# Quantum Assisted Graph Partitioning

Aryan Palimkar EP21B007, Hersh Samdani EP22B027

May, 2025

Code can be found at: `https://github.com/hershsam/Quantum-Assisted-Graph-Partitioning`

## Abstract

The task of graph partitioning is often encountered in processes, such as in VLSI floorplanning to minimize wire length and optimize size,in load balancing of high performance computing (HPC) codes to minimize total communication between processors and for processing various complex networks.

## Motivation: Graph Partitioning in QMD Simulations

Molecular dynamics (MD) simulations are essential for modeling the time evolution of atomistic systems. These simulations, which usually span time scales of picoseconds to nanoseconds ($10^{-12}$ to $10^{-9}$ s), calculate the trajectory of atoms based on interatomic forces at time steps as small as a femtosecond ($10^{-15}$ s). These forces are obtained from quantum mechanical calculations of the electronic structure in *quantum-based molecular dynamics* (QMD), which results in more precise but computationally demanding simulations.

Self-consistent tight-binding theory is the foundation of one well-known QMD technique. This method's primary computational task is to calculate the density matrix $D$ from the Hamiltonian $H$, which represents the quantum mechanical energy of the system. Traditionally, this requires diagonalization of $H$, an operation with cubic complexity $\mathcal{O}(N^3)$, where $N$ is the matrix dimension. Such operations are computationally feasible only for small systems.

To reduce this burden, alternative linear-scaling methods have been developed. One efficient algorithm is the *Sparse-Matrix Second-Order Spectral Projection* (SM-SP2) method, which approximates the density matrix using recursive polynomial expansions:

$$D = \lim_{i\to\infty} f_i[f_{i-1}[\cdots f_0[X_0]\cdots]], \tag{1}$$

where $X_0$ is a scaled version of $H$, and each $f_i(X)$ is either $X^2$ or $2X - X^2$ depending on trace conditions. Typically, 20–30 iterations suffice. Thresholding is applied during computation to retain matrix sparsity by setting small elements (e.g., below $10^{-5}$) to zero.

### Parallel Polynomial Evaluation via Graph Partitioning

The SM-SP2 method, while linear in complexity, requires repeated evaluations of matrix polynomials across many time steps. Efficient parallelization is crucial to reduce wall-clock time. This motivates the *Graph-based SP2* (G-SP2) algorithm, which models the sparsity pattern of $H$ as a graph and partitions it to parallelize polynomial evaluation.

We first represent the sparsity pattern of the matrix as a graph. For any symmetric matrix $X = \{x_{ij}\}$, define the *sparsity graph* $G(X)$ as a graph where each row (or column) $i$ is associated with a vertex, and there is an edge between vertices $i$ and $j$ if $x_{ij} \neq 0$.

We define a general class of thresholded matrix polynomials of degree $m = 2s$ as a superposition of quadratic operators and thresholding steps:

$$P = P_1 \circ T_1 \circ \cdots \circ P_s \circ T_s, \tag{2}$$

where each $P_i$ is a polynomial operator of degree 2 and $T_i$ is a thresholding operator that removes edges with small weights to maintain sparsity. Denote by $P(A)$ the application of this composite operator to matrix $A$, typically the Hamiltonian. The result $P(A)$ approximates the density matrix $D$.

Let $\mathscr{P}(G)$ be the worst-case sparsity graph of $P(A)$, constructed by including all potential non-zero positions that may arise during the polynomial evaluation—ignoring any numerical cancellations due to opposite-sign terms. This conservative structure ensures that no potential dependencies are missed during parallelization.

To evaluate $P(A)$ in parallel, we define a *core-halo (CH) partition* of $\mathscr{P}(G)$:

- Let $\Pi = \{\Pi_1, \ldots, \Pi_q\}$ be a partition of the vertex set $V(G)$, where each $\Pi_i$ is composed of a *core* $U_i$ and a *halo* $W_i$.
- The cores $U_i$ are disjoint and cover the graph: $\bigcup_i U_i = V(G)$, $U_i \cap U_j = \emptyset$ for $i \neq j$.
- The halo $W_i$ consists of neighbors of $U_i$ not in $U_i$: $W_i = N(U_i) \setminus U_i$.

