Graphical Models for Minesweeper
Project Report

Dmitry Kamenetsky and Choon Hui Teo

November 12, 2007

1 Introduction

1.1 The game

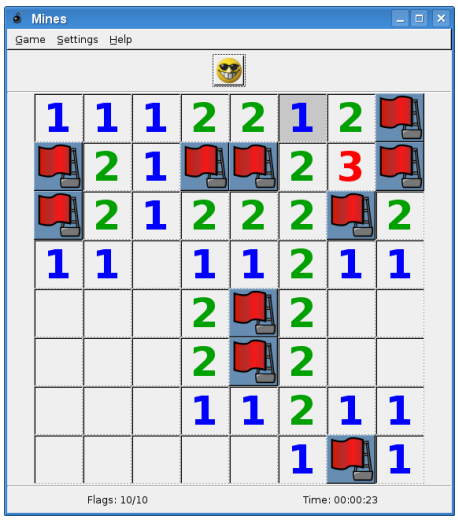
Minesweeper is a popular single-person game that was invented in the 1970's (Figure 1). A fixed number N of mines are randomly placed on a $W \times H$ grid and are hidden from the player. The game begins with an empty field. The player opens (via clicking) one square at a time. If the square contains a mine then the player blows up and the game is over. Otherwise the square becomes *uncovered* containing a single value from 0 to 8. This value indicates the total number of mines adjacent (horizontally, vertically and diagonally) to that uncovered square. The aim of the game is to incrementally determine the location of all N mines without blowing up.

1.2 Theoretical Study of Minesweeper

> Given a Minesweeper board does there exist a configuration of mines (i.e., a set of locations of mines) that satisfies all the visible numbers according to the rules of Minesweeper?

This decision problem (known as Minesweeper consistency problem) above has recently been shown to be reducible to Boolean satisfiability problem (SAT) by [1] and is therefore NP-Complete. For the purpose of the proof [1] presented Minesweeper configurations that mimic the behaviour of a *wire*, *splitter* and *crossover* as well as a set of *logic gates*: NOT, AND, OR and XOR. He then showed how those gates could be assembled to represent any type of logic circuit, thus showing its NP-completeness. In addition, [2] defined an infinite variant of Minesweeper that imitates the Game of Life, thus showing that it is also Turing-complete. [3] brought this problem to another level by defining a corresponding counting problem and show that Minesweeper is #P-complete.

Figure 1: A solved Minesweeper game on the easy 8×8 board.



2 Existing Strategies for Solving Minesweeper

2.1 Single Point

The most naive strategy for Minesweeper that is reminiscent to how beginners play is the Single Point strategy (SP). This strategy can be summarised in two simple rules:

1. If the number of mines adjacent to a square is equal to the value in that square, then we know that all the neighbouring unknown squares can be *probed* (opened) since they contain no mines. Refer to Figure 2 for an example.
2. If the number of unknown squares plus the number of mines adjacent to a square equals the value in that square, then we know that all the unknown neighbours of that squares can be *marked* (with a mine). Refer to Figure 3 for an example.

The SP examines the whole board in an attempt to match any of those two rules. If neither rule matches, then SP will perform a random probe.

Figure 2: An example of the use of the first Single Point rule. The leftmost '2' already has 2 adjacent mines and thus we can open the squares around it. Similarly we can open the squares around the '1's in the top-right corner.

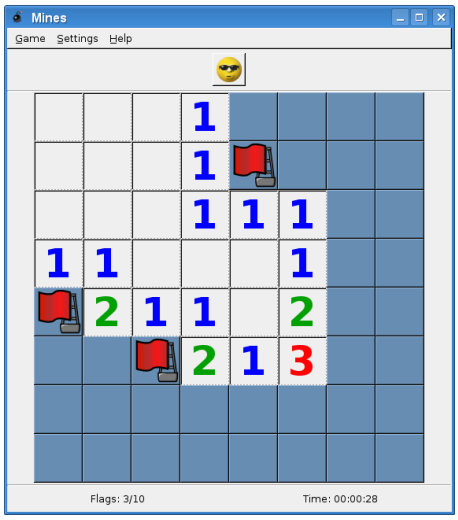


Figure 3: An example of the use of the second Single Point rule. The '2' only has 2 unknown neighbours so we mark each one as a mine. Using the same knowledge we can mark all the unknown squares around the '8'.

