2×n Minesweeper Consistency Problem is in P

Shu-Chiung Hu
Department of Computer Science,
National Chiao Tung University
linear.hu@gmail.com

Shun-Shii Lin
Graduate Institute of Computer Science and Information Engineering,
National Taiwan Normal University
linss@csie.ntnu.edu.tw

Abstract

Minesweeper is a popular single-player game included with Windows operating systems. Since Richard Kaye [12] proved that "Minesweeper is NP-complete" in 2000, it has been recently studied by many researchers. Meredith Kadlac [7] had showed that one-dimensional Minesweeper consistency problem is regular and can be recognized by a deterministic finite automaton. We extend the consistency problem to 2×n Minesweeper, which is two-dimensional but with its one dimension restricted to 2. We find that this problem is also tractable and design a finite automaton which can solve 2×n Minesweeper consistency problem in linear time. Hence, we are able to show that 2×n Minesweeper consistency problem is also in P.

Keywords: Minesweeper, Minesweeper consistency problem, finite automata, P

1 Introduction

Minesweeper [10] is a single-player computer game which was invented by Robert Donner and Curt Johnson in 1989. The game has been rewritten for many computer platforms and is most famous for the version that comes with Microsoft Windows.

The game consists of a rectangular field of squares much like a chess or checker board, and all squares are covered initially. Some mines are randomly and secretly distributed throughout the board.

A player can uncover or mark any square by left- or right-clicking on it. If a covered square with a mine is left-clicked upon by a player, the mine would expose and the game is over. At the time, what a player should do is to try his/her best to guess where the mines are. If a player is sure that a mine is hidden under a square, he/she can mark (right-clicked once) that square. However, if he/she is not sure that a mine is hidden under a square or not, he can mark a question mark('?') by right-clicking twice on that square instead. A player just uses the question mark to remind himself/herself that those squares are probably mines, but actually those squares are still covered squares.

So we treat the '?'-marked squares and the covered squares as the same. If a covered square without a mine is left-clicked upon by a player, two possible results could happen. A number between 0 and 8, indicating the amount of adjacent (including diagonally-adjacent) squares containing mines, would appear on this square. If the number 0 appears on the square, then all the squares reachable from this square will be uncovered and their amounts of adjacent squares containing mines will be appeared on these uncovered squares. The game is won when all squares without mines are uncovered. The goal of Minesweeper is to locate all mines (or "bombs") without touching any square with a mine as quickly as possible.

The complexity class P is the set of languages accepted by deterministic Turing machines in polynomial time. And the class NP is the set of languages accepted by nondeterministic Turing machines in polynomial time. One famous open problem is "P=NP?" question: to determine whether there exists an efficient algorithm which can solve an NP-complete problem or alternatively to prove no efficient algorithm exists for these NP-complete problems. This is one of the biggest and most important open problems at this moment, and is the subject of a $1,000,000 prize offered by the Clay Math institute in the USA. Richard Kaye's [12] result states that a decision problem called "Minesweeper Consistency Problem" (abbreviated as MCP) is equivalent to the problem of playing the Minesweeper game which is another NP-complete problems. That is, the problem of simply determining which squares are mines or not is equivalent to MCP.

Meredith Kadlac [7] had showed that one-dimensional MCP is easy. One-dimensional MCP is the original problem with one dimension restricted to one. One-dimensional MCP Problem is regular and can be recognized by a deterministic finite automaton.

In this paper, we will extend his work to 2×n MCP which is more complicated and difficult to be dealt with.

This paper is organized as follows. In Section 2, we describe some properties and definitions of MCP. Section 3 introduces a nondeterministic finite autom automaton (NFA) to solve 2×n MCP. In Section 4, we simplify the original NFA and discuss the corresponding DFA. In Section 5, we analyze the time to find consistent configurations. Section 6 exhibits our conclusions.

2 Properties and definitions of Minesweeper consistency problem

What is Minesweeper Consistency Problem? Richard Kaye defined this problem. On the FAQ in his Minesweeper site [6] he said:

"This is a question one can ask about any particular rectangular grid with the squares decorated by numbers 0-8, mines, or left blank. It asks: is there a configuration of mines in the grid that would result in the pattern of symbols one sees (according to the usual Minesweeper rules)? "

For the example of Figure 1(a), there is only one legal configuration of mines as shown in Figure 1(b). So we know this Minesweeper board is consistent, where "B" means a mine, "?" means an unknown square which could

0	?	?	?
?	1	4	?
1	2	?	B
B	2	2	2

(a)

0	0	2	B
0	1	4	B
1	2	B	B
B	2	2	2

(b)

2	3	3
?	B	?
2	2	2

(c)

Figure 1. (a) a given 4×4 Minesweeper board (b) one legal configuration of mines for (a) (c) an inconsistent Minesweeper board

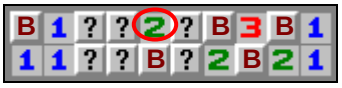
be a mine or a safe square, and a number between 0 and 8 means how many mines are in its surrounding squares. In Figure 1(c), there is an inconsistent square on the upper-right corner, for lacking one mine adjacent to it.

In Richard Kaye's article [12], "Minesweeper is NP-Complete", he proved that MCP is NP-complete by reducing the circuit satisfiability problem to Minesweeper. Since the general two-dimensional MCP is NP-complete, and Meredith Kadlac [7] had proved one-dimensional MCP is tractable, we make an effort to extend the one-dimensional MCP to two dimensions but with one dimension restricted to two in this paper. Here we call this kind of problem as 2×n MCP. 2×n Minesweeper game is a simplification version of the general Minesweeper game. However, it is more complex and difficult to prove the tractability than the one-dimensional one's. There are lots of possible input patterns to be dealt with. Fortunately, we find a way to simplify the finite automaton to avoid the explosive growth of the possible configurations. As a result, we are able to show that 2×n MCP is also tractable.

