Universiteit Leiden
Opleiding Informatica

A Difficulty Measure
for Light Up Puzzles

Name: Victor Dekker
Date: 02/03/2015
1st supervisor: W.A. Kosters
2nd supervisor: H.J. Hoogeboom

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

A Difficulty Measure for Light Up Puzzles

Victor Dekker

March 25, 2015

Abstract

Light-Up puzzles are fun logic puzzles that consist of walls and empty space that needs to be lit by placing lights. In this paper, we propose a system of difficulty measures that can rank different Light-Up puzzles.

1 Introduction

The Light Up puzzle (also known as Akari) is a popular puzzle, played worldwide. This puzzle consists of a grid of cells, signifying walls and empty spaces. The goal of the puzzle is to place lights in order to light up all empty spaces. This kind of puzzle is proven to be $\mathcal{NP}$-complete [1]. A lot of different Light Up puzzles exist, usually with difficulty measures such as "Easy", "Medium" or "Hard". In this paper, however, we propose a system of difficulty measures that gives a certain level of a puzzle, as well as an algorithmically determined number to characterize the puzzles' difficulties (in practice, two 'Medium' puzzles can be very different in difficulty).

In Section 2 we will explain the rules to the Light Up puzzles. Section 3 shows several approaches to solve the puzzles. In Section 4 we will propose our system of difficulty measures, and in Section 5 we will use this system to design two asymptotically difficult puzzles.

This paper is a bachelor project at LIACS, the Computer Science department of Leiden University. It was supervised by W.A. Kosters and H.J. Hoogeboom.

1.1 Related Work

The Light-Up puzzle has been the subject of quite some papers. Many discuss how to solve these puzzles, for example by using methods similar to how a human player would solve the puzzles [2], [3].

More complicated algorithms of solving Light-Up puzzles are also addressed [4], [5], [6]: these all have their own approaches to solving puzzles, that do not necessarily model the method a human player would go about solving them.

In this paper, a human-like method of solving the puzzles is used; this allows for the difficulty measure to represent how difficult a Light-Up puzzle is to solve to a human player, without the use of algorithms that require much computational power or many instances of the same puzzle.

2 Rules of a Light Up puzzle

One of the appealing qualities of a Light Up puzzle is that it is very easy to learn the rules: they are very few, and they are very clear. Given a grid of cells (usually rectangular, although this would matter little), with some cells containing a specified value (black cells or numbered cells), one has to "light up" all the white spaces in the puzzle by placing an undefined number of lights in cells.

A light shines in four directions: left, right, up and down. This means that the lightbeams originating from the cell the light was placed in, travel left, right, up and down, until they encounter a wall, numbered wall or edge of the puzzle. All cells the light shines upon are considered lit.

Of course there are constraints that limit the player of placing lights just anywhere:

- Two lights cannot be placed in the same row or column, except when a wall or numbered wall is in between (two lights cannot shine on each other).
- A numbered wall must have an amount of lights directly neighboring it (left, right, above and below) equal to the number it contains.

Following these rules, the puzzle has to be solved; all non-wall cells must be lit; see Figure 1. In this and all following figures, walls are marked by

black cells. Numbered cells are also walls (or numbered walls). A '$' is used to mark lights. Cells that are lit by one or more lights will be colored yellow and contain a '.', and 'non-light' cells (that cannot contain lights, but are otherwise empty), are marked by a '-'.

Figure 1: A simple Light Up puzzle and its unique solution

2.1 Decent puzzles

In puzzles such as described, cells can form a certain pattern, in which one empty cell is surrounded on four sides by a wall. These cells must always be filled by a light (there is no other way to light up the cell). Puzzles without this pattern will be called *decent puzzles*, and we will confine this paper solely to these decent puzzles. Any puzzle with such a pattern in them can be made into a decent puzzle by changing the empty cell into a wall and, if any of the surrounding walls are numbered, decreasing the number in surrounding numbered walls by 1. See Figure 2 for an example.

Figure 2: To change a puzzle into a decent puzzle

3 Solving a puzzle

To solve a Light Up puzzle, multiple approaches can be taken. One of these approaches is to use the numbered walls to supply direction as to where the lights must go; see Figure 3.

Figure 3: The 2 in the right-lower corner indicates two lights are neighboring that cell

This allows us to define the NumberedWallApproach (NWA) as the following: given a numbered wall with number x and y neighboring empty cells (that might contain a light) and z neighboring cells that already contain a light. If $x-z=y$ then all neighboring empty cells must be filled with lights. If $x=z$ all neighboring empty cells cannot contain a light. See Figure 4.

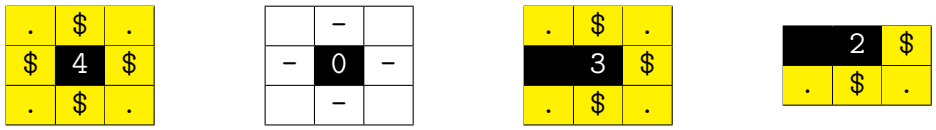

Figure 4: Simple deductions about numbered walls

Furthermore, adding to the NumberedWallApproach is a deduction about where to place non-light cells: if $x-z=y-1$, all cells that are diagonally next to the numbered wall and directly next to two empty neighbors of the numbered wall can be marked non-lights; see Figure 5.

