Shedding Some Light on Light Up with AI

Anonymous Author(s)
Affiliation
Address
email

Abstract

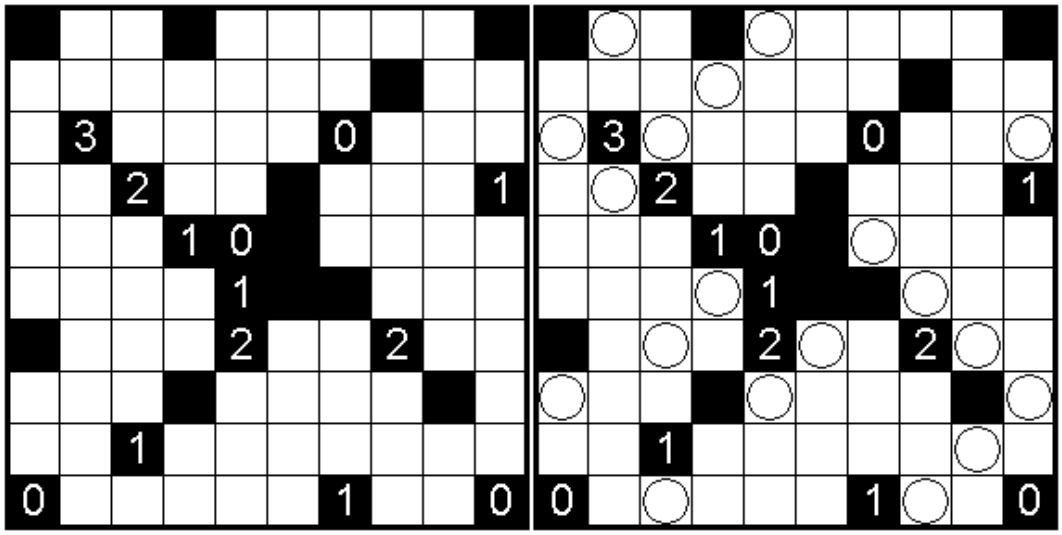
This project was an effort to apply techniques learned in Auburn University's COMP 6600/6606 Artificial Intelligence course to solving the Light Up puzzle. Several algorithms were explored for producing optimal solutions to randomly-generated initial boards, including hill climbing, simulated annealing, and deep neural networks.

1 Proposal

For our final project, we will attempt to develop an artificial intelligence capable of solving the Light Up logic puzzle, also known as "Akari." The game works like this:

1. Each puzzle begins with a rectangular grid of black and white cells.
2. Players must place light bulbs in white cells, in an arrangement such that:
   (a) Every white square is illuminated (light bulbs send rays of light up, right, down, and left, unless blocked by a black square).
   (b) No two light bulbs shine on each other.
3. Players must pay attention to the numbers inside some black squares, which impose a limit on how many light bulbs can be placed in adjacent white squares.

Below is an example board (left) and its optimal solution (right), provided by Wikipedia. Developing simulators for the board is simple enough, so we will be able to focus on solving the problem, likely through an informed search algorithm like hill climbing, as well as simulated annealing.



Initial states: The boards will be generated at random using user-defined inputs, such as board dimensions and number of black cells.

Approach: First, lightbulbs will be placed at random initial positions, based on the given limit of adjacent bulbs for each black square. The board may contain some initial violations (such as two bulbs shining on each other).

Output: The system will attempt to produce an optimal solution, based on the performance measure, using several different search algorithms. Allowed moves will include placing a bulb on an empty cell and removing a bulb from a filled cell.

Algorithms: First, we will implement informed search algorithms — including various forms of hill climbing, as well as simulated annealing — to determine what moves to make. Second, after the simulated annealing method has been implemented successfully, a large dataset of randomly generated initial states, as well as the moves needed to solve them, will be saved by running the simulated annealing for approximately one week. Ideally, this will generate enough training data to make training a deep neural network possible, with the goal of solving the puzzle given any initial state. Our definition of performance will include how many white squares a move lights up (increasing performance), as well as how many violations it might cause (decreasing performance).

Baselines: Baselines will include various methods of hill climbing, simulated annealing, and deep neural networks. To compare their performance, we plan to analyze factors such as the total average time it takes for each method to solve a series of puzzles, the average number of moves made, and the average performance cost. It will be extremely important to explore various tuning parameters for each algorithm, such as temperature and probability for simulated annealing, and number of hidden layers, number of neurons per hidden layer, learning rate, and activation functions (e.g. cross entropy, sigmoid) for deep neural networks. Additionally, other game-specific parameters must be tuned, such as the number of neighbors which are generated for selection, conditions of search termination, and choice of hill climbing variant.

According to research by Brandon McPhail [1], the light up puzzle is NP-complete, though it is not discussed nearly as much in artificial intelligence cycles as problems like eight queens—making it a perfect problem to solve for a project like this.

