable-boxes, and intersects the top (resp. bottom) of the clause-box corresponding to C_j (including the internal rectangle) *without* intersecting any other clause-boxes (see Figure 2). These obstacles ensure that a path that sets a literal $\overline{x_i}$ to TRUE can traverse C_j at no additional cost. For each positive clause $\{x_i\}$, we place an axes-parallel rectangular obstacle of weight 1 (yellow) in the right side of B_i so that any path setting x_i to false intersects this obstacle. We call all these (orange plus yellow) obstacles *incidency* obstacles.

For each variable x_i, we add two axes-parallel rectangular obstacles (purple), each of weight c_1. The first obstacle intersects the left sides of B_i and B'_i *without* intersecting any other variable-boxes, and the second intersects the right sides of these boxes; these obstacles, referred to as *consistency obstacles*, are used to ensure that we do not set both a variable and its negation to TRUE.

Finally, for each x_i, let p_i be the number of occurrences of $\overline{x_i}$ in F; we place p_i many weight-1 axes-parallel rectangular obstacles (blue), referred to as *balancing* obstacles, in the left side of B_i. Let $k = c_1 \cdot n + 2m_2 + m_1 - (m' - m_2)$, and note that $k < c_2$. This completes the construction of the instance (I, k) of RECTANGLE-CONSTRAINT REMOVAL. We claim that (F, m') is a yes-instance iff (I, k) is. We first draw the following observations. (1) By the choice of c_2, any path that intersects at most k obstacles cannot intersect a heavy obstacle, and hence, must stay within $\mathcal{R}$. (2) Any path corresponding to a consistent assignment intersects exactly n consistency obstacles.

Suppose that F has a truth assignment τ that satisfies at least m' clauses. By Observation 1, we can assume that τ satisfies the m_2 binary clauses in F and at least $m' - m_2$ unit clauses. Consider the s-t path P in $\mathcal{R}$ that travels through the right side of boxes B_i and B'_i, for each variable x_i assigned FALSE by τ, and through the left side of these boxes otherwise; and for each satisfied binary clause, P travels through the part of the clause-box intersecting the incidency obstacle corresponding to a satisfied literal in the clause. We claim that P intersects at most k obstacles. To see this, note that by observation (2), this path intersects exactly n consistency obstacles, for a total cost of $c_1 \cdot n$. Second, for each B_i, regardless of whether P traverses the left or right passage in B_i, P crosses the number of occurrences of $\overline{x_i}$ many balancing obstacles, plus one incidency obstacle if P traverses the right side of B_i, and hence τ does not satisfy the unit clause $\{x_i\}$. It follows that the total number of obstacles that P intersects in its variable truth-setting portion is equal to the total number of occurrences of negative literals in F, which is $2m_2$, plus the total number of unit clauses that are dissatisfied by τ, which is at most $m_1 - (m' - m_2)$. Since τ satisfies all binary clauses, no additional cost is incurred in the remaining portion of P. It follows that P intersects a total of at most $c_1 \cdot n + 2m_2 + m_1 - (m' - m_2) = k$ obstacles.

Conversely, let P be an s-t path that intersects at most k obstacles. By observation (1), P lies inside $\mathcal{R}$. Since P has to go through each of the n variable-boxes, for each $i \in [n]$, if P traverses the left (resp. right) side of B_i, then we can assume, without loss of generality, that P traverses the left (resp. right) side of B'_i. This assumption can be justified as follows. If P traverses the left side (resp. right side) of B_i but then traverses the right side (resp. left side) of B'_i, then it would incur an additional cost of $c_1 > 4$. Since the ultimate gain/saving from such a switch is the number of occurrences of x_i in F, which is at most $4 < c_1$, rerouting P so that it traverses the same side of B'_i as B_i decreases the cost, and results in an alternative path that is cheaper than P. We can also assume that P incurs no additional cost in its second portion that traverses the clause-boxes. This assumption can be justified as follows. Suppose that P intersects a new obstacle while traversing the clause-box of a clause C_j. Pick a literal $\overline{x_i} \in C_j$. Then P must have traversed the left passage of B_i. If we reroute P so that it traverses the right passages of B_i and B'_i instead, then it is easy to see that the cost of the truth-setting portion of P due to this rerouting can increase by at most 1 due to the incidency obstacle corresponding to $\{x_i\}$ in the right part of B_i that the path has now to intersect. However, this additional cost is annulled since the new path now traverses C_j at no cost.

