# The Complexity of Camping

Marzio De Biasi

marziodebiasi [at] gmail [dot] com

July 2012

Version 0.04: a re-revised sketch of the proof

**Abstract**

We prove that the `tents` puzzle game is $NP$-complete using a reduction from planar 3SAT.

## 1 Introduction

`Tents` is an addictive paper game that can be found in many puzzle magazines and is also available on several online games sites [6].

You have a grid of squares, some of which contain trees. Your aim is to place tents in some of the remaining squares, in such a way that the following conditions are met:

- There are exactly as many tents as trees.
- The tents and trees can be matched up in such a way that each tent is directly adjacent (horizontally or vertically, but not diagonally) to its own tree. However, a tent may be adjacent to other trees as well as its own.
- No two tents are adjacent horizontally, vertically or diagonally.
- The number of tents in each row, and in each column, matches the numbers given round the sides of the grid (*hints*).

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

In line with the recent interest in the complexity of puzzle games [4] [1], we study how hard it can be to solve a `tents` game.

In Section 2 we formally define the decision problem associated to the puzzle game; in Section 3 we briefly describe the planar 3SAT $NP$-complete problem; in Section 4 we describe the gadgets that can be used to reduce a planar 3SAT problem to a `tent` game and we prove that it is $NP$-complete.

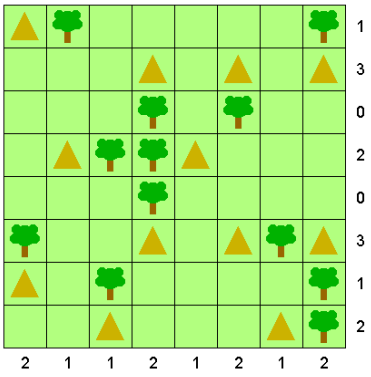



Figure 1: A solved `tents` game.

## 2 Definitions

To underline the role of the *hints* – i.e. the number of tents that must be placed on each row and each column — we define two slightly different decision problems:

**Definition 2.1.** decision problem `TENTS WITHOUT HINTS` :

**Instance:** A `tents` game, i.e.: an $n \times n$ grid with $m$ trees placed on it at coordinates $(x_1, y_1), ..., (x_m, y_m)$.
**Question:** Does a valid solution for the game exist? I.e. can we place $m$ tents in such a way that each tent is associated with exactly one tree and is adjacent to it (horizontally or vertically but not diagonally) and no two tents are adjacent horizontally, vertically or diagonally?

A solution of the game is simply a list of $m$ integers : $(d_1, d_2, ..., d_m)$, $d_i \in \{1, 2, 3, 4\}$; where $d_i = 1$ (respectively $2, 3, 4$) if the tent associated with the $i$-th tree is placed on its `left` (respectively `right`, `top`, `bottom`) side.

**Definition 2.2.** decision problem `TENTS` (with hints) :

**Instance:** A `tents` game, i.e.: an $n \times n$ grid with $m$ trees placed on it at coordinates $(x_1, y_1), ..., (x_m, y_m)$, the number of tents $r_1, r_2, ..., r_m$ that must be placed on each row and the number of tents $c_1, c_2, ..., c_m$ that must be placed on each column.
**Question:** Does a valid solution for the game exist? I.e. can we place $m$ tents in such a way that each tent is associated with exactly one tree and is adjacent to it (horizontally or vertically but not diagonally); no two tents are adjacent horizontally, vertically or diagonally; for each $i$ the number of tents in row $i$ is equal to $r_i$ and for each $j$ the number of tents in column $j$ is equal to $c_j$?

## 2.1 Elements of the game

In Figure 2 we show the elements of the game and some graphic notations that will be used in the following of this paper:

**empty cell** : an empty cell of the grid;

**tree** : a cell with a tree;

**tent** : a cell with a tent;

**tree+tent** : a tree and its associated tent (connected with a segment);

**red cross** : in a particular game configuration, red crosses will denote cells that cannot contain a tent without violating the rules of the game;

**highlighted cells** : the highlighted cells or *I/O cells* are normal cells, but they will behave like the input/output signals of a logic gate.

