A Letters Journal Exploring the Frontiers of Physics

November 2019

EPL, **128** (2019) 40007
doi: 10.1209/0295-5075/128/40007

www.epljournal.org

Solving Sudoku game using a hybrid classical-quantum algorithm

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*

received 21 February 2019; accepted in final form 1 December 2019
published online 31 January 2020

PACS 03.67.-a – Quantum information
PACS 03.67.Ac – Quantum algorithms, protocols, and simulations

Abstract – Sudoku is a fun combinatorial game, based on the Latin square, which has wide applications as being an efficient design in controlling multiple sources of variable nuisance simultaneously. Quantum Sudoku solver uses basics of quantum mechanics such as superposition and entanglement. Using the concept of duality quantum computing, we propose an algorithm which solves a 4×4 Sudoku puzzle with a space complexity $[O(N^2(4 + \log(N))]$ (where $N = 4$). Our proposed algorithm has a primary aim to provide a theoretical framework for an application of duality quantum computing.

Introduction. – Sudoku is a logic-based, combinatorial number placement puzzle, known to cause brain stimulation and relaxation [1]. The history of Sudoku most likely stems from the mathematical concept of Latin squares [2]. In the 1780s, a Swiss mathematician, Leonhard Euler, put forward the idea of arranging a given set of numbers in a grid such that any given number (or symbol) occurs just once in each row and column. Today, Latin squares are extensively used in statistical analysis [3]. Howard Garns, an Indianapolis architect, by adding another following constraint to the aforementioned, derived Sudoku from Latin squares. The constraint is "Every main 3×3 block must have every given number occurring just once". By "main", it refers to 9 mutually exclusive grids such that no two grids contain a single common cell [4]. For over 25 years, according to Dell magazines, the puzzle was known under the name of "Number Place".

Sudoku is a member of an important class of constraint satisfaction problems (CSP) [5]. The Sudoku puzzle on $N^2 \times N^2$ grid of $N \times N$ blocks is an NP-complete problem [6,7]. Algorithms such as backtracking, SAT-based solver (using enumeration of solutions [8]), quantum annealing [9,10] can solve most of the 9×9 puzzles. However, for a large value of N, a combinatorial explosion occurs arresting the number of the Sudoku that can be constructed and solved. In this paper, our focus is to solve a proper 4×4 Sudoku (unique solution Sudoku with square main blocks) which can be generalized to higher-order proper Sudoku. Sudoku is closely associated with permutation groups in group theory [11,12]. Over time, many variants of Sudoku have been introduced [13]. Research works are also carried out in developing quantum games using quantum strategies [14]. Efforts have also been made to incorporate Sudoku in quantum cryptography for generation of quantum key distribution using a Sudoku variant of the BB84 protocol [15]. Although Sudoku solvers using classical algorithms existed for finite small grid Sudoku's, the first "commercial" quantum chip for solving finite Sudoku was proposed by D-Wave [16], where they used quantum annealing to solve it.

There are two main challenges to this problem. One is the logical complexity of individual steps involved in solving the problem and the second is the structural dependence amongst individual steps, *i.e.*, whether independent (applied in parallel) or dependent (applied sequentially) steps [17]. One is yet to come up with a quantum algorithm which can deterministically achieve this feat for any $N \times N$ Sudoku.

Quantum computing. – Quantum computing, in contrast to classical computing, uses qubits $|0\rangle$ and $|1\rangle$, which display properties like superposition and entanglement. They provide quantum parallelism to speed up the processing speed. In quantum computing, 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. In our algorithm, we have used duality mode on quantum computer to simulate duality quantum computing along with a Python program to efficiently solve the Sudoku game.

