Binary Puzzle is NP–complete

Marzio De Biasi

marziodebiasi [at] gmail [dot] com

July 2012

Version 1.01: second version with figures and more details

The latest version of this paper can be found at:

http://www.fractalmuse.org

Abstract

We prove that the puzzle game `Binary Puzzle`, also known as `Binary Sudoku`, is NP-complete.

1 Introduction

Binary Puzzle (also known as *Binary Sudoku*) is an addictive puzzle played on a $n \times n$ grid; intially some of the cells contain a zero or a one (*fixed cells*); the aim of the game is to fill the remaining *empty cells* according to the following rules:

- Each cell should contain a zero or a one.
- No more than two similar numbers next to or below each other are allowed.
- Each row and each column should contain an equal number of zeros and ones.
- Each row is unique and each column is unique.

An example of a solved game is shown in Figure 1.

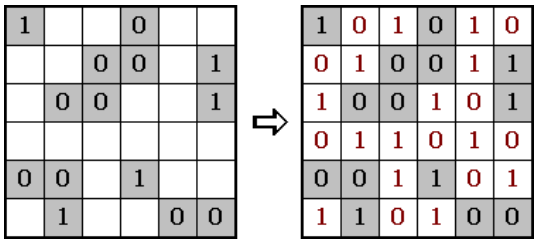


Figure 1: A solved Binary Puzzle game.

In line with the recent focus on the complexity of puzzle games [2] [1], we study how hard it can be to solve a Binary Puzzle game; in particular we prove that given a partially filled $n \times n$ grid, it is NP-complete to decide if the game has a valid solution.

In Section 2 we formally define the decision problem associated to the puzzle game; in Section 3 we briefly describe the planar 3CNF NP–complete problem; in Section 4 we describe the gadgets that can be used to reduce a planar 3CNF problem to a Binary Puzzle game and in Section 5 we give the details of the reduction to prove the NP–completeness of the game.

2 Definitions

Definition 2.1. decision problem `BINARY PUZZLE` :

Instance: A `Binary Puzzle` game, i.e. a partially filled $n \times n$ grid specified with a string $g \in \{\bot, 0, 1\}^{n^2}$ ($\bot$ is used to represent an empty cell).

Question: Does a valid solution for the game exist? I.e. can we fill the empty cells with a zero or a one following three rules:

R1. no more than two similar numbers next to or below each other are allowed;

R2. each row and each column should contain an equal number of zeros and ones;

R3. each row is unique and each column is unique.

We also define a variant of the game, that is equal to Binary puzzle but rules $R2$ and $R3$ are ignored.

Definition 2.2. decision problem `BARE BINARY PUZZLE` :

Instance: An $n \times n$ `Binary Puzzle` game.

Question: Does a valid solution for the game exist? I.e. can we fill the empty cells with a zero or a one in a way that:

R1. no more than two similar numbers next to or below each other are allowed.

3 Planar 3CNF

A CNF formula is *planar* if the bipartire graph between clauses and literals plus all edges $(x_i, \bar{x_i})$ forms a planar graph.

For example the planar graph corresponding to the planar CNF formula:

$$(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (x_3 \vee x_4 \vee \bar{x}_5)$$

is shown in Figure 2.

The problem of deciding whether a planar CNF formula is satisfiable or not is NP–complete [3] and it remains NP–complete even with the additional constraint that each clause must contain exactly three literals (*planar 3CNF*).

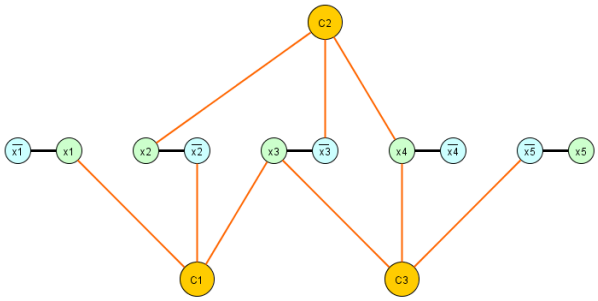


Figure 2: A planar 3CNF formula $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (x_3 \vee x_4 \vee \bar{x}_5)$.

