Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

A Simple, but NP-Hard, Motion Planning Problem

Lawrence H. Erickson and Steven M. LaValle
University of Illinois at Urbana-Champaign
Department of Computer Science
201 N. Goodwin Ave.
Urbana, IL 61801

Abstract

Determining the existence of a collision-free path between two points is one of the most fundamental questions in robotics. However, in situations where crossing an obstacle is costly but not impossible, it may be more appropriate to ask for the path that crosses the fewest obstacles. This may arise in both autonomous outdoor navigation (where the obstacles are rough but not completely impassable terrain) or indoor navigation (where the obstacles are doors that can be opened if necessary). This problem, the *minimum constraint removal problem*, is at least as hard as the underlying path existence problem. In this paper, we demonstrate that the minimum constraint removal problem is NP-hard for navigation in the plane even when the obstacles are all convex polygons, a case where the path existence problem is very easy.

1 Introduction

In most robotic path planning problems, the goal is to travel from a starting state to a goal state without colliding with an obstacle along the way. However, there are many situations where collision with an obstacle may be inconvenient, but not insurmountable. The obstacles could be closed doors that the robot is capable of opening. In outdoor navigation, the obstacles could be patches of rough terrain that the robot is capable of crossing, but with a risk of damage. Alternately, it might be desirable to know the smallest set of obstacles that block a path between two points (perhaps a road is being constructed, and the builders want to do it while demolishing as few pre-existing features of the environment as possible).

With that in mind, a natural question arises: If no collision-free path from the starting state to the goal state exists, what is the fewest number of obstacles intersected by any path from the start to the goal? This problem, introduced by Hauser, is called the *minimum constraint removal problem*, as it asks for the minimum number of constraints (obstacles) that would need to be removed from the environment to produce a collision-free path (Hauser 2012). A sample problem is shown in Figure 1.

Hauser demonstrated that the discrete version of this problem (in which the environment is represented by a graph, with obstacles being subsets of the vertices) is NP-hard, via a reduction from the minimum set cover problem. This result is interesting, as the corresponding path existence problem is very easy to solve in graphs with a breadth-first search.

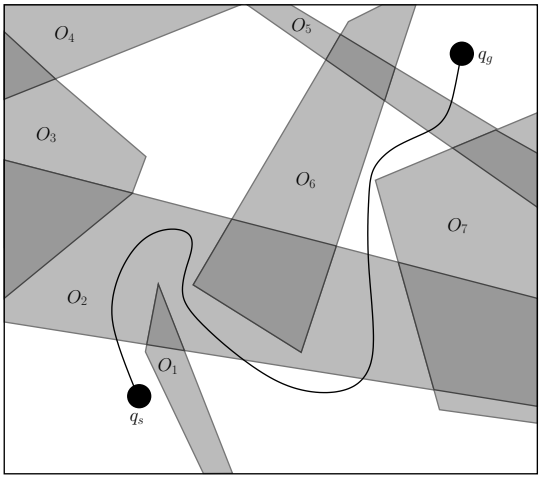

Figure 1: An instance of the minimum constraint removal problem. A path from the start (q_s) to the goal (q_g) that intersects the minimum number of obstacles (O_2 and O_5) is shown.

The problem of determining the "minimal" or "most important" reasons why a task cannot be accomplished has been examined from the perspectives of propositional logic (Göbelbecker et al. 2010) and linear temporal logic (Cimatti et al. 2008). One can also view this problem as a generalization of the task of determining the non-existence of a path between two points (Canny 1987; McCarthy, Bretl, and Hutchinson 2012; Zhang, Kim, and Manocha 2008).

Since determining the existence of a collision-free path between two points (the *piano-mover's problem*) is PSPACE-hard in general (Reif 1979), the minimum constraint removal problem is also at least that hard. Even some very restricted versions of the problems remain difficult. Sokoban is a motion planning game played on a subgraph of the square grid, in which the goal is to push movable obstacles into certain goal locations. Sokoban remains

PSPACE-complete (Culberson 1997). Sliding block puzzles are PSPACE-complete even when all blocks are 1x2 dominoes (Hearn and Demaine 2005).

However, there exist situations in which the piano-mover's problem is easy. If the robot is navigating in the plane and the obstacle regions are polygonal sets, then a straightforward cell decomposition will efficiently reveal the existence or non-existence of a path from the start to the goal (Chazelle 1987; LaValle 2006).

