A simple proof that PushPush PullPull F is np hard

Oscar Temprano
oscartemp@hotmail.es

Abstract

We are going to prove that solving a variant of PushPush in which we can push as well as pull is NP-hard. We are going to do this by reducing satisfiability to this problem.

1 Introduction

In this paper we are going to show that a variant of the game PushPush [1] in which the player can push as well as pull is np hard.

We will prove several interesting things along the way. Like, for instance, that this variant of the game is not reversible. We will do this by showing how to construct a no return gadget for the puzzle (see section 2.1).

By reversible we mean that if the player is standing in a position (x_1, y_1) and wants to go to another position (x_2, y_2), there is a sequence of movements to go from (x_1, y_1) to (x_2, y_2) and another one to go from (x_2, y_2) to (x_1, y_1)

In the PushPush game, the objective is to find a way to reach a particular position. There can be movable obstacles blocking the way. We can push these blocks, but once we push them, they will slide until they crash with another obstacle. We cannot push two or more blocks if they are next to each other in the same direction.

In the variant of PushPush we are studying now we can pull blocks as well as push them. Once we decide to pull a block, we must pull that block the maximum ammount possible, see [2] for more information.

2 Description of gadgets

In this section we are going to show the gadgets that we are going to use in our reduction. In the figures, a black block represents the player, brown blocks represent movable blocks and red blocks represent fixed blocks.

2.1 No return gadget

This gadget allows us to go to from a position A (x_1, y_1) to a position B (x_2, y_2) (See figure 1). In order to go to position B, we must block the way to A, so that we can´t return to the A position anymore.

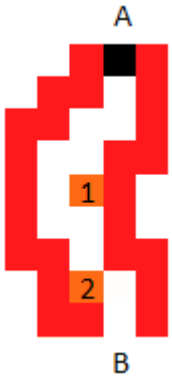

Figure 1: The no return gadget as it is in its original state.

Lemma 1. *The no return gadget can be traversed from A to B in figure 1, but after that, we cannot go back from B to A.*

Proof. First we will show how we can go from A to B.

To go to B, the first thing we observe, is that we need to get the block "2" out of the way to B . If we try to pull this block upwards (See figure 2), we will not be able to get this block out of the way to B because block "1" is not letting us pull block "2" upwards the required ammount to get block "2 out of the way to B".

Also, we observe that pushing down the block "1" or pulling up and pushing down block "1" afterwards will block the exit to B permanently.

If instead of pulling block "2" upwards, we decide to push block 1 upwards, we will "jump" to the situation described in figure 6a.

Now, we will show the movements we have to do in order to go to B if we decided to pull the block "2" upwards as our first movement (See figure 2).

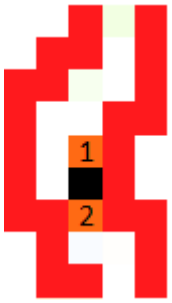

Figure 2: We moved block "2" upwards

In order to move block "2" out of the way, we must push block "1" upwards, see figure 3 (If we push block "1" downwards we will block the path to B permanently, and if we push block "2" downwards, we will return the gadget to its original state (shown in Figure 1))

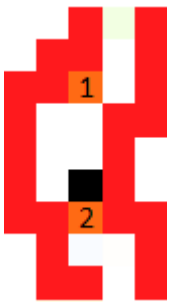

Figure 3: We pushed block "1" upwards, blocking exit A

Now we can pull block "2" upwards and go towards the exit marked with the letter "B" in figure 1 (If we push block "2" downwards we will go to the situation depicted in figure 6a; if we pull block "1" downwards we will go back to the situation depicted in Figure 2)

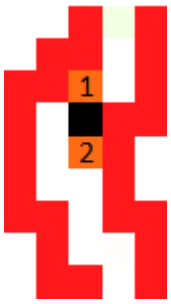

Figure 4: We moved block "2" upwards, now we can exit through B

Now, we can exit through B.

The only remaining thing to do now, is to prove that the player can´t go back from exit B to exit A

To do that, we observe that we have to move the block marked "1" downwards in order to exit through A. We observe that we cannot pull or push the block marked with the number "1", without first moving the block marked with the number "2" downwards.

If we push block "2" upwards (see figure 5), we will not be able to exit through A no more.