3 The 2×n Minesweeper Consistency problem

On a 2×n Minesweeper board, there are squares decorated by numbers 0 to 5, mine-marked, or '?'-marked squares (equivalent to covered squares). A configuration of a 2×10 Minesweeper board is shown in Figure 2(a).

(a)



(b)

Figure 2. (a) and (b) are the same for the circled square "2" in the fifth column contributes the same unit to its surrounding squares.

Of course, we can treat the 2×n and the n×2 Minesweeper boards as the same, for just rotating the n×2 Minesweeper board.

Definition 1: Given a 2×n Minesweeper grid with numbers and mines, some squares being covered, the 2×n Minesweeper consistency problem is to determine if there is a configuration of mines in those covered squares that give rise to the number seen.

That means a 2×n Minesweeper puzzle is consistent if there exists at least one correspondence between the information in each square and mines or covered squares. Note that a Minesweeper puzzle is solved correctly if each square numbered with m is surrounded by exactly m mines.

In this section, we will show that the 2×n MCP is tractable by exhibiting a nondeterministic finite automaton (abbreviated as NFA) which determines the consistency of any 2×n Minesweeper puzzle.

We describe how the NFA be created to solve this problem first. A 2×n Minesweeper board can be

represented by a sequence of n symbols. Each symbol represents a column of the board and is a pair over the alphabets β={0, 1, 2, 3, 4, 5, B, ?}. For example, the board in Figure 2(a) can be represented by <"B1", "11", "??", "??", "B2", "??", "B2", "3B", "B2", "11">. A symbol may be "00", "11", "B?", "??", ⋯, etc. There are 64 possible pairs as shown in Table 1. But with some interesting properties of 2×n Minesweeper, most pairs could be eliminated, and only 19 pairs left as shown in Table 2. One property is if both alphabets of the pair are numbers, they should be the same (such as '00', '11', '22', '33', '44') for each mine will contribute a same unit of count to both of its upper and lower squares in adjacent columns. That is, these two alphabets have the same impact on a 2×n Minesweeper board. Furthermore, some pairs like '0B', 'B0', '55' are always inconsistent, so we can also eliminate these kinds of pairs.

Table 1. All possible symbols for 2×n Minesweeper problem

00	01	02	03	04	05	0B	0?
10	11	12	13	14	15	1B	1?
20	21	22	23	24	25	2B	2?
30	31	32	33	34	35	3B	3?
40	41	42	43	44	45	4B	4?
50	51	52	53	54	55	5B	5?
B0	B1	B2	B3	B4	B5	B5	B?
?0	?1	?2	?3	?4	?5	?B	??

Table 2. All legal symbols for 2×n Minesweeper problem

00	0?	11	1B	1?	22	2B	2?
33	3B	3?	44	4B	4?	5B	5?
BB	?B	??					

In addition, alphabets of a pair can be exchanged with each other and will not affect the consistency result, so we just take account of one of the pairs. For example, if we exchange the two alphabets in the fifth column of Figure 2(a), we can get the other board as shown in Figure 2(b). A mine has already appeared above the circled square in Figure 2(a) and under the circled square in Figure 2(b), so the circled squares of both boards contribute the same unit of count to their surrounding squares—only one mine in these unknown surrounding squares of both boards. Hence, we can treat Figure 2(a) and Figure 2(b) as the same board.

Definition 2: A 2×n Minesweeper sequence of length n is a sequence of n symbols over the alphabet ∑={00, 11, 22, 33, 44, 1B, 2B, 3B, 4B, 5B, BB, ?B, 0?, 1?, 2?, 3?, 4?, 5?, ??}.

Definition 3: A Minesweeper sequence is globally consistent if no local inconsistency is found in the Minesweeper sequence.

The NFA takes a Minesweeper sequence of length n as input. The NFA is a 5-tuple (Q, Σ, δ, s_0, F), where Q is a finite set of 43 states, i.e., Q={s_0, $?_0?_0$, $?_B?_B$, $?_BB_B$, B_BB_B, $?_0?_B$, $?_0B_B$, 0_00_0, $0_0?_0$, 1_11_1, $1_1?_0$, 2_22_2, $2_2?_0$, 1_01_0, $1_0?_0$, 2_12_1, $2_1?_0$, 3_23_2, $3_2?_0$, $2_1?_B$, 2_1B_B, $3_2?_B$, 3_2B_B, $4_3?_B$, 4_3B_B, $1_1?_B$, 1_1B_B, $2_2?_B$, 2_2B_B, $3_3?_B$, 3_3B_B, 2_02_0, $2_0?_0$, 3_13_1, $3_1?_0$, $3_1?_B$, 3_1B_B, 4_24_2, $4_2?_0$, $4_2?_B$, 4_2B_B, $5_3?_B$, 5_3B_B}. The set of input alphabet is Σ={00, 11, 22, 33, 44, 1B, 2B, 3B, 4B, 5B, BB, ?B, 0?, 1?, 2?, 3?, 4?, 5?, ??}. δ: Q × Σ → Q is the state transition relation. δ defines the rules for state moving. s_0∈Q is the start state. F∈Q is the set of accepting states, F={s_0, $?_0?_0$, $?_B?_B$, '$?_BB_B$', 'B_BB_B', '$?_0?_B$', '$?_0B_B$', '0_00_0', '$0_0?_0$', '1_11_1', '$1_1?_0$', '2_22_2', '$2_2?_0$', '$1_1?_B$', '1_1B_B', '$2_2?_B$', '2_2B_B', '$3_3?_B$', '3_3B_B'}.