Define $H = \mathscr{P}(G)$ and let $H_{U_i}$ be the subgraph induced by all the nodes and neighbors of $U_i$ in $H$. Let $A_{U_i}$ be the submatrix of $A$ indexed by the vertices in $V(H_{U_i})$. The following result justifies evaluating $P(A)$ using only these local submatrices.

**Lemma** *Let $v \in U_i$ and $w \in U_i \cup W_i$, where $W_i$ is the halo defined using an approximation of $\mathscr{P}(G)$. Then the matrix entry $P(A)_{vw}$ is equal to the corresponding entry in $P(A_{U_i})$.*

This lemma ensures that any non-zero entry of $P(A)$ involving a vertex in the core $U_i$ and its halo can be computed by evaluating the polynomial locally on $A_{U_i}$. In practice, since $\mathscr{P}(G)$ is not known a priori, we use an approximation such as $G(D)$ , where $D$ is the density matrix from the previous QMD time step to construct halos conservatively, as we expect the sparsity structure of this matrix to be similar to that of current H.

**Algorithm.** The full partitioned evaluation proceeds as follows:

1. Divide $V(G)$ into disjoint sets $U_1, \ldots, U_q$ and define the CH-partition $\Pi = \{\Pi_1, \ldots, \Pi_q\}$ with halos $W_i = N(U_i, G(D)) \setminus U_i$.
2. Construct submatrices $A_{U_i}$ containing the rows and columns of $U_i \cup W_i$.
3. Compute $\mathscr{P}(A_{U_i})$ for each $i$ independently using dense matrix operations.
4. Assemble $\mathscr{P}(A)$ by placing the computed entries for each $v \in U_i$ into the global matrix.

**Computational Cost.** Each polynomial evaluation on $A_{U_i}$ is a dense matrix computation over a matrix of size $c_i + h_i$, where $c_i = |U_i|$ and $h_i = |W_i|$. Since each superpositioned operator incurs cubic cost, the total cost is approximately:

$$\sum_{i=1}^{q} s(c_i + h_i)^3, \tag{3}$$

where $s$ is the number of polynomial iterations (e.g., 20–30 in SM-SP2). Therefore, the CH-partitioning problem is defined as:

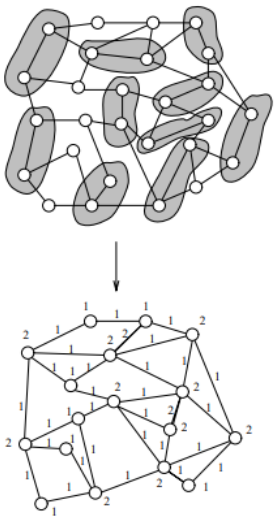
$$\text{Minimize } \sum_{i=1}^{q}(c_i + h_i)^3. \tag{4}$$

This objective captures the true computational cost of polynomial evaluation in SM-SP2 and provides the optimization target for graph-partitioning heuristics. Given the increasing complexity of quantum molecular simulations, integrating quantum algorithm for partitioning the SP2 workload is a promising direction for improving simulation throughput, particularly on hybrid quantum-classical architectures.

# METIS algorithm

METIS is a state-of-the-art graph partitioning algorithm that uses a multilevel scheme to generate partitions with minimal number of cross-partition edges.
A k-way graph partitioning problem is defined as follows: given a graph $G(V,E)$ with $|V| = n$, partition $V$ into k subsets, $V_1, V_2, ..., V_k$ such that $V_i \cap V_j = \emptyset$. The k-way partition can be solved by recursive 2-way partitions or bisections. We first obtain a bisection of V, then we further subdivide each part into two sections. After $log(k)$ iterations, the graph G is partitioned into k parts.Recursive bisection is one of the techniques employed by METIS to achieve k-way partitioning.
Multilevel graph bisection is carried out in the following three phases:

