The complexity of the puzzle game Net: rotating wires can drive you crazy

Marzio De Biasi

marziodebiasi [at] gmail [dot] com

December 2012

Version 1.00: first version

Abstract

The puzzle game Net, also known as FreeNet or NetWalk, is played on a grid filled with terminals and wires; each tile of the grid can be rotated and the aim of the game is to connect all the terminals to the central power unit avoiding closed loops and open-ended wires. We prove that Net is NP–complete.

1 Introduction

Net is an addictive tile rotation game included in the Simon Tatham's Portable Puzzle Collection [8]; it was inspired by another flash game called *FreeNet* by Pavils Jurjans and there are other variants of it on the web (e.g. *NetWalk* [6], *Tubing*).

The game is played on a square grid: the computer prepares a network of terminals connected to a central power unit through wires, and then shuffles the network by rotating every tile randomly. Your job is to rotate it all back into place. The successful solution will be an entirely connected network, with no closed loops. As a visual aid, all tiles which are connected to the one in the middle are highlighted.

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

In line with some recent papers focused on the computational complexity of puzzle games [1] [5] we study the decision problem associated with the game: given an initial configuration of the game on a $n \times n$ grid, decide if it has a solution or not.

The game Net can be considered a variant of the *tile rotation problem* introduced and studied in [3] (*Tile Rotation Problem* and *Minimization Tile Rotation Problem*), with the additional constraint that the tiles cannot form closed loops.

In Section 2 we formally define the decision problem associated to the game; in Section 3 we describe the particular configurations of the game that we use in the reduction; in Section 4 we prove the NP–completeness using a reduction from the Hamiltonian cycle problem on grid graphs with degree ≤ 3.

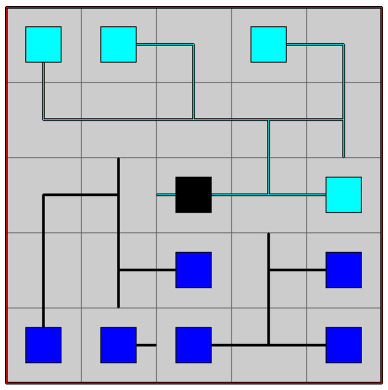

Figure 1: A 5×5 `Net` game: the terminals must be connected to the central power unit using the wires. Closed loops and open-ended wires are forbidden.

2 Definitions

The game `Net` is played on a $n \times n$ grid; each tile of the grid can be one of the following objects:

1. **Terminal**: it has a single connector on one of its sides;
2. **Line wire**: a straight line with two endpoints;
3. **Corner wire**: a line that makes a 90 degree turn;
4. **T wire**: a splitted line with three endpoints;
5. **Power unit**: has one, two or three connectors; in every game there is only one power unit.

The objects of the game are shown in Figure 2.

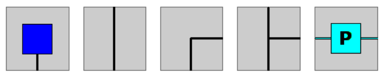

Figure 2: The objects of the puzzle game `Net`: terminal, line wire, corner wire, T wire and power unit.

A `Net` *initial configuration* (or simply a *game*), is a list of $n \times n$ integers $(t_{11}, t_{12}, ..., t_{nn})$ representing the tiles of the grid.

Every tile can be rotated 0, 90, 180 or 270 degrees; a *valid solution of the game* is a configuration reached from the initial configuration rotating the tiles and such that: every terminal is connected to the power unit, there are no closed loops, there are no wires with unconnected endpoints, and the power unit has no unused connectors. A solution can be represented using a list of $n \times n$ integers

$(r_{11}, r_{12}..., r_{nn})$ where each $r_{ij} \in \{0, 90, 180, 270\}$ represents the rotation of the i–th tile.

We can formalize the *decision problem* `Net`: given a list of $n \times n$ integers that represent the initial configuration of a `Net` game, does a valid solution exist?

3 Gadgets

For an introduction to the theory of NP–completeness see [2]; for an introduction to the study of computational complexity applied to puzzle games see [4].