Duality computing and duality mode. – 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. The duality computer endows us to perform summation of unitary operations apart from known unitary operations in a quantum computer. Duality computing has improved Shor's algorithm to prime factorize large numbers [18]. Further, the fixed-point quantum search proposed by Mizel [19] can effectively be implemented in a duality quantum computer [20]. Duality quantum computing has also been applied to simulate a generalized anti-PT-symmetric two-level system [21]. In our algorithm, once we know that a particular larger superposition has to be reduced to a smaller superposition, deleting a marked state (a state from a superposition) from an arbitrary set of basis can be done by simulating a duality computer [22] on a quantum computer. This can be made by simulating it on a quantum computer via duality mode and recycling quantum computing. Unitary operations can then be performed on each of these obtained superposition states. We use a $(n+1)$-qubit quantum computer to simulate a n-qubit duality computer via duality mode. Using the principle of relativity, instead of a moving quantum computer through stationary slits, we pass the stationary quantum computer through moving slits. This helps in simulating duality mode on a quantum computer. The n qubits make the stationary quantum computer and the auxiliary qubit is used to make slits (2 in our case). Depending upon the type of problem under consideration, one can use more slits. One just needs to use more auxiliary qubits to simulate more slits. Two additional gates other than the already known quantum gates, quantum wave divider (QWD) and quantum wave combiner (QWC), are used in our algorithm to simulate an n-qubit 2-slit duality mode (fig. 1). We make the following correspondence between the duality computer and the duality mode simulated on a quantum computer:

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

where $|k_u\rangle$ ($|k_d\rangle$) is the center of mass for the translational motion of upper (lower) subwave function [23]. When the auxiliary qubit is in the $|0\rangle$ ($|1\rangle$) state, it resembles a duality computer subwave from the upper (lower) slit. Initial and final wave functions of the duality computer are ascribed to the auxiliary qubit being in the $|0\rangle$ state. Thus, the initial state of the duality computer is $|\phi\rangle|0\rangle$. We perform the QWD [24], by using a Hadamard gate, to switch on the duality mode, and the state becomes $|\phi\rangle\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, namely, the QWD operation is equivalent to a Walsh-Hadamard operation on the auxiliary qubit. After conditional gate operations on different slits, the state takes the following form:

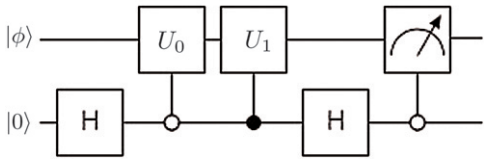

Fig. 1: Representation of a 2-slit duality mode on a quantum computer. Duality mode is simulated using the product space of element state $|\phi\rangle$ and auxiliary qubit $|0\rangle$.

$$\frac{U_0|\phi\rangle|0\rangle + U_1|\phi\rangle|1\rangle}{\sqrt{2}}. \tag{2}$$

The QWC [25] operation can be simulated by a Walsh-Hadamard operation on auxiliary qubit to switch off the duality mode. After QWC, the wave function becomes

$$\frac{U_0+U_1}{2}|\phi\rangle|0\rangle + \frac{U_0-U_1}{2}|\phi\rangle|1\rangle.$$

Measurement is performed on the n qubits on the condition that the auxiliary qubit is in $|0\rangle$ state. Then the wave function is collapsed and $(U_0+U_1)|\phi\rangle$ result is read out. The probability of obtaining a result is $P_0 = \frac{\langle\phi|(U_0+U_1)^\dagger(U_0+U_1)|\phi\rangle}{4}$.

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

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

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. This calculating process recycles in loop until the conditional measurement is performed to obtain a result. This process is known as the recycling quantum computing mode.

Sudoku: rules and outline of the algorithm. – Sudoku is played under the rule that every given number from the set of four states (for 4×4 Sudoku) must appear:

- exactly once in every row;
- exactly once in every column;
- exactly once in the main 2×2 block;

where the earlier definition of "main" stands true. We develop a solution using the given constraints (initially filled cells) in an otherwise empty Sudoku. First, we put every empty cell in a superposition of all the four states (using 2 qubits). Next, we reduce the superposition of

	A		B		C		D	
1	1	2	5	6	17	18	21	22
	3	4	7	8	19	20	23	24
2	9	10	13	14	25	26	29	30
	11	12	15	16	27	28	31	32
3	33	34	37	38	49	50	53	54
	35	36	39	40	51	52	55	56
4	41	42	45	46	57	58	61	62
	43	44	47	48	59	60	63	64

Fig. 2: Representation of a 4×4 Sudoku board. Each cell must have $\log_2 4$ qubits to represent the 4 states in binary (table 1) and 4 additional qubits: 1 to mark the clues and 3 as auxiliary qubits, respectively. Each cell requires $(2 + 4)$ qubits. For a 4×4 Sudoku having 16 cells, a total of 96 qubits $(16 * 6)$ are used. Each uniquely coloured square represents a 2×2 block and within it, 4 qubits represent one cell.

