Benchmark TSP- Mathematical Formulations

Ignacio Acedo Blanco[1]
Quantum Mads

This document is made to formulate the Travelling Salesman Problem (TSP) as a QUBO. This is used for the KPI Benchmark.

The TSP is one of the most well-known and studied combinatorial optimization problems in mathematics and computer science. It asks the question: *Given a set of cities and the distances between them, what is the shortest possible route that visits each city exactly once and returns to the starting point?* Despite its simple formulation, the TSP is computationally challenging, belonging to the class of NP-hard problems.

1 Formulation

Let $G = (E, V, C)$ denote an undirected graph, where E denotes the list of edges in the path, V the list of vertices and C a matrix encoding the cost of traversing each edge. Each node is denoted as $i \in V$, a node as $(i, j) \in E$ and the cost for this edge is C_{ij}. Denote $|V| = M$ the number of cities to be visited

As it is usually done, we will assume that this graph contains no negative loops (this is, paths with the same source and destination node such that the sum of the cost of the total travelled edges is negative) and that there are no path departing and arriving to the same node (in other words, $\text{diag}(C) = \mathbf{0}$).

The chosen formulation is the so-called Miller–Tucker–Zemlin (MTZ). Define as decision variable: $x_{ij} = 1$ if the path goes from node i to node j. As part of the subtour elimination (constraint that avoids subtours to be formed), we define some auxiliary variables $u_i \in \mathbb{R}^+$. These variables keep track of the other of visit: if $x_{ij} = 1$, then traveller goes from i to j and $u_i < u_j$. Then:

Objective:

$$\text{Minimize} \sum_{i \in V} \sum_{j \in V} C_{ij} \cdot x_{ij}$$

Constraints:

1. **Visit each city exactly once:**

$$\sum_{j \in V,\, j \neq i} x_{ij} = 1 \quad \forall i \in V$$

$$\sum_{i \in V,\, i \neq j} x_{ij} = 1 \quad \forall j \in V$$

2. **Subtour elimination (MTZ constraints):**

$$u_i - u_j + M \cdot x_{ij} \leq M - 1 \quad \forall i, j \in V,\, i \neq j \tag{1}$$

$$1 \leq u_i \leq M - 1 \quad \forall i \in V \setminus \{0\}$$

[1] ibenito@quantum-mads.com

3. **Binary constraints:**

$$x_{ij} \in \{0,1\} \quad \forall i,j \in V$$

Summary: this is a linear objective function formulation using M^2 binary variables $(M-1)$ continuous/integer variables with $2M + ((M-1)^2 - (M-1)) = M^2 - M + 2$ constraints (plus the lower and upper bounds of the u_i variables, that represent another $2(M-1)$ additional constraints).

1.1 Custom-defined constraints

In order to impose constraints as *city i_k is visited at time t_k*, we can proceed as follows: define auxiliary variables $u_i \in \mathbb{Z} \cap \{0,\dots,M-1\}$ for $i = 0,\dots,M$ and impose $u_{i_k} = t_k$. Therefore, we impose impose that variables u_i represent the time arrival of each city, not just keep track of the ordering. However, this implies that we need to modify the subtour elimination constraints. Now, they impose uniquely that $x_{ij} = 1$ implies $u_i < u_j$ (and of course that $u_i > u_j$ implies $x_{ij} = 0$). Now, I need to add the condition that $x_{ij} = 1$ if and only if $u_i - u_j = 1$. Note that this is a harder condition that includes the previous subtour elimination. Therefore, equation (1) must be replaced by:

$$1 - B_M(1 - x_{ij}) \leq y_i - y_j \leq 1 + B_M(1 - x_{ij}) \quad \forall\, i,j = 0,\dots,M; \tag{2}$$

where B_M represents a big M constant. This implies that $x_{ij} = 1 \Leftrightarrow y_i - y_j = 1$ and $x_{ij} = 0 \Leftrightarrow y_i - y_j \neq 1$.

Finally, we can tajke $B_M = M + 1$

Summary: this is a linear objective function formulation using M^2 binary variables M integer variables with $2M + 2 \cdot (M^2 - M) = 2M^2$ constraints (plus the lower and upper bounds of the u_i variables, that represent another $2M$ additional constraints).

2 Qubo formulation

One could take the MTZ formulation and convert it into a QUBO. However, this approach has drawbacks, primarily because it requires the inclusion of slack variables. Specifically, there are $(M-1)^2$ subtour elimination constraints, which are inequalities. To handle these, slack variables must be introduced. Each slack variable requires $\mathcal{O}(M)$ binary variables. As a result, the total number of variables needed to reformulate the problem as a QUBO is $\mathcal{O}(M^3)$.

