PUSH-2-F is PSPACE-Complete

Erik D. Demaine* Robert A. Hearn† Michael Hoffmann‡

Abstract

We prove PSPACE-completeness of a class of pushing-block puzzles similar to the classic Sokoban, extending several previous results [1, 5, 12]. The puzzles consist of unit square blocks on an integer lattice; some of the blocks are movable. The robot may move horizontally and vertically in order to reach a specified goal position. The puzzle variants differ in the number of blocks that the robot can push at once, ranging from just one (PUSH-1-F) up to arbitrarily many (PUSH-*-F). We prove that PUSH-k-F and PUSH-*-F are PSPACE-complete for $k \geq 2$ using a reduction from Nondeterministic Constraint Logic (NCL) [8].

1 Introduction

Algorithmic motion planning is a large area of computational geometry with applications in robotics, assembly planning, and computer animation; see, e.g., [11] for a survey. The standard type of problem involves moving a robot from one configuration to another while avoiding fixed obstacles. A recent direction introduced by Wilfong [12] is a class of problems in which robots are permitted to move some of the obstacles in order to increase maneuverability. As robots become more powerful at manipulation, an understanding of such models becomes increasingly important. Current-day applications include automated warehouse control and warehouse navigation; see, e.g., [7]. A representative abstraction of such applications is the popular Sokoban puzzle [2, 6], which is known to be PSPACE-complete [2].

Problems. Our hardness results are particularly surprising given the simplicity of the model of motion and obstacle manipulation. Consider a rectangular $n \times m$ grid in which each square is marked either *free* or *blocked*. A *robot* can move horizontally and vertically in the grid, and thereby push up to k blocks in front of it, for some constant k. See Figure 1, in which the blocked positions are shaded and the robot is shown as a circle, pushing two blocks.

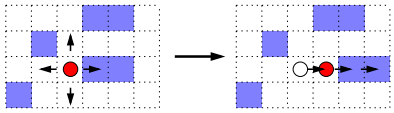


Figure 1: Example of pushing blocks.

The PUSH-k problem [3, 4] is to decide whether there is a sequence of moves starting at a specified free position and ending at a specified goal position. If we omit the restriction on how many blocks the robot can push at once (i.e., $k = \infty$), we obtain the problem PUSH-* [1, 5, 9]. In this paper, we study the PUSH-F model where some of the blocks are fixed to the board, making them unpushable.

Related Work. Out of these many problem variations, several individual cases have been studied. The original paper by Wilfong [12] studies a more flexible model in which the blocks can be more general than squares, and the robot can both push and pull blocks. Dhagat and O'Rourke [5] initiated the PUSH- line of models, and proved that PUSH-*-F is NP-hard. This result was later strengthened to PSPACE-completeness [1].

The PUSH-PUSH model [3] requires that, once a block is pushed, it slides the maximal extent in that direction. This model can be thought of representing either sliding blocks on a frictionless surface, or the situation in which blocks cannot be pushed by precise amounts but can be consistently pushed against other blocks.

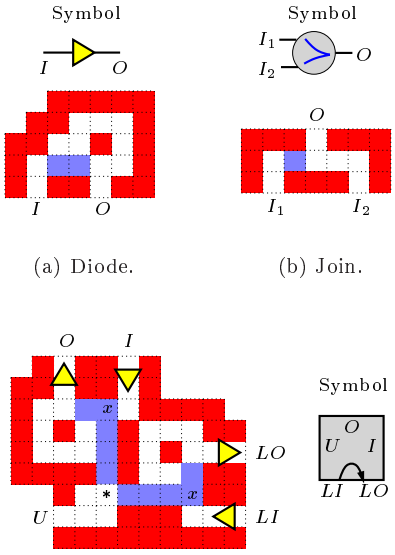
Over the past few years, there have been several results regarding the restricted model in which all blocks are movable. NP-hardness was established for PUSH-PUSH-1 [3], PUSH-1 [3], PUSH-k [4], and PUSH-* [9]. The latter two constructions even extend to the so-called PUSH-X model in which the robot is not allowed to cross its own path, which immediately places the corresponding problems in NP.

