University of Alberta

Sokoban is PSPACE-complete

by

Joseph C. Culberson

Technical Report TR 97-02
April 1997

DEPARTMENT OF COMPUTING SCIENCE
The University of Alberta
Edmonton, Alberta, Canada

Sokoban is PSPACE-complete

Joseph C. Culberson
University of Alberta

April 4, 1997

Abstract

It is shown that the popular puzzle Sokoban can be used to emulate a linear bounded automata (finite tape Turing Machine (TM)). In particular, a construction is given that has a solution if and only if the corresponding Turing Machine on its input halts in the accept state. Further, if the TM halts and accepts, then the pusher will make $\Theta(n + t(n))$ moves and pushes, where n is the number of symbols on the input tape, and $t(n)$ is the number of transitions made by the TM during its computation. This construction shows that the puzzles are PSPACE-complete, solving the open problem stated in [1].

1 Introduction

Sokoban is a puzzle game that can be found at various sites on the Internet [2, 3, 5], and through commercial vendors. If sources are correct, Sokoban is Japanese for warehouse person. We will refer to this person as "the pusher".

The game consists of the pusher who must push a number of boxes into a set of designated storage locations, without getting himself or the boxes stuck. The warehouse is a set of barriers, passages and storage areas. All elements are aligned in a two dimensional grid. The tricky part is that the pusher may only push one box at a time, cannot pull a box, and cannot occupy the same grid location as a box or barrier. Thus, pushing a box into a corner or allowing two boxes to come together on a wall means those boxes cannot be moved further. A puzzle is considered solved if all boxes are pushed into storage locations. The number of storage locations is equal to the number of boxes. Any box may occupy any storage location, (the locations and boxes are unlabeled), although our constructions are such that there is usually only one feasible location for each box.

In this paper we show that we can emulate a Turing Machine (TM) in linear time using an infinite version of the puzzle in which only a finite number of containers are initially out of storage. Restricting the tape to finite length (linear bounded automata) shows that finite puzzles are PSPACE hard. Since the problem is in PSPACE [1], this means the puzzles are PSPACE complete, solving the open problem posed by Dorit and Zwick[1]. We refer the reader to this paper for a summary of related research. Unlike the proof of PSPACE-completeness of SOKOBAN$^+$ presented in [1], our constructions rely critically on the the fact that each of the boxes must be placed in a storage location.

Most of the constructions used in this paper are not much like those in the popular games. In any solution sequence in our constructions, the boxes never stray more than one or two pushes away from the storage location they must eventually occupy in the solution state. The designs are adapted to the goal of restricting the movements of the pusher. This is different in flavor from the design patterns of popular versions, where usually boxes are scattered but must be moved some distance to one of several contiguous sets of storage locations.

Throughout this paper, diagrams will be used to represent constructions. The pusher will be represented by a man-like figure, boxes will be represented by a circle. Filled circles will represent boxes that are in storage locations, while empty circles will represent boxes not in storage locations.

2 Basic Forbidden Configurations and Elementary Devices

As with most such arguments, the emulator constructed in this paper will rely on a number of devices with special properties.

First is the notion of an unrecoverable configuration. A set of boxes in certain locations is said to be *an unrecoverable configuration* if no possible move sequence exists to move all the boxes in the set to storage locations. Note in general that we will use this term to discuss small subsets of the puzzle construction.

For example, in figure 1(a) it is not possible to restore the box to its only possible storage location indicated by the shaded area just above it.

Similarly, in figure 1(b) and (c) we see two configurations of two boxes that cannot be moved, and are not in their final destinations. These and similar configurations will be used frequently to limit the possible moves of the pusher in our construction.

