Motion Planning Amidst Movable Square Blocks

Arundhati Dhagat
Joseph O'Rourke*

Abstract

We explore a particular motion planning problem with movable obstacles: all obstacles are unit square blocks on the integer lattice that can be moved horizontally or vertically by a pushing robot. We consider four cases, depending on (1) whether the obstacles are all movable (but confined to a rectangle), or some are affixed to the plane, and (2) whether or not the robot path must be monotonic. Not surprisingly, the monotonic path problems are polynomial. For general paths, the case where some blocks are immovable we prove to be NP-complete. This is our main result, which can be viewed as a sharpening of a similar result due to Wilfong. For the case where all blocks are movable, we were only able to find a weak necessary condition for the existence of a solution. We leave the complexity of this interesting motion planning problem open.

1 Introduction

1.1 Planning Environment

Inspired by a computer game for the Macintosh called "Beast 1.0,"[1] we studied the following motion planning problem. All obstacles are unit squares with corners on the integer lattice. Obstacles come in two varieties: movable and immovable. The former can be pushed (not pulled) by a robot, while the latter are fixed to the plane. The robot may move horizontally or vertically in discrete unit steps, pushing any number of movable obstacles in front of it. The robot has no orientation; it can be thought of as a circle occupying an otherwise empty unit lattice square.

We imagine all motion to take place in the positive quadrant of a Cartesian coordinate system. It is convenient to view all obstacle blocks, and the robot, to be centered on integer coordinates. The robot starts in the square $s = (0, 0)$, and its goal is to reach the square $t = (a, b)$. The robot and all obstacles are confined to the rectangle $R = (s, t)$. Pushed blocks are obstructed by the sides of this rectangle. Note that there is a solution only if t is empty, that is, does not contain an obstacle block.

1.2 Problem Matrix

Our main interest was in solving the motion planning just described, where all obstacles are movable and there is no restriction on the robot's path. But at this writing we have not resolved the complexity of this problem. We therefore looked at several variations, summarized in the following table.

Obstacle Mobility	Path	
	Monotonic	Unrestricted
Some fixed	Polynomial	NP-complete
All movable	Polynomial	*Open*

In this table, a *monotonic* path for the robot is one that never takes a step leftwards or downwards, i.e., is monotonic with respect to both the positive x- and the y-directions. For monotonic paths, a dynamic programming algorithm can solve the problems in polynomial time.

When some blocks can be fixed to the plane, we prove the problem to be NP-complete. This theorem is very much related to (but apparently does not follow from) a result of Wilfong [Wil91].

*Authors' address: Department of Computer Science, Smith College, Northampton, MA 01063, USA. Supported by NSF grants CCR-882194 and CCR-9122169.

For the case where all obstacles are movable, we establish one necessary condition for the existence of a solution. We first describe this theorem.

2 Necessary Condition

Let $D(k)$ be the set of lattice points that are a distance at most k from the destination t, where distance is measured by the Manhattan or L_1 metric. Thus the points in $D(k)$ fall within the disk of radius k centered on t.

Theorem 2.1 *For the problem version where all blocks are movable, if at any time the robot stands on a square at distance k from t, then there is a path to t only if there are at least k empty squares in $D(k)$.*[2]

Thus if R is too cluttered with blocks huddled around the destination, there can be no solution. This necessary condition is unfortunately weak, in that many arrangements that satisfy it do not admit any solution.

3 Monotonic Paths

We define a Boolean array `Path[x,y]` to be true iff, if and when the robot enters (x, y) from $(x-1, y)$, it can reach the destination from there via a monotonic path. The value of `Path[x,y]` can be defined from the values of `Path[x+i,y+j]` for $0 < i \leq a - x$ and $0 < j \leq b - y$. Because the path is monotonic, the value at (x, y) depends only on the arrangement within the rectangle $((x, y), (a, b))$.

Consequently, `Path` can be computed via dynamic programming. Our naive algorithm requires $O(n^5)$ time, where n is the larger of a and b. We do not doubt that a better time complexity is possible, but the main point is that the problem is polynomial. Whether all blocks are movable, or some are fixed, is not an issue for this algorithm.

4 NP-Completeness

4.1 Wilfong's Theorem

Wilfong proved a certain motion planning problem with movable obstacles is NP-complete [Wil91]. His model is much broader than the special case we are considering. For example, he permits his robot to pull objects under certain circumstances. Pulling greatly increases the agility of the robot: if all obstacles are movable in our problem, there is a solution for any configuration of blocks as long as the robot has a small constant number of empty squares around it initially.[3] In addition, Wilfong's obstacles are not restricted to be unit squares, nor are they restricted to the integer lattice.

Despite these significant differences, his proof of NP-completeness only uses a small amount of the latitude he allows himself. In particular, he arranges that his robot can never pull an object, and that all obstacles and motions are orthogonal. So his result is closer to our case than it might initially appear.

