Some Minesweeper Configurations

Richard Kaye
School of Mathematics and Statistics
The University of Birmingham
Birmingham
B15 2TT

R.W.Kaye@bham.ac.uk
http://for.mat.bham.ac.uk/R.W.Kaye

13th August 2000

Contents

Note This is the first version of this paper. It has now been superceded. See files minesw.∗ for the latest version.

1 Introduction

In a recent paper in *The Mathematical Intelligencer*, I prove that Minesweeper is NP-complete.

The purpose of this particular document is to collect together some interesting configurations I have found in the course of the research leading to the result mentioned above. I welcome people to email me (at R.W.Kaye@bham.ac.uk) with their comments, questions, and suggestions. My home page is

http://www.mat.bham.ac.uk/R.W.Kaye/

And my Minesweeper page is

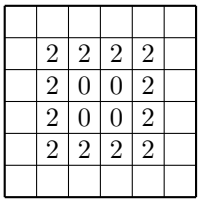

Figure 1: Determine the location of all mines.

http://www.mat.bham.ac.uk/R.W.Kaye/minesw/minesw.htm

I will be happy to include any other interesting configurations here, with full credit, that I receive. For background information, please see my *Intelligencer* paper. The *Minesweeper Problem* is to determine if a given configuration is *consistent* (i.e., that there can be real positions of mines that give the position you see). See my paper for the reason that this really is the right question to be asking about a Minesweeper configuration and why a good algorithm to solve it will enable you to play the game better.

In the configurations here, a star, $*$ will be used to denote a mine that has been identified. A blank denotes an unknown square, and a number denotes a square that has been cleared. Squares outside the thick black lines should be thought of as having been cleared, all with zeros. The rules are as for Minesweeper as provided with Microsoft's Windows.[1]

2 Minesweeper is difficult

Figure 1 gives a nice minesweeper puzzle to think about. If you find it easy you probably are a real minesweeper addict, and are getting to be quite good at the game. Personally, I can solve it, but don't consider it to very easy. There are many variations of this puzzle, such as changing the size of the playing area. Please email me with other entertaining puzzles like this that you have found or know about.

In general, it is not easy to tell if a guess is required at a given position in the game, or if the whole position can be calculated directly. In Figure fig:mineswpuz no guess is required, but many players may not spot this fact. My theorem on minesweeper, that it is NP-complete demonstrates these points very nively.

3 What is NP-completeness about?

My first observation when playing Minesweeper is that it seems that the whole of the playing area must be considered when one makes a move (i.e., clears a square or labels a mine). For example see the configuration in Figure 2. (Real

[1] 'Windows' is a trademark of Microsoft. The author has no connections with Microsoft, and nothing here should be regarded as comment on any of Microsoft's products.

2	3	∗	2	2	∗	2	1
∗	∗	5			4	∗	2
	?	∗	∗	∗		4	∗
∗	6		6	∗	∗		2
2	∗	∗		5	5		2
1	3	4		∗	∗	4	∗
0	1	∗	4			∗	3
0	1	2	∗	2	3	∗	2

Figure 2: Consider the whole playing area!

minesweeper addicts will not find this puzzle at all difficult, but it illustrates the point.)

The idea that the whole of the playing area must be considered is the reason why Minesweeper configurations can be difficult, and the main reason why I originally suspected that Minesweeper is NP-complete.

I don't know any good mathematical way of proving that 'to solve some kinds of minesweeper problems you must consider the whole of the configuration', but the theory of NP-completeness comes quite close. No one knows if any NP-complete problem can be solved efficiently, but most people believe none of them can.

NP-complete problems all take the form of general problems requiring a yes/no answer. A rather nice (non-minesweeper) example of such a problem is the 'BIN-PACKING' problem which asks, given n floppy disks each of size x and m computer files of size $y_1, y_2, y_3, \ldots, y_m$, is it possible to copy all of the files to the floppy disks given? Obviously some times the answer will be 'yes', and sometimes 'no'. It depends on the numbers n, m, x and $y_1, \ldots, y_m$. What is required is an *efficient* computer program or algorithm that will give the answer in all cases. (Here and thoughout, I mean 'efficient' in the technical sense of 'running in polynomial time'.)

