SOKOBAN and other motion planning problems

Dorit Dor *   *Uri Zwick* *

June 25, 1996

Abstract

We consider a natural family of motion planning problems with movable obstacles and obtain hardness results for them. Some members of the family are shown to be PSPACE-complete thus improving and extending (and also simplifying) a previous NP-hardness result of Wilfong. The family considered includes a motion planning problem which forms the basis of a popular computer game called SOKOBAN. The decision problem corresponding to SOKOBAN is shown to be NP-hard. The motion planning problems considered are related to the "warehouseman's problem" considered by Hopcroft, Schwartz and Sharir, and to geometric versions of the motion planning problem on graphs considered by Papadimitriou, Raghavan, Sudan and Tamaki.

1 Introduction

An instance of a generic motion planning problem consists of a description of an environment, containing some objects and obstacles, and a description of a desired state, or states, into one of which the environment is to be transformed. The answer to such an instance is a plan describing the co-ordinated motion of all the objects and obstacles that satisfies the constraints of the system, or a claim that such a plan does not exist. The constraints imposed depend on the exact nature of the problem. A constraint that is almost always used is that objects and obstacles should never overlap.

Various forms of motion planning problems were already considered by many researchers. Geometric motion planning problems were considered, among others, by Reif [Rei79], Hopcroft, Schwartz and Sharir [HSS84], Wilfong [Wil91] and Dhagat and O'Rourke [DO92]. A motion planning on graphs which forms an abstraction of such problems was considered by Papadimitriou, Raghavan, Sudan and Tamaki [PRST94].

In this work we consider a family of motion planning problems obtained by generalising the rules of a computer game called SOKOBAN. [1] A typical level of SOKOBAN is shown in Figure 1. Each level is composed of a layout of 'warehouse' laid down on a rectangular grid (not shown in the figure). Each cell of the grid either forms part of the warehouse's floor, or forms part of a 'wall' separating the warehouse into rooms and halls. Some of the floor cells contain packets. Each packet is a 1×1 square occupying a single cell (the packets are shown in the figure as disks for aesthetic reasons). A poor porter is supposed to move these packets to certain designated target positions, shown shaded. The initial position of the porter is also given. The porter has enough strength to push a single packet. She cannot push more than one packet at once. She is not able to pull packets. The porter may move freely on the warehouse's floor but she is not allowed to step on packets, i.e., she is not allowed to be in a cell occupied by a packet. Readers interested in playing SOKOBAN may point their WWW client to `http://clef.lcs.mit.edu/~andru/xsokoban.html`. In the SOKOBAN game, all levels, or puzzles, have solutions, though some of them are extremely long.

*Department of Computer Science, School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, ISRAEL. E-mail addresses: {ddorit,zwick}@math.tau.ac.il. Contact author: Uri Zwick

[1] SOKOBAN, if our sources are correct, is the Japanese word for 'a warehouse keeper'.

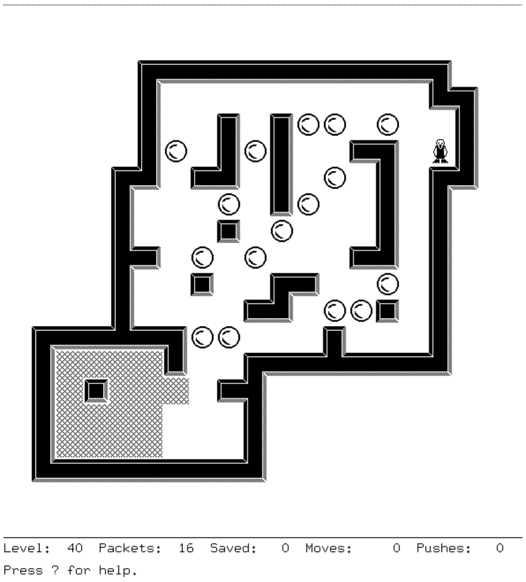

Figure 1: Level 40 of SOKOBAN.

We are looking at the problem of deciding whether a given SOKOBAN puzzle has a solution. We show that this problem is NP-hard. It is an interesting open problem whether this problem is in NP. It is clearly contained in nondeterministic PSPACE and therefore, by Savitch's result [Sav70], also in PSPACE.

We let SOKOBAN(k,ℓ) be the following generalisation of SOKOBAN. The packets are still 1×1 squares but the porter is now powerful enough to push up to k packets at once. She can also *pull* up to ℓ packets. For example, suppose that positions $(x+1,y),\ldots,(x+r,y)$ all contain packets and that the porter is in position (x,y). If position $(x+r+1,y)$ is vacant and $r \le k$, then the porter may push the packets to positions $(x+2,y),\ldots,(x+r+1,y)$ and move to position $(x+1,y)$ while doing so. If position $(x-1,y)$ is vacant and $r \le \ell$, then the porter may move to position $(x-1,y)$ while pulling the r packets into positions $(x,y),\ldots,(x+r-1,y)$. The most natural choices for ℓ are $\ell = 0$ (i.e., no pulling) and $\ell = 1$ (the porter may pull the packet next to her). Note that SOKOBAN$(1,0)$ is the original SOKOBAN game. As already mentioned, we show that SOKOBAN$(1,0)$ is NP-hard. We also show that SOKOBAN$(k,1)$ is NP-hard, for every $k \ge 5$, even if the goal is just getting the porter to a specific target position.