In this paper, we demonstrate that the corresponding minimum constraint removal problem is NP-hard, even when the obstacles are restricted to being convex polygons. This is done via a reduction from the maximum satisfiability problem for binary Horn clauses, a special case of MAX2SAT that remains NP-hard (Jaumard and Simeone 1987). This augments Hauser's discrete NP-hardness result. While it is possible to transform the convex polygon minimum constraint removal problem into a discrete, graph-based constraint removal problem by associating each connected region intersected by a particular set of obstacles with a single graph vertex, the graphs and obstacles obtained from this transformation are very restricted (see Figure 2). The graphs must be planar, and the subgraph induced by the set of vertices intersected by a particular obstacle must be connected (though other restrictions also apply). The graphs used by Hauser in the discrete NP-hardness proof are generally non-planar, and the graphs induced by the set of vertices intersected by an obstacle are not generally connected.

Section 2 formally defines the minimum constraint removal problem. Section 3 describes the maximum satisfiability problem for binary Horn clauses. Section 4 describes types of obstacles that are used in the reduction. Section 5 describes the gadgets that encode the variables and clauses of the satisfiability problem. Section 6 explains how to obtain the solution to the maximum satisfiability problem from the result of the minimum constraint removal problem. Section 7 discusses the implications of this NP-hardness result and lists open problems.

2 Problem Formulation

A point robot navigates in $\mathbb{R}^2$. Let $\{O_1, O_2, \ldots, O_m\}$ be a set of m *obstacles*. Each obstacle is a convex polygonal subset of $\mathbb{R}^2$. It is acceptable for obstacles to be degenerate 1-gons (points) or 2-gons (line segments). It should be noted that either type of degenerate polygon can be simulated by a non-degenerate polygon that is sufficiently small (in the case of 1-gons) or sufficiently thin (in the case of 2-gons). Let $q_s \in \mathbb{R}^2$ be the *starting point*. Let $q_g \in \mathbb{R}^2$ be the *goal point*.

Let y be a continuous path from q_s to q_g. We will treat y as a set of points, as the parameterization of the path is irrelevant. The *cover* of y is the set of obstacles that intersect y. A *minimum constraint removal path* y^* is a path whose cover has a minimal size among all paths from q_s to q_g. A *minimum constraint removal* S^* is the cover of a minimum constraint removal path.

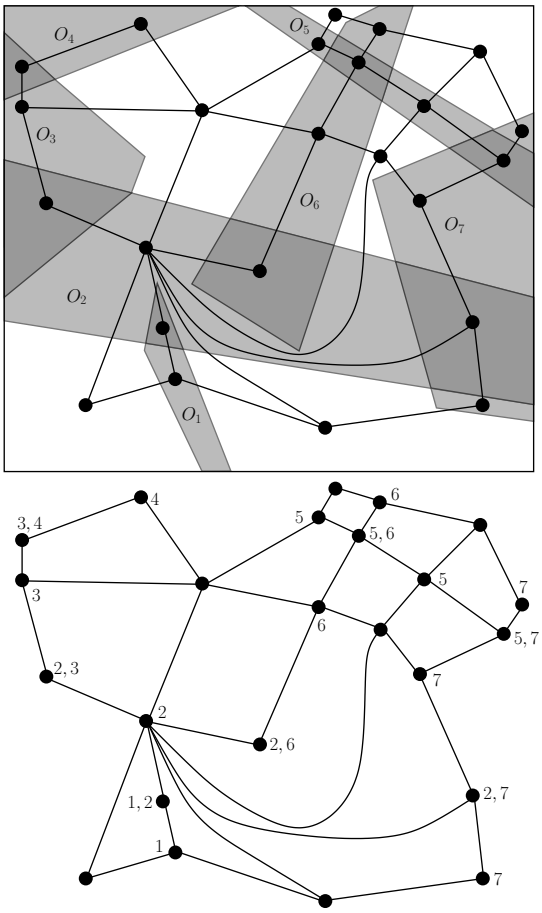

Figure 2: [top] The environment from Figure 1 with each connected region intersected by a particular set of obstacles assigned a graph vertex. Since movement within a single region does not intersect any new obstacles, the minimum constraint removal problem for this obstacle set can be transformed into a discrete problem. [bottom] The graph for the discrete minimum constraint removal problem. The numbers next to a vertex denote the obstacles intersecting that vertex.

