BLOXORZ

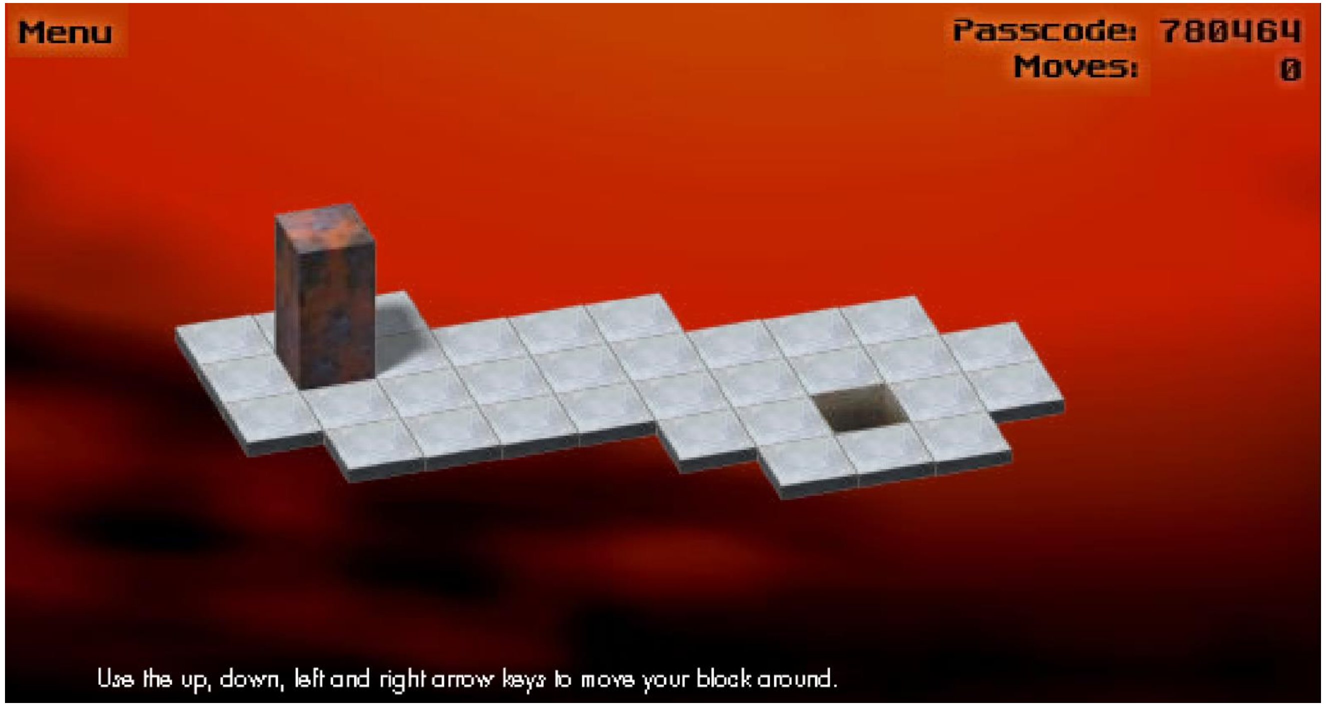
B07901020 劉昀昇

CONTENT

- Brief introduction
- Problem formulation
  - Bloxorz graph
- CNF encoding
  - Definition
  - Rules
- Experiment

BRIEF INTRODUCTION

- https://www.mathplayground.com/logic_bloxorz.html
- An online puzzle game where players move a 1 by 1 by 2 block by tilting it on a subset of the two dimensional grid. The block was moved from initial position to upright position on the destination square



BRIEF INTRODUCTION

Type of tiles

- Normal tiles
- Switches (on/off states)
- Trapdoors (close/open states) → switches and trapdoors are one-to-one mapping
- **Single-use tiles**

COMPLEXITY OF BLOXORZ

	$1 \times 1 \times 2$			$1 \times 1 \times 1$		
	Dec.	Opt.	Moves	Dec.	Opt.	Moves
-	P	P	$\Theta(n)$	P	P	$\Theta(n)$
Switches+Trapdoors	PSPACE-C	PSPACE-C	$2^{\Theta(n)}$ $\boxed{2}$	P	NP-C $\boxed{3}$	$\Theta(n^2)$
Single-use tiles	NP-C	NP-C	$\Theta(n)$	P	P	$\Theta(n)$
S+T+Single-use	PSPACE-C	PSPACE-C	$2^{\Theta(n)}$ $\boxed{2}$	NP-C $\boxed{4}$	NP-C	$\Theta(n^2)$

Table 2: Complexity of various Bloxorz variants

Van Der Zanden, Tom C., and Hans L. Bodlaender. "PSPACE-completeness of Bloxorz and of games with 2-buttons." *arXiv preprint arXiv:1411.5951* (2014).