The states of the NFA have the form (X_xY_y), where "XY" means the input symbol, and the subscripts "x" and "y" indicate the information of mines for input alphabets X and Y. Table 3 explains the meaning of X_x (or Y_y). Take a state '$1_0?_0$' for example, "1?" is the input symbol which causes the machine to go to this state. Looking into Table 3, we can know that for the input '1', the subscript '0' means that no mine appeared in this and the left columns, and the next column should have only one mine. Then for the input '?', the subscript '0' means that this '?' is not a mine. Since no mine appeared in this column, so the next column must have one mine in order to keep consistency. The subscripts for those numbered squares reveal mine information—numbers of mines, and the subscripts "B" and "0" for '?'-marked squares reveal whether the '?' is a mine or not.

Table 3. Meaning of X_x (or Y_y), where the subscript x for X

X_x (or Y_y)	Meaning
0_0	There is no mine adjacent to this column.
1_0	There is no mine in this and the left columns, and the next column should have only one mine.
1_1	There is totally a mine in this and the left columns, and the next column should not have any mine.
2_0	There is no mine in this and the left columns, and the next column should have 2 mines.
2_1	There is totally a mine in this and the left columns, and the next column should have only one mine.
2_2	There are totally 2 mines in this and the left columns, and the next column should not have any mine.
3_1	There is totally a mine in this and the left columns, and the next column should have 2 mines.
3_2	There are totally 2 mines in this and the left columns, and the next column should have only one mine.
3_3	There are totally 3 mines in this and the left columns, and the next column should not have any mine.
4_2	There are totally 2 mines in this and the left columns, and the next column should have 2 mines.
4_3	There are totally 3 mines in this and the left columns, and the next column should have only one mine.
5_3	There are totally 3 mines in this and the left columns, and the next column should have 2 mines.
$?_0$	For the input alphabet '?', the subscript '0' means that this '?' is not a mine.

$?_B$	For the input alphabet '?', the subscript 'B' means that this '?' is a mine.
B_B	Input alphabet 'B' with the subscript 'B' means it is a mine.

We consider all cases which are possibly happened in

any 2×n Minesweeper board. Some state combinations are inconsistent such as '1_0B_B', '2_0B_B', '1_01_B', '3_24_2', '4_25_2', …,etc. For the example of '1_0B_B', "1_0" means that no mine has appeared in this and the left column, but "B_B" means the square is a mine in this column, a contradiction. The cases '3_24_2' and '4_25_2' are inconsistent for the illegal input symbols, and '1_0B_B', '2_0B_B', '1_01_B' are inconsistent for their impossible occurrences. As we described before, if both alphabets of the input symbol are numbers, they should be the same. For the states, this property still holds. So we can not get states like '1_01_B', '3_24_2', '4_25_2', '1_B2_B', '4_11_0'…, etc. In this way, we have totally 43 possible states in Q.

Now we construct the 2×n MCP state transition relations as shown in Table 4, where state 's_0', '$?_0?_0$', '$?_B?_B$', '$?_BB_B$', 'B_BB_B', '$?_0?_B$', '$?_0B_B$', '0_00_0', '$0_0?_0$', '1_11_1', '$1_1?_0$', '2_22_2', '$2_2?_0$', '$1_1?_B$', '1_1B_B', '$2_2?_B$', '2_2B_B', '$3_3?_B$', and '3_3B_B' are accepting states. We use double circle to represent them in Table 4. If the NFA ends at any one (say, '$k_k?_0$') of these accepting states, then there are totally k mines in the last two columns. We do not need extra mines to equalize the quantity k.

2	?	2	?
2	B	2	?

Figure 3. A 2×4 Minesweeper board

Let us see how this NFA works. A 2×n Minesweeper board is given in Figure 3, and the 2×n Minesweeper sequence is represented as <"22", "?B", "22", "??">. Initially, the machine is in the start state s_0 (in the state set q_0) and the first input symbol is "22", it goes to only one state '2_02_0' (in the state set q_8). From the state '2_02_0', there is only one state '$?_BB_B$' (in the state set q_2) to go on the next input symbol "?B". The third input symbol is "22", and the machine goes to the state '2_22_2' (in the state set q_4). Then the machine will go to the state '$?_0?_0$' (in the state set q_1) for the fourth input symbol is "??". The state '$?_0?_0$' is an accepting state, so we know that this 2×n Minesweeper board is consistent.

Now let us see an easy 2×n Minesweeper board shown in Figure 4. The 2×n Minesweeper sequence is represented as <"22">. Initially the machine is in the start state s_0 (in the state set q_0) and the first input symbol is "22", and it will go to the state '2_02_0' (in the state set q_8) which is a rejecting state. So we can know this board is not consistent.

2
2

Figure 4. A 2×1 Minesweeper board

Take another example, a 2×n Minesweeper board is shown in Figure 5. The 2×n Minesweeper sequence is represented as <"?B", "2?">. The machine is initially in the start state s_0 (in the state set q_0) and the first input symbol is "?B", and then the machine will have two states '$?_0B_B$' and '$?_BB_B$' to go. And the next input symbol is "2?", so the machine will have 2 states '$2_1?_B$' and '$2_2?_B$' to go if it is from state '$?_0B_B$'. The state '$2_2?_B$' is an accepting state. But

the state '$2_1?_B$' is a rejecting state because it needs an extra mine in the third column which is not present. On the other hand, if it is from the state '$?_BB_B$', then the machine will go to an accepting state '$2_2?_0$'. When the machine gets an input symbol consisting of one or two '?'s, it will have two or more paths to go. If the machine takes more and more inputs like these, it may have lots of possible paths to follow. Hence if we get a 2×n Minesweeper board with many inputs like "??", "?B", "BB", "2?", and etc., would the machine go to lots of states with explosive growth? In the follows, we will deal with this problem.

