Quantum Computing Approaches for Sudoku and Graph Coloring

Anooshka Pendyal

January 3rd, 2023

1 Introduction

Sudoku is a common number-based logic puzzle where players must fill in a 9x9 grid with digits so that no digit can appear multiple times in a row, column, or 3x3 subsquare. Sudoku puzzles can be solved using combinatorial optimization techniques [1], which has garnered interest in developing mathematical expressions that represent the requirements of the puzzle in order to solve the Sudoku puzzle. Furthermore, Sudoku puzzles can be represented as graph coloring problems, where the goal is to create a 9-coloring of a graph when already given a partial 9-coloring [2].

Graph coloring is a subset of graph theory, where the aim is to assign labels to elements of a graph, given certain constraints [3]. Graph coloring algorithms have many relevant applications in modern computer science such as data mining, image segmentation, clustering, and image capturing [4]. Thus, developing new methods of solving Sudoku will advance graph coloring algorithms and their related fields like artificial intelligence, which requires large amounts of data and processing of that data to train neural networks.

One current way of solving Sudoku is a brute-force approach known as backtracking, where each branch is completely explored before moving to another branch that could lead to the solution. While filling in the empty cells with a digit from available choices, if the algorithm finds that a digit violates the rules of Sudoku, it backtracks and changes the digit of previous cell [5]. Another method of solving Sudoku is simulated annealing, which is a probabilistic optimization method. First, the empty cells are filled with random values and the number of errors in the puzzle are counted. Then, the values of two cells are swapped and the number of errors is recounted. The algorithm will use the updated filled version as a starting point for future iterations if it has less errors than the original filled version [5]. However, there are issues that exist with these algorithms, such as the backtracking algorithm's slow solving time compared to other algorithms that use deductive methods.

Quantum computing and modeling Sudoku as an optimization problem presents itself as an effective way of solving even the most difficult Sudoku problems. The fundamental unit of a quantum computer is a qubit, which can represent the values of 0, 1, or some combination between, in a concept known as superposition, as opposed to a classical bit, which can represent only 0 or 1 [6]. This allows the quantum computer to represent an exponentially larger amount of data with a small number of qubits [6]. There are many ways of physically representing qubits, such as single atoms and single electrons or as whole systems like in complex superconducting electrical circuits [7]. Logical qubits, in contrast, are emulated more robust and stable qubits [6]. Quantum optimization harnesses the advantages of quantum computing while also using the powerful global search ability of intelligent optimization algorithms [8]. Optimization problems are often formulated as minimization problems, where the optimal solution has the smallest error and the goal is to obtain the smallest possible error.

Quantum unconstrained binary optimization (QUBO) problems are mathematical formulations that can be used to represent a wide variety of problems in the field of combinatorial optimization [9]. Combinatorial optimization problems focus on finding an optimal solution based on a large number of decisions that yield a corresponding objective function value, such as a cost [9], and have a wide range of applications from artificial intelligence to finance [10]. QUBO problems are NP-hard [10] and due to their relatedness to Ising problems, can be used to represent problems solved through quantum annealing [9]. Quantum annealing is the process of using quantum fluctuations to solve optimization problems. The properties of quantum mechanics can be harnessed to solve multivariable optimization problems framed as energy minimization problems [11].

The maximum cut problem is a well-known example of a combinatorial optimization problem [9]. Given a graph with vertices connected by edges, the goal of the max cut problem is to use one continuous cut to create as many partitions between the edges as possible [12]. This problem has applications in fields like statistical physics [12].

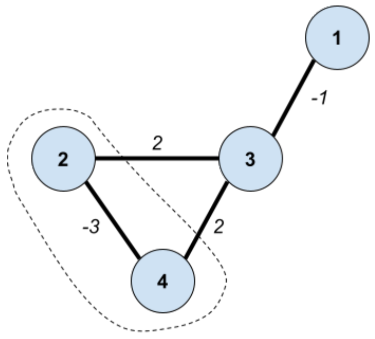

Figure 1: The maximum cut of a graph with weighted edges.