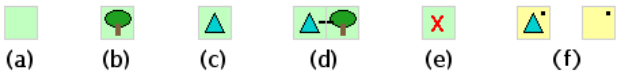



Figure 2: Elements of the game and notation: a) empty cell; b) tree; c) tent; d) a tree and its associated tent; e) a cell that cannot contain a tent; f) highlighted cells (*I/O cells*).

We call *a tree configuration* a particular portion of the grid of a `tent` puzzle that has only one or a restricted number of valid ways to assign the tents to the trees. Figure 3 shows several types of tree configurations:

- A fixed configuration (a1): there is only one valid way to place the tents near the trees; the cells marked with a red cross cannot contain a tent. The configuration can be extended horizontally (a2).
- A configuration in which the middle tree has only two possible valid tent assignments: a tent in cell $L$ (b1) or a tent in cell $R$ (b2).
- In order to free cell $L$ (c1), the tents must be "shifted" to the right, but the cell $R$ marked with the red cross must be free (c2).

# 3 Planar 3-SAT

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

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

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

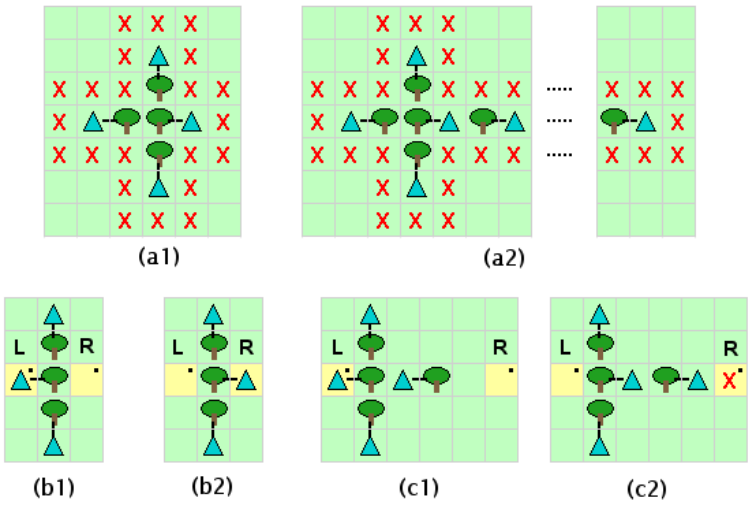



Figure 3: Several types of tree configurations.

is shown in Figure 4.

The problem of deciding whether a planar CNF formula is satisfiable or not is $NP$-complete [5] and it remains $NP$-complete even with the additional constraint that each clause must contain exactly three literals (*planar 3SAT*).

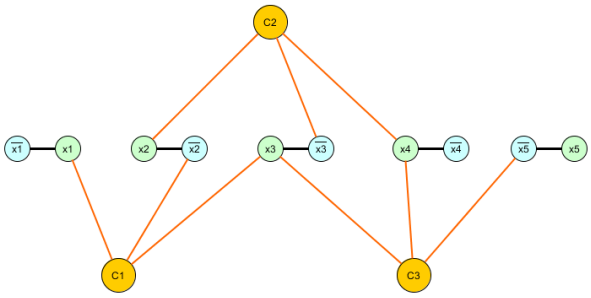



Figure 4: A planar 3CNF formula $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (x_3 \vee x_4 \vee \bar{x}_5)$.

# 4 Complexity

The solutions of both problems can be checked in polynomial time:

**Theorem 4.1.** *TENTS WITHOUT HINTS and TENTS are in NP*

*Proof.* Given a game solution, its validity can be easily checked in polynomial time: check that the solution has exactly $m$ elements, check that every tent

is in a valid position i.e. on an empty space, check that tents do not overlap and that they are not adjacent horizontally, vertically or diagonally. If $n$ is the size of the grid, the steps above can be done in $O(n^2)$. Finally - in the `TENTS` problem - check that the number of tents on each row and each column matches the corresponding hint ($O(n^2)$). □

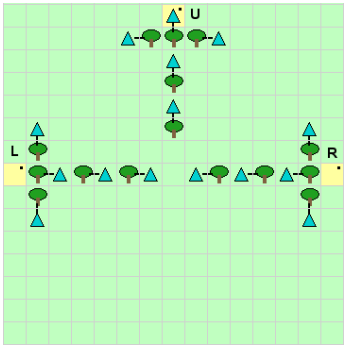
Now, we examine some tree configurations (we will call them *gadgets*) and the valid ways in which we can place the tents around them, but without taking into account the hints.