With the above assumptions in mind, consider now the truth assignment τ that assigns x_i to TRUE iff P traverses the left side of B_i. Then τ is consistent. The path P intersects: $c_1 \cdot n$ many consistency obstacles, a total of $2m_2$ many balancing obstacles, an incidency obstacle for each box B_i that P traverses its right side, and intersects no new obstacles in its portion that traverses the clause-boxes. Since P intersects at most k obstacles, it follows that the number of variable-boxes that P traverses their right side is at most $k - c_1 \cdot n - 2m_2 = m_1 - (m' - m_2)$. Therefore, τ dissatisfies at most $m_1 - (m' - m_2)$ unit clauses, and hence satisfies at least $m' - m_2$ unit clauses. Since τ satisfies all m_2 binary clauses, τ satisfies a total of at least m' clauses. □

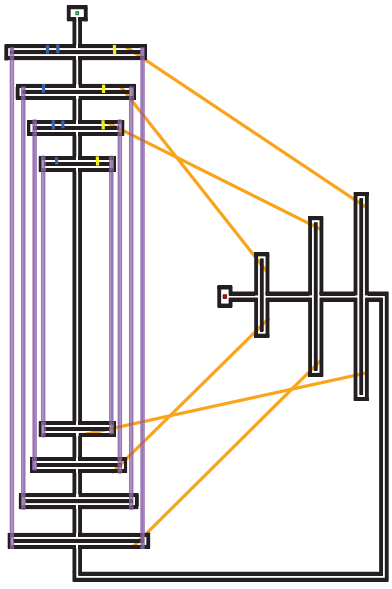

Figure 3: Illustration for the proof of Theorem 3 for $F = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge (\bar{x}_1 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee \bar{x}_1)$.

Straight-line Segments

For an instance I of CONSTRAINT REMOVAL, define the *overlap number* of I to be the maximum number of obstacles whose intersection is nonempty. Let LINE-CONSTRAINT REMOVAL be the restriction of CONSTRAINT REMOVAL to instances in which each obstacle is a line segment.

Theorem 3. LINE-CONSTRAINT REMOVAL*, restricted to instances whose overlap number is at most 2, is* NP*-hard.*

Proof. As in the proof of Theorem 2, we reduce from R-MN2Sat. The reduction is very similar to that in Theorem 2, except for the shapes and layout of the obstacles, as we no longer can overlay obstacles since we need to keep the overlap number at most 2.

Let (F, m') be an instance of R-MN2Sat, where F has n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. We refer to Figure 3 for illustration. We create the same region $\mathcal{R}$ as in the proof of Theorem 2, whose boundary is outlined in black in Figure 3. We use line-segments to mimic the rectangles used in the proof of Theorem 2. We describe how this is done for each obstacle-type used in that proof.

Let c_1, c_2 be the constants defined in the proof of Theorem 2. To form the boundary of $\mathcal{R}$ in the instance I of LINE-CONSTRAINT REMOVAL, we mimic the heavy obstacles forming the boundary of $\mathcal{R}$ with c_2 many nested boundaries, each formed using a sequence of distinct line-segment obstacles. We start by overlaying a distinct line-segment obstacle along each edge of the boundary of $\mathcal{R}$. We then create c_2-many nested copies of $\mathcal{R}$, each using a distinct set of obstacles, so that the path remains confined within the innermost copy of $\mathcal{R}$, as going out of the nested regions would incur a cost of c_2. See Figure 4 for illustration of the nesting around a box B_i. The inner rectangle subdividing a box B_i is created by nesting $c_1 + 5$ many rectangles, each created using 4 distinct line segments forming its sides; any path going directly through this rectangle and avoiding both the left and right passages of B_i, would incur a cost higher than going through either of the two passages, and hence, could be replaced by a path that sets a truth value for x_i.

To mimic the consistency obstacles, we replace each of the c_1 overlayed rectangles used in the formation of the consistency obstacle for x_i, with a bundle of c_1-many distinct vertical-segment obstacles (of the same length) intersecting only boxes B_i and B_i' (see Figures 3 and 4).

