

The computational complexity of Minesweeper

Michiel de Bondt
Radboud University Nijmegen, The Netherlands
M.deBondt@math.ru.nl

April 23, 2012

Abstract

We show that the Minesweeper game is PP-hard, when the object is to locate all mines with the highest probability. When the probability of locating all mines may be infinitesimal, the Minesweeper game is even PSPACE-complete. In our construction, the player can reveal a boolean circuit in polynomial time, after guessing an initial square with no surrounding mines, a guess that has 99 percent probability of success. Subsequently, the mines must be located with a maximum probability of success.

Furthermore, we show that determining the solvability of a partially uncovered Minesweeper board is NP-complete with hexagonal and triangular grids as well as a square grid, extending a similar result for square grids only by R. Kaye. Actually finding the mines with a maximum probability of success is again PP-hard or PSPACE-complete respectively.

Our constructions are in such a way that the number of mines can be computed in polynomial time and hence a possible mine counter does not provide additional information. The results are obtained by replacing the dyadic gates in [3] by two primitives which makes life more easy in this context.

Keywords: PP-hard, PSPACE-complete, NP-complete, boolean circuit, stochastic boolean variable.

1 Introduction

On almost every computer, one can play the game called Minesweeper. Minesweeper is played on a grid of square compartments, each of them surrounded by eight other such compartments, except on the border of the grid. The object is the game is to find all compartments which do not contain a mine. Any compartments that does not contain a mine contains a number which indicates how many of the surrounding compartments contains a mine. These numbers can be used to locate the mines. But such a number is only revealed after clicking on the compartment, something that is fatal when the compartment does contain a mine.

Richard Kaye showed in [3] that determining the solvability of a Minesweeper board is NP-complete, that is, whether the numbers revealed thus far correspond to a distribution of the mines. Although he ignores the fact that the total number of mines is also known in the Minesweeper game, it is a very nice article, where he builds a logical circuit on the Minesweeper board, in order to compute a boolean expression. Next, he enforces the outcome of the boolean expression to be true, whence the problem becomes finding boolean values for the variables such that the logical expression evaluates to true.

One can modify the game such that it can be played on other grids, which has already been done several times. We are going to show that we can determine the satisfiability of a boolean circuit by evaluating the solvability of a minesweeper board, for all three regular tessellations of the plane (triangular, normal and hexagonal Minesweeper). For this purpose, we build circuitry as well.

Since Minesweeper is a game with numbers, it is natural to associate the boolean values *false* and *true* with 0 and 1 respectively. In order to make boolean circuits on a minesweeper board, we first need a way to code wires on it. This is done in figure 1, where the wire on the square grid is taken from [3].

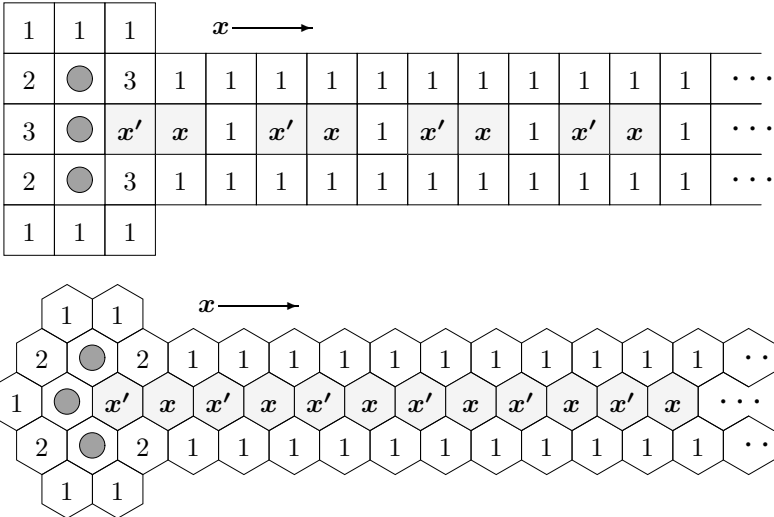


Figure 1: A wire that conducts x , running from the starting point of x

In figure 1, we do not know what the compartments with x and x' contain, but we do know that either exactly all compartments with x contain a mine (figure 2, corresponding to $x = 1$) or exactly all compartments with x' do (figure 3, corresponding to $x = 0$). Which of both cases are possible follows from the global structure of the Minesweeper board. To make polynomial equations from the polynomial expressions, it suffices to force these expressions to have a given value in $\{0, 1\}$. This can be done by forcing a wire that conducts a certain expression to conduct a given value. Figure 4 shows how to force a wire to conduct one and zero respectively.