## 4.1 Gadgets and links

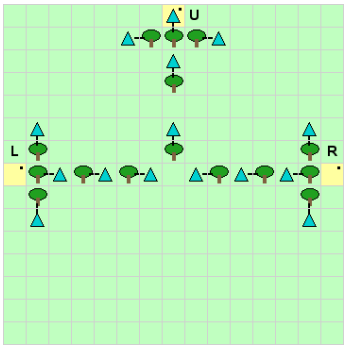
Using the basic configurations we are able to build the following four gadgets:

- The *AND* gadget is shown in Figure 5a: in order to free the topmost highlighted cell $U$, we need to be able to shift a tent to both the bottom-left highlighted cell $L$ *and* the bottom-right highlighted cell $R$.
- The *OR* gadget is shown in Figure 5b: in order to free the topmost highlighted cell $U$, we need to be able to shift a tent to the bottom-left highlighted cell $L$ *or* to the bottom-right highlighted cell $R$ (or both).
- The *CHOICE* gadget is shown in Figure 6a: at most one of the two highlighted cells $L$ or $R$ can be empty.
- The *TURN* gadget is shown in Figure 6b: same as CHOICE, but it makes a 90 degree turn.

In order to better understand the behaviour of the gadgets, the four figures also display a valid tent assignment, but each gadget should be considered as a tree configuration without any tent. All gadgets are built using a $15 \times 15$ subgrid and can be rotated 0, 90, 180 or 270 degrees.

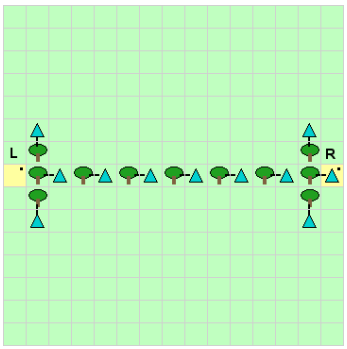



(a) AND gadget.

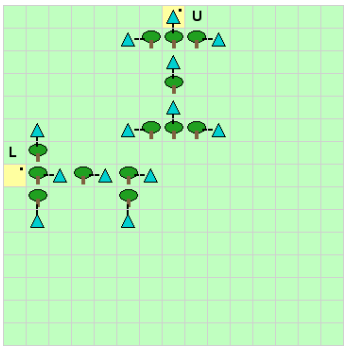



(b) OR gadget.

Figure 5: AND and OR gadgets




(a) CHOICE gadget.




(b) TURN gadget.

Figure 6: CHOICE and TURN gadgets

We will also used a FIXED gadget that is a simple four tree configuration that forces a tent in its I/O cell (see Figure 7).

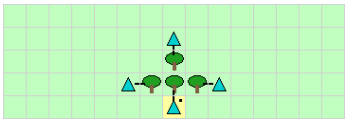


Figure 7: The FIXED gadget that forces a tent in its I/O cell.

The gadgets has also been checked with a constraint solver program to verify the valid configurations for the gadgets; Table 1 summarizes the results and Figure 8 show a schematization of the gadgets and their possible valid I/O cells configurations.

Combining the gadgets (without the tents) horizontally or vertically we can build a larger `tent` game. When two gadgets are adjacent two of their I/O cells can be adjacent, in this case both cannot contain a tent. We can interpret the I/O cells like a digital signal 1 or 0: when two I/O cells are adjacent the signal is *negated* (NOT gate) and propagated from one gadget to another. The I/O cells of the gadgets can also be connected using CHOICE gadgets (that can behave like straight links) and TURN gadgets.

