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 (**?**). 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.

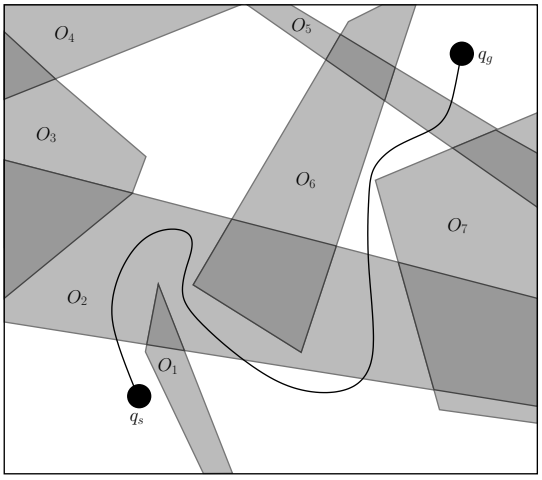

Figure 1: An instance of the minimum constraint removal problem. A path from the start (q_s) to the goal (q_g) that intersects the minimum number of obstacles (O_2 and O_5) is shown.

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 (**?**) and linear temporal logic (**?**). One can also view this problem as a generalization of the task of determining the non-existence of a path between two points (**?**; **?**; **?**).

Since determining the existence of a collision-free path between two points (the *piano-mover's problem*) is PSPACE-hard in general (**?**), 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 (**?**). Sliding block puzzles are PSPACE-complete

even when all blocks are 1x2 dominoes (**?**).

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 (**?**; **?**).

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 quadratic Horn clauses, a special case of MAXHORNSAT that remains NP-hard (**?**). 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 quadratic 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.

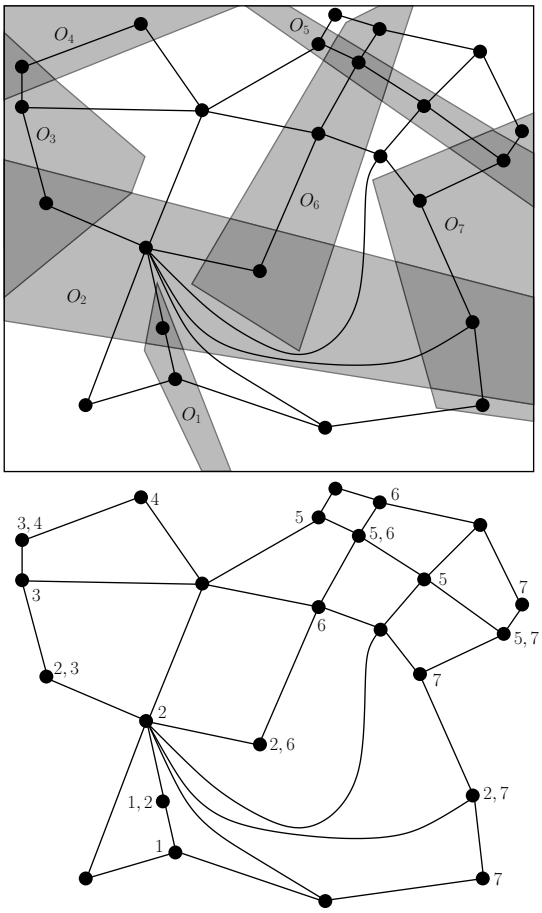

Figure 2: [top] The environment from Figure 1 with each connected region intersected by a particular set of obstacles assigned a graph vertex. Since movement within a single region does not intersect any new obstacles, the minimum constraint removal problem for this obstacle set can be transformed into 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 Quadratic Horn Clause Satisfiability

A *clause* is a disjunction of literals. A *Horn clause* is a clause that contains at most one positive literal. A *quadratic clause* is a clause that contains at most two literals. The decision problem about the satisfiability of a conjunction of Horn clauses (HORNSAT) is solvable in linear time (**?**). 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 quadratic clauses (QSAT) is in P, but the corresponding maximization problem (MAXQSAT, equivalent to the more well-known MAX2SAT problem) is also NP-hard.

The maximization problem remains NP-hard even if all clauses are required to be both Horn and quadratic (**?**). This version of the problem is referred to as MAXQHORNSAT. [1] We will reduce MAXQHORNSAT to the planar minimum constraint removal problem with convex polygonal obstacles. To do this, we will use an input MAXQHORNSAT 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 MAXQHORNSAT problem. The number of clauses of the input problem will be represented by c, and the number of variables will be represented by v.