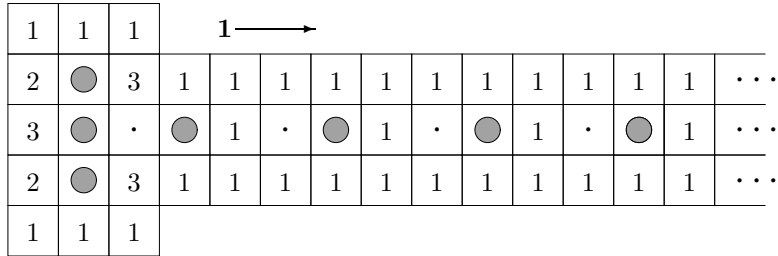


Figure 2: A wire that conducts one

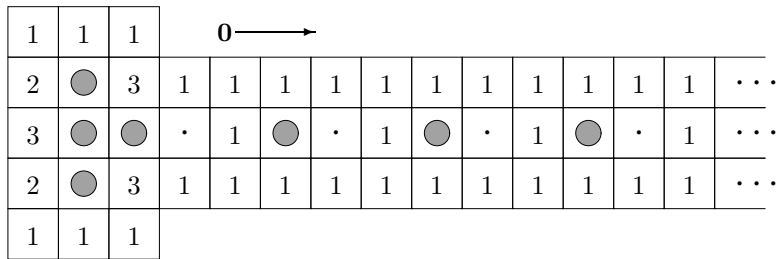


Figure 3: A wire that conducts zero

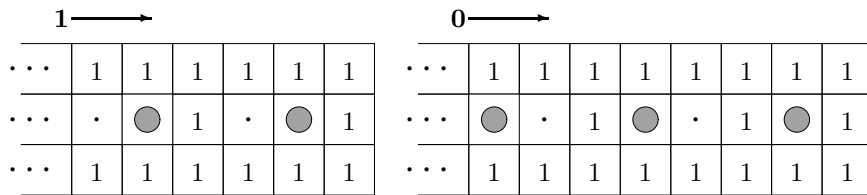


Figure 4: A wire that is forced to conduct a given value

2 Small computational components

Next, we need to have curves to bend wires (figure 5) and splitters to duplicate wires (figure 6).

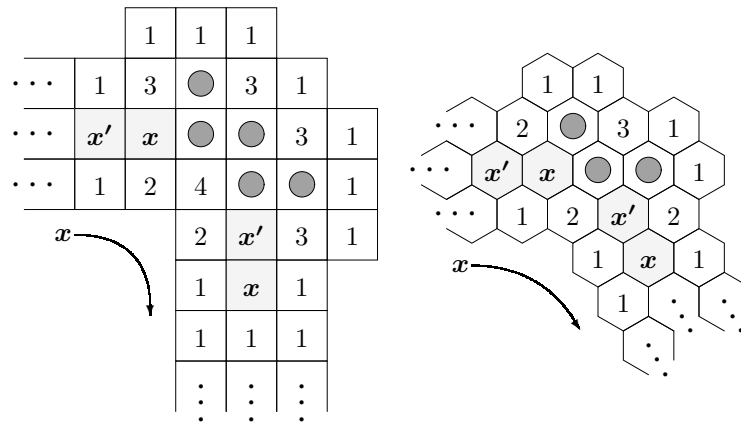


Figure 5: A curve of a wire with x

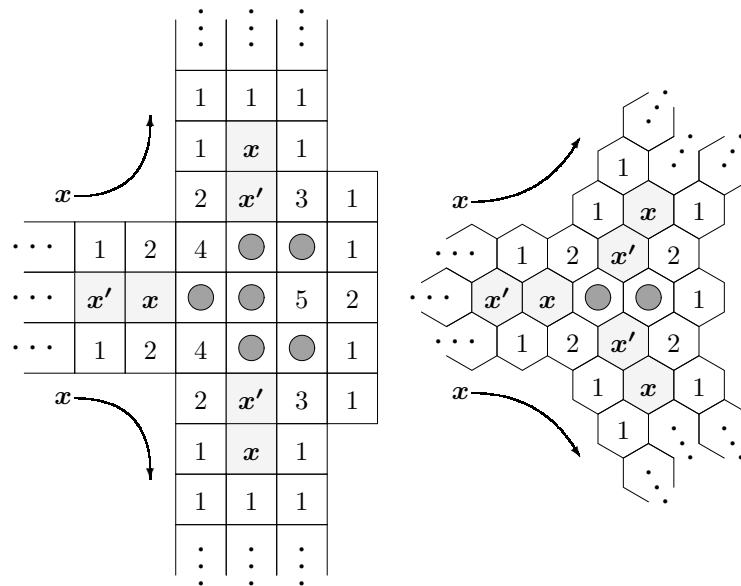


Figure 6: Merge two curves to get a splitter

Something that we might need as well are so-called phase-shifters (figure 7). The phase-shifter with square compartments is stolen from R. Kaye [3].

