Quadratic Unconstrained Binary Optimization

Sam Muir

June 2024

1 Introduction

This project will begin by covering how QUBO models are constructed. Then we explore how QUBO models can be applied to the graph matching problem and the related quadratic assignment problem. Finally, we will discuss the application of 'quantum annealers' to solve QUBO problems.

This project will follow ideas from 'Continuous optimization methods for the graph isomorphism problem' by Stefan Klus and Patrick Gelß.

2 What are QUBO problems?

Definition 1 *A QUBO problem is an optimisation problem of form,*

$$\min \; f(\mathbf{x}) = \mathbf{x}^T Q \mathbf{x}$$

where $\mathbf{x}$ *is a vector of* n *binary variables and* Q *is a* $n \times n$ *matrix.*

These are important because many famous optimisation problems can be expressed as QUBO problems.

2.1 Linear Assignment problem

The linear assignment problem can be stated as:

Suppose we have n machines and n tasks which must be completed. We can assign any machine to do any task, but each task-machine pair has an associated cost. We want to allocate one machine to each task such that the sum of the costs is minimised.

If we let A be the matrix of machine-task costs then the problem can expressed as:

$$\min \; g(P) = \operatorname{tr}(PA)$$

where P is a permutation matrix which represents a task allocation.

We model this problem as a QUBO as follows:

First, let $A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$ and $P = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ x_{n+1} & x_{n+2} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n(n-1)+1} & x_{n(n-1)+2} & \dots & x_{n^2} \end{pmatrix}$.

Then the function to be minimized is,

$$\begin{aligned} \operatorname{tr}(PA) &= \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \cdot \begin{pmatrix} a_{11} \\ \vdots \\ a_{n1} \end{pmatrix} + \begin{pmatrix} x_{n+1} \\ \vdots \\ x_{2n} \end{pmatrix} \cdot \begin{pmatrix} a_{12} \\ \vdots \\ a_{n2} \end{pmatrix} + \dots + \begin{pmatrix} x_{n(n-1)+1} \\ \vdots \\ x_{n^2} \end{pmatrix} \cdot \begin{pmatrix} a_{1n} \\ \vdots \\ a_{nn} \end{pmatrix} \\ &= \sum_{i=1}^{n} \begin{pmatrix} x_{n(i-1)+1} \\ \vdots \\ x_{ni} \end{pmatrix} \cdot \begin{pmatrix} a_{1i} \\ \vdots \\ a_{ni} \end{pmatrix} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ji} x_{n(i-1)+j} \end{aligned}$$

But since x_i is binary, $x_i = x_i^2$, and so

$$\begin{aligned} \operatorname{tr}(PA) &= \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ji} x_{n(i-1)+j} \\ &= \mathbf{x}^T \operatorname{diag}(\operatorname{vec}(A)) \mathbf{x}. \end{aligned}$$

So we can now write the problem as

$$\min g(\mathbf{x}) = \mathbf{x}^T \operatorname{diag}(\operatorname{vec}(A)) \mathbf{x} \tag{1}$$

where $\mathbf{x} = \operatorname{vec}(P^T)$ and P is a permutation matrix.

Now we need a penalty function which forces P to be a permutation matrix. To accomplish this we use the equation $C\mathbf{x} = d$ given in [2, p. 8] where

$$C = \begin{pmatrix} \mathbb{1}_n^T & & \\ & \ddots & \\ & & \mathbb{1}_n^T \\ e_1^T & \dots & e_1^T \\ \vdots & \ddots & \vdots \\ e_n^T & \dots & e_n^T \end{pmatrix} \in \mathbb{R}^{2n \times n^2}$$

and $d = \mathbb{1}_{2n}$.

This equation is 0 if and only if P is doubly stochastic. But if P is doubly stochastic it is also a permutation matrix since each entry is binary. So this constraint forces P to be a permutation matrix.