The above problem can be modeled using a QUBO formulation. First, the formulation to maximize the number of edges in the cut can be represented as

Maximize $y = \sum_{(i,j)\in E} (x_i + x_j - 2x_i x_j)$. Thus, the above graph can be formulated as *Maximize* $y = -1(x_1+x_3-2x_1x_3)+2(x_2+x_3-2x_2x_3)-3(x_2+x_4-2x_2x_4)+2(x_3+x_4-2x_3x_4)$ while accounting for the weights. Further simplifying the above formulation yields *Maximize* $y = -x_1 - x_2 + 3x_3 - x_4 + 2x_1x_3 - 4x_2x_3 + 6x_2x_4 - 4x_3x_4$. Finally, the formulation can be represented as a QUBO matrix.

$$\begin{bmatrix} -1 & 0 & 2 & 0 \\ 0 & -1 & -4 & 6 \\ 0 & 0 & 3 & -4 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

2 Formating a Sudoku Puzzle into a Quadratic Unconstrained Binary Optimization (QUBO) Matrix

2.1 Constraint Definitions

In Sudoku, there are a number of requirements that must be fulfilled in order to consider the puzzle as complete.

From these requirements, constraints can be derived in order to optimize the solution to the problem, which is solving the Sudoku puzzle. Let i represent the row, j represent the column, and k represent the digit of the hint.

1. Each cell can contain only a single number.

$$\sum_{k=1}^{9} X_{ijk} = 1, \forall ij \in cell$$

A cell is each individual box in the puzzle. For example, in a traditional 9x9 Sudoku puzzle, there are a total of 81 cells. The purpose of the above equation is to represent how there can only be one k value, or hint, for each cell, and all other expressions representing the other k values should be 0 because the cell cannot contain more than one digit.

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

In the above example, there is the digit 8 in row 2 and column 3. In the context of constraint 1, this digit would be represented as $X_{238} = 1$ because in row 2 and column 3, there exists the digit 8. However, $X_{231} = 0$ because the digit 1 is not in row 2 and column 3. From this, by taking the sum of the above expression with all possible values of k from 1-9, the sum should be zero because there can only be one digit in each cell.

2. Each row in the whole puzzle must contain numbers 1-9 only once.

$$s.t. \sum_j X_{ijk} = 1, \forall i \in row, \forall k \in K (K = \{1...9\})$$

   A row represents a horizontal line of cells in the puzzle.

3. Each column in the whole puzzle must contain numbers 1-9 only once.

$$s.t. \sum_i X_{ijk} = 1, \forall j \in column, \forall k \in K (K = \{1...9\})$$

   A column represents a vertical line of cells in the puzzle.

4. Each 3x3 subsquare in the whole puzzle must contain numbers 1-9 only once.

$$s.t. \sum_j^2 \sum_i^2 x_{(i+v)(j+v)k} = 1, \forall k \in K (K = \{1...9\})$$

$$u, v \in \{0, 3, 6\} : \textit{offset to each grid}$$

   A subsquare represents a smaller set of cells in the overall Sudoku puzzle. For example, in a 9x9 Sudoku puzzle, a subsquare is be made of 9 cells, with 3 rows and 3 columns. In a 4x4 puzzle, a subsquare might have 4 cells in 2 rows and 2 columns.

5. Hints cannot be altered.

$$s.t. \sum_{hint} X_{ijk} = 1$$

   A hint represents the starting digits that are already filled out in the Sudoku puzzle.

2.2 QUBO Formulation

Next, the constraints are transformed into a QUBO formulation. With a penalty weight of $\alpha = 2$, the solver will attempt to minimize the formulation in order to reach the optimal solution.

1. Each cell can contain only a single number.

$$\alpha \sum_{ij} (\sum_{k=1}^{9} x_{ijk} - 1)^2$$

2. Each row in the whole puzzle must contain numbers 1-9 only once.

$$\alpha \sum_{k=1}^{9} \sum_{i} (\sum_{j} x_{ijk} - 1)^2$$

3. Each column in the whole puzzle must contain numbers 1-9 only once.

$$\alpha \sum_{k=1}^{9} \sum_{j} (\sum_{i} x_{ijk} - 1)^2$$

4. Each 3x3 subsquare in the whole puzzle must contain numbers 1-9 only once.

$$\alpha \sum_{k=1}^{9} (\sum_{j}^{2} \sum_{i}^{2} x_{(i+v)(j+v)k} - 1)^2$$

$$u, v \in \{0, 3, 6\} : \textit{offset to each grid}$$

5. Hints cannot be altered.

$$2\alpha \sum_{hint} (1 - x_{ijk})$$

The final objective function is set to H, the minimum combinatorial value.