You might think that the phase-shifter with hexagonal compartments is a so-called inverter as well. In that case, I do not agree with you. A real inverter is shown in figure 8.

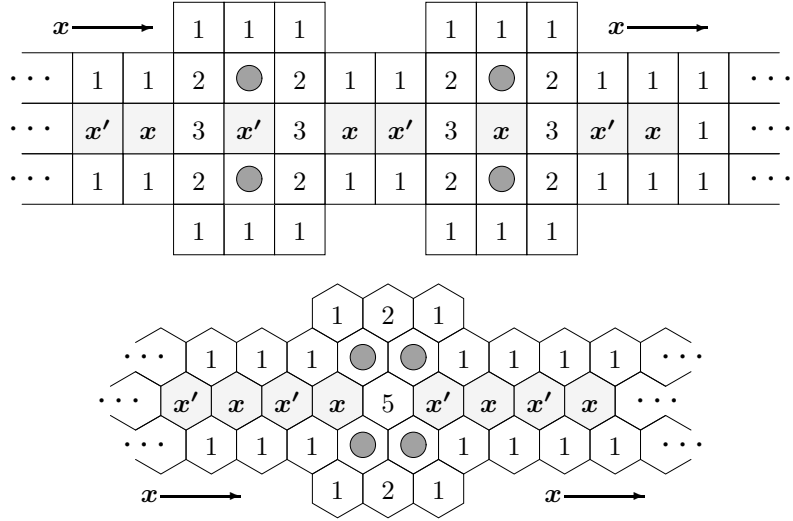


Figure 7: A phase-shifter

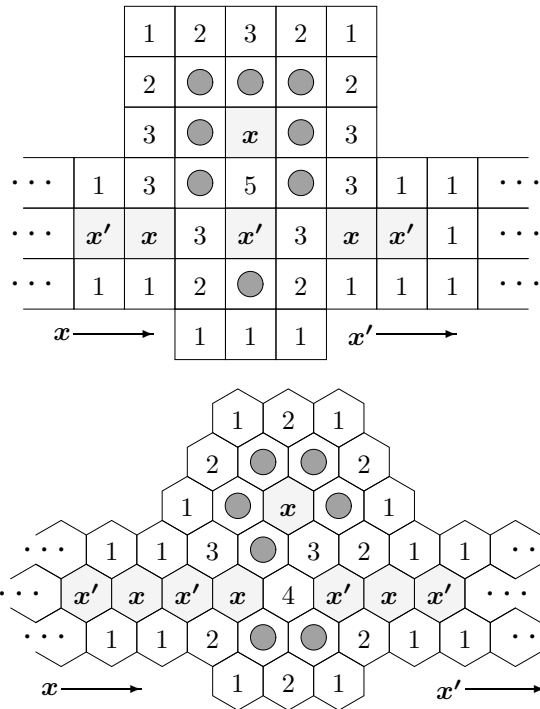


Figure 8: An inverter

In case you might not have noticed already, the hexagonal phase-shifter is not a good inverter because the number of mines might given information about the circuit then, possibly screwing up all our efforts with it.

3 Larger computational components

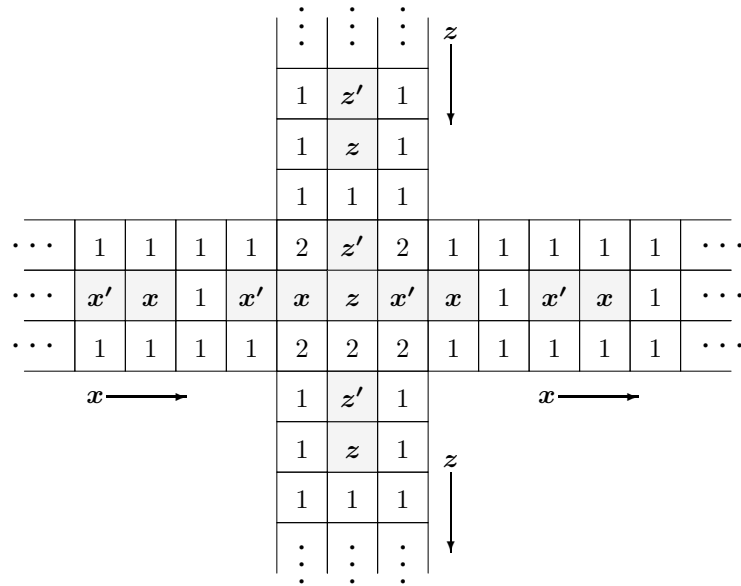


Figure 9: A crossover by R. Kaye

A crossover for Minesweeper with square compartments is shown in figure 9. It is taken from Richards Kaye's slides, see [4]. You might wonder why there is no crossover given with hexagonal compartments. The answer is that I did not found one by direct construction. But a crossover can also be made from three splitters and the same number of adders, where the adders act modulo 2, see [3] or figure 10.