Note that some points in $\mathcal{R}$ are the intersection points of an obstacle forming the (nested) boundary of $\mathcal{R}$ with an obstacle that is part of a consistency obstacle. Such points have overlap number 2, but no point in $\mathcal{R}$ has overlap number larger than 2. We refer to any point in $\mathcal{R}$ that has overlap number 2 as an *intersection point* of $\mathcal{R}$.

To mimic the rectangular balancing obstacles in B_i, each balancing obstacle is replaced with a vertical line-segment obstacle, located on the same side of B_i, and blocking the passage of any path going through that side. To ensure that the overlapping number stays at most two, we place each of the balancing line-segment obstacles in such a way that none of its endpoints is an intersection point in $\mathcal{R}$. It is easy to see that this can always be done with the proper placement of these line-segment obstacles.

Finally, to place the incidency obstacles, for each literal $\overline{x_i}$ and clause C_j such that $\overline{x_i}$ is the first (resp. second) literal in clause C_j, we create a line-segment obstacle that intersects the right part of box B_i (resp. B_i') at non-intersection points in $\mathcal{R}$, and the top (resp. bottom) part of the clause-box corresponding to C_j at non-intersection points in $\mathcal{R}$. For each positive clause $\{x_i\}$, we place a vertical line-segment obstacle that blocks the right side of B_i such that its endpoints are non-intersection points in $\mathcal{R}$. All these obstacles are placed in such a way that no three line-segment obstacles intersect at the same point. Again, it is not difficult to see that this can be done with a proper layout of the boundary or $\mathcal{R}$, and the placement of these obstacles.

The constructed instance I has an overlap number at most 2. We define $k = c_1 \cdot n + 2m_2 + m_1 - (m' - m_2)$. Similar arguments to the ones made in the proof of Theorem 2 show that (F, m') is a yes-instance of R-MN2Sat iff (I, k) is a yes-instance of LINE-CONSTRAINT REMOVAL. □

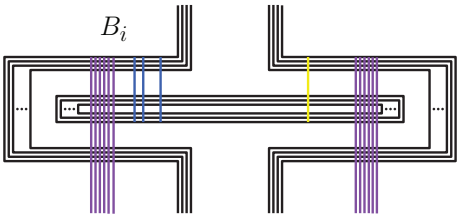

Figure 4: Illustration of the nesting used to confine the path.

Corollary 4. *Unless ETH fails,* LINE-CONSTRAINT REMOVAL *restricted to instances whose overlap number is at most 2 cannot be solved in time $2^{o(\sqrt{N})}$, where N is the number of regions in the input instance.*

Proof. Consider the NP-hardness reduction in the proof of Theorem 3. This reduction maps an instance of R-MN2SAT to an instance of LINE-CONSTRAINT REMOVAL of overlap number at most 2. It is not difficult to verify that the number of regions in the instance I of LINE-CONSTRAINT REMOVAL produced is quadratic in the number of variables n of the R-MN2Sat formula. It follows that an algorithm for LINE-CONSTRAINT REMOVAL that runs in time $2^{o(\sqrt{N})}$ would give an algorithm for R-MN2Sat that runs in time $2^{o(n)}$, and this would imply that ETH fails. □

Corollary 5. *Unless ETH fails,* CONSTRAINT REMOVAL *restricted to instances whose overlap number is at most 2 cannot be solved in time $2^{o(n)}$, where n is the number of obstacles in the input instance.*

Proof. Consider the NP-hardness reduction in the proof of Theorem 3, but instead of using a linear number of line-segment obstacles to form each layer of the boundary of $\mathcal{R}$, form the layer using a single rectilinear obstacle that is the union of all these segments. It is easy to verify that the modified reduction results in $O(n)$ obstacles, where n is the number of variables in the instance of R-MN2SAT. It follows that

an algorithm for CONSTRAINT REMOVAL restricted to instances whose overlap number is at most 2 that runs in time $2^{o(n)}$ would give an algorithm for R-MN2Sat that runs in time $2^{o(n)}$, and this would imply that ETH fails. □