## 4.2 Reduction from a planar 3CNF graph

Given a planar 3CNF graph $G$ we can build a corresponding `tent` game $T$ in the following way:

- For each pair of literals $(x_i, \bar{x}_i)$ we add a CHOICE gadget. Only one of its two I/O cells can be empty: $L = 0$ will represent a *true* assignment to the variable $x_i$, $R = 0$ will represent a *false* assignment to the variable $x_i$. We call $X_i$ and $\bar{X}_i$ the two I/O cells. If the literals have more than one

| Gadget | U | L | R | Valid |
|---|---|---|---|---|
| AND | 0 | 0 | 0 | NO |
| | 0 | 0 | 1 | NO |
| | 0 | 1 | 0 | NO |
| | 0 | 1 | 1 | YES |
| | 1 | 0 | 0 | YES |
| | 1 | 1 | 0 | YES |
| | 1 | 1 | 1 | YES |
| | 1 | 1 | 1 | YES |
| OR | 0 | 0 | 0 | NO |
| | 0 | 0 | 1 | YES |
| | 0 | 1 | 0 | YES |
| | 0 | 1 | 1 | YES |
| | 1 | 0 | 0 | YES |
| | 1 | 1 | 0 | YES |
| | 1 | 1 | 1 | YES |
| | 1 | 1 | 1 | YES |
| CHOICE | | 0 | 0 | NO |
| | | 0 | 1 | YES |
| | | 1 | 0 | YES |
| | | 1 | 1 | YES |
| TURN | 0 | 0 | | NO |
| | 0 | 1 | | YES |
| | 1 | 0 | | YES |
| | 1 | 1 | | YES |

Table 1: Valid configurations for the gadgets. A one means that a tents is present in the cell, a zero means that the cell is empty.

incident edge we can use one or more AND gadgets to *split* the signal into $X_i^1, X_i^2, ...$ distinct I/O cells with the following property: if $X_i = 1$ then all the $X_i^s$ I/O cells of the AND gadgets must hold a tent (Figure 9).

- For each clause $C_j = (X_{j_1} \vee X_{j_2} \vee X_{j_3})$ of the 3CNF formula, we add two OR gadgets and link the two I/O cells $R_1$ and $U_2$ together. Using a FIXED gadget we force the topmost I/O cell $U_1 = 0$; in this way at least one of the bottom three I/O cells must hold a tent: $L_1 = 1$ or $L_2 = 1$ or $R_2 = 1$ (Figure 10). We call $O_{j_1}$, $O_{j_2}$, $O_{j_3}$ the three bottom cells that represent the three litearls of clause $C_j$.
- We link the $O_{j_k}$ to the corresponding I/O cell $X_i$ or $\bar{X}_i$ using links and turns.

A `tents` puzzle that corresponds to the planar 3CNF formula of Figure 4 is schematized in Figure 11.

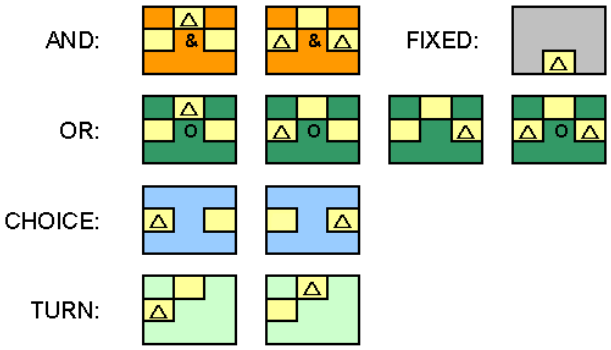



Figure 8: A schematization of the valid configurations of the I/O cells in the AND, OR, FIXED, CHOICE and TURN gadgets

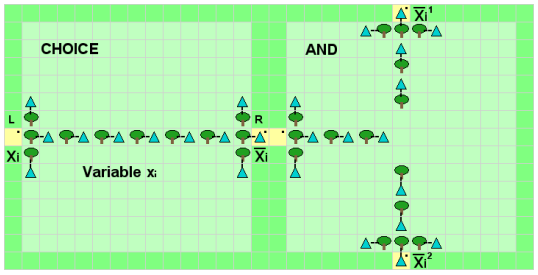



Figure 9: A CHOICE gadget that represent the truth assignment of a variable (empty $X_i$ represents $x_i$ = true); and an adjacent AND gadget that is used to split the I/O cell $\bar{X}_i$ corresponding to literal $\bar{x}_i$.