Table 1: Quantum states and corresponding elements.

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

	4		
			3
2			
		1	

	4	*	
*			**3**
2			*
	*	**1**	

Fig. 3: An unsolved Sudoku (left) and marking (by $*$) of cells containing clues (right).

4 states to 3, 2 or 1 with the help of constraints given. For reducing the superposition, we incorporate "deleting a marked state algorithm" using duality mode on a quantum computer. To mark the cells that have been solved, we use a classical Python program.

Detailed algorithm with quantum circuit. – We take 96 qubits to represent our Sudoku board. Six qubits are assigned to every cell, amongst which the first two qubits associated with a cell represent the possible elements of the cell (see fig. 2). For Sudoku, we need superposition of four elements that can be prepared by using two qubits. Each element is assigned to the corresponding quantum state as shown in table 1.

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

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. The fourth, fifth and sixth qubits are used as auxiliary qubits for deletion of a marked state from a superposition of possible states. The first six qubits represent the first cell in the first 2×2 main block, the next six represent the second cell in the same 2×2 main block, and so on as given in fig. 2. 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 the $|1\rangle$ state. One such unsolved Sudoku is shown in fig. 3. All the cells with no initial constraints (clues) must have the first two qubits in $|+\rangle$ states and the third qubit in $|0\rangle$ state. The three auxiliary qubits (fourth, fifth and sixth qubits) of all cells are initially assigned to the $|0\rangle$ state.

In accordance with our algorithm, the circuit checks cell by cell whether it is a given constraint. The cells are "marked" by the Python code, thereafter which turns out to be a clue in the next step, as shown in fig. 3. We check for the third qubit of every cell to be in the $|1\rangle$ 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 main blocks [13] according to the clue as shown in fig. 4.

For the deletion process, we use the above-mentioned duality mode. The quantum circuit used for deletion is given in fig. 5.

The auxiliary qubits are put in $|+\rangle$ state by using a Hadamard gate, which is analogous to a quantum wave being split into two subwaves (the Hadamard gate acts as the QWD here). The $|0\rangle$ state represents one subwave and $|1\rangle$ state represents the other. The next step is to mark the state to be deleted by adding π phase to it in one of the subwaves (in our case the $|0\rangle$ state subwave). The states are marked in correspondence with the clue (table 1). We then apply another Hadamard gate to simulate the combination of two subwaves (QWC), thereby, 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 a deletion is performed twice, we get back the original state. Therefore

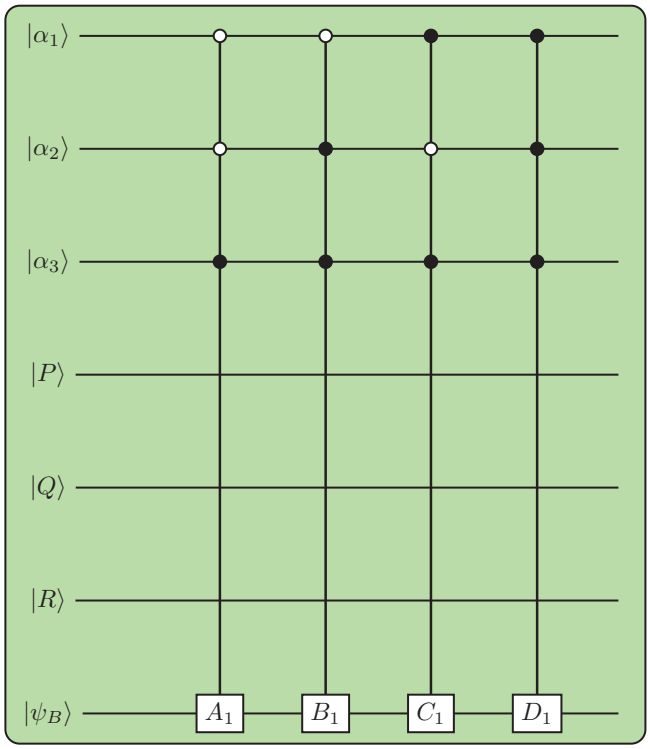

Fig. 4: Quantum circuit illustrating the deletion of states. Here, $|\alpha_1\rangle$ and $|\alpha_2\rangle$ represent the corresponding quantum states of a cell (see table 1), and $|\alpha_3\rangle$ represents whether the cell has a clue or not and it only triggers deletion process if $|\alpha_3\rangle$ has been marked as a clue. From the state $|\psi_B\rangle$ (the current superposed state of the cell), the operations A_1, B_1, C_1, and D_1 are used to delete the states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, respectively, from $|\psi_B\rangle$ (see fig. 5). $|P\rangle$, $|Q\rangle$ and $|R\rangle$ are auxiliary qubits.