Finally, we let SOKOBAN$^+(k,\ell)$ be a game similar to SOKOBAN but with 1×2 rectangular packets instead of the 1×1 square packets of SOKOBAN. We show that SOKOBAN$^+(k,1)$, for any $k \ge 2$, is PSPACE-complete.

The rest of this paper is organised as follows. We begin in the next section with the PSPACE-completeness result for SOKOBAN$^+$. In Section 3 we prove that SOKOBAN$(k,1)$, for $k \ge 5$, and also SOKOBAN$(\infty,1)$, are NP-hard. Finally, in Section 4 we show that SOKOBAN$(1,0)$, i.e., the original SOKOBAN game, is also NP-hard. A comparison of our results with previously known results is given in Section 5. We then end with some concluding remarks and open problems.

2 PSPACE-completeness of SOKOBAN$^+$

In this section we consider a version of SOKOBAN in which the packets are 1×2 rectangles and in which the porter has enough strength to push two packets at once (we can allow her to push more packets at once if we like) as well as the ability of *pulling* one packet. We show that the decision problem corresponding to this version is PSPACE-complete.

The two basic gadgets used in the construction are shown in Figures 2 and 3. The construct shown on the

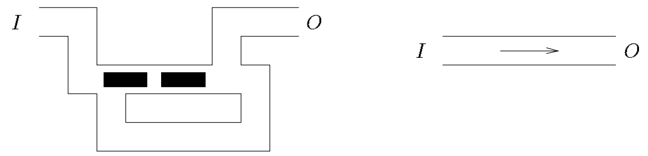

Figure 2: A one-way corridor and its schematic description.

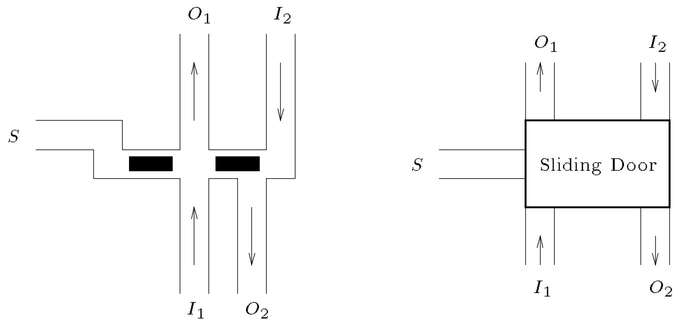

Figure 3: A sliding door and its schematic description.

left of Figure 2 is called a *one-way corridor* as the porter can enter it from the left and leave it from the right but not vice versa. When the porter enters from the left, she pushes the two packets one position to the right, goes around and pushes the two packets to their original position and then exits from the right. If the porter comes from the right, the only thing she can do are the following: she can exit from the right without moving the packets ; she can push the two packets one position to the left, thereby blocking the passage through the corridor, in both directions, forever; she can pull the right packet one position to left but to get out of the bypass she would have to push it back to its original position. In either case, the porter will not be able to leave the gadget from the left, and will either leave it in its original state, or in a state in which it could not be used again in either direction. One-way corridors will be used extensively in the construction of more complicated gadgets. In these constructions, one way corridors will be depicted schematically as shown on the right of Figure 2.

The gadget shown in Figure 3 is called a *sliding door*. Its functionality is less natural than that of the one-way corridor but, as we shall see, many useful and more natural gadgets can be obtained using it. The figure shows the sliding door in its `open` state. In this state, the porter can freely use the passage $I_1 \to O_1$. To use the passage $I_2 \to O_2$, the porter must push the right packet one position to the left, thereby moving the door to its `closed` position. In this position the porter can freely use the passage $I_2 \to O_2$ but not the passage $I_1 \to O_1$. Finally, the porter can open the door again by coming through the corridor marked by S, pushing the two packets one position to the right and then *pulling* the left packet one position to the left.

By connecting exit O_1 to entrace I_2 of a sliding door, we get a *gate*, a gadget shown in Figure 4. The gadget is shown in the figure in its `open` position. The porter can now move from I to O but by doing so she has to move the gate to its `closed` position. The gate can be opened again only by coming through the corridor denoted by S.

A gate is usually depicted schematically as a box and a circle, as shown on the right of Figure 4. The box denotes the gate itself. The circle denotes an activation point of the gate. The dotted line connecting the gate and its activation point stands for a corridor that connects them. We shall soon describe the construction of a crossover. Using crossovers we can always construct a corridor connecting a gate and its activation point. To simplify the diagrams describing our constructions we therefore omit these dotted lines and do not show these corridors explicitly.

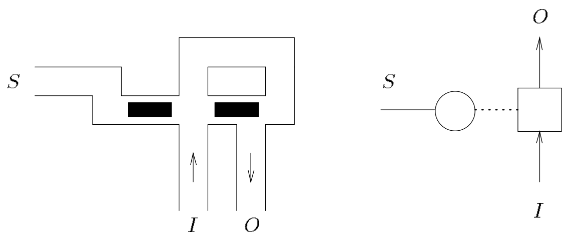

Figure 4: A gate and its schematic description.