Subexponential-time Algorithm

For an integer $b \geq 2$, define b-OVERLAP MINIMUM CONSTRAINT REMOVAL to be the restriction of MINIMUM CONSTRAINT REMOVAL to instances whose overlap number is at most b. (For $b \geq 2$, b-OVERLAP MINIMUM CONSTRAINT REMOVAL is NP-hard by Theorem 3.) In this section, we present an algorithm that solves an instance I of b-OVERLAP MINIMUM CONSTRAINT REMOVAL in time $2^{\mathcal{O}(\sqrt{N})}$, where N is the number of vertices in the auxiliary graph G_I. Recall that the number of vertices N in G_I is the number of regions determined by the intersections of the n obstacles in I, and hence $N = \mathcal{O}(n^2)$. Whereas this algorithm does not improve on the brute-force algorithm when the number of regions N is quadratic in the number of obstacles n, it does give a subexponential-time algorithm in terms of n when the number of regions is $o(n^2)$, which improves on the brute-force $2^{O(n)}$-time algorithm for the problem. By Corollary 4, it is unlikely that b-OVERLAP MINIMUM CONSTRAINT REMOVAL can be solved in time $2^{o(\sqrt{N})}$.

The algorithm we present is a divide-and-conquer algorithm, based on a variant of the well-known balanced separator theorem for planar graphs (Lipton and Tarjan 1979). This variant theorem (Miller 1986) states that the vertex-set of a triangulated plane graph on N vertices can be partitioned into three parts A, B, S such that: (1) S is a cycle separating A from B and $|S| \leq \sqrt{8N}$; (2) $|A| \leq 2N/3$ and $|B| \leq 2N/3$; and (3) A is interior to S and B is exterior to S (w.r.t. the plane embedding). The reason why we use this variant theorem, as opposed to the celebrated planar separator theorem (Lipton and Tarjan 1979), is that for the problem under consideration, this variant theorem allows for a more efficient enumeration of the separator, as will be discussed later. Our algorithm follows the approach in Woeginger *et al.* (Deineko, Klinz, and Woeginger 2006), for computing a Hamiltonian path in a planar graph on N vertices in time $2^{\mathcal{O}(\sqrt{N})}$. There are complications, however, that are particular to b-OVERLAP MINIMUM CONSTRAINT REMOVAL. We describe below how to deal with these complications.

Consider an instance I of b-OVERLAP MINIMUM CONSTRAINT REMOVAL on n obstacles, and let G_I be its auxiliary graph. We assign each obstacle in I a distinct representative color and assume that each vertex v in G_I is colored by the color-set representing the obstacles forming the region of v. As in (Deineko, Klinz, and Woeginger 2006), we add edges to G_I so that the resulting graph is a triangulation, and then apply the cycle separator theorem (Miller 1986) to partition the vertex-set of the resulting graph into A, B, S; the added edges are removed afterwards, and play no role other than determining A, B, S.

As in (Deineko, Klinz, and Woeginger 2006), the algorithm maintains a configuration, which is a tuple, and an auxiliary graph stipulating a partial ordering that the current enumeration dictates on the path vertices. We skip these details since they are very similar to those in (Deineko, Klinz, and Woeginger 2006), and highlight those that are particular to b-OVERLAP MINIMUM CONSTRAINT REMOVAL. After computing A, B, S, we enumerate every subset of S, as the subset of vertices that are contained in the path, P, we seek. For each enumerated subset F, we enumerate every subset of colors C that appear both on vertices in $S \setminus F$ and on P. We then remove all colors in S from G_I, and mark every vertex containing a color that is in $S \setminus F$ but not in C as "forbidden". Afterwards, the color-set appearing on vertices in A is disjoint from that appearing on vertices in B, because the colors that appear in both A and B must appear in S (the vertices on which the same color appears induce a connected subgraph of G_I), and those colors have been removed. The number of enumerations so far is at most $2^{\mathcal{O}(\sqrt{8N})} \cdot 2^{\mathcal{O}(b \cdot \sqrt{8N})} = 2^{\mathcal{O}(\sqrt{N})}$.