4 Gadgets

We prove the NP–hardness of `BARE BINARY PUZZLE` using the following idea: given a planar 3CNF formula, we can reduce the maximum degree of its graph to *degree* 3 replacing the edges from each variable to the clauses with a cascade of degree 3 "Y" nodes. Then we can build its orthogonal representation in polynomial time [4]. From the orthogonal representation we build an equivalent Binary Puzzle grid replacing the literal pairs $(x_i, \bar{x}_i)$, the clauses c_i and the (orthogonal) edges with a corresponding *gadget*.

Every gadget is a closed 21×21 portion of the grid with a border of fixed cells; some of the inner cells will be fixed, and some will be empty. A gadget can interact with the adjacent gadgets only through the middle empty cells; we will call them *interface cells* (see Figure 3). The construction of each gadget is such that not all combination of values $0, 1$ can be assigned to the interface cells: some combinations lead to an *invalid configuration* i.e. the empty cells of the inner part of the gadget cannot be filled without breaking rule $R1$ of the game. We will also logically distinguish between *input interface cells* and *output interface cells*: a particular value placed on an input cell of a gadget will force a corresponding value on the output cell(s).

We describe each gadget using the following convention in the figures: gray cells represent fixed border cells, blue cells represent the fixed cells that "implements" the logic of the gadget, orange cells represent the interface cells.

We used a simple constraint solver program to test every combination of $0, 1$ values in the interface cells of each gadget in order to check if a corresponding *valid configuration* of the inner empty cells exist.

4.1 Variable gadget

The *Variable gadget* has two output interface cells T, F, and is used to simulate an assignment of truth value to a variable. As shown in Figure 4 it is not possible to correctly fill the Variable gadget if we place a 1 in both the interface cells.

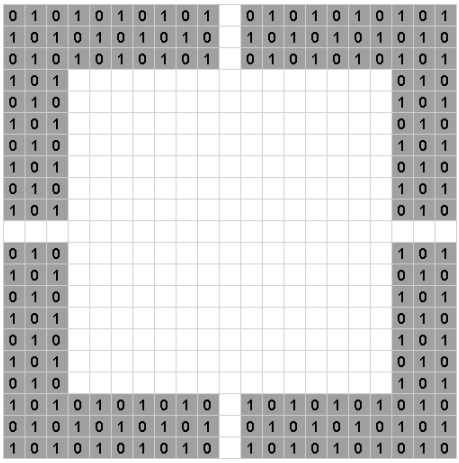
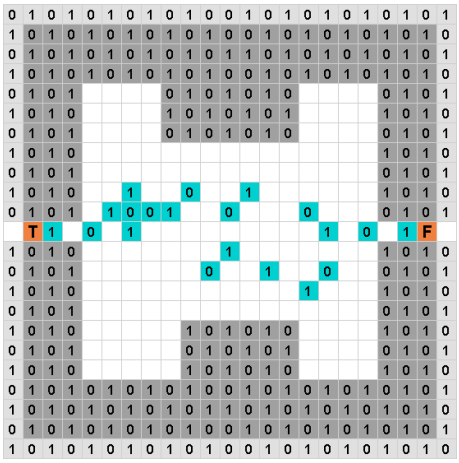
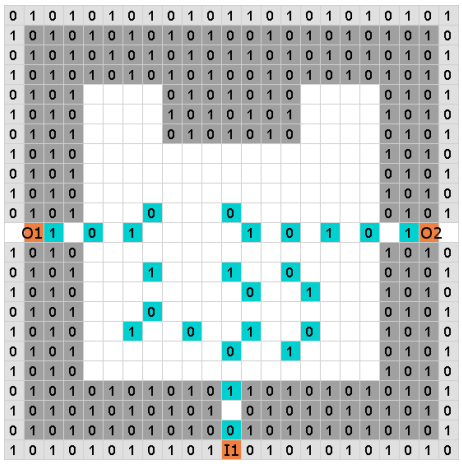


Figure 3: The 21×21 border of fixed cells common to all gadgets, and the interface (empty) cells.