Overview. This paper reproves and strengthens the results of [1] to robots with possibly limited strength, establishing PSPACE-completeness of PUSH-*-F and PUSH-k-F for $k \geq 2$. More generally, we prove that any model of pushing-block puzzles is PSPACE-complete if one can build a set of three gadgets with a certain functionality. In particular, we describe such gadgets for PUSH-3-F in Section 2. These gadgets also cover PUSH-k-F for $k > 3$ and PUSH-*-F. The gadgets for PUSH-2-F are more complicated, and briefly sketched

*MIT Laboratory for Computer Science, 200 Technology Square, Cambridge, MA 02139, USA, edemaine@mit.edu.

†MIT Artificial Intelligence Laboratory, 200 Technology Square, Cambridge, MA 02139, USA, rah@ai.mit.edu.

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



(a) Diode. (b) Join.

(c) Lock. Triangles denote diodes.

Figure 2: PUSH-3-F gadgets.

in Section 4. Our reduction is from Nondeterministic Constraint Logic (NCL) [8], described in Section 3.

2 PUSH-3-F

Observation 1 *The diode shown in Figure 2(a) can be traversed arbitrarily often from I to O, but never the other way round.*

Observation 2 *The join shown in Figure 2(b) can be traversed from both I_1 and I_2 to O arbitrarily often, but never from I_1 to I_2 or vice versa. Furthermore, the robot can always go from O to the most recently entered input (I_1 or I_2).*

Lemma 1 *Any sequence of traversals for the lock gadget in Figure 2(c) consists of only the following three "atomic" traversals: $LI \to LO$, $I \to O$, and $U \to U$. Moreover, $LI \to LO$ and $I \to O$ cannot occur next to each other in such a sequence, i.e. are always separated by an $U \to U$ traversal.*

Proof. During both, $LI \to LO$ and $I \to O$, position $*$ is blocked, rendering the respective other traversal impossible. Similar to the diode, the positions marked with x must be reblocked in order to exit. The only way to free $*$ is from U; because both x positions are blocked, the three blocks can be pushed back only one step, leading again to the initial configuration. □

We refer to the $LI \to LO$ traversal as "locking the gadget", while $I \to O$ is called "passage", and $U \to U$ "unlock". Remember that, somewhat counterintuitively, the gadget must be unlocked before it can be relocked after passage. Similarly, the state of a lock gadget is called *unlocked* if $I \to O$ traversal is possible, and *locked* otherwise.

From six copies of the lock gadget, some of which are reflected, we can build a unidirectional crossing.

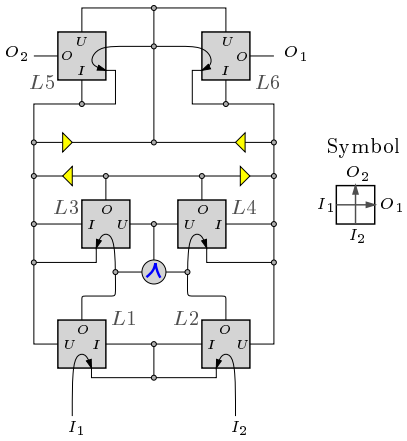


Figure 3: A unidirectional crossing.

Lemma 2 *The gadget shown in Figure 3 can be traversed from I_1 to O_1 and from I_2 to O_2 arbitrarily often. Moreover, these are the only traversals possible.*

Proof. Consider that the robot enters from I_1; the other case is symmetric. First, it has to lock $L1$, so it can then only pass $L2$. Next, both $L3$ and $L4$ can be unlocked (if they are not already), but then $L4$ has to be locked again. Then the robot is on the right corridor from where it can unlock $L2$. Because $L4$ is locked, there are only two ways to proceed: pass $L6$ to reach O_1 or pass the diode to unlock both $L5$ and $L6$, lock $L5$ (at this point, locking $L6$ is another option, but just leads the robot where it was before passing the diode), unlock $L1$, pass $L3$, go back to the right wire and finally reach O_1 after passing $L6$. Because $L4$ is locked, the only way to reach the left wire is by locking L_5; hence, the robot cannot reach O_2. □

A fully bidirectional crossing can be built from four unidirectional crossings, as indicated in Figure 4 (Fig. 12 of [3]).