Fix such an enumeration. Next, we need to enumerate the order in which P traverses the vertices in F. Enumerating all permutations of the vertices in F will not result in a $2^{\mathcal{O}(\sqrt{N})}$-time algorithm. Instead, we adopt a similar enumeration method to the one in Woeginger *et al.* (Deineko, Klinz, and Woeginger 2006), which is based on the following observation. Suppose for now that the order in which P visits the vertices in F has been revealed. For any two nonadjacent vertices u, v in $F \cap V(P)$, say that u and v are *A-consecutive* (resp. *B-consecutive*) on P if the subpath of P between u and v, excluding u and v, is contained in A (resp. in B). Let $E_A \subseteq F \times F$ (resp. $E_B \subseteq F \times F$) be the set of edges between A-consecutive (resp. B-consecutive) vertices (these edges are not in G_I). The algorithm makes two recursive calls, one on $G_I[A \cup S] + E_B$ after modifying the auxiliary structure so that to enforce the order imposed by E_A and E_B, and the other on $G_I[B \cup S] + E_A$ after modifying the auxiliary structure so that to enforce the order imposed by E_A and E_B. The algorithm returns an s-t path that is the concatenation of a path having the minimum number of colors resulting from the recursive call on $G_I[A \cup S] + E_B$, with a path having the minimum number of colors resulting from the recursive call on $G_I[B \cup S] + E_A$. The recursion stops when the instance size reaches a suitable small number.

Now, to enumerate E_A and E_B efficiently without enumerating all permutations of F, we observe the following. Since each edge in E_A corresponds to a path in A interior to the cycle S, the edges in E_A are embeddable inside S, and hence E_A is a subset of edges of the edge-set of a triangulation of S. It is well known that the number of triangulations of S is $2^{\mathcal{O}(|S|)} = 2^{\mathcal{O}(\sqrt{N})}$ (*e.g.*, see (Michaels and Rosen 2007)[Theorem 6, Chapter 7]), and these triangulations can be easily enumerated in time $2^{\mathcal{O}(\sqrt{N})}$. Since each triangulation contains $\mathcal{O}(|S|)$ many edges, the number of subsets of edges in any triangulations is also $2^{\mathcal{O}(\sqrt{N})}$. Therefore, the total number of subsets of edges of triangulations of S is $2^{\mathcal{O}(|S|)} \cdot 2^{\mathcal{O}(\sqrt{N})} = 2^{\mathcal{O}(\sqrt{N})}$. It follows that the number of subsets E_A that need to be enumerated is $2^{\mathcal{O}(\sqrt{N})}$, and they can be enumerated in time $2^{\mathcal{O}(\sqrt{N})}$. Similarly, the number of subsets E_B that need to be enumerated is $2^{\mathcal{O}(\sqrt{N})}$, and they can be enumerated in time $2^{\mathcal{O}(\sqrt{N})}$. After enumerating

E_A and E_B, we can enumerate the first and last vertices in F that P visits. Knowing these vertices and E_A and E_B is sufficient to know the order in which P traverses F. Clearly, the number of enumerations is still $2^{\mathcal{O}(\sqrt{N})}$.

In conclusion, the number of enumerations needed to divide the instance of size N into two, each of size at most $2N/3 + \mathcal{O}(\sqrt{N})$, is $2^{\mathcal{O}(\sqrt{N})}$. This gives a recurrence relation $T(N) = 2^{\mathcal{O}(\sqrt{N})}T(2N/3)$, where $T(N)$ is the running time of the algorithm (plus an additive polynomial term), whose solution is $T(N) = 2^{\mathcal{O}(\sqrt{N})}$.

Theorem 6. *b-Overlap Minimum Constraint Removal can be solved in time $2^{O(\sqrt{N})}$, and unless ETH fails, it cannot be solved in time $2^{o(\sqrt{N})}$, even when the obstacles are line segments.*

Proof. The upper bound follows from the algorithm above. The lower bound follows from Corollary 4. □

Experimental Evaluation

In this section, we experimentally evaluate the performance of Hauser's algorithm (Hauser 2014), several exponential time algorithms and greedy approaches.

Implementations

Hauser. Hauser's algorithm explores a state space in which each state is represented as a vertex, v, and the cover of some path P leading from the start, s, to v; the cover is defined as the set of obstacles on P, which if removed, yield an obstacle-free path from s to v. A state corresponding to a vertex v is pruned if its cover is a super-set of the cover of another state corresponding to v. The search ends when no new state can be created. This search is guaranteed to be optimal and Hauser argued that the pruning step eliminates a large number of states making the algorithm more practical.