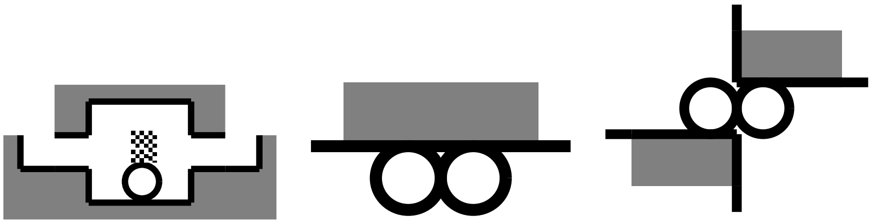

Figure 1: Unrecoverable Configurations, (a), (b) and (c)

Clearly, if at some time a puzzle contains an unrecoverable subset, then the puzzle has no solution from that configuration. In our arguments, we make frequent claims such as "the pusher can only traverse the device in one direction". More formally, this means that doing forbidden moves is either impossible, or would leave the boxes in the device in an unrecoverable configuration. We will refer to move sequences that leave a subset of the puzzle in an unrecoverable configuration as *infeasible*.

The first device we consider is the one-way, or directed passage device. This device was used by Steven Sabey[4] to show that it is NP-hard to minimize the number of pushes required to solve (a version of) this problem.

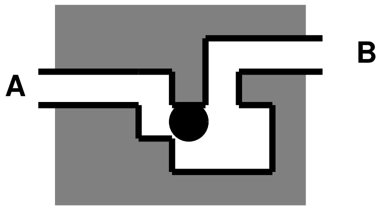

Figure 2: One Way Device

As its name suggests, the one-way device shown in figure 2 has the property that the pusher can move from **A** to **B** as often as the device is reached, always leaving the box in the same storage location. However, it is not possible to move from **B** to **A** without pushing the box into a location such that it is not recoverable. It is also not possible to remove the box from the device. This device is quite simple and we leave the reader to convince herself of these properties.

The reverser shown in figure 3 in its solved state on the left (i.e. when the boxes are in the storage locations) will allow the pusher to pass from

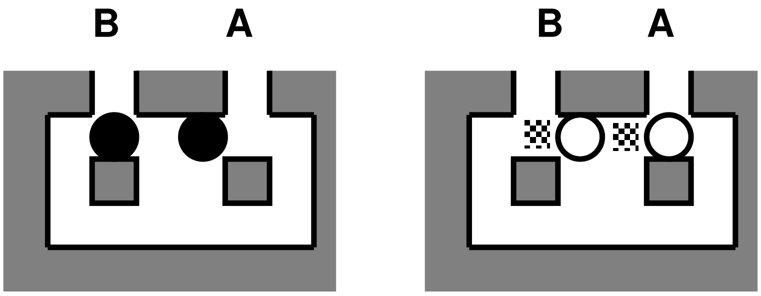

Figure 3: Reverser

A to **B**. However, it is easy to see that this traversal will either leave the device in its unsolved state, as shown on the right, or will leave the device in an unrecoverable state.

To restore the device the pusher must re-enter at **B** and exit at **A**. Thus, every passage through the device will have to be paired with one in the opposite direction. To solve the puzzle, the last passage must be from **B** to **A**.

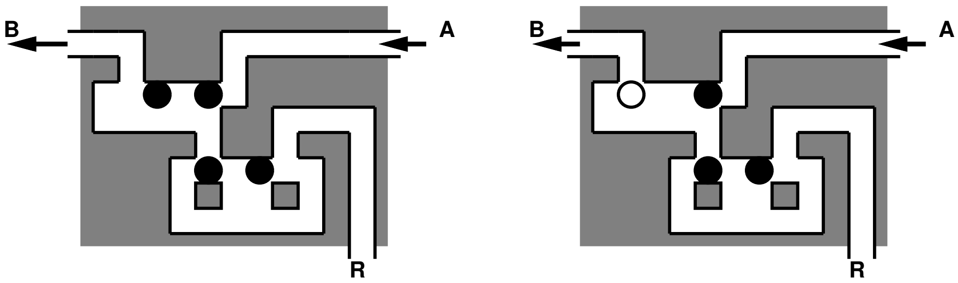

Figure 4: Pass-Reset

The pass-reset device is shown in two configurations in figure 4. The arrows at **A** and **B** indicate that there are one-way devices located at the entrance and exit of the device, restricting the pusher's direction of travel as indicated. These simplify the arguments about the properties of the device, since they prevent entry at **B** or egress at **A**. Also, note that the bottom of the device is a reverser.