3 Maximum Binary Horn Clause Satisfiability

A *clause* is a disjunction of literals. A *Horn clause* is a clause that contains at most one positive literal. A *binary clause* is a clause that contains exactly two literals. The decision problem about the satisfiability of a conjunction of Horn clauses (HORNSAT) is solvable in linear time (Dowling and Gallier 1984). However, the corresponding maximization problem "Given a set of n Horn clauses, what is the maximum number that can be simultaneously satisfied?" (MAXHORNSAT) is NP-hard. Similarly, the decision problem about the satisfiability of a conjunction of binary clauses (2SAT) is in P, but the corresponding maximization problem (MAX2SAT) is also NP-hard.

The maximization problem remains NP-hard even if all

clauses are required to be both Horn and binary (Jaumard and Simeone 1987). This version of the problem is referred to as MAX2HORNSAT. We will reduce MAX2HORNSAT to the planar minimum constraint removal problem with convex polygonal obstacles. To do this, we will use an input MAX2HORNSAT problem to design a start point, goal point, and set of obstacles. The solution to the minimum constraint removal problem using that start point, goal point, and set of obstacles can quickly be transformed into the solution to the input MAX2HORNSAT problem. The number of clauses of the input problem will be represented by c, and the number of variables will be represented by ℓ.

4 Obstacle Weighting

While the penalty for entering one obstacle is the same as the penalty for entering any other obstacle, an obstacle of weight k can be simulated by placing k standard obstacles on the same set of points. The reduction uses three different obstacle weights.

- *Low-weight obstacles* - These obstacles have unit weight. In figures, low-weight obstacles will be represented by grey lines and grey squares.
- *Medium-weight obstacles* - These obstacles have a weight of $a+1$, where a is the number of low-weight obstacles. In figures, medium-weight obstacles will be represented by dotted lines.
- *High-weight obstacles* - These obstacles have a weight of $b(a+1)+a+1$, where a is the number of low-weight obstacles, and b is the number of medium-weight obstacles.

All obstacles used in the reduction are squares and line segments.

The obstacle sets that we use will leave a path from q_s to q_g that does not cross a high-weight obstacle. Since the cost of crossing a high-weight obstacle is greater than the cost of crossing all low and middle-weight obstacles combined, no minimum constraint removal path will cross a high-weight obstacle. Similarly, each path from q_s to q_g that avoids high-weight obstacles must cross at least c medium-weight obstacles, and there exists at least one path that crosses exactly c medium-weight obstacles. Since the cost of crossing a medium-weight obstacle is higher than the cost of crossing all the low-weight obstacles, each minimum constraint removal path will cross exactly c medium-weight obstacles.

5 Gadgets

In order to force the robot to take a predictable route, a path leading from q_s to q_g will be outlined with high-weight obstacles. In the figures, this path is shown in black. Portions of this path will be blocked by low-weight and medium-weight obstacles. Since the cost of intersecting a high-weight obstacle is higher than the cost of intersecting all of the low and medium-weight obstacles combined, there is no reason for the robot to leave the outlined path.

The input MAX2HORNSAT problem will be encoded as a minimum constraint removal problem through the use of gadgets. Gadgets are sets of obstacles that represent portions of the input problem. There are three primary gadgets that will be used to encode the MAX2HORNSAT problem. One type of gadget is used to assign values to the variables, one type of gadget is used to represent clauses containing one positive and one negative literal, and one type of gadget is used to represent clauses containing two negative literals.

Gadgets that establish the variables and clauses for the MAX2HORNSAT problem take the form of loops in the path from q_s to q_g. These loops force the robot to choose one of two routes to get to q_g. In the variable gadgets, the route chosen determines whether the variable is set to true or false. In the clause gadgets, the routes represent the two possible variable assignments that satisfy the clause.

5.1 Variable Gadgets

These gadgets exist to assign true or false values to variables. Each variable gadget consists of two loops. Taking the left path through a loop assigns the value of "true" to the corresponding variable. Taking the right path assigns a value of "false". A line-shaped medium-weight obstacle intersects the left path of both loops, and another line-shaped medium-weight obstacle intersects the right path of both loops. This is done to ensure that the path taken in the second loop corresponds to the same value as the path taken in the first loop. In order for the medium-weight obstacles to not intersect loops of variables that they are not associated with, the loops are nested and decrease in size (both loops of x_2 are in between the loops of x_1, and the loops of x_1 are wider than the loops of x_2, etc.). Figure 3 shows a variable gadget, and their placement relative to start point, goal point, and other gadgets.