Divide-and-Conquer Algorithm. We implemented the divide-and-conquer algorithm discussed in the previous section combined with some heuristics to speed-up the search. Although the separators we computed were small and near-balanced, the algorithm itself was computationally infeasible, even for instances with 20 obstacles. Therefore, at this point, this algorithm remains mainly of theoretical interest, and further research on using heuristics to optimize its running time is recommended.

Iterative Deepening Search. Iterative deepening search (IDS) is a classic search strategy. In the context of the Minimum Constraint Removal problem, depth-first search is performed on the auxiliary graph associated with the input instance, with a limit to how many obstacles are removed. We start this limit at 0, and increase it until a solution is found. IDS is guaranteed to discover an optimal solution, but may be impractical. Selecting which obstacles to remove based on some heuristic is likely to improve the time. In addition to implementing the naive IDS, we implemented two variations of IDS with heuristics:

- **IDS with Minimum Cover Heuristic.** At a state in the search corresponding to a vertex v in G_I, the search proceeds to the neighbor of v whose cover has minimum cardinality until it reaches its limit and then backtracks. The search remains optimal, but in practice is quicker.
- **IDS with Euclidean Distance Heuristic.** In this search, the vertices nearest to t are given priority: At a state in the search corresponding to a vertex v in G_I, the search proceeds to the neighbor of v whose Euclidean distance to t is minimum until it reaches its limit and then backtracks.

Greedy Search. Greedy, or best-first, search makes a locally optimal choice as it navigates through the auxiliary graph. Unlike IDS, greedy search explores a single path and does not guarantee an optimal solution. We rely on the same heuristics as in the IDS implementations:

- **Greedy Search with Minimum Cover Heuristic.** At a state in the search corresponding to a vertex v in G_I, the search proceeds to the neighbor of v whose cover has minimum cardinality.
- **Greedy Search with Euclidean Distance Heuristic.** At a state in the search corresponding to a vertex v in G_I, the search proceeds to the neighbor of v whose Euclidean distance to t is minimum.

Empirical Results

The above algorithms were evaluated on three types of input instances: (1) instances in which the obstacles are polygons; (2) instances in which the obstacles are axes-parallel rectangles; and (3) instances in which the obstacles are line segments. Shapes were generated by selecting random points from a uniform distribution. The results for polygons and rectangles are presented in Figure 5; the results for line segments are similar and omitted for lack of space.

For all simulations where n is greater than 50, Hauser's algorithm and IDS quickly become infeasible, requiring several hours to produce a solution. Heuristic-informed IDS, however, remain practical. For instances constructed with 100 random polygons, IDS with the Euclidean distance heuristic (IDS-E) takes on average 31.4 seconds to discover a solution. IDS with the minimum cover heuristic (IDS-MC) requires 28.5 seconds. The greedy algorithms (Greedy-E and Greedy-MC) are much faster (4.4 and 3.2 seconds respectively). The same pattern is observed in simulations with segments and rectangles: Greedy approaches are much more efficient than optimal algorithms with heuristics, which, in turn, are more efficient than naive implementations.

To test the limitations of the algorithms, simulations with up to 1000 rectangles were evaluated. The results indicate that simple heuristics can greatly increase the size of instances that optimal and greedy algorithms can tackle. For instance, IDS-E was able to find a solution in 684 seconds, while Greedy-E discovered a solution in 277 seconds.

While the greedy algorithms with heuristics ran faster than the iterative deepening searches, their solutions are not guaranteed to be optimal. To evaluate the quality of their solutions, we computed the approximation ratio—the ratio between the discovered solution and an optimal solution. In all simulations of different shapes and sizes, the approximation ratio was 1 or close to it. This finding suggests that the solution space for random imputations of the problem is dense and easily navigated by simple heuristics. However, one can construct examples that confound these greedy algorithms,