- **Coarsening phase:**The graph G is coarsened down to a few hundred vertices or less by generating a sequence of graphs $G_0, G_1, .., G_m$, where $G_0$ is the original graph such that $|V_0| > |V_1| > .. > |V_m|$.
  During this phase, a set of vertices of $G_i$ is combined to form a single vertex of the next level coarser graph $G_{i+1}$,called multinode. Let $V_i^v$represent the set of vertices in $G_i$ which are combined to give the vertex $v$ in $G_{i+1}$.To preserve the connectivity information in the coarser graph, in case more than one vertex of the set $V_i^v$ is connected to a vertex $u$ in $G_i$, the weight of the corresponding edge to $v$ in $G_{i+1}$ is the sum of the weights of these edges.
  One of the approaches used by METIS to combine vertices into a multinode is to find a random matching and collapse the matched vertices into a multinode.A matching of a graph is essentially a subset of its edges where no two edges share a vertex.As the goal of the matching is to reduce the number of edges, typically a maximal matching is used to obtain successively coarser graphs.



Coarsening by random matching

- **Partitioning phase:** A 2-way partition $P_m$ of the graph $G_m = (V_m, E_m)$ is found that divides $V_m$ into two parts, each containing half the vertices of $G_0$.
  An algortihm that can be used for this is the Kernighan-Lin (KL). As the partitioning phase contributes only 10 percent of the typical total runtime, the choice of the algorithm for this phase does not make a significant difference.
  In the K-L partitioning, we start with an initial bipartition, search for a subset of vertices from each part such that swapping them leads to a graph with smaller number of cut-edges. If such a subset exists,then the swap is performed and this becomes the partition for the next iteration.The algorithm continues by repeating the entire process.The Fiduccia-Mattheyses algorithm,a modification of the KL partitioning, uses certain data structures which reduce the time complexity to $O(|E|)$.
- **Uncoarsening phase:**The partition $P_m$ of $G_m$ is projected back to $G_0$ by going through intermediate partitions. As each vertex of $G_{i+1}$ contains a distinct subset of vertices of $G_i$, we can obtain $P_i$ from $P_{i+1}$ by assigning the set of vertices $V_i^v$ collapsed to $v \in G_{i+1}$ to the partition $P_{i+1}[v]$(i.e.$P_i[u] = P_{i+1}[v] \forall u \in V_i^v$).Since $G_i$ is finer, it has more information which can be used to improve the partition so obtained to the local minimum.Hence we use a refinement technique to get a smaller edge-cut (number of cut-edges). The KL algorithm can be used for this purpose :select subsets $A'$ and $B'$ from partitions $A$ and $B$ such that $A \backslash A' \cup B'$ and $B \backslash B' \cup A'$ is a bisection with a smaller edge-cut.

## Quantum annealing

The process of quantum annealing is based on the adiabatic theorem which states that a quantum system in its ground state will remain in the ground state, provided the Hamiltonian governing the dynamics changes sufficiently slowly.
The quantum algorithm is started by preparing the system in the ground state of a Hamiltonian, which is known and easy to prepare, denoted as $\mathscr{H}_i$ and is known as the initial Hamiltonian.Then the system is changed such that the contribution of $\mathscr{H}_i$ is reduced while the magnitude of a final Hamiltonian, denoted as $\mathscr{H}_f$ is increased:

$$\mathscr{H}(t) = A(t)\mathscr{H}_i + B(t)\mathscr{H}_f$$

The annealing schedule is decided by the monotonic functions $A(t)$ and $B(t)$ with $A(t = 0) = 1, B(t = 0) = 0$ and $A(t = T_a) = 0, B(t = T_a) = 1$ with $T_a$ being the max time limit on the annealing schedule.Typically the initial Hamiltonian has an eigenstate with equal amplitude superposition of $|0\rangle$ and $|1\rangle$ states for each qubit. The final Hamiltonian represents the objective function we want to minimize and is of the form:

$$\mathscr{H}_f = \sum_{i \in V} Q_i \sigma_i^z + \sum_{(i,j) \in E} Q_{ij} \sigma_i^z \sigma_j^z$$