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 MAXQHORNSAT 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 MAXQHORNSAT 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 MAXQHORNSAT 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.

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 Clause Gadgets

Additional gadgets are required to establish the relationships between the clauses and the variables. Since all clauses are quadratic and Horn, three different types of clause gadgets are required. One for clauses with a positive and a negative literal ($x_i \vee \overline{x_j}$), one for clauses with two negative literals ($\overline{x_i} \vee \overline{x_j}$), and one for clauses with a single positive literal (x_i). While the quadratic and Horn conditions would also allow clauses with a single negative literal, a separate gadget is not needed, as those clauses can be handled by the gadgets for clauses with two negative literals ($\overline{x_i} = \overline{x_i} \vee \overline{x_i}$).

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

[1] The original version of this paper erroneously used MAX2HORNSAT as the problem that was reduced to MCR to demonstrate NP-hardness. In MAX2HORNSAT, each clause has *exactly* two literals instead of *at most* two literals. Even though MAX2SAT and MAXQSAT are of equivalent difficulty, MAX2HORNSAT is not NP-hard, as each clause can be satisfied by setting all variables to "false", while MAXQHORNSAT remains NP-hard.

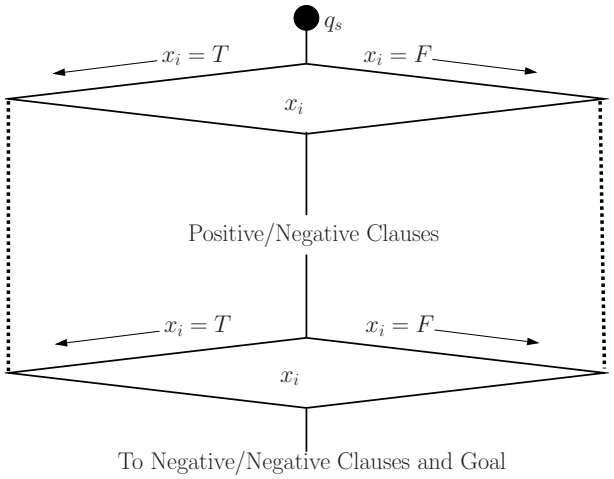

Figure 3: A variable gadget for x_i. Taking the left path sets x_i to true, and taking the right path sets x_i to false. Medium-weight obstacles (the dotted lines) ensure that the same assignment path is taken in both the upper and lower loops. The upper loop of the gadget appears before the positive/negative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

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.

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.

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.

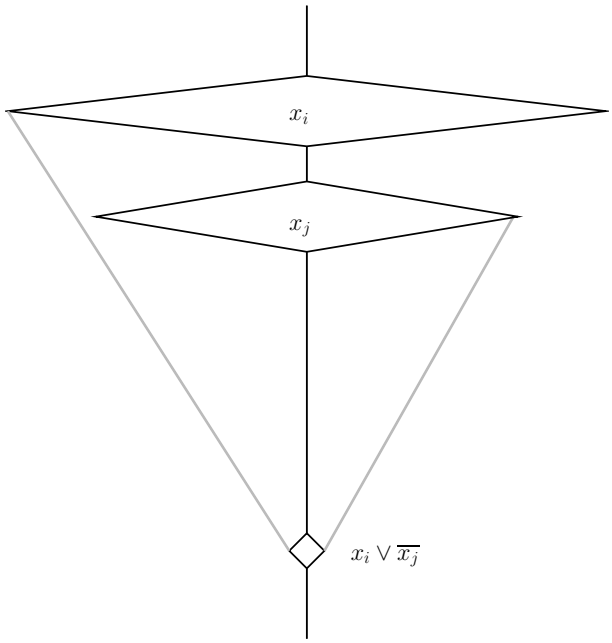

Figure 4: A clause gadget for $x_i \vee \overline{x_j}$. A single low-weight obstacle blocks the left path of the x_i variable loop and the left path of the clause loop. Another low-weight obstacle blocks the right path of the x_j variable loop and the right path of the clause loop. If x_i is true, then 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.