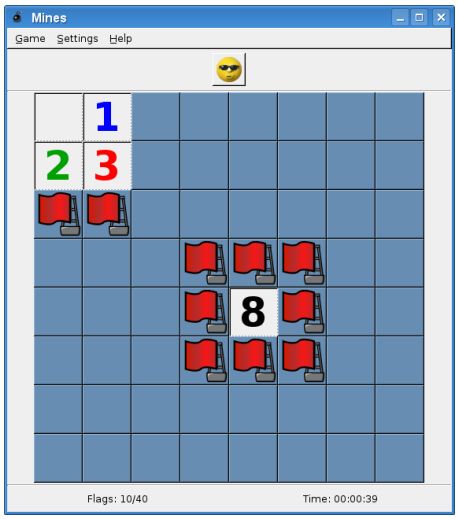


Table 1: Example of CSP solver.

1	A
1	B
1	C

2.2 Equation

A more sophisticated Minesweeper solver is the Equation Strategy (EQ) solver designed by [4]. EQ represents the information available on the board as a set of linear integer equations. The strategy assumes that each UNKNOWN square t takes the value of x_t, which is either 1 if t has a mine or 0 if it is not a mine. An equation is generated for each square with a NUMBER in the form of $c = \sum_{t \in S} x_i$, where c is the value of that square and S is the set of its UNKNOWN neighbours. For example consider Table 1. For this position we can write down the following set of equations:

1. $A + B = 1$
2. $A + B + C = 1$
3. $B + C = 1$

Solving this system we find that $A = 0$, $B = 1$ and $C = 0$ is the only solution. When presented with a board for which multiple solutions exist, EQ computes the probability $P(t)$ that each UNKNOWN square t is a mine. $P(t)$ is defined as $\max_{t \in S}(c/|S|)$. EQ then probes the square t with the lowest $P(t)$.

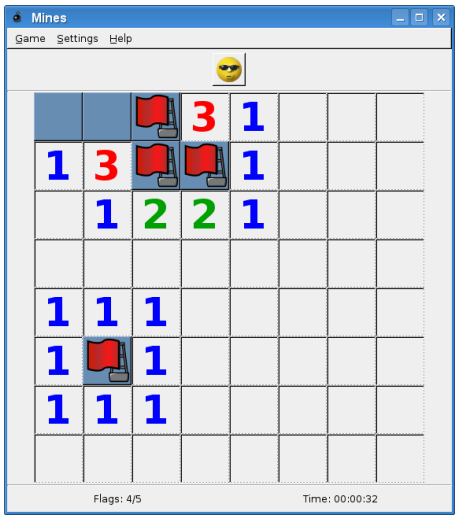
2.3 Constraint Satisfaction

The current state-of-the-art Minesweeper solver is the Constraint Satisfaction Problem (CSP) solver written by [5]. Just like EQ, CSP builds a set of constraints in the form of linear integer equations. CSP uses a more sophisticated rule set for determining the solution, as well as a backtracking algorithm. When it comes to guessing, CSP generates the set of all solutions and then computes the probability that each square is not a mine. Often its estimated probability of success is within a few percent of the actual success rate [5].

3 Importance of Guessing

95% of expert Minesweeper boards cannot be solved by logic alone and require some kind of guessing [5]. Even worse, in some cases the guess must be completely random. For example, in Figure 4 the last mine could be at either UNKNOWN square but not both. The probability of the mine being in one of them is 50% and in this situation we have no alternative but to guess one of them.

Figure 4: In this situation we need to guess. The probability of our guess succeeding is exactly 50%.



In other cases an educated guess can be made. Consider the position in Table 2. If we consider all the possible configurations of mines such that the two numbers are satisfied then we will see that location A is least likely to be a mine and location is most likely to be a mine. Thus we can make an educated guess at A with probability of succeeding (not blowing up) at 75%.

Table 2: The probability of a mine being at A is 25%, at B is 50% and at C is 75%.