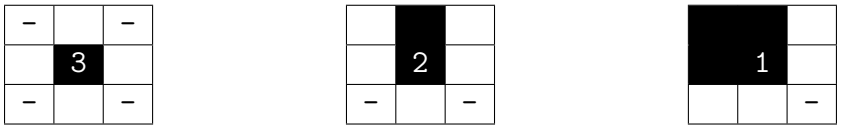

Figure 5: More deductions about numbered walls

Using the NumberedWallApproach, one can possibly place many lights. However, it is usually not enough to complete a whole puzzle: another approach is needed to fill in the remaining blanks (as one can see when solving the puzzle from Figure 1: the upper left cell cannot be lit by the NumberedWallApproach). The second approach used considers not the numbered walls, but the unlit cells.

The UnlitCellApproach (UCA) is defined as follows: for any given unlit cell A (A can be empty, but could also be a non-light cell), x is the

number of empty cells in the same row or column as A with no walls separating these cells from A (this is including cell A itself). If $x = 1$ then a light should be placed on the single empty cell in the same row or column as A with no walls separating this cell from A. See Figure 6, where the cell in the upper left corner is used as cell A.

Figure 6: Using the UnlitCellApproach

Combined, these two approaches can complete easy puzzles:

Figure 7: Solving a puzzle using both approaches

3.1 Patterns

The aforementioned approaches can solve quite some puzzles. Sometimes, however, more information is needed to place lights, or to prove that some cells do not contain lights. From the context of multiple cells (usually numbered walls), such proofs can be given. These specific arrangements of cells are called *patterns*. These patterns can be mirrored or rotated without losing the deductions of the value of cells that we make here.

To find the lights of these patterns, proof by contradiction is used; when ensuring no light is placed where a light should be, one encounters contradictions in the puzzles. Therefore, a light has to be placed in those cells. This more general "guess step" will be discussed in Section 3.2.

These patterns will be explained using puzzles, but they will also be put in mathematical terms, by reformulating them into constraints. In these constraints, we attach a Boolean variable to every non-wall cell. If this variable is true, a light needs to be placed in the corresponding cell; if it is false, no light can be placed in that cell. Any variable a that is in contact with another variable b, so that placing a light in cell a will directly light up cell b and the other way around, causes the constraints $a \Rightarrow \neg b$ and $b \Rightarrow \neg a$.

Function $L(x)$ returns the number of true values in x, where x is a set of Boolean variables. We will use conditions of the form $L(x) = k, L(x) \geq k, L(x) \leq k$ to indicate some property about the amount of lights in the set x. This function $L(x)$ can be seen as an abbreviation for a series of predicate logic expressions; for example, $L(a, b, c) \leq 2$ equals the expression $(((a \vee b) \wedge \neg c) \vee ((a \vee c) \wedge \neg b) \vee ((b \vee c) \wedge \neg a) \vee (\neg a \wedge \neg b \wedge \neg c))$. Although the function is interesting, and it is possible to extend it into some sort of "light-up logic", this is not necessary for the following proofs.

3.1.1 The square pattern

The first pattern we will discuss is the square pattern: this is a pattern that can extend over a variable distance. Its basic layout is shown in Figure 8.

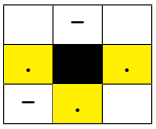

Figure 8: The square pattern

The solution to the pattern cannot be found by one of the previously mentioned approaches: the NumberedWallApproach is of no use since there are no numbered walls affecting this situation, and the UnlitCellApproach will not work either since every unlit cell has at least two possibilities of being lit. However, placing a light in the upper right corner will prevent the lower left corner from ever being lit.

Solving this pattern by placing a '–' at the upper right corner is a shortcut to the guess step (a somewhat more advanced move), and can save a lot of trouble at certain puzzles; see Figure 9.

Figure 9: Solution to the square pattern

Important to note is that the square pattern can range over a long distance; as long as there are four different *unlit* cells that together form a rectangle, with sides that do not contain a wall (and thus block light), and

cannot contain lights themselves. Furthermore, the four corner cells should not be able to be lit by any cell outside the rectangle. Figure 10 shows the general layout of the square pattern, in which '$*$' fields mark a field of cells whose values do not matter, the '!' fields mark cells that cannot contain a light, but may be lit, and the 'N' fields mark a cell that cannot contain a light or a wall, but may be lit.

$*$	!	$*$	$\cdots$	$*$	!	$*$
!		N	$\cdots$	N		!
$*$	N	$*$	$\cdots$	$*$	N	$*$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$
$*$	N	$*$	$\cdots$	$*$	N	$*$
!	-	N	$\cdots$	N		!
$*$	!	$*$	$\cdots$	$*$	!	$*$

Figure 10: Layout of the square pattern

To put it in mathematical terms, we use four different variables: a, b, c and d, see Figure 11.

(c)	-	(d)
.		.
$-(a)$	.	(b)

Figure 11: The square pattern

See Table 1 for the logical proof of this pattern.