Positive Clause Gadget These gadgets consist of a single obstacle, and do not involve a loop. If x_i is a clause, then place a low-weight obstacle with one endpoint on the left path of the upper loop of x_i variable gadget, and another endpoint placed on the path between the upper looks of the x_i and x_{i+1} gadgets (or between the x_i gadget and the positive/negative gadgets if x_i is the innermost set of variable loops).

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

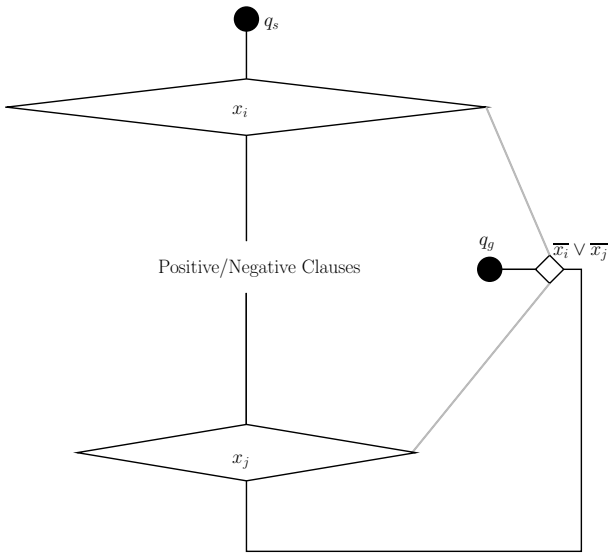

Figure 5: A clause gadget for $\overline{x_i} \vee \overline{x_j}$. A single low-weight obstacle blocks the right path of the x_i variable loop and the left path of the clause loop. Another low-weight obstacle blocks the right path of the x_j variable loop and the right path of the clause loop. If x_i 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.

by grey squares, and can be seen in Figure 7.

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. Let ℓ be the total number of literals in all clauses. A minimum constraint removal path will cross at least ℓ 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

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

in which $a+1$ is the weight of a medium-weight obstacle.

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 with ℓ total literals over v variables that produces a minimum constraint removal of size z, the maximum number of clauses that can be simultaneously satisfied is

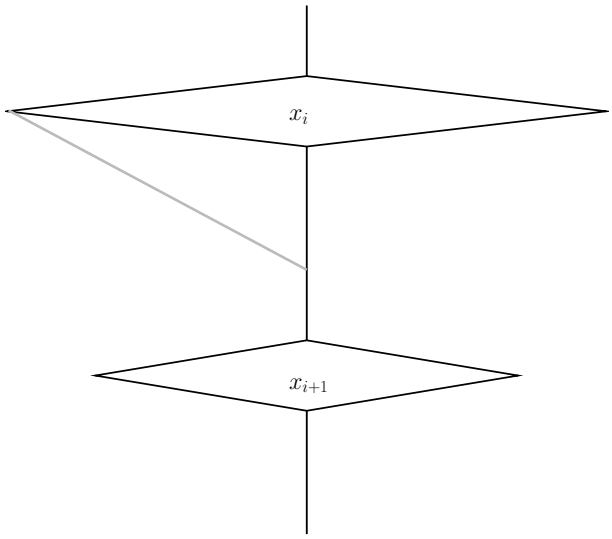

Figure 6: A clause gadget for x_i. A single low-weight obstacle blocks the left path of the x_i variable loop and path between the upper loops of x_i and x_{i+1}.

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

Therefore, MAXQHORNSAT 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 MAXQHORN-SAT problem as a minimum constraint removal problem is shown in Figure 7.

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 "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