1	A	
	B	
	C	2

1			1			1			1	⋆	
	⋆	⋆		⋆		⋆		⋆			⋆
		2		⋆	2		⋆	2		⋆	2

When guessing, it is not sufficient to consider the probability of success (probability of not blowing up). One must also take into account the amount of useful information a successful guess can reveal. Consider the position in Table 3. The location at 4 has 3 empty neighbours and needs a further of 2, so the probability that A is a mine is $2/\binom{3}{2} = 2/3$. Thus the probability that guessing at A will succeed is 1/3. Similarly the probability that the guess at B or C will succeed is 1/2. Intuitively it seems that guessing at B or C is better. However, consider the outcome of those guesses. If guessing at B succeeds then 5 will appear under B and we know that C is a mine. If guessing at C succeeds then 3 will appear under C and we know that B is a mine. None of those guesses help us to determine anything about A. In contrast, if guessing at A succeeds then either 6 or 7 will appear under A, from which we can deduce what B is and hence solve the position.

Table 3: ⋆ indicates a mine. In this case, guessing at A is more dangerous than at B or C. However, unlike the guesses at B or C, the guess at A, if successful, provides us enough information to solve the position.

⋆		⋆	2
4	A	B	C
⋆		⋆	2

4 Our Approach

4.1 Graphical Model for Minesweeper

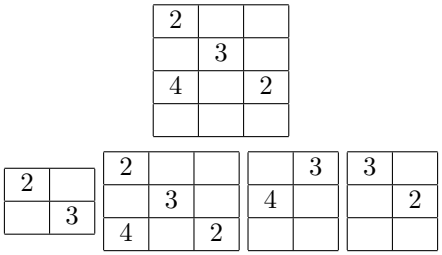
Minesweeper is played on a grid board B and the state of a square B_{ij} *depends* on its neighbours that are on the board $\text{Ne}(B_{ij}) := \{B_{kl} : k \in \{i-1, i, i+1\}, 1 \le k \le W, l \in \{j-1, j, j+1\}, 1 \le l \le H\}$, hence it is natural to model it as a grid-based graphical model. Each square of the board can be in one of the following states {UNKNOWN, MINE, NUMBER}, where NUMBER$\in$ {0,1,2,3,4,5,6,7,8}. In the graphical model terminology, UNKNOWNsquare B_{ij} corresponds to an *unobserved* discrete-valued random variable $X_{ij} \in$ {MINE, NOT_MINE}, whereas MINEand NUMBERcorrespond to an *observed* random variable.

The most difficult part of this project was finding the right way to define cliques and their corresponding potential functions. As the default, each clique must contain at least one NUMBER square, otherwise we cannot infer anything from it. This implies that we never consider the whole board and only make predictions for regions around the uncovered squares, thus saving us computation. Furthermore, each clique must contain at least one UNKNOWN, otherwise its value is fixed. The Hammersley-Clifford theorem states that the joint distribution $p(x)$ can be always factorized as a product of potentials over maximal cliques. In our case the maximal cliques are fully connected graph of 4 nodes (K_4), however we use larger cliques since we could not define suitable potential functions for K_4 cliques. Below are the cliques and potential functions that we tried.

4.1.1 Hard cliques

For each B_{ij} that is a NUMBER we construct a clique containing all the adjacent nodes of that square (*hard cliques*) (see Table 4, ie. $C_{ij} = \{X_{kl} : B_{kl} \in \text{Ne}(B_{ij})\}$. The potential for clique C_{ij} is then 1 if the number of MINE's in $\text{Ne}(B_{ij})$ is equal to B_{ij} and 0 otherwise. This idea did not work well because the potential function only attempts to satisfy the central number, instead of all the numbers, in the clique.

Table 4: Above: The original board position. Below: The 4 *hard cliques* that are extracted from the original position. Each clique is centered around one of the 4 numbers.



4.1.2 3x3 cliques

The second idea was to use all possible 3×3 cliques and associate each one with one of two potential functions. The cliques that happened to be *hard cliques* are most naturally associated with the same potential as above. All the other cliques are associated with a new potential. The new potential returns 0 if any of the NUMBER's in the clique are not satisfied and 1 otherwise.