There remain, we believe, two differences: Wilfong's proof uses four sizes of rectangular blocks, and uses L-shaped blocks. This of course falls outside of our domain, which only permits unit square blocks. Only having square blocks makes it more difficult to control the construction. One can view our contribution as a sharpening of Wilfong's result, proving intractability in an even more constrained environment.

4.2 Overall Design of Reduction

Our proof (like Wilfong's) is a reduction from 3SAT: we construct an arrangement of obstacles that can be traversed from s to t iff there is a satisfying truth assignment for a given instance of 3SAT. The overall design is as follows.[4]

The robot first enters a "variable component" that corresponds to setting the boolean value of the first variable, u_1. Here the robot will have the option of choosing one of two parallel paths. Once the choice is made, return is blocked by insertion of an "irreversible component," as shown in Fig. 1.[5] The parallel paths for u_1 visit each of the "clause junctions" corresponding to all the clauses in which the variable u_1 participates. The paths may need to be twisted over one another so that the negative literal can be closest to the clause. At the

[2] The square on which the robot stands is not counted among the empty squares.

[3] Seven squares seems to suffice, but we did not prove this formally.

[4] Our design is different from Wilfong's. We suspect that the choice of design is not significant.

[5] This is essentially the same mechanism as used by Wilfong (his Fig. 4).

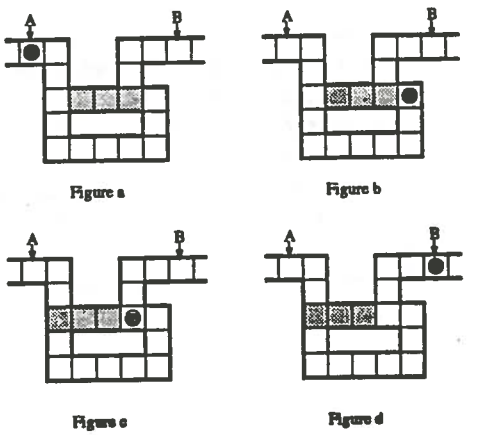

Figure 1: Irreversible component: after traversal $A \rightarrow B$, neither $B \rightarrow A$ nor $A \rightarrow B$ are possible.

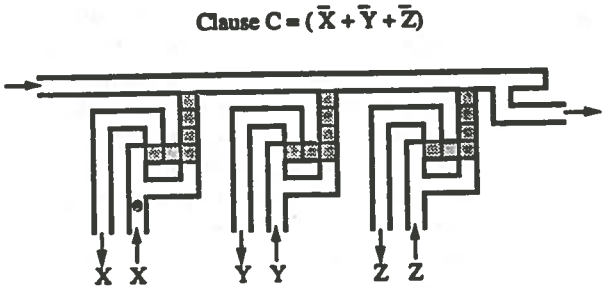

Figure 2: Clause junction.

clause, blocks are or are not deposited into a path that needs to be traversed later.

After visiting each relevant clause junction, the path returns and enters the u_2 variable component, and so on through the other variable components. After all variable components have been traversed, the robot then must pass through all the clause junctions to reach the destination, and it can do so only if at least one literal in the clause is true (as we will show below). The overall design is depicted in Fig. 3.

4.3 Clause Junction

The clause junction is designed so that each false literal will deposit one block into a corridor; see Fig. 2. Later traversal will be possible if 0, 1, or 2 blocks have been deposited, but not if 3 have been. Thus if the clause evaluates to false, the corridor is impassable and therefore the destination is unreachable. Conversely, if all clauses evaluate to true, the destination can be reached.

4.4 Crossovers

These are the most complex parts of the construction.[6] We found it necessary to arrange the crossovers between different variable paths[7] so that both the X and $\bar{X}$ paths (for some variable $u_i = X$) cross a path Y (for some literal $Y = u_j$ or $\bar{u}_j$) for $i < j$ in a three-way intersection. See Fig. 4. Call the three paths entering the intersection X_i, $\bar{X}_i$, and Y_i. They emerge in reverse order; call these paths Y_o, $\bar{X}_o$, and X_o.

Suppose the robot first travels down X_i (A in the figure); then we want to ensure that it can only exit along X_o (B in the figure). The delicate point is when the robot is at the triple intersection (G in the figure): we must prevent "leakage" into any path except X_o. Leakage along $\bar{X}_o$ (to D in the figure) is not possible because traversal of X_i deposits a block in $\bar{X}_o$ that clogs it. Leakage along Y_o (to F in the figure) is not possible because one of the paths X_o or $\bar{X}_o$ must be traversed first to "unlock" Y_o. And similarly all other potential leakages are impossible.

5 Open Problem

Whether the problem with all obstacles movable is polynomial or NP-complete is open.