?	2
B	?

Figure 5. A 2×2 Minesweeper board

Table 4. The state transition relations for 2×n MCP NFA

state \ input	**0?**	**1?**	**2?**	**3?**	**4?**	**5?**	**?B**	**??**	**00**	**11**	**1B**	**22**	**2B**	**33**	**3B**	**44**	**4B**	**5B**	**BB**
q_0	s_0	$0_0?_0$	$1_0?_0$ $1_1?_B$	$2_0?_0$ $2_1?_B$	$3_1?_B$			$?_0B_B$ $?_BB_B$	$?_0?_0$ $?_0?_B$ $?_B?_B$	0_00_0	1_01_0	1_1B_B	2_02_0	2_1B_B		3_1B_B				B_BB_B
q_1	$?_0?_0$	$0_0?_0$	$1_0?_0$ $1_1?_B$	$2_0?_0$ $2_1?_B$	$3_1?_B$			$?_0B_B$ $?_BB_B$	$?_0?_0$ $?_0?_B$ $?_B?_B$	0_00_0	1_01_0	1_1B_B	2_02_0	2_1B_B		3_1B_B				B_BB_B
q_2	$?_B?_B$			$2_2?_0$	$3_2?_0$ $3_3?_B$	$4_2?_0$ $4_3?_B$	$5_3?_B$	$?_0B_B$ $?_BB_B$	$?_0?_0$ $?_0?_B$ $?_B?_B$				2_22_2		3_23_2	3_3B_B	4_24_2	4_3B_B	5_3B_B	B_BB_B
	$?_BB_B$			$2_2?_0$	$3_2?_0$ $3_3?_B$	$4_2?_0$ $4_3?_B$	$5_3?_B$	$?_0B_B$ $?_BB_B$	$?_0?_0$ $?_0?_B$ $?_B?_B$				2_22_2		3_23_2	3_3B_B	4_24_2	4_3B_B	5_3B_B	B_BB_B
	B_BB_B			$2_2?_0$	$3_2?_0$ $3_3?_B$	$4_2?_0$ $4_3?_B$	$5_3?_B$	$?_0B_B$ $?_BB_B$	$?_0?_0$ $?_0?_B$ $?_B?_B$				2_22_2		3_23_2	3_3B_B	4_24_2	4_3B_B	5_3B_B	B_BB_B
q_3	$?_0?_B$		$1_1?_0$	$2_1?_0$ $2_2?_B$	$3_1?_0$ $3_2?_B$	$4_2?_B$		$?_0B_B$ $?_BB_B$	$?_0?_0$ $?_0?_B$ $?_B?_B$		1_11_1		2_12_1	2_2B_B	3_13_1	3_2B_B		4_2B_B		B_BB_B
	$?_0B_B$		$1_1?_0$	$2_1?_0$ $2_2?_B$	$3_1?_0$ $3_2?_B$	$4_2?_B$		$?_0B_B$ $?_BB_B$	$?_0?_0$ $?_0?_B$ $?_B?_B$		1_11_1		2_12_1	2_2B_B	3_13_1	3_2B_B		4_2B_B		B_BB_B
q_4	0_00_0	$0_0?_0$	$1_0?_0$	$2_0?_0$					$?_0?_0$	0_00_0	1_01_0		2_02_0							
	$0_0?_0$	$0_0?_0$	$1_0?_0$	$2_0?_0$					$?_0?_0$	0_00_0	1_01_0		2_02_0							
	1_11_1	$0_0?_0$	$1_0?_0$	$2_0?_0$					$?_0?_0$	0_00_0	1_01_0		2_02_0							
	$1_1?_0$	$0_0?_0$	$1_0?_0$	$2_0?_0$					$?_0?_0$	0_00_0	1_01_0		2_02_0							
	2_22_2	$0_0?_0$	$1_0?_0$	$2_0?_0$					$?_0?_0$	0_00_0	1_01_0		2_02_0							