To solve this issue, we will use the GPS formulation, that uses $2M^2$ binary variables. The idea is the following: for each pair of nodes $(i.j)$, define:

- $x_{ij,1} = 1$ if (i,j) appears in the path (and of course i is reached earlier that j
- $x_{ij,2} = 1$ if the edge (i,j) does not appear in the path and the node j is reached earlier than the i.

Objective:

$$\text{Minimize} \sum_{i \in V} \sum_{j \in V} C_{ij} \cdot x_{ij,1}$$

Constraints:

1. **Variable link**: if $x_{ij,1} = 1$ that means that $(i.j)$ appears in the path, so $x_{ij,2} = 1$

$$x_{ij,1} + x_{ij,2} = 1 \quad \forall\, i,j \in V$$

2. **If node i is visited before j, then the opposite is false, and viceversa:**

$$x_{ij,2} + x_{ji,2} = 1 \quad \forall i,j \in V$$

3. **Visit each city exactly once:**

$$\sum_{j\in V,\, j\neq i} x_{ij,1} = 1 \quad \forall i \in V$$

$$\sum_{i\in V,\, i\neq j} x_{ij,1} = 1 \quad \forall j \in V$$

4. **Subtour elimination - Penalty term:** the details are a bit not elegant. One can find the origin of this term in appendix X of the original paper.

$$\sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{k=1}^{M}(x_{ji,2}x_{kj,2} - x_{ji,2}x_{ki,2} - x_{kj,2}x_{ki,2} + x_{ki,2})$$

   Note this is the penalty term to be added in the QUBO directly.

5. **Binary constraints:**

$$x_{ij,r} \in \{0,1\} \quad \forall i,j \in V \; \forall r \in \{1,2\}$$

Summary: the QUBO defining the problem is:

$$\begin{aligned}
\underset{\boldsymbol{x}\in\{0,1\}^M}{\operatorname{argmin}} &\left(\sum_{i,j=1}^{M} C_{ij}x_{ij,1}\right) + \\
&+ \lambda_1 \sum_{i,j=1}^{M}\left(x_{ij,1} + x_{ij,2} - 1\right)^2 \\
&+ \lambda_2 \sum_{i=1}^{M}\left(\left(\sum_{\substack{j=1\\ j\neq i}}^{M} x_{ij,1} - 1\right)^2 + \left(\sum_{\substack{j=1\\ j\neq i}}^{M} x_{ji,1} - 1\right)^2\right) + \\
&+ \lambda_3 \sum_{i,j=1}^{M}\left(x_{ij,2} + x_{ji,2} - 1\right)^2 + \\
&+ \lambda_4 \sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{k=1}^{M}(x_{ji,2}x_{kj,2} - x_{ji,2}x_{ki,2} - x_{kj,2}x_{ki,2} + x_{ki,2})
\end{aligned} \tag{3}$$

This can be simplified once we expand the quadractic terms.

For the first constraint (variable link 1-2):

$$\begin{aligned}
P_1 &:= \sum_{i,j=1}^{M}(x_{ij,1} + x_{ij,2} - 1) = \sum_{i,j=1}^{M}(x_{ij,1} + x_{ij,2} + 1 - 2x_{ij,1} - 2x_{ij,2} + x_{ij,1}x_{ij,2}) \\
&= \underbrace{-\sum_{i,j=1}^{M} x_{ij,1}}_{=-1(\text{cons } 2)} + \sum_{i,j=1}^{M}(-x_{ij,2} + 1 + x_{ij,1}x_{ij,2}) \longrightarrow \tilde{P}_1 = \sum_{i,j=1}^{M}(x_{ij,2}(-1 + x_{ij,1}))
\end{aligned}$$

For the second set of constraints:

$$P_2 \coloneqq \sum_{i=1}^{M}\left(\left(\sum_{\substack{j=1\\j\neq i}}^{M} x_{ij,1}-1\right)^2+\left(\sum_{\substack{j=1\\j\neq i}}^{M} x_{ji,1}-1\right)^2\right)=$$

$$=\sum_{i=1}^{M}\left(-\sum_{\substack{j=1\\j\neq i}}^{M}(x_{ij,1}+x_{ji,1})+\sum_{\substack{j=1\\j\neq i}}^{M}\sum_{\substack{k=1\\k\neq j}}^{M}(x_{ij,1}x_{ik,1}+x_{ji,1}x_{ki,1})+2\right)=$$

$$\overset{*}{=}-2\sum_{i=1}^{M}\sum_{\substack{j=1\\j\neq i}}^{M}x_{ij,1}+2\sum_{i=1}^{M}\sum_{\substack{j=1\\j\neq i}}^{M}\sum_{\substack{k=1\\k\neq j}}^{M}x_{ij,1}x_{ik,1}+2\rightarrow \tilde{P}_2=\sum_{i=1}^{M}\sum_{\substack{j=1\\j\neq i}}^{M}\left(-x_{ij,1}+\sum_{\substack{k=1\\k\neq j}}^{M}x_{ij,1}x_{ik,1}\right)$$