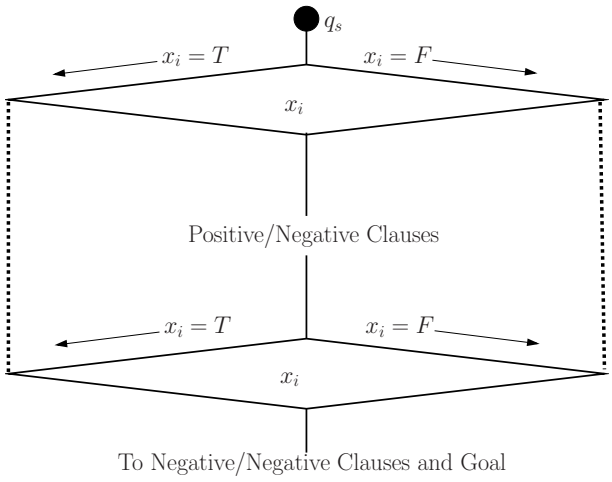

Figure 3: A variable gadget for x_i. Taking the left path sets x_i to true, and taking the right path sets x_i to false. Medium-weight obstacles (the dotted lines) ensure that the same assignment path is taken in both the upper and lower loops. The upper loop of the gadget appears before the positive/negative clause gadgets, and the lower loop appears after the positive/negative clause gadgets.

In order to establish the relationships between the variables and the clauses, the left and right paths of the loops in

these gadgets are blocked by low-weight obstacles that also block paths of the clause gadgets.

5.2 Positive/Negative Clause Gadget

These gadgets are located between the upper and lower loops of the variable gadgets. A positive/negative clause gadget consists of a single loop. The left path is blocked by a low-weight obstacle that also intersects the left path in the upper loop of the variable corresponding to the positive literal. The right path is blocked by a low-weight obstacle that intersects the right path in the upper loop of the variable corresponding to the negative literal. Figure 4 shows a positive/negative clause gadget, the variable loops associated with it, and the low-wight obstacles (in grey) that establish the relationship between the clause and the variables.

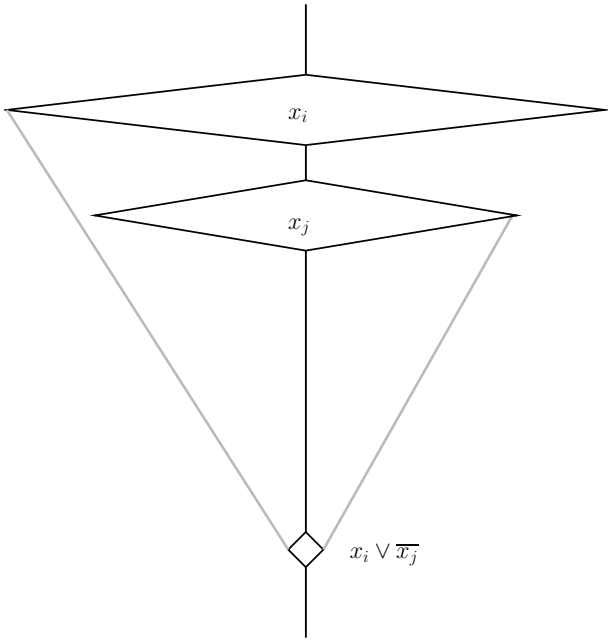

Figure 4: A clause gadget for $x_i \vee \overline{x_j}$. A single low-weight obstacle blocks the left path of the x_i variable loop and the left path of the clause loop. Another low-weight obstacle blocks the right path of the x_j variable loop and the right path of the clause loop. If x_i is true, then the left path of the clause loop can be taken without intersecting an additional obstacle. If x_j is false, then the right path of the clause loop can be taken without intersecting an additional obstacle.

Since the robot has already passed through the upper variable gadget loops before reaching the positive/negative clauses, the variables have already had values assigned to them. If the clause is satisfiable, then either the obstacle blocking the left path of the clause gadget has already been crossed in a variable gadget, or the obstacle blocking the right path of the clause gadget has already been crossed in a variable gadget. Since there is no penalty for crossing a single obstacle multiple times, if the clause is satisfied, the robot can pass through it without crossing any new obstacles. If the clause is not satisfied, the robot crosses a new obstacle, increasing the size of the path's cover.

5.3 Negative/Negative Clause Gadget

These gadgets are located after the lower loops of the variable gadgets, and before q_g. The upper path of this gadget is blocked by an obstacle that also blocks the right path of the upper loop of one of the variable gadgets. The lower path of the gadget is blocked by an obstacle that also blocks the right path of the lower loop of one of the variable gadgets. If either of those variables is assigned a negative value, then at least one of those obstacles will have already been crossed in the variable gadget, and the clause gadget can be crossed without additional penalty. Otherwise, as above, the robot must cross an additional obstacle. A negative/negative clause gadget and associated variable loops and low-weight obstacles are shown in Figure 5.