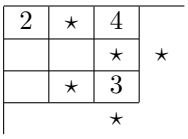
4.1.3 Correct potential

Finally we realized that a single potential function could be written for the *hard cliques*. let m_{ij} be the number of mines around B_{ij} and e_ij the number of UNKNOWN squares around B_{ij} *outside* the current clique. The potential is 1 only if for each X_{ij} corresponding to a NUMBER B_{ij} the following two conditions hold; otherwise it is 0:

- There cannot be more mines than the value of X_{ij}: $m_{ij} \leq X_{ij}$.
- There must be enough UNKNOWN squares outside the current clique to satisfy the value of X_{ij}: $m_{ij} + e_{ij} \geq X_{ij}$.

See Table 5 for an example.

Table 5: Consider a 3x3 clique in the top-left corner of the board. For '2' $m = 1$ and $e = 0$, thus '2' fails because there are not enough empty squares outside this clique to satisfy its value. For '4' $m = 3$ and $e = 1$, if we place a mine in the extra empty square then we can still achieve the required 4 mines. For '3' $m = 4$ and $e = 3$. Although there are only 2 mines neighbouring '3' inside the clique, '3' fails once we also include the mines outside the clique.



4.1.4 Merging cliques

To assist the inference process we increase the size of the cliques. This is achieved by repeatedly merging *hard cliques*. At each iteration of the algorithm, we find the clique with the most number of UNKNOWN. We then check whether this clique can be merged with any of its neighbouring cliques, such that the number of UNKNOWN in the resulting clique is less than a threshold K. If the clique can be merged then we perform the merge and find the next clique to merge; otherwise the clique cannot be merged and is removed from the list of cliques available for merging. The algorithm terminates when there are no more cliques to be merged. The threshold K plays an important role, since the complexity of the inference algorithm grows exponentially with K. However as K increases to $W \times H$, the inference becomes closer to being exact. In our experiments we use $8 \leq K \leq 12$. Note that K must be at least 8, since this is the value of a *hard clique* with a single central NUMBER.

	Beginner	Intermediate	Expert
Board size	8×8	13×15	16×30
No. of Mines	10	40	99
Mine ratio (%)	15.63	20.51	20.63

Table 6: Common difficulty levels of Minesweeper

4.2 Inference

For each supplied board position we perform the following steps:

1. Construct all the cliques. Merge cliques if necessary. Associate each clique with a potential function and define its semiring.
2. Initialize the Loopy Belief Propagation (LBP) algorithm with this set of cliques.
3. Compute the marginal distribution for each node X_{ij} that is UNKNOWN.
4. If any of the node's marginal probabilities are exactly 0.0 or 1.0 then we can return the result immediately. A probability of 0.0 means that the square is a *sure mine* and can be *marked*, while 1.0 means that the square is a *sure empty* and can be *probed*.
5. If a simple marginalization found no answers then we perform another 3 iterations [1] of loopy belief propagation. After each single iteration we repeat steps 3 and 4.
6. If loopy belief propagation could not converge to an exact answer then we need to make an educated guess. Here we probe a square whose probability of being UNKNOWN is greatest. If there are more than one such squares then we pick one randomly.

For our experiments we use the sum-product semiring. We found that the boolean semiring does not give us the probabilities required for making a guess (see below). The max-product semiring is also a bad choice because the probabilities that it gives do not correspond to the actual probabilities.

5 Experiments

5.1 Results

We compare the performance of our method with merging cliques (LBP-MC), and without merging cliques (LBP) to the existing methods, namely, Single Point (SP), Equation (EQ) and Constraint Satisfaction (CSP) using the minesweeper engine and GUI – PGMS written by [4].

Instead of comparing the methods with different and randomly generated games, as was done in previous work [5], we fix a (randomly generated) game set of 1000 games for each level of difficulty: beginner, intermediate and expert. The details of these levels of difficulty is presented in Table 6. Then we measure the performance in terms of:

1. **Win ratio:** the number of games won over total number of games played.
2. **Area uncovered:** the number of squares uncovered over the total number of squares on the board.
3. **Mines found:** the number of mines identified correctly over the total number of mines on the board

The results of the experiments are summarised in Tables 7, 8, and 9.