	$2_2?_0$	$0_0?_0$	$1_0?_0$	$2_0?_0$					$?_0?_0$	0_00_0	1_01_0		2_02_0							
q_5	1_01_0		$1_1?_B$	$2_1?_B$	$3_1?_B$			$?_0B_B$	$?_0?_B$			1_1B_B		2_1B_B		3_1B_B				
	$1_0?_0$		$1_1?_B$	$2_1?_B$	$3_1?_B$			$?_0B_B$	$?_0?_B$			1_1B_B		2_1B_B		3_1B_B				
	2_12_1		$1_1?_B$	$2_1?_B$	$3_1?_B$			$?_0B_B$	$?_0?_B$			1_1B_B		2_1B_B		3_1B_B				
	$2_1?_0$		$1_1?_B$	$2_1?_B$	$3_1?_B$			$?_0B_B$	$?_0?_B$			1_1B_B		2_1B_B		3_1B_B				
	3_23_2		$1_1?_B$	$2_1?_B$	$3_1?_B$			$?_0B_B$	$?_0?_B$			1_1B_B		2_1B_B		3_1B_B				
	$3_2?_0$		$1_1?_B$	$2_1?_B$	$3_1?_B$			$?_0B_B$	$?_0?_B$			1_1B_B		2_1B_B		3_1B_B				
q_6	$2_1?_B$			$2_2?_B$	$3_2?_B$	$4_2?_B$		$?_0B_B$	$?_0?_B$					2_2B_B		3_2B_B		4_2B_B		
	2_1B_B			$2_2?_B$	$3_2?_B$	$4_2?_B$		$?_0B_B$	$?_0?_B$					2_2B_B		3_2B_B		4_2B_B		
input / state		**0?**	**1?**	**2?**	**3?**	**4?**	**5?**	**?B**	**??**	**00**	**11**	**1B**	**22**	**2B**	**33**	**3B**	**44**	**4B**	**5B**	**BB**
q_6	$3_2?_B$			$2_2?_B$	$3_2?_B$	$4_2?_B$		$?_0B_B$	$?_0?_B$					2_2B_B		3_2B_B		4_2B_B		
	3_2B_B			$2_2?_B$	$3_2?_B$	$4_2?_B$		$?_0B_B$	$?_0?_B$					2_2B_B		3_2B_B		4_2B_B		
	$4_3?_B$			$2_2?_B$	$3_2?_B$	$4_2?_B$		$?_0B_B$	$?_0?_B$					2_2B_B		3_2B_B		4_2B_B		
	4_3B_B			$2_2?_B$	$3_2?_B$	$4_2?_B$		$?_0B_B$	$?_0?_B$					2_2B_B		3_2B_B		4_2B_B		
q_7	$1_1?_B$		$1_1?_0$	$2_1?_0$	$3_1?_0$				$?_0?_0$		1_11_1		2_12_1		3_13_1					
	1_1B_B		$1_1?_0$	$2_1?_0$	$3_1?_0$				$?_0?_0$		1_11_1		2_12_1		3_13_1					
	$2_2?_B$		$1_1?_0$	$2_1?_0$	$3_1?_0$				$?_0?_0$		1_11_1		2_12_1		3_13_1					
	$2_2?_B$		$1_1?_0$	$2_1?_0$	$3_1?_0$				$?_0?_0$		1_11_1		2_12_1		3_13_1					
	$3_3?_B$		$1_1?_0$	$2_1?_0$	$3_1?_0$				$?_0?_0$		1_11_1		2_12_1		3_13_1					
	$3_3?_B$		$1_1?_0$	$2_1?_0$	$3_1?_0$				$?_0?_0$		1_11_1		2_12_1		3_13_1					
q_8	2_02_0							$?_BB_B$	$?_B?_B$											B_BB_B
	$2_0?_0$							$?_BB_B$	$?_B?_B$											B_BB_B
	3_13_1							$?_BB_B$	$?_B?_B$											B_BB_B
	$3_1?_0$							$?_BB_B$	$?_B?_B$											B_BB_B
	$3_1?_B$							$?_BB_B$	$?_B?_B$											B_BB_B
	3_1B_B							$?_BB_B$	$?_B?_B$											B_BB_B
	4_24_2							$?_BB_B$	$?_B?_B$											B_BB_B
	$4_2?_0$							$?_BB_B$	$?_B?_B$											B_BB_B
	$4_2?_B$							$?_BB_B$	$?_B?_B$											B_BB_B
	4_2B_B							$?_BB_B$	$?_B?_B$											B_BB_B

	$5_3?_B$							$?_BB_B$	$?_B?_B$										B_BB_B
	5_3B_B							$?_BB_B$	$?_B?_B$										B_BB_B

As described before, we must care about the growth of possible moving paths in the NFA. Here we give another example. A 2×n Minesweeper board is given in Figure 6.

2	?	4	?	?
?	?	?	B	2

Figure 6. A 2×4 Minesweeper board

The 2×n Minesweeper sequence is represented as <"2?", "??", "4?", "?B", "2?">. In the NFA, the machine will have 4 possible moving paths according to the state transition relations of Table 4.

Initially the machine on the input symbol "2?" has two possible states '$2_0?_0$' and '$2_1?_B$' to go. The machine in the state '$2_0?_0$' will go to the state '$?_B?_B$' while reading the input symbol "??". The machine in the state '$2_1?_B$' will go to the state '$?_0?_B$' while reading the input symbol "??". If the machine goes to the state '$?_B?_B$' and reads the next input symbol '4?', then it splits again and gets two possible states '$4_2?_0$', '$4_3?_B$'. On the other hand, if the machine goes to the state '$?_0?_B$' and reads the input symbol "4?", then it can only go to the state '$4_2?_B$'. The next input symbol is "?B", the machine will go to the state '$?_BB_B$' if it is from states '$4_2?_0$' or '$4_2?_B$', or go to the state '$?_0?_B$' if it is from the state '$4_3?_B$'. The machine in the state '$?_B?_B$' reads the final input symbol "2?" will go to the state '$2_2?_0$'. On the other hand, the machine in the state '$?_0?_B$' will have 2 states '$2_1?_0$' or '$2_2?_B$' to go. But the state '$2_1?_0$' is a rejecting state because it needs an extra mine in the next column which is not present. So only 4 possible paths are consistent. See below for a depiction.

