ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/326978036

Solving Sudoku Game Using Quantum Computation

Preprint · August 2018

DOI: 10.13140/RG.2.2.19777.86885

CITATIONS
0

READS
338

5 authors, including:

Sanghita Chandra
Indian Institute of Science Education and Research Kolkata
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Vardaan Mongia
Panjab University
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Bikash K. Behera
Indian Institute of Science Education and Research Kolkata
29 PUBLICATIONS **72** CITATIONS
SEE PROFILE

Prasanta K. Panigrahi
Indian Institute of Science Education and Research Kolkata
521 PUBLICATIONS **4,229** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Performing Quantum Computational and Quantum Informational Tasks on IBM Quantum Computer View project

Many-Body Systems View project

All content following this page was uploaded by Bikash K. Behera on 11 August 2018.

The user has requested enhancement of the downloaded file.

Solving Sudoku Game Using Quantum Computation

Ankur Pal,[1,*] Sanghita Chandra,[1,†] Vardaan Mongia,[2,‡] Bikash K. Behera,[1,§] and Prasanta K. Panigrahi[1,¶]

[1] *Department of Physical Sciences, Indian Institute of Science Education and Research Kolkata, Mohanpur 741246, West Bengal, India*

[2] *Department of Physics, Panjab University, Chandigarh 160014, Punjab, India*

Quantum Sudoku solver uses counter-intuitive concepts of quantum mechanics such as superposition and entanglement as opposed to its classical counterpart, which has a linear complexity [O(n)]. Using concepts of dual computing and quantum computing we propose an algorithm which takes any incomplete 4 × 4 Sudoku puzzle and uses superposition and deletion of marked states with a logarithmic complexity [O(log(n)], thus providing an exponential speedup. Sudoku is a fun combinatorial game based on the Latin square which has wide applications in being an efficient design in controlling multiple sources of variable nuisance simultaneously. Our proposed algorithm has two primary aims; first is to solve a 4 × 4 Sudoku puzzle and then use quantum mechanics to show how superposition can be used by the human memory while solving Sudoku manually.

Keywords: Quantum Computing, Sudoku Game, Duality Computing, Duality Mode, Shi Doku

I. INTRODUCTION

Sudoku is a logic-based, combinatorial number-placement puzzle, known to cause brain stimulation, helps relax and is fun to play [1]. The history of Sudoku puzzles likely has it roots in the mathematical concept of Latin squares [2]. In 1780's, Swiss mathematician Leonhard Euler put forward the idea of arranging a given set of numbers such that any given number (or symbol) occurs only once in each row or column. Sudoku rules add the constraint that each region may only have the numbers (or symbols) occurring but once. Howard Garns, an Indianapolis architect, is credited for creating this rule when he developed the puzzle we today call as Sudoku [3]. For over 25 years, according to Dell Magazines the puzzle remained under the name of Number Place. A completed Sudoku grid (9 x 9) is a special type of Latin square with the additional property of no repeated values in any of the 9 blocks (or boxes of 3 x 3 cells called regions). Latin squares are extensively used in statistical analysis [4]. The general problem of solving Sudoku puzzles on $N^2 \times N^2$ grids of $N \times N$ blocks is known to be NP-complete [5, 6]. Algorithms such as backtracking and annealing [7] can solve most of the 9 x 9 puzzles efficiently. However, for a large value of n, a combinatorial explosion occurs which limits the properties of the Sudokus that can be constructed, analyzed and solved. There exist two categories of Sudoku in terms of being logically solvable; proper and improper. Improper Sudoku, which are considered to involve guesswork, pose a challenge in terms of solving even classically. Over the time, many variants of Sudoku have been introduced [8]. Work has been done on developing quantum games [9] with quantum strategies [10]. Efforts have also been made to use Sudoku for key distribution, describing how a Sudoku grid key can be secretly transported using quantum key distribution methods whereby partial grid (or puzzle) can be received and the full key can be recreated by solving the puzzle [11]. Though Sudoku solvers using classical algorithms existed, the first 'commercial' quantum computer for solving Sudoku was proposed by D-Wave [12].