- Dec. : decision problem
- Opt. : optimization problem, whether it takes less than k steps
- **Here we focus on single-use tiles problem, it can be reduced from 3-CNF (NPC)**

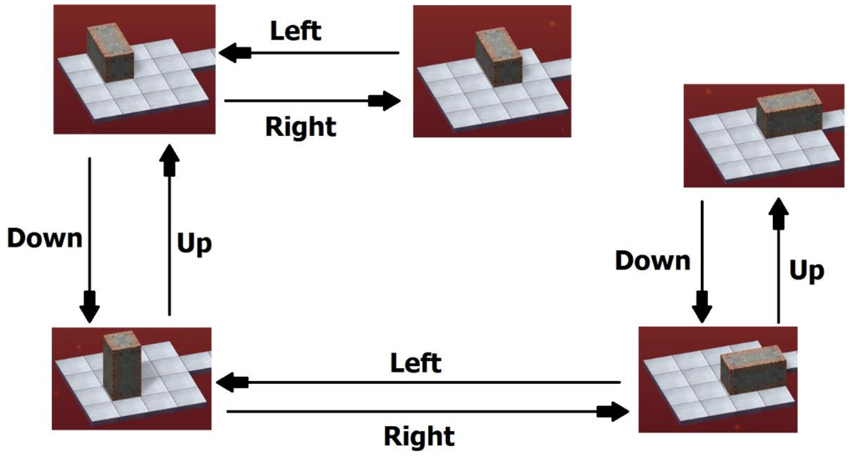
PROBLEM FORMULATION

- Instance

  Given a level of Bloxorz graph made up by a list of single-use tiles, and a 1 by 1 by 2 block

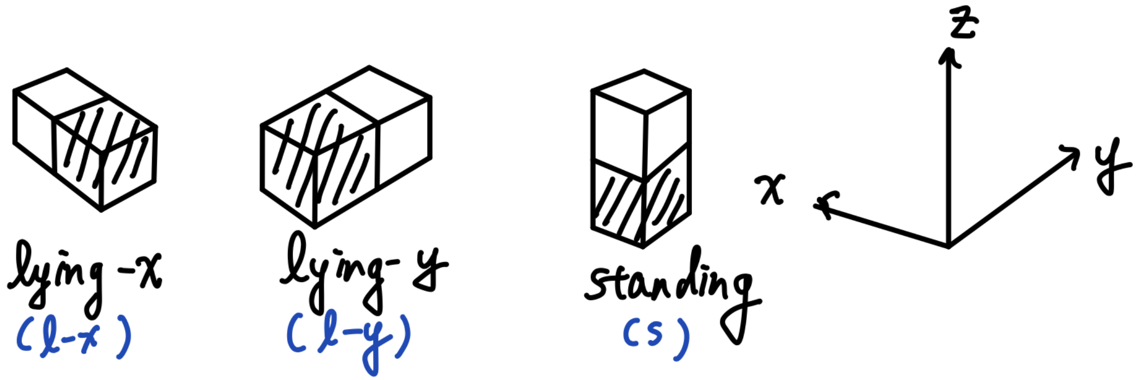
- Question

  Is there a sequence of **legal move** (**right, left, down, up**) to move from start point to up-right position on goal square