3 auxiliary qubits are required for each cell: one each for 2×2 block, row, and column of the board. Since there cannot be more than one clue in the same row, column or the 2×2 block, deletion only takes place once. If all the measurement results are $|0\rangle$, then we move forward with our algorithm, and if at least one of them is $|1\rangle$, then we know that it is not a desired result, so we discard the result and repeat the process. After the completion of this process, we get some new clues due to deletion of three of the four entangled 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 the $|1\rangle$ state. Based on the cell index, the clue in the cell can be found by simply measuring the two qubits pertaining to the cell 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 occur, resulting in new clues. After this iterative process the 4×4 Sudoku is solved.

Circuit for a main 2×2 block. – In fig. 6, we provide the quantum circuit for the 2×2 block. Noticeably, this circuit has a symmetry to it as can be expected from

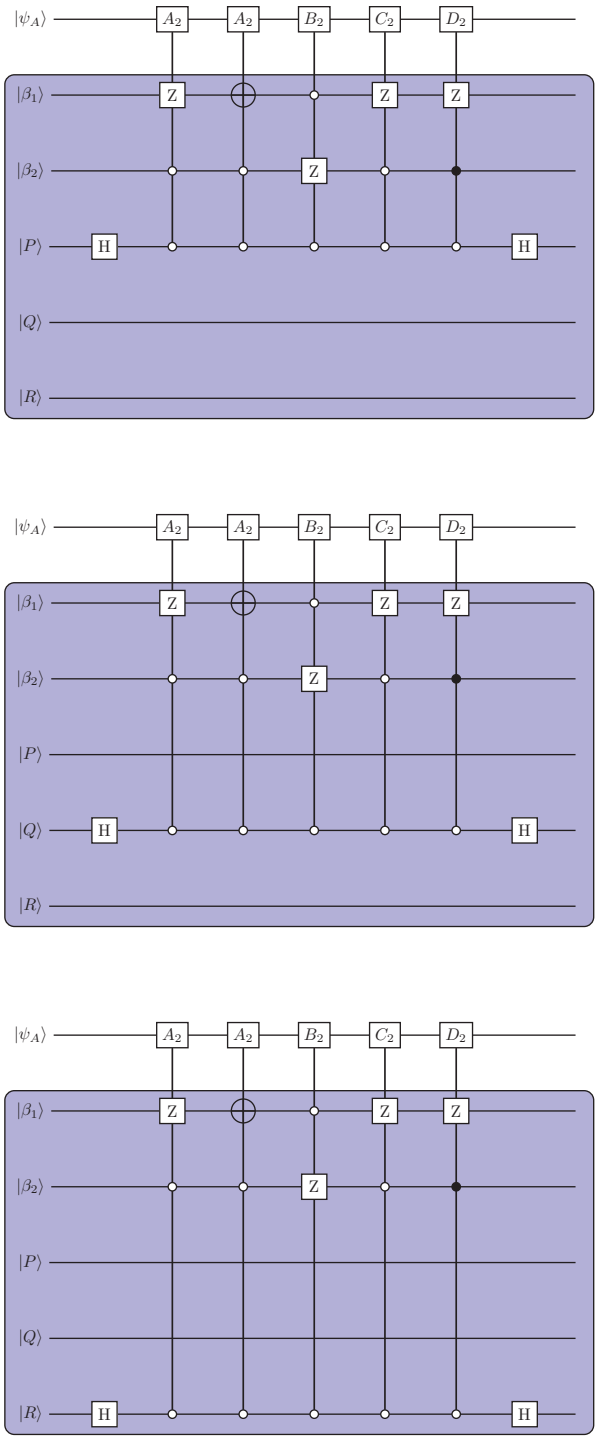

Fig. 5: Quantum circuit depicting deletion algorithm. Here A_2, B_2, C_2 and D_2 are the combination of controls and anti-controls in fig. 4 corresponding to A_1, B_1, C_1, and D_1, respectively. A_2, B_2, C_2, or D_2 is on when $|\psi_A\rangle$ is in the $|001\rangle$ (anti-control, anti-control, and control), $|011\rangle$ (anti-control, control, and control), $|101\rangle$ (control, anti-control, and control), or $|111\rangle$ (control, control, and control) state, respectively. Gates are shown operating on each of the entangled auxiliary qubits, $|P\rangle$, $|Q\rangle$ and $|R\rangle$, when deletion is triggered from the 2×2 block, the same row and the same column, respectively. It is to be noted that, $|\psi_A\rangle$ here is a three-qubit state and hence A_2, B_2, C_2, and D_2 are three-qubit operations.