5.2 Analysis

Minesweeper is essentially a constraint satisfaction problem, since it can be easily transformed into a set of equations; one equation for each NUMBER. Despite this fact, we have shown that it is possible to model the

[1] The reason for such a small number of iterations is explained in the Results section.

	LBP-MC	LBP	SP	EQ	CSP
Win ratio (%)	78.60	73.70	44.60	85.70	84.80
Area uncovered (%)	92.47	87.52	73.74	95.20	94.43
Mines found (%)	89.77	84.83	61.97	92.96	92.09

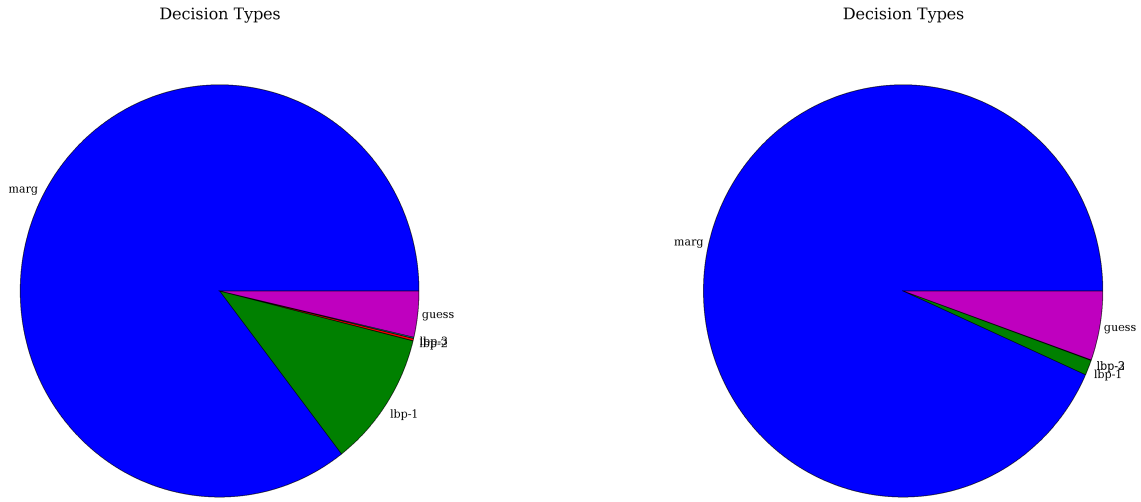
Table 7: Experimental results of LBP-MC, LBP, SP, EQ and CSP on 1000 beginner games.

	LBP-MC	LBP	SP	EQ	CSP
Win ratio (%)	44.80	37.90	5.10	50.20	54.50
Area uncovered (%)	77.45	69.15	39.01	77.75	80.77
Mines found (%)	75.15	67.13	30.49	75.17	78.37

Table 8: Experimental results of LBP-MC, LBP, SP, EQ and CSP on 1000 intermediate games.

	LBP-MC	SP	EQ	CSP
Win ratio (%)	33.10	0.40	34.60	39.20
Area uncovered (%)	70.18	27.96	71.40	72.19
Mines found (%)	68.91	22.90	69.60	70.69

Table 9: Experimental results of LBP-MC, SP, EQ and CSP on 1000 expert games.



(a) (b)

Figure 5: The frequency of each decision type during inference: marginalization, loopy BP (1st, 2nd and 3rd iteration) and guess; (a) for LBP and (b) LBP-MC.

Bins (%)	LBP	LBP-MC
(40,50]	38.50	44.03
(50,60]	50.00	70.00
(60,70]	72.73	72.32
(70,80]	50.00	80.99
(80,90]	50.00	85.71
(90,100]	48.25	72.32

Table 10: Accuracy of Guesses for LBP and LBP-MC algorithms. The bin column shows the predicted success rate of the guess, while LBP and LBP-MC columns show the actual success rate of that guess, for LBP and LBP-MC respectively.

Minesweeper game using a graphical model. Our method (LBP-MC) clearly outperforms the naive Single Point strategy and comes close to the two methods (EQ and CSP) that are better suited for this task (see Tables 7, 8 and 9 for details).