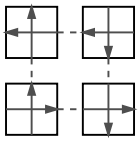


Figure 4: Bidirectional crossing.

3 Nondeterministic Constraint Logic

We show that PUSH-k-F is PSPACE-hard by a reduction from Nondeterministic Constraint Logic (NCL) [8].

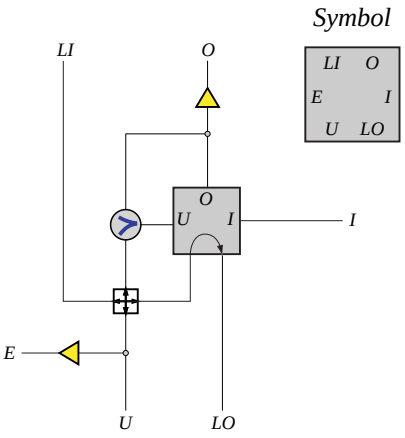


Figure 5: Buffered lock.

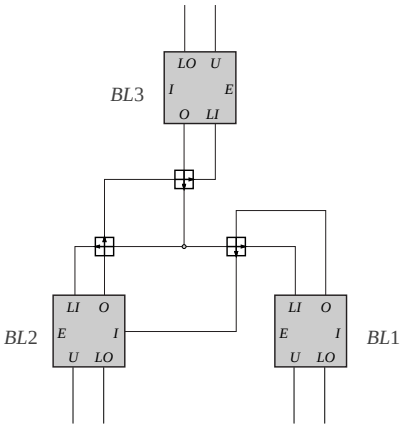


Figure 6: NCL AND vertex.

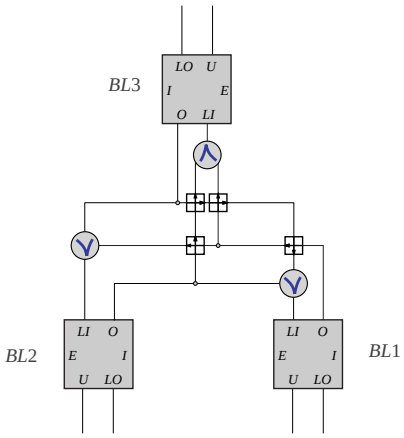


Figure 7: NCL OR vertex.

Unattached terminals are connected to free space, which can lead to any other unattached terminal.

This reduction involves constructing "NCL vertex gadgets" out of our previously constructed gadgets.

An NCL "machine" is specified by a *constraint graph*: an undirected graph together with an assignment of weights from $\{1, 2\}$ to each edge. A configuration of this machine is an orientation (direction) of the edges such that the sum of incoming edge weights at each vertex is at least 2. A move is made by reversing a single edge such that the configuration remains valid. The standard decision question from a particular NCL machine and configuration is whether a specified edge can be eventually reversed by a sequence of moves. This problem is PSPACE-complete [8].

In fact, only two types of vertices are necessary for PSPACE-completeness to hold: those with incident edge weights of 1-1-2 (AND) and 2-2-2 (OR). These vertex types have properties similar to the logical operations of the same name. For example, for the weight-2 edge to be directed away from an AND vertex, both of the weight-1 edges must be directed inward.

We build NCL vertex gadgets out of buffered locks. A buffered lock (Figure 5) has the same properties as a lock, except that it may be unlocked during an $I \to O$ traversal, it may be unlocked by a $U \to E$ traversal, and the arrangement of terminals is different.

Each vertex gadget (Figures 6 and 7) is made of three buffered locks, plus associated circuitry to enforce the necessary constraints. Each buffered lock acts as half of an edge: locked corresponds to "directed outward," and unlocked corresponds to "directed inward." The unattached buffered-lock terminals (E, some I's) are open to free space, which can reach any other such terminal, via appropriately placed crossing gadgets.