Obviously problems like this are of great practical importance, and an efficient algorithm would be very useful. Unfortunately no-one has such an algorithm, and many people suspect that there isn't such an algorithm. In principle, it would be possible to *prove* that there is no such algorithm, but again no-one has managed to do this either. Just about all that we do know is that if any of the known NP-complete problems has an efficient algorithm, then they all do—and conversely if one of them can be proved not to have any efficient algorithm then they all have the same status.

The famous 'P,NP question' or 'P=NP?' question is just this: to determine whether any NP-complete problem has an efficient algorithm (in which case they all would have), or if any NP-complete problem can be proved *not* to have an efficient algorithm (in which case none of them would have). This is one of the biggest and most important open problem in mathematics at the moment, and is the subject of a $1,000,000 prize offered by the Clay institute in the USA.

There are hugely important reasons why one would want to know the answer to this question, whatever the answer would be. Even a 'negative' answer that P is *not* equal to NP would have important practical consequences. In particular, many codes used on the internet are designed on the basis that a potential code-

$\mathbf{X} \longrightarrow$

$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\cdots$
$\cdots$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	$\cdots$
$\cdots$	x	1	x'	x	1	x'	x	1	x'	x	1	x'	x	1	x'	x	$\cdots$
$\cdots$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	$\cdots$
$\cdots$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\cdots$

Figure 3: A wire.

breaker would have to find an efficient algorithm for an NP-complete problem to break the code in reasonable time.

I showed the 'consistency problem for minesweeper' is NP-complete. This says that to present knowledge there is no algorithm for Minesweeper much better[2] than simply checking all of the 2^n or so possible ways the remaining mines can be distributed around the n uncovered squares.

The diagram in Figure 2 has what I call a single 'wire' looping round on itself. A simpler kind of wire is featured in Figure 3. I came across this idea of a wire when playing the game (and other people who play minesweeper must have come across similar things). It was when I started to think about wires as wires carring a logical signal that I started to ask myself whether it would be possible to construct logic gates. This turned out to be the way to prove Minesweeper to be NP-complete.

4 Configurations proving NP-completeness

The way I proved minesweeper is NP-complete is by designing Minesweeper configurations for 'logic circuits', and making a Minesweeper configuration behave like a computer. (Experts will recognise that this is a reduction of the known NP-complete problem SAT to Minesweeper.) For the details, you'll have to go to my original paper, but I present a few configurations here as hints for those people that might like to try to figure out on their own what they do. (As a bonus, there are some configurations here that don't appear in the original paper.) As before, if anyone finds other interesting configurations, please send them to me, and if I like them I'll include them in future versions of this article.

Figure 3 gives a wire carrying either TRUE or FALSE, but to start with there is no real difference between these truth values: we must take a somewhat arbitrary decision to define which of the two possibilities is to be called 'true' and which 'false'.

The way I define the value in the wire is this. First, give the wire some orientation (here, going from left to right). Now look at the squares just behind the central 1's in the wire. (Behind, meaning with respect to the direction of the wire which we just fixed once and for all.) If this is a mine, the value is TRUE else it is FALSE.

Wires can be bent, terminated, or split as Figure 4 and Figure 5 show. (Figure 4 (b) corrects a minor error in the original paper: thanks to Harry Hutchinson for pointing this out to me.)

[2]The phrase 'much better' has a technical meaning which I won't go in to right here

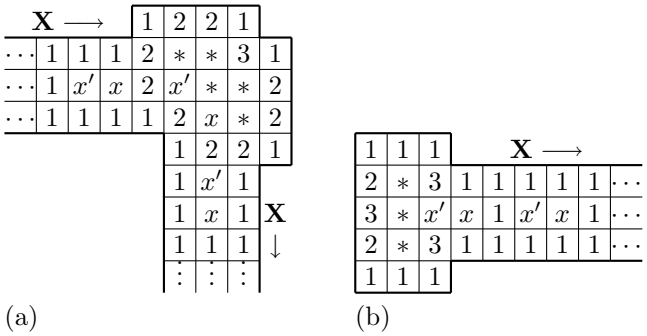

Figure 4: (a) A bent wire. (b) A terminated wire.