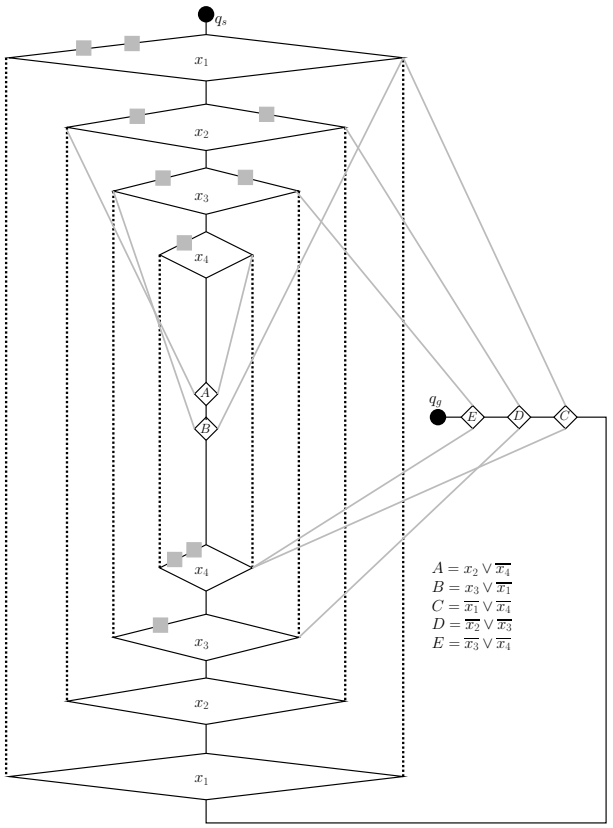

Figure 7: A MAXQHORNSAT problem with four variables and five clauses represented as a minimum constraint removal problem. High-weight obstacles surround the solid black path between q_s and q_g. Medium-weight obstacles are represented by dotted black lines. Low-weight obstacles are represented by grey lines and grey squares.

Systems), DARPA SToMP grant HR0011-05-1-0008, and MURI/ONR grant N00014-09-1-1052.

一个简单的，但是属于 NP 难的运动规划问题

Lawrence H. Erickson and Steven M. LaValle
伊利诺伊州大学香槟分校

摘 要

在机器人领域中一个十分基础的问题之一就是在两点之间确定是否存在一条无碰撞的路径。但是，在一些场景中跨越一些障碍物虽然有一定的耗费但也不是不可，所以更加合适的是为了找寻到一条路径可以越过或者移除一些障碍物。这种情况在室外或者室内都有可能出现。室外中机器人可以穿越一些硬质的障碍物，室内机器人可以推开门等障碍物。这个问题，形容为最小约束移除问题，被认为是和路径存在问题一样难度的。在本篇论文中，我们展示了在导航用途时最小约束移除问题是 NP 难的，即使所有障碍物都是凸多边形的时候。

1. 介绍

在大多数机器人路径规划问题中，任务目标是从起始状态行进到目标状态并且在途中不会触碰到障碍物。然而，在许多情况下，与障碍物的碰撞虽然会产生不便，但也并非不可克服。例如障碍物是机器人能够打开或者关闭的门。在室外导航中，障碍物可以是机器人能够穿越的崎岖地形，但穿越这种地形存在损坏机器人的风险。或者同等的，可能需要知道阻挡两点之间路径的最小障碍物集合。

考虑到这一点，出现了一个自然的问题：如果从起始状态到目标状态没有可行路径存在，那么与路径交叉的障碍物数量最少为多少？由 Hauser 引入的这个问题被称为最小约束去除问题，因为它要求从环境中移除最小数量的约束（障碍物）以产生无碰撞路径。示例问题如图 1.1 所示。

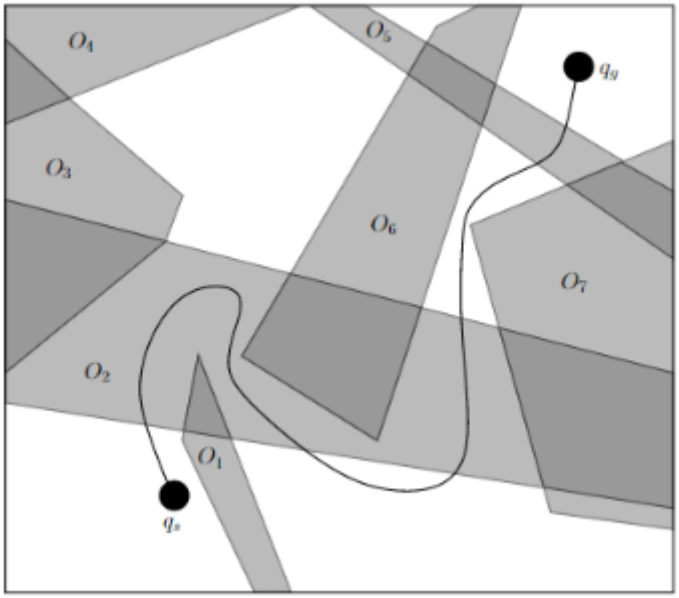

图 1.1 最小约束移动问题举例

Hauser 通过对最小集覆盖问题的简化展示了这个问题的离散版本是 NP 难的问题（当环境利用图表示出来的时候，其中障碍物顶点集是整个顶点集的子集）。这个结果非常有意思，因为相同的路径存在问题在使用广度优先搜索算法很容易在图结构中解决。