.
Here, $V$ is the set of vertices of the graph $G(V,E)$ representing the lattice sites where qubits are located, $E$ the set of edges of the graph connecting the qubits, $J_{ij} = J_{ji}$ are the symmetric interaction strength of the qubits i and j connected by an edge and $h_i$ is the on-site energy(local field) of qubit i.

# Quantum Annealing for Graph Partitioning

Different disciplines such as social and biological networks,VLSI deign and image segmentation require the application of graph partitioning to reduce the complexity of the problem by dividing it into computationally feasible subproblems or to enable parallelization.

## k-Concurrent approach

The k-Concurrent approach enables us to partition a graph into k parts in parallel without going through a recursive routine. Each graph vertex is represented by a super-node consisting of k subnodes,where k is the number of partitions. Each of these k subnodes is represented by a qubit. An assignment of value 1 to the 'i'-subnode indicates that the node should belong to the 'i'th partition.

In order to partition a graph concurrently into an arbitrary number of parts, we formulate the problem as a QUBO (Quadratic Unconstrained Binary Optimization), which takes binary variables.In accordance with the super-node structure described above, we define the following decision variables:

$$x_{ij} = \begin{cases} 1, & \text{if node i is in part j} \\ 0, & \text{otherwise} \end{cases}$$

### Constraints

- The constraint $\sum_{j=1}^{k} x_{ij} = 1$ for each node i ensures that each node is in exactly one part.
- We also impose balancing constraints for the part sizes: $\sum_{i=1}^{n} x_{ij} = \frac{n}{k}$ for each part j$\in \{1,k\}$.

**Formulation**

The number of cut-edges across the k-parts is given by
$\frac{1}{2}(\sum_{j=1}^{k} \mathbf{x}_j^T L \mathbf{x}_j)$ where we have defined $\mathbf{x}_j$ ,the 'j'th-partition vector as

$$\begin{bmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{bmatrix}$$

and $L$ is the Laplacian matrix of the graph concerned (G). The Laplacian matrix of a graph G is given by

$$L = D - A$$

where $D$ is a diagonal matrix with entry $D_{ii}$ equal to the degree $g_i$ of vertex i and $A$ is the adjacency matrix.
*Proof*:
Number of cut-edges in this formulation is $\mathrm{N}_c = \frac{1}{2}(\sum_{j=1}^{k} \sum_{(a,b)\in E} (x_{aj} - x_{bj})^2)$.We sum over all parts from 1 to k to obtain the total number of edges, but in doing so count each cut-edge twice (once for each endpoint's respective part). Hence the division by 2.By taking any arbitrary orientation (arbitrary assignment of directions to the edges) of the graph G, we can define its incidence matrix $C$ as

$$C_{il} = \begin{cases} 1, & e_l = (i,k) : \text{if } e_l \text{ is an edge from } i \text{ to } k \\ -1, & e_l = (k,i) : \text{if } e_l \text{ is an edge from } k \text{ to } i\,. \\ 0, & \text{otherwise} \end{cases}$$

Thus the number of edges whose one endpoint is in part j is $||\mathbf{x}_j^T C||^2 = \mathbf{x}_j^T C C^T \mathbf{x}_j$.It can be shown that the Laplacian matrix of the graph is equal to $CC^T$. This proves the formula for the number of cut-edges.
The QUBO form of the graph partitioning problem can then be written as:
$\beta(\sum_{j=1}^{k} \mathbf{x}_j^T L \mathbf{x}_j) + \sum_{j=1}^{k} \alpha_j(\sum_{i=1}^{n} x_{ij} - \frac{n}{k})^2 + \sum_{i=1}^{n} \gamma_i(\sum_{j=1}^{k} x_{ij} - 1)^2)$
where we have included the constraints as quadratic penalties. To further develop our formalism, we introduce the following matrices:

- $\alpha I$ - a block diagonal matrix of k $n$x$n$ blocks given by

$$\alpha I = \begin{pmatrix} \alpha_1 \mathbb{1}_{nxn} & \alpha_2 \mathbb{1}_{nxn} & \dots & \alpha_k \mathbb{1}_{nxn} \end{pmatrix}$$

- $\mathbf{X}^T$ is the vector with the elements of the partition vectors appended one after the other.

$$\mathbf{X}^T = \begin{bmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T & \dots & \mathbf{x}_k^T \end{bmatrix}$$

-

$$\Gamma^T = \begin{pmatrix} \gamma^T & \gamma^T & \dots & \gamma^T \end{pmatrix}$$

where

$$\gamma = \begin{pmatrix} \gamma_1 & \gamma_2 & \dots & \gamma_n \end{pmatrix}$$

is the vector of Lagrangian coefficients of the second constraint term.

- $L$- a block diagonal matrix with k copies of the Laplacian matrix of the concerned graph on the diagonal.
- $\mathbf{Z}_i$ is the $nk$x$nk$ diagonal matrix whose j'th diagonal element is 1 if and only if j≡i(mod n),otherwise zero. For example in $\mathbf{Z}_2$, every $2^{nd}$,$(n+2)^{th}$,...,$((k-1)n+2)^{th}$diagonal element is 1 and has zero everywhere else.

Representing each term of the QUBO relaxation in terms of the above matrices we get:

- For the first term (objective), we get:

$$\beta \sum_{j=1}^{k} \mathbf{x}_j^T L \mathbf{x}_j = \beta \mathbf{X}^T L \mathbf{X}$$

- $$\begin{aligned} \sum_{j=1}^{k} \alpha_j (\sum_{i=1}^{n} x_{ij} - \frac{n}{k})^2 &= \sum_{j=1}^{k} \alpha_j ((\sum_{i=1}^{n} x_{ij})^2 - 2\frac{n}{k} \sum_{i=1}^{n} x_{ij} + \frac{n^2}{k^2}) \\ &= \sum_{j=1}^{k} \alpha_j (\mathbf{x}_j^T \mathbb{1}_{nxn} \mathbf{x}_j - 2\frac{n}{k} \mathbb{1}_n^T \mathbf{x}_j + \frac{n^2}{k^2}) \\ &= \mathbf{X}^T \alpha I \mathbf{X} - 2\frac{n}{k} \alpha \mathbb{1}_{nk}^T \mathbf{X} + \frac{n^2}{k^2} \sum_{j=1}^{k} \alpha_j \end{aligned}$$

- $$(\sum_{j=1}^{k} x_{ij} - 1)^2 = (\sum_{j=1}^{k} x_{ij})^2 - 2\sum_{j=1}^{k} x_{ij} + 1$$

  Using the $\mathbf{Z}_i$ matrices discussed in the previous section, we get:

$$(\sum_{j=1}^{k} x_{ij})^2 = \mathbf{X}^T \mathbf{Z}_i \mathbb{1}_{(nk)\mathbf{x}(nk)} \mathbf{Z}_i \mathbf{X}$$

$$\sum_{j=1}^{k} x_{ij} = \mathbb{1}_{nk}^T \mathbf{Z}_i \mathbf{X}$$

  Now we can condense $\sum_{i=1}^{n} \gamma_i (\sum_{j=1}^{k} x_{ij} - 1)^2$ into:

$$\begin{aligned} &\sum_{i=1}^{n} \gamma_i (\mathbf{X}^T \mathbf{Z}_i \mathbb{1}_{(nk)\times(nk)} \mathbf{Z}_i \mathbf{X} - 2\mathbb{1}_{nk}^T \mathbf{Z}_i \mathbf{X} + 1) \\ &= \mathbf{X}^T \sum_{i=1}^{n} \gamma_i (\mathbf{Z}_i \mathbb{1}_{(nk)\times(nk)} \mathbf{Z}_i) \mathbf{X} - 2\sum_{i=1}^{n} \gamma_i \mathbb{1}_{nk}^T \mathbf{Z}_i \mathbf{X} + \sum_{i=1}^{n} \gamma_i \end{aligned}$$

