PHYS4061 Project B – A Stochastic Way to Solve the Sudoku Puzzle

LO Man Hei

Department of Physics, The Chinese University of Hong Kong, HKSAR, China

(Dated: December 19, 2019)

This paper will introduce an algorithm to solve the Sudoku puzzle based on a stochastic optimization approach. This approach is being compared with the brutal-force searching and the Metropolis–Hastings algorithm. Although this stochastic optimization approach is defeated by the Brute-force searching in solving the Sudoku. But this approach shows its capability of solving a complex system that cannot be done with the Metropolis–Hastings algorithm

I. INTRODUCTION

This paper will present an interesting method to solve the well-known mathematic puzzle, Sudoku. Sudoku is a number-placement puzzle invented by American Howard Garns in 1979 [1]. The objective is to fill a 9×9 grid with digits such that every column, row, and the nine subgrids contain all of the digits from 1 to 9.

Sudoku can be solved by a wide range of solving algorithms, including brute-force searching, constraint programming, and stochastic optimization methods [1]. This paper will focus on the last technique, stochastic optimization method, to examine the performance of this method over the others, and to compare the similarity of Sudoku with realistic physical system.

II. WORD DEFINITION

Before continuing to the next section, some words need to be clarified here

A. Grid(s)

Grid(s) refers to the biggest question frame of Sudoku, which contains 9 subgrids and 81 numbers

B. Subgrid(s)

Subgrid(s) refers to that 9 blocks in the Sudoku.

C. Energy

Energy refers to the total repeated numbers across each row, column and among subgrids. For example, there is a row filled like:

1	2	2	3	3	3	5	6	7

Then the number 2 is repeat once and number 3 is repeated twice, then the total repeated numbers of this row is 1+2=3. The above calculation will be done on all rows, columns and grids

III. METHOD

The stochastic optimization method used in this paper is of the family of simulated annealing [2] that modified to play the role of a sudoku solver.

A. The Solving Algorithm

1. Assign the remaining numbers to each subgrid. For example:

3		
	1	**6**
		4

   The remaining numbers of the above subgrid are 2, 5, 7, 8 and 9. If assigning the number based on ordered scheme, the subgrid becomes

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

   If assigning the number based on randomized scheme, the subgrid can be

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

2. Randomly select a (non-given) number from a subgrid, i.e. 2,5,7,8 and 9 are the non-given numbers of the above subgrid. Then evaluate the corresponding energy as if the possible exchange in the selected subgrid is being performed.
3. The exchange is based on a probability mapping over the energies of possible exchanges. This probability function is:

$$P(i) = \frac{exp(-E_i \,/\, Constant)}{sum[\, exp(-E_j \,/\, Constant)\,]}$$

   where i is the selected exchange, j is all of the possible exchange, E_i is the energy regarding the i exchange, Constant is selected as 0.47466 here, its

property will be discussed in the coming section. Now take an example, let the number 5 is selected, then the possible exchanges are 5<-->2, 5<-->5, 5<-->7, 5<-->8 and 5<-->9. And let the energies after these possible exchanges as 30, 28, 26, 25 and 24 respectively. Then the probabilities to perform each exchange is 2.85E-6, 1.93E-4, 1.30E-2, 0.107 and 0.880 respectively.

4. Goto (2) until zero energy is achieved.

B. Performance Benchmark

As to reduce the bias due to the random nature of the stochastic optimization, the result will be based on solving 100 different questions and solving each by 10 times.

This paper will benchmark the performance of the stochastic optimization method by comparing the time and memory usage with brute-force searching. It will also compare the performance of the aforementioned method and the Metropolis–Hastings algorithm [3].

IV. DEMONSTRACTION

The demonstration of stochastic Sudoku solver is available on: http://lomanhei.ddns.net/sudoku.php .
After entered the page, you may see the solver like Fig. 1. There are several functions in that page:

1. You may input your question by filling in the grid on the left.
2. If you do not have any questions, you may obtain one by clicking "GIVE ME A QUESTION"
3. You may empty the grid by clicking "RESET ALL".
4. You may solve the input question by clicking "SOLVE IT"
5. After the question is solved, the solution will be displayed next to the question, as shown in Fig. 1
6. You may see the evolution of the solving process by clicking "PLAY". By default, it will start from the first frame. And you may stop the animation by clicking "STOP".
7. You may see any specified frame when the animation is stopped and input the frame number in the text box next to the "PLAY" button.