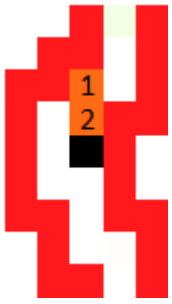

Figure 5: We pushed block "2" upwards, we cannot go back to A

The only thing we can do is push block "2" downwards as in figure 6a (pulling block "2" upwards will return us to the situation depicted in Figure 4) and then pull block "1" downwards, but because

we are forced to pull the maximum ammount possible, that will left the player "sandwiched" between the two blocks (figure 6b)

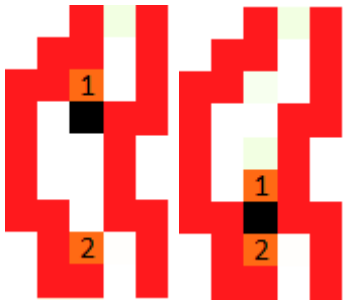

Figure 6: Figure "6a"(left) and "6b"(right). In the situation that is shown in figure 6a, the player has pushed block "2" downwards, we can pull the block "2" upwards; which leads to the situation that is shown in figure 4. Or we can pull block "1" downwards, which leads to the situation shown in figure 6b. In the figure 6b, the only movement we can do is push the block "1" upwards which would lead us to the position shown in figure 6a. In no way can the player return through exit A

Now, we can push block "1" upwards and then pull the block "2" upwards. This will return the gadget to the situation shown in figure 4. We see that the only things we can do is cycle between the situations shown in figure 4 and figure 6, or go from the situation shown in figure 4 to the situation depicted in figure 5. In none of those situations we can go back to A, with this, we finish our proof.

□

In some diagrams, we will represent the no return gadget with an arrow indicating the direction we can take through that wire.

2.2 Two way gadget

This is a gadget that allows us to go to a position and return back. Once we return, we cant go back to the position we visited previously.

This gadget is formed by two no return gadgets as it is shown in figure 7

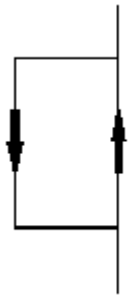

Figure 7: Two way gadget, the direction in which the arrow is pointing represents the direction we must follow to pass through the wire

This gadget is placed in two locations. One is before and after a reversible xor crossover (see section 2.3), and the other is before the player reachs a clause gadget (see section 2.5)

2.3 Reversible xor crossover

This gadget allows us to go left to right or down to up without "leaking". That means that if we enter into the gadget through the left, we must exit through the right. And if we enter the gadget from below, we must exit the gadget through the top of it. This is a reversible gadget because, if we push a block, we can pull it back to its original position.

There are two kinds of xor crossovers, one is used to prevent leakage between two literal wires, type A (figure 8); and the other is used inside the variable gadgets, type B (figure 9).

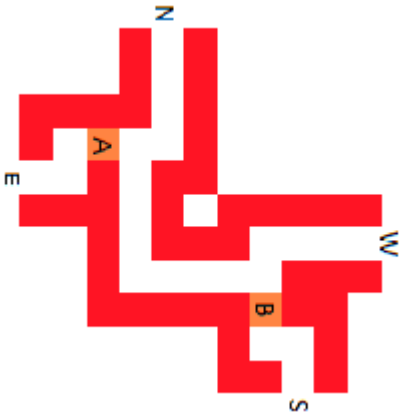

Figure 8: Reversible xor crossover (Type A), we can go "south" to "north" or "east to "west"

The difference between the two kinds of crossovers, is simply that they have different kinds of gadgets at the entrance and at the exit of the gadget. In the crossover type A, there is a two way gadget before all entries and after all the exits. In the crossover type B there is a two way gadget right before the "east" entrance and another one after the "west" exit. But there is a no return gadget before the "north" entrance and another one after the "south" exit.