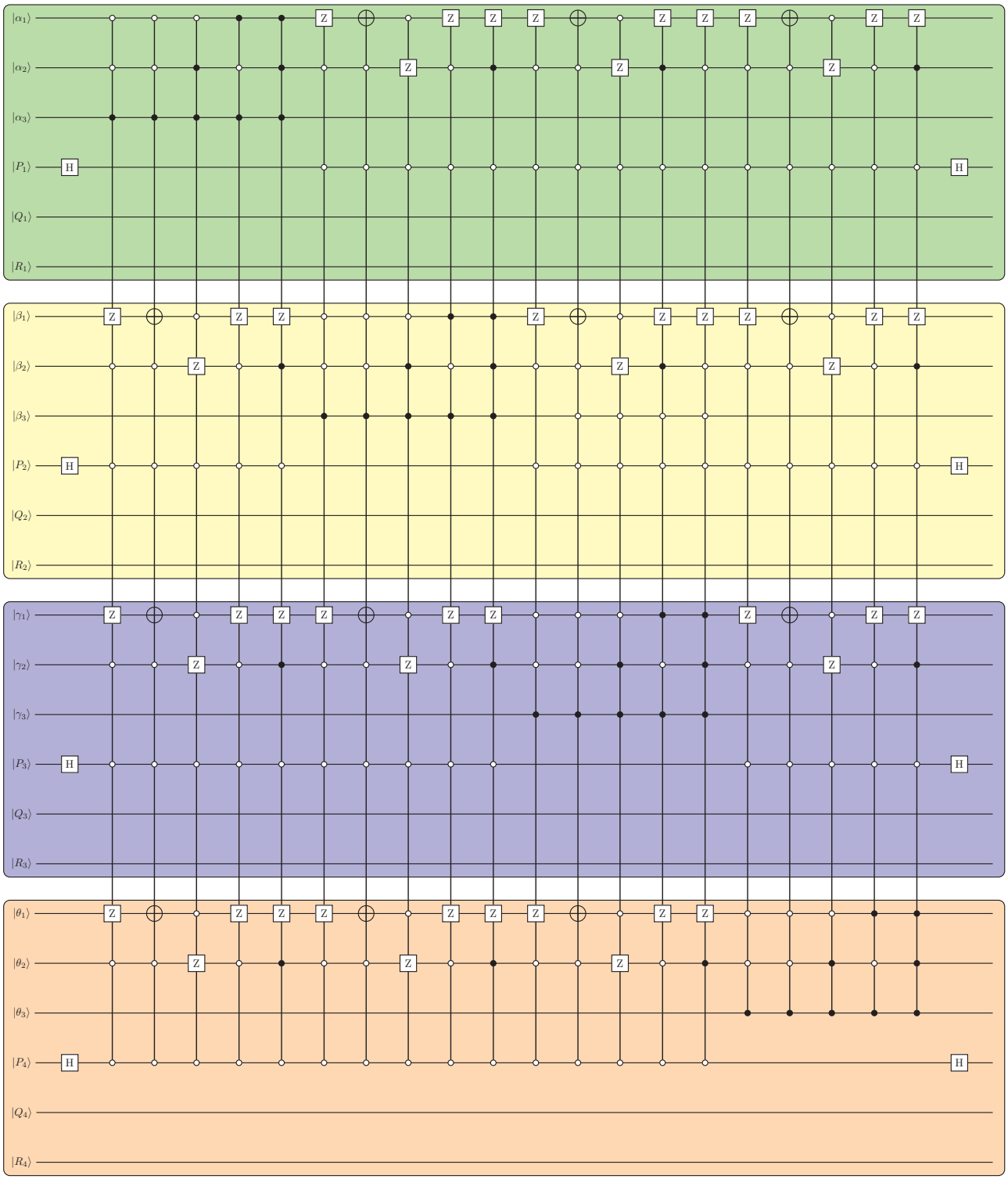

Fig. 6: Quantum circuit for 2×2 block. The four cells in the 2×2 block are represented by $|\alpha_i\rangle$, $|\beta_i\rangle$, $|\gamma_i\rangle$, $|\theta_i\rangle$, where $0 \leq i \leq 3$ qubits represent different cells in a 2×2 block. We input according to table 1 and the third qubit ($|\alpha_3\rangle$, $|\beta_3\rangle$, $|\gamma_3\rangle$, 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 serves as a clue in the next iteration thereby solving the entire Sudoku in further steps. Since all the deletion are performed in 2×2 blocks, only $|P\rangle$ auxiliary qubits are used. The clues in the remaining cells of the same row or column trigger deletion using $|Q\rangle$ and $|R\rangle$ auxiliary qubits.

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 the first iteration, we need to feed input to the circuit (fig. 6). If one of the cells in the 2×2 block is a clue, we input qubits corresponding to that cell accordingly (table 1). Also, the third qubit of every cell, namely, $|\alpha_3\rangle$, $|\beta_3\rangle$, $|\gamma_3\rangle$, and $|\theta_3\rangle$, is in state $|1\rangle$ if the corresponding cell is a clue. However, for empty cells, two qubits representing the state of elements remain in the $|+\rangle$ state, and the third qubit stays in the $|0\rangle$ state. For demonstration, let us take an example where the first cell occupies a state representing 1 and the third cell representing 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 the second, third, and fourth cell given that the 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 the first, second and fourth cell given that the twelfth qubit remains as $|0\rangle$. In the case of the whole quantum circuit, deletions from cells in band and stack (collection of 3 main blocks along a row and column, respectively) of the clue's cell take place. This results in the deletion of three clues in some of the cells, making it a new clue. The indices of these new clues could be found out classically (using the 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 the next set of clues. For the second, and further iterations the input can be made accordingly.