When in its solved (or closed) configuration, as on the left, it is not feasible to enter at **A** and exit the device. We remind the reader that by "not feasible" we mean it is either impossible, such as exiting through **R**, or would leave the device in an unrecoverable configuration. For example, an entry at **A**, followed by a single push of the first box encountered against the next one, followed by a detour around the boxes and egress at **B** would leave the two boxes adjacent on a wall, one of the unrecoverable configurations shown in figure 1.

Thus, the only feasible entry into the device when in the solved configuration is through the reverser via entry point **R**. This allows resetting the device as shown on the right. Once reset, egress can only be through **R** as demonstrated in the next paragraph. On a subsequent arrival at **A** the pusher may pass through the device, exiting at **B** and perforce putting the device into the solved state on the left of the figure. To do this the pusher would push the first box encountered one step to the left, then go around it and push it back, then push the exit box one step to the right (so both boxes are in storage locations) and exit through **B**.

Finally, we must verify that it is not feasible for the pusher to enter at **R** and exit at **B**. If the pusher enters at **R** and exits at **B**, then the device cannot be successfully entered again. The pusher cannot enter at **R** because the reverser will be in its alternate position. This also means that the device is in an unsolved state. Entry at **B** is forbidden by the one way device, and since the exit was through **B** the leftmost upper box must be in its storage position (if the device has not already been rendered unrecoverable). This means we cannot enter at **A**, since the only way to enter would push the two upper boxes together. Now since the pusher cannot feasibly enter the device again, and the reverser is in the unsolved state, this passage leaves the device in an unrecoverable state.

In figure 5 we show icons for the devices we need. The pass-reset device may be oriented either way; arrows indicating direction will be added in the final construction. We will discuss the planar crossover device in section 4. For now the reader may imagine a simple underpass, requiring the puzzles to be in three dimensions (but only two levels). The key idea is that the pusher may pass straight through the device, but not make a turn. Junctions are easily implemented as halls with passages leading off.

3 The Turing Machine Emulator

The basic form of the TM emulator is shown in figure 6.

The shaded boxes are collections of devices representing particular subsystems of the TM. This figure represents the state control system and tape

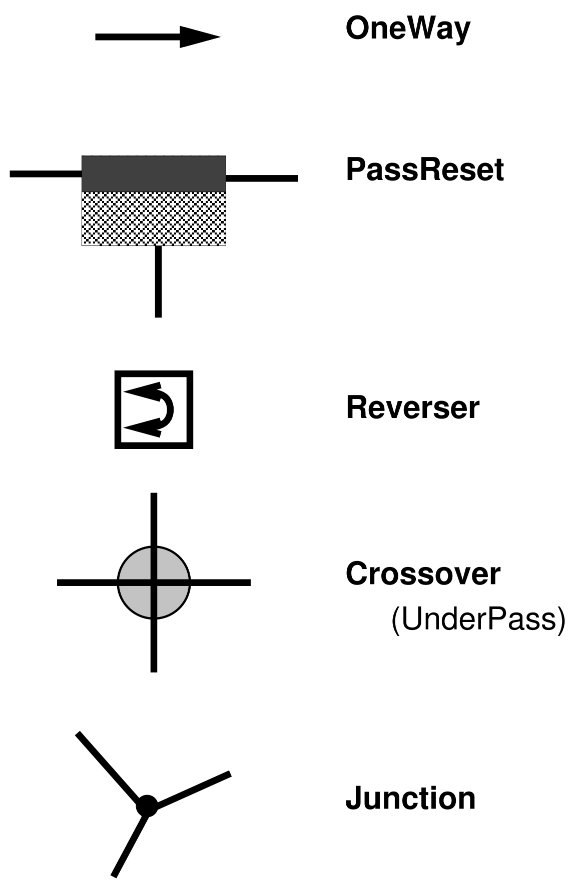

Figure 5: Device Icons

symbol for one tape cell and for only one entry state. Other states would need copies of this structure, sharing only the Tape Symbol section. These "other states" are represented by cloud-like regions.