$$H = \alpha \sum_{ij} (\sum_{k=1}^{9} x_{ijk} - 1)^2 + \alpha \sum_{k=1}^{9} \sum_{i} (\sum_{j} x_{ijk} - 1)^2 + \alpha \sum_{k=1}^{9} \sum_{j} (\sum_{i} x_{ijk} - 1)^2 + \alpha \sum_{k=1}^{9} (\sum_{j}^{2} \sum_{i}^{2} x_{(i+v)(j+v)k} - 1)^2 + 2\alpha \sum_{hint} (1 - x_{ijk})$$

2.3 Deriving the QUBO Matrix with a Hand-Calculated Example

An easy 2x2 Sudoku puzzle will be used to illustrate how the QUBO formulation can be applied to an actual puzzle and how the QUBO matrix is obtained. In this example, the formulation has been modified to include values of 1 and 2 and not 1 through 9, as the Sudoku puzzle only has values of 1 and 2. Additionally, constraint 4, which states that each subsquare can contain numbers 1-9 only once, has been removed because subsquares cannot exist in a 2x2 Sudoku puzzle.

1	
2	1

The final objective function for the 2x2 Sudoku puzzle is as follows, with α set to 2.

$$H = \alpha \sum_{ij} (\sum_{k=1}^{2} x_{ijk} - 1)^2 + \alpha \sum_{k=1}^{2} \sum_{i} (\sum_{j} x_{ijk} - 1)^2 + \alpha \sum_{k=1}^{2} \sum_{j} (\sum_{i} x_{ijk} - 1)^2 + 2\alpha \sum_{hint} (1 - x_{ijk})$$

First, substitute the corresponding values for i, j, and k into each constraint, which should be in a QUBO formulation. Constraint 1, which is used to favor solutions where each cell only has one number, will be used as an example.

$$\alpha \sum_{ij} (\sum_{k=1}^{2} x_{ijk} - 1)^2$$

The sum of all possible values for k, which is 1 and 2, is taken at this step.

$$\alpha \sum_{ij} (x_{ij1} + x_{ij2} - 1)^2$$

The formulation is further expanded and constants, such as 1, are disregarded from the formulation.

$$\alpha \sum_{ij} (-x_{ij1}^2 - x_{ij2}^2 + 2x_{ij1}x_{ij2})$$

Finally, the sum of all possible combinations of i and j are taken and the penalty of $\alpha = 2$ can be applied.

$$-2x_{111}^2 - 2x_{112}^2 + 4x_{111}x_{112} - 2x_{121}^2 - 2x_{122}^2 + 4x_{121}x_{122} - 2x_{211}^2 - 2x_{212}^2 + 4x_{211}x_{212} - 2x_{221}^2 - 2x_{222}^2 + 4x_{221}x_{222}$$

After substituting the values for i, j, and k for all 4 constraints in the QUBO formulation, the final formulation for the 2x2 Sudoku will be as follows.

$$-10x_{111}^2 - 6x_{112}^2 - 6x_{121}^2 - 6x_{122}^2 - 6x_{211}^2 - 10x_{212}^2 - 10x_{221}^2 - 6x_{222}^2 + \\ 4x_{111}x_{112} + 4x_{121}x_{122} + 4x_{211}x_{212} + 4x_{221}x_{222} + 4x_{111}x_{211} + 4x_{121}x_{221} + \\ 4x_{112}x_{212} + 4x_{122}x_{222} + 4x_{111}x_{121} + 4x_{211}x_{221} + 4x_{112}x_{122} + 4x_{212}x_{222}$$

The final step is to transform the formulation into the QUBO matrix. This is done by placing the coefficient of each variable into the matrix. The row and column of each coefficient is determined by the i, j, and k values of the variable, as shown below.

- An i value of 1, a j value of 1, and a k value of 1 (1,1,1) corresponds to 1
- (1,1,2) corresponds to 2
- (1,2,1) corresponds to 3
- (1,2,2) corresponds to 4
- (2,1,1) corresponds to 5
- (2,1,2) corresponds to 6
- (2,2,1) corresponds to 7
- (2,2,2) corresponds to 8

The final QUBO matrix can be created in this method.

$$\begin{bmatrix} -10 & 4 & 4 & 0 & 4 & 0 & 0 & 0 \\ 0 & -6 & 0 & 4 & 0 & 4 & 0 & 0 \\ 0 & 0 & -6 & 4 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & -6 & 4 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & -10 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & -10 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6 \end{bmatrix}$$