Variable gadget		
T	F	Valid
0	0	YES
0	1	YES
1	0	YES
1	1	NO

Figure 4: Variable gadget



Split gadget			
I1	O1	O2	Valid
0	0	0	YES
0	0	1	NO
0	1	0	NO
0	1	1	NO
1	0	0	YES
1	0	1	YES
1	1	0	YES
1	1	1	YES

Figure 5: Split gadget

4.2 Split gadget

The *Split gadget* is used to duplicate an edge and has one input interface cell $I1$ and two output interface cell $O1, O2$. As shown in Figure 5 the two output cells can contain 1 only if $I1$ contains 1.

4.3 Edge gadgets

The edge gadgets are used to connect the other gadgets and simulate the edges of the planar 3CNF formula. They are of two types: *Line gadget* and *Turn gadgets*. As shown in Figure **??**, Both have an input interface cell $I1$ and an output interface cell $O1$: if cell $O1$ contains a 1 then cell $I1$ must contain a 1, too.

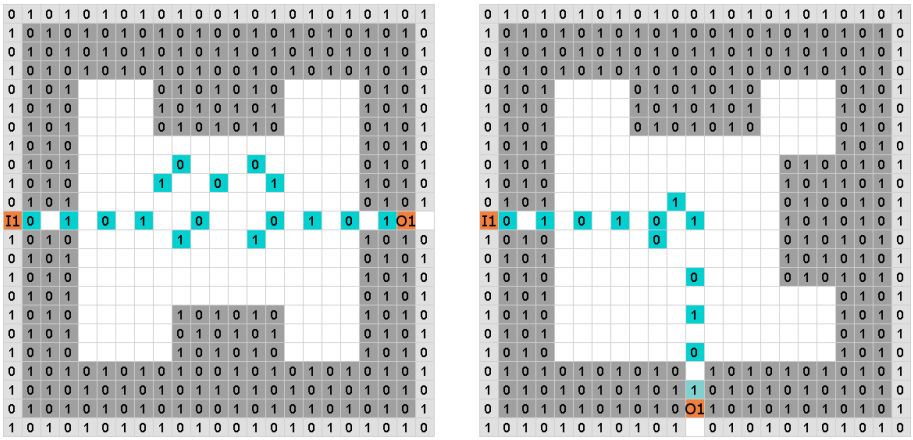
4.4 OR gadget

The *OR gadget* is used like a logical OR gate: it has two input interface cells $I1, I1$ and an output interface cell $O1$. As shown in Figure 7, in order to fill cell $O1$ with a 1 at least one of the two inputs must be 1.

4.5 Clause gadget

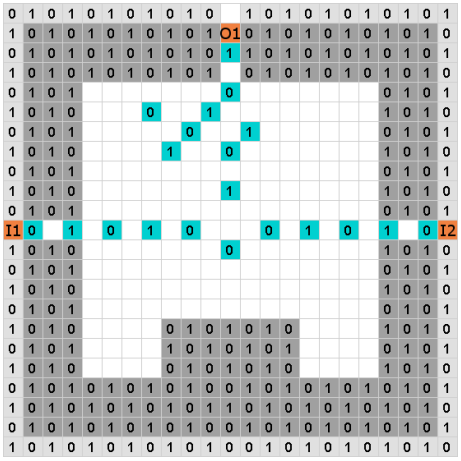
The *Clause gadget* is built using two OR gadgets linked together; one of the two gadgets has its output interface cell O_1 forced to 1. As shown in Figure **??**, it has three input interface cells are I_1, I_2, I_3 and at least one of them must be filled with a 1.

We can extend the set of gadgets rotating their inner logic by 90, 180 and 270 degrees (the fixed border blocks remain unchanged).