Where in $*$ I have used that i and j are dummy variables and the sum commute, so $\sum_{i=1}^{M}\sum_{\substack{j=1\\j\neq i}}^{M}x_{ji}\overset{\text{dummy}}{=}\sum_{j=1}^{M}\sum_{\substack{i=1\\i\neq j}}^{M}x_{i,j}\overset{\text{commute}}{=}\sum_{i=1}^{M}\sum_{\substack{j=1\\j\neq i}}^{M}x_{ij}$. Similarly:

$$P_3 \coloneqq \sum_{i=1}^{M}\sum_{j=1}^{M}(x_{ij,2}+x_{ji,2}-1)^2=\sum_{i=1}^{M}\sum_{j=1}^{M}(-(x_{ij,2}+x_{ji,2})+1+2x_{ij,2}x_{ji,2})\rightarrow$$

$$\rightarrow \tilde{P}_3=\sum_{i=1}^{M}\sum_{j=1}^{M}(-x_{ij,2}+x_{ij,2}x_{ji,2})$$

Once again, we have compacted everything by changing dummy indices and using the commutativity of the sum (analogous to $*$). On the other hand, the subtour elimination remains untouched:

$$\sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{k=1}^{M}(x_{ji,2}x_{kj,2}-x_{ji,2}x_{ki,2}-x_{kj,2}x_{ki,2}+x_{ki,2})$$

Taking $\lambda_l=\lambda\ \forall\ l=1,2,3,4$, changing dummy indices as convenient and reordering, the final QUBO encoding the problem is:

$$\underset{\boldsymbol{x}\in\{0,1\}^M}{\operatorname{argmin}}\left(\sum_{i,j=1}^{M}C_{ij}x_{ij,1}\right)+$$
$$+\lambda\left\{\sum_{i,j=1}^{M}x_{ij,2}(-1+x_{ij,1})+\sum_{i=1}^{M}\sum_{\substack{j=1\\j\neq i}}^{M}\left[x_{ij,1}\cdot\left(-1+\sum_{k\neq j}^{M}x_{ik,1}\right)+x_{ij,2}\cdot(-1+x_{ji,2})+\right.\right.$$
$$\left.\left.+\sum_{k=1}^{M}(x_{ji,2}x_{kj,2}-x_{ji,2}x_{ki,2}-x_{kj,2}x_{ki,2}+x_{ki,2})\right]\right\}$$

Where to simplify and reduce the number of loops, I have used that $x_{ii,2}=2\ \forall i\in V$

2.1 Constrained case

When imposing constraints like *at time k go through node i*, if there is just one, one can solve the problem as usual and then reorder. If there is more than one, the formulation needs to be changed.

The best formulation I have found is defining $x_{i,t} = 1$ if the node i is reached at position t, otherwise $x_{i,t} = 0$. The problem is then:

$$\begin{aligned} \text{Minimize} \sum_{t\in T}\sum_{i\in V}\sum_{\substack{j\in V\\ j\neq i}} C_{ij}\cdot x_{i,t}x_{j,t+1} \\ s.t \sum_{i\in V} x_{it} = 1 \quad \forall\, t\in T \\ \sum_{t\in T} x_{it} = 1 \quad \forall\, i\in T \end{aligned}$$

where $T = \text{range}(\text{len}(V))$. Note that there is no need of subtour elimination.