3.1.2 Diagonal patterns

Two more patterns can be found when observing numbered walls that stand diagonally with respect to one another; they share two neighboring cells in which lights might be placed. If these two numbered walls are a `3` and a `1`, this allows for some deductions. Because of the `3`, at least one of cells c or d must be a light, while the `1` ensures at most one of cells c and d is a light. Therefore, cells a and b must be lights and cells e and f cannot be lights; see Figure 12.

Mathematical proof of this is given in Table 2.

1	$a \Rightarrow \neg b$	given
2	$a \Rightarrow \neg c$	given
3	$b \Rightarrow \neg d$	given
4	$c \Rightarrow \neg d$	given
5	$\neg a$	given
6	$L(a, b, c) \geq 1$	to light up a
7	$L(a, b, d) \geq 1$	to light up b
8	$L(a, c, d) \geq 1$	to light up c
9	$L(b, c, d) \geq 1$	to light up d
10	d	assumption
11	$\neg c$	3, 10
12	$\neg b$	4, 10
13	a	6, 11, 12
14	$a \wedge \neg a$	5, 13; a contradiction
15	$\neg d$	10, 14
16	b	5, 7, 15
17	c	5, 8, 15

Table 1: Proof of the Square Pattern

Figure 12: Diagonal 3 on 1 pattern

This pattern can be altered to create slightly different patterns by a two simple changes: first, one of the cells with a light is filled with a wall instead. Then, the numbers of all numbered walls neighboring this cell are decreased by 1. These changes do not affect the deductions of the pattern, and the patterns are still usable. The new pattern can also be changed using the same steps again; see Figure 13.

3.1.3 Neighboring pattern

Another pattern that gives us lights where our other approaches will not, is the *neighboring pattern*. In this pattern, two 2 cells are each other's direct

1	$L(a, b, c, d) = 3$	given
2	$L(c, d, e, f) = 1$	given
3	$\neg a$	assumption
4	$b \wedge c \wedge d$	1, 3
5	$L(c, d, e, f) \geq 2$	4
6	$1 \geq 2$	2, 5; a contradiction
7	a	3, 6
8	$\neg b$	assumption
9	$a \wedge c \wedge d$	1, 8
10	$L(c, d, e, f) \geq 2$	9
11	$1 \geq 2$	2, 10; a contradiction
12	b	8, 11
13	$e \vee f$	assumption
14	$\neg c \wedge \neg d$	2, 13
15	$L(a, b, c, d) \leq 2$	14
16	$3 \leq 2$	1, 15; a contradiction
17	$\neg(e \vee f)$	13, 16
18	$\neg e \wedge \neg f$	17
19	$a \wedge b \wedge \neg e \wedge \neg f$	7, 12, 18

Table 2: Proof of the Diagonal Pattern

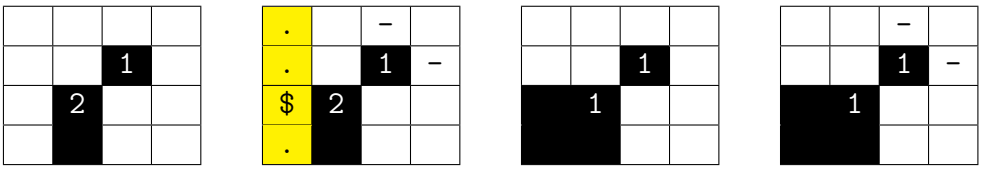

Figure 13: Diagonal 2 on 1 and diagonal 1 on 1 patterns

neighbor. Due to the fact that neither of the two cells can have two lights in the same column (above and below the numbered wall), both of the two numbered walls must have one light opposite to their neighbor; see Figure 14.

The neighboring pattern can, just like the square pattern, extend over a long distance of cells. To do this, both numbered walls need a neighboring wall, and they need to be able to shine their lights upon each other. Again, in Figure 15, the '$*$' fields mark the fields of cells whose values are unimportant for the pattern and can be anything. The '!' fields mark cells that cannot

Figure 14: Neighboring pattern

contain a light, but could be lit, and the 'N' fields mark cells that cannot contain a light or a wall, but could be lit.

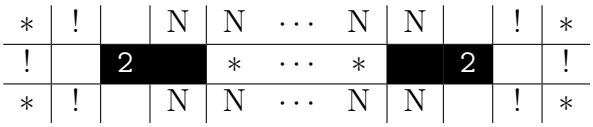

Figure 15: Layout of the neighboring pattern

Applying a formal proof requires six different variables; a, a', b, b', c and d; see Figure 14 and Table 3.

3.1.4 Linear patterns

Two more interesting patterns can be found when examining numbered walls on either the same horizontal or vertical line, with one empty cell in between them. These two patterns can then be altered into more patterns using the same two steps for changing patterns as explained in Section 3.1.2.

