Sudoku solver with Simulated Annealing

Guadalupe Cañas Herrera
Palmerina González Izquierdo

June 23, 2014

Abstract

We have developed a program, written in C language, that solves sudokus following the Simulated Annealing algorithm and we have compared it to a thermodynamical system, where the main variables are the energy and the temperature.

1 Introduction

A sudoku is a mathematical puzzle published for the first time at the end of the 70's. It became popular in Japan in 1986, and in 2005 it was spread around the world when several newspapers started to publish them in their pastime sections.
A sudoku is a 9x9 matrix subdivided in nine 3x3 squares. In a sudoku, some subset of cells are initially fixed. The aim is to fill each cell with an integer number between 1 and 9 so that the following criteria are fullfilled:

- Each row contains each of the integers 1 through 9 exactly once;
- Each column contains each of the integers 1 through 9 exactly once;
- Each 3x3 square contains each of the integers 1 through 9 exactly once.

	7	5		9				6	1	7	5	2	9	4	8	3	6
	2	3		8			4		6	2	3	1	8	7	9	4	5
8					3			1	8	9	4	5	6	3	2	7	1
5			7		2				5	1	9	7	3	2	4	6	8
	4		8		6		2		3	4	7	8	5	6	1	2	9
			9		1			3	2	8	6	9	4	1	7	5	3
9			4					7	9	3	8	4	2	5	6	1	7
	6			7		5	8		4	6	1	3	7	9	5	8	2
7				1		3	9		7	5	2	6	1	8	3	9	4

Figure 1: Example of an unsolved (left) and a solved (right) sudoku. [5]

Instead of solving a sudoku using the manual method, it is possible to work it out by writting a computer algorithm that finds the solution via simulating

annealing. Simulating annealing is a procedure of metaheuristic search for global optimization problems. The general goal for these kinds of algorithms is to find a good approximation of a function's optimal value in a large search space. The name annealing comes from a method used to clean crystalline surfaces: diffusion of contaminants from the bulk to the surface by annealing the sample (for further information about this process the reader is kindly referred to [4]).

2 Method

Our program is written such as it follows these steps:

2.1 Random Configuration

We initially asign to the non-fixed cells in each 3x3 block a unique random integer value between 1 and 9 so that the last criterion is satisfied. By summing the number of non-used values in each row and column of the puzzle we can compute the "energy". The total energy E is the sum of these values (this term would be an analogous to the energy in a thermodynamic system, for instance a NVT ensemble. In [1] the reader can find a detailed explanation about the NVT ensemble). The sudoku will be solved when the program gets a zero-energy configuration. A thermodynamic system is ordered when it fullfils the minimum energy principle.

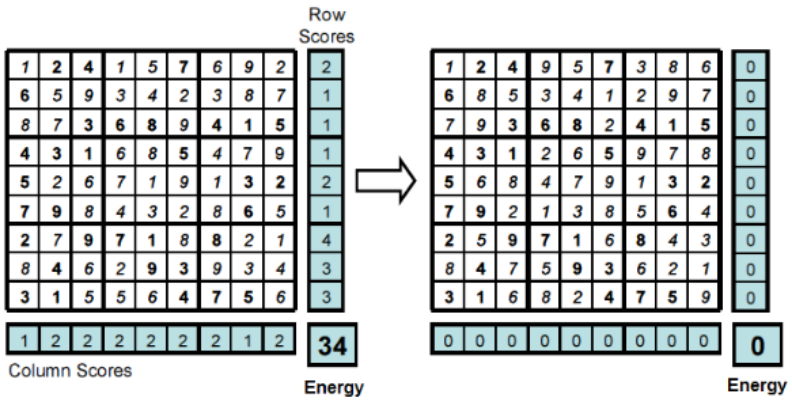

Figure 2: Example of the energy in an unsolved (left) and solved (right) sudoku ([2])

2.2 Choosing a neighbour

Given an initial configuration, it is possible to generate a neighbour configuration as follows,

1. Choose random values for cells i and j such as those cells are non-fixed and belong to the same 3x3 square.

2. Choose another random pair k and l in the same 3x3 square as before.

3. Swap the values in cells (i,j) and (k,l).

Note that this choice always preserves the last criterion mentioned in the Introduction.

2.3 Simulated Annealing

The last step of the simulation is the proper simulated annealing algorithm. First we define an initial "temperature" T (it is explained in the appendix how to find this value), a Cool Down Factor α and a number of Markov chains N with length M. Every time we repeat the two processes mentioned above we calculate the new energy E' and we accept the new configuration if:

$$E' < E \text{ or random number in (0,1)} < e^{\frac{E-E'}{T}} \tag{1}$$

And we reject it in any other case. Finally we decrese the temperature using the Cool Down factor ($T' = \alpha T$). We repeat this process until we get $E = 0$. The pseudocode will look like,

```
while energy is not zero {
        T = initial temperature;
        for n=1 ... N {
                for m=1 ... M {
                Generate a neighbour configuration
                //As explained in subsection 2.2
                E' = number of repeated values in the same column or row;
                if E' = 0 quit
                if E' < E or random number in [0,1) < exp((E-E')/T)
                E=E';
                else
                Reverting back to previous configuration
                }
                T=alpha*T;
        }
}
```

The while loop is necessary in case the program does not find the solution after N Markov chains. This can be seen as a reheat of the system at the initial value of the temperature.