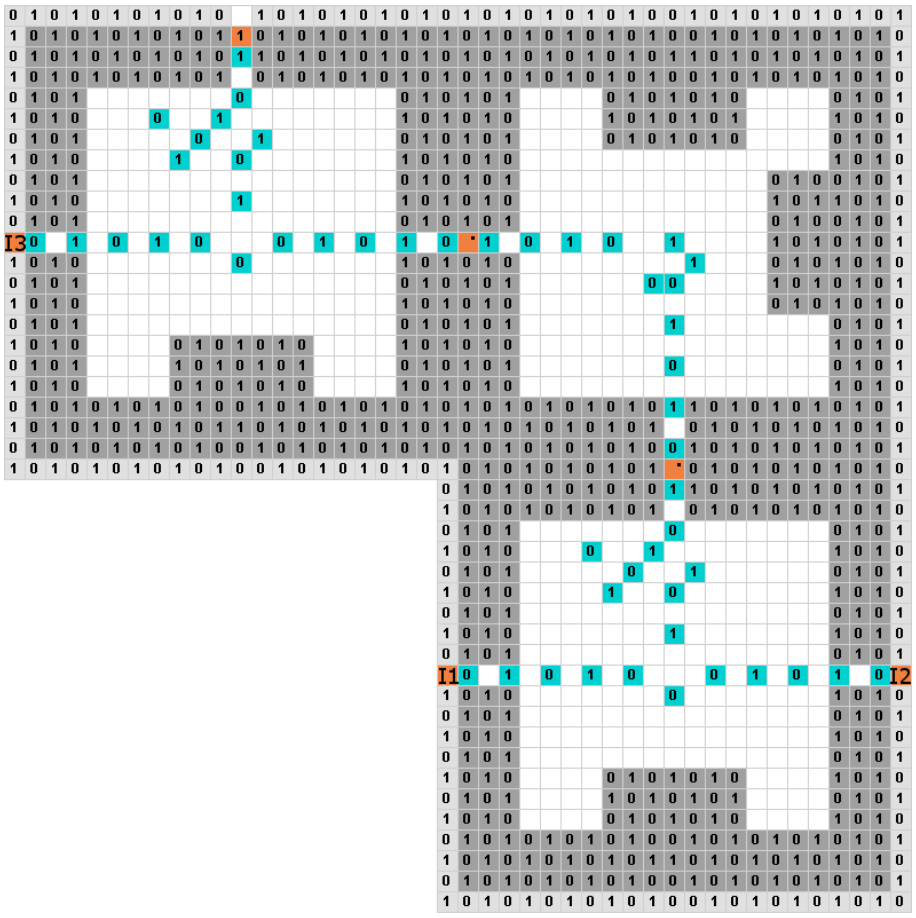
Edge gadgets		
I1	O1	Valid
0	0	YES
0	1	NO
1	0	YES
1	1	YES

Figure 6: Edge gadgets.



Or gadget			
I1	I2	O1	Valid
0	0	0	YES
0	0	1	NO
0	1	0	YES
0	1	1	YES
1	0	0	YES
1	0	1	YES
1	1	0	YES
1	1	1	YES

Figure 7: OR gadget



Clause gadget			
I1	I2	I3	Valid
0	0	0	NO
0	0	1	YES
0	1	0	YES
0	1	1	YES
1	0	0	YES
1	0	1	YES
1	1	0	YES
1	1	1	YES

Figure 8: Clause gadget.

5 Reduction

Given a planar 3CNF formula F with n variables $x_1, x_2, ..., x_n$ and m clauses $c_1, c_2, ..., c_m$; we first reduce the max degree of the corresponding planar graph to degree 3 adding some *split nodes* between the variables and the clauses: if the positive literal x_i is connected to clauses $c_{i_1}, c_{i_2}..., c_{i_k}, k \geq 2$, we add $k-1$ split nodes $y_{i_1}, ..., y_{i_{k-1}}$ and replace the k edges (x_i, c_{i_j}) with new edges $(x_i, y_{i_1}), (y_{i_1}, y_{i_2}), ..., (y_{i_{k-2}}, y_{i_{k-1}})$ and $(y_{i_1}, c_{i_1}), (y_{i_2}, c_{i_2}), ...,(y_{i_{k-1}}, c_{i_{k-1}}), (y_{i_{k-1}}, c_{i_k})$. We do the same for the negative literals $\bar{x}_i$.

Then we build an orthogonal representation of the resulting planar graph that can be embedded in a square grid of size $N = (n+m)^2)$ (see an example in Figure 9).