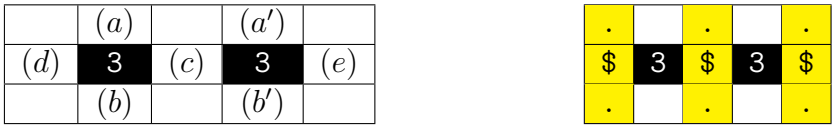

Figure 16: Linear 3 on 3 pattern

Figure 17: Linear 3 on 2 pattern

1	$a \Rightarrow \neg a'$	given
2	$b \Rightarrow \neg b'$	given
3	$L(a, b, c) = 2$	given
4	$L(a', b', d) = 2$	given
5	$\neg c$	assumption
6	$a \wedge b$	3, 5
7	$\neg a' \wedge \neg b'$	1, 2, 6
8	$L(a', b', d) \leq 1$	7
9	$2 \leq 1$	4, 8; a contradiction
10	c	5, 9
11	$\neg d$	assumption
12	$a' \wedge b'$	4, 11
13	$\neg a \wedge \neg b$	1, 2, 12
14	$L(a, b, c) \leq 1$	13
15	$2 \leq 1$	3, 14; a contradiction
16	d	11, 15
17	$c \wedge d$	10, 16

Table 3: Proof of the Neighboring Pattern

Figure 18: Linear 2 on 2 pattern

It should be noted that one of the alterations of the linear 3 on 3 pattern is similar to the neighboring pattern: when a wall is placed between the two 3's, and both numbers are decreased by 1, what remains are two 2's with a single wall in between. However, this does not mean the neighboring pattern is just an alteration of the linear 3 on 3 pattern; the neighboring pattern can still extend over an arbitrary number of cells, while the linear 3 on 3 pattern is limited to its current size.

For proof of the linear 3 on 3 pattern (see Figure 16 for the variables), see Table 4.

The second linear pattern is the linear 3 on 2+ pattern. This pattern is slightly different from the linear 3 on 3 pattern, but also useful.

1	$a \Rightarrow \neg a'$	given
2	$b \Rightarrow \neg b'$	given
3	$L(a, b, c, d) = 3$	given
4	$L(a', b', c, e) = 3$	given
5	$\neg c$	assumption
6	$a \wedge b \wedge d$	3, 5
7	$\neg a' \wedge \neg b'$	1, 2, 6
8	$L(a', b', c, e) \leq 1$	7
9	$3 \leq 1$	4, 8; a contradiction
10	c	5, 9
11	$\neg d$	assumption
12	$a \wedge b \wedge c$	3, 11
13	$\neg a' \wedge \neg b'$	1, 2, 12
14	$L(a', b', c, e) \leq 2$	13
15	$3 \leq 2$	3, 14; a contradiction
16	d	11, 15
17	$\neg e$	assumption
18	$a' \wedge b' \wedge c$	4, 17
19	$\neg a \wedge \neg b$	1, 2, 18
20	$L(a, b, c, d) \leq 2$	19
21	$3 \leq 2$	3, 14; a contradiction
22	e	17, 21
23	$c \wedge d \wedge e$	10, 16, 22

Table 4: Proof of the Linear Pattern

Figure 19: Linear 3 on 2+ pattern

Figure 20: Linear 2 on 2+ pattern

Figure 21: Alteration of the linear 3 on 2 pattern: linear 3 on 1

Figure 22: Alteration of the linear 3 on 2 pattern: linear 2 on 1

3.1.5 Pattern overview

Using these patterns and their variants, one can find lights where they might be less obvious. Of course, more patterns exist; all possible configurations of walls, numbered walls and lights could be called patterns. These patterns, however, are rather small in size and they occur frequently in real world puzzles. Their "input" and "output" are noted in Table 1, where the input are the minimal amount of predetermined cells needed to recognize the pattern.

Name	Input (minimal)	Output (lights)	Output (non-lights)
Square pattern	3	2	0
Diagonal 3 on 1 pattern	2	2	2
Diagonal 2 on 1 pattern	3	1	2
Neighboring pattern	2	2	0
Linear 3 on 3 pattern	2	1	0
Linear 3 on 2 pattern	3	2	0
Linear 2 on 2 pattern	4	1	0
Linear 3 on 2+ pattern	2	1	2
Linear 2 on 2+ pattern	3	1	0
Linear 3 on 1+ pattern	3	1	4
Linear 2 on 1+ pattern	4	1	2

Table 5: Pattern overview

3.2 The guess step

In certain puzzles, it may occur that all aforementioned approaches and patterns are still not enough to light up everything. Many hard puzzles exist in which no real logic deduction trick other than proof by contradiction can be applied. This can also be called a "guess": one guesses that a certain cell contains a light, then continues deducing more lights until one encounters a contradiction. This means the guessed light is *wrong*, and thus the cell ought to contain a non-light.

An example can be seen in Figure 23.

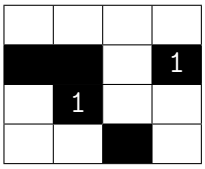

Figure 23: Hard puzzle

This puzzle is unsolvable using the aforementioned steps, except when the guess step is used. In fact, all other approaches cannot even deduce one single cell value of this puzzle. However, the guess step can be used; see Figure 24.