If one defines $\boldsymbol{x} = (x_{00}, x_{01}, \ldots, x_{0N}, x_{10}, \ldots, x_{NN}) \in \mathbb{R}^{NN}$, then this cost function can be re-expressed as:

$$\begin{aligned} \text{Minimize} \quad & \boldsymbol{x}^t\boldsymbol{C}\boldsymbol{x} \\ \text{s.t.} \quad & A_{\text{time}}\boldsymbol{x} = \boldsymbol{b} \\ & A_{\text{space}}\boldsymbol{x} = \boldsymbol{b} \\ \text{where:} \quad & \boldsymbol{C} \coloneqq \left[\underbrace{\begin{pmatrix} 0 & 1 & \ldots & 1 \\ 1 & 0 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 0 \end{pmatrix}}_{\text{ones with diag} = 0} \odot C\right] \otimes \underbrace{\begin{pmatrix} 0 & 1/2 & 0 & \ldots & 1/2 \\ 1/2 & 0 & 1/2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1/2 \\ 1/2 & 0 & 0 & \ldots & 0 \end{pmatrix}}_{\text{super \& sub diag} = 0.5 \text{ plus corners}} \in \mathbb{R}^{NN\times NN} \\ & A_{\text{time}} = \left(\begin{array}{cccc|cccc|c|cccc} 1 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & \ldots & 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 & 0 & 1 & \ldots & 0 & \ldots & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 & 0 & 0 & \ldots & 1 & \ldots & 0 & 0 & \ldots & 1 \end{array}\right) = (\vec{1}_N^{\,t} \otimes I_N) \in \mathbb{R}^{NN\times N} \\ & A_{\text{space}} = (I_N \otimes \vec{1}_N^{\,t}) \in \mathbb{R}^{NN\times N} \\ & \boldsymbol{b} = \vec{1}_N \in \mathbb{R}^N \end{aligned}$$

To impose an equality matrix constraint: $Ax = b$, we impose it in a QUBO by adding a penalty term:

$$\lambda \sum_{i=1}^{m}\left(\sum_{j=1}^{n} A_{ij}x_j - b_i\right)^2.$$

Expanding the square (and neglecting constant terms):

$$\lambda \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{n} A_{ij}A_{ik}x_jx_k - 2\lambda\sum_{i=1}^{m} b_i\sum_{j=1}^{n} A_{ij}x_j = \lambda x^tA^tAx - 2\lambda b^tAx.$$

Thus, the final QUBO objective function becomes:

$$\begin{aligned} \text{minimize } & \boldsymbol{x}^t\boldsymbol{C}\boldsymbol{x} \\ & + \lambda\boldsymbol{x}^t\left(A_{\text{time}}^tA_{\text{time}} + A_{\text{space}}^tA_{\text{space}}\right)\boldsymbol{x} - 2\lambda\boldsymbol{b}^t(A_{\text{time}} + A_{\text{space}})\boldsymbol{x} \end{aligned}$$

or more compact:

$$\text{minimize } \boldsymbol{x}^t\boldsymbol{Q}\boldsymbol{x}$$

where:

$$Q = \boldsymbol{C} + \lambda(A^t_{\text{time}} A_{\text{time}} + A^t_{\text{space}} A_{\text{space}}) - 2\lambda(\boldsymbol{b}^t(A\_\text{time} + A\_\text{space})) \odot I\_N$$

And $\odot$ is the Hadamard (element-wise) product. Nota que el ultimo en termino que contiene el producto de Hadamard pone en la diagonal el vector resultante de la multiplicacion matricial que está entre paréntesis.

3 Real QUBO

Finally, let us now move on to Ising variables. To do this, for every component $x_i \in \{0, 1\}$ we define $z_i \in \{-1, 1\}$ as:

$$x_i = \frac{1 - z_i}{2}$$

Then, it is clear that $x_i = 0 \leftrightarrow z_i = 1$ and $x_i = 1 \leftrightarrow z_i = -1$. Susbtituing this into the last constrained case qubo formulation (the one with variables x_{it}):

Following [1], one can directly move to the Ising Space as:

$$\text{minimize } \boldsymbol{x}^t \boldsymbol{J} \boldsymbol{x} - \boldsymbol{h}^t \boldsymbol{x}$$

and:

$$J = Q$$

$$h = -(J + J^t) \cdot \vec{1}_N \qquad \left(h_i = -\left(\sum_j J_{ij} + \sum_j J_{ji} \right) \right)$$

To impose the constraints that at some time τ the traveler pass through node k, one can just add: $\lambda z_{k,\tau}$ to the QUBO, since it is minimum for $z_{k,\tau} = -1$ equivalent to $x_{k,\tau} = 1$. Note that $z_{k,\tau}$ is actually index $k * N + \tau$

4 Mixer

To construct a mixer that only explores the feasible set, use the equations (54)-(58) of this paper: https://arxiv.org/pdf/1709.03489, where the TSP problem is described in section 5.1.