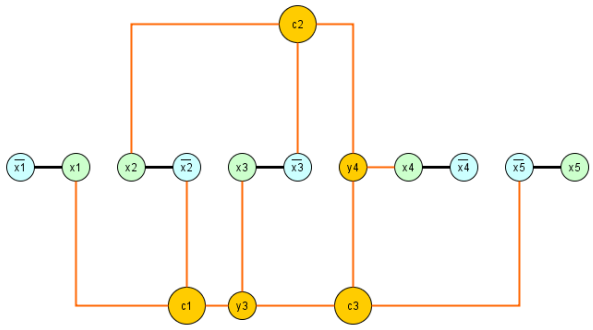


Figure 9: Orthogonal representation of the planar 3CNF formula of Figure 2; y_3 and y_4 are the split nodes added to reduce the maximum degree to degree 3.

We build an equivalent `BARE BINARY PUZZLE` game P of size $21N \times 21N$ replacing:

- nodes $x_i, \bar{x}_i$ and edge $(x_i, \bar{x}_i)$ with a Variable gadget X_i;
- each split node y_{i_j} with a Split gadget S_{i_j} (with the input interface cell I_1 directed toward the corresponding source variable);
- each clause c_j with a Clause gadget C_j;
- the (orthogonal) edges with the corresponding Edge gadgets.

5.1 `BARE BINARY PUZZLE` is NP–hard

Theorem 5.1. *The decision problem* ***`BARE BINARY PUZZLE`*** *is* NP*–hard.*

Proof. First, we notice that each step of the construction of P (degree reduction, orthogonal representation and equivalent game construction) can be performed in polynomial time and space.

We must prove that the resulting game P has a solution if and only if the original planar 3CNF formula F has a solution.

($\Rightarrow$) Suppose that the game P has a solution, then every Clause gadget has at least one input interface cell containing 1, which is also the output cell of an Edge gadget; but if an output cell of an Edge gadget is 1 then its input cell must be a 1, too. If we follow the Edge gadgets we can reach one (or more) Split gadget, but, again, if an output cell of a Split gadget is 1 its input cell must be 1, too. At the end we reach one of the two output interface cells of a Variable gadget X_i and it must contain 1. We set the associated variable x_i to the corresponding truth value. Each Clauses can generate only one assignment ($T_i = 1$ (exclusive) or $F_i = 1$); and in this way we can find an assignment that *satisfies* all the clauses of the formula F. If a Clause gadget is not reached by the above procedure, the corresponding variables can be set arbitrarily.

($\Leftarrow$) In a similar manner if there is a truth assignment that satisfy the formula F, we can set the output interface cell of Variable gadget X_i in this way: if $x_i = true$ then $T_i = 1, F_i = 0$ else $T_i = 0, F_i = 1$, and there is a valid configuration for their inner empty cells. The output cells of the Split gadgets can be set according to the value of their input cells (if $I1 = 1$ then $O1 = O2 = 1$ else $O1 = O2 = 0$, and this guarantees that there is a valid configuration for their inner empty cells. The same can be applied to Edge gadgets. Finally every clause in the formula is made true by the assignment, and this guarantees that at least one of the input cells of each Clause gadget is set to 1 and all of them have a valid configuration.

□

5.2 `BINARY PUZZLE` is NP-complete

Theorem 5.2. *The decision problem* `BINARY PUZZLE` *is* NP*–hard.*

Proof. We must prove that adding the rules:

R2. each row and each column should contain an equal number of zeros and ones;

R3. each row is unique and each column is unique.

don't change the difficulty of the problem (in particular don't make it easier).

We can use a direct reduction from `BARE BINARY PUZZLE` using two tricks that transform a `BARE BINARY PUZZLE` instance B into a `BINARY PUZZLE` instance P.

R2. Equal number of zeros and ones in each row and column.

If B is the given `BARE BINARY PUZZLE` $n \times n$ starting grid, let $\overline{B}$ be the same grid with the fixed 0s and 1s inverted. We can build a new $2(n+2) \times 2(n+2)$ grid P placing a copy of B on the top–left (B_1) and bottom–right (B_4) quadrants of P, and a copy of $\overline{B}$ on the top–right ($\overline{B}_2$) and bottom–left ($\overline{B}_3$) quadrants of P. Between the four grids there is a *frame* of empty cells of width 4; we call F_1, F_2, F_C, F_3, F_4 the five parts of the frame (see Figure 10).

