Motion Planning Amidst Movable Square Blocks: Push-* is NP-hard

Michael Hoffmann
Institute for Theoretical Computer Science
ETH Zurich
CH-8092 Zurich, Switzerland
hoffmann@inf.ethz.ch

June 26, 2000

Abstract

We show that a particular pushing-block puzzle is NP-hard, settling an open problem posed by O'Rourke et al [2, 4]. The puzzle consists of unit square blocks on an integer lattice. The robot may move horizontally and vertically in order to reach its goal position. Thereby it can push an arbitrary number of blocks in sequence as long as there is at least one free square ahead. The proof is by reduction from 3-SAT.

1 Problem Definition

For $m, n \in N$ consider a rectangular $n \times m$-grid where each position $(x, y) \in \{1, \ldots, m\} \times \{1, \ldots, n\}$ is either free (F) or blocked (B) and a robot moving on this grid. The robot can move horizontally and vertically and thereby push an arbitrary number of blocks in front of it; see Figure 1 where the blocked positions are shaded and the robot is shown as circle.

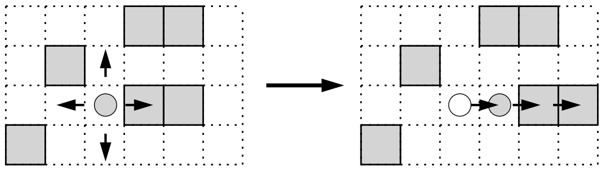

Figure 1: Pushing Blocks.

More formally, the robot may move from (x, y) to (x', y') under the conditions indicated in Table 1.

If $(x', y') = \mathtt{B}$, the robot pushes the parts blocking its way, i.e. the grid is changed such that $(x', y') \leftarrow \mathtt{F}$ and $(\xi, \upsilon) \leftarrow \mathtt{B}$, where (ξ, υ) is the free position ahead existing by the condition in Table 1.

The problem is to decide whether there exists a sequence of moves starting at $(1, 1)$ ending in (m, n). (Assume, $(1, 1) = \mathtt{F}$.) We will show below that this problem – we call it Push-* – is NP-hard by giving a reduction from 3-SAT.

(x', y')	*condition*
$(x-1, y)$	$\exists\, z \,:\, 1 \le z \le x-1 \;\wedge\; (z, y) = \mathtt{F}$
$(x+1, y)$	$\exists\, z \,:\, x+1 \le z \le m \;\wedge\; (z, y) = \mathtt{F}$
$(x, y-1)$	$\exists\, z \,:\, 1 \le z \le y-1 \;\wedge\; (x, z) = \mathtt{F}$
$(x, y+1)$	$\exists\, z \,:\, y+1 \le z \le m \;\wedge\; (x, z) = \mathtt{F}$

Table 1: Robot Movement.

2 The Reduction

In the 3-SAT problem, we are given a boolean formula in conjunctive normal form where each clause consists of exactly three distinct literals. Let $X := \{x_1, \ldots, x_k\}$ be the set of variables and $C_1, \ldots, C_\ell$ the clauses. For $i \in \{1, \ldots, k\}$ define

$$n_i := \left|\left\{r \in \{1, \ldots, \ell\} \mid x_i \text{ occurs in } C_r\right\}\right|$$

to be the number of occurrences of x_i in the formula. Analogously, let $\overline{n_i}$ be the number of occurrences of $\overline{x_i}$. Without loss of generality, we assume $n_i \ge \overline{n_i}$ for all $i \in \{1, \ldots, k\}$.

We will construct an instance of the described pushing-block puzzle such that the puzzle is solvable if and only if there is a satisfying assignment for the given formula. Refer to Figure 7 at the end of the paper for a complete example.

The construction consists of four major blocks, as depicted in Figure 2. All positions outside these blocks are blocked initially.

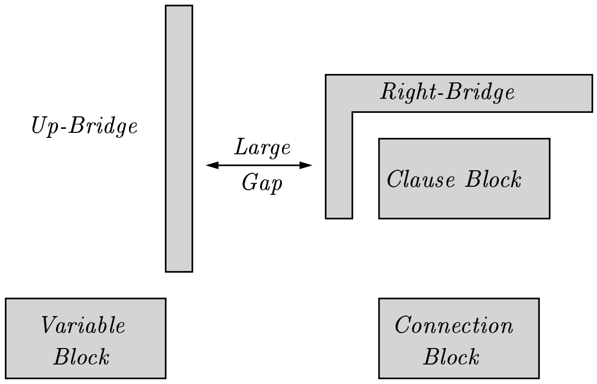

Figure 2: Construction: A Schematic View.