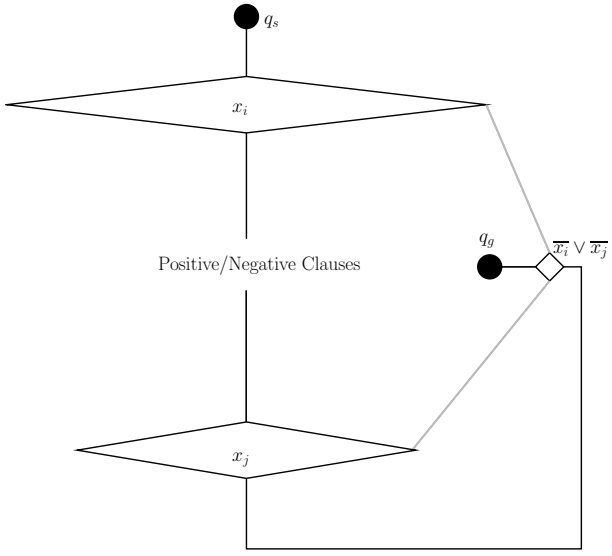

Figure 5: A clause gadget for $\overline{x_i} \vee \overline{x_j}$. A single low-weight obstacle blocks the right path of the x_i variable loop and the left path of the clause loop. Another low-weight obstacle blocks the right path of the x_j variable loop and the right path of the clause loop. If x_i is true, then the left path of the clause loop can be taken without intersecting an additional obstacle. If x_j is false, then the right path of the clause loop can be taken without intersecting an additional obstacle.

5.4 Balancing Obstacles

At this point, the number of obstacles blocking the left path of a variable loop might be different than the number of obstacles blocking the right path of a variable loop. For example, if the literal x_i appears in several clauses, but $\overline{x_i}$ does not appear in any clauses, the left path of the upper loop of x_i's variable gadget will be blocked by many low-weight obstacles, but the right path will not be blocked by any. In order to ensure that the cost of setting a variable to be true is the same as the cost of setting it to be false, for each obstacle

that is used to establish a relationship between a variable and a clause, a *balancing obstacle* will be added to the other path of the variable loop. This balancing obstacle will block the variable loop, but will not intersect any clauses. Balancing obstacles are represented by grey squares, and can be seen in Figure 6.

6 Reduction

Since a path from q_s to q_g exists that does not intersect any high-weight obstacles, a minimum constraint removal path will not intersect any high-weight obstacles. Additionally, the minimum number of medium-weight obstacles that can be crossed on a path from q_s to q_g is c, the number of clauses. Each literal within a clause adds an obstacle to the left and right paths of the associated variable gadget loop (one obstacle used to establish the relationship between the clause and the variable, and one balancing obstacle), one of which needs to be crossed to reach the goal. Since there are two literals per clause, a minimum constraint removal path will cross at least $2c$ low-weight obstacles while setting the truth values of the variables. Finally, each clause that is not satisfied requires the crossing of an additional low-weight obstacle. Therefore, if no clauses can be satisfied, the size of the minimum constraint removal is

$$(a+1)\ell + 3c. \tag{1}$$

For each clause that can be satisfied, the size of the minimum constraint removal is decreased by 1. Therefore, given a set of c clauses over ℓ variables that produces a minimum constraint removal of size z, the maximum number of clauses that can be simultaneously satisfied is

$$(a+1)\ell + 3c - z. \tag{2}$$

Therefore, MAX2HORNSAT is reducible to the minimum constraint removal problem in the plane even when all obstacles are convex, indicating that the latter problem is NP-hard. A complete representation of a MAX2HORNSAT problem as a minimum constraint removal problem is shown in Figure 6.

7 Discussion and Open Problems

Navigation in $\mathbb{R}^2$ with convex polygonal obstacles is a highly restricted form of path planning. The NP-hardness of the minimum constraint removal problem under these restrictions immediately implies the NP-hardness of the problem for higher dimensions and for less restricted types of obstacles.