B_1	F_1	$\overline{B}_2$
F_2	F_C	F_3
B_3	F_4	$\overline{B}_4$

Figure 10: The grid P obtained by cloning the original starting grid B.

It is easy to see that if P has a valid solution, then B has a valid solution: just keep the top–left $n \times n$ portion (B_1) of the solution of P.

Vice versa if B has a valid solution S, then we can build a valid solution for P that doesn't violate rule $R2$ in the following way:

- put the values of S on top–left, bottom–right quadrants (S_1, S_4);
- put the inverted values of S on top–right, bottom–left quadrants ($\bar{S}_2, \bar{S}_3$),
- extend S_1 to the right (over frame F_1) placing an inverted copy and a copy of its last column $(x_1, ..., x_n)$:

x_1	$\bar{x}_1$	x_1	$\perp$	$\perp$	y_1
x_2	$\bar{x}_2$	x_2	$\perp$	$\perp$	y_2
...	...	...	$\perp$	$\perp$	...
S_1	...	F_1	$\perp$	$\perp$	S_2
...	...	...	$\perp$	$\perp$	...
x_n	$\bar{x}_n$	x_n	$\perp$	$\perp$	y_n

- extend S_2 to the left (over frame F_1) placing an inverted copy and a copy of its first column $(y_1, ..., y_n)$:

x_1	$\bar{x}_1$	x_1	y_1	$\bar{y}_1$	y_1
x_2	$\bar{x}_2$	x_2	y_2	$\bar{y}_2$	y_2
...	...	...	...	...	...
S_1	...	F_1	...	...	S_2
...	...	...	...	...	...
x_n	$\bar{x}_n$	x_n	y_n	$\bar{y}_n$	y_n

- fill frames F_2, F_3, F_4 in a similar manner;
- fill the central 4×4 frame F_C inverting its borders (see Figure 11).

The resulting grid is a valid solution for P because it generates no more than two consecutive numbers next to or below each other; and by construction it contains an equal number of 0s and 1s in each row and column. See figure Figure 12 for a full example of the construction.

x	$\bar{x}$	x	y	$\bar{y}$	y
$\bar{x}$	x	$\bar{x}$	$\bar{y}$	y	$\bar{y}$
x	$\bar{x}$	x	y	$\bar{y}$	y
z	$\bar{z}$	z	w	$\bar{w}$	w
$\bar{z}$	z	$\bar{z}$	$\bar{w}$	w	$\bar{y}$
z	$\bar{z}$	z	w	$\bar{w}$	w

Figure 11: Filling the central frame F_C: the configuration is valid for all values of $x, y, z, w \in \{0, 1\}$.

R3. Uniqueness of each row and column.

In order to complete the reduction from `BARE BINARY PUZZLE` to `BINARY PUZZLE` we trasform P to an equivalent grid P' with distinct rows and columns. In order to do this we first extend the $2(n+2) \times 2(n+2)$ grid P to its right and to its bottom adding two more empty columns and two more empty rows. Then we add to its right and to its bottom $\lceil \log \frac{n+3}{2} \rceil$ unique sequences of *binary strings*, where each digit is represented with two fixed 2×2 *digit blocks* (see Figure 13 and Figure 14). Each 2×2 block contain exactly two $1s$ and two $0s$, so they keep an equal number of 0s and 1s (don't break rule $R2$).

Again, if P' has a solution, then it can be easily transformed to a valid solution for the original `BARE BINARY PUZZLE` game B (just keep the upper–left quadrant).

Vice versa if B has a solution, it can be transformed to a valid solution S' for P': use the same construction used for P and extend it to the new two columns and the new two rows with the same technique used for extending S_1: make a double inversion of the last column and the last row. This guarantees that there are no conflicts with the adjacent numbers in the fixed digits blocks (no more than two consecutive 0s or 1s). By construction the two new columns and rows have an equal number of $0s$ and $1s$ and the same applies to the sequences of fixed digits; so S' satisfies rule $R2$. Furthermore the sequences of fixed digits make each row and column unique and thus S' satisfies rule $R3$, too. So S' is a valid solution for the `BINARY PUZZLE` problem P' (see figure Figure 12 for a full example of the construction of the solution of P').