To prove the NP–hardness of `Net` we will use some gadgets that are based on the configuration shown in Figure 3. There are two stacked blocks of wires, the upper block have three pairs of wires A, B and C, we call them the *interfaces* of the gadget. Using a constraint solver program, we examined the possible configurations in which there are no loops and all endpoints are connected except those of the three interfaces. If the three interfaces are all directed outward (Figure 3a), the lower block is isolated and forms a loop. If two interfaces are directed outward and one is directed inward, then the lower block can be correctly connected (Figure 3b,c,d). If only one interface is directed outward, then there is at least one inner loop (Figure 3e); if all the interfaces are directed inward there are at least two inner loops (Figure 3f). Furthermore no valid configuration exists if the two wires of one or more interfaces are directed in opposite directions. If the two wires of an interface are directed outward we say that the interface is *active*.

The two interfaces A and C can be rotated and the gadget above can be embedded in the 16×16 square grid shown in figure 4a; spaces are filled with terminals and wires. We will call this gadget *T–gadget*. If we remove one of the interfaces of the T–gadget and force the other two to point outward we obtain the *Line–gadget* (Figure 4b) and the *Corner–gadget* (Figure 4c). We can also can build a *Fill–gadget* that can be connected using only one wire and it can optionally be extended on the other three sides (Figure 4d): in order to connect a Fill-gadget to another gadget it suffices to replace one of the border terminals with a single line wire.

All the gadgets can be rotated 0, 90, 180 or 270 degrees, and they are surrounded by a *border* made with terminals. When two gadgets are placed side by side, some of the terminals on the border are *forced* to point inward in order to satisfy the rules of the game. These forced terminals "influence" the feasible rotations of the adjacent cells and a larger portion of the gadget becomes fixed. Some examples of forced configurations are shown in Figure 5: forced terminals and wires are shown in red, the asterisk marks the terminals and wires forced by the surrounding cells.

In particular the valid configurations of the T–gadegt correspond to those of the basic gadget: exactly two interfaces must point outward. A T–gadget surrounded by three T–gadgets (on top, left and right) and a Fill–gadget (on bottom) is shown in Figure 6; forced terminals/wires colored with red have been checked with a constraint solver program and the only free cells (colored

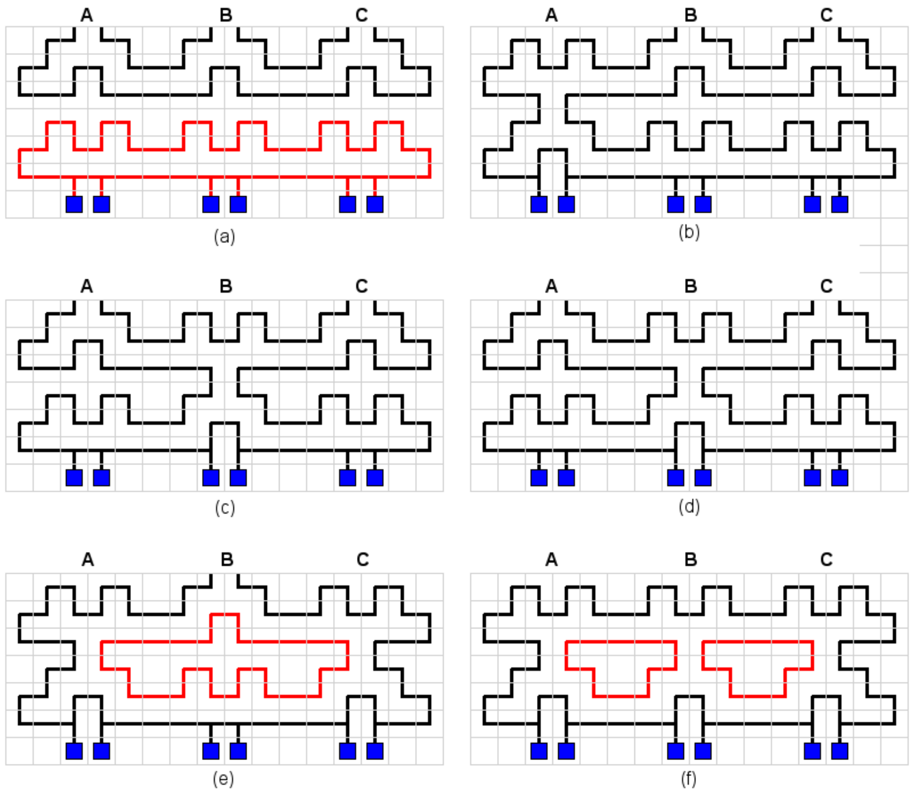

Figure 3: Basic gadget; in order to connect all wires without loops exactly two of the three *interfaces* A, B, C must be directed outward (figures (b), (c), (d)).