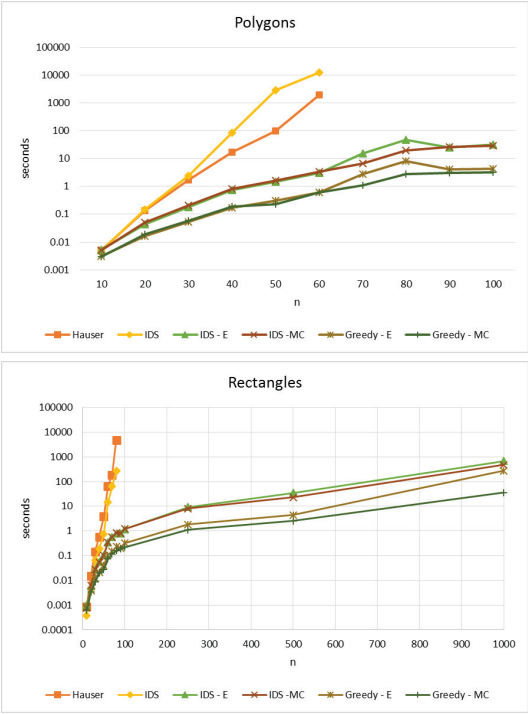

Figure 5: Average time for the six algorithms on 20 simulations of n random polygons and rectangles.

and from a theoretical perspective, it can be shown that their ratios approach the total number of obstacles.

Conclusions

Our study shows that, from the theoretical perspective, MINIMUM CONSTRAINT REMOVAL is a very intractable problem, as it remains computationally hard even when the obstacles have very simple geometric shapes. However, from a practical perspective, the problem seems to be amenable to standard heuristics, and can be solved efficiently and accurately for large instances.

The major theoretical question left open, and posed in (Erickson and LaValle 2013), is whether MINIMUM CONSTRAINT REMOVAL becomes polynomial-time solvable if we restrict it to instances in which no obstacle intersects more than a constant number of other obstacles. Certainly, this is a natural restriction on the problem that seems to be challenging, and deserves further investigation. From the practical side, it would be interesting to see implementations of the divide-and-conquer algorithm, that lead to an exact algorithm for the problem with a better performance than the exact algorithms implemented in the current work.

Acknowledgments The authors are thankful to the reviewers. Eiben was supported by Pareto-Optimal Parameterized Algorithms (ERC Starting Grant 715744) and by the Austrian Science Fund (FWF, projects P26696 and W1255-N23). Gemmell, Kanj, and Yougdahl were supported by DePaul University Academic Initiatives Pool grant #601347.

References

Aghighi, M.; Bäckström, C.; Jonsson, P.; and Ståhlberg, S. 2016. Refining complexity analyses in planning by exploiting the Exponential Time Hypothesis. *Annals of Mathematics and Artificial Intelligence* 78(2):157–175.

Alt, H.; Cabello, S.; Giannopoulos, P.; and Knauer, C. 2011. On some connection problems in straight-line segment arrangements. In *Proceedings of EuroCG*, 27–30.

Chan, D., and Kirkpatrick, D. 2014. Multi-path algorithms for minimum-colour path problems with applications to approximating barrier resilience. *Theoretical Computer Science* 553:74–90.

de Haan, R.; Kanj, I.; and Szeider, S. 2015. On the subexponential-time complexity of CSP. *Journal of Artificial Intelligence Research* 52:203–234.

Deineko, V.; Klinz, B.; and Woeginger, G. 2006. Exact algorithms for the Hamiltonian cycle problem in planar graphs. *Operations Research Letters* 34(3):269–274.

Erickson, L., and LaValle, S. 2013. A simple, but NP-hard, motion planning problem. In *Proceedings of AAAI*. AAAI Press.

Hauser, K. 2014. The minimum constraint removal problem with three robotics applications. *International Journal of Robotics Research* 33(1):5–17.

Johnson, D., and Szegedy, M. 1999. What are the least tractable instances of Max independent set? In *Proceedings of SODA*, 927–928. ACM/SIAM.

Kumar, S.; Lai, T.; and Arora, A. 2005. Barrier coverage with wireless sensors. In *Proceedings of MOBICOM*, 284–298. ACM.

Lipton, R., and Tarjan, R. 1979. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics* 36(2):177–189.

Lokshtanov, D.; Marx, D.; and Saurabh, S. 2011. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS* 105:41–72.

Michaels, J., and Rosen, K. 2007. *Applications of Discrete Mathematics*. New York: McGraw Hill.

Miller, G. 1986. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences* 32(3):265–279.

Tseng, K., and Kirkpatrick, D. 2012. On barrier resilience of sensor networks. In *Proceedings of ALGOSENSORS*, 130–144.

Yang, S. 2012. *Some Path Planning Algorithms in Computational Geometry and Air Traffic Management*. Ph.D. Dissertation, University of New York at Stony Brook.