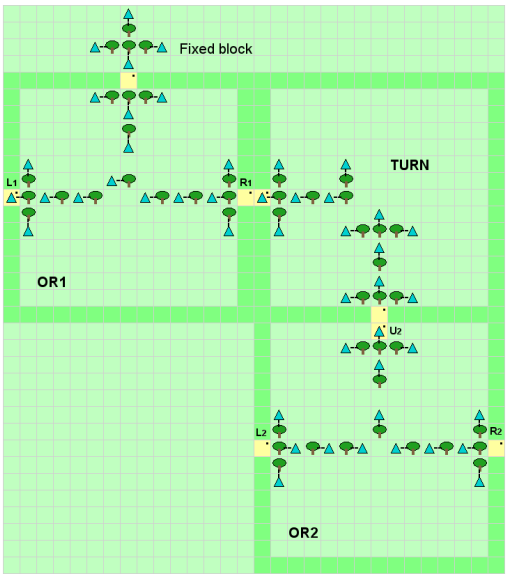



Figure 10: Two linked OR gadgets that represent a clause in the planar 3CNF formula. One of the three bottom I/O cells ($L_1$, $L_2$, $R_2$) must hold a tent.

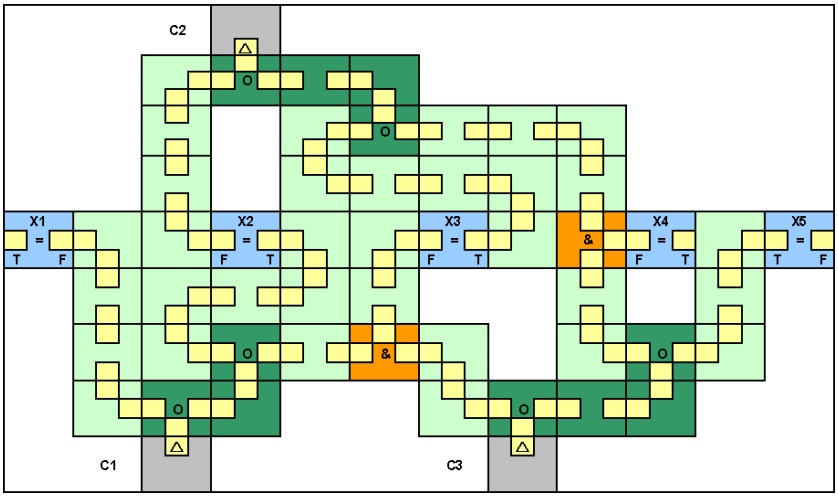



Figure 11: A schematized `tents` puzzle that corresponds to the planar 3CNF formula $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (x_3 \vee x_4 \vee \bar{x}_5)$ of Figure 4.

## 4.3 Tents without hints

**Theorem 4.2.** *TENTS WITHOUT HINTS is NP-hard*

*Proof.* Given an istance of planar 3SAT, we can build the planar graph $G$ of its 3CNF formula in polynomial time and then use the gadgets to build an equivalent TENTS WITHOUT HINTS game $T$.

If $(v_{1,2}, ..., v_n)$ is a satisfying assignment for $\phi$ then we can build a valid solution for $T$ in this way: if $v_i$ = true we leave empty the I/O cell $X_i$ of the CHOICE gadget and if the signal is splitted with some AND gadgets the I/O cells $X_i^1, X_i^2, ...$ can also be left empty. This allows us to make a valid tent assignment to all the OR gadgets that are linked to $X_i$. On the opposite side $\bar{X}_i$ must hold a tent and if it is splitted every $\bar{X}_i^1, \bar{X}_i^1, ...$ must also hold a tent. So an I/O cell of an OR gadget linked to $\bar{X}_i$ must be empty, but this is not a problem because the same OR gadget will have another I/O cell linked to the literal that makes the corresponding clause true and that can contain a tent. If $v_i$ = false the reason is similar. The literals cannot be true and false at the same time, so given a satisfying assignment for $\phi$ we can build a valid solution for $T$.