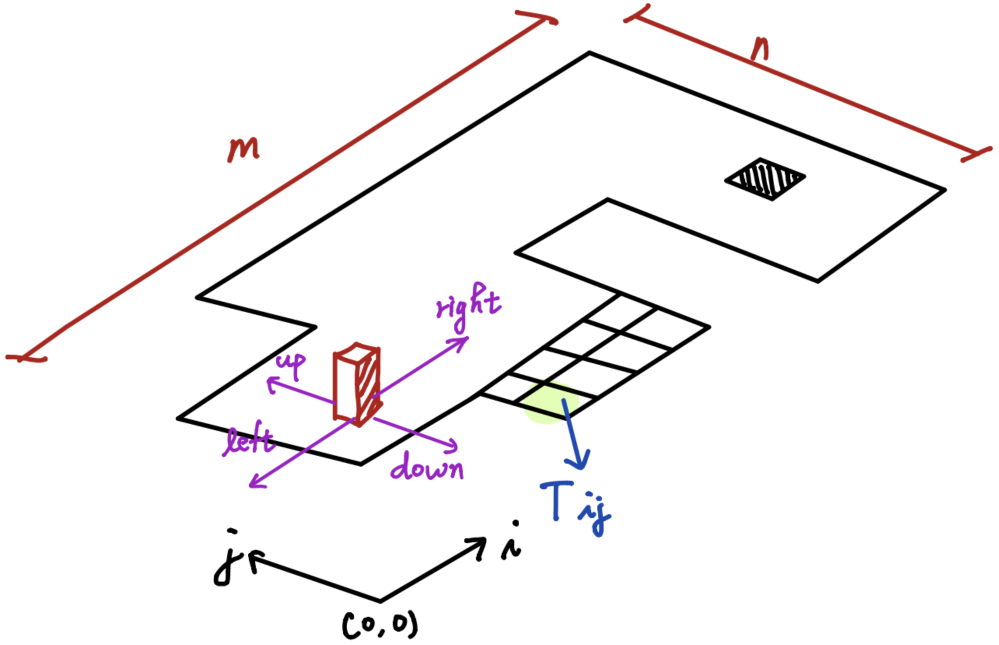
BLOXORZ GRAPH

- Given a level of Bloxorz, suppose it has total $\mathbb{T}$ tiles. For each tile the block can have three unique states on it (choose the closer-to-origin unit). There are total $3 \cdot |\mathbb{T}|$ vertices.



BLOXORZ GRAPH

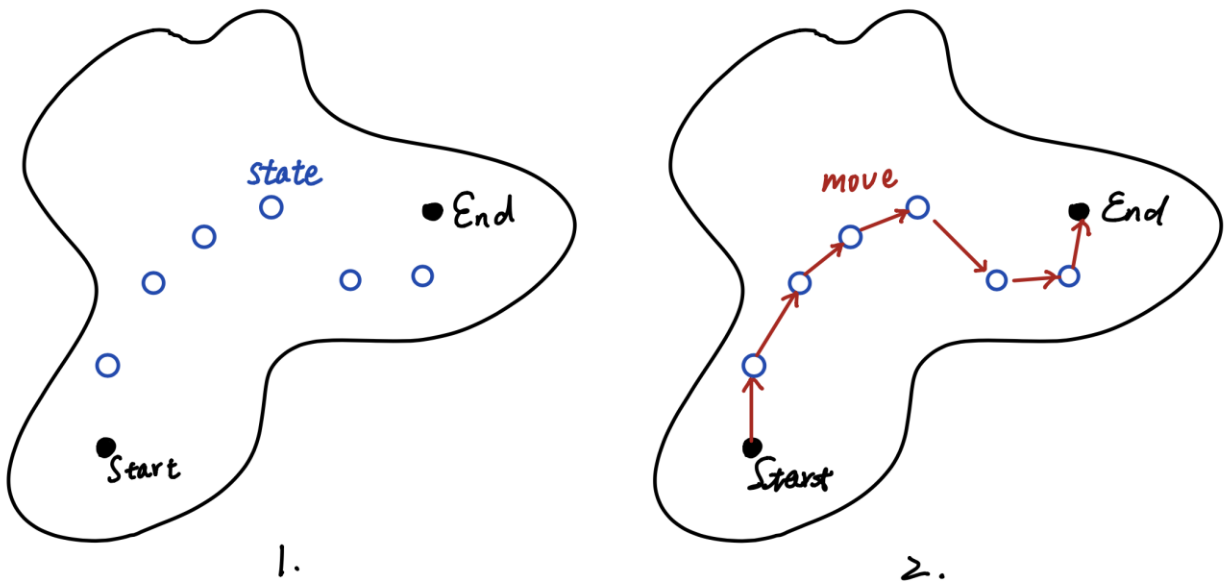
- Edges (v_1, v_2) means that three is a legal move from vertex v_1 to v_2
- For each vertex, only four edges are connected to other vertices, total edge count is at most $4 \cdot 3 \cdot |\mathbb{T}|$
- Transformation between Bloxorz game and Bloxorz graph is linear to map size



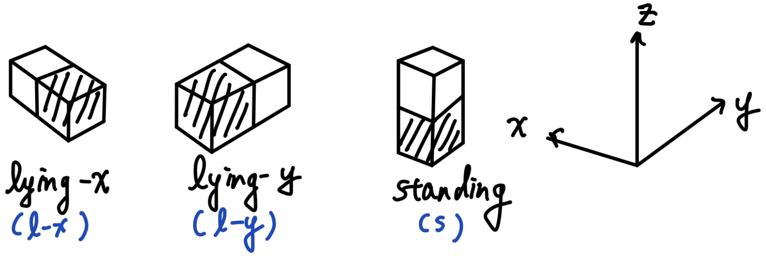
CNF ENCODING

OVERALL IDEA

1. Select a bunch of nodes on Bloxorz graph (these node incorporate location and state info of the block)
2. Check whether there exist a set of moves (edge) assigned on each selected nodes, to achieve path connection from start to end