Variable Block The start position is here and the assignment is implicitly chosen when passing this block. Described in section 3.

Clause Block The goal position is here. It can be reached if and only if the assignment satisfies all clauses. Described in section 5.

Connection Block connects (logically) variable and clause block. Described below.

Bridge Blocks connect (in terms of movement) variable and clause block. Described in section 4.

For simplicity of exposition, we use the rightmost position in the clause block instead of the top right corner of the whole grid as goal position. It will be clear how to extend the scheme to satisfy this condition by adding a sufficiently large empty block above-right our construction.

There will be a row associated to each literal from $X \cup \overline{X}$, where $\overline{X} := \{\overline{x_1}, \ldots, \overline{x_k}\}$, and three columns associated to each clause. In the connection block – that will not actually be passed by the robot – there is one free position in any clause column, all other positions are blocked. These free positions correspond to the literals the clause consists of and are, hence, in the rows associated to these literals. See Figure 3 for an example. Since most positions on our grid will be blocked, we rather mark free positions block-wise as polygons, so e.g. in Figure 3 there are three free positions.

Observation 1 *In the connection block there are n_i free positions in the row associated to x_i and $\overline{n_i}$ free positions in the row associated to $\overline{x_i}$ for $1 \leq i \leq k$.*

3 Variable Gadgets

Figure 4 shows the construction of a variable gadget. For each variable x_i there is one such gadget in the variable block, representing both two corresponding literals. The bottom two rows are associated to x_i and $\overline{x_i}$, respectively, as described in the previous section. The gadgets are placed diagonally on top of each other, from the bottom left corner up to the top right corner of the variable block, such that the goal position of the previous is directly below the start position of the next gadget. The gadgets are to be passed one after another and accordingly, the start and goal positions (denoted by a small circle and cross, respectively) of each gadget also are in the mentioned corners. The gadget's size depends on n_i: the height is 4 and the width is $2n_i + 2$.

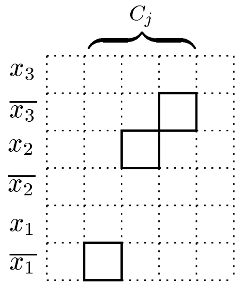

Figure 3: Connections for $C_j = \overline{x_1} \vee x_2 \vee \overline{x_3}$.

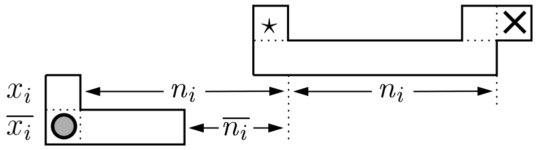

Figure 4: Variable Gadget.

Lemma 1

1. *The variable gadget for x_i can be passed, if and only if at least one of the following two statements is true.*
   - *the robot pushes n_i times right in row x_i or*
   - *the robot pushes $\overline{n_i}$ times right in row $\overline{x_i}$.*
2. *The positions outside the gadget cannot be changed, except for the $n_i + \overline{n_i}$ free positions in the connection block corresponding to x_i and $\overline{x_i}$.*
3. *The only way to leave the gadget (its bounding box) is through its start or goal position.*

Proof. The *if*-part of 1 is obvious. For the *only-if*-part note that initially, the only pushing positions are in the x_i and (possibly) $\overline{x_i}$ row where the robot can push right. This does not change, until the column of $\star$ is reached. Moreover, there are no free positions vertically outside the gadget, except for (possibly) below the start and above the goal position. Hence, 1 is true for the first gadget. For the following gadgets, we can argue analogously, as soon as we have proved 2. There are no free positions horizontally outside the gadget, except for the $n_i + \overline{n_i}$ free positions in the connection block to the right. This number is exactly the difference in coordinates between the rightmost free position in the free starting component and the right border of the gadget. Thus, the robot cannot pass the gadget's left or right border. The only way left to change positions outside the gadget is to push out some blocks from the start or goal position which is easily seen to be impossible. Hence, 2 and 3 follow. □

4 Bridge Gadgets

There is only one such gadget, but it consists of three components placed within large horizontal distance. Therefore it has been divided into two blocks in the schematic view of Figure 2. Figure 5 shows the gadget as a whole where the left part corresponds to the Up-Bridge and the two parts on the right form the Right-Bridge.