On entry, the only pass-reset not in the closed position would be one in the Tape Symbol region; that one would represent the symbol currently held at this tape location. In this construction, only '0', '1' and blank ('b') are represented.

The pusher would proceed through the crossover devices to the pass-reset in the Entry State Indicator region, which could then be reset to the open position. Note that to exit this Cellular Unit (to the next or pervious cell) the pusher must pass through this device.

The pusher may then proceed only to the Tape Symbol region, and since only one symbol is represented, only one of these devices will be open. This forces the exit route from the Tape Symbol, and passage through it erases the symbol by closing the pass-reset device. Since only one Entry State Indicator device is open, only one path is open to the pusher. Note that this path is determined by the entry state and the tape symbol, thus ensuring the TM emulation.

On the path from the Tape Symbol region to the Entry State Indicator, the pusher has the opportunity to reset exactly one tape symbol, and to open one device in the Select Next State region. The latter is required, for otherwise the pusher will not be able to exit this Cellular Unit. Note the use of reversers to allow multiple path entries to the Tape Symbol devices and the Select Next State devices while ensuring that the pusher cannot switch to another exit path.

This construction shows that it is possible for the pusher to follow the TM execution. However, to complete our proof we need to consider the following:

1. If the TM halts and accepts, there may be (an arbitrary number of) unvisited tape cells. In the emulator, these unvisited cells will still have the pass-reset device in the open state, meaning the puzzle has not yet been solved.

2. While some of these cells may be input characters, the remaining ones may be blanks beyond the end of the input. We cannot meet the linear time bounds if there are too many of these (in particular, if we want an infinite extension).

3. The TM may visit every cell on a finite tape, but halt without accepting, or not halt due to cycling. In either case, the puzzle may still be solved by the emulator because the pusher may fail to reset the Tape Symbol before exiting the Cellular Unit.

First, we add a Halt-Accept corridor that can only be reached by exiting some Cellular Unit by emulating a transition to a halt accept state. From this corridor, connections to each Tape Symbol region, controlled by a pass-reset device in the Entry State Indicator, will allow closure of all open Tape Symbol devices.

To ensure the linear time emulation, we will use on-the-fly initialization. We add a special end-of-tape symbol that behaves identically to the blank symbol except that it allows the pusher to reset the end-of-tape symbol in the next Tape Symbol region. All Tape Symbol devices beyond the end-of-tape symbol will be closed initially. Thus, even in an infinite version, only a finite number of boxes, representing the input symbols, will be out of storage location initially.

Finally, to prevent false solutions, the Halt-Accept corridor will have one box not in a storage location initially. This means the puzzle can only be solved if the TM halts in the accept state, which allows the pusher to reach this last box. To meet the time bound, this box must be located near the first Cellular Unit.

The Sokoban puzzle will have a solution if and only if the TM being emulated halts in the accept state, and in this case the solution will be $\Theta(n + t(n))$ moves (and pushes) long.