DEFINITION



- Set of block states $\mathbb{A} = \{L_x, L_y, S\}$
- Define state variable b_{ija}, linear growth to map size

$$b_{ija,\, a\in\mathbb{A}} = \begin{cases} 1, & block\ at\ position\ (i,j), and\ its\ state\ is\ a \\ 0, & otherwise \end{cases}$$

- legal move $\mathbb{M} = \{left, right, down, up\}$
- Define move variable $move_{ijam}$, linear growth to mapsize

$$move_{ijam,\ m\in\mathbb{M}} = \begin{cases} 1, & Given\ orignal\ block's\ state\ b_{ija}, take\ a\ legal\ move\ m \\ 0, & no\ move\ is\ taken \end{cases}$$

RULES - SELECT A BUNCH OF STATES b_{ija}

- Since a tile can not be passed twice, a whenever a state is taken, some of states are no longer available
- Block is originally stand at start point, and it must end standing at goal hole, these two states must be 1

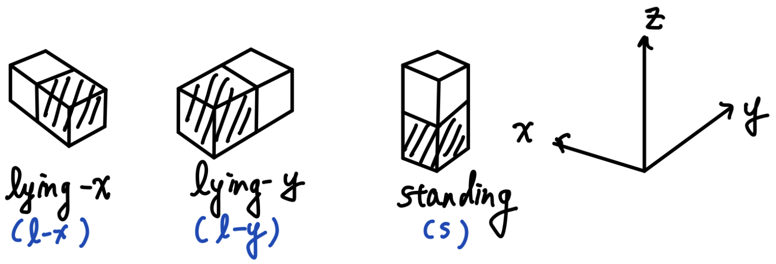
Total clause number: O(map size)

$$\bigwedge_{i\in[m],j\in[n]}\left(b_{ijL_x} \longrightarrow \bigwedge_{\substack{a\in A\\ a\neq L_x}}\left(\overline{b_{ija}}\right) \bigwedge_{a\in A}\left(\overline{b_{i+1ja}}\right)\right)$$

$$\bigwedge_{i\in[m],j\in[n]}\left(b_{ijL_y} \longrightarrow \bigwedge_{\substack{a\in A\\ a\neq L_y}}\left(\overline{b_{ija}}\right) \bigwedge_{a\in A}\left(\overline{b_{ij+1a}}\right)\right)$$

$$\bigwedge_{i\in[m],j\in[n]}\left(b_{ijS} \longrightarrow \bigwedge_{\substack{a\in A\\ a\neq S}}\left(\overline{b_{ija}}\right)\right)$$

$$b_{i_{start}j_{start}S}=1,\ b_{i_{end}j_{end}S}=1$$



RULES – CHOOSE A SET MOVES

- A state must have only one outgoing move, except end state
- If there is no moves outgoes from state b_{ija}, then b_{ija} must not be chosen, except end state

Total clause number: O(map size)

$$\bigwedge_{i\in[m]/i_{end},\, j\in[n]/j_{end},\, a\in A} \left\{ b_{ija} \leftrightarrow \bigvee_{m\in M}\left(move_{ijam}\right)\right\}$$

$$\bigwedge_{m\in M}\left(\overline{move_{i_{end}j_{end}am}}\right)$$

$$\bigwedge_{i\in[m],\, j\in[n],\, a\in A}\left(\bigwedge_{m_1,m_2\in M,\, m_1\neq m_2}\left(\overline{move_{ijam_1}} \vee \overline{move_{ijam_2}}\right)\right)$$

RULES – SUCCESSOR FUNCTION

Now, we have a set of k states, and $k - 1$ moves, each state has one outgoing move expect end node

So far, outgoing and ingoing edges are not restricted on the path

→ **How to ensure these states and moves can form a single path with no sub-cycle, no branch**

Reference : SAT Encoding for the Hamiltonian Cycle Problem

RULES – SUCCESSOR FUNCTION

Definition: $\mathcal{U}_{ijap} = 1$ iff b_{ija} 's position is $p, p \in 1,2 \ \dots mapsize$

Variable count : O(map size^2)