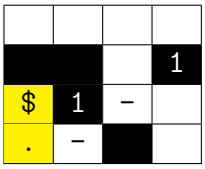

Figure 24: Guess step used

A light is placed by guess, and this immediately leads to a contradiction (the numbered wall is satisfied, and one of the cells cannot be lit). Therefore, the guess is wrong, and that particular cell should not contain a light. From there, further deductions complete the puzzle; see Figure 25.

4 Defining a difficulty measure

To define a difficulty measure, it should be noted that there are different kinds of puzzles, for which some approaches to solving them are not sufficient. For example, using only the NumberedWallApproach and the

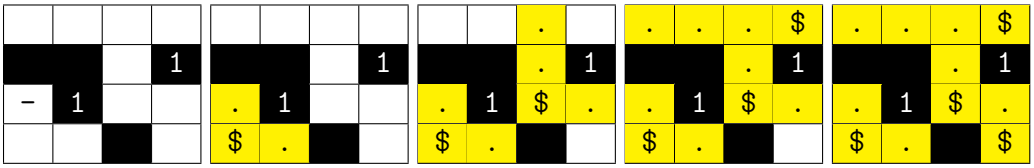

Figure 25: Solving the hard puzzle

UnlitCellApproach, one cannot solve truly hard puzzles which require at least one guess step. Therefore, we propose not one, but *several* difficulty measures for Light Up puzzles.

4.1 System of difficulty measures

The system of difficulty measures consists of three ratings: "easy", "medium" and "difficult". These ratings determine the kind of difficulty measure used for the puzzle.

Easy puzzles can be solved using only the NumberedWallApproach and the UnlitCellApproach. Medium puzzles cannot be solved using only NWA and UCA, but need to use patterns as well to solve them. Finally, difficult puzzles cannot be solved using NWA, UCA and patterns, but can be solved using guesses too. See Figure 26.

Figure 26: Overview of difficulty measures

The method of finding a difficulty measure to a given puzzle is as follows: first, one tries to apply an easy difficulty measure; the 'step' or 'switch' difficulty will be calculated.

When calcultating the *step* difficulty of a puzzle, the difficulty counter starts at 0. The puzzle is solved using only NWA and UCA steps; every time

a numbered wall or unlit cell is observed and one or more cells are given values because of this observation, this is called a step and the difficulty counter is increased by 1. When the puzzle is solved, the counter shows the step difficulty. However, it is important to note that the *minimal difficulty* is of importance. Since the sequence in which certain steps are done can result in different difficulty ratings, the lowest possible step difficulty for a puzzle (the minimal amount of steps needed) is what is used as a difficulty measure.

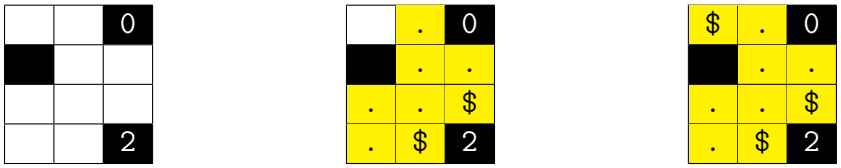

Figure 27: Solving this puzzle takes 2 steps: first NWA, then UCA

The *switch* difficulty of a puzzle is somewhat different: again, the difficulty counter starts at 0, but instead of increasing the difficulty counter for every single use of NWA or UCA, the difficulty counter is increased for every "switch" between approaches. This means that, at the start of the calculation, one of the two approaches is selected as the current approach. This approach can be used any number of times, but if this yields no more results, the current approach switches to the other approach, and the difficulty counter is increased by 1. Because of this different way of calculating the difficulty, it is considerably faster than the step difficulty.

To calculate the difficulty using switches, two calculations are needed: one where the current approach starts with NWA, and one where it starts with UCA. The average of both approaches is the final result of the switch difficulty. The difference between results of both calculations is either 0 or 1.

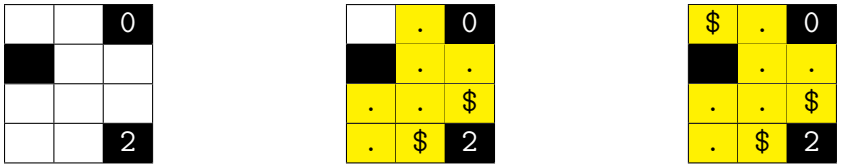

Figure 28: Solving this puzzle takes 1 switch: from NWA to UCA

If the puzzle is not easy (that is: if repeated NWA and UCA steps cannot solve the puzzle), one tries to apply a medium difficulty measure, including patterns in their calculations. Doing this means the puzzle is solved using NWA and UCA steps for free (they do not increase the difficulty of the puzzle), but when stuck, patterns are used. Every use of a pattern increases the difficulty by 1.

Just like the step difficulty, the minimal pattern difficulty is needed: the minimal amount of patterns used to solve the puzzle.

However, if using NWA and UCA steps, as well as patterns, is not enough to solve the puzzle, the final difficulty measure will be applied: the guess difficulty. Using guesses, one can solve every single puzzle, by using brute force guessing until the solution is found. NWA steps, UCA steps and even patterns can be used for free, but when stuck, a guess is made and the difficulty is increased by one. Nested guesses are also allowed.