From a valid solution of $T$ we can build a satisfying assignment for $\phi$ in this way: at least one I/O cell of each OR gadget must hold a tent; if the cell is linked to $X_i$ then we set $x_i$ = true; otherwise if it is linked to $\bar{X}_i$ we set $x_i$ = false. If there are $x_i$ left out, we can assign them an arbitrary value. By construction if an I/O cell of an OR gadget is linked to $X_i$ then all I/O cells of the OR gadgets linked to $\bar{X}_i$ must be empty so there cannot be conflicts between the values assigend to the $x_i$. Thus $(v_1, v_2, ..., v_n)$ is a valid satisfying assignment for $\phi$.

Hence the planar 3SAT problem has a solution if and only if the corresponding tent game has a solution.

□

A solution of the tents puzzle that corresponds to a satisfying assignment of the planar 3CNF formula of Figure 4 is schematized in Figure 12.

## 4.4 Making dummy hints

If we add the hints to the formulation of the decision problem (see Definition 2.2 of the TENTS problem), we certainly don't get a harder problem; but, in order to keep the reduction from planar 3SAT valid, we must find a way to build hints without knowing the solution of the game and keeping the constraint that a valid solution for the game exists if and only if the original problem has a solution.

The idea is to extend the game and add *dummy trees configurations* on the borders. For each dummy trees configuration more than one valid tent assignment exists, and they are placed in correspondence of the tent assignments that can occurr in the gadgets in order to "mask" them.

For example suppose that in a gadget there is a tree at coordinates $(i, j)$, and its tent $t_k$ can be placed above or below it (Figure 13.a1). Then it is enough

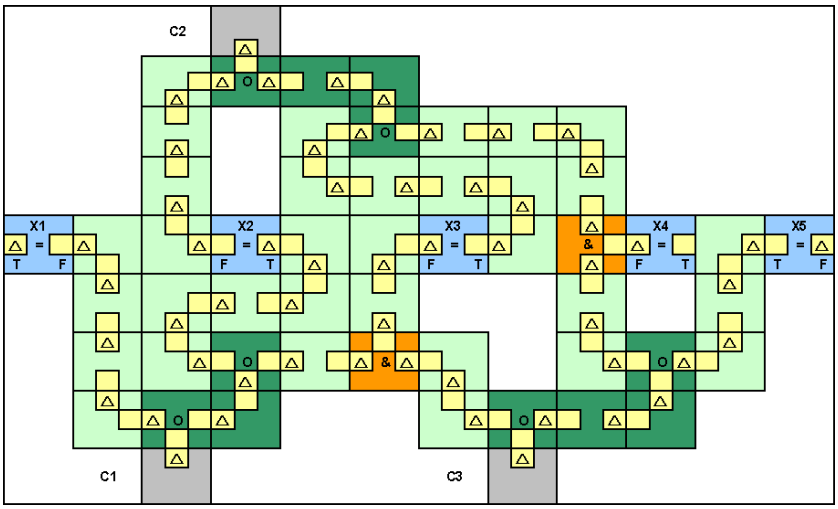



Figure 12: A schematized `tents` puzzle solution that corresponds to a satisfying assignment of the planar 3CNF formula $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (x_3 \vee x_4 \vee \bar{x}_5)$ of Figure 4: $x_1 = True; x_2 = True; x_3 = False; x_4 = False, x_5 = false$.

to add a new column $n+1$ to the game and place a single dummy tree on the same row $i$. Then we add $+1$ to the number (hint) $r_{i-1}$ of tents in row $i-1$; $+1$ to the number $r_{i+1}$ of tents in row $i+1$; and $+2$ to both $c_j$ and $c_{n+1}$ (the number of tents in columns $j$ and $n+1$). Whatever the final position of $t_k$ is, there is a valid tent assignment to the dummy trees configuration that keeps the hints constraints satisfied (Figure 13.a2-a3).