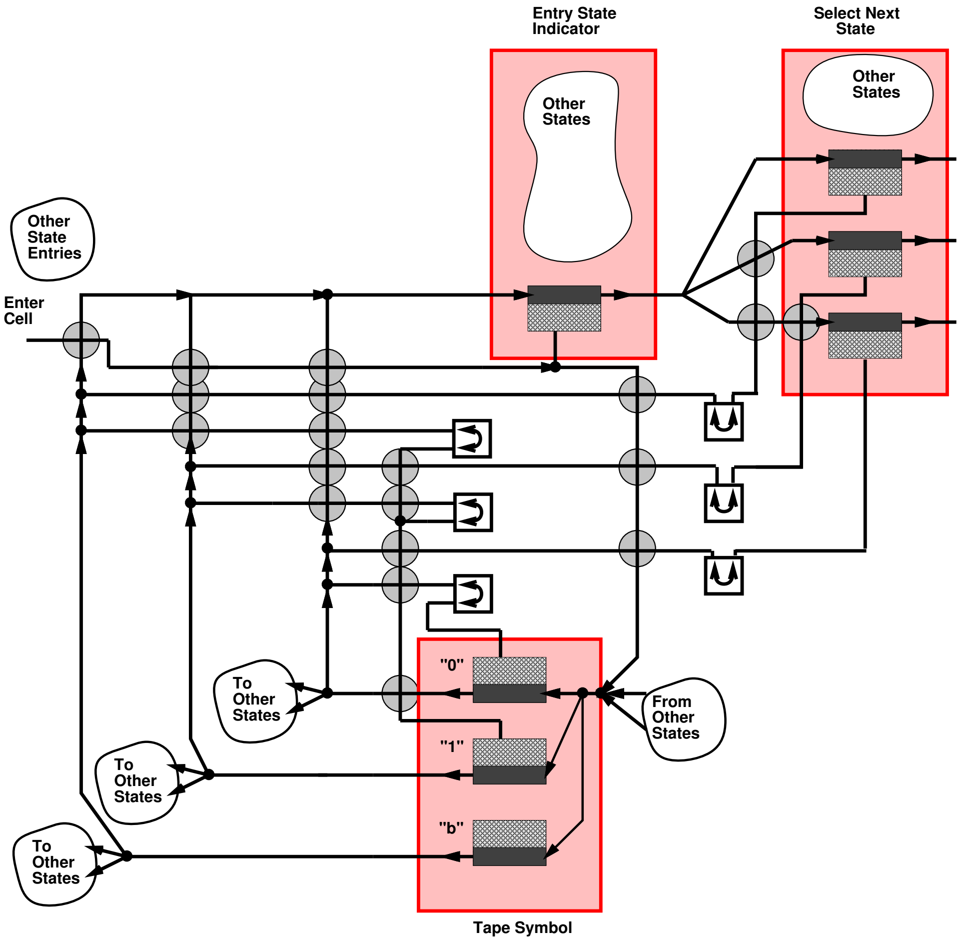

Figure 6: Cellular Unit of the TM

4 The Planar Crossover Construction

The planar crossover device is illustrated in figure 9.

This is by far the most complex device of the construction. The basic properties are that the device is initially in the solved state, that any passage through the device may leave it in the solved state, and that if entry is at **A** then the only feasible exit is at **A'** while if entry is at **B** then the only feasible exit is at **B'**.

Note that the device is directional since no entry can be forced at either **A'** or **B'**. This means that to build the TM emulator, each circle in figure 6 may require either one, two or four of these devices suitably rotated and reflected, depending on whether the intersecting passages are uni-directional or bi-directional.

To discuss the device, it has been broken into three subdevices. Two of these, device 2 and device 3, are essentially the same device under rotation. While discussing these devices, we will not consider infeasible moves of the pusher, that is those which clearly leave a box or boxes in unrecoverable positions.

The first of these devices is shown somewhat compressed in figure 7. Its purpose is to force the pusher to move the box in the Lock to block the path from **B** to **B'**, whenever the pusher enters at **A**.

Clearly the only feasible path requires the pusher to enter the Interlock. This device is an extension of the reverser. The pusher must move each box one push downwards (i.e. towards **A'**) and exit the Interlock at the top. However, the pusher cannot exit immediately at **A'**, since to do so would force two boxes together in an unrecoverable configuration.

The pusher may move the box in the Lock one step to the right, then return through the Interlock and re-enter the Lock from the top. An exit at **A'** will now be feasible, restoring the last box in the Interlock, and all boxes in this subdevice except the one in the Lock will be in storage locations. This last box may be stored while the pusher is passing through subdevice three.

