

# SAT Project Report – OX Arrangement Problem

b07901135 EE3 何國瑋

2021-05-02

## 1 Problem Definition

Given  $n$  Xs and  $n$  Os that are alternately arranged in  $2n$  slots ( $n \geq 3$ ). The Goal is to rearrange them with  $n$  moves so that all Xs and Os are grouped together respectively. 2 groups must also be adjacent to each other.

For example, "XOXOXO" has to become "OOOXXX" with only 3 moves.

A Legal move is defined as the following:

- A O/X has to be moved with one of its neighbors.
- The two O/Xs can only be moved to two adjacent empty slots.
- After they are moved, their original slots become empty.
- The total number of slots is unlimited.

### 1.1 Input

The Input of the problem is a number  $n$ , which represents  $n$  pairs of "XO" in the beginning.

### 1.2 Output

The output is a sequence of moves with length  $n$ .

Let  $s_k$  denotes the  $k^{th}$  slots. A move that moves two O/Xs from  $s_i$  and  $s_{i+1}$  to  $s_j$  and  $s_{j+1}$  is denoted by

$$i \rightarrow j$$

## 2 SAT Model

Although there're solutions for  $n = 3$ , The folloing model only focus on solving  $n > 3$ .

To model this problem, the number of slots is limited to  $2n + 2$ . The initial states is the sequence "XO...XOE", and the final states is the sequence "EEO...OX...X" where E denotes the empty slots.

Consider a single move between two states, the following variables are defined:

$$\begin{array}{lll} O_0, O_1, \dots, O_{2n+1} & X_0, X_1, \dots, X_{2n+1} & E_0, E_1, \dots, E_{2n+1} \\ O'_0, O'_1, \dots, O'_{2n+1} & X'_0, X'_1, \dots, X'_{2n+1} & E'_0, E'_1, \dots, E'_{2n+1} \\ T_0, T_1, \dots, T_{2n} & Y_0, Y_1 & \end{array}$$

Where

- $O_i/X_i/E_i$  denotes whether  $s_i$  is O/X/Empty before the move. Exactly one of these three variables will be true.
- $O'_i/X'_i/E'_i$  denotes whether  $s_i$  is O/X/Empty after the move. Exactly one of these three variables will be true.
- $T_i$  denotes that  $s_i$  and  $s_{i+1}$  is to be moved. Since there's only 2 (adjacent) empty slots at any time, there's only one choice for the destination. Exactly one among all  $2n + 1$  variables will be true.
- $Y_0$  and  $Y_1$  denotes whether the two O/Xs to be moved is O or not.

For  $n$  moves,  $n + 1$  states are needed. Therefore, the variables used in CNF formula includes

$$\begin{array}{ll} O_i^m, X_i^m, E_i^m & m \text{ denotes the } m^{th} \text{ state. } \quad 0 \leq m < n + 1, \quad 0 \leq i < 2n + 2 \\ T_i^m, Y_0^m, Y_1^m & m \text{ denotes the } m^{th} \text{ move. } \quad 0 \leq m < n, \quad 0 \leq i < 2n + 1 \end{array}$$

## 2.1 Basic Constraints

### 2.1.1 Exactly One

The "Exactly One" constraints on a set of variables  $\{v_1, v_2, \dots, v_k\}$  are translated to

$$(\sum_{i=1}^k v_i) \wedge \prod_{i=1}^{k-1} (\prod_{j=i+1}^k (\bar{v}_i \vee \bar{v}_j))$$

in the CNF formula.

For each states  $m$  and each  $i$ , exactly one of  $\{O_i^m, X_i^m, E_i^m\}$  is true.

For each move  $m$ , exactly one of  $\{T_0^m, T_1^m, \dots, T_{2n}^m\}$  is true.

### 2.1.2 Initial and Final States

The variables  $O$ ,  $X$ , and  $E$  in the initial states and the final state are directly assigned to desired values.

### 2.1.3 Valid Move between 2 States

During a move from the  $m^{th}$  state to the  $m+1^{th}$  state, the following terms are constructed to ensure a valid move (the superscript  $m$  is ignored and the superscript  $m+1$  is denoted by  $'$  for simplicity)

The term for take rule is

$$\bar{T}_i \vee (\bar{E}_i \bar{E}_{i+1} E'_i E'_{i+1} (\bar{Y}_0 \vee O_i)(Y_0 \vee X_i)(\bar{Y}_1 \vee O_{i+1})(Y_1 \vee X_{i+1}))$$

The term for place rule is

$$\bar{T}_i^{m-1} \vee (E_i E_{i+1} \bar{E}'_i \bar{E}'_{i+1} (\bar{Y}_0 \vee O'_i)(Y_0 \vee X'_i)(\bar{Y}_1 \vee O'_{i+1})(Y_1 \vee X'_{i+1}))$$

And the term for maintaining the value of unmoved slots is

$$T_i^{m-1} \vee T_{i-1}^{m-1} \vee T_i \vee T_{i-1} \vee (\bar{O}_i \vee O'_i)(\bar{E}_i \vee E'_i)(\bar{X}_i \vee X'_i)$$

The above three terms can be translated to clauses by simply applying distribution law to the last product terms.