Sudoku is a member of an important class of constraint satisfaction problems (CSP) [13]. There are two main aspects of problem difficulty. The first is the complexity of individual steps (logical operations) involved in solving the problem and the second aspect being the structure of dependency among individual steps, i.e., whether steps are independent (can be applied in parallel) or they are dependent (must be applied sequentially) [14]. One is yet to come up with a quantum algorithm which can deterministically achieve this feat for any N x N Sudoku. In the present work, we propose to solve the simplest 4 x 4 Sudoku (also known as Shi Doku) using quantum computing and implement the concept of duality computing using a probabilistic approach.

The rest of the paper is organized as follows. In Secs. II & III, we discuss briefly on quantum computing, and duality computing and duality mode respectively. In Sec. IV, we propose a Python program which takes an unsolved Sudoku as input and identifies the cells where there will be a collapse of states in the next step. In Sec. V, we present the quantum circuit which is used to perform the collapse of the correct number (state) in those marked cells. In Sec. VI, we propose how the algorithm propagates. Following which we discuss on how this algorithm can be generalized using more resources with increased success rate. Finally we conclude in Sec. VII, summarizing our work and what further developments can be made from the open ends.

* ap17ms047@iiserkol.ac.in
† sc17ms135@iiserkol.ac.in
‡ vardaan.chess.gm@gmail.com
§ bkb13ms061@iiserkol.ac.in
¶ pprasanta@iiserkol.ac.in

II. QUANTUM COMPUTING

Quantum computing is a process of computation which uses quantum principles such as superposition and entanglement. Classical computing uses binary digits (bits 0 and 1) to encode the data, whereas quantum computation, uses quantum bits (qubits $|0\rangle$ and $|1\rangle$), which can be in superpositions of states.

In quantum computing, it is necessary to manipulate the qubits in a manner that they return the desired result when read. This comes at the cost of design complexity of the circuit. In our algorithm we show that by using a quantum computer we may be able to solve some particular set of problems compromising exponential growth in the problem state space at the expense of exponential growth in computational time. Quantum computers operated in collaboration with classical algorithms for some parts can receive unprecedented results. In our algorithm, we use superposition to represent four elements in one cell by using two qubits, and quantum gates to make conditional deletions, and entanglement to isolate the marked state, thereby, deleting it with the aid of quantum operations. Here we use a Python program along with the above quantum principles to efficiently solve the Sudoku game.

III. DUALITY COMPUTING AND DUALITY MODE

Deleting a marked state from an arbitrary set of basis can be done by simulating a duality computer [15] on a quantum computer. This can be made by running a duality mode and recycling quantum computing, thus, providing a quantum computer simulation of the duality computer [16]. A duality computer is a moving quantum computer passing through d-slits, i.e., it takes the qubit to a superposition state, which is retrieved at each slit with a certain weightage. Unitary operations can then be performed on each of these obtained superposition states. In addition to this, the duality mode empowers us to perform summation of unitary operations apart from known unitary operations in a quantum computer. Two such gates, namely Quantum Wave Divider (QWD) and Quantum Wave Combiner (QWC), are used in our algorithm. To simulate an n-qubit 2-slit duality mode, we need an (n+1)-qubit quantum computer where one qubit is required to create two slits and the rest n qubits work exactly as in a stationary quantum computer. We make the following correspondence between duality computer and duality mode simulated on a quantum computer:

$$|\psi\rangle|k_u\rangle \leftrightarrow |\psi\rangle|0\rangle, |\psi\rangle|k_d\rangle \leftrightarrow |\psi\rangle|1\rangle \tag{1}$$