Python program to find further clues. – So far, we have incorporated classical and quantum computing conjointly in our algorithm. And our quantum circuit isolates the elements, which violates given constraints. However, we are not able to gather any information about the cells containing the collapsed state without measuring them. We cannot measure them because we need the superposition of states to proceed to the next step. To achieve this, we use a Python program which marks the cells, by looking out for the third qubit for every cell ($4n+3$ qubit where n is an integer from 1 to 16), which would collapse to a specific number (new clue) in the next step of solving the Sudoku.

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 [26]. Our algorithm does give the desired output deterministically. For each deletion, there is $(k-1)/k$ chance, credits to recycling duality mode, of getting the desired superposition state as output; where k represents the number of superposed states in the input state $|\psi\rangle$. The proposed algorithm works with 64 qubits (16×4). 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. It must be emphasized that the space complexity arises because of circuit design of Sudoku rather than due to duality mode computing.

In comparison to the method used by D-Wave, we have tried to give order of complexity. They have used quantum annealing while we use duality quantum computing in solving the Sudoku puzzle. Our central idea of using duality quantum computing to solve Sudoku can be extended to a larger Sudoku. Furthermore, this protocol can be applied to other CSP problems such as timetabling, scheduling, etc. [27]. Various sections of our algorithm use identification, targeting, marking and elimination which can have varied applications independently. The space complexity of our algorithm is $n^2(\log_2 n + 4)$. Each cell must have $\log_2 n$ qubits to represent n states in binary and 4 additional qubits: 1 to mark the clues and 3 as auxiliary qubits, respectively. However, it should be noted that this cannot be applied to all $n \times n$ Sudoku as the rules of the game itself change.