## 2.2 Additional Constraints and Modification

The following constraints and modifications are found by observing the generated solutions.

### 2.2.1 Number of Slots

There's always a solution when the number of slots is restricted to  $2n+2$ . Also, with this restriction, the destination of each move has only one choice, so the model can be simplified.

### 2.2.2 The Second Last State

The second last state always looks like the following

$$XXO \dots OX \dots XEEX$$

So the last state and the last move is removed from the SAT solver.

### 2.2.3 No Duplicated Moves

A move on at slot  $s_i$  will happen at most once. That is, among  $\{T_i^0, T_i^1, \dots, T_i^{n-2}\}$ , at most one variable is true. The constraint is translated to

$$\Pi_{m=0}^{n-3} (\Pi_{k=m+1}^{n-2} (\bar{T}_i^m \vee \bar{T}_i^k))$$

### 2.2.4 The Pattern of the O/X Moved

For the first  $\lceil n/2 \rceil$  moves, the pattern is "OX, XO, OX, ...".

For the last  $\lfloor n/2 \rfloor$  moves, the pattern is "..., XX, OO, XX".

Therefore, the value of  $Y_0$  and  $Y_1$  in each move can be assigned directly.

### 2.2.5 Putting Os and Xs Together

Except the first move and the  $\lceil (n+1)/2 \rceil^{th}$  move, all moves take the O/Xs with different neighbors, and put them next to correct neighbors. So, the following two terms can be added to the take rule and place rule mentioned before.

$$\begin{aligned} & \overline{T}_i \vee ((\overline{Y}_0 \vee X_i)(Y_0 \vee O_i)(\overline{Y}_1 \vee X_{i+1})(Y_1 \vee O_{i+1})) \\ & \overline{T}_i^{m-1} \vee ((\overline{Y}_0 \vee O'_{i-1})(Y_0 \vee X'_{i-1})(\overline{Y}_1 \vee O'_{i+2})(Y_1 \vee X'_{i+2})) \end{aligned}$$

## 3 General Solution

With the constraints found and the pattern of solutions, though didn't expected, a general solution to this problem can be derived in linear time.

First, the sequence is partitioned into the following form (E denotes the empty slots)

$$XOXO \ XOXO \ \dots \ OXOX \ OXOX \ OEE$$

For a problem with input size  $n = 4k + c$ ,  $0 \leq c < 4$ ,

In the first  $2k - 1$  moves, alternately perform the following two moves

- Move the "OX" in "the left most untouched group" to the empty slots.
- Move the "XO" in "the right most untouched group" to the empty slots.

After  $2k - 1$  moves, the whole sequence will look like this

$$XXOO \ XXOO \ \dots \ OOX \ OOX \ OOX$$

Then, in the middle of the sequence, there're 4 possible cases. For these 4 cases, apply the following moves (The symbol "/" denotes the boundary between Os and Xs at the end)

For  $c = 0$ , apply 1 move

$$\begin{aligned} & XEEO \ XO/X \\ & XXOO \ EE/X \end{aligned}$$

For  $c = 1$ , apply 2 moves

$$\begin{aligned} & XEEO \ XOX/O \ X \\ & XXOO \ XOE/E \ X \\ & XXOE \ EOO/X \ X \end{aligned}$$

For  $c = 2$ , apply 3 moves

$$\begin{aligned} & XEEO \ XOXO/ \ XOX \\ & XXOO \ EEXO/ \ XOX \\ & XXOO \ OXXO/ \ XEE \\ & XXOO \ OEEO/ \ XXX \end{aligned}$$

For  $c = 3$ , apply 4 moves

$$XEEO \ XOXO \ X/OXO \ X$$

$XXOO \quad XOXO \quad E/EXO \quad X$   
 $XXOO \quad XEEO \quad O/XXO \quad X$   
 $XXOO \quad XXOO \quad O/XEE \quad X$   
 $XXOO \quad EEOO \quad O/XXX \quad X$

Finally, after the above moves, there're  $k$  pairs of "XX" at the LHS of the sequence and  $k$  pairs of "OO" at the RHS of the sequence.

For the next  $2k$  moves, alternately move all the "OO" to the left and all the "XX" to the right. Then the whole process is completed.