where $|k_u\rangle$ ($|k_d\rangle$) is the center of mass for translational motion of upper (lower) sub-wave function. When the auxiliary qubit is in $|0\rangle$ ($|1\rangle$), it resembles a duality computer sub-wave from the upper (lower) slit. Initial and final wave functions of the duality computer are ascribed to the auxiliary qubit that is in state $|0\rangle$. Thus the initial state of the duality computer is $|\psi\rangle|0\rangle$. We perform the QWD [17], by using a Hadamard gate, to switch on the duality mode, and the state becomes

$$|\psi\rangle\frac{|0\rangle+|1\rangle}{\sqrt{2}},$$

namely, the QWD operation is equivalent to a Walsh-Hadamard operation on the auxiliary qubit. Conditional gates can simulate gate operations on different slits.

Then the wave function becomes

$$\frac{U_0|\psi\rangle|0\rangle+U_1|\psi\rangle|1\rangle}{\sqrt{2}}$$

The QWC [17] operation can be simulated by a Walsh-Hadamard operation on auxiliary qubit to switch off the duality mode. After QWC, the wave function becomes, $\frac{U_0+U_1}{2}|\psi\rangle|0\rangle + \frac{U_0-U_1}{2}|\psi\rangle|1\rangle$ measurement is performed on the n qubits in the condition that the auxiliary qubit is in $|0\rangle$ state. Then the wave function is collapsed and $(U_0+U_1)|\psi\rangle$ result is read out. The probability of obtaining a result is $P_0 = \frac{\langle\psi|(U_0+U_1)(U_0+U_1)|\psi\rangle}{4}$.

The probability of not obtaining a result is $1-P_0$ and if this occurs, the state in $|1\rangle$ collapses, and the wave function becomes,

$$|\psi'\rangle = N'\frac{U_0-U_1}{2}|\psi\rangle|1\rangle$$

where N' is the renormalization factor.

Then a unitary recovering operation V is performed on the n qubits to restore the initial input state. It then flips the auxiliary qubit state $|1\rangle$ to $|0\rangle$. The (n+1) qubits are recovered to the initial states of the quantum circuit. The calculating process then recycles again and again until the conditional measurement is performed to obtain a result. This process is known as the recycling quantum computing mode.

IV. PYTHON PROGRAM TO FIND FURTHER CLUES

Here in the algorithm, we incorporate classical and quantum computing conjointly. Our quantum circuit isolates the elements which violate constraints, however we will not be able to gather any information about which cells contain only one element after deletion without measuring the qubits. And if we do perform measurements, the qubits will collapse to one of the states in the superposition, hence making it impossible to know how many states were there in the superposition before measurement. Interestingly, this can be achieved by using classical methods. Hence, here we use Python to program our code, which marks the cells which would collapse to a specific number (new clue) in the next step of solving the Sudoku. It has been provided in the supplemental material.

V. QUANTUM CIRCUIT

We take 64 qubits to represent our Sudoku board. Four qubits are assigned to each cell, amongst them two represent the possible elements in a cell (See Fig. 1). For Sudoku, we need superposition of four elements that can be prepared by using two qubits. Each element has been assigned to the corresponding quantum state as shown in Table I.

State	Element
$\|00\rangle$	1
$\|01\rangle$	2
$\|10\rangle$	3
$\|11\rangle$	4

TABLE I. **Quantum states and corresponding elements**

In the following Fig. 1, we represent the representation of qubits for a 4 × 4 Sudoku board. Different 2×2 blocks are illustrated in different colours and each cell are indexed as (X,Y), where X={A, B, C, D} and Y={1, 2, 3, 4}.