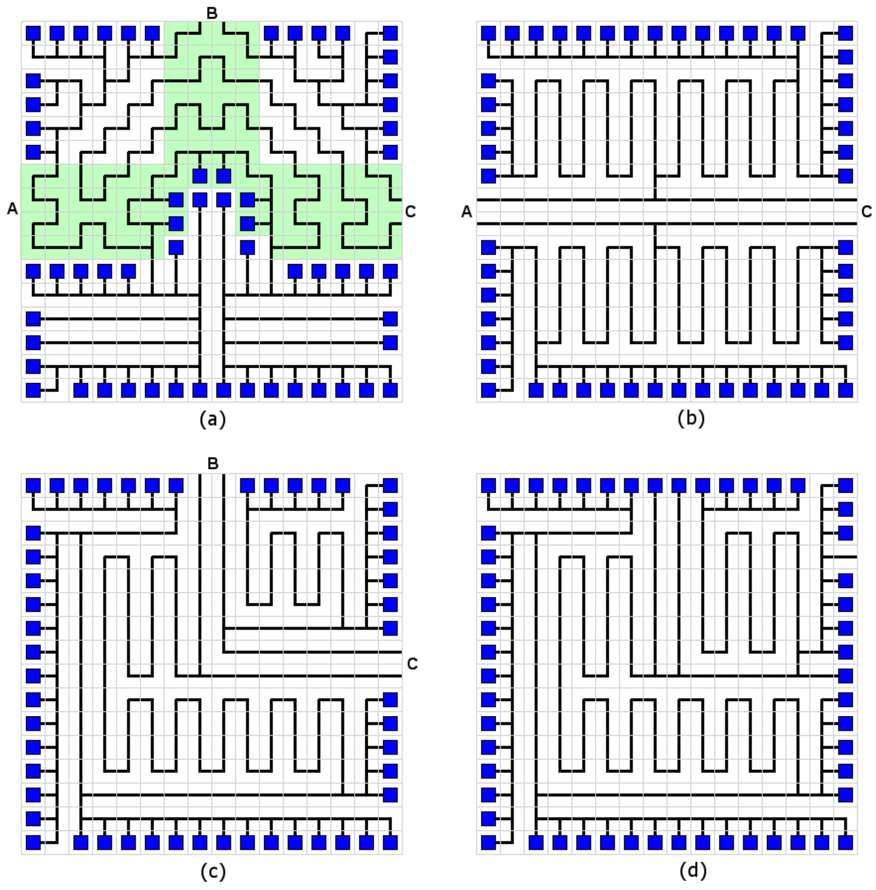

Figure 4: The basic gadget embedded in a 16 × 16 square grid. Spaces are filled with terminals and wires. The green area corresponds to the three original interfaces (a). Removing one interface we obtain the Line–gadget (b) and Corner–gadget (c). The Fill–gadget has only a single wire pointing outward (d).

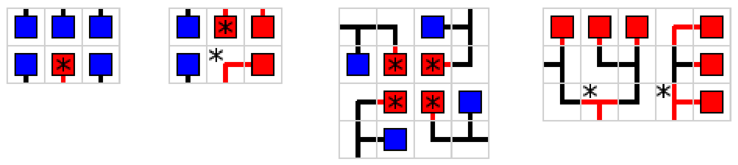

Figure 5: Examples of forced configurations: red terminals/wires are already forced, cells marked with an asterisk become forced by the surrounding cells.

with black) that can be rotated without violating the game rules are those corresponding to the three interfaces A, B, C.