## 4 Result of SAT Solver

The model described at the beginning is actually a version with a few modifications already. The original version has more empty slots on both sides and have an additional variable  $P$  to denote the destination slots. In that version, the SAT solver can only solve the problem up to  $n = 16$  within a few minutes.

After the modifications and improvements mentioned in section 2.2, the SAT solver can solve the problem up to  $n = 150$  in reasonable time. before the general solution is found.

```

=====MINISAT=====
| Conflicts | ORIGINAL | LEARNT | Progress | | | |
|           | Clauses  | Literals | Clauses  | Literals | Lit/cl |
|=====|=====|=====|=====|=====|=====|
|         0 |   72381  | 175780 |         0 |         0 |   -nan | 0.000 % |
=====
SAT
=====MINISAT=====
| Conflicts | ORIGINAL | LEARNT | Progress | | | |
|           | Clauses  | Literals | Clauses  | Literals | Lit/cl |
|=====|=====|=====|=====|=====|=====|
|        59 |   44191  | 108747 |        59 |        609 |   10.3 | 0.000 % |
=====

11 --> 48 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
22 --> 11 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
7 --> 22 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
18 --> 7 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1 --> 18 X XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
40 --> 1 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
43 --> 40 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
14 --> 43 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
27 --> 14 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
32 --> 27 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
35 --> 32 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
4 --> 35 XXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
36 --> 4 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
15 --> 36 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
44 --> 15 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
6 --> 44 XXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
28 --> 6 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
10 --> 28 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
31 --> 10 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
19 --> 31 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
39 --> 19 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
23 --> 39 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
47 --> 23 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
0 --> 47 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
=====

```

Figure 1: An example output of  $n = 24$

The runtime(ticks), number of original clauses, and number of learned clauses corresponding is showed in the graph below.

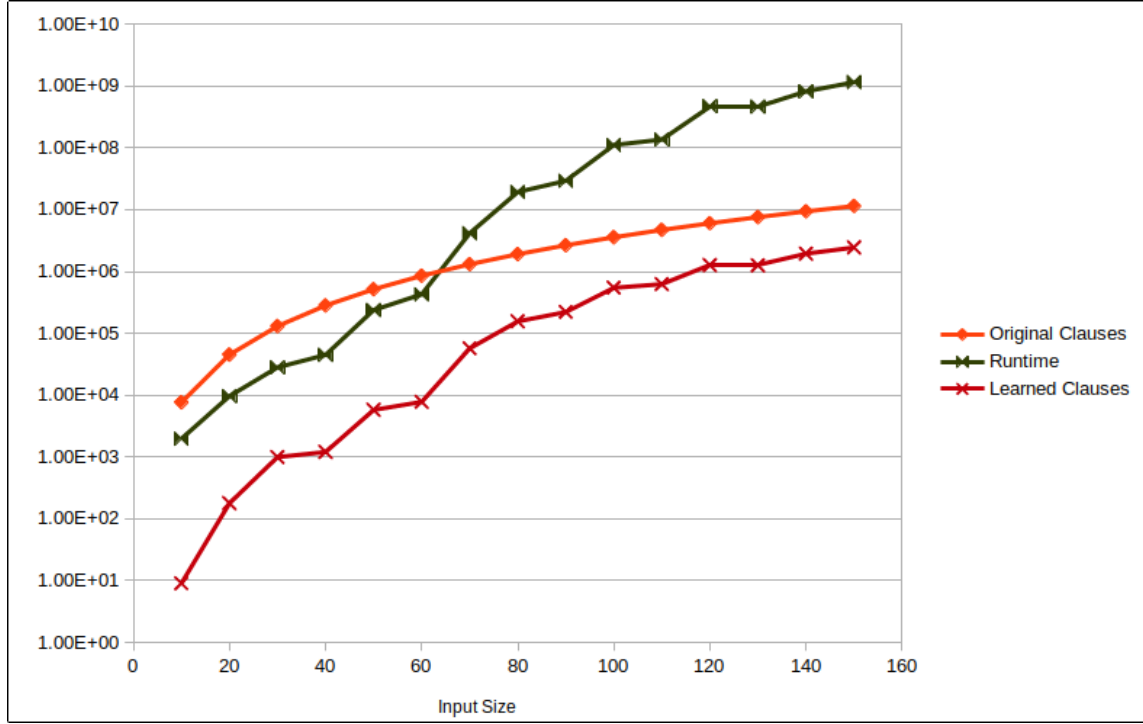


Figure 2: Runtime(ticks), number of original clauses, and number of learned clauses versus Input size

## 5 Reference

GitHub Link: <https://github.com/allen1236/sat-ox-problem>