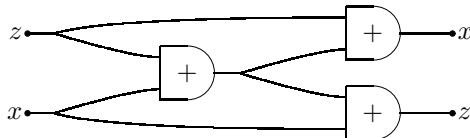


Figure 10: A crossover circuit

But before we have such a crossover, we must first make a hexagonal adder. Figure 11 shows an adder-multiplier combi, where the adder acts modulo 2. If you only want to use the adder, you just cut off the wire of the multiplier output. A wire cut-off is the same as the start of a variable in figure 1.

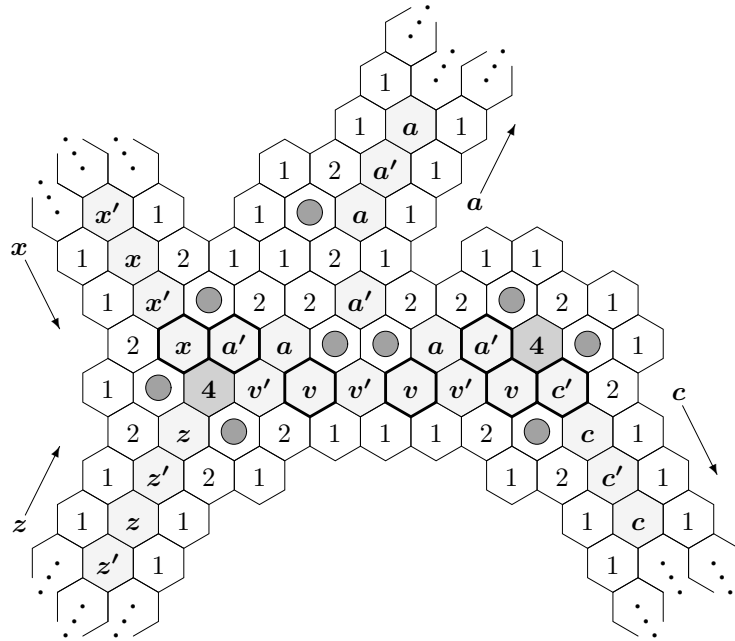
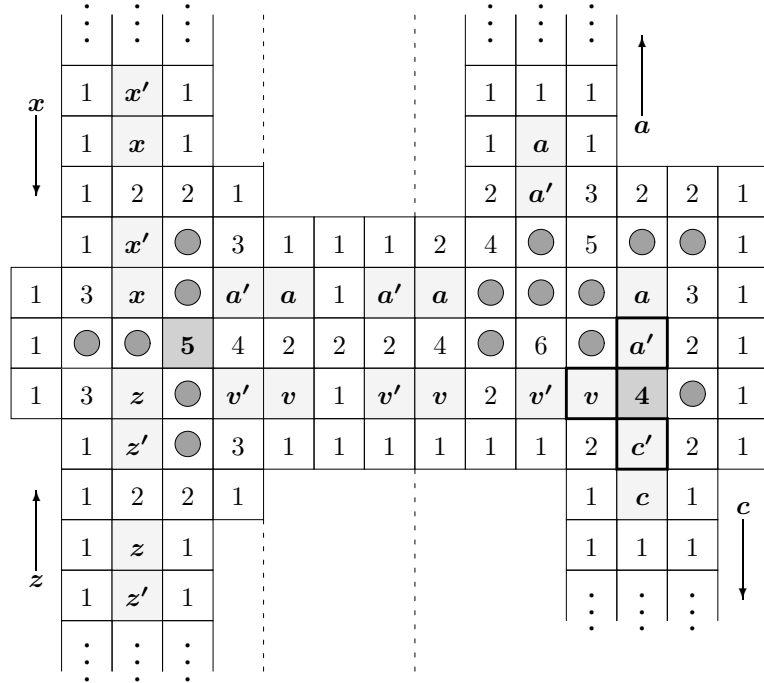


Figure 11: A component that outputs $a = x \cdot z$ and $c = x + z$

Since the adder-multiplier combi is the most complicated component by far, it needs some explanation. Let us concentrate on the one for normal minesweeper first. The part between the dashed lines is optional and its benefit will be discussed later.

The dark shaded compartment on the left hand side with the number 4 enforces the equation $x + z + a' + v' = 2$. Since also $a + v + a' + v' = 2$, the sets $\{x, z\}$ and $\{a, v\}$ are enforced to be equal. So the dark shaded compartment on the left hand side is in fact the heart of a *shaker*: its outputs a and v are a nondeterministic permutation of x and z .