- One to one mapping between b_{ija} and U_{ijap}, that is for given $b_{ija} = 1$, a unique p exists such that $\mathcal{U}_{ijap} = 1$
- If $b_{ija} = 0$, then $\mathcal{U}_{ijap} = 0$ for all p

Total clause number: O(map size^3), "unique p" is costly

$$\bigwedge_{ija} \left(\overline{b_{ija}} \vee \left(\bigvee_{p=1}^{mapsize} \left(\mathcal{U}_{ijap} \right) \right) \right)$$

$$\boxed{\bigwedge_{ija} \left(\bigwedge_{p_1 \neq p_2} \left(\overline{\mathcal{U}_{ijap_1}} \vee \overline{\mathcal{U}_{ijap_2}} \right) \right)}$$

$$\bigwedge_{ija} \bigwedge_{p=1}^{mapsize} \left(\overline{\mathcal{U}_{ijap}} \vee b_{ija} \right)$$

RULES – SUCCESSOR FUNCTION

P's continuality

- Each p maps to at most one state, it is costly → **Do we need this ?**

Total clause count: O(map size^3)

It accounts for more than half of total clause number

$$\bigwedge_{p=1}^{mapsize} \left(\bigwedge_{i_1 \neq i_2,\, j_1 \neq j_2,\, a_1 \neq a_2} \left(\overline{U_{i_1 j_1 a_1 p}} \vee \overline{U_{i_2 j_2 a_2 p}} \right) \right)$$

RULES – SUCCESSOR FUNCTION

Implication

- Initialization of start position $p = 0$
- Recursive implication
- Is ending condition needed ? → No, since end state has no outgoing move, so implication can not proceed

Total clause count : O(map size)

$$U_{i_{start} j_{start} S\, 0} = 1$$

$$\underset{(i_1 j_1) \neq End}{U_{i_1 j_1 a_1 p}} \wedge \underset{m \in M}{move_{i_1 j_1 a_1 m}} \longrightarrow U_{i_2 j_2 a_2\, p+1}$$

RULES – SUCCESSOR FUNCTION

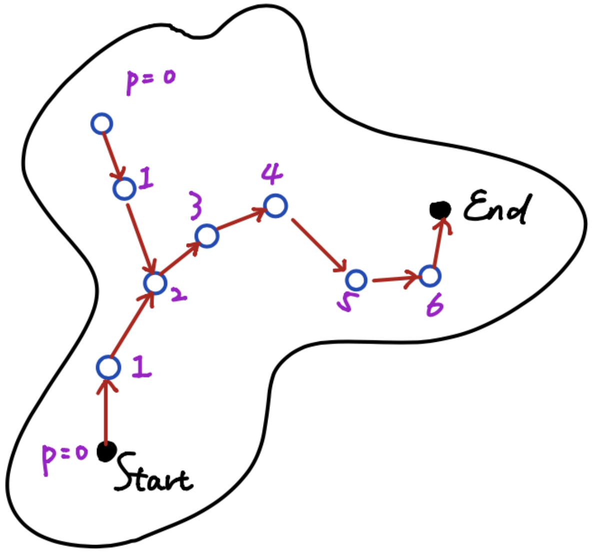
Combine **P's continuality** and **Implication**, we can ensure a state's both outgoing and ingoing moves are on the path from start to end, it also prevent branch and sub-cycle

Do we need to make sure that each p maps to at most one state ?

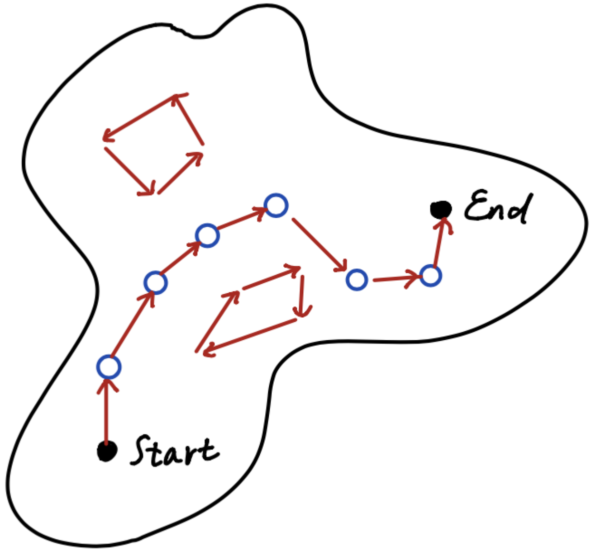
→ If we remove this constraint, **branch** may occur

→ Restricted every state's has at most one ingoing move? It is cheaper, since the graph is sparse (degree <= 4), it take O(map size) clauses