- $s_0 \rightarrow 2_0?_0 \rightarrow ?_B?_B \rightarrow 4_2?_0 \rightarrow ?_B?_B \rightarrow 2_2?_0$ (accepting state)
- $s_0 \rightarrow 2_0?_0 \rightarrow ?_B?_B \rightarrow 4_3?_B \rightarrow ?_0?_B \rightarrow 2_1?_0$ (rejecting state)
- $s_0 \rightarrow 2_0?_0 \rightarrow ?_B?_B \rightarrow 4_3?_B \rightarrow ?_0?_B \rightarrow 2_2?_B$ (accepting state)
- $s_0 \rightarrow 2_1?_B \rightarrow ?_0?_B \rightarrow 4_2?_B \rightarrow ?_B?_B \rightarrow 2_2?_0$ (accepting state)

Since the rules of transitions only depend on the number information of mines between current and the previous columns as well as the next input symbol, the NFA can correctly reach an accepting state or a rejecting state.

4 Simplified NFA and DFA for 2×n MCP

According to the state transition relations shown in Table 4, we find that some states have the same behavior in the table, so we can combine these states to a new state set. Then we can get 8 equivalent state sets.

- $q_0 = \{s_0\}$, q_0 is the start state set.
- $q_1 = \{?_0?_0\}$
  q_1 is the state set which means no mine is present in this and the left columns.
- $q_2 = \{?_B?_B, ?_BB_B, B_BB_B\}$
  q_2 is the state set which means 2 mines are present in this and the left columns.
- $q_3 = \{?_0?_B, ?_0B_B\}$
  q_3 is the state set which means only one mine is present in this and the left columns.
- $q_4 = \{0_00_0, 0_0?_0, 1_11_1, 1_1?_0, 2_22_2, 2_2?_0\}$,
  q_4 is the state set which means 2 numbered squares (the 2 numbers are the same) are present in this column, and they do not need extra mines to equalize them.

- $q_5 = \{1_01_0, 1_0?_0, 2_12_1, 2_1?_0, 3_23_2, 3_2?_0\}$,
  q_5 is the state set which means 2 numbered squares (the 2 numbers are the same) are present in this column, and they need one extra mine to equalize them.
- $q_6 = \{2_1?_B, 2_1B_B, 3_2?_B, 3_2B_B, 4_3?_B, 4_3B_B\}$,
  q_6 is the state set which means a numbered square and a mine are present in this column, and the number square needs one extra mine to equalize it.
- $q_7 = \{1_1?_B, 1_1B_B, 2_2?_B, 2_2B_B, 3_3?_B, 3_3B_B\}$,
  q_7 is the state set which means a numbered square and a mine are present in this column, and the number square does not need extra mines to equalize it.
- $q_8 = \{2_02_0, 2_0?_0, 3_13_1, 3_1?_0, 3_1?_B, 3_1B_B, 4_24_2, 4_2?_0, 4_2?_B, 4_2B_B, 5_3?_B, 5_3B_B\}$,
  q_8 is the state set which means 2 numbered squares are present, and they need two extra mines to equalize them.

  After combining those states, we can get a simplified state transition table as shown in Table 5, where there are 6 accepting states: q_0, q_1, q_2, q_3, q_4, and q_7.

The state transition diagram is shown in Figure 7. Since it is an NFA— several choices may exist for the next state for some inputs. For example, when the machine goes to the state set 'q_3' with the next input '??', the machine will have three possible state sets q_1, q_2 or q_3 to go. According to computation theory [8], we have the following theorem.

Theorem 1: An NFA has an equivalent deterministic finite automaton.

Deterministic and nondeterministic finite automata recognize the same class of languages. Such equivalence is both surprising and useful. Now we are certainly able to find an equivalent DFA for the NFA we constructed for 2×n MCP.

Table 5. Simplified state transition Table for 2×n MCP NFA

input / state sets	1?	2?	3?	4?	?B	??	00	0?	11	1B	22	2B	33	3B	44	4B	5?	5B	BB
◎q_0	q_5 q_7	q_6 q_8	q_8		q_2 q_3	q_1 q_2 q_3	q_4	q_4	q_5	q_7	q_8	q_6		q_8					q_2
◎q_1	q_5 q_7	q_6 q_8	q_8		q_2 q_3	q_1 q_2 q_3	q_4	q_4	q_5	q_7	q_8	q_6		q_8					q_2
◎q_2		q_4	q_5 q_7	q_6 q_8	q_2 q_3	q_1 q_2 q_3					q_4		q_5	q_7	q_8	q_6	q_8	q_8	q_2
◎q_3	q_4	q_5 q_7	q_6 q_8	q_8	q_2 q_3	q_1 q_2 q_3			q_4		q_5	q_7	q_8	q_6		q_8			q_2
◎q_4	q_5	q_8				q_1	q_4	q_4	q_5		q_8								

q_5	q_7	q_6	q_8		q_3	q_3				q_7		q_6		q_8					
q_6		q_7	q_6	q_8	q_3	q_3						q_7		q_6		q_8			
(q_7)	q_4	q_5	q_8			q_1			q_4		q_5		q_8						
q_8					q_2	q_2													q_2

Theorem 2: 2×n MCP is in P.

Proof: The equivalent DFA can determine 2×n MCP in O(n) time, for the DFA takes linear time to scan a 2×n Minesweeper board. So we proved that 2×n MCP is in class P.