We show that $a = \min(x, z)$ and $v = \max(x, z)$. Suppose that this is not the case. Then $a = 1$ and $v = 0$. So $a' = v = 0$. It follows that the dark shaded compartment on the right hand side with the number 4 is surrounded by three mines at most. This contradicts the number 4 in it, so $a = \min(x, z)$ and $v = \max(x, z)$. Thus the dark shaded compartment on the right hand side is the heart of a *tester-adder combi*: its inputs are tested and at least one of them must be equal to one.

Since $x \cdot z = \min(x, z)$, the output a equals $x \cdot z$. Next, the heart on the right hand side enforces the equation $a' + v + c' = 2$. Since $v = 0$ implies $a' = 1$ and $a' = 0$ implies $v = 1$, $1 \leq a' + v \leq 2$. Thus c' can be chosen such that $a' + v + c' = 2$.

Modulo 2, the heart on the right hand side gives the following information:

$$0 \equiv a' + v + c' \equiv a + v + c \equiv x + z + c \pmod{2}$$

It follows that $c \equiv x + z$ modulo 2.

But we are not yet done now with the adder-multiplier combi for normal minesweeper. This is because for the square compartments which are bold, revealing these compartments might give new information about the board at first glance. But that only seems so. If you e.g. reveal the bold compartment with v , then $v = 0$ and you already know before revealing it that $a' = c' = 1$.

The adder-multiplier combi for hexagonal minesweeper works essentially the same as that for normal minesweeper. Notice that the heart on the left hand side is the heart of a somewhat buggy shaker this time, since you can reveal x when $a = 1$ by way of the bold compartment with a' . But since $a = 1$ implies $x = 1$, the shaker subcomponent is correct within its context.

v is not an output of the adder-multiplier combi, but one can reconstruct v by adding a and c with another adder-multiplier combi or just only a modified version of the tester-adder combi. One can make a tester-adder combi without 'fragile' literal compartments for both normal and hexagonal Minesweeper.

4 Triangular Minesweeper

One can reduce hexagonal minesweeper to triangular minesweeper as follows. In figure 12, each hexagonal compartment on the left hand side is replaced by two triangular compartments: one with a light mine that points downwards and another that points upwards. The triangle that points upwards contains a

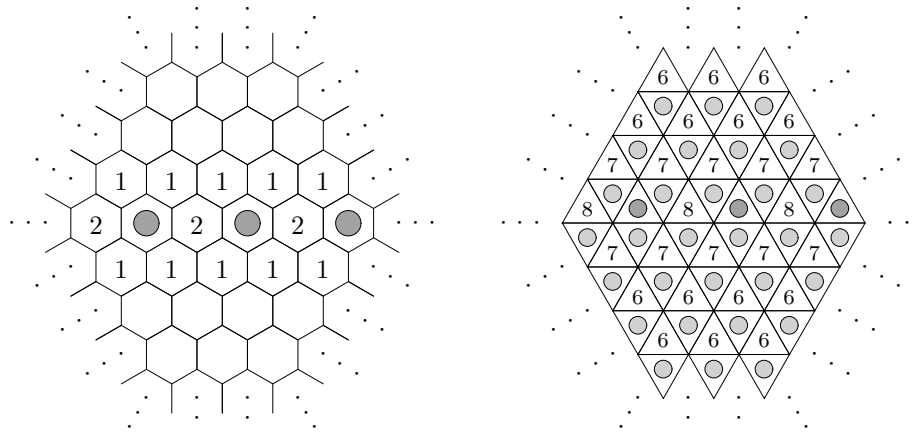


Figure 12: Hexagonal Minesweeper solvability reduces to triangular Minesweeper solvability

dark mine if the corresponding hexagonal compartment on the left hand does. Otherwise, it contains the number of the corresponding hexagon plus 6. The ‘plus 6’ counts (for) the light mines surrounding the triangular compartment. So apart for some minor border issues that are ignored here, we have a reduction from hexagonal minesweeper to triangular minesweeper here. It follows that triangular minesweeper is NP-complete.

5 A stronger solvability result for normal Minesweeper

Since Richard Kaye already proved that Minesweeper solvability is NP-complete, it would be nice to improve on that. One way to do so is to observe that Minesweeper solvability is ASP-complete. It is true that all three variants of Minesweeper discussed above are indeed shown to be ASP-complete, while Richard Kaye’s proof is not an ASP-proof (if the inputs u and v of his AND-gate are both zero, then r and s can be interchanged).

But that is not what I want to discuss here. No, we are going to show that determining the solvability of a minesweeper board of which only one square is uncovered initially is NP-complete. Of course, the uncovered square can only be surrounded by zero mines, since otherwise it would be impossible to uncover any other square, in which case you will not get any further.