Subdevice Two is illustrated in figure 8.

If entry to this device is from **A** then the **B'** exit has been blocked by passage through device one, so the only feasible exit is through **A'**. The only feasible path requires one box (on the far right) to be moved in the Interlock, which can be restored before leaving the subdevice by entering the Interlock through the Forward Reset Entrance.

We now consider other paths available on entry at **A**, and show they are not feasible. As discussed in the next paragraph when considering entrance

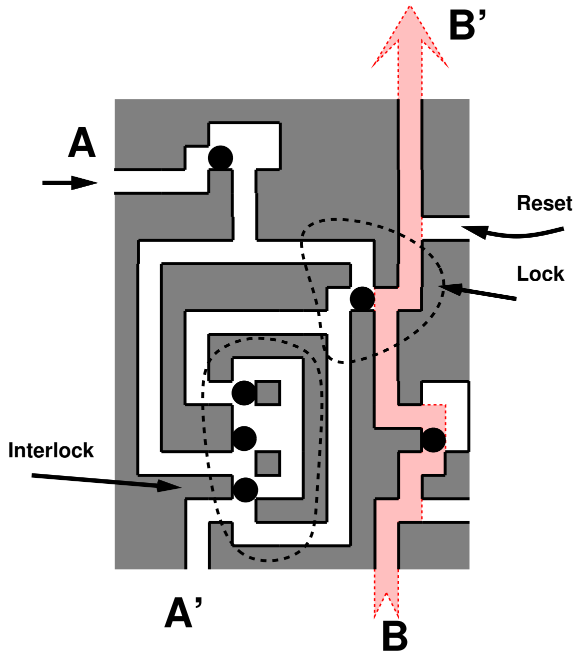

Figure 7: Subdevice One

at **B**, it is not feasible to enter through the Lock Reset Entrance and try to exit at **A'**, because the two rightmost boxes in the Interlock will be left adjacent. For similar reasons entering the Interlock at the Forward Reset Entrance and circling back through the Lock and exiting at **A'** is also infeasible. Entering the Interlock through the Forward Reset Entrance and exiting the Interlock via the Lock Reset Entrance is impossible. These exhaust the possibilities when entering at **A**.

When entering at **B**, the pusher encounters the box in the Lock, which must be moved two cells blocking the path to **A'**. The only feasible path takes the pusher to the junction of the Lock Reset Entrance. The pusher now enters the Interlock through this reverser.[1] By pushing each box in the Interlock except the rightmost one step to the right, the pusher may exit the Interlock to the Lock and restore the box to its solved position. The pusher

[1] He may of course exit at **B'**, but this leaves the box in the Lock in an unsolved position. If the pusher will enter this crossover device again at **B** before an entry at **A**, then this box may be left until the next passage. But eventually it must be restored as described.