从命题逻辑和线性时态逻辑的角度考察了确定无法完成任务的“最小”或“最重要”原因的问题。也可以将这个问题看成两点之间确定不可行路径的任务的统一化。

因为在两点之间确定是否存在无冲突路径（钢琴移动问题）通常是 PSPACE 难得，所以最小约束重新问题也至少是那么难。 甚至一些约束很多的问题版本依旧很难。Sokoban 是一个在方形网格的子图上进行的运动规划游戏，其目标是将可移动的障碍物推入某些目标位置。推箱子仍然是 PSPACE 完全的。即使所有移动块都是 1x2 多米诺骨牌，滑块拼图也是 PSPACE 完全难度的。

然而，存在钢琴移动问题很容易解决的情况，如果机器人在平面上导航并且障碍物区域是多边形集合，那么直接进行分解法将有效地得到从开始到目标的路径的存在或不存在的结果。

在本文中，我们证明了最小约束去除问题是 NP 难的，即使障碍被严格限制为凸多边形。 这是通过减少二次 Horn 子句的最大可满足性问题来实现的，这个条件被认为是 MAXHORNSAT 的特殊情况，但它仍然是 NP 难的问题。这个验

证了 Hauser 的离散 NP 难的结果。虽然可以将凸多边形最小约束移除问题转化为离散的图论约束移除问题（图中的顶点分别从每一个连通域中采样得到），但是从这个转化中所得到的图和障碍物其实是十分有限的（见图 1.2）。图形必须是平面的，并且必须连接由特定障碍物所包围的顶点集合引起的子图。 Hauser 使用的离散 NP 难的图通常是非平面的，并且由与障碍物相交的顶点集产生的图通常是不连通的。

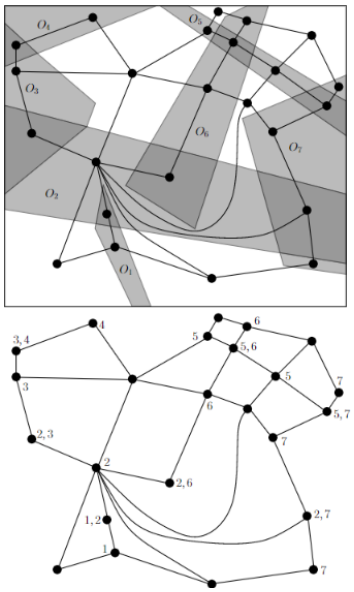

图 1.2 环境采样离散化过程

第 2 节正式定义了最小约束移动问题。第 3 节描述了二次 Horn 子句的最大满足的问题。第 4 节描述了减少过程中使用的障碍类型。第 5 节描述了编码可满足性问题的变量和条款的小工具。第 6 节解释了如何从最小约束去除问题的结果中获得最大可满足性问题的解。第 7 节讨论了这种 NP 难结果的含义并列出了开放性问题。

2. 问题阐述

将机器人表示为一个质点在R^2中进行运动，让$O_1, O_2, ..., O_m$表示为 m 个障碍物域集合，每一个障碍物时凸多边形，并且为R^2的子集，其中障碍物可以退化为一个点或者时一条线段。让$q_s \in \mathrm{R}^2$表示为初始点，让$q_g \in \mathrm{R}^2$表示为目标点。

用 y 表示从q_s到q_g的一条连续路径，我们将 y 理解为点的集合，y 的覆盖表

示为和 y 相交的障碍物域集合。最小约束移除路径y^*是指在初始域到目标域的路径中代价最小的一条。最小约束移除S^*是指最小约束移除路径的一个覆盖。

3. 最大化二次 Horn 子句满足性

子句是文字的一个分离，一次 Horn 子句表示只有一个正字的语句，二次 Horn 子句表示最多有两个正字的语句。关于 Horn 子句连接的满足行的决定问题是可以在线性时间内解决的。但是相应的最大值问题“给定 n 个 Horn 子句，哪个是最大的能够被同时满足的值”是 NP 难的问题。类似的，关于二次向子句的连接可满足性是 P 难得，但是在相应的最大值问题确是 NP 难的。