2 Puzzle Formalization

The first step of this project was to formalize the Light Up puzzle into a format that could be understood by an AI — to this end, we chose to represent boards as matrices of single digits, with each representing a possible state for the space. They are as follows:

CELL_BLACK_ZERO = 0

CELL_BLACK_ONE = 1

CELL_BLACK_TWO = 2

CELL_BLACK_THREE = 3

CELL_BLACK_FOUR = 4

CELL_BLACK_FIVE = 5

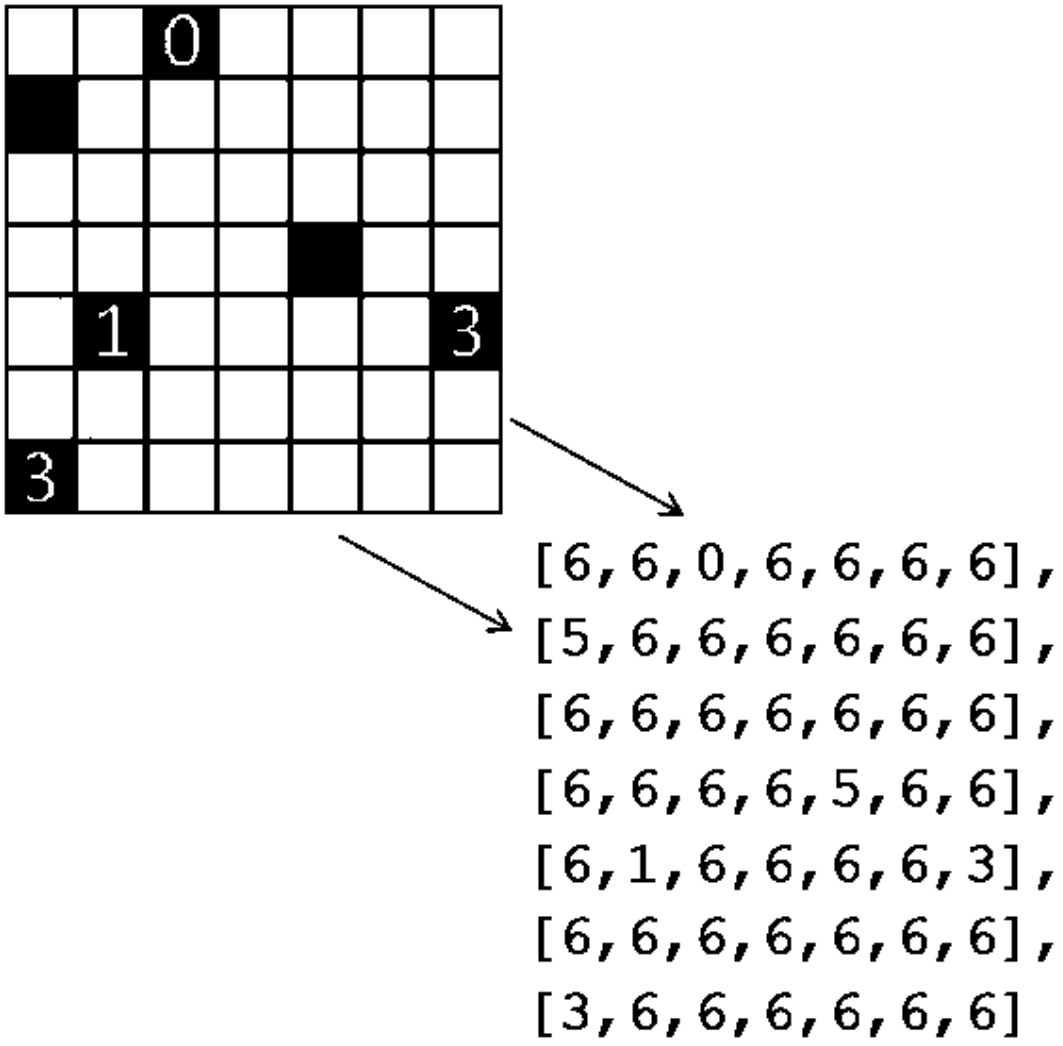
CELL_EMPTY = 6 (This value indicates an empty white cell.)

CELL_BULB = 7 (This value indicates a white cell with a bulb inside it.)

CELL_LIGHT = 8 (This value indicates that a cell is lit up by a bulb.)

CELL_BULB_ZERO = 9 (This value indicates that a bulb cannot be placed adjacent to this cell.)

With these definitions, any conceivable Light Up puzzle can be translated into a format readable by our program.



3 Hill Climbing

The work we did with hill climbing was divided into three phases:

1. Vanilla hill climbing with no special puzzle parameters.
2. Vanilla hill climbing with a puzzle with a single, unique optimal solution.
3. Simulated annealing with a puzzle with a single, unique optimal solution.