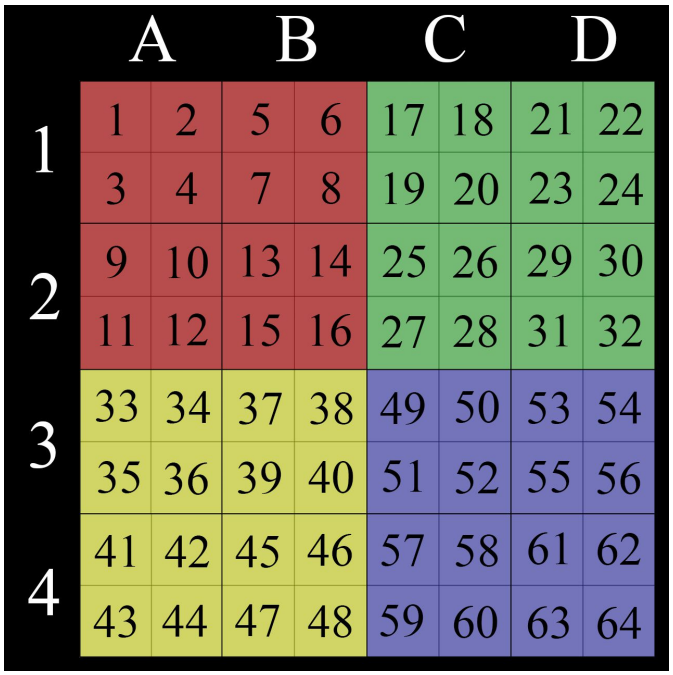

FIG. 1. **Representation of Sudoku board.**

The third qubit of every cell represents whether the cell has a clue (constraint) or not. Here we assume the state $|1\rangle$ as a clue. Fourth qubit is used as an auxiliary qubit for deletion of a marked state from a superposition of possible states. The first four qubits represent first cell in the first 2 × 2 blocks, next four represent the second cell in the first 2 × 2 blocks, and so on as given in Fig. 1. To begin solving our Sudoku, we must first input the clues in their proper places according to our index and put the third qubit of that cell in $|1\rangle$ state. One such unsolved Sudoku is shown in Fig. 2. All the cells with no initial clues (constraints) must have first two qubits in $|+\rangle$ states and the third qubit in $|0\rangle$ state. The fourth qubit of all cells is initially assigned to $|0\rangle$ state.

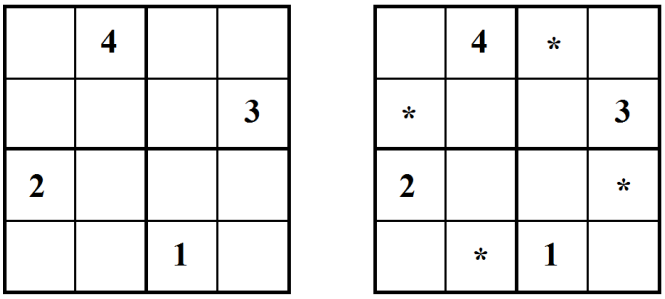

FIG. 2. **An unsolved Sudoku and marking of cells containing clues.**

In accordance with our algorithm, the circuit checks cell by cell whether it is a given constraint or not. The cells are 'marked' by the Python code thereafter which will turn out to be a clue in the next step, as shown in Fig. 2. We check for the third qubit of every cell to be in $|1\rangle$ state, and if so, our circuit triggers the deletion process depending on the state of the first two qubits of that cell. We use controls and anti-controls to trigger deletion of different states in 2 × 2 blocks, the band, and the stack [8] according to the clue as shown in Fig. 3.

For deletion process, we use the above mentioned duality mode. The quantum circuit used for deletion is given in Fig. 4.