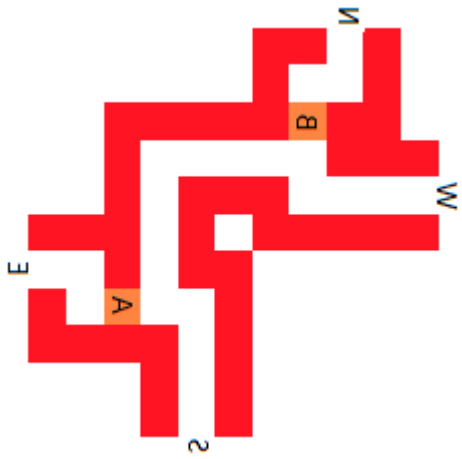

Figure 9: Reversible xor crossover (Type B), we can go "north" to "south" or "east to "west"

Lemma 2. *In the xor crossover gadgets, we can go south to north (or north to south) ,or east to west; and those are the only available traversals of the gadget*

Proof. In order to go from "south" to "north" (or "north" to "south") we must push the block marked with a letter "B", this blocks the "west" path from being traversed. Also, we cannot go to the "east" path because the block marked with the letter "A" is blocking the way. For the "east" to "west" case, we must push the block marked with an "A" blocking the access to the "south" (or "north" in the type B gadget) path. We also cannot go to the "north" (or "south" in the type B gadget) because the block marked with a "B" is blocking the way □

We can traverse the gadget "north" to "south" ("south" to "north") and "east" to "west". Or "east" to "west" and "north" to "south" ("south" to "north"). This is because we can "revert" the changes by traversing the gadget in the opposite direction (i.e. "west" to "east" instead of "east" to "west") and pull back all the blocks that have been pushed previously.

2.4 Variable gadget

This gadget represents a variable belonging to the SAT instance. The gadget is formed by a path that bifurcates left and right. The left path represents the negative literal, the right path, the positive literal. Inside each literal path there are two no return gadgets, one at entry and another at the exit. This is to make sure that we can only visit the variable wires belonging to that literal. The variable wires belonging to a negative literal are above those of a positive literal. Each literal wire is connected to a clause gadget in which the literal appears

In the positive side of the gadget, there are xor crossovers of type B to avoid leakage between literals

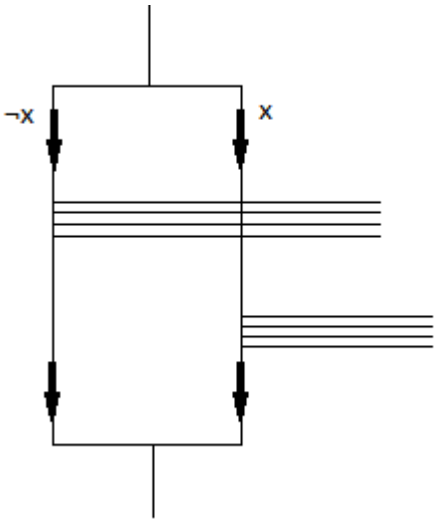

Figure 10: Variable gadget

2.5 Clause gadget

This gadget represents a clause that belongs to the SAT instance. The clause is connected to literal wires corresonding to the literals that appear in that clause. As seen in Figure 11, we enter the clause from the entrance A and we have to exit through B. In order to do that, we must have pushed the block that is at the end of a literal wire that is connected to the clause gadget (C and D

in the example shown in figure 11) and push the block at the end of the wire. Then, we can push the block at the beginning of the clause and exit through B.

There is a no return gadget just before entering the gadget from the A entrance.

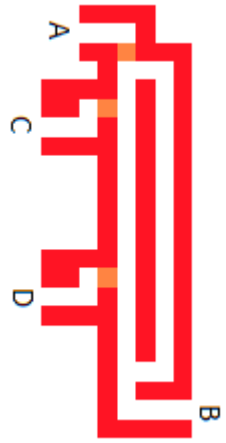

Figure 11: Clause gadget. In this example there are two literal wires going to the clause(C and D). A and B are the entrance and exit of the clause

Lemma 3. *If we haven´t been able to push any block at the end of a literal wire, we will not be able to traverse the clause gadget.*

Proof. When we enter the gadget, there is a block blocking the entrance, we have to push the block in order to pass. Because there is no other block to stop the block from sliding; the block will slide the maximum amount, closing exit B. The only thing we can do is pull the block back to its original position (see figure 11). Also we cannot go through the literal exits because there is a block at the end of the wires blocking the path. We cannot pull out the blocks at the end of the literal wires because there is not enough space to do it □

Lemma 4. *We cannot enter the clause gadget from a literal wire and we must return through the same wire we came to the clause.*

Proof. At the end of a literal wire there is a block that is blocking our way. The only movement we can do is push the block forward, but the block is still blocking the way. Now, the only movement we can do is pull the block back to its original position. Therefore, we can only return through the wire we came. □

Lemma 5. *We cannot enter to a literal wire from inside the clause gadget*

Proof. By lemma five we know that we cannot go into a literal wire that we haven´t visited previously. By lemma six we know that if we go to the clause gadget from a literal wire, we must return through that same wire. There is a two way gadget just before going to the clause from inside a literal wire. In order to go to the clause, we have to use one of the no return gadgets. Because we have to return through the same wire; we will use the other no return gadget to come back, this means that we cannot enter to the literal wire from inside the clause gadget. □

3 General view of the board

We have seen the gadgets that we are going to use. Now we are going to shown how to arrange them to create a board.