Our graph structure changes for each new board configuration. To make matters worse, it is not a grid, which is the model typically used in graphical models. Therefore there is no efficient algorithm for constructing a junction tree for our model. This rules out exact inference such as the Junction Tree.

Overall, we conjecture our method does not outperform other methods due the following deficiencies:

1. It does not enforce global constraint, namely, the total number of mines on the board.

   We give a scenario where global information such as the total number of mines on the board helps in playing the game (See Table 11).

Table 11: Suppose the board configuration is the one shown at the top. The placement of the mines varies depending on the total number of mines available (global constraint). If we know that there must be exactly 3 mines, then the placement of mines can only be the one shown in bottom left. However, if we know that there must be exactly 2 mines then we can place the mines as shown in bottom right.

	2	
1		1

⋆	2	⋆
1	⋆	1

	2	
⋆		⋆
1		1

2. Since the graph contains loops, Loopy Belief Propagation may not converge to the global optimum, hence the prediction may not be very accurate.

   It is well-known that loopy BP does not guarantee global optimum. So, we are left to check that the percentage of the board configurations that cannot be solved by marginalization alone is relatively high.

   Table 12 gives a clue on how frequently our method needs to rely on results of loopy BP (called "guess"). Since loopy BP does not guarantee to converge to an optimal result, its prediction is likely to be incorrect.

From the results (Tables 7 and 8) we can see that merging cliques (LBP-MC) improves the playing strength of our old method (LBP). The main reason for this is that once the cliques are merged our algorithm relies much less on loopy BP's propagation, which as noted earlier, can converge to incorrect results. In fact as Fuigure 5 shows, in most board configurations LBP-MC can obtain a result with marginalization alone. Table 10 shows that LBP-MC's guesses are much closer to the actual success of the guess.

6 Conclusion and Future Work

We have seen that the graphical model is a very competitive tool for solving board games which require interaction between cells. However, the model cannot handle global constraint efficiently (if not impossible). One way this can be achieved is to create a single clique (*global clique*) that contains all the unopened squares. The potential for this clique will then be 1 if the number of mines matches the total number of mines available. Note that the complexity of such a method would grow exponentially with the size of the *global clique*, thus we would only use this method if the number of UNKNOWN squares is sufficiently low (say less than 15).

Other improvements can be envisaged:

- At the moment we are only using binary labels (MINE and NOT_MINE). We could also introduce labels for each of the 9 classes of NOT_MINE.

Level	Lost With Guess	Win With Guess	Win Without Guess
Beginner	21.4%	22.8%	55.8%
Intermediate	55.2%	30.9%	13.9%

Table 12: Proportion of win and lost games due to guess.

- We can improve our guesses by looking at the outcome of each guess, as well as its probability of success (see Table 3 for an example).
- We could find a better strategy for the merging of cliques that improves the convergence of loopy BP. Alternatively we could find a way to construct the junction tree from our cliques and hence use the Junction Tree algorithm.
- We can redesign the potential function so that it returns a probability, instead of a boolean value. This should help us when it comes to guessing.
- At the moment, for each new move we have to reconstruct the nodes and cliques from scratch, which is incredibly inefficient. It would be good to add/remove cliques incrementally as we play the game.

Finally, we plan to extend the current LBP Minesweeper solver to a *new* game – codenamed `GoldMine` which is similar to Minesweeper but with "gold" placed on the board. The gold has higher density than the mine thus the NUMBER indicating the gold is twice the NUMBER of mine. The aim of the game is then to dig as much gold as possible yet avoid being killed by the mines.

References

[1] Richard Kaye. Minesweeper is NP-complete. *Mathematical Intelligencer*, 22:9–15, 2000.

[2] Richard Kaye. Infinite versions of minesweeper are Turing-complete. 2002.

[3] Preslav Nakov and Zile Wei. Minesweeper,#-minesweeper. 2003.

[4] John Ramsdell. Programmer's minesweeper.

[5] Chris Studholme. Minesweeper as a constraint satisfaction problem. *Unpublished project report*, 2000.