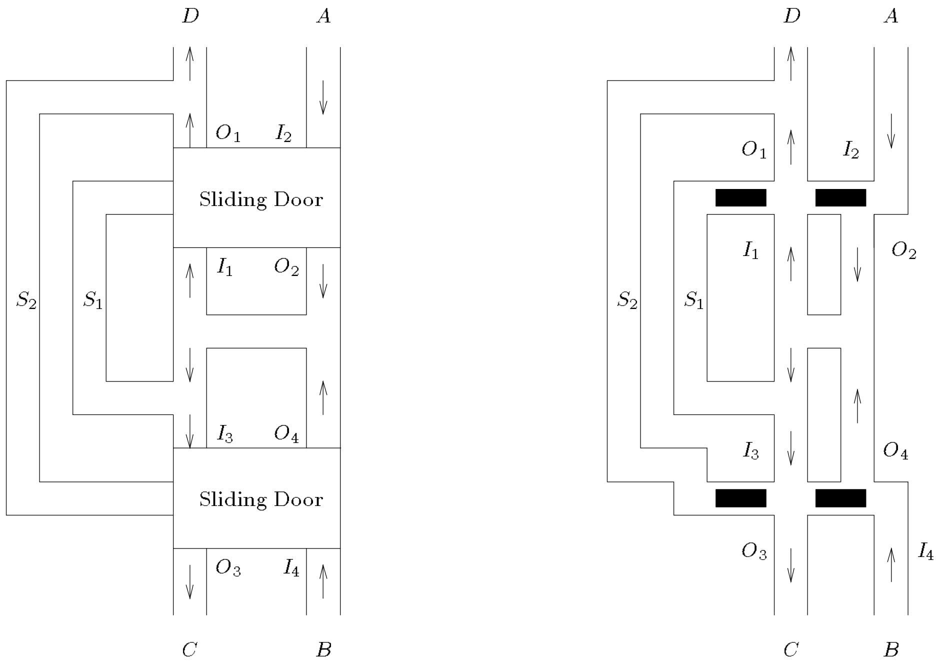

Figure 5: Constructing a crossover using two sliding doors.

The gate described in Figure 4 has a single activation point. By connecting such gates in parallel we can get gates with an arbitrary number of activation points.

Using two sliding doors we can construct a *crossover*, as shown in Figure 5. If the porter enters the crossover through A she must exit through C and if she enters through B she must exit through D. For example, if she enters through A, she must block the passage $I_2 \to O_2$ when she passes through the upper sliding door and could only exit through C. If the porter exits while leaving the upper sliding door in its `closed` position, she would not be able to use the passage $B \to D$ without previously using the passage $A \to C$ again. The porter may however move the upper sliding door to its `open` position by using the corridor S_1. Similarly, if she enters through B, she must block the passage $I_3 \to O_3$ when she passes through the lower sliding door and could only exit from D. She can reset the lower sliding door using the corridor S_2.

We now describe a simulation of a linear bounded automata (i.e., a Turing machine with a fixed length tape) that uses the binary alphabet. The problem of deciding, given such an automata, its initial state, the initial state of the tape, and the initial position of the head on the tape, whether the head would ever reach, say, the last (i.e., the rightmost) tape cell is easily seen to be PSPACE-complete.

Let T be a Turing machine with k states. Each cell of T's tape will be simulated by a construct with k entrances, k exits to the right and k exits to the left. The right (left) exits of a cell are connected to the entrances of the cell to its right (left). The connection between three adjacent cells, with $k = 3$, are shown