Ok, say that there is one uncovered square with no mines surrounding it. Then you can uncover all surrounding squares, and for each such square that has no mines surrounding it either you can uncover the surrounding squares as well, etc. All programs for minesweeper do this automatically.

So we get an area of uncovered squares consisting of connected squares with no mines around them, from now on called a *whitespace component*. Furthermore, the border of the first whitespace component is uncovered as well, but

the border squares do have mines around them.

In order to show that determining the solvability of a minesweeper board of which only one square is uncovered initially is NP-complete, it suffices to be able to do the following by local reasoning:

- reason through wires to uncover all whitespace components,
- get to know all components except for the values of their variables.

Figure 13 shows how to get to know wires and to reason through them.

...	1	1	1	1	1	1	1	1	1	1	1	1	1	1	...	
...	?	?	?	?	?	?	?	?	?	?	?	?	?	?	...	
...	?	?	?	?	?	?	?	?	?	?	?	?	?	?	...	
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		
...	1	1	1	1	2	2	2	1	1	1	1	1	1	1	...	
...	?	?	•	?	?	●	?	?	1	?	?	•	?	?	•	...
...	?	?	?	?	?	?	?	•	1	•	?	?	?	?	?	...
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

Figure 13: Mark the phase of a wire by an extra mine and you can reason through it

All other components of normal Minesweeper that are presented here can be figured out as well, provided all whitespace components are uncovered and the phases of all wires are known (see figure 13 as well). The adder-multiplier combi needs the part between the dashed lines very sorely now.

To increase the probability that one gets as far as this solvability check, one can easily ensure that at least 99 percent of all squares do not have surrounding mines.

6 Minesweeper is PP-hard

In order to show that Minesweeper is PP-hard, we reduce from weak MAJSAT. Weak MAJSAT is the problem of estimating the probability that a circuit is satisfied by 0 or 1, with an error of at most 0.5, assuming that the inputs are random. Hence the error will always be 0.5 when the probability of satisfaction is exactly 0.5, and both 0 and 1 are valid estimates in that case. MAJSAT differs from weak MAJSAT that 0 must be the output when the probability of satisfaction is exactly 0.5, and is known to be PP-complete [6, Problem 11.5.16 (a)].

Lemma 1. *Weak MAJSAT is PP-complete.*

Proof. It is clear that weak MAJSAT is in PP, thus it suffices to show that weak MAJSAT is PP-hard. For that purpose, assume we have a circuit that computes $f(x_1, x_2, \dots, x_n)$, where the x_i are the inputs of the circuit. Build another circuit which computes $\min\{f(x_1, x_2, \dots, x_n), \max\{y_1, y_2, \dots, y_n\}\}$. The latter circuit cannot have probability 0.5 of satisfaction, and the weak MAJSAT value of it is the MAJSAT value of the original circuit. \square

Theorem 2. *Minesweeper is PP-hard.*

Proof. The only thing that we need to do in addition to the NP-completeness proof of Minesweeper solvability is to wire back the output s of our circuit to the starting points of the inputs, in such a way that these inputs can be revealed when s is known. This is done in figure 14.

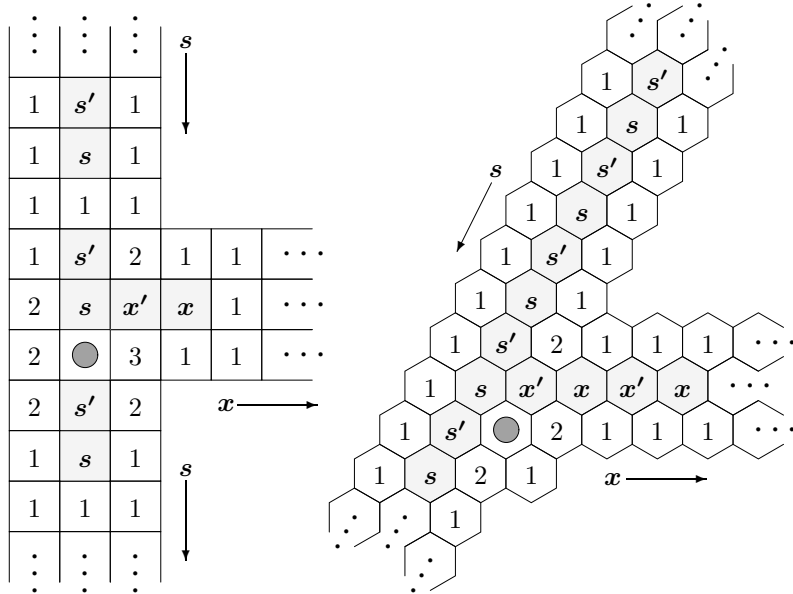


Figure 14: x can be revealed when s is known

Guessing on other spots than one with s or s' first is useless. One can already see in advance that such a guess does not give more information than that a subcircuit has a certain value, in case one does not die. It is equally useful just to assume that that subcircuit has a certain value without checking it. Thus we have to round the number ϑ that satisfies

$$\exists x_1 \exists x_2 \dots \exists x_n \Pr[f(x_1, x_2, \dots, x_n)] = \vartheta$$

to either 0 or 1, with a rounding error of at most 0.5. \square

See [5, §4] for the meaning of the random quantifier \exists . This quantifier will be used again in the next section.