□

Corollary 5.3. *The decision problem `BINARY PUZZLE` is* NP-complete.

Proof. The input of the problem is a string of length n^2; a solution of the game can also be represented with a string of the same length and its validity can be checked in linear time $O(n^2)$, so `BINARY PUZZLE` is contained in NP and NP–hard, thus it is NP–complete . □

6 Conclusion

Is this proof correct? .... 01011001001100110101001 1 !?!

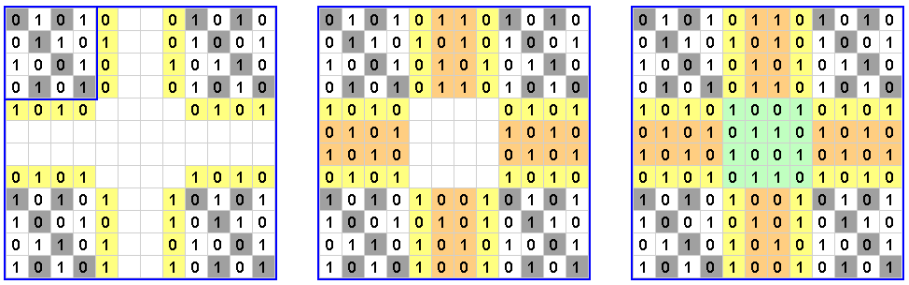


Figure 12: The construction of a Binary 12×12 puzzle board solution with equal number of 0s and 1s starting from a 4×4 Bare Binary puzzle solution (upper–left box).

0	1
1	0

1	0
0	1

Figure 13: 2×2 fixed blocks representing digit 0 and digit 1.

	$\perp$	$\perp$	D0 ... D0 D0
	$\perp$	$\perp$	D0 ... D0 D1
P	$\perp$	$\perp$	D0 ... D1 D0
	...	...	...
$\perp$ ... $\perp$	$\perp$	$\perp$	...
$\perp$ ... $\perp$	$\perp$	$\perp$	D1 ... D1 D1
D0 D1 ...	...	D1	D1 D0 ... D0
D0 D0 ...	...	D1	D0 D1 ... D0
... ...	...	...	...
D0 D0 ...	...	D1	D0 D0 ... D1

Figure 14: The grid P' built from P adding fixed blocks ($D0$, $D1$) to make each row and column unique.

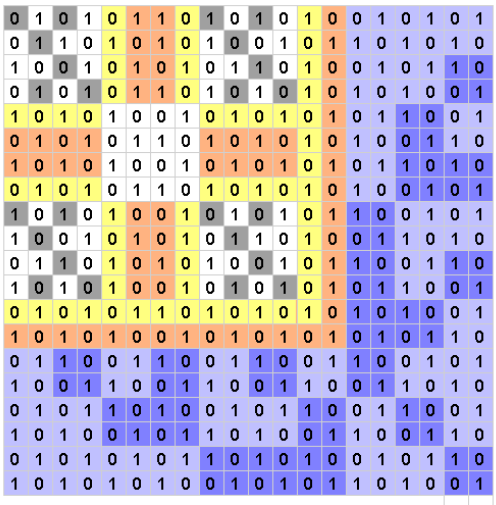


Figure 15: The final expansion of the solution of a `BARE BINARY PUZZLE` problem with a sequence of 2×2 fixed "digits" that make each column and row unique (rule $R3$) and thus make it a valid solution for the reduced `BINARY PUZZLE` problem, too.

References

[1] Erik D. Demaine and Robert A. Hearn. Playing games with algorithms: Algorithmic combinatorial game theory. In Michael H. Albert and Richard J. Nowakowski, editors, *Games of No Chance 3*, volume 56 of *Mathematical Sciences Research Institute Publications*, pages 3–56. Cambridge University Press, 2009.

[2] Graham Kendall, Andrew J. Parkes, and Kristian Spoerer. A survey of np-complete puzzles. *ICGA Journal*, 31(1):13–34, 2008.

[3] David Lichtenstein. Planar Formulae and Their Uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[4] Md. Saidur Rahman. Efficient algorithms for drawing planar graphs, 1999.