Let $h := 3\,\ell - 1$ denote the height of the clause block and $x := 10\,\ell - 1$ be the number of free positions in the clause block including the $h + 1$ free positions in the lower Right-Bridge component. (To check the numbers, refer to section 5. Recall that ℓ denotes the number of clauses.)

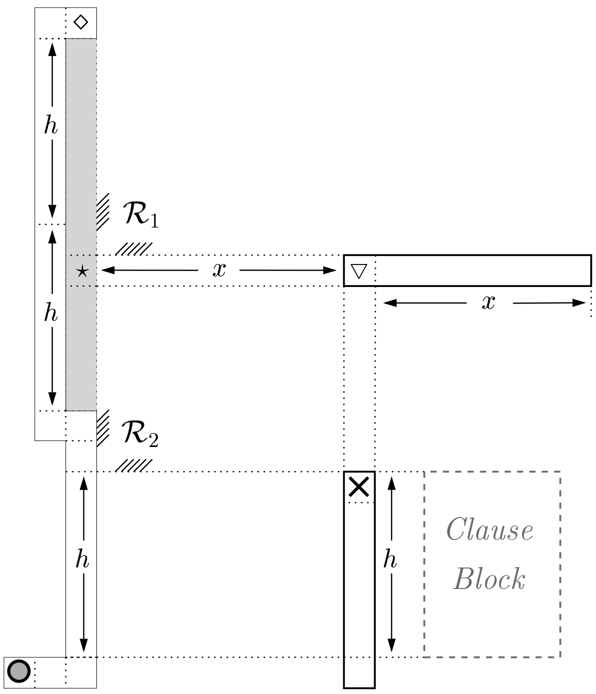

Figure 5: Bridge Gadget.

Lemma 2 *After passing the bridge gadget, there is no free position in the rows and columns of the clause block outside the clause block, except for*

- *those left free in the connection block and*
- *the goal position of the bridge block.*

Proof. There is a straightforward way for the robot to pass the bridge gadget: on the free positions up to $\diamond$, then pushing down to $\star$, right to $\bigtriangledown$ and finally down to the goal position. We will show in the following that this is basically the only way to pass the gadget.

There are two sets of rows where the robot can go (and push) right in order to cross the gap: the row of $\star \to \bigtriangledown$ and the rows of the clause block. But in between these two sets, there is no vertical connection. Since all the positions in the gap area are blocked initially, there is no way to move to any of them without pushing or having been there before.

By the choice of x, there is no way to reach the goal column without using the free positions in the upper right component. Moreover, by the above reasoning, a combination of using some of the free positions from the row of $\star$ and some of those in the clause block to push right will not help. Thus, the row $\star \to \bigtriangledown$ is the only way to cross the gap.

But even in that row, we have to get one more free position between $\star$ and $\bigtriangledown$ in order to reach the right components. However, the region $\mathcal{R}_1$ above-right the blocked part of that row between the bridge components is completely blocked. Hence, there is no way to push any of the blocks out there, except for the first one, at position $\star$. The robot must not push this block right, since then it would be impossible to remove the block from the row for the reason stated above. Hence, there is no way for the robot to enter the region $\mathcal{R}_2$ to the right of the left component and above the clause block, before the block at $\star$ has been removed from its row.

There is no way to push away any of the shaded blocks below $\star$ and pushing them upwards does not help, since $\star$ would still be blocked. The only way to clear $\star$ is to push the shaded blocks down, completely blocking this column in the rows to the right of the clause block. Obviously, the robot has to go down $h+1$ times from $\bigtriangledown$, blocking the positions to the right of the clause block in this column as well, except for the goal position. Also the part above the clause block is blocked from going right to $\bigtriangledown$. The only possibility that is left to have free positions in some rows to the left of the clause block is, if some blocks have been pushed into the clause block from the column of $\star$. But then at least one position in the column below $\bigtriangledown$ would be blocked and it would be impossible to reach the goal position. □

5 Clause Gadgets

There is one such gadget, depicted in Figure 6, for every clause. They are aligned diagonally, from the top left down to the bottom right corner of the clause block, such that the goal position of the previous is directly above the start position of the next gadget. In the last gadget, we save one free position by moving the goal position one row up. Thus, the total number of free positions in the clause block is $7\ell - 1$ and its height (in terms of rows) is $3\ell - 1$.