3 Results

3.1 Solved Sudokus

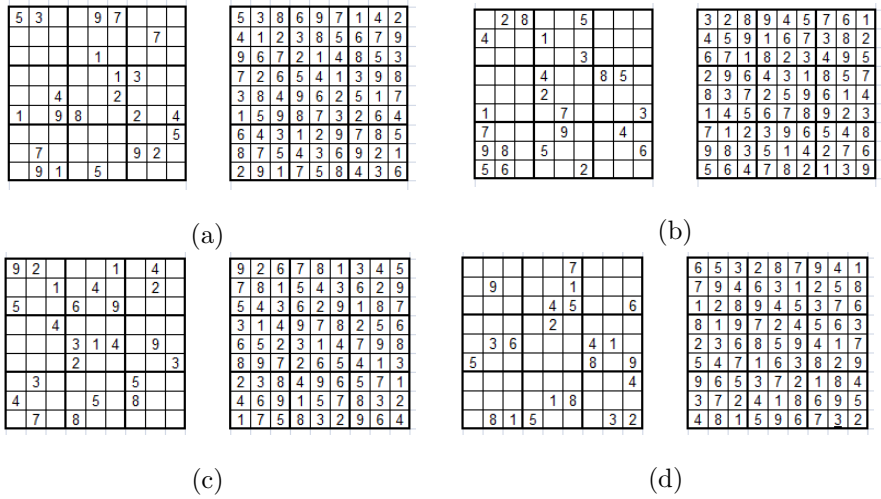

Figure 3: Examples of solved sudokus using our program

3.2 Outputs

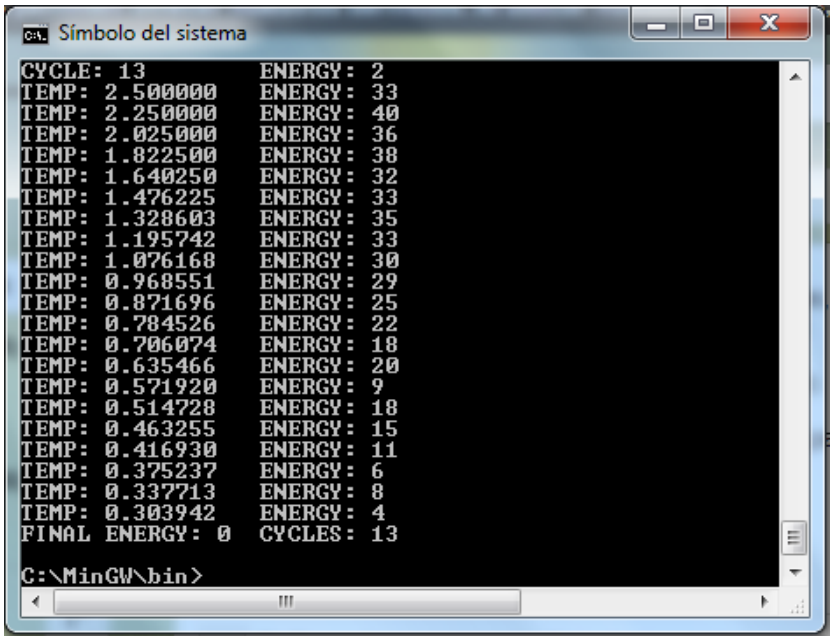

Figure 4: Snapshot of the terminal output showing one annealing process.

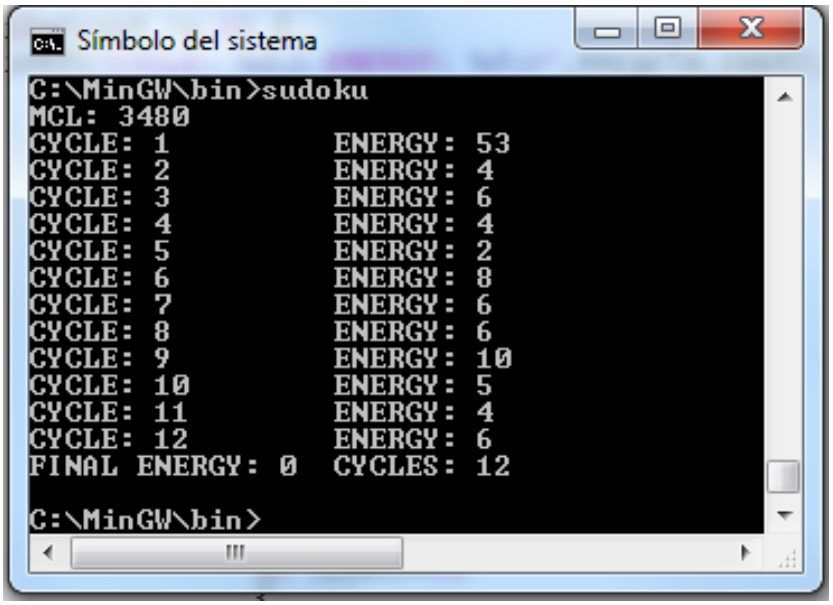

Figure 5: Snapshot of the terminal output showing the whole process composed by 12 annealings (cycles). After every annealing the temperature is restored to its initial value (re-heat process).