The first of these proved to be the weakest implementation — if the puzzle lacked a single, unique optimal solution, vanilla hill climbing could rarely produce a globally optimal solution. If there *was* a single, unique optimal solution, however, vanilla hill climbing *could* find it — given enough function evaluations. In one trial with an optimized puzzle, at 300 function evaluations, the vanilla hill climbing algorithm was able to find a globally optimal solution. This particular run was not representative of all runs with these parameters, however, which often required thousands of evaluations.

It was also found that allowing the algorithm three actions (adding bulbs, removing bulbs, and relocating existing bulbs) generally improved performance relative to only allowing the algorithm to remove bulbs and add new ones. Allowing the algorithm three actions generally allowed it to produce optimal solutions faster. A *t*-test was able to verify the statistical significance of this improvement.

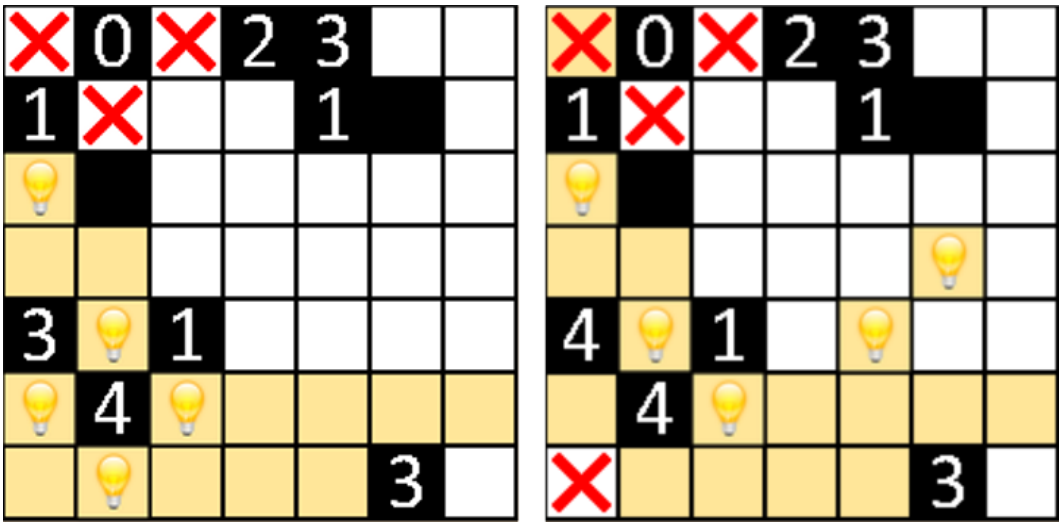
3.1 Simulated Annealing

Simulated annealing was found to be a much more robust algorithm than vanilla hill climbing, often producing globally optimal solutions in less than one thousand function evaluations. "Often" does not mean "always," however — in one instance, 8,700 evaluations were needed to find the global optimum. This is still an improvement over vanilla hill climbing with the same puzzle parameters, though, which sometimes needed over 10,000 evaluations to solve a problem of similar complexity.

4 Neural Network

The neural network we developed to predict optimal solutions was a feedforward neural network consisting of a 49-neuron input layer (for a 7x7 board), two hidden layers of 1,024 neurons each), and an output layer of 49 neurons. The activation functions utilized were ReLU and Softmax, and the optimization function utilized was stochastic gradient descent.

To train our network, we used a training set of 4,000 randomly-generated boards and their solutions, with 1,000 more reserved as a test set. Training on this data set took about 24 hours. This neural network proved to be quite good at predicting the outputs of the hill climbing algorithms, as can be seen in an example below (original on the left, predicted on the right).



The neural network did produce some common errors, however — some squares were labelled incorrectly. We suspect this is because the training set was too limited to fully capture the relationship between initialized boards and their solutions. We suspected using a convolutional neural network and gathering more data could help correct these errors, and attempted to pursue this.

4.1 Convolutional Neural Network

Our convolutional neural network used 49 input neurons, four two-dimensional convolution layers, and two dense layers (one being the 49-neuron output layer). The convolutional neural network was trained on the same data as the feedforward network, and ultimately produced similar errors. We take this to be evidence that, when training neural networks, having a large volume of quality data is far more important than the particular choice of neural network used.

5 Next Steps

While we are proud of the work we have done here, there is no doubt that more work (particularly, exploration of different algorithms) could yield even better results. We strongly believe that the Light Up puzzle is an excellent logic puzzle for introducing newcomers to the field of AI to some of the more advanced algorithms available to them.

Acknowledgments

Maybe our TA. He was nice, right? Gave us some points back on our homeworks?

And our professor, of course. Brownie points.

References

[1] McPhail, B. (2005, February 28). *Light Up is NP-Complete*. Retrieved October 10, 2020, from http://mountainvistasoft.com/docs/lightup-is-np-complete.pdf