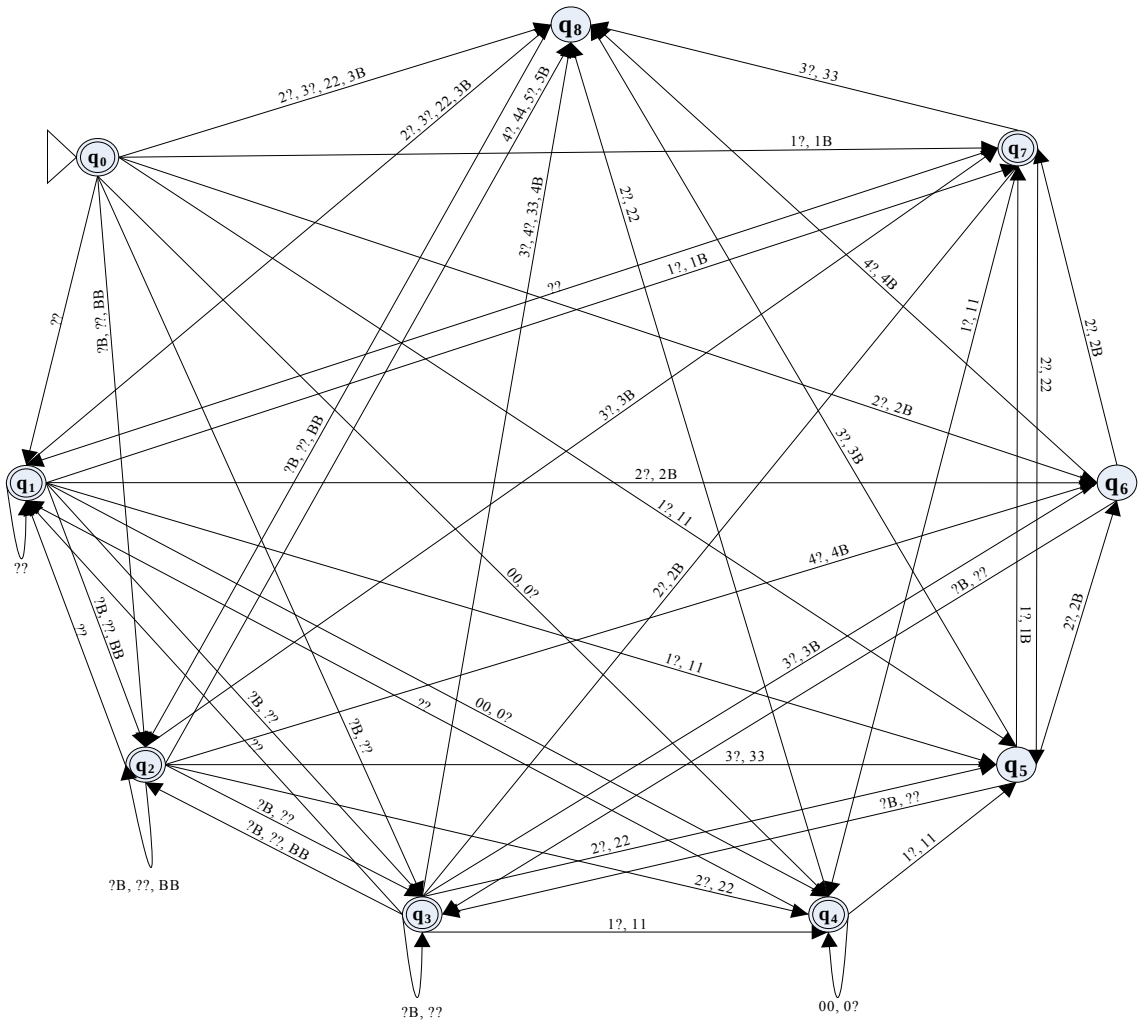


Figure 7. Simplified state transition diagram for 2×n MCP NFA

5 Finding consistent configurations

For the example of Figure 6, we can find 4 consistent configurations as shown in Figure 9 according to the possible state transition paths as shown in Figure 8. Note that there are only three accepting states, but there are two possible configurations '$?_0?_B$' and '$?_B?_0$' for the second column in the lowest path of Figure 9, hence we have totally 4 consistent configurations.

When the DFA determines that a Minesweeper sequence is consistent, it passes the state sets which include all possible consistent configurations for all input symbols. In order to find these consistent configurations, we use the depth-first search to traverse the search tree whose search space is the possible state sets. We can find all consistent configurations after finishing depth-first search, so it may take exponential time to find all consistent configurations which is equal to the time to do depth-first search. However, if we only want to find a consistent configuration, it only takes O(n) time to walk on any path of the search tree from the root to a leaf node.