7 PSPACE-completeness of Minesweeper

Now we know that Minesweeper is PP-hard, but the question remains if Minesweeper is complete for some known complexity class. Since one can show that Minesweeper is in PSPACE, it seems natural to ask whether Minesweeper is PSPACE-complete.

Theorem 3. *Minesweeper is PSPACE-complete when the probability of revealing all mines may be infinitesimal.*

Proof. Let $f(x_1, x_2, \dots, x_n)$ be a boolean formula and E be a proper subset of $\{1, 2, \dots, n\}$ of cardinality e . Assume that either $m = 2$ or $m = n + 2 - e$. We define quantifiers A_i and R_i for all positive $i \leq n + 1 + m$, as follows:

$$A_i = \begin{cases} \exists & \text{if } i \in E, \\ \forall & \text{if } i \notin E, \end{cases} \quad R_i = \begin{cases} \exists & \text{if } i \in E, \\ \exists & \text{if } i \notin E. \end{cases}$$

When $m = n + 2 - e$, we reduce from QBF, by determining the validity of

$$A_1 x_1 A_2 x_2 \cdots A_n x_n f(x_1, x_2, \dots, x_n) \quad (1)$$

When $m = 2$, we reduce from a combination of QBF and weak MAJSAT, say weak nonalternating SSAT, by rounding

$$\max \left\{ \vartheta \mid R_1 x_1 R_2 x_2 \cdots R_n x_n \Pr[f(x_1, x_2, \dots, x_n)] = \vartheta \right\} \quad (2)$$

to either 0 or 1, with a rounding error of at most 0.5. Weak nonalternating SSAT is PSPACE-complete as well, since (alternating) SSAT [5, Th. 2] can be reduced to it in a similar manner as weak MAJSAT was reduced to MAJSAT in the proof of lemma 1.

The probability of removing all mines in our Minesweeper game will lie between $\frac{3}{4} \cdot 2^{-e}$ and 2^{-e} inclusive, which is (unfortunately) infinitesimal for large e .

We lay down a circuit for $s_1 = f(x_1, x_2, \dots, x_n)$ and circuitry for $s_2 = \max\{s_1, x_{n+1}\}$ and $s_3 = \max\{x_{(n+1)+1}, x_{(n+1)+2}, \dots, x_{(n+1)+m}\}$. Besides the variables x_i for all positive $i \leq n + 1 + m$, we make variables z_{ij} , where $1 \leq i < j \leq n + 1 + m$, such that any variable z_{ij} can be revealed when x_i is known, as in figure 14.

Additionally, we make variables z_{jj} , where $1 \leq j \leq n + 1 + m$, which can be revealed when either s_2 or s_3 is known. This can be done with a reversed splitter, as in figure 15. Furthermore, for each $j \notin E$, we replace the starting point of x_j by circuitry for $x_j = z_{1j} + z_{2j} + \cdots + z_{jj} \bmod 2$, thus x_j is no longer a real variable when $j \notin E$.

To show that this construction works, notice first that there is no way to get information about the variables x_i for which $i \in E$. Thus we have to guess them. Besides that, it suffices to guess one of s_2 and s_3 . When we guess either s_2 or s_3 , we guess them to be equal to one, since that value is the most likely.