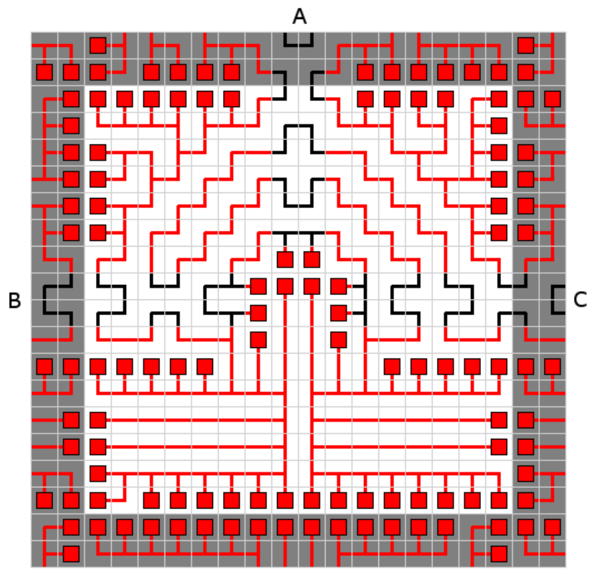

Figure 6: Forced terminals/wires of a T–gadget; the only free cells that can be rotated without violating the game rules are drawn in black and correspond to the three interfaces A,B,C.

The Line–gadget, Corner–gadget and Fill–gadget are entirely *forced.*

We notice that, in every valid configuration, the T/Corner/Line gadgets are splitted in two separated halves and in order to power them, both wires of the interfaces must be linked to the power unit.

4 Complexity of the puzzle game Net

Theorem 4.1. *Net is in* NP

Proof. A solution can be checked in polynomial time using a depth first search following the wires from the power unit. During the search we keep track of the terminals connected and at the end of each branch we must find a terminal, otherwise there is an open–ended wire or a loop. □

Theorem 4.2. *Net is* NP*–hard*

Proof. We use a reduction from the Hamiltonian cycle problem on grid graphs with degree ≤ 3 which is NP–complete [7]. Without loss of generality we can assume that the degree of each node of the grid graph is 2 or 3 (if a graph has a degree 1 node it cannot have an Hamiltonian cycle). Given a grid graph $G = (V, E)$ with $2 \leq deg(u) \leq 3$ for every $u \in V$, we can scale it by 16 and replace each degree 3 node with a T–gadget, and each degree 2 node with a Line–gadget or Corner gadget. The gadgets are rotated according to the edges of G: if (u_i, u_j) is an edge of G and g_i, g_j are the corresponding gadgets, then an interface of g_i is adjacent to an interface of g_j. Empty spaces of the grid are replaced with Free-gadgets connected to the adjacent gadegts.

Finally the power unit with two connectors is inserted in the center of one Corner–gadget (we will call it *Power–gadget*) that corresponds to a corner of the grid graph as shown if Figure 7: it "sends" the power outward using one of the two wires of the interface A, while the other wire of interface A is connected to a terminal (F in the figure); the two wires of the interface C are short–circuited (corner wires L in the figure).

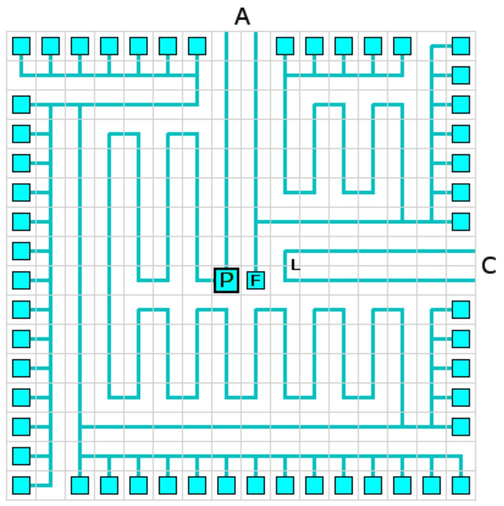

Figure 7: The power unit placed in one of the Corner–gadget (*Power gadget*). It sends the power outward using one of the two wires of the interface A, while the other wire of interface A is connected to a terminal (F); the two wires of the interface C are short–circuited (L).

The graph G has an Hamiltonian cycle if and only if the corresponding `Net` game has a solution.