→ Or, just make sure there is only one starting point with $p = 0$



Branch



Sub-cycle

EXPERIMENT

Map1

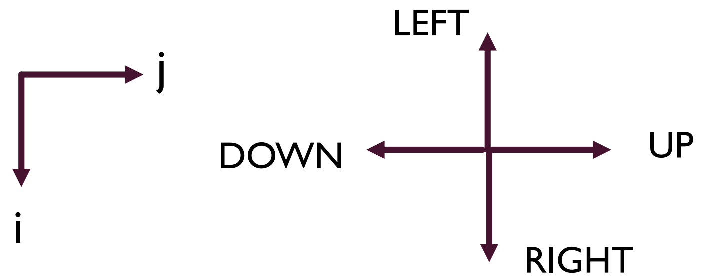
```
011110
110311
110111
010111
110111
000211
```

```
============================[MINISAT]============================
| Conflicts |      ORIGINAL      |          LEARNT          | Progress |
|           | Clauses Literals | Clauses Literals   Lit/Cl |          |
=================================================================
|         6 |   21270    45150 |        3        8     2.7 |  0.000 % |
=================================================================
SAT

state U_5_3_S_0
                      step 0 : UP
state U_5_4_Ly_1
                      step 1 : LEFT
state U_4_4_Ly_2
                      step 2 : LEFT
state U_3_4_Ly_3
                      step 3 : LEFT
state U_2_4_Ly_4
                      step 4 : LEFT
state U_1_4_Ly_5
                      step 5 : DOWN
state  U_1_3_S_6

Total memory usage: 5980
Total time usage: 0.036443 s
```

Map2

```
0010
1310
0111
1111
1111
0210
1110
```

```
==============================[MINISAT]==============================
| Conflicts |      ORIGINAL      |          LEARNT          | Progress |
|           | Clauses Literals | Clauses Literals   Lit/Cl |          |
=====================================================================
|         0 |   14810    31473 |        0        0    -nan |  0.000 % |
=====================================================================
UNSAT


Total memory usage: 5104
Total time usage: 0.017157 s
```



EXPERIMENT

Map3

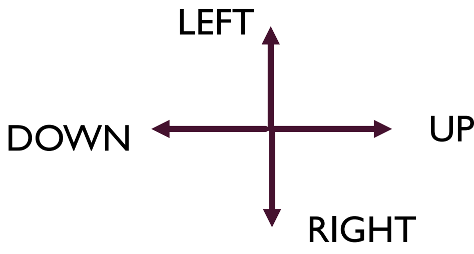
```
000000000000000
000110000000000
001311000000000
001111100000000
000111110000000
000110111111000
000110001111000
000110000011000
000110000110000
001101111110000
011111111111000
011111000111100
001110000121100
000000000011000
000000000000000
```

```
map boundary (_m, _n) = (15, 15), size = 75
starting index = (12, 10)
ending index   = (2, 3)

nodes size     = 225
moves size     = 900
unary var size = 16875

=====  Begin to solve  =====
============================[MINISAT]============================
| Conflicts |      ORIGINAL      |          LEARNT         | Progress |
|           | Clauses Literals | Clauses Literals  Lit/Cl |          |
=================================================================
|        86 |  519031  1067205 |      86      316     3.7 |  0.000 % |
=================================================================
SAT

state U_12_10_S_0
                        step 0 : LEFT
state U_10_10_Lx_1
                        step 1 : DOWN
state U_10_9_Lx_2
                        step 2 : LEFT
state U_9_9_S_3
                        step 3 : DOWN
state U_9_7_Ly_4
                        step 4 : RIGHT
state U_10_7_Ly_5
                        step 5 : DOWN
state U_10_6_S_6
                        step 6 : DOWN
state U_10_4_Ly_7
                        step 7 : DOWN
state U_10_3_S_8
                        step 8 : LEFT
state U_8_3_Lx_9
                        step 9 : LEFT
state U_7_3_S_10
                        step 10 : LEFT
state U_5_3_Lx_11
                        step 11 : LEFT
state U_4_3_S_12
                        step 12 : UP
state U_4_4_Ly_13
                        step 13 : LEFT
state U_3_4_Ly_14
                        step 14 : LEFT
state U_2_4_Ly_15
                        step 15 : DOWN
state  U_2_3_S_16

Total memory usage: 46608
Total time usage: 0.777054 s
```