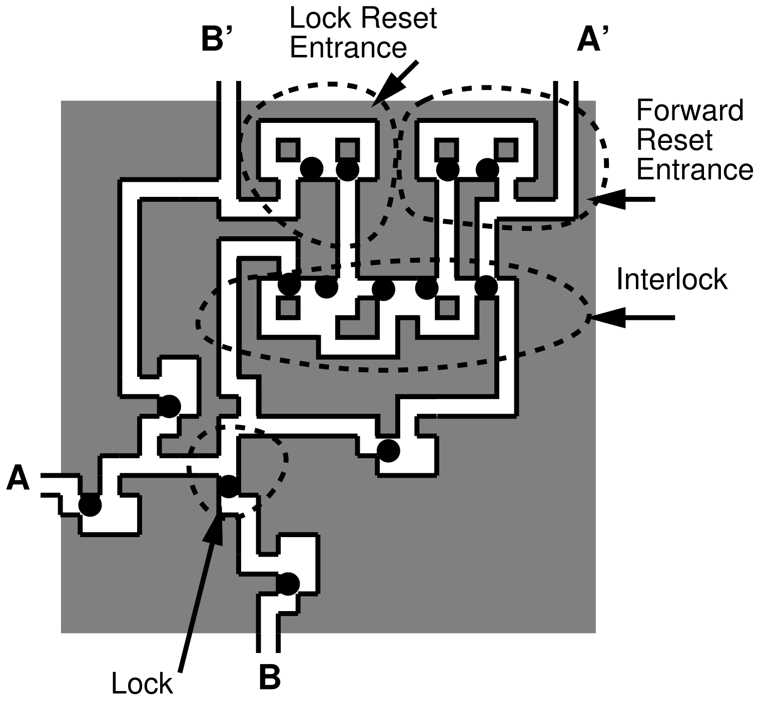

Figure 8: Subdevice Two

cannot now exit through **A'** because to do so he would push the rightmost box of the Interlock against the next box in the Interlock, thus creating an unrecoverable configuration. The only way to restore the boxes inside the Interlock is to return through the Interlock, and then exit through the Lock Reset Entrance, and then perforce to exit at **B'**.

Thus, entering at **A** forces an exit at **A'**, while entrance at **B** forces an exit at **B'**. In each case, all boxes are returned to storage locations.

The third subdevice is the same as subdevice two. Before exiting at **A'** (in figure 9) it allows restoring the box in the Lock in subdevice one, without allowing an exit at **B'**.

We have shown that the planar crossover device in figure 9 fulfills its function. Namely, entry at **A** requires exit at **A'**, while entry at **B** requires exit at **B'**, and all boxes in the device remain in storage locations after either passage.