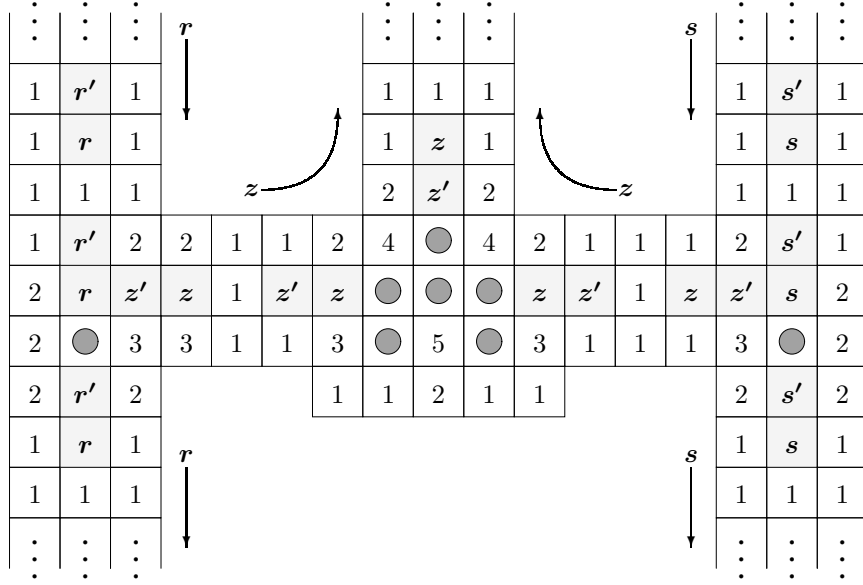


Figure 15: z can be revealed when either r or s is known

The crucial question is which of s_2 and s_3 will be subjected to a guess. At first glance, one might think that that should be the one which is most likely to equal to one, but that is not the right criterion. No, the criterion for selecting s_2 just appears to be that (1) is true or that rounding (2) yields 1, depending on the value of m .

In fact, when $s_2 = 1$ is guessed, the probability of winning will seem to be

$$2^{-e} \left(\frac{1}{2} + \frac{1}{2} \max \left\{ \vartheta \mid R_1 x_1 R_2 x_2 \cdots R_n x_n \Pr[f(x_1, x_2, \dots, x_n)] = \vartheta \right\} \right) \quad (3)$$

The probability of winning is equal to $2^{-e}(1 - 2^{-m})$ when $s_3 = 1$ is guessed.

Notice that the first guess to be done is either $s_2 = 1$ or $s_3 = 1$, since other guesses do not help more for choosing between s_2 and s_3 than making a corresponding assumption without checking. If $s_3 = 1$ is guessed with success, then the probability of revealing all mines is 2^{-e} at this stage, and that is it. But if $s_2 = 1$ is guessed with success, then the probability of revealing all mines is dependent of subsequent guesses, and these guesses are to be chosen to maximize the probability of winning.

After guessing $s_2 = 1$ with success, we do the following for $j = 1, 2, \dots, n + 1 + m$, in that order.

- If $j \in E$ and $m = n + 2 - e$, then we guess x_j in order to satisfy

$$A_{j+1}x_{j+1}A_{j+2}x_{j+2} \cdots A_{n+1+m}x_{n+1+m}f(x_1, x_2, \dots, x_n)$$

when $m = n + 2 - e$. If this is not possible, then $s_3 = 1$ should have been guessed instead of $s_2 = 1$, since in that case, (3) can be estimated by

$2^{-e}(1 - 2^{-(n+1)+e})$, which is less than the probability $2^{-e}(1 - 2^{-(n+2)+e})$ of winning when $s_3 = 1$ is guessed.

- If $j \in E$ and $m = 2$, then we guess x_j in order to maximize ϑ such that

$$R_{j+1}x_{j+1}R_{j+2}x_{j+2} \cdots R_{n+1+m}x_{n+1+m} \Pr[f(x_1, x_2, \dots, x_n)] = \vartheta$$

- If $j \notin E$, then we can reveal x_j since we know x_1, x_2, \dots, x_{j-1} and s_2 .

In a similar manner as with choosing between s_2 and s_3 , doing another guess before guessing x_j with $j \in E$ (except guessing s_2 and x_i for all $i \in E$ with $i < j$) is not better for making a good guess for x_j than a corresponding assumption without checking. Hence the answer to the crucial question which of s_2 and s_3 should be subjected to a guess is as claimed. \square

References

- [1] Anne Condon, Joan Feigenbaum, Carsten Lund, and Peter Shor, Random Debaters and the Hardness of Approximating Stochastic Functions, *SIAM J. Comput.* **26**, pp. 369–400 (32 pages), 1997.
- [2] D. Eppstein, Computational Complexity of Games and Puzzles, <http://www.ics.uci.edu/~eppstein/cgt/hard.html>
- [3] R. Kaye, Minesweeper is NP-complete, *The Mathematical Intelligencer* **22**, nr. 2, pp. 9–15, 2000.
- [4] R. Kaye, Minesweeper talk at the ASE meeting in Birmingham, Jan 3-5 2003, <http://web.mat.bham.ac.uk/R.W.Kaye/minesw/ASE2003.pdf>
- [5] C. Papadimitriou, Games Against Nature, *J. Comput. System Sci.*, **31**, pp. 288–301, 1985.
- [6] C. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.
- [7] T. Yato and T. Seta, Complexity and Completeness of Finding Another Solution and its Applications to Puzzles, *IEICE Trans. Fundamentals*, Vol. E86-A, No. 5, pp. 1052-1060, 2003.