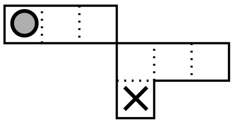

Figure 6: Clause Gadget.

Lemma 3 *The robot can pass a clause gadget, if and only if it pushes down in one of the first three columns.*

Proof. The if part is obvious. For the only if, consider the first (leftmost) clause gadget. Due to Lemma 2, there are no free positions above and to the left, except for the position immediately to the left of the start position. Clearly, there are no free positions to the right either. Hence, the claim is true for the first gadget. For the following gadgets we can argue analogously, noting that the above observation is true here as well: even if all of the first three columns have a free position in the connection block (there is at most one free position per column by construction, see section 2) and the robot uses all of them to push down, the bottommost position it gets to is the one diagonally left below the goal position and hence, immediately to the left of the starting position of the next gadget (if there is any). □

6 Putting Things Together

In the connection block, for any $i \in \{1, \ldots, k\}$ either the free positions in the row associated to x_i or those in the row associated to $\overline{x_i}$ get blocked when passing the variable block, as stated in Lemma 1. On the other hand, by Lemma 3 the gadget of a clause can be passed only if for one of the three literals the clause consists of there is a free position in the connection block. Thus, if the robot found its way through the puzzle, we can construct a satisfying assignment for the 3-Sat formula by setting those literals to false that had no free positions in the connection block after the robot passed the variable block. It might happen that for some i this is true for both x_i and $\overline{x_i}$, but then it does not matter how we set x_i and we can choose arbitrarily.

On the other hand, if we have a satisfying assignment, by Lemma 1 the robot can pass the variable block by pushing only those rows associated to literals that are assigned false. Then the robot will find its way to the goal position as described in Lemma 2 and 3. Together, we have proved the following theorem.

Theorem 4 *The constructed puzzle is solvable, if and only if the 3-Sat formula has a satisfying assignment.*

From here, NP-hardness of the puzzle follows immediately, noting that its size is polynomial, in fact at most quadratic, in the number of variables and clauses of the 3-Sat formula.

7 Conclusions

We have shown NP-hardness of a particular pushing-block puzzle, a question that has been stated several times [2, 4]. It remains open, whether the problem is in NP, since it is not clear, whether there always exists a solution path of polynomial length. The problem might even be PSPACE complete, like the variant where some of the blocks can be tied to the board [1].

If the constructed puzzle is solvable, it is possible to find a solution path that is x-monotone except for one backward step and consists of two y-monotone parts (up-down). Obviously, the problem is in NP and thus NP-complete, if the robot is restricted to such paths only. On the other hand, the problem is known to be in P, if the robot is restricted to monotonic paths only [4]. So the border is somewhere in between these two.

Also note that if the constructed puzzle is solvable, it is possible to find a push-push solution path, i.e. whenever the robot pushes, the pushed blocks slide the maximal extent as without friction until there is no free position ahead anymore. Thus, our reduction extends the results of [3, 5] to more than one pushing blocks, a problem that one might call Push-Push-∗.

References

[1] Bremner, D., O'Rourke, J., and Shermer, T. Motion planning amidst movable square blocks is PSPACE complete. Draft, 1994.

[2] Demaine, E. D., and O'Rourke, J. Open problems from CCCG'99. Technical Report 066, Dept. Comput. Sci., Smith College, Northampton, MA, Mar. 2000.

[3] Demaine, E. D., and O'Rourke, J. PushPush is NP-hard in 2D. Technical Report 065, Dept. Comput. Sci., Smith College, Northampton, MA, Jan. 2000.

[4] DHAGAT, A., AND O'ROURKE, J. Motion planning amidst movable square blocks. In *Proc. 4th Canad. Conf. Comput. Geom.* (1992), pp. 188–191.