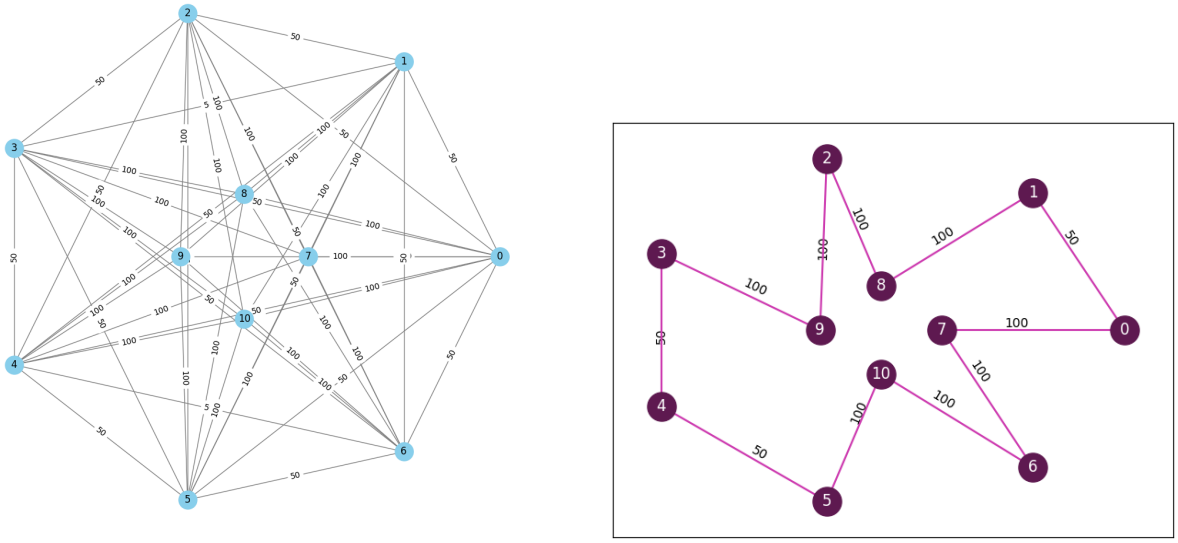
5 Heuristic solvers

To solve the problem with heuristic solvers, I change the formulation to find variables as x_{it} representing that at time t the salesman goes to i. Now the constraints are $\sum_i x_{it} = 1$ and $\sum_t x_{it} = 1$. Note that the solution to this problem can be seen as finding a permutation of `range(M)` (not the same as for MTZ formulation since now we cannot have subtours). The heuristic solvers use this approach to find the solution, so there are $M!$ candidates and M integer variables.

When imposing constraints, the corresponding permutation indices are fixed so the problem becomes easier for these solvers.

The problem can be formed as that as well directly for the binary case. However, this formulation need M^2 variables, $2M$ constraints and **a quadratic objective function** $\sum_{ijt} d_{ij} x_{it} x_{jt+1}$

6 Proposed example



(a) Proposed graph to be solved.

(b) Proposed solution to the graph.

Figure 1: Example of a proposed graph with 8 outer nodes and 4 inner nodes. The solution when imposing the starting node is 0 with constraints {8: 2, 9: 4, 10:8, 7:10}

.

The proposed graph to be solved for the benchmark is depicted in figure 1a.

The main idea is that the optimal solution is to traverse the graph on the outside making the pseudo-circumference. In order to put constraints on the problem, I have created interior nodes. The idea of these is to put some restrictions like {node: time} in which the traveler is forced to pass through a specific (inner) node in a specific time. We can put as many restrictions/inner nodes as there are outer nodes. Note that one cannot impose that the salesman goes consecutively to two inner nodes since there are not edges conecting these nodes. Thus, the optimal path will be to always go through outer edges and pass to an inner one when necessary. After passing through that node to fulfill the constraint, we immediately go to an exterior node to continue the trip. The weights of the edges have been selected to assert this is the optimal path

These are the specifications for this graph. Denote N the number of outer nodes and n the number of inner nodes. Then:

- $|V| = N + n$.
- $|E| = \frac{N(N-1)}{2} + Nn$.
- $n \leq N$

Finally, note that there are $n!$ possible states and $2n$ optimal solutions (since $(1, 2, 3)$ represents the same path as $(3, 1, 2)$, for instance, so we have n possible origin nodes for the same path, but also $(3, 2, 1)$-the inverse path- has the same solution)

References

[1] IBM Quantum Learning. *Solve utility-scale quantum optimization problems*. `https://learning.quantum.ibm.com/tutorial/quantum-approximate-optimization-algorithm`. Accedido: 3 de febrero de 2025.