$\Rightarrow$ Suppose that G has an Hamiltonian cycle $(u_1, u_2, ..., u_n, u_1)$, and without loss of generality assume that u_1 is the node corresponding to the Power–gadget. Then by construction we can rotate the wires of each T–gadget to follow the cycle; in particular if $..., u_{i-1}, u_i, u_{i+1}, ...$ is a portion of the cycle and g_{i-1}, g_i, g_{i+1} are the corresponding gadgets in the game:

- if u_i is a degree 2 node, then the two interfaces of g_i can be activated; and the power can be transferred from g_{i-1} to g_{i+1};

- if u_i is a degree 3 node, then its wires can be rotated in order to activate the interfaces adjacent to g_{i-1} and g_{i+1} (and the power can be transferred from g_{i-1} to g_{i+1}).

At the end of the process, one of the two halves of every T/Corner/Line gadget is directly linked to the power unit; but at the end of the Hamiltonian cycle the short–circuit L of the Power–gadget transfers the power to the other side and it can return back activating the unpowered half of each gadget. Finally the last terminal F is activated. Free–gadgets are powered from adjacent gadgets. No loops in the game are generated and all terminals are powered.

$\Leftarrow$ If the `Net` game corresponding to the graph G has a solution, then in every gadget that correspond to a node in G exactly two interfaces are active; but each active interface must be adjacent to another active interface of another gadget (otherwise there is an open–ended wire). So starting from the Power–gadget we can build a list of linked gadgets $g_1, g_2, ..., g_m$. Every gadget must be linked so m is greater or equal to the number of gadgets; and every gadget can appear only once in the sequence because exactly two interfaces are active (i.e. $m = n = |V|$). Moreover the only way to transfer the power from one half of each gadget to the other is through the loop L, so g_m must be connected to the C interface of the Power–gadget.

By construction, g_i, g_{i+1} are linked through their interfaces if and only if the corresponding nodes n_i, n_{i+1} belong to an edge of G ($(u_i, u_{i+1}) \in E$), so the sequence $u_1, u_2, ..., u_n, u_1$ is an Hamiltonian cycle in G.

$\square$

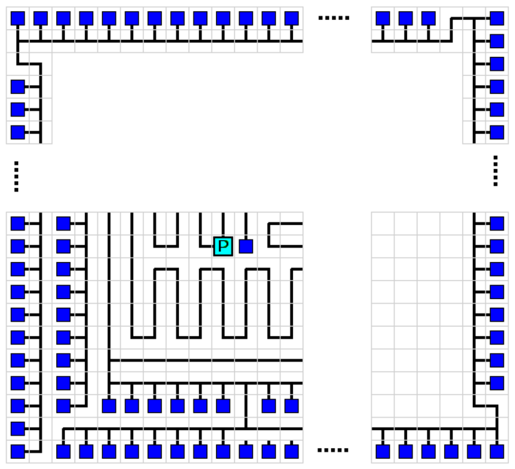

Figure 8: In the `Net wrapping` variant the wrapping wires can be forbidden adding an extra border of terminals.

There is also a variant of the game that allows the wires to wrap around the borders; we call this version `Net wrapping`.

Corollary 4.3. *`Net wrapping` is* NP*–complete*

Proof. We can reduce the game `Net` to a `Net wrapping` game adding an extra border of terminals that blocks the transfer of power from one border to another and therefore makes the `Net wrapping` game equivalent to a `Net` game (see Figure 8).

□

5 Conclusions

Happy wiring!

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] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, San Francisco, 1979.

[3] Eric Goles and Iván Rapaport. Complexity of tile rotation problems. *Theor. Comput. Sci.*, 188(1-2):129–159, November 1997.

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

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

[6] Ben Lynn. Netwalk. `http://code.google.com/p/netwalk/`.

[7] Christos H Papadimitriou and Umesh V Vazirani. On two geometric problems related to the travelling salesman problem. *Journal of Algorithms*, 5(2):231 – 246, 1984.

[8] Simon Tatham. Net. `http://www.chiark.greenend.org.uk/~sgtatham/puzzles/java/net.html`.