The auxiliary qubit is put in $|+\rangle$ state by using a Hadamard gate, this is analogous to a quantum wave being split into two sub-waves (Hadamard gate acts as the QWD here). The $|0\rangle$ state represents one subwave and $|1\rangle$ state represents the other. Next step is to mark the state to be deleted by adding π phase to it in one of the subwave (in our case $|0\rangle$ state subwave). The state will be marked in correspondence with the clue (Table I). We then, apply another Hadamard gate to simulate the combination of two sub-waves (QWC), therereby, entangling the marked state with the $|1\rangle$ state of the auxiliary qubit and hence obtaining the rest of the states entangled with $|0\rangle$, consequently deleting the marked state. If the measurement result is $|0\rangle$, then we move forward with our algorithm, and if it is $|1\rangle$, then we know that it's not a desired result, so we discard the result and repeat the process. After the completion of this process, we will get some new clues due to deletion of three states. The cell index of these new clues can be anticipated using classical means, without knowing the clue itself. We only need to know the cell index of these new clues in order to prepare the third qubit of that cell in $|1\rangle$ state. Based on the cell index, the clue in the cell can be found by simply measuring the two qubits pertaining to the cell

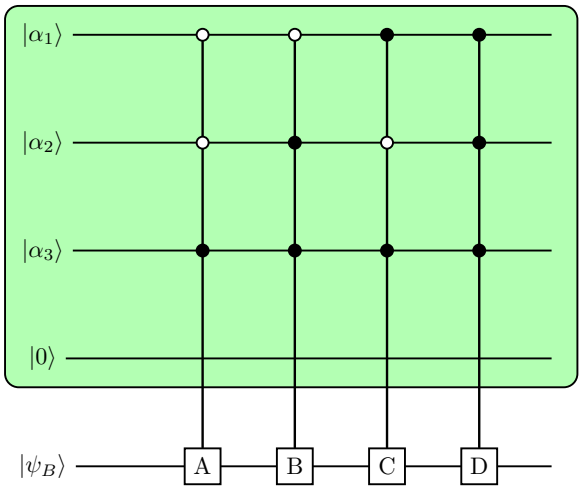

FIG. 3. **Quantum circuit illustrating the deletion of states**. Here, $|\alpha_1\rangle$, $|\alpha_2\rangle$, $|\alpha_3\rangle$, and $|0\rangle$ represent the corresponding quantum states of a cell respectively. From the state $|\psi_B\rangle$, the operations A, B, C, and D are used to delete the states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ respectively.

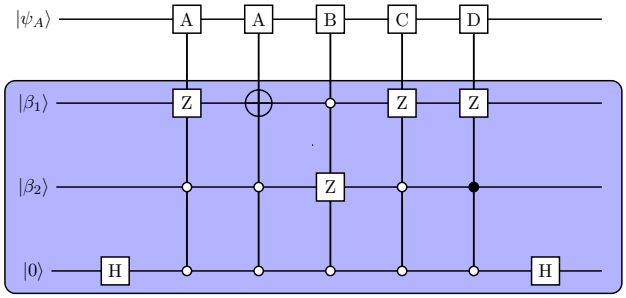

FIG. 4. **Quantum circuit depicting deletion algorithm**. Here A, B, C and D are control gates. A, B, C, or D is on when $|\psi_A\rangle$ is in $|001\rangle$, $|011\rangle$, $|101\rangle$, or $|111\rangle$ state respectively.

that contains new clues. These new clues can be fed to the classical part to anticipate the appearance of newer clues in the next iteration. Then further deletions will occur, resulting in new clues. After this iterative process 4×4 Sudoku will be solved.

VI. CIRCUIT FOR A 2×2 BLOCK

In Fig. 5, we provide the quantum circuit for 2×2 block. Noticeably, this circuit has a symmetry to it as can be expected from a generalized circuit built for solving any 4×4 Sudoku. Here, $|\alpha_1\rangle$ and $|\alpha_2\rangle$, $|\beta_1\rangle$ and $|\beta_2\rangle$, $|\gamma_1\rangle$ and $|\gamma_2\rangle$, and $|\theta_1\rangle$ and $|\theta_2\rangle$ represent the possible elements in 4 blocks respectively.