In each vertex, we assume that the lock states initially satisfy the vertex constraints. Then any possible robot traversal maintains those constraints.

Lemma 3 *The gadget shown in Figure 6 satisfies the same constraints as an NCL* AND *vertex.*

Proof. To lock $BL3$, both $BL1$ and $BL2$ must be unlocked: the robot may then traverse $I(BL1) \to LO(BL3)$, passing through $BL2$. To lock either $BL1$ or $BL2$, $BL3$ must be unlocked: the robot may then traverse either $I(BL3) \to LO(BL1)$ or $I(BL3) \to LO(BL2)$. □

Lemma 4 *The gadget shown in Figure 7 satisfies the same constraints as an NCL* OR *vertex.*

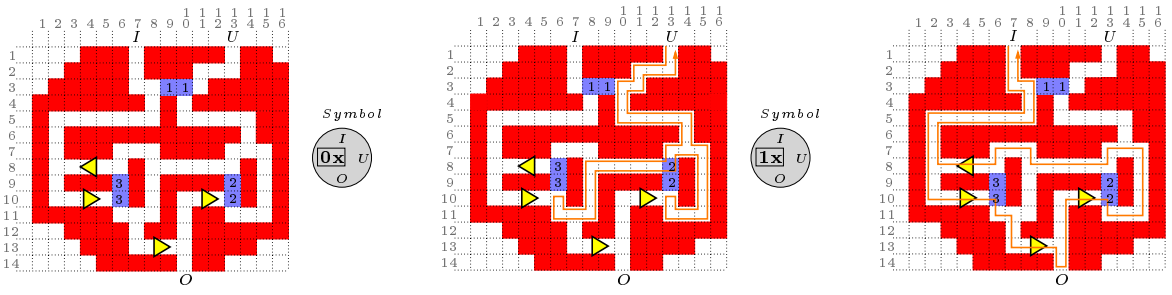
Proof. Any buffered lock may be locked if and only if any other lock is unlocked. For example, if $BL1$ is unlocked, the robot may traverse $I(BL1) \to LO(BL3)$; the other cases are symmetric. The join gadgets ensure that only the appropriate paths may be taken. □

Constraint Graphs. Vertices are connected together into arbitrary NCL constraint graphs by connecting the buffered locks in pairs, matching LO terminals to U terminals. Then any buffered lock can be unlocked precisely when its connecting buffered lock is locked. This property represents that a half-edge can be directed inwards precisely when the other half is directed outwards.

For example, suppose that $BL1$ and $BL2$ in an AND vertex are unlocked. Then the robot may traverse $I(BL1) \to LO(BL3)$, and continue on to traverse $U \to E$ in the adjoining buffered lock.

Theorem 5 PUSH-k-F *and* PUSH-$*$-F *are PSPACE-complete for* $k \geq 2$.

Proof. Reduction from NCL, by the construction described. A given NCL constraint graph may be represented as a PUSH-k-F configuration. The target edge



(a) (b) (c)

Figure 8: Traversals of the one-time passage gadget: (a) initial state with no traversals remaining, (b) unlock to enable one travel, and (c) passage.

in the NCL graph may be eventually reversed if and only if the robot may reach the unlock terminal of a corresponding buffered lock.

PUSH-2-F is in PSPACE: a simple nondeterministic algorithm traverses the state space, maintaining only the current state, so PUSH-2-F is in NPSPACE, and Savitch's Theorem [10] says that NPSPACE = PSPACE. □

4 A Lock for PUSH-2-F

While the diode and join gadget from Section 2 work for PUSH-2-F as well, the lock in Figure 2(c) relies on the robot pushing three blocks at once. Because our lock gadget for PUSH-2-F is pretty involved, we cannot fully describe it within the scope of this abstract. Instead, we just show the canonical traversal sequence (Figure 9) to give an impression on how it looks. Note that the figures are a mixture of actual block puzzle and symbolic notation (e.g., diodes). Also, the "One-time-passage" gadget shown in Figure 8 is a subcomponent of the lock. It allows the robot to pass once from I to O and back to I; after such a traversal, it must be reset during a $U \to U$ traversal.