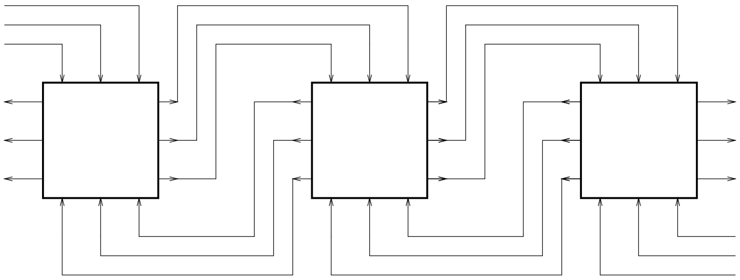

Figure 6: The simulation of the Turing Machine's tape

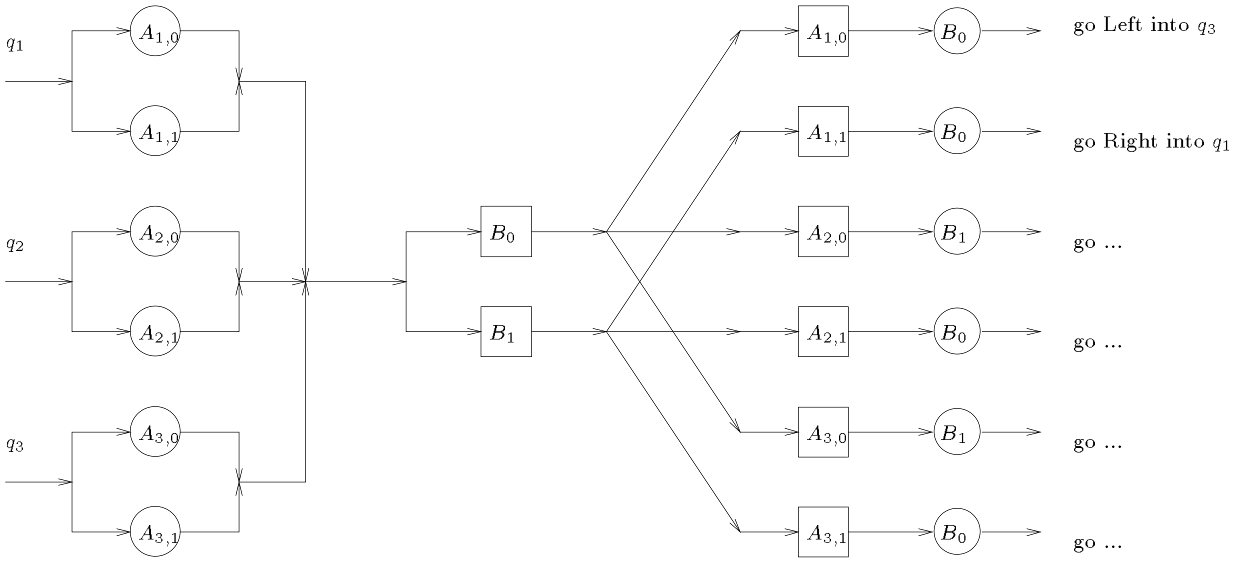

Figure 7: The simulation of each tape cell.

in Figure 6. Each connection shown in the figure is a one-way corridor. For simplicity, each cell is shown to have $2k$ entrances, k at the top and k at the bottom. These however are the same entrances. As we can implement crossovers we can easily connect each pair of these corridors into the same entrance of the cell.

The structure of each cell of the Turing machine is depicted in Figure 7. We again assume in the figure that $k = 3$ but the construction clearly generalises to other values of k. Each cell of a Turing machine contains $2k + 2$ gates denoted by B_0, B_1 and $A_{i,j}$, where $1 \leq i \leq k$ and $j = 0, 1$. When the cell is entered, exactly one of the gates B_0 and B_1 is open. Gate B_0 is open if and only if the tape cell contains 0, and gate B_1 is open if and only if the tape cell contains 1. All the $A_{i,j}$ gates are initially closed.

The cell is entered through the i-th entrance if and only if the Turing machine is in state q_i. When the porter enters through the i-th entrance, she can open one of the gates $A_{i,0}$ and $A_{i,1}$. To be able to leave the cell, the porter should 'guess' the content of the cell and open gate $A_{i,0}$ if it is 0 and $A_{i,1}$ if it is 1. The porter must then 'read' the content of the cell by passing through either B_0 or through B_1 (recall that exactly one of them is open). Both B_0 and B_1 are then closed. If the porter guessed correctly the content j of the cell, she can now pass through gate $A_{i,j}$. Otherwise, she is stuck. The porter is now in one of $2k$ corridors, one for each pair (i, j) of state and content. If the Turing machine is supposed to write j', enter state $q_{i'}$ and move Left (Right), then the porter may now open gate $B_{j'}$ and the corridor she is in is connected to the i'-th entrance of the cell to the left (right) of the current cell. If the Turing machine is supposed to halt, then this corridor is a dead end.

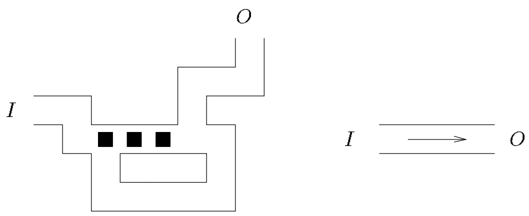

Figure 8: A one way corridor

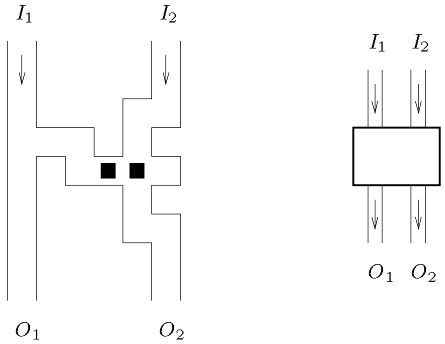

Figure 9: An orderer and its schematic description

This completes the simulation of Turing machines and the PSPACE-completeness proof. The proof described is for the version of the problem in which there is a terminal position for the porter, not for the packets. It is easy to obtain a version of the proof that works for the version of the problem in which there are terminal positions for the packets. We can simply add a packet which the porter has to push when reaching its terminal position. It is easy to verify that the porter may reach her terminal position while leaving all crossovers in their original position and all gates in their `closed` position.

The general structure of the above construction is similar to the general structure of a construction by Chalcraft and Greene [CG94] (see also [Ste94]). Chalcraft and Greene describe a simulation of a Turing machine using train sets.

3 NP-hardness of SOKOBAN$(\infty, 1)$

In this section we show that SOKOBAN(5,1) is NP-hard. Recall that in this version of SOKOBAN the packets are 1×1 squares and the porter may push up to five packets and pull one. The construction is similar to the constructions of Wilfong [Wil91] and Dhagat and O'Rourke [DO92]. The gadgets used, however, are completely different. While Wilfong uses objects of many different shapes, some of them not even rectangular, all the objects that and Dhagat and O'Rourke [DO92] and us use are squares of the same size. Dhagat and O'Rourke [DO92] obtain however a different result. They show, in our terminology, that SOKOBAN$(\infty, 0)$ is NP-hard. The work of Dhagat and O'Rourke [DO92] was inspired by a computer game for Macintosh called "Beast".