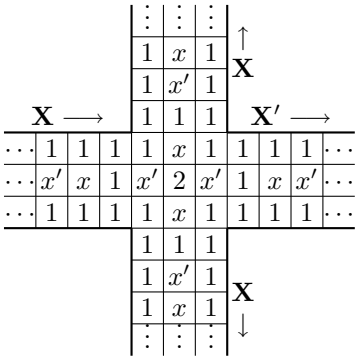

Figure 5: A three-way splitter.

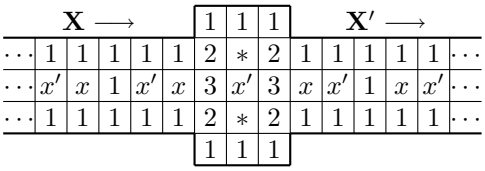

Figure 6: A NOT gate.

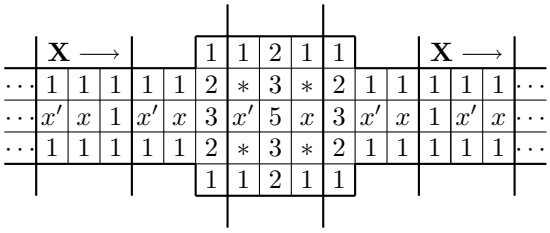

Figure 7: A phase-changer made from two NOT gates.

Figure 6 shows a NOT-gate. I'll leave you to work out how this works.

To change phase, you can use two NOT gates, as Figure 7 shows.

The next configuration, in Figure 8, doesn't appear in my paper, but describes an XOR gate. Again, I leave you the fun of working out the details.

The final and most difficult configuration in my *Intelligencer* article is the AND gate, and this is reproduced in Figure 9. A full explanation of how it works is my *Intelligencer* article, as is some informaion of how to put these gates together to make up more complicated boolean circuits.

The main difficulties in putting these gadgets together are concern firstly how to make other standard gates (such as OR, and so on) out of the ones already found, and secondly how to cross wires over one another. It turns out that both these problems were well-understood, and I showed how they can be solved in principle in my article. On the other hand, if one follows the method prescribed the resulting configurations can be rather large. After reading my article, Stefan Schwoon sent me (on 5th October 2000) two of his own configurations that manage to reduce this size quite a lot. They are: an OR gate (Figure 10); and a way of crossing wires (Figure 11) and are also reproduced here.

Stefan's OR gate is substatially smaller than my original AND gate and is based on a slightly different and simpler method. (I had previously spotted this sort of trick, and had used it in the write-up to my 'infinite' version of Minesweeper, but I noticed it too late to appear in the original article.)

5 Other questions concerning Minesweeper

The NP-completeness of Minesweeper answers many questions about the game, including some I have seen posted on the web. For example, it is *not* true that

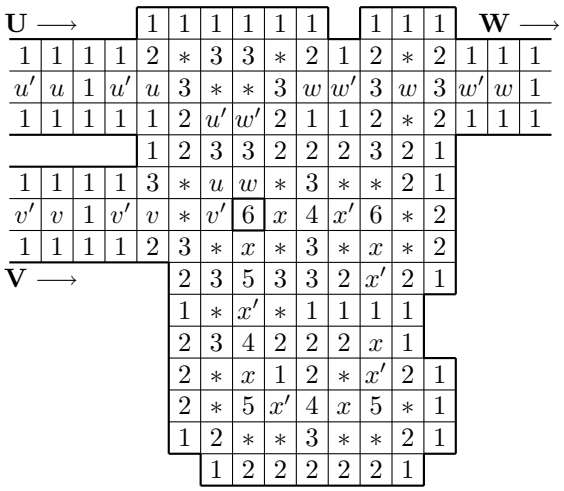

Figure 8: An XOR gate.