For first iteration, we will need to feed input to the circuit (Fig. 5). If one of the cells in the 2×2 block is a clue, we will input qubits corresponding to that cell accordingly (Table I). Also the third qubits, namely, $|\alpha_3\rangle$, $|\beta_3\rangle$, $|\gamma_3\rangle$, and $|\theta_3\rangle$, will be in state $|1\rangle$ if the corresponding cell is a clue. However, both the qubits that represent elements of non-clue cells, remain in $|+\rangle$ state, and the third qubit stays in $|0\rangle$ state.

To demonstrate, we will take an example where the first cell is given as 1 and third cell is given as 4. We input $|\alpha_1\rangle$ as $|0\rangle$, $|\alpha_2\rangle$ as $|0\rangle$, $|\gamma_1\rangle$, $|\gamma_2\rangle$ as $|1\rangle$; and $|\beta_1\rangle$, $|\beta_2\rangle$, $|\theta_1\rangle$, and $|\theta_2\rangle$ as $|+\rangle$. Then we input $|\alpha_3\rangle$ and $|\beta_3\rangle$ as $|1\rangle$, and $|\gamma_3\rangle$ and $|\theta_3\rangle$ as $|0\rangle$.

It can be observed that the states of $|\alpha_1\rangle$, $|\alpha_2\rangle$, and $|\alpha_3\rangle$ trigger the deletion of 2 ($|00\rangle$) from second, third, and fourth cell given that fourth qubit remains as $|0\rangle$ state. The states of $|\gamma_1\rangle$, $|\gamma_2\rangle$, and $|\gamma_3\rangle$ trigger deletion of 4 ($|11\rangle$) from first, second and fourth cell given that twelfth qubit remains as $|0\rangle$.

In case of the whole quantum circuit, deletions from cells in band and stack of the clue's cell take place. This results in deletion of three clues in some of the cells, making it a new clue. They indices of these new clues could be found out classically (using Python code). By measuring the qubits at these indices, we can find out the new clues, which can be fed back to the classical program to find out the indices of next set of clues. For second, and further iterations the input will be made accordingly.

VII. DISCUSSION

We have exploited superposition and combined it with duality mode in quantum computing to achieve a speed up in solving our Sudoku. The deletion process however, is probabilistic. Much like the Grover's algorithm [18], our algorithm doesn't give the desired output deterministically which is required for solving the Sudoku. For each deletion, there is (n-1)/n chance of getting the desired superposition state as output; where n represents the number or superposed states in input state $|\psi\rangle$. The proposed algorithm works with 64 qubits. We have reduced time complexity but at the cost of increasing our space complexity. These are two primary drawbacks which can be improved using better algorithms. Our central idea of using quantum computing to solve Sudoku can be extended to larger Sudokus. Apart from this, this protocol can be further applied to other CSP problems such as timetabling, scheduling etc [19]. Various sections of our algorithm uses identification, targeting, marking and

elimination which can have varied applications independently.

Here in the present work, we use duality and quantum computing to solve a Sudoku problem that involves marking of states and deletion of states. We propose new quantum circuits for the above purpose and use Python code for finding the clues in the Sudoku game. Our proposed algorithm provides an exponential speedup with a logarithmic complexity [O(log(n)].