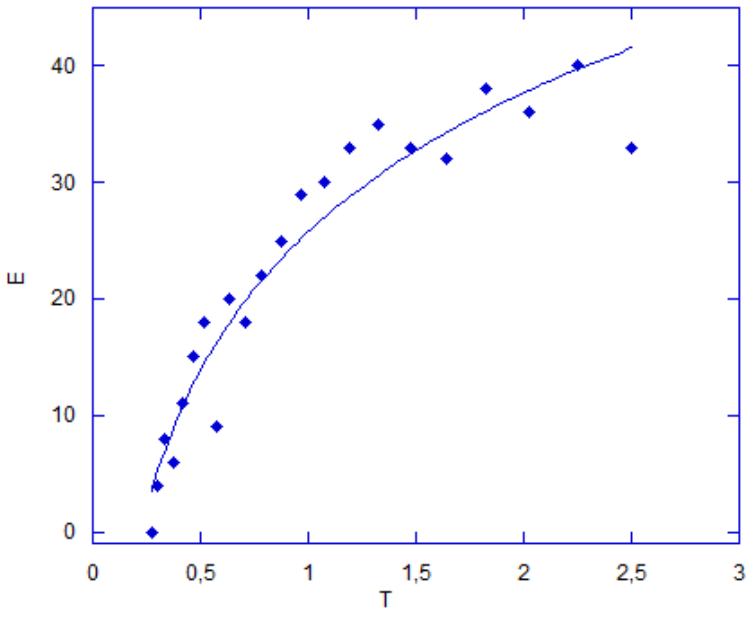

Figure 6: Dependence of the Energy as a function of the temperature for the data shown in figure 4. As it is posible to observe, the data follows a logarithmic dependence.

3.3 Cooling Factor

The cooling factor α is used to decrease the temperature after N Markov Chains. This cooling factor should be between 0 and 1; as α approaches to 1, the temperature is reduced too slowly. Nevertheless, if α is nearely zero, the temperature is reduced quickly. For these reasons, the best values of α need to be found experimentally. We found that the best values in order to optimize the program are $0.8 < \alpha < 0.95$.

4 Conclusions

We have proved how a thermodynamic model based on the idea of cooling the sample can be used to solve sudoku puzzles, trying to get the minimum energy and therefore the solution. This problem can be seen as an akin to the "Ising Model", where the minimum energy is reached when the system is organized.

Regarding the results, it is possible to observe how the energy decreases as the temperature does so. However, we can see in figure 5 how the energy is hardly reduced after the first cycle and then it experiments small fluctuations. This is due to the fact that every cycle is more difficult to find the cells that are needed to be swapped. The fluctuations can be explained thanks to the re-heating process, when the system is getting disordered.

5 Appendix

5.1 Initial Temperature

As we know, the acceptance rule is defined in equation (1). The maximum increase of the energy value every time we generate a new neighbour configuration is four (in the worst case scenario the energy of each row and column of the two swapped cells is increased by one).
Therefore we choose an initial temperature so that only the 20% of these "worst" transitions are rejected. If we substitute in equation (1):

$$\frac{(E - E')}{T} = ln(0.2) \rightarrow T = \frac{-4}{ln(0.2)} = 2.48 \tag{2}$$

where we have selected an approximated value of $T = 2.50$ for our program.

5.2 Markov Chain

Markov chains are the generating engine for the used set of states. Our aim is to generate number N of Markov chains with length M, which are states built up with probability proportional to the Boltzmann Distribution (see Chapter 1.2 in [3]). This probability is the transition probability $P(\mu \rightarrow \nu)$ and must satisfy four constraints,

- it must not depend on time
- it should only depend on the properties of the states μ and ν

- Ergodicity: it is possible to reach any state of the system from another one if we run it for long enough.
- Detailed balance: on average, the system should go from μ to ν just as often as it goes from ν to μ.

As it is shown in [2], the best value for the length of the Marcov chain is $M = A^2$ being A the number of non-fixed cells in the sudoku. Experimentally, using $N > 25$ did not suppose an increasing efficience of the program.

References

[1] L.Filion/R.Van Roij. "Advanced Statistical Physics: Lecture Notes". Utrecht, September 10, 2013

[2] R. Lewis, "Metaheuristics can solve sudoku puzzles". Journal of Heuristics, pages 387–401, volume 13, number 4, August, 2007.

[3] M.E.J Newman/G.T Barkema, "Monte Carlo Methods in Statistical Physics". Ed. Oxford University Press. Oxford, 1999.

[4] I. Swart/D. Vanmaekelbergh, "Solids and Surfaces. Reader", Chapter 2, page 15. Debye Institute for Nanomaterials Science. 2013.

[5] www.sciencenews.org/article/sudoku-solution

.