Here in the present work, we have solved a 4×4 Sudoku. Generalizing to 9×9 and higher-order Sudoku's requires an additional simple modification. For a 4×4 Sudoku, one can always find a new constraint after one iteration otherwise uniqueness of the solution fails. This may not hold for higher-order Sudoku's. However, in 9×9 or higher Sudoku's, there may come a stage after some iterations where a smaller cyclic entanglement might occur and the quantum circuit cannot reduce further superposition by itself. In such cases, the following modification aids in reducing smaller entanglements to find the next clue. Once we reach a state when the Sudoku cannot be solved further, we ask our program to collapse the cyclic entanglement to a particular state, which may not give the right answer. But since we know which cells have undergone smaller cyclic entanglement, we can put the system back in superposition after its 1st collapse (which did not result in a solution).

To put back the state in the same superposition, we need the full spectrum of eigenvalues it was in before cyclic entangled states collapsed. This data can be collected from all the different cell states which have recently collapsed. It may appear that this might increase the number of superposed states but our circuit does not allow that. After every collapse, one needs to check whether the Sudoku has been solved to which row and column checks from N-Queens Solver problem [28] can be applied for every number. The above method applied iteratively gives a solution for higher-order Sudoku's. To conclude, we have used duality quantum computing and classical computing to solve a Sudoku problem that involved marking of states and deletion of states. We have proposed new quantum circuits for the same. Our proposed algorithm provides an exponential speedup with a logarithmic complexity [$O(\log(k))$ [23].

* * *

AP, SC and VM acknowledge the hospitality provided by IISER Kolkata during the completion of the project work. BKB acknowledges the financial support provided by IISER-K Institute fellowship.

REFERENCES

[1] Delahaye J.-P., *Sci. Am.*, **294** (2006) 80.
[2] Hedayat A. and Seiden E., *Ann. Math. Stat.*, **41** (1970) 2035.
[3] The Pennsylvania State University, URL: `https://onlinecourses.science.psu.edu/stat503/node/21/`.
[4] Brouwer A. E., *Nieuw Arch. Wiskd.* **5/7** (2006) 258.
[5] Eiben A., Raue P.-E. and Ruttkay Z., *GA-Easy and GA-Hard Constraint Satisfaction Problems*, in *Proceedings of the ECAI-94 Workshop on Constraint Processing*, edited by Meyer M., *Lecture Notes in Computer Science*, Vol. **923** (Springer-Verlag) 1995, pp. 267–284.
[6] Kendall G., Parkes A. and Spoerer K., *ICGA J.*, **31** (2008) 13.
[7] Aaronson S., *ACM SIGACT News*, **36** (2005) 30.
[8] Felgenhauer B. and Jarvis F., *Enumerating Possible Sudoku Grids* (2005).
[9] Davis T., *The Mathematics of Sudoku* (2010).
[10] Chi E. C. and Lange K., arXiv:1203.2295 (2012).
[11] Felgenhauer B. and Jarvis F., *Mathematics of Sudoku I* (2006).
[12] Russell E. and Jarvis F., *Mathematics of Sudoku II* (2006).
[13] Glossary of Sudoku (Wikipedia).
[14] Khan F. S., Solmeyer N., Balu R. and Humble T., *Quantum Inf. Process.*, **17** (2018) 309.
[15] Jones S. K., *Recreat. Math. Mag.*, **3** (2016) 87.
[16] Minkel J. R., *Sci. Am.* (13 February 2007).
[17] Pelanek R., *Human Problem Solving: Sudoku Case Study*, Technical Report FIMURS-2011-01 (Masaryk University Brno) 2011.
[18] Wan-Ying W., Bin S., Chuan W. and Gui-Lu L., *Commun. Theor. Phys.*, **47** (2007) 471.
[19] Mizel A., *Phys. Rev. Lett.*, **102** (2009) 150501.
[20] Ding L. and Zhou T., *EPL*, **126** (2019) 20004.
[21] Zheng C., *EPL*, **126** (2019) 30005.
[22] Long G. L. and Liu Y., *Front. Comput. Sci. China*, **2** (2008) 167.
[23] Yang L., *Chin. Sci. Bull.*, **58** (2013) 24.
[24] Long G. L., *Commun. Theor. Phys.*, **45** (2006) 825.
[25] Long G.-L., Liu Y. and Wang C., *Commun. Theor. Phys.*, **51** (2009) 65.
[26] *Grover L. K.* in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* (ACM, New York) 1996, pp. 212–219.
[27] Constraint Satisfaction Problem (Wikipedia).
[28] Jha R., Das D., Dash A., Jayaraman S., Behera B. K. and Panigrahi P. K., arXiv:1806.10221 (2018).