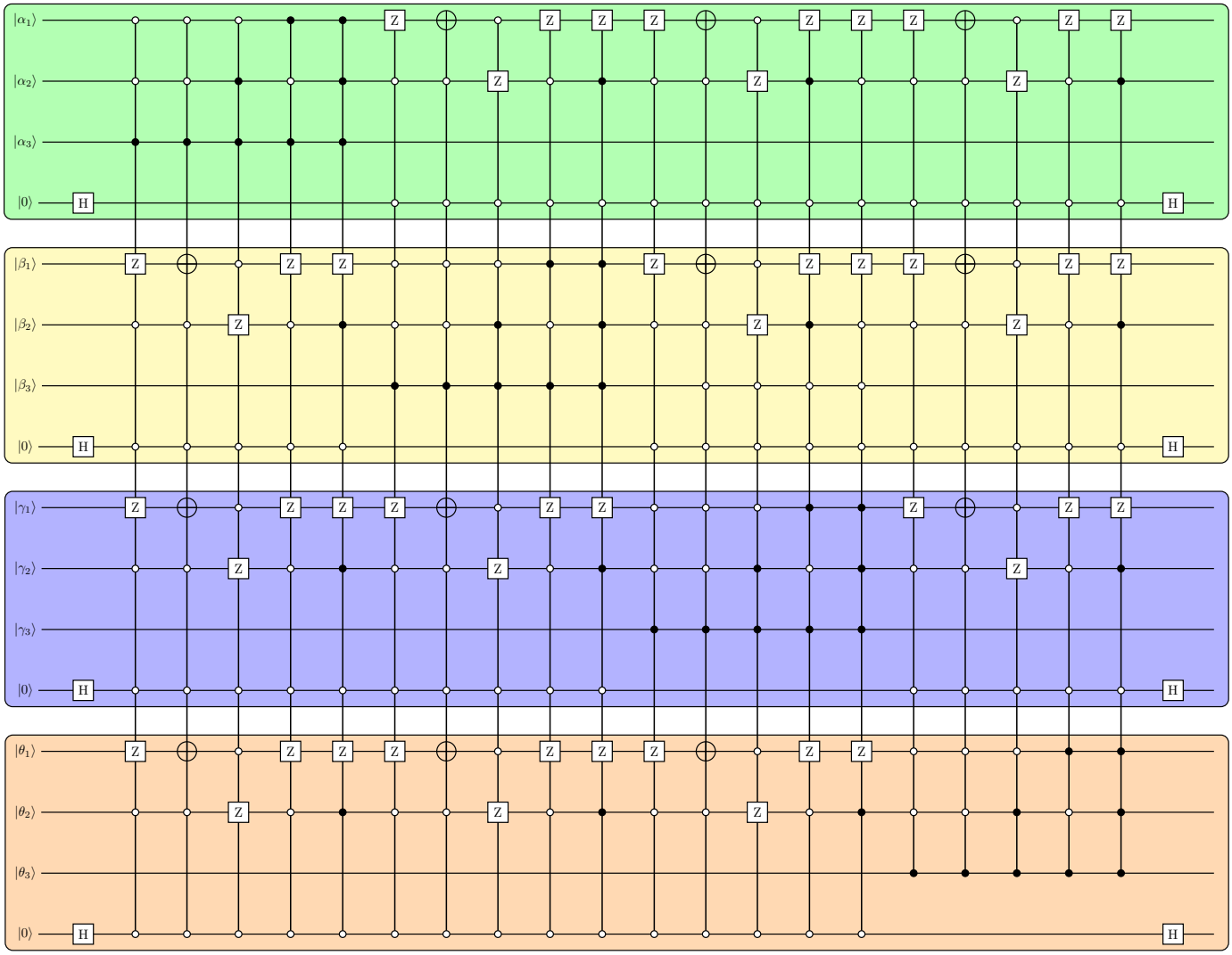

FIG. 5. **Quantum circuit for 2×2 block.** The four cells in 2×2 block are represented by $|\alpha_i\rangle$, $|\beta_i\rangle$, $|\gamma_i\rangle$, $|\theta_i\rangle$, where $0 \leq i \leq 3$ qubits represent different cells in a 2×2 block. We input according to I and the third qubits ($|\alpha_3\rangle$, $|\beta_3\rangle$, $|\gamma_3\rangle$, and $|\theta_3\rangle$) remains in state $|1\rangle$ if the corresponding cell is a clue. If the third qubit is a clue, it triggers deletion of the clue from other qubits according to the constraint, although for our circuit, deletions in only the 2×2 block are applicable. If three deletions occur in a particular block, it will serve as a clue in the next iteration thereby solving the entire Sudoku in further steps.