Since the *minimal difficulty* needs to be calculated, this can take a very long time to calculate in practice.

4.2 Step VS. Switch

The step difficulty and the switch difficulty are quite similar; they both use the same approaches and are meant for the same set of puzzles, the puzzles of the easy category. However, there are two practical differences between the two difficulty measures.

Because the step difficulty measure needs to find the minimal difficulty of a puzzle, it needs to calculate every different sequence of steps. Using certain steps before others can affect the difficulty measure, and therefore, it might require a huge amount of calculation time. Opposed to that, the switch difficulty measure does not care about the sequence, and can be calculated very quickly compared to the step difficulty (even though that requires two separate calculations).

However, there is also a difference in the end result. The step difficulty measure yields more distinct difficulties than the switch difficulty measure. Where two puzzles might have step difficulties of 6 and 8, they can both have the same switch difficulty of 3. Of course, duplicate difficulties also occur using the step difficulty measure, but significantly less frequent.

Because of these two differences in difficulty measures, it is hard to pinpoint which is the best. If calculation time is not important, using the step difficulty is certainly recommended. But if there is a limited amount of time to calculate the difficulty of a puzzle, the switch difficulty measure is more useful, since it yields results a lot quicker.

5 Asymptotically difficult puzzles

Having found a system of difficulty measures, it is only natural to wonder how to make a puzzle that is as difficult as possible; that is, a puzzle that gets the highest rating from these difficulty measures. We only used the step and switch difficulty measures, to find puzzle formats that are asymptotically difficult.

This means we aim to find some format that will require as many operations (steps or switches) per cell in the puzzle as possible. To do so, a format has to be found in which one operation will reveal as few cells as possible (preferably just 1), so the upper bound of the format becomes n^2.

These formats do not need to be complicated; the only goal is to get a high difficulty rating. If a human player were to solve the puzzles, it would be easy to recognize patterns.

5.1 Step

To create the most difficult puzzle (or rather: format) for the step difficulty measure, we had to make sure that every placement of a light or non-light will cause the fewest new opportunities for marking other cells. This to decrease the calculation time of the puzzle, and simply to reduce the amount of cells found per operation. To do so, the "rooms" in which the lights can shine ought to be as small as possible (reducing the number of cells marked by a single light when it is placed). Note that the puzzle should not be impossible, therefore, one has to make sure that it is still solvable by the two (easy) operations.

The small 2-cell rooms we use are connected by means of a numbered cell (a `1`, to limit as many options as possible); see Figure 29.

Solving these puzzles is not difficult, but does call for a lot of operations; a puzzle of size $n \times 2$ (where n has to be an even number) will need $n/2$ NumberedWallOperations and $\lfloor n/4 \rfloor$ unlit cell-operations. In total: $\lfloor 3n/4 \rfloor$

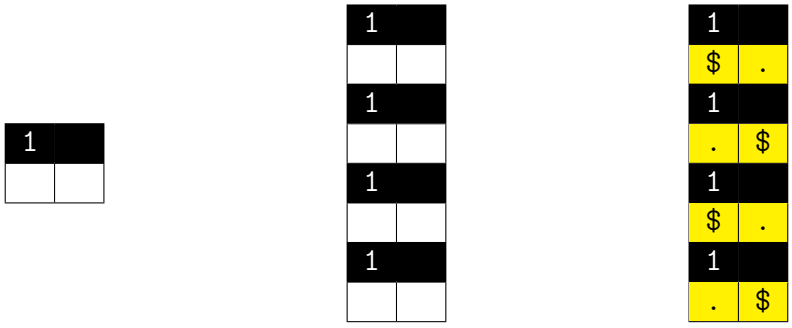

Figure 29: Connecting rooms

operations.

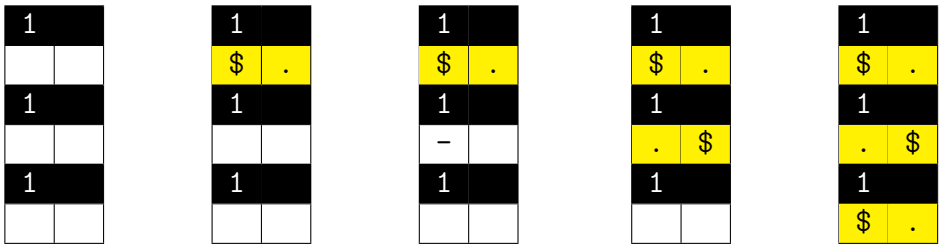

Figure 30: Using sequence (NWO, NWO, UCO, NWO) until finished: 4 operations

In short: we are capable of creating a puzzle of linear difficulty using this format.

We can extend this construction to a puzzle format with difficulty $c \times n^2$, for some constant c. To do so, we need to extend the current format sideways, creating multiple columns. This does cause for the rooms to swap their rows between columns, so that the rooms remain separate. This also enables passing from one column to the next. What also needs to be done is changing some of the numbered walls in the second and all next columns from `1` to `2`, in order to accomodate the lights in the rooms in the previous column. See Figure 31.