最大值问题依旧属于 NP 难即使所有的子句都被规定为是 Horn 子句并且为二次的，这个版本可以引申为 MAXQHORNSAT 问题，我们将会将 MAXQHORNSET 问题简化为平面中的在凸多边形障碍物环境中的最小约束移除问题。为了解决这个问题，我们将会使用输入 MAXQHORNSET 问题来设计起始点，目标点还有障碍物集合。最小约束移除问题的解决方法可以通过起始点、终点域障碍域集合快速地转换为输入 MAXQHORNSET 的解决方案。子句（clause）的数量用 c 表示，变量的个数用 v 表示。

4. 障碍物加权

因为进入每一个障碍物的惩罚和进入另一个障碍物的惩罚是一样的，所以一个权重为 k 的障碍物可以由 k 个标准障碍物在同一个点集中聚集来表示。路径缩减过程中利用了三种不同的障碍物加权的方法。

（1） 低权障碍物，这些障碍物拥有标准权重。在图像中，低权障碍物用灰色线段与多边形表示；

（2） 中权障碍物，这些障碍物拥有 a+1 的权重，其中 a 表示低权障碍物的个数。在图像中，中权障碍物用点线边缘来表示；

（3） 高权障碍物，这些障碍物拥有 b(a+1)+a+1 的权重，其中 a 表示低权障碍物的个数，b 表示中权障碍物的个数。

在本文中，路径只有穿过了低权与中权障碍物域，因为穿过高权障碍物的代价比穿过低权加上中权障碍物的代价还要高。类似的，每条由q_s到q_g的路径都会穿过 c 个中权障碍物域。由于穿越中权障碍物的成本高于穿越低权障碍物的成本，因此每个最小约束重新移动路径将跨越精确的 c 个高权障碍物。

5. 工具插件

为了使得机器人能够沿着预计得路径行走，一条由q_s到q_g的路径会用高权线段表现出来。在图像中，这些路径会用黑色表现出来。输入 MAXQHORNSET 问题会通过工具插件被编码为最低约束移除问题。Gadgets 表示输入问题的一部分子集，用来表示障碍物集合。一共有三种主要的工具用来给 MAXQHORNSET 问题编码。一种是给变量赋值，一种是用子句来表示，其中正反义的文字各一个，另一种是用子句表示两个反义的文字。

用来为MAXQHORNSET构建变量与子句的工具在从q_s到q_g的一条连续路径中不断的循环。这个循环强制机器人从两个路径中选择一条来到达q_g。在变量插件中，选择的路径用来表示变量为真或者为否。在从句插件中，路径用来表示两个可能的变量赋值是否满足从句。

5.1 变量插件

变量插件是用来给变量赋值 true 或者 false 的，每一个变量插件包含两个循环。选择左侧路径在循环中相应的变量赋值 true，选择右侧路径赋值 false。在循环中，中权线段障碍物与左侧与右侧的路径均相交，这样做的目的是为了保证第二次循环能够对第一次循环的值相互对应。为了使得中权障碍物不与和它们不相关的变量的交叉循环，我们将循环嵌套并减小尺寸。图 5.1 表示了变量插件相对于初始点、目标点和其他插件的相对位置。为了建立变量与子句之间的关系，循环左边和右边的路径都被低权障碍物阻隔了，这些低权障碍物同时也阻隔了子句插件。

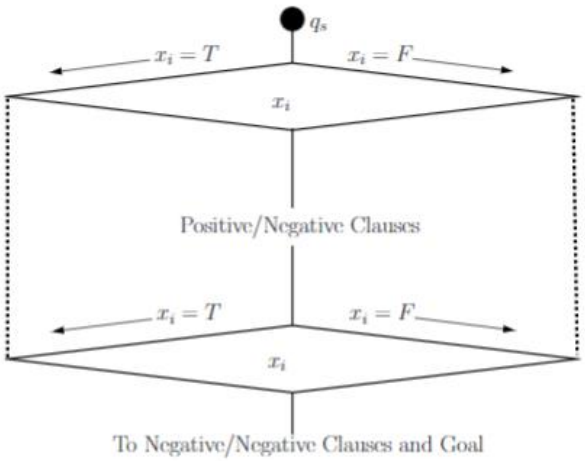

图 5.1 整体流程

5.2 从句插件

需要其他小工具来建立子句和变量之间的关系，因为所有子句都是二次 Horn 的，因此需要三种不同类型的子句小工具，第一种为一正一负的表达式$(x_i \vee \overline{x_j})$，第二种为全负表达式$(\overline{x_i} \vee \overline{x_j})$，第三种表示是只含有唯一正义的表达式$(x_i)$。虽然二次的 Horn 条件也允许使用单个否定文字的子句，但这并不需要单独工具，因为这些子句由具有两个否定文字的子句处理就行，例如$(\overline{x_i} = \overline{x_i} \vee \overline{x_i})$。