References

[Wil91] G. Wilfong. Motion planning in the presence of movable obstacles. *Annals of Mathematics and Artificial Intelligence*, 3:131–150, 1991.

[6] And incidentally it is here that Wilfong used L-shaped blocks. His crossovers are much simpler.

[7] Crossovers between u_i and $\bar{u}_i$ are much easier and not discussed here.

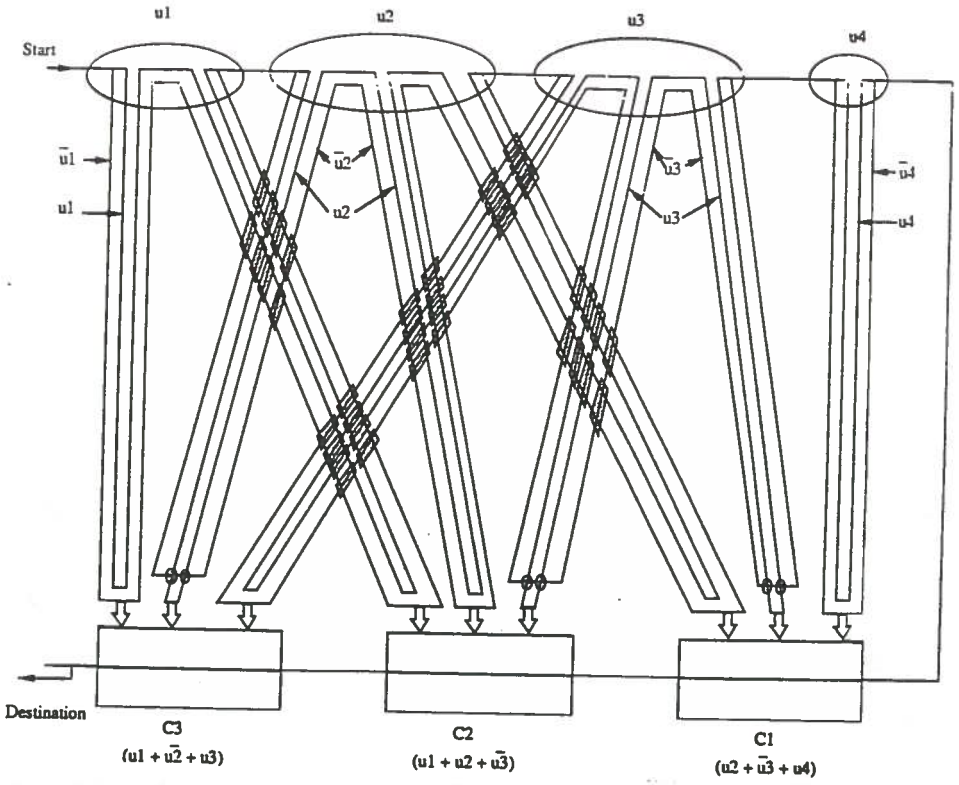

Figure 3: Overall design for $(u_1 + \bar{u}_2 + u_3)(u_1 + u_2 + \bar{u}_3)(u_2 + \bar{u}_3 + u_4)$.

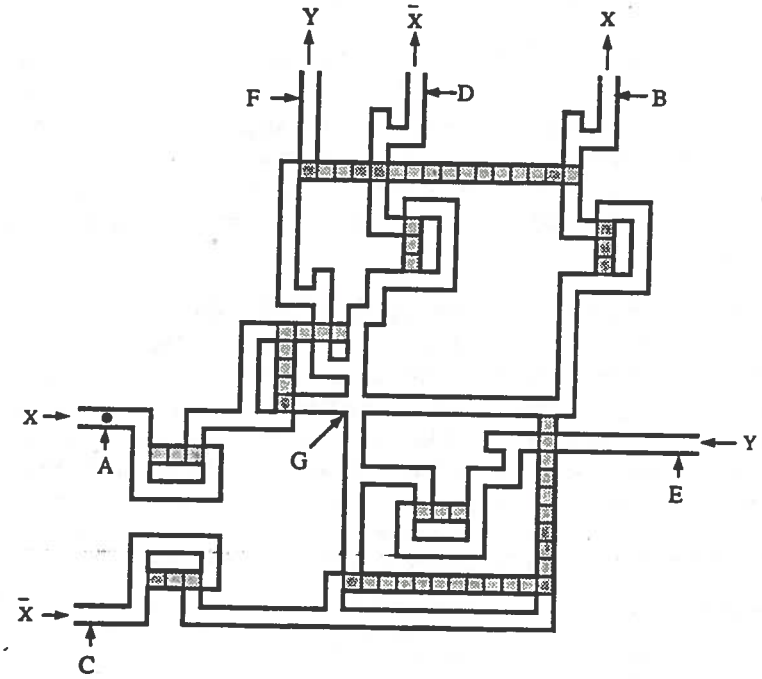

Figure 4: Crossover.