Map4

```
00000000000000000000
00000000000000000000
00000111111111111000
00001111311111111100
00001110111011011100
00000110011111011100
00000111111111111100
00001111110000111100
00011111100000011100
00111111000000011100
00111000000000111100
00110000000111110110
00110000001111111110
00110000011110011110
00111001111110011110
01111111110111111100
01111111111121111000
00111110011100000000
00000000000000000000
00000000000000000000
```

```
============================[MINISAT]============================
| Conflicts |      ORIGINAL      |          LEARNT         | Progress |
|           | Clauses Literals | Clauses Literals  Lit/Cl |          |
=================================================================
|       177 | 7439236 15089283 |     177     1019     5.8 | 15.763 % |
=================================================================
SAT

state U_16_12_S_0
                        step 0 : UP
state U_16_13_Ly_1
                        step 1 : LEFT
state U_15_13_Ly_2
                        step 2 : UP
state U_15_15_S_3
                        step 3 : UP
state U_15_16_Ly_4
                        step 4 : LEFT
state U_14_16_Ly_5
                        step 5 : LEFT
state U_13_16_Ly_6
                        step 6 : LEFT
state U_12_16_Ly_7
                        step 7 : DOWN
state U_12_15_S_8
                        step 8 : LEFT
state U_10_15_Ly_9
                        step 9 : LEFT
state U_9_15_Ly_10
                        step 10 : LEFT
state U_8_15_Ly_11
                        step 11 : LEFT
state U_7_15_Ly_12
                        step 12 : LEFT
state U_6_15_Ly_13
                        step 13 : DOWN
state U_6_14_S_14
                        step 14 : DOWN
state U_6_12_Ly_15
                        step 15 : DOWN
state U_6_11_S_16
                        step 16 : DOWN
state U_6_9_Ly_17
                        step 17 : LEFT
state U_5_9_Ly_18
                        step 18 : LEFT
state U_4_9_Ly_19
                        step 19 : LEFT
state U_3_9_Ly_20
                        step 20 : DOWN
state  U_3_8_S_21

Total memory usage: 528712
Total time usage: 10.2323 s
```



EXPERIMENT

M5

```
0000000000000000000000000
0000011111100000000000000
0011111011113110111110000
011111000111011110111000
0111001111100000001111000
0011100110000000000111000
0001111100110000000110000
0001101111111100000110000
0001101111111111000110000
0011111000001011100111000
0011111111001101111111100
0000000111001111111111100
0000000011111111110111000
0000000011111100110110000
0000000001111111101100000
0001000000100111111100000
0011100001110011111110000
0001111111111011100111000
0001111111111111000011100
0011111100111110000011100
0011111111111110000111100
0011110011111111001111000
0000000001111111121100000
0000000000000000011000000
0000000000000000000000000
```

```
map boundary (_m, _n) = (25, 25), size = 299
starting index = (22, 17)
ending index   = (2, 5)

nodes size     = 897
moves size     = 3588
unary var size = 268203

=====  Begin to solve  =====
============================[MINISAT]============================
| Conflicts |     ORIGINAL     |          LEARNT          | Progress |
|           | Clauses Literals | Clauses Literals  Lit/Cl |          |
=================================================================
|   2352962 | 31369287 63275793 |  800163 35286400    44.1 | 16.919 % |
=================================================================
UNSAT


Total memory usage: 2810064
Total time usage: 18511.1 s
```