We can transform the constraint $C\mathbf{x} = d$ into the penalty $(C\mathbf{x} - d) \cdot (C\mathbf{x} - d)$. Adding this to (1) will favour solutions which are permutation matrices. Expanding this penalty we get,

$$\begin{aligned}(C\mathbf{x} - d) \cdot (C\mathbf{x} - d) &= C\mathbf{x} \cdot C\mathbf{x} - 2d \cdot C\mathbf{x} + d \cdot d \\ &= \mathbf{x}^T C^T C\mathbf{x} - 2\mathbf{x}^T C^T d + d \cdot d\end{aligned}$$

Then since $\mathbf{x}$ is binary $x_i = x_i^2$ the linear term $-2\mathbf{x}^T C^T d$ can be expressed as the quadratic term $-2\mathbf{x}^T \mathrm{diag}(C^T d)\mathbf{x}$. So the penalty term becomes

$$\begin{aligned}&\mathbf{x}^T C^T C\mathbf{x} - 2\mathbf{x}^T \mathrm{diag}(C^T d)\mathbf{x} + d \cdot d \\ =\,&\mathbf{x}^T (C^T C - 2\mathrm{diag}(C^T d))\mathbf{x} + d \cdot d.\end{aligned}$$

And so we can rewrite problem (1) as

$$\min g(\mathbf{x}) = \mathbf{x}^T (\mathrm{diag}(\mathrm{vec}(A)) + \lambda C^T C - 2\lambda \mathrm{diag}(C^T d))\mathbf{x} + \lambda d \cdot d$$

where λ is the penalty weight. Then to get it in QUBO form we drop the consant to get

$$\min g(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x}$$

where $Q = \mathrm{diag}(\mathrm{vec}(A) - 2\lambda C^T d) + \lambda C^T C$.

2.1.1 Example: Linear Assignment Problem

Consider the cost matrix,

$$A = \begin{pmatrix} 7 & 9 & 1 \\ 4 & 2 & 6 \\ 7 & 8 & 7 \end{pmatrix}$$

where each column corresponds to a task and each row corresponds to the machine costs. Using A and setting $\lambda = 10$ we obtain the matrix

$$Q = \begin{pmatrix} -13 & 10 & 10 & 10 & 0 & 0 & 10 & 0 & 0 \\ 10 & -16 & 10 & 0 & 10 & 0 & 0 & 10 & 0 \\ 10 & 10 & -13 & 0 & 0 & 10 & 0 & 0 & 10 \\ 10 & 0 & 0 & -11 & 10 & 10 & 10 & 0 & 0 \\ 0 & 10 & 0 & 10 & -18 & 10 & 0 & 10 & 0 \\ 0 & 0 & 10 & 10 & 10 & -12 & 0 & 0 & 10 \\ 10 & 0 & 0 & 10 & 0 & 0 & -19 & 10 & 10 \\ 0 & 10 & 0 & 0 & 10 & 0 & 10 & -14 & 10 \\ 0 & 0 & 10 & 0 & 0 & 10 & 10 & 10 & -13 \end{pmatrix}$$

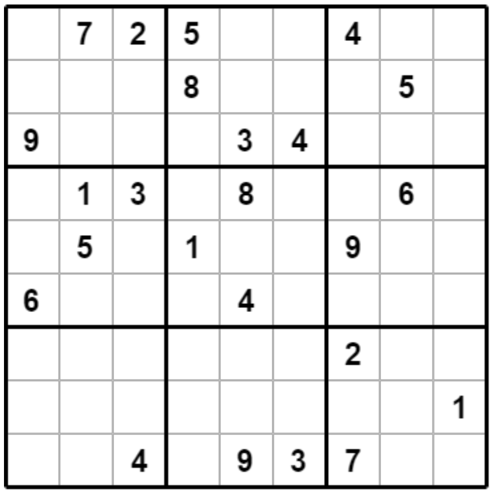
Solving this gives $X = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}^T$ and so we have

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad PA = \begin{pmatrix} 7 & 8 & 7 \\ 4 & 2 & 6 \\ 7 & 9 & 1 \end{pmatrix}$$

which tells us that the minimum cost is $\mathrm{tr}(PA) = 10$.

2.2 Sudoku

In the game of Sudoku we are given a partially filled 9×9 grid (81 squares) and the aim is to populate the remainder of the grid such that every column, row and box contains each of the numbers from 1 to 9. These boxes split the 9×9 grid into 9 non-overlapping 3×3 grids.