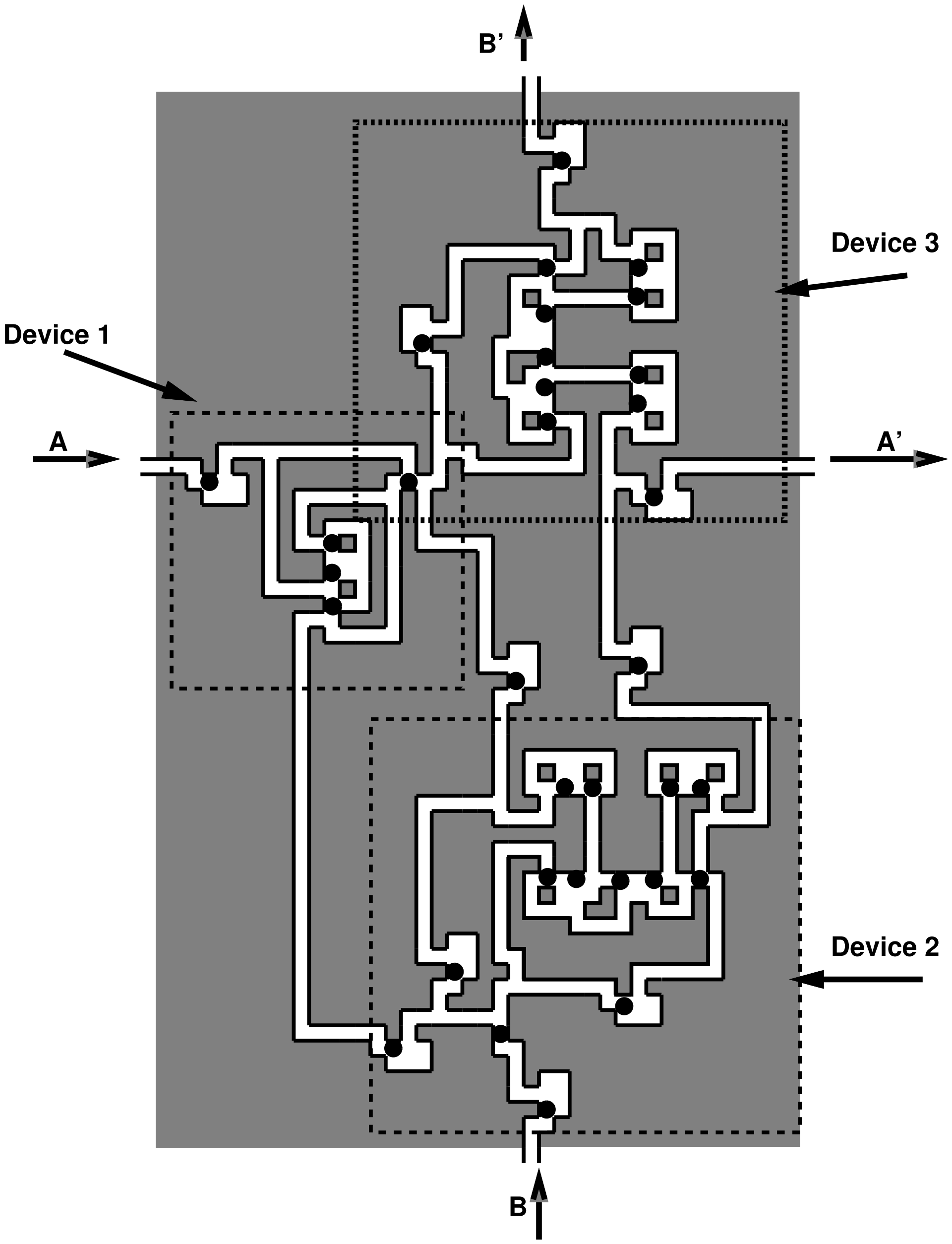

Figure 9: Planar Crossover

5 Summary

The construction presented herein is clearly polynomial (linear) and so we have shown that Sokoban is PSPACE-complete. This answers the open question left in [1].

It is also easy to see that using the construction it is possible to build an infinite (i.e. a uniform extension) version of the puzzle in which only a finite number of boxes are not in storage, and that this version would be uncomputable by a reduction to the halting problem.

We point out that some simplification of our devices could be obtained if the barriers could be made thinner, that is if walls could be placed between grid points. For example, Subdevice One of figure 7 could be replaced by the device in figure 10. Such thin wall design might make the design of popular

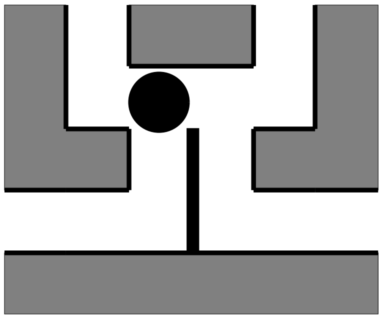

Figure 10: Thin Walled Device

puzzles easier and more interesting.

In addition to thinner walls, interesting puzzles might also be created by the addition of three dimensional underpasses. Note that our planar crossover construction, aside from being too clumsy for practical puzzle design, also does not allow boxes to be pushed through it. By allowing simple third dimension bridges, additional complexity could be built into small puzzles.

The instances of the puzzle that are popular usually require the boxes to move some distance to a set of contiguous storage locations. We wonder what the complexity of the puzzle would be under the constraint that all storage locations must be contiguous. Notice that under this constraint, the use of three dimensions to allow underpasses may be of particular importance.

References

[1] Dorit Dor and Uri Zwick. Sokoban and other motion planning problems (extended abstract). Preprint http://www.math.tau.ac.il/ ddorit/, 1995.

[2] Scott Lindhurst. Sokoban for the Macintosh http://www.math.wisc.edu/ lindhurs/sokoban/sokoban.html

[3] Andrew Myers. XSokoban http://xsokoban.lcs.mit.edu/xsokoban.html

[4] Stephen Sabey. On the complexity of Sokoban. Unpublished 1996.

[5] David L. Tyler. Altus Software Marketing - Sokoban http://www.axxis.com/altus/sokoban.html (A long list of shareware/freeware Sokoban sites)