[5] O'ROURKE, J., AND GROUP, T. S. P. S. Push-Push is NP-hard in 3D. Technical Report 064, Dept. Comput. Sci., Smith College, Northampton, MA, Nov. 1999.

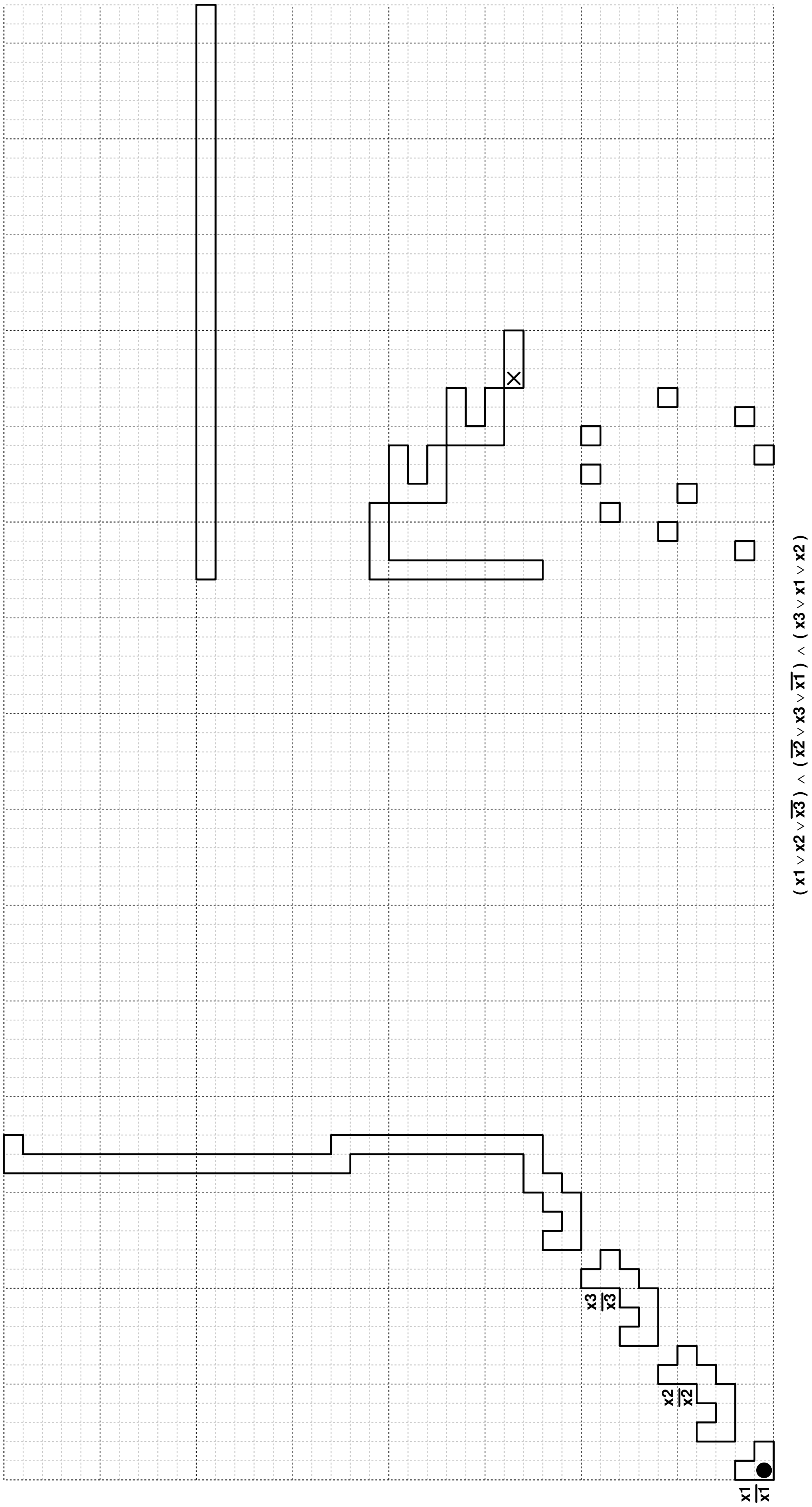

Figure 7: A Complete Example.