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

Figure 1: Medium Sudoku puzzle from New York Times on June 24, 2024

We can formulate the Sudoku problem as follows. Following [3, p. 1] the vector $\mathbf{x}$ will have 9 binary entries for each Sudoku square $x_{i,j,k} = x_{81i+9j+k}$. So each variable $x_{i,j,k}$ is defined as

$$x_{i,j,k} = \begin{cases} 1 & \text{if row } i \text{ column } j \text{ has value } k \\ 0 & \text{otherwise.} \end{cases}$$

The constraints of this problem are:

	Constraint
1	Each column contains 1 to 9
2	Each row contains 1 to 9
3	Each box contains 1 to 9
4	Every square must have a value

Using [1, p. 326] the constraints above become

$$(1) \quad \sum_{i=1}^{9} x_{i,j,k} = 1 \quad \forall j, k \in \{1 \ldots 9\}$$

$$(2) \quad \sum_{j=1}^{9} x_{i,j,k} = 1 \quad \forall i,k \in \{1\dots9\}$$

$$(3) \quad \sum_{j=3q-2}^{3q} \sum_{i=3p-2}^{3p} x_{i,j,k} = 1 \quad \forall k \in \{1\dots9\}, \forall p,q \in \{1\dots3\}$$

$$(4) \quad \sum_{k=1}^{9} x_{i,j,k} = 1 \quad \forall i,j \in \{1\dots9\}$$

where F is the set containing all triples corresponding to the indices of the filled squares.

We can transform these constraints into penalty functions by subtracting 1 and squaring them. Then to obtain a solution to these constraints we minimise the sum of these penalties. This also means that we don't need any penalty weights.

Consider the 1st constraint for some fixed j and k. Expanding the corresponding penalty gives us

$$\begin{aligned}\left(\sum_{i=1}^{9} x_{i,j,k} - 1\right)^2 &= \sum_{i=1}^{9} x_{i,j,k} \sum_{p=1}^{9} x_{p,j,k} - 2\sum_{i=1}^{9} x_{i,j,k} + 1 \\ &= \sum_{i=1}^{9}\sum_{p=1}^{9} x_{i,j,k}x_{p,j,k} - 2\sum_{i=1}^{9} x_{i,j,k}^2 + 1 \\ &= \begin{pmatrix} x_{1,j,k} & \dots & x_{9,j,k}\end{pmatrix} (J_9 - 2I) \begin{pmatrix} x_{1,j,k} \\ \vdots \\ x_{9,j,k}\end{pmatrix} + 1\end{aligned}$$

where J_n is the $n \times n$ matrix of ones.

This we can replace $\begin{pmatrix} x_{1,j,k} & \dots & x_{9,j,k}\end{pmatrix}^T$ with $\mathbf{x}$ if we embed $J_9 - 2I$ in a larger matrix with zeros everywhere but the rows and columns representing $x_{1,j,k} \dots x_{9,j,k}$.

If we do this for every constraint and drop the constant we can add all 324 matrices to get a 729×729 matrix U[1].

Then we can take the squares which have already been filled and turn them into constraints. So the constraint $x_{i,j,k} = 1$ becomes

$$\begin{aligned}(x_{i,j,k} - 1)^2 &= x_{i,j,k}^2 - 2x_{i,j,k} + 1 \\ &= -x_{i,j,k}^2 + 1\end{aligned}$$

[1]This matrix is provided on the GitHub page under the name 'sudokuQ.mat'

which can be interpreted as subtracting 1 from (i, j, k)th element on the diagonal of U. Doing this for every square which has been filled creates the matrix Q.

Finally, solving the QUBO problem

$$\min \mathbf{x}^T Q \mathbf{x}$$

will give us the vector $\mathbf{x}$ which is the solution of the original Sudoku problem.

3 Graph Isomorphism and Matching problems

3.1 The Graph Isomorphism problem

Two graphs G_A and G_B are isomorphic iff there exists a bijection between the vertices of the graphs such that every edge connecting vertices in G_A corresponds to a edge connecting vertices in G_B.
Essentially, isomorphism means each graph can be rearranged and relabelled to look identical to the other.

Look at these two graphs,
First we need to establish some results from [2].

Definition 2 *[2, p. 6] Suppose we have graphs G_A and G_B with adjacency matrices A and B respectively. The doubly stochastic relaxation of the graph isomorphism problem can be formulated as*

$$c_D = \min_{X \in D(n)} ||XA - BX||_F^2$$

where $D(n)$ is the set of doubly stochastic matrices and $||\cdot||_F$ denotes the Frobenius norm.