What restrictions on the environment and obstacles make the problem tractable? In the plane, the minimum constraint removal problem is trivial when the obstacles are forced to be infinite lines or half-planes. It is not known whether forcing all obstacles to be unit circles, circles, or axis-aligned rectangles makes the problem tractable, though the authors conjecture that the first is tractable and the latter two are not. One could also place restrictions on the ways in which the obstacles are allowed to intersect. The problem becomes trivial when the obstacles are allowed to intersect only pairwise. It is not known at what point restrictions of the form

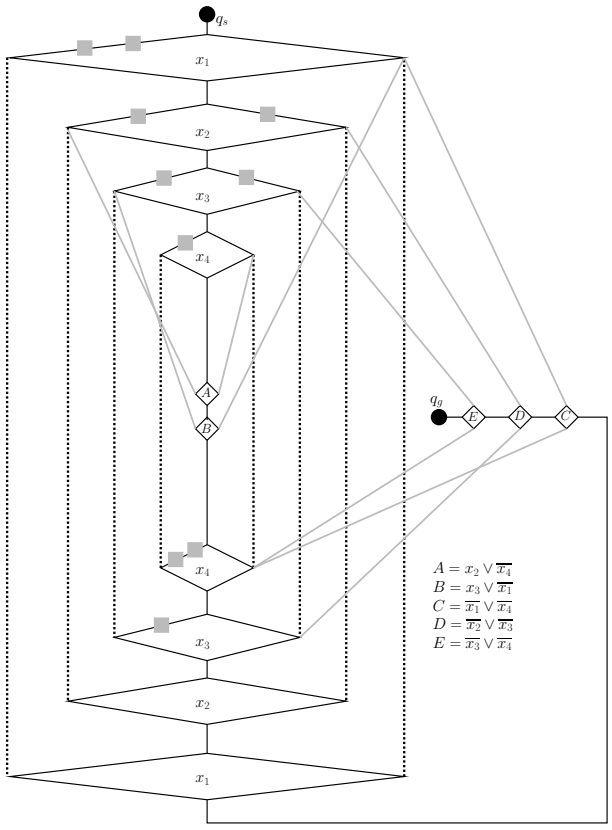

Figure 6: A MAX2HORNSAT problem with four variables and five clauses represented as a minimum constraint removal problem. High-weight obstacles surround the solid black path between q_s and q_g. Medium-weight obstacles are represented by dotted black lines. Low-weight obstacles are represented by grey lines and grey squares.

"There is no point at which more than k obstacles intersect" or "Each obstacle intersects no more than k other obstacles" render the problem tractable.

Acknowledgements

This work is supported in part by NSF grant 0904501 (IIS Robotics), NSF grant 1035345 (CNS Cyberphysical Systems), DARPA SToMP grant HR0011-05-1-0008, and MURI/ONR grant N00014-09-1-1052.

References

Canny, J. 1987. A new algebraic method for robot motion planning and real geometry. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, 39–48.

Chazelle, B. 1987. *Approximation and Decomposition of Shapes*. Lawrence Erlbaum Associates. chapter 4, 145–185.

Cimatti, A.; Roveri, M.; Schuppan, V.; and Tchaltsev, A. 2008. Diagnostic information for realizability. In *Verification, Model Checking, and Abstract Interpretation*, 52–67.

Culberson, J. C. 1997. Sokoban is pspace-complete. Technical Report TR 97-02, The University of Alberta, Edmonton, AB.

Dowling, W. F., and Gallier, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming* 1(3):267 – 284.

Göbelbecker, M.; Keller, T.; Eyerich, P.; Brenner, M.; and Nebel, B. 2010. Coming up with good excuses: What to do when no plan can be found. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling*, 81–88.

Hauser, K. 2012. The minimum constraint removal problem with three robotics applications. In *Proc. Workshop on the Algorithmic Foundations of Robotics*.

Hearn, R., and Demaine, E. 2005. Pspace-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science* 343(1):72–96.

Jaumard, B., and Simeone, B. 1987. On the complexity of the maximum satisfiability problem for horn formulas. *Information Processing Letters* 26(1):1 – 4.

LaValle, S. M. 2006. *Planning Algorithms*. Cambridge, UK: Cambridge University Press. Also available at http://msl.cs.uiuc.edu/planning/.

McCarthy, Z.; Bretl, T.; and Hutchinson, S. 2012. Proving path non-existence using sampling and alpha shapes. In *Proc. IEEE International Conference on Robotics and Automation*, 2563–2569.

Reif, J. 1979. Complexity of the mover's problem and generalizations extended abstract. In *Proceedings of the 20th Annual IEEE Conference on Foundations of Computer Science*, 421–427.

Zhang, L.; Kim, Y.; and Manocha, D. 2008. A simple path non-existence algorithm using c-obstacle query. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, 269–284.