In a similar way, dummy trees configurations can be added for tents that can shift from the top of their tree to their left (Figure 13.b1-b3), but in this case we need to add two new columns and two new rows. And the same idea is also used for tents that can shift from the top of their tree to their left or right (Figure 13.c1-c4), but in this case we need to add two new columns and five new rows.

Note that in order to avoid conflicts among the tents of the dummy trees configurations we can simply add two extra empty columns and two extra empty rows after each dummy trees configuration.

**Theorem 4.3.** *TENTS (with hints) is NP-hard*

*Proof.* We can reduce in polynomial time a planar 3SAT problem to a `tents` game as seen above. In order to calculate the hints we first sum the unmoveable tents, then for each moveable tent we expand the grid, add the corresponding dummy trees configuration and update the values of the hints. At most $m$ expansions can occurr and each one of them add a fixed number of columns and trees, so the whole reduction can still be done in polynomial time. Like in the `TENTS WITHOUT HINTS` problem the game has a solution if and only if the

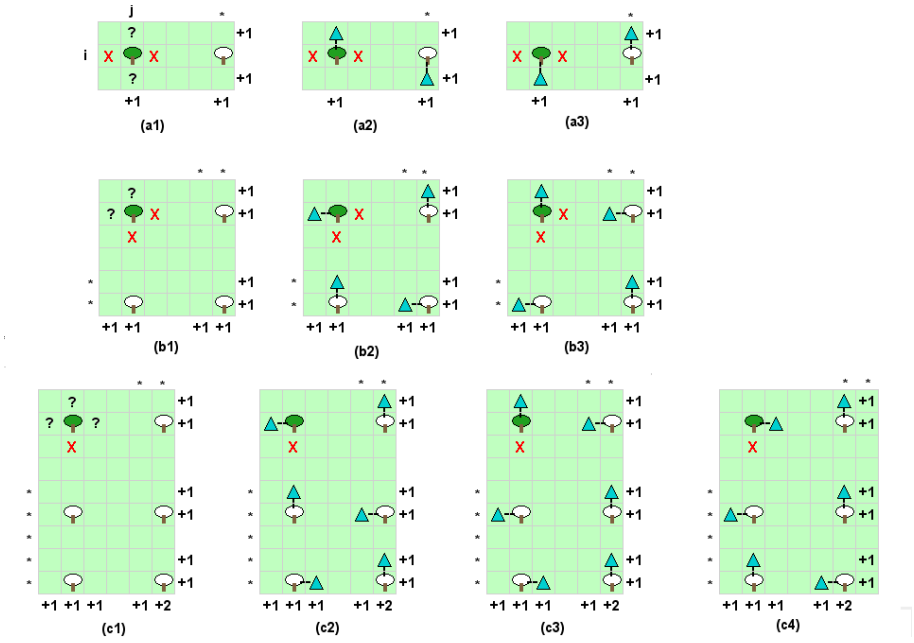



Figure 13: Dummy trees configurations (white trees) that keep the hints valid whatever tent assignment is made to the trees that belongs to the gadgets. The numbers represent the quantity that must be added to the hints. Asterisks represent the rows or columns that must be added.

original planar 3SAT problem has a solution, and the hints constraints are kept satisfied by construction. □

Finally, from Theorem 4.1 and Theorem 4.3 it follows that:

**Corollary 4.4.** *`TENTS` is NP-complete*

# 5 Conclusion

We proved that camping can be tricky!

It would be interesting to consider a classical $NP$-complete problem like the Hamiltonian circuit problem on grid graphs [3] and verify that its complexity doesn't change in the presence of *hints* similar to those found in the `tents` game, i.e. a number that is associated to every horizontal and vertical line that crosses the grid graph and that represents the number of edges of the Hamiltonian circuit that intersect that line.

# 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] Robert A. Hearn and Erik D. Demaine. *Games, puzzles and computation.* A K Peters, 2009.

[3] Alon Itai, Christos H Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

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

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

[6] Simon Tatham. Tents. `http://www.chiark.greenend.org.uk/~sgtatham/puzzles/java/tents.html`.