Four basic gadgets are used this time. A one-way corridor is shown in Figure 8. An *orderer* is shown in Figure 9. In this gadget, the porter may freely use the passage $I_1 \to O_1$. It may only use the passage $I_2 \to O_2$ after using the passage $I_1 \to O_1$ at least once. The passage $I_2 \to O_2$ is opened by pushing the two packets one position to the right, and then pulling the left packet one position to the left. A *switch* is shown in Figure 10. The porter may use only one of the passages $I_1 \to O_1$ and $I_2 \to O_2$, once one of these passages is used, the other passage is blocked forever. The reader may care to verify that a similar construct with two pairs of packets instead of two triplets does not function properly. By joining together

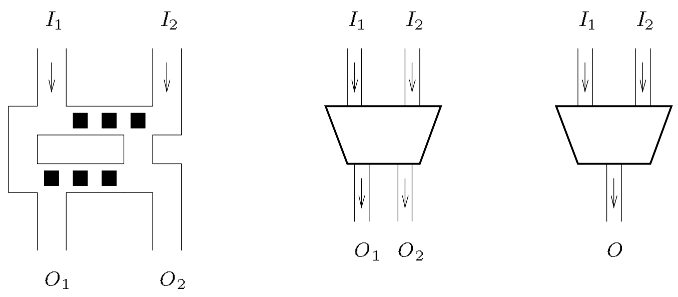

Figure 10: A Switch and its schematic description

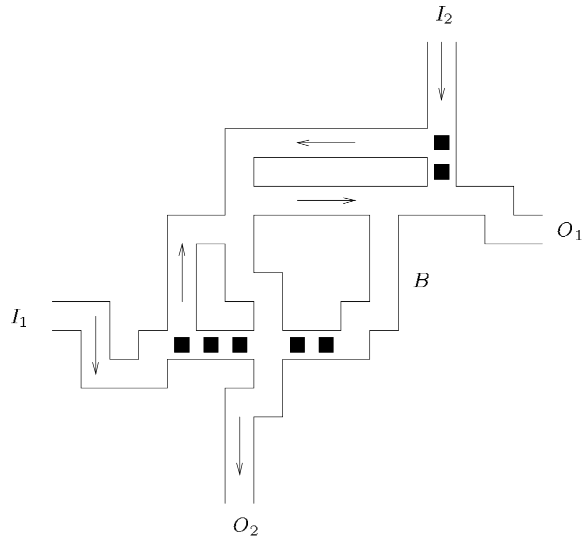

Figure 11: A Limited crossover

the two exits O_1 and O_2 we get a gadget with two entrances, I_1 and I_2, and one exit, O. The porter is now able to use one and only one of the passages $I_1 \to O$ and $I_2 \to O$.

A limited crossover is shown in Figure 11. The porter may use the passage $I_1 \to O_1$ as many times as she wants. In the first time the passage $I_1 \to O_1$ is used, the three adjacent packets are pushed one position to the right thus blocking the exit O_2. During one of the passages from $I_1 \to O_1$, the porter may decide to open the passage $I_2 \to O_2$. To do this the porter uses the bypass, denoted by B, pushes all five packets one position to the left and then pulls the rightmost packet one position to the right. The porter may then leave from O_1. The porter will now be able to use the passage $I_2 \to O_2$ for as many times as she wants but she will not be able to use the passage $I_1 \to O_1$ again. The porter may also use the passage $I_2 \to O_2$ without previously using the passage $I_1 \to O_1$. It can be verified that whenever the porter enters from I_1 (I_2), she must exit through O_1 (O_2).

Let F be a 3SAT instance. Let $C_1, \ldots, C_m$ be the clauses of F and let $v_1, \ldots, v_n$ be the variables appearing in F. We construct a warehouse with m orderers and n switches, as shown in Figure 12. The two entrances of the i-switch correspond to the literals v_i and $\overline{v_i}$. The first exit of the i-th orderer splits into three corridors leading to the three literals contained in the i-th clause. Limited crossover, like the ones described above are used whenever two corridors cross each other. When a corridor coming from the i-th orderer crosses a corridor coming from the j-th orderer, where $i < j$, the limited crossover is set so that the corridor coming from the j-th orderer can be used after using the corridor coming from the i-th orderer.