In the board, the variable gadgets are at the left of the board and the clause gadgets are at the right of the board. Each variable gadget is put above the next one, the same for the clause gadgets. For the first variable in the SAT formula, we put a variable gadget in the left and we put one clause gadget for every clause in which the variable appears. First we put a clause gadget for every clause in which there is a negated literal of that variable and then we put the clauses in which there is a positive literal of that variable. Then we connect the corresponding literals to the clauses

For the second and remaining variables, we follow this procedure:

We define two kind of clauses, "old clauses" and "new clauses". Old clauses are those that have a clause gadget already in the board and new clauses are those that dont have a clause gadget in the board yet. That is, old clauses are those that belong to a variable gadget that has been put in the board and new clauses are those in which the current variable appears but not the previous variables.

For the negated literals of the variable, we connect the wires in the variable gadget to the old clauses if there is any old clause that contains the negated literal. Then we put the variable gadgets corresponding to the new clauses (if there is any) and we connect the literals to them.

We follow the same procedure for the positive literals of the variable.

We decided to set the board this way to force us to cross through horizontal crossovers to go to new clauses and to force us to cross vertical crossovers to go to old gadgets. In this way, if we decide to not reset a xor crossover, and then we go through the crossover again from another literal, we will only be able to visit a variable gadget that we have already been visited. This is because the only way that we can reuse a crossover is by going through a literal wire that goes to an old clause. To visit a new clause we go through unused crossovers and we don´t have this problem.

Note that the two way gadgets at the beginning and at the end of the crossovers already prevent leakage between wires, so the variables and gadgets can be arranged as we want as long as they respect the connections between literals and clauses. We decided to arrange the variables and the clauses this way because we thought it looked more beatiful

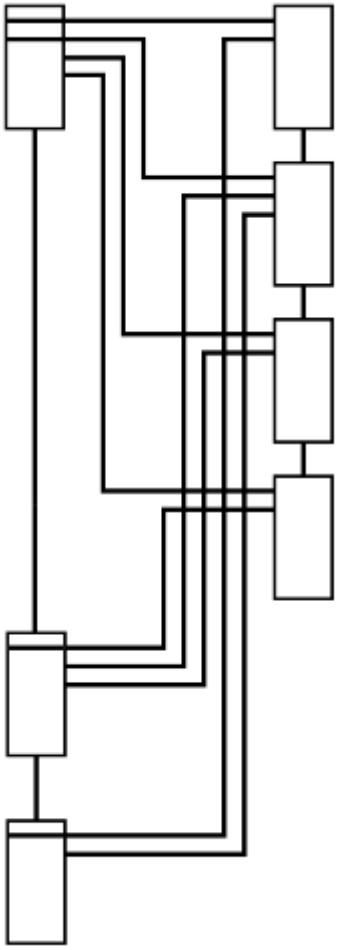

Figure 12: An example board. The variables at the left are x_1 x_2 and x_3. The clauses at the right are $(\neg x_1 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)$

4 Open questions and future directions

Prove that this variant is in NP or prove that is harder than that.

Is a variant of this problem where we can push and pull all the blocks, i.e. there are no fixed blocks, tractable?.

We observe that the no return gadget, described in section 2.1, no longer works if the player is allowed to pull the fixed blocks in the figures.

If the problem is in P, this will allow us to solve the open question of [3] about finding an interesting but tractable block-moving puzzle.

References

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

[2] Marcus Ritt *"Motion planning with pull moves"*. http://arxiv.org/abs/1008.2952

[3] Erik D. Demaine, Martin L. Demaine, Michael Hoffman and Joseph O'Rourke *"Pushing blocks is hard"*. Computational Geometry, 26:21-36, 2003