V. RESULT AND DISCUSSION

A. Solving Process

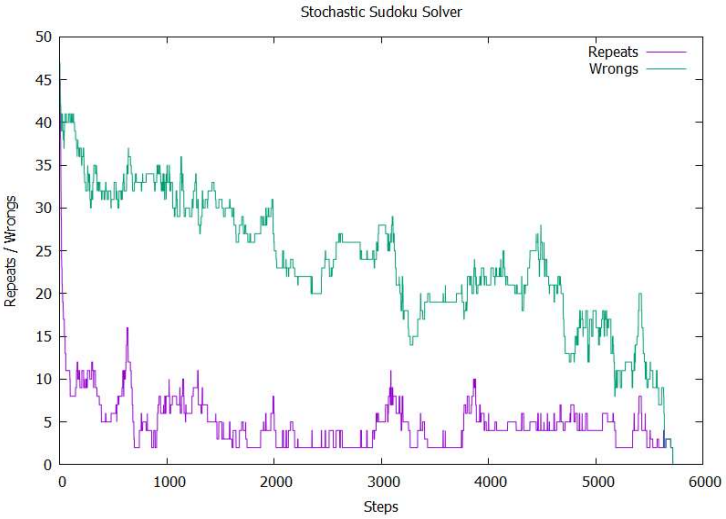

Fig. 2. Profile of the solving process

We will use the same question of Fig. 1 as example, the

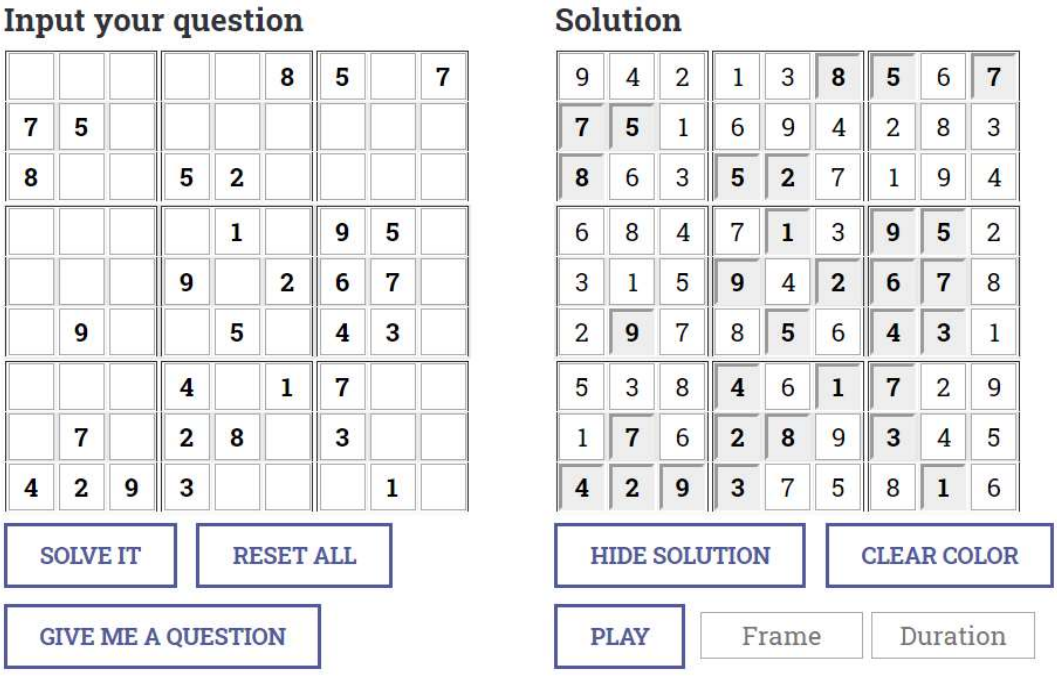

Fig. 1. Snapshot of the online solver

solving profile of showing the number of wrong numbers with green line and the number of repeated numbers with purple line, the energy, is shown in Fig. 2.