The porter starts at position A. She is supposed to reach position B. It is easy to see that to reach position B, the porter must first use the passage $1 \to 2$, then $3 \to 4$, and so forth. After using the passage $1 \to 2$,

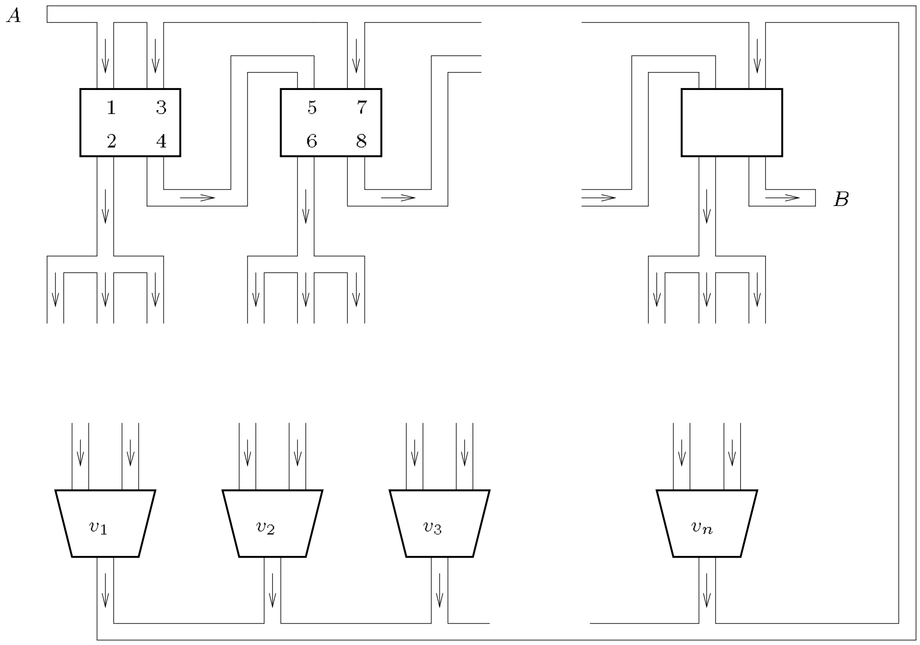

Figure 12: A representation of a 3SAT instance

the porter must assign the value "true" to at least one of the literals appearing in C_1. Similarly, after using the passage $5 \rightarrow 6$, the porter must assign the value "true" to at least one of the literals appearing in the C_2. The porter can not assign to value "true" to a literal and its negation. It follows therefore that the porter can reach B if and only if the 3SAT formula F is satisfyable.

In the construction described there is again a terminal position for the porter, not for the packets. It is again possible to alter the proof so that it would also apply to the version of the problem with terminal positions for the packets. The changes required this time are non-trivial and require a few more gadgets. We only give a sketch of them here. We add *service entrances* to all the gadgets used in the current construction. The porter can access these entrances only after reaching her terminal position. Coming through a service entrance of a gadget, the porter may push the packets contained in the gadget into appropriate terminal positions within the gadget. We add a corridor that connects the service entrances of all the gadgets used in the construction. This corridor may of course cross corridors of the original construction. We need a new crossover gadget to handle these crossings.

4 NP-Hardness of SOKOBAN$(1,0)$

In this section we show that the SOKOBAN$(1,0)$, i.e., the original SOKOBAN game, is NP-hard by giving a polynomial time reduction to it from the planar 3SAT problem (P3SAT for short) which is defined before. A different NP-harness proof for SOKOBAN$(1,0)$ was independently obtained recently by Fryers and Greene [FG95].

The P3SAT problem is a sub-problem of the 3SAT problem. An instance $I = (C, X)$ of 3SAT consists of a set of clauses $C = \{C_1, \ldots, C_m\}$ and a set of variables $X = \{x_1, \ldots, x_n\}$. Each clause C_i consists of at most three literals $l_{i,1}, l_{i,2}, l_{i,3}$ where a literal $l_{i,j}$ is either a variable x_k or its negation $\overline{x_k}$. The problem is to determine whether C is satisfiable, that is, whether there exists a Boolean assignment to the variables which simultaneously satisfies all the clauses in C. A clause is satisfied if one or more of its literals has value "true".

For the P3SAT problem, we consider only instances $I = (C, X)$ of 3SAT whose *connection graph*, $G_I =$

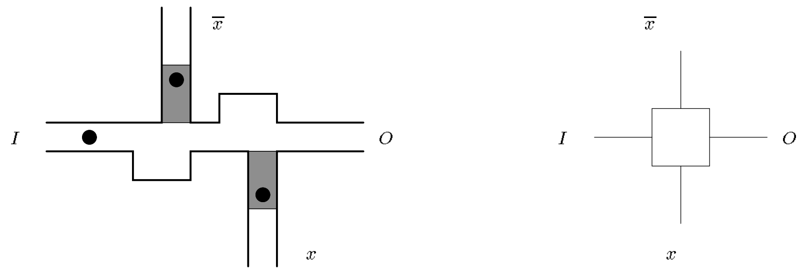

Figure 13: A selector and its schematic representation.

(V, E), is planar. The vertex set of the graph G_I is $V = X \cup C$. The edge set of G_I is

$$E = \{(x_i, x_{i+1}) \mid 1 \le i \le n\} \ \cup \ \{(x_j, C_i) \mid x_j \in C_i \text{ or } \overline{x_j} \in C_i\} \ ,$$

where the index of x_{i+1} is interpreted modulo n. The graph G_I contains an edge between each variable x and each clause that contain either x or $\overline{x}$. It also contains a simple cycle passing through all the variables. Lichtenstein [Lic82] showed that the P3SAT problem is NPC. Moreover, he showed [Lic82] that the P3SAT problem is NPC even when, at every variable vertex x, all edges representing positive instances of the variable are incident to one 'side' of the node and all edges representing negative instances of the variable are incident to the other 'side' of the node, or more precisely, if there is an embedding of G_I in the plane in which for each variable x, either all clauses containing x are inside the cycle passing through all the variables and all clauses containing $\overline{x}$ are outside this cycle, or vice versa.