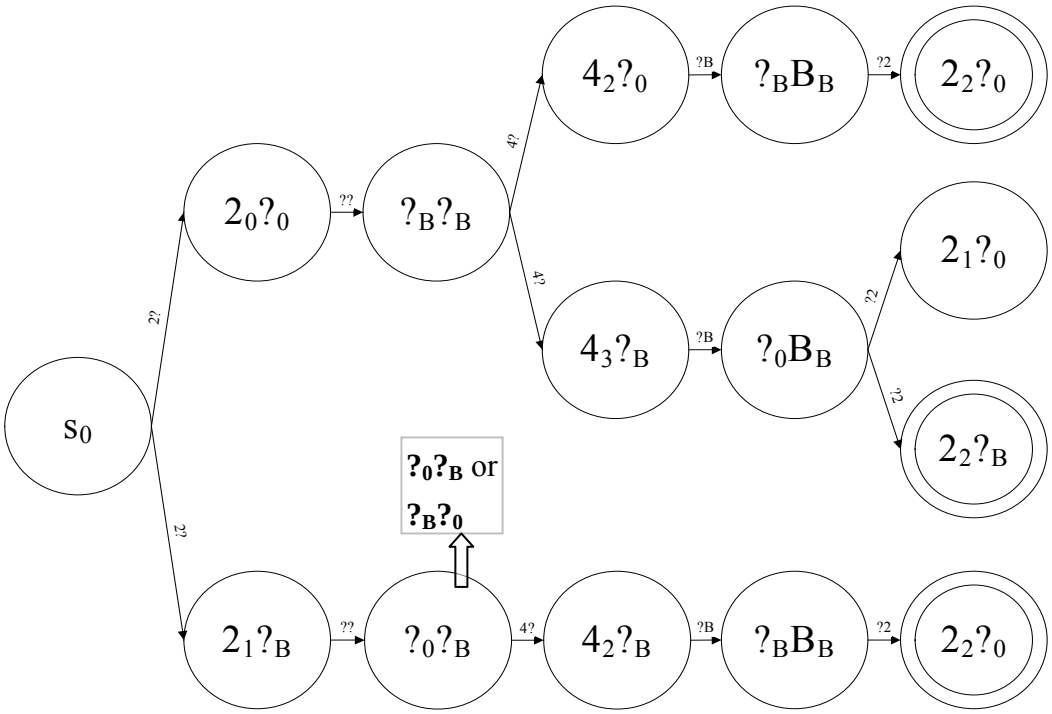


Figure 8. The possible state transition paths for Figure 6

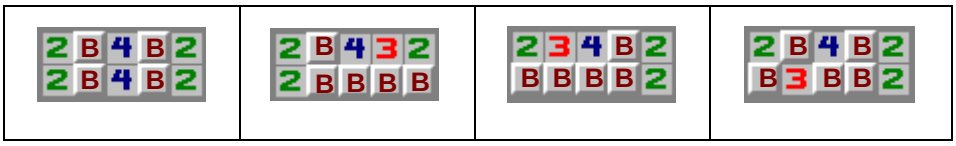


Figure 9. All consistent configurations for Figure 6

6 Concluding remarks

In this paper, we extend Meredith Kadlac's one-dimensional MCP [7] to 2×n MCP which is more complex and difficult to be dealt with. According to the properties of 2×n Minesweeper game, we analyze all possible input symbols, states, and state transitions and successfully construct an NFA which can determine the consistency of 2×n MCP. We further simplify the original 43 states to 8 state sets according to their behavior. Then we can convert this NFA to a corresponding DFA which also takes linear time to solve 2×n MCP. Hence we proved that 2×n MCP is tractable and in class P.

When we know a 2×n Minesweeper board is consistent, we may spend exponential time expanding the search tree to find all consistent configurations for that board or spend linear time walking on any path of the search tree from the root to a leaf node.

The topics of "Minesweeper consistency problem" are worth studying further in the future. Furthermore, we hope that we can extend the problem to more general problems and try to prove the complexity of these kinds of problem. Because Richard Kaye has proved general MCP is NP-complete, we may devote to find a number m which causes m×n MCP to be NP-complete.

Let us consider another problem. The complexity of 2-SAT belongs to *P*, which means we can find a NFA with finite states to solve it. However, 3-SAT is NP-complete, which means no NFA can be found to solve 3-SAT at the present time. That is to say, we even cannot derive all accepting patterns to form a correct NFA. In this paper, we are able to show that 1×n and 2×n Minesweeper consistency problems belong to *P*. When the board is extended to 3×n or even larger, does there exist an NFA which can solve this problem? This is a quite interesting open problem. We hope this paper will prompt researchers to study other related problems.

7 Acknowledgement

This research was supported in part by a grant NSC94-2213-E-003-004 from National Science Council, R.O.C.

8 References

[1] B. P. McPhail, "The Complexity of Puzzles: NP-Completeness Results for Nurikabe and Minesweeper," Senior Thesis, Reed College, 2003.

[2] C. Studholme, "Minesweeper as a Constraint Satisfaction Problem," 2005. http://www.cs.toronto.edu/~cvs/Minesweeper/

[3] F. Wester, "The Minesweeper Page," 2005. http://www.frankwester.net/winmine.html

[4] I. Stewart, "Ian Stewart on Minesweeper," http://www.claymath.org/Popular_Lectures/Minesweeper/

[5] J. D. Ramsdell, "Programmer's Minesweeper," http://www.ccs.neu.edu/home/ramsdell/pgms/

[6] J. Palumbo, "The P Vs NP Problem, NP Completeness, and Minesweeper," 2003. http://www.math.rutgers.edu/~greenfie/currentcourses/sem090/pdfstuff/palumbo.pdf

[7] M. Kadlac, "Explorations of the Minesweeper Consistency Problem," Proceedings of the Research Experiences for Undergraduates Program in Mathematics, Oregon State University, pp.78-126 , 2003.

[8] M. Sipser, Introduction to the Theory of Computation, Second Edition, Course Technology, 2005.

[9] Pedro Gimeno Fortea., "Minesweeper Designer v0.1 beta," http://www.formauri.es/personal/pgimeno/comp urec/Minesweeper.php

[10] R. Donner and C. Johnson., "Minesweeper," http://www.planat-minesweeper.com/download.php

[11] R. Kaye, "Richard Kaye's Minesweeper Website," http://web.mat.bham.ac.uk/R.W.Kaye/minesw/

[12] R. Kaye, "Minesweeper is NP-Complete," The Mathematical Intelligencer, **22**(22) pp. 9-15, 2000.

[13] R. Kaye, "Some Minesweeper Configurations," 2000. http://web.mat.bham.ac.uk/R.W.Kaye/minesw/

[14] Raphaël Collet, "Playing the Minesweeper with Constraints," Second International Mozart/Oz Conference, pp.251-262, 2004.

[15] T. A. Sudkamp, Languages and Machines: An Introduction to the Theory of Computer Science, 3rd Edition, Addison-Wesley, 2005.