The wrong number is that, during the solving process, if the current number mismatched with the solution at a certain position, then that number is consider as wrong.

As shown in Fig. 2, the quantity being optimized, the energy, is gradually decreased in the first few hundred steps. The value of energy was dropped as low as 2 at around the 800th step. However, the puzzle was solved until the 5721st step. It shows that the correlation between the energy and the number of wrongs is not 100%. Although zero energy guaranteed the correct solution, a slightly positive energy still leading to several possible configurations.

But considering the number of wrongs cannot be examined until the Sudoku is fully solved, the performance by optimizing the energy is still astonishing.

B. Property of the Probability Function

Now, discuss the property of the probability function in the algorithm:

$$P(i) = \frac{exp(-E_i \,/\, Constant)}{sum[\ exp(-E_j \,/\, Constant)\ ]}$$

Recalling the formula of the Boltzmann distribution [4] of a certain state in a system:

$$P(i) = \frac{exp(-E_i \,/\, kT)}{sum[\ exp(-E_j \,/\, kT)\ ]}$$

where P(i) is the probability of state i, Ei is the energy state i, k is the Boltzmann constant, T is the temperature, and j is the index of all accessible state. By inspection, these two equations are almost the same just differing by the notation of constant kT.

The sense of the probability function being the Boltzmann distribution is that: assuming each step of iteration is to relax a selected position with another arbitrary position in the subgrid. Then the possible exchanges are the possible accessible state given that other subgrids are rigid.

Thereafter, it raised an interesting meaning to the Constant, the temperature. And the solving process is analogy to annealing the Sudoku toward the given temperature. If that temperature is high, the Sudoku will slowly or not converge to the solution. If the temperature is too low, the Sudoku may suck into a local minimum and not converge to the solution. Nonetheless, every Sudoku has its own "melting point", so the choice of the Constant must be very careful. The selection of the number 0.47466 is based on a trial and error approach on solving a set of Sudoku questions with various difficulties.

C. Compare with the Metropolis Rule

Now to compare the different between this method and the Metropolis Rule [2]. The Metropolis Rule is to perform swapping between two numbers, if the energy is lowered, then the swap is 100% taken, otherwise the chance to swap is $exp(-\Delta E \,/\, Constant)$ where ΔE is the change in energy.

Recall the example in the method section, let the number 5 is selected, then the possible exchanges are 5<-->2, 5<-->5, 5<-->7, 5<-->8 and 5<-->9. And let the energies after these possible exchanges as 30, 28, 26, 25 and 24 respectively. If now the Metropolis Rule is being implemented, the probability of each exchange becomes $0.25exp(-2/Constant)$, $0.25[1 - exp(-2/Constant)]$, 0.25, 0.25 and 0.25 respectively. The probability is even out between the exchange of 5 to 7, 8 or 9.

In comparison, though Metropolis Rule is less consuming, it has much lower chance getting to the lowest energy state. The result of implementing the Metropolis Rule has a converging rate of 0%.

D. Performance Benchmark

TABLE I. The total memory and time usage of solving 100 different Sudoku and solving each by 10 times.

Method	Memory Usage	Time Usage
Brute-force searching	10.7GB	15.8sec
Stochastic optimization	282.1GB	498.8sec
Metropolis–Hastings	Not Converge	

VI. CONCLUSION

To conclude, the stochastic optimization approach maybe not the best method to solve a Sudoku, as it takes 28 times more memory and is 30 times slower than the Brute-force searching. But this approach shows its capability of solving a complex system that cannot be done with the Metropolis–Hastings algorithm,

VII. REFERENCE

[1] *Sudoku solving algorithms*. Wikipedia. (Visited at 2019). en.wikipedia.org/wiki/Sudoku_solving_algorithms

[2] W.H. Press et al. *Simulated Annealing Methods*. Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press. (2007)

[3] B.A. Berg. *Markov Chain Monte Carlo Simulations and Their Statistical Analysis*. Singapore, World Scientific. (2004).

[4] L. Landau et al. *Statistical Physics*. Course of Theoretical Physics. **5** (3 ed.). Oxford: Pergamon Press. (1976)