Given an instance I of P3SAT, we construct an instance S_I of the SOKOBAN game which has a solution if and only if I is satisfiable. Two types of gadgets are used in this construction. For each variable x, the SOKOBAN instance includes a *selector*, a gadget shown on the left of Figure 13. A selector is connected to the outside world by four corridors denoted by I, O, x and $\overline{x}$. Corridor I is main entrance and corridor O is the main exit of the selector. The other two exits are the x-exit and the $\overline{x}$-exit. In contrast to the previous constructions, all the corridors used in this construction are two-way and the porter may therefore enter a selector via an exit. This possibility will of course be taken into account. A selector contains four terminal positions, two in the x-exit and two in the $\overline{x}$-exit. Three packets are initially placed in each selector, one in the entrance and one in each of the literal exits. The selectors will be depicted schematically as a box, as shown on the right of Figure 13.

The SOKOBAN instance S_I includes a chain of selectors corresponding to the variables $x_1, \ldots, x_n$, as shown in Figure 15. The main exit of the i-th selector is connected to the main entrance of the $i+1$-st selector. The x_i-exit is the upper exit of the i-selector if and only if the clauses that contain x_i are inside the cycle of G_I that passes through all the variables. The main entrance of the first selector is connected to a reservoir containing $n+m$ packets. The main exit of the n-th selector is a dead-end (we could have connected it, via a 'valve', back to the reservoir but as this is not required we choose not to do it). For each clause, the instance S_I contains a gadget like the one shown in Figure 14. Each such gadget contains one terminal position. To solve the puzzle, the porter would therefore have to push a single packet to each one of the clauses. From each literal, we connect corridors to all the clauses containing it. All these corridors are wide enough so that the porter can push a packet through all the necessary turns. The clauses are positioned according a planar embedding of the graph G_I. There are no intersections therefore between the different corridors and no crossovers should therefore be used. The initial position of the porter is inside the reservoir.

It is easy to see that if the P3SAT instance is satisfiable, then the SOKOBAN puzzle constructed is solvable. The porter begins by taking a stroll through the chain of selectors. In each selector the porter pushes the packet initially placed at the entrance of the selector and blocks one its literal. The porter blocks the literal exit that corresponds to the literal that gets the value "false" under a satisfying assignment of the P3SAT

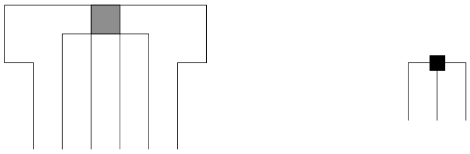

Figure 14: A clause gadget and its schematic representation.

formula. Each literal exit that corresponds to a "true" literal remains unblocked, though it still contains the packet initially placed at that exit. The porter may push the original packet placed in such a literal exit to one of the clauses containing this literal. She may then push packets from the reservoir through this literal exit to all the other cluases containing this literal. As the unblocked literal exits correspond to a satisfying assignment of the P3SAT formula, the porter may push a packet to each one of the clauses. After satisfying all the clauses, the porter pushes additional packets from the reservoir and blocks all the literal exits of the selectors. This constitues a solution to the SOKOBAN puzzle.

We now show that if the SOKOBAN instance S_I is solvable then the P3SAT instance I is satisfiable. We begin by showing that in any solution of the SOKOBAN puzzle, the first entrance to each selector contained in the chain of selectors must be through its main entrance and that before leaving the selector for the first time, the porter must block one of its literal exits. Note that the first entrance to a selector cannot be through one of its literal exits. If the porter enters a selector for the first time from a literal exit, she would have to push the packet initially placed in this exit into a corner. The packet could never be moved out of such a corner and the sequence of moves made by the porter could not be completed to a solution of the puzzle. If a selector is first entered through its main exit, then one of the selectors following it in the chain must have been entered, for the first time, though one of their literal exits, a contradiction. It follows therefore that each selector is indeed first entered through its main entrance. Consider now the first entrance to the i-th selector. As this entrance is through its main entrance, the porter must push the packet initially placed at the entrance of the selector. If the porter pushes this packet past the two literal exits then this packet and the packet placed at the entrance to the $i+1$-st selector could never be pushed to terminal positions (the packet at the entrance of the $i+1$-st selector must still be there as the $i+1$-st selector was not visited yet). The porter must therefore block one of the literal exits of the selector, as claimed.

Finally, we observe that a packet can be pushed into a clause only through one of the literal exits that correspond to the literals that appear in it. Each solution of the SOKOBAN puzzle yields therefore a satisfying assignment of the P3SAT formula.

5 Comparion with related works

As mentioned, SOKOBAN is similar to a motion planning problem with 'movable obstacels' studied by Wilfong [Wil91]. In Wilfong's problem the porter (or *robot*, as she is called there) is allowed to push obstacles. She is also allowed to *grasp* an obstacle and move along with it, as though they were a single object. In our formulation, grasping is replaced by pulling, which we think is more basic. We obtain a PSPACE-completeness result while Wilfong only gets an NP-hardness result. In his NP-hardness results, Wilfong uses objects of several different shapes, not all of them rectangles. Our proof uses only 1×2 rectangles. Our proof holds even if grasping is allowed and hence Wilfong's original problem is also PSPACE-complete. Dhagat and O'Rourke [DO92] consider motion planning problems in which the porter is allowed to push objects but not to pull them.