5.2.1 正/负子句插件

这些小工具位于变量小工具的上下循环之间。正/负子句插件含有单个循环，左侧路径被低权障碍物阻挡，该障碍物也与对应于正表达式的变量上部循环中的左侧路径相交。右路径被一个低权障碍物阻挡，该障碍物与变量的上部循环中的右侧路径相交，对应于负表达式。图 5.2 表达了该插件的关系，变量循环在子句与变量之间建立起了联系。

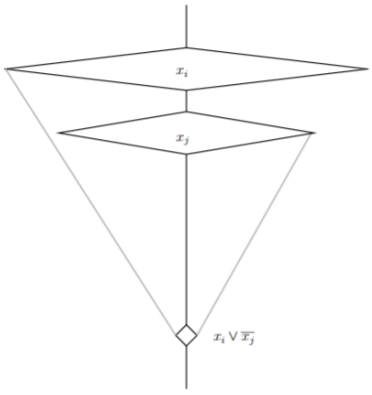

图 5.2 正/负子句

由于机器人在到达正/负子句之前已经通过了上循环，因此变量已经具有分配给它的值。如果该子句是可满足的，则子句插件的左侧路径的障碍物已经在变量工具中被穿越，或者子句插件的障碍物阻塞路径已经在变量插件中被穿越。由于多次越过单个障碍没有惩罚，所以如果表达式满足，机器人可以通过它而不会越过任何新的障碍。如果不满足该子句表达式，则机器人穿过新的障碍物，增加路径的覆盖。

5.2.2 负/负子句插件

这些插件位于变量下循环之后，并且是在目标点之前的。该插件的上部路径被障碍物阻挡，该障碍物也阻挡其中一个变量的上部循环的右侧路径。插件的下部路径被障碍物阻挡，障碍物也阻挡了其中一个变量插件的下部环路的右侧路径。如果这些变量中的任何一个被赋予负值，那么至少有一个障碍已经在变量插件中被越过，并且子句插件可以在没有额外的情况下进行障碍物的翻越。否则的话，机器人只能跨越多余的障碍物。一个负/负子句插件如图 5.3 所示。