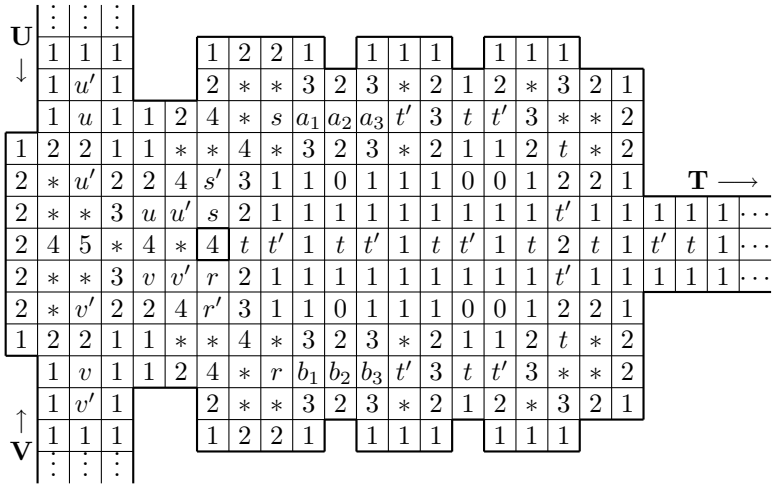

Figure 9: An AND gate.

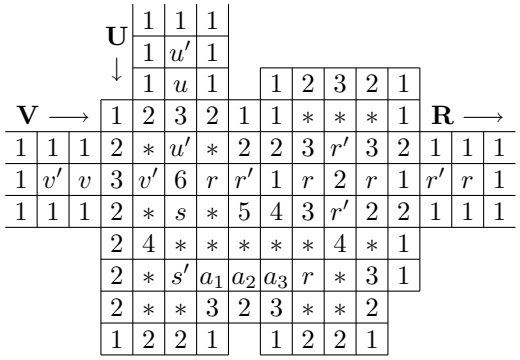

Figure 10: An OR gate

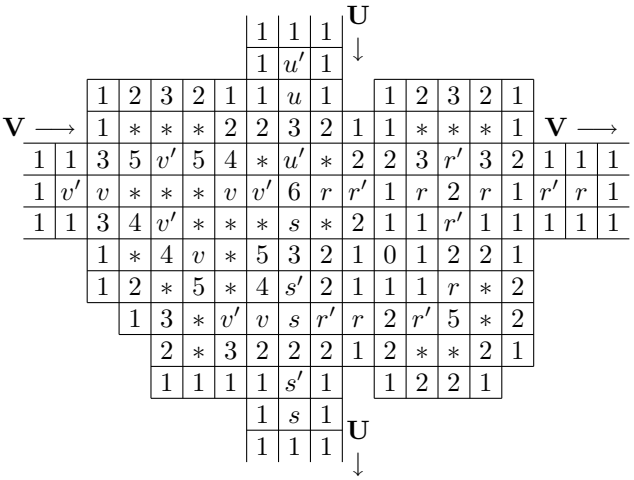

Figure 11: A wire crossing

if you are playing Minesweeper and are working on a symmetrical configuration then the final solution will be symmetrical.

Some people have mentioned to me that playing Minesweeper is *really* about probabilities, not certainties. I agree, but the NP-completeness result shows that the question 'what is the probability of a mine in square X?' is even harder than solving an NP-complete problem such as the travelling salesman problem!

6 Variations on the original Minesweeper game

There is no reason why Minesweeper need be played on a square grid, and you will find plenty of other Minesweeper games on other grids avaiable if you look on the web. I suspect that they are all NP-complete for much the same sorts of reasons.

I even played a three-dimensional version of Minesweeper once. (This was particularly fiendish!)

I have recently been working on an 'infinite' version of Minesweeper that is mathematically as complicated as an arbitrary computer, in the same way that ordinary minesweeper is as complicated as finite logic circuits. A (somewhat technical) paper on this is now available via my Minesweeper web page, http://www.mat.bham.ac.uk/R.W.Kaye/minesw/minesw.htm

7 Revision history and acknowledgements

Unless stated otherwise, all the configurations and ideas presented here are my own. (I will continue to credit any future contributions fully here in later versions of this article.)

This article was originally written and published on the web on 4th August 1999. The article was expanded on 14th August 2000 and revised again on November 3rd 2000.