M6

```
0000000000000000000000000
0000000001000000000000000
0011000011100011111110000
0111111111100011111111000
0111111101101113111110000
0011011100111111111110000
0011100011111100111011000
0001110111111000110011000
0000111111111000110011100
0000011111111111100111000
0000011111111111100110000
0000111111100111111100000
0001111111100011111100000
0001111111111111111100000
0001111111111111101100000
0001110111001110001100000
0001100011000111001100000
0001111111001110001100000
0001111111011100001100000
0011111111111100011101100
0011111111111000011111000
0001111100011111111111000
0000111000001112111110000
0000000000000000000000000
0000000000000000000000000
```

```
============================[MINISAT]============================
| Conflicts |     ORIGINAL     |          LEARNT          | Progress |
|           | Clauses Literals | Clauses Literals  Lit/Cl |          |
=================================================================
|       314 | 35802368 72252024 |     313     1336     4.3 | 14.443 % |
=================================================================
SAT

state U_22_15_S_0
                        step 0 : UP
state U_22_16_Ly_1
                        step 1 : LEFT
state U_21_16_Ly_2
                        step 2 : UP
state U_21_18_S_3
                        step 3 : UP
state U_21_19_Ly_4
                        step 4 : LEFT
state U_20_19_Ly_5
                        step 5 : DOWN
state U_20_18_S_6
                        step 6 : LEFT
state U_18_18_Ly_7
                        step 7 : LEFT
state U_17_18_Ly_8
                        step 8 : LEFT
state U_16_18_Ly_9
                        step 9 : LEFT
state U_15_18_Ly_10
                        step 10 : LEFT
state U_14_18_Ly_11
                        step 11 : LEFT
state U_13_18_Ly_12
                        step 12 : LEFT
state U_12_18_Ly_13
                        step 13 : LEFT
state U_11_18_Ly_14
                        step 14 : DOWN
state U_11_17_S_15
                        step 15 : DOWN
state U_11_15_Ly_16
                        step 16 : DOWN
state U_11_14_S_17
                        step 17 : LEFT
state U_9_14_Ly_18
                        step 18 : UP
state U_9_16_S_19
                        step 19 : LEFT
state U_7_16_Ly_20
                        step 20 : LEFT
state U_6_16_Ly_21
                        step 21 : LEFT
state U_5_16_Ly_22
                        step 22 : DOWN
state U_5_15_S_23
                        step 23 : DOWN
state U_5_13_Ly_24
                        step 24 : LEFT
state U_4_13_Ly_25
                        step 25 : UP
state U_4_15_S_26
                        step 26 : LEFT
state U_2_15_Ly_27
                        step 27 : UP
state U_2_17_S_28
                        step 28 : RIGHT
state U_3_17_Ly_29
                        step 29 : RIGHT
state U_4_17_Ly_30
                        step 30 : DOWN
state  U_4_16_S_31

Total memory usage: 2376840
Total time usage: 49.5189 s
```

IMPROVEMENT

- How about do not maintain successor property, just make sure that every node has one input and output edge

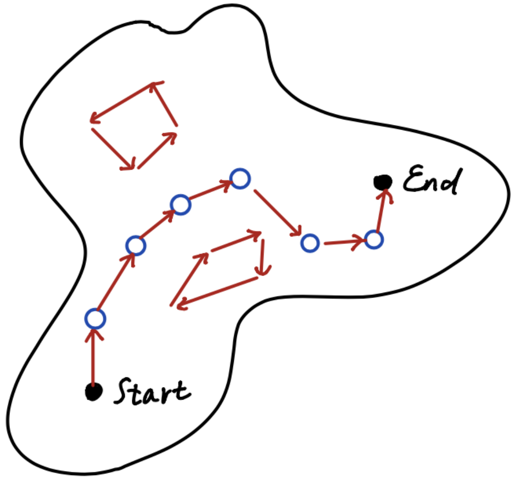
→ Sub-cycles may occur

- New requirement

  Each node except start and end node has one output edge to selected nodes and one input edge from selected nodes

  Start node has no edge input, end node has no edge output

- Using while loop to construct the path from start point to end point



EXPLAATION

A bloxorz map is reachable from input to output

Iff

The CNF formula is satisfiable

“→” The connected single path satisfies all CNF constraint.

“←” Start from start node, since for each non-terminal node u, it has one and only one edge (u, v) output, and v is also selected. Because every node except start node can has one and only one input edge (start node has no input), v can not be the traversed nodes, this recursion must end at end point, which has no edge output. To satisfies these constraints, there must be a simple path from start node to the end node

COMPARISON

	Map1(6x6)	Map2(7x4)	Map3(15x15)	Map4(20x20)	Map5(25x25)	Map6(25x25)
	SAT	UNSAT	UNSAT	SAT	UNSAT	SAT
Clauses	21270	14810	463310	7439236	31369287	35802368
	613	881	1894	7135	10454	13319
Literals	45150	31473	952257	15079283	63275793	72252024
	1540	2036	4739	17615	25872	32630
Conflicts	6	0	139	177	2352962	314
	0	0	48	1039	25	39
Memory use	5980	5104	43440	528712	2810064	2376840
	6836	6836	6836	6836	6836	6836
Time use (s)	0.036	0.17	0.74	10.23	18511	49.5
	0.0025	0.0017	0.007	0.070	0.027	0.029

: Encode without successor function : Encode with successor function

FURTHER TESTING

Improvement	Map7(50x50)	Map8(50x50)	Map9(70x70)	Map10(70x70)	Map11(100x100)	Map12(100x100)
	SAT	UNSAT	SAT	SAT	SAT	SAT
Clauses	36787	35665	89543	91104	167413	190554
Literals	91181	88416	221206	225023	415220	470979
Conflicts	1517	22991	1801428	15387	3374571	582
Memory use	10796	12560	209988	21020	352504	37160
Time use (s)	0.21	2.41	1059.1	1.68	3932.6	0.64