[1] J.-P. Delahaye, Sci. Am. **294**, 80-87 (2006).
[2] A. Hedayat and E. Seiden, Ann. Math. Stat. **41**, 2035-2044 (1970).
[3] A. E. Brouwer, Nieuw Archief voor Wiskunde **5/7**(4), 258-259 (2006).

[4] The Pennsylvania State University. URL: https://onlinecourses.science.psu.edu/stat503/node/21/
[5] G. Kendall, A. Parkes, and K. Spoerer, ICGA J. (2008).
[6] S. Aaronson, ACM SIGACT News (2005).
[7] T. Davis, The Mathematics of Sudoku (2010).
[8] Glossary of Sudoku (Wikipedia).
[9] F. S. Khan, N. Solmeyer, R. Balu, and T. Humble, arXiv:1803.07919.
[10] E. C. Chi and K. Lange, arXiv:1203.2295.
[11] S. K. Jones, Recreat. Math. Mag. **3**, 87-93 (2016).
[12] J. R. Minkel, Sci. Am. (2007).
[13] A. Eiben, P.-E. Raue, and Z. Ruttkay. (1995b). "GA-Easy and GA-Hard Constraint Satisfaction Problems." In M. Meyer (ed.), Proceedings of the ECAI-94 Workshop on Constraint Processing, number 923 in Lecture Notes in Computer Science. Springer-Verlag, pp. 267–284.
[14] R. Pelanek, Human problem solving: Sudoku case study. Technical Report FIMURS-2011-01, Masaryk University Brno, 2011.
[15] G. L. Long and Y. Liu, Front. Comput. Sci. China, **2**, 167–178 (2008).
[16] L. Yang, Chin. Sci. Bull. **58**, 24 (2013).
[17] L. Long, Commun. Theor. Phys. **45**, 825-844 (2006).
[18] L. K. Grover, in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* (ACM, New York, 1996), pp. 212–219.
[19] Constraint Satisfaction Problem (Wikipedia).

SUPPLIMENTAL MATERIAL

The Python code for detecting the location of new clues is presented below.

```
import numpy as np
a=np.zeros((4,4))
print "input no. of constraints"
i =input()
k=0
p=0
m=0
while(k<i):
  p=input("input row: ")
  m=input("input column: ")
  p=p-1
  m=m-1
  a[p][m] = input("input element: ")
  k=k+1
i=0
k=0
e=np.ones((4,1))
e=e*5
b=0
print "Indices of new clues are: "
while( i<4):

  k=0
  while(k<4):

    e=np.ones((4,1))
    e=e*5
    b=0
    p=0
    if(a[i][k] == 0):
      if((i+k)%2==0):
        if(i%2 == 1):
          while(b<4):

            if(a[i-1][k-1]==b+1):
              a[i][k]+=e[b]
              e[b]=0
            b=b+1

          b=0
        else:
```

```
            while(b<4):

               if(a[i+1][k+1]==b+1):
                  a[i][k]+=e[b]
                  e[b]=0
               b=b+1
            b=0
         else:
            if(i%2 == 1):
               while(b<4):

                  if(a[i-1][k+1]==b+1):
                     a[i][k]+=e[b]
                     e[b]=0
                  b=b+1
               b=0
            else:
               while(b<4):
                  if(a[i+1][k-1]==b+1):
                     a[i][k]+=e[b]
                     e[b]=0
                  b=b+1
               b=0
         p=0
         b=0
         while(b<4):
            p=0
            while(p<4):

               if(a[b][k]==p+1):
                  a[i][k]+=e[p]
                  e[p]=0
               p=p+1
            b=b+1
         b=0
         p=0
         while(b<4):
            p=0
            while(p<4):

               if(a[i][b]==p+1):
                  a[i][k]+=e[p]
                  e[p]=0
               p=p+1
            b=b+1
         if (a[i][k] == 15):
            print i+1,k+1

      k=k+1
   i=i+1
```

q.py

View publication stats