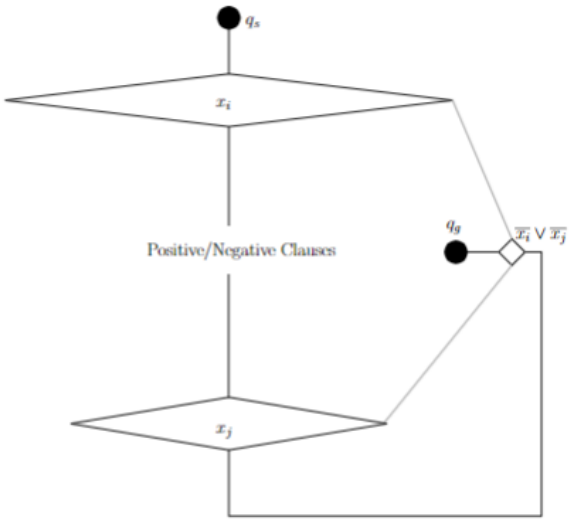

图 5.3 负/负子句

5.2.3 正义子句插件

这些子句插件只包含一个障碍物并且不包含循环。如果x_i是一个从句，那么一个低权障碍物将在左侧路径和上循环中被放置，并且在另外的上循环x_i和x_{i+1}放置。

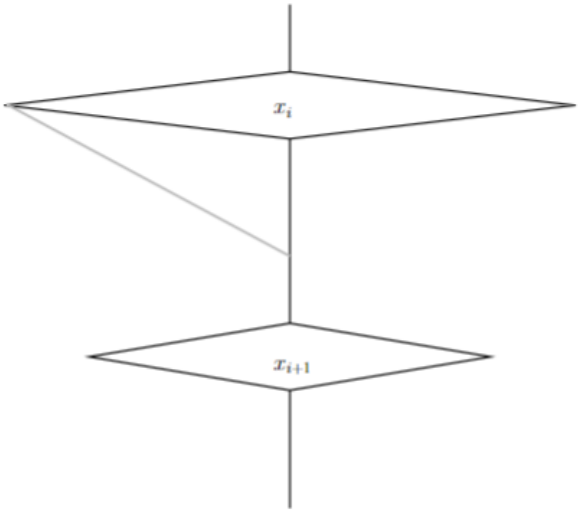

图 5.4 正义表达式

5.2.4 平衡障碍物

此时，阻塞变量循环的左侧路径的障碍物数量可能与阻变量循环的右路径障碍物数量不同。举个例子，如果x_i在很多子句中都出现了，但是$\overline{x_i}$没有在任何子句中出现，那么上循环左侧的x_i变量插件就是被很多低权障碍物所阻碍了，但是右侧路径则完全没有被阻碍。为了确保设定变量成本与设置成本的成本相同，对于用于建立变量和子句之间关系的每个障碍，我们将平衡障碍物添加到变量循环的其他路径中去。这种平衡障碍物将阻碍变量的循环，但不会影响子句。平衡障碍物由图 5.5 所示。

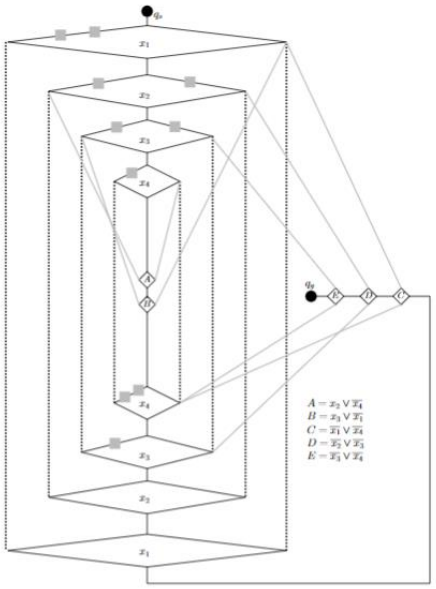

图 5.5 QMAXHORN 表达式

6. 缩减过程

因为从q_s到q_g的一条连续路径没有与高权障碍物相交，所以在最小约束移除的过程中不会和高权障碍物有关系。更多的，在最小约束移除的过程中被移动的障碍物个数为 c，也就是子句的个数。子句中的每个表达式都为相关变量插件循环的左侧和右侧路径添加了障碍（一个子句与变量之间关系的障碍物，以及一个平衡障碍物），其中一个需要交叉从而达到目标区域。用 l 表示全部的子句中表达式的个数，那么至少会穿过 l 个低权障碍物，当我们将所有的变量赋值为真的时候。最后，每个不符合的从句都会需要穿过一个多余的低权障碍物。因此，如果没有一个从句满足，那么最小约束移除的个数为$(a+1)v+l+c$，其中，a+1 表示中权障碍物的个数。

如果每一个从句都满足，那么在最小约束移动问题中每一个尺寸都减 1，因此给定一个 c 从句和总长为 l 并且包含 v 个变量的集合，一共可以产生规格为 z 的最小约束移除结果。同时，从句可满足的最大值为$(a+1)v+l+c-z$。因此，MAXQHORNSAT 可以被缩减为最小约束移除问题当平面内的障碍物全是凸多边形时。一个完整的流程图由 5.5 所示。

7. 讨论

机器人在R^2中进行运动是收到很多约束的。在平面中的最小约束移动问题是 NP 难的，这个现象表明了在高维的情况下最小约束移动问题也是 NP 难的，在约束更加小的情况下也是如此。

什么样的限制施加在环境和障碍上可以解决问题？在平面中，当障碍物被迫成为直线或半平面时，最小约束去除问题显然是微不足道的。目前尚不清楚的是如果将所有障碍都强行转化为单位圆或轴对齐的矩形会不会使问题易于处理，尽管我们猜想的是第一种想法是易处理的而后一种不是。当障碍物允许成对地相交时，问题将会变得难以理解。目前尚不清楚形式的限制是什么。“没有更多与障碍物相交的点”或“每个障碍物不再相交更多的障碍物”可能使问题变得更加易于处理。