Hopcroft, Schwartz and Sharir [HSS84] consider a problem they call the "Warehouseman's problem". The problem considered by Hopcroft *et al.* is a generalization of the famous 15-puzzle. A large rectangle contains many small rectangles of many different sizes, together with some gaps between them. Recangles

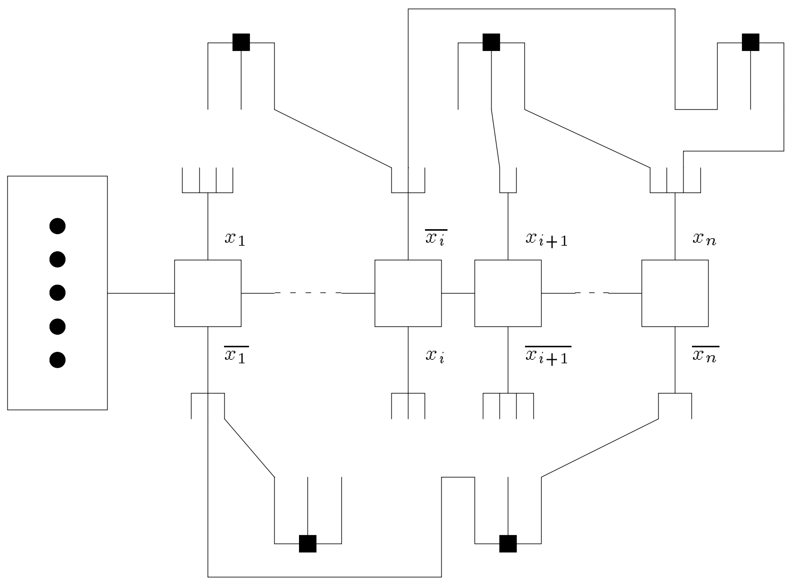

Figure 15: A representation of a P3SAT instance.

may be slid, either horizontally or vertically, into gaps. Rectangles are not allowed to overlap. Hopcroft *et al.* show that the problem of deciding whether there exists a coordinated motion of the rectangles between given initial and final configurations is PSPACE-complete. A major difference between their problem and SOKOBAN is that they have no porter. The rectangles are moved by an agent which is, in a sense, outside the system. The techniques used by Hopcroft *et al.* and the techniques used by us in the PSPACE-completeness proofs are completely different.

Papadimitriou, Raghavan, Sudan and Tamaki [PRST94] consider a natural motion planning problem on graphs. An instance of the problem is an undirected graph. There is *robot* in one of the vertices of the graph. Several other vertices contain movable *obstacles*. In each step, an object (i.e., the robot or an obstacle) may be moved along an edge into a currently unoccupied vertex. Two objects may never reside in the same vertex. The goal is to move the robot to a designated vertex, pushing obstacles out of the way, using a *minimum* number of steps. Papadimitriou *et al.* show that it is easy to decide whether the robot can at all be moved to its designated vertex (it can be decided in linear time). They also show that deciding whether the problem can be solved using at most k steps, where k is part of the input, is NP-complete. Although a 'robot' appears in the statement of this problem, its role is different from the role played by the robot, or porter, in our problem. Objects are moved by some outside agent and not by the robot. There are a few natural ways of defining versions of SOKOBAN that are played on general graphs, not necessarily on rectangular grids. Studying these versions may be interesting direction for further research.

6 Concluding remarks and open problems

SOKOBAN$^+$ is perhaps the simplest PSPACE-complete motion planning problem. The exact status of original version of SOKOBAN remains a chalanging open probelm. We have shown that it is NP-hard and that it is in PSPACE. Is it in NP? Is it PSPACE-complete?

Acknowledgement

We would like to thank Micha Sharir and Pankaj Agarwal for the relevant computational geometry references, and Michael Greene for a stimulation exchange of e-mail messages.

References

[CG94] D.A. Chalcraft and M.T. Greene. Train sets. *Eureka*, 53:5–12, 1994.

[DO92] A. Dhagat and J. O'Rourke. Motion planning amidst movable square blocks. In *Proceedings of the 4th Canadian Conference on Computational Geometry*, pages 188–191, 1992.

[FG95] M. Fryers and M.T. Greene. Sokoban. *Eureka*, 54, 1995. To appear.

[HSS84] J.E. Hopcroft, J.T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independant objects; PSPACE-hardness of the "Warehouseman's problem". *The International Journal of Robotics Research*, 3:76–88, 1984.

[Lic82] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11:329–343, 1982.

[PRST94] C.H. Papadimitriou, P. Raghavan, M. Sudan, and H. Tamaki. Motion planning on a graph (extended abstract). In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, Santa Fe, New Mexico*, pages 511–520, 1994.

[Rei79] J. Reif. Complexity of the movers' problem and generalizations. In *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science, San Juan, Puerto Rico*, pages 421–427, 1979.

[Sav70] W.J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[Ste94] I. Stewart. A subway named Turing. *Scientific American*, pages 104–107, September 1994.

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