This format can create puzzles of a $m \times n$ rectangular grid, provided both height and width are even numbers. The following formulas apply to solving these puzzles, where O_{total} denotes the amount of operations needed, O_{NWA} the amount of NumberedWallApproach operations needed, and O_{UCA} the amount of UnlitCellApproach operations needed:

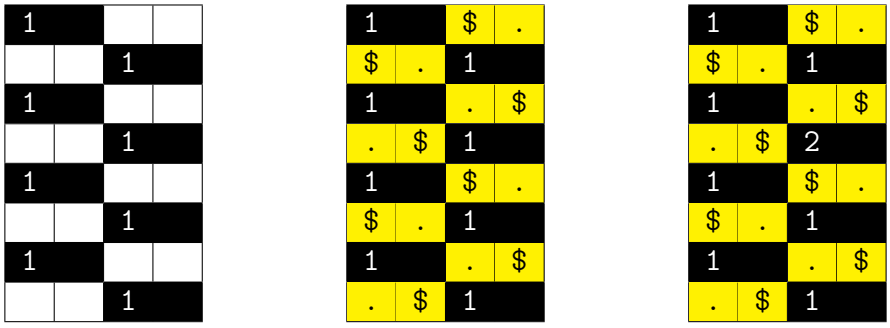

Figure 31: Creating multiple columns

$O_{NWA} = \frac{n}{2} \times \frac{m}{2}$
$O_{UCA} = \lfloor \frac{n}{4} \rfloor \times \frac{m}{2}$
$O_{total} = O_{NWA} + O_{UCA}$
$O_{total} = \lfloor \frac{3}{4} n \rfloor \times \frac{m}{2} = \lfloor \frac{3}{8} mn \rfloor$

5.2 Switch

To create a similar puzzle format for the switch difficulty, one needs a somewhat different approach. After all, simply eliciting step after step is not enough; these steps need to be sequenced ⟨..., NWA, UCA, NWA, UCA, ...⟩. Of course, preferably one NWA step is followed by one UCA step, etcetera, to ensure a maximum amount of switches accompanying the steps. Furthermore, the amount of cells discovered by the steps needs to be as small as possible.

We found a linearly extendable format that does just this, not using "rooms", but by using a slithering pathway; see Figure 32.

When solving this puzzle, it is clear that there is no other place to start than with the `0` at the top: a NWA step. This then leads to a UCA step of lighting up one of the cells just marked containing no light. This again causes for another NWA step to mark every empty cell next to the now-satisfied `1` to contain no light, etcetera.

When starting with NWA as the current approach, this puzzle (of size 6×2) will take 5 switches to solve it. When starting with UCA, it will take 6 switches to solve it. The switch difficulty measure is 5.5; see Figure 33.

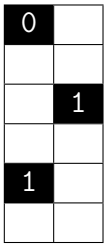

Figure 32: Slithering pathway

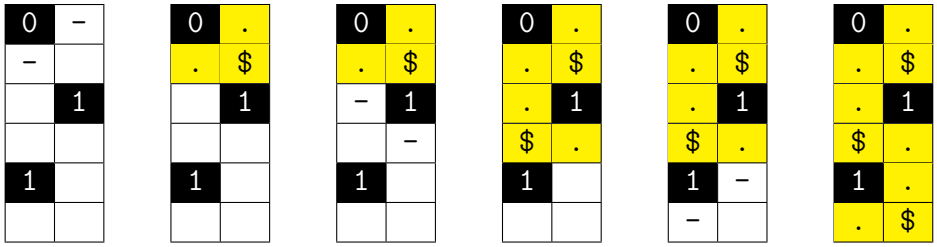

Figure 33: Solving the puzzle: 5 switches

Of course, the puzzle can be extended vertically by simply attaching more blocks of one numbered wall (with number 1) and three empty cells, so that the location (left or right) of the numbered wall switches every time. Per block, the puzzle difficulty will increase by 2 (one more NWA is needed, and one more UCA is needed). The difficulty of a $n \times 2$ puzzle with this format is $n - \frac{1}{2}$.

However, just like our difficult step puzzle, we want to create a puzzle that can also increase in width. By using more columns, we are able to do so, although these columns need to be separated by walls. To enable passing from one column to the next (at the end of a column), a single wall can be omitted; see Figure 34.

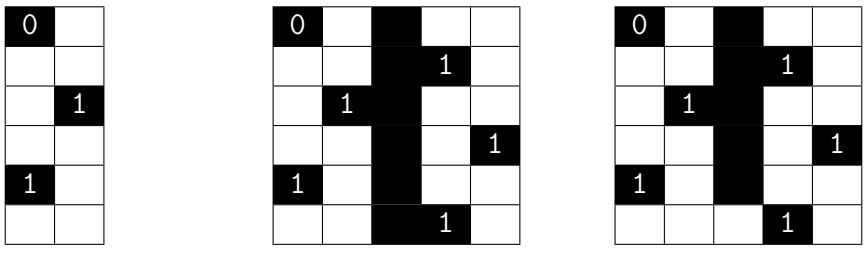

Figure 34: Adding another column

This can be done any number of times to increase the width of the puzzle. This extra column brings more switches with it: the amount of blocks it has, times 2. Unfortunately, it also brings a number of walls that do not offer more switches, but this seems unavoidable.