Lemma 3 *[2, p. 13] If we penalize non-binary matrices, this problem can be written as*

$$\min_{\substack{\mathbf{x} \geq 0 \\ C\mathbf{x} = d \\ H\mathbf{x} = 0}} -\mathbf{x}^T \mathbf{x}$$

with $\mathbf{x} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$ *where $X_1, \ldots X_n$ are the columns of X and C, d, and H are defined as shown on [2, p. 8].*

To obtain a QUBO formulation of this problem we can transform the constraints into penalties as shown below.

	Constraint	Penalty
1	$C\mathbf{x} = d$	$(C\mathbf{x} - d) \cdot (C\mathbf{x} - d)$
2	$H\mathbf{x} = 0$	$H\mathbf{x} \cdot H\mathbf{x}$

These penalties are 0 for functions which satisfy the constraints and positively-valued otherwise. Therefore by scaling each penalty by the weights λ_1 and λ_2 then adding them to the objective function we can ensure that the optimal solution to the QUBO problem corresponds to a isomorphism (if it exists).

Penalty 1 can be simplified to,

$$\begin{aligned}(C\mathbf{x} - d) \cdot (C\mathbf{x} - d) &= C\mathbf{x} \cdot C\mathbf{x} - 2d \cdot C\mathbf{x} - d \cdot d \\ &= \mathbf{x}^T C^T C\mathbf{x} - 2d \cdot Cx - d \cdot d.\end{aligned}$$

But $\mathbf{x}$ is a binary vector, so $\mathbf{x}_i = \mathbf{x}_i^2$ for all i and therefore the linear term $-2d \cdot C\mathbf{x}$ can be expressed as a quadratic term. So we can write $-2d \cdot C\mathbf{x} = -2\mathbf{x}^T \mathrm{diag}(C^T d)\mathbf{x}$ and thus penalty 1 becomes

$$(C\mathbf{x} - d) \cdot (C\mathbf{x} - d) = \mathbf{x}^T (C^T C - 2\mathrm{diag}(C^T d))\mathbf{x} - d \cdot d$$

where $\mathrm{diag}(v)$ is the function which maps the ith entry of v to the ith diagonal entry of a diagonal matrix.

Penalty 2 can be written as,

$$H\mathbf{x} \cdot H\mathbf{x} = \mathbf{x}^T H^T H\mathbf{x}.$$

So the QUBO formulation of this problem is

$$\begin{aligned}& -\mathbf{x}^T\mathbf{x} + \lambda_1 (C\mathbf{x} - d) \cdot (C\mathbf{x} - d) + \lambda_2 H\mathbf{x} \cdot H\mathbf{x} \\ =& -\mathbf{x}^T\mathbf{x} + \lambda_1(\mathbf{x}^T (C^T C - 2\mathrm{diag}(C^T d))\mathbf{x} - d \cdot d) + \lambda_2 \mathbf{x}^T H^T H\mathbf{x} \\ =& \ \mathbf{x}^T(-I + \lambda_1(C^T C - 2\mathrm{diag}(C^T d)) + \lambda_2 H^T H)\mathbf{x} - \lambda_1 d \cdot d.\end{aligned}$$

We can drop the constant $-\lambda_1 d \cdot d$ at the end and then write the problem as

$$\min\ f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x}$$

where $Q = -I + \lambda_1(C^T C - 2\mathrm{diag}(C^T d)) + \lambda_2 H^T H$.

Note that depending on the penalty weights used the optimal solution may not correspond to an isomorphism. So each optimal solution should be tested against the constraints.

3.2 Example: Graph Isomorphism problem

References

[1] Fahren Bukhari et al. “Formulation of Sudoku Puzzle Using Binary Integer Linear Programming and Its Implementation in Julia, Python, and Minizinc”. In: *Jambura Journal of Mathematics* 4.2 (2022), pp. 323–331. ISSN: 2656-1344. DOI: 10.34312/jjom.v4i2.14194.

[2] Stefan Klus and Patrick Gelß. *Continuous optimization methods for the graph isomorphism problem.* 2023. arXiv: 2311.16912.

[3] Sascha Mücke. *A Simple QUBO Formulation of Sudoku.* 2024. arXiv: 2403.04816.