Lemma 6 *Consider the gadget shown in Figure 8 and assume that port O is a "dead end", i.e. there is no way to reach I or U (nor the overall goal position) from there except for re-traversing the gadget.*

Then the only possible traversals of the gadget are $I \to O \to I$ and $U \to U$ (without reaching O in between). The robot can traverse the gadget in these ways arbitrarily often, with the following restriction: there cannot be two consecutive $I \to O \to I$ traversals; i.e. between any two $I \to O \to I$ traversals, there has to be a $U \to U$ traversal.

Proof. First note that none of the three pushing block pairs can be separated, nor can any of these blocks pushed out of its initial row (pair 1) or column (pair 2 and 3). Hence, for example, pair 1 always blocks either field (8/3) or field (10/3), effectively preventing traversal between I and U.

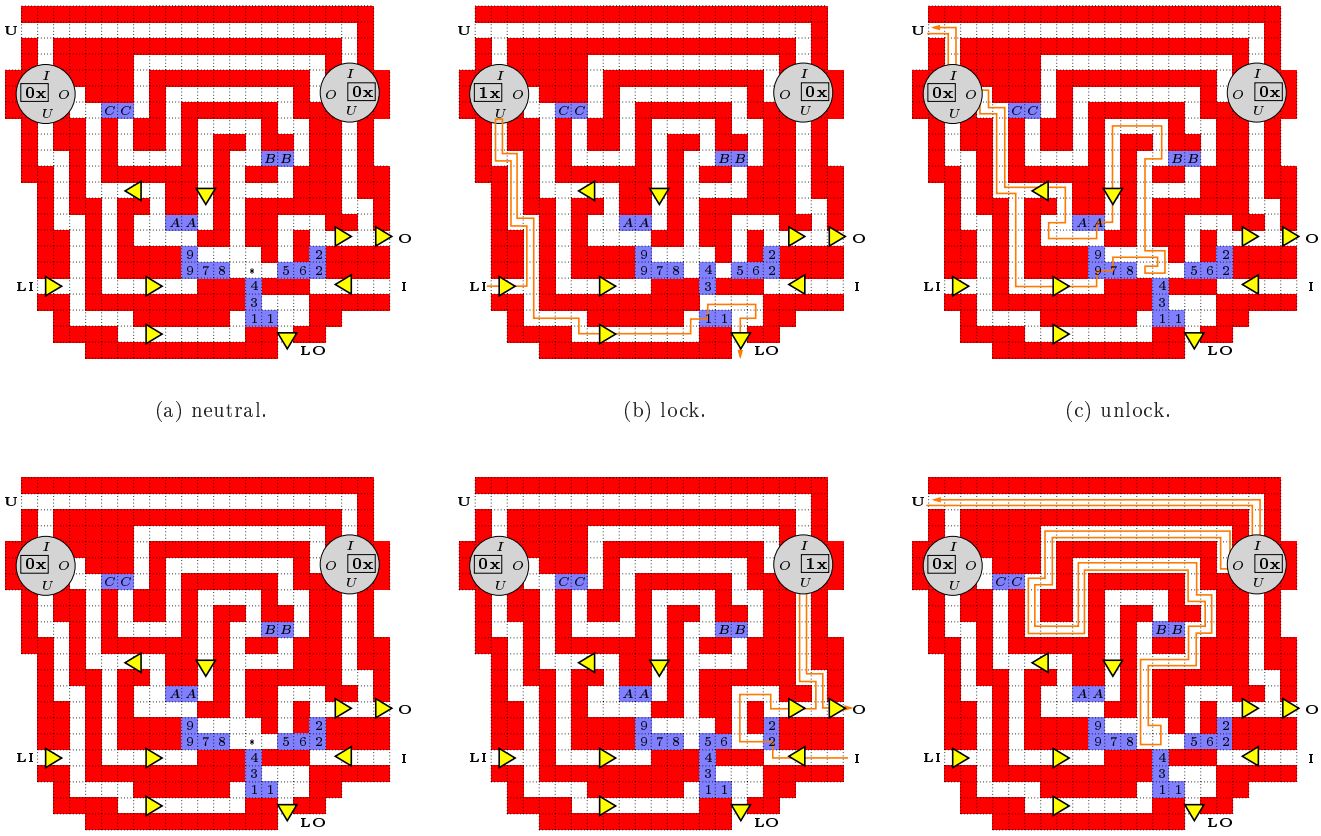
Consider the robot entering at U. If it leaves the cycle of pair 2 (13–15/7–11) through row 8, field (13/10) is blocked. Thus, after passing the diodes and reaching O, there is no way back to U (nor to I). Since traversal from O is not possible by assumption, the only possible traversals remaining are $I(\to O) \to I$ and $U \to U$, as claimed.