The formulas applying to the difficulty measure of a $m \times n$ puzzle of this format, where $\frac{m-2}{3} + 1 \in \mathbb{N}$ and n is even, S_{NWA} is the amount of switches needed when starting with the NumberedWallApproach as the current approach, S_{UCA} is the amount of switches needed when starting with the UnlitCellApproach as the current approach,and $S_{average}$ is the average amount of switches, are:

$S_{NWA} = (n \times (\frac{m-2}{3} + 1)) - 1$
$S_{UCA} = (n \times (\frac{m-2}{3} + 1))$
$S_{average} = (n \times (\frac{m-2}{3} + 1)) - \frac{1}{2}$

6 Conclusions and further research

In this paper, we have discussed Light Up puzzles; we have shown several approaches to solving these puzzles, either by using numbered walls, unlit cells, patterns or guesses. We have also proposed a system of difficulty measures with which every Light Up puzzle can be supplied with a difficulty rating. Using step, switch, pattern and guess difficulties, one can always supply a correct difficulty to a puzzle.

This system consists of three levels: easy puzzles, medium puzzles and difficult puzzles. To solve easy puzzles, the NumberedWallApproach and UnlitCellApproach are sufficient. Their difficulty is measured by either finding the minimal number of uses of a single approach (the step difficulty) or by finding the minimal number of switches in between approaches (the switch difficulty).

A medium puzzle is a puzzle that cannot be solved using only the NumberedWallApproach and UnlitCellApproach, but also needs to use predefined patterns to solve it. A difficulty is applied by finding the minimal number of patterns that need to be used in order to solve the puzzle.

Finally, a difficult puzzle cannot be solved by using the NumberedWallApproach, UnlitCellApproach and predefined patterns, but also needs guesses in order to solve it. By placing a light in a currently empty cell

and continuing to solve the puzzle with it, until it is either completed, or a contradiction arises, one can make 'guesses'. The difficulty of this puzzle is determined by the minimal number of guesses needed to solve the puzzle.

Furthermore, we have shown two different puzzle formats whose difficulty rating increase with the size of the puzzles. These two formats apply to either the step or the switch difficulty measure, and they are designed to optimize the number of steps or switches needed to solve them; see Figure 35.

Figure 35: Formats for the difficult step and switch puzzles

Any further research we propose can involve either patterns, guesses, Boolean expressions, or "light-up logic". Our list of predefined patterns is anything but complete; it contains some of the smallest useful patterns, but not all of them. A more complete database of patterns can be made, as well as a formal definition of what is and what is not a pattern.

The subject of guesses allows for more research as well. Currently, no distinction is made between linear guesses and nested guesses. Making such a distinction can, of course, affect the system of difficulty measures.

Another subject of further research we propose is that of Boolean expressions; since Light-Up puzzles are $\mathcal{NP}$-complete, they can be transformed into SAT problems. There are many SAT solvers that could solve Light-Up puzzles that way, and research could be done if any sort of difficulty measure in SAT problems also apply to the Light-Up puzzles.

Finally, the "light-up logic" hinted at in Section 3.1 is an interesting approach as well: extending the function $L(x)$, its definition and its aspects to see what can be done with this sort of logic.

References

[1] B. McPhail, *Light Up is NP-complete*, 2005, http://www.mountainvistasoft.com/docs/lightup-is-np-complete.pdf.

[2] S. Yen and S. Chiu, *A Simple and Rapid Lights-Up Solver*, in Proceedings of the 2010 International Conference on Technologies and Applications of Artificial Intelligence (TAAI), pages 440–443, IEEE, 2010.

[3] Í. Tatlı, *CEng 561 Term Project Implementation Report Guidelines*, white paper, METU Computer Engineering, Ankara, Turkey, without year, http://www.ceng.metu.edu.tr/~e1395557/akari.

[4] I. Rosberg, E. Goldbarg and M. Goldbarg, *Solving the Light-Up with Ant Colony Optimization*, in Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC), pages 566–573, 2011.

[5] E.G. Ortiz-García, *A Hybrid Hopfield Network-Genetic Algorithm Approach for the Lights-Up Puzzle*, in Proceedings of the 2007 IEEE Congress on Evolutionary Computation, pages 1403–1407, 2007.

[6] S. Salcedo-Sanz, L. Carro-Calvo, E.G. Ortiz-García, Á.M. Pérez-Bellido and J.A.Portilla-Figueras, *A Nested Two-Steps Evolutionary Algorithm for the Light-Up Puzzle*, ICGA Journal 32 (2009), 131–139.

[7] K.J. Batenburg and W.A. Kosters, *On the Difficulty of Nonograms*, ICGA Journal 35 (2012), 195–205.

[8] N. Fagerburg, F. Mascia and T. Stutzle, *It is Easy to Light Up in Practice*, Technical Report TR/IRIDIA/2014-004, Université Libre de Bruxelles, 2014.