While $I \to O$ traversal is not possible initially, the robot can unlock both block pair 2 and 3 by pushing them up during a $U \to U$ traversal as shown in Figure 8(b). Thereafter, a single $I \to O \to I$ traversal (see Figure 8(c)) is possible, necessarily restoring both block pairs to their original position. □

The basic idea for the lock gadget is the same is in the PUSH-3-F-lock described in Section 2: there is a central position (marked with a star in Figure 9(a)) that is needed to push a sequence of blocks into during two different traversals, $I \to O$ and $LI \to LO$. But now these sequences of blocks consist of only two blocks instead of three, such that separating both traversal corridors becomes pretty tricky. If we would just have a gadget similar to the PUSH-3-F-lock, consisting of only blocks one through six, the robot could, e.g., push up block 3 and 4 twice during a $LI \to LO$ traversal, and in a subsequent unlock traversal from U block 3 could be pushed right, and suddenly LO would become reachable from U. Block 7 and 8 have been thrown in to avoid this, and block pairs 9 and A in turn are there to keep these blocks in place. The proof that this gadget is indeed a lock as required for the crossing gadget described in Section 2 consists of a pretty tedious case analysis that is contained in the full paper.

References

[1] BREMNER, D., O'ROURKE, J., AND SHERMER, T.



(a) neutral. (b) lock. (c) unlock.

(d) neutral. (e) traverse. (f) unlock.

Figure 9: Traversals of the lock gadget.

Motion planning amidst movable square blocks is PSPACE complete. Draft, 1994.

[2] Culberson, J. Sokoban is PSPACE-complete. In *Proc. Internat. Conf. Fun with Algorithms* (Elba, Italy, June 1998), Carelton Scientific, pp. 65–76.

[3] Demaine, E. D., Demaine, M. L., and O'Rourke, J. PushPush and Push-1 are NP-hard in 2D. In *Proc. 12th Canad. Conf. Comput. Geom.* (2000), 211–219.

[4] Demaine, E. D., and Hoffmann, M. Pushing blocks is NP-complete for noncrossing solution paths. In *Proc. 13th Canad. Conf. Comput. Geom.* (2001), pp. 65–68.

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

[6] Dor, D., and Zwick, U. Sokoban and other motion planning problems. *Computational Geometry: Theory and Applications 13*, 4 (1999), 215–228.

[7] Everett, H. R., and Gage, D. W. From laboratory to warehouse: Security robots meet the real world. *Internat. J. Robotics Research 18*, 7 (July 1999), 760–768.

[8] Hearn, R. A., and Demaine, E. D. The Nondeterministic Constraint Logic model of computation: Reductions and applications. In *Proc. 29th Int. Colloq. Automata, Languages, and Programming* (2002).

[9] Hoffmann, M. Push-* is NP-hard. In *Proc. 12th Canad. Conf. Comput. Geom.* (2000), pp. 205–210.

[10] Savitch, W. J. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. System Sci. 4*, 2 (1970), 177–192.

[11] Sharir, M. Algorithmic motion planning. In *Handbook of Discrete and Computational Geometry*, J. E. Goodman and J. O'Rourke, Eds. CRC Press LLC, Boca Raton, FL, 1997, ch. 40, pp. 733–754.

[12] Wilfong, G. Motion planning in the presence of movable obstacles. *Ann. Math. Artif. Intell. 3* (1991), 131–150.