  $\sum_{i=1}^{n} \gamma_i (\mathbf{Z}_i \mathbb{1}_{(nk)\mathbf{x}(nk)} \mathbf{Z}_i)$ is the block matrix, consisting of k $n$x$n$ blocks on its diagonal, where each block is equal to

$$\begin{pmatrix} \gamma_1 & & & \\ & \gamma_2 & & \\ & & \ddots & \\ & & & \gamma_3 \end{pmatrix}$$

Designating such a block matrix with $\mathbf{B}_\Gamma$, we have: $\sum_{i=1}^{n} \gamma_i \mathbf{Z}_i \mathbb{1}_{(nk)\mathrm{x}(nk)} \mathbf{Z}_i = \mathbf{B}_\Gamma$. Further, it we note that $\sum_{i=1}^{n} \gamma_i \mathbb{1}_{nk}^T \mathbf{Z}_i$ is equivalent to $\Gamma_T$, which was introduced earlier. Hence, the second constraint can be re-written as $\sum_{i=1}^{n} (\sum_{j=1}^{k} x_{ij} - 1)^2 = \mathbf{X}^T \mathbf{B}_\Gamma \mathbf{X} - 2\Gamma^T \mathbf{X} + \sum_{i=1}^{n} \gamma_i$

Taking all the above points into consideration, the final QUBO which will be submitted to the annealer is of the form:
$min_x(\mathbf{X}^T(\beta L + \alpha I + \mathbf{B}_\Gamma)\mathbf{X} - (2\Gamma^T + 2\frac{n}{k}\alpha \mathbb{1}_{nk}^T)\mathbf{X})$

# Quantum Approximate Optimization Algorithm

QAOA is a classical-quantum hybrid method for solving combinatorial optimization problems. It makes use of two Hamiltonians - the cost $H_p$ Hamiltonian and the mixer $H_m$ Hamiltonian. These are alternatively repeated in the circuit in multiple "layers" and QAOA approximates the solution of the problem by obtaining the optimum configuration of the circuit parameters.

Illustrated below, taken from [7] is a flowchart depicting the flow of our algorithm. The problem is first formulated as a QUBO/Ising model. The model is used to determine $H_p$ and $H_m$, which are then used to design the unitaries for QAOA, $e^{-i\beta H_m}$ and $e^{-i\gamma H_p}$, which are alternatively repeated in the quantum circuit in the aforementioned layers. The circuit iteratively optimizes $\beta$ and $\gamma$ using classical optimization to maximize the expectation value of $H_p$. In our project, we implement QAOA on Pennylane.

To actually implement it, we map the QUBO to a quantum circuit by transforming our variables to Pauli operators.

Two sets of quantum gates, a mixer unitary and problem unitary are designed using the quantum Hamiltonian. These unitary operators are used to create the initial QAOA Ansatz circuit. This circuit is trained with classical optimization for updating the variational parameters $\beta$ & $\gamma$ in all layers. The decision variables are mapped to Pauli gates for the Pauli Ising Hamiltonian:

$$[h_1 \dots h_n] \cdot \begin{bmatrix} s_1 \mapsto Z_1 \\ \vdots \\ s_n \mapsto Z_n \end{bmatrix} + [s_1 \mapsto Z_1 \dots s_n \mapsto Z_n] \cdot \begin{bmatrix} 0 & \dots & J_{1,j} & \dots & J_{1,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & J_{i,j} & \dots & J_{i,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & \dots & 0 \end{bmatrix}_{(j>i)} \cdot \begin{bmatrix} s_1 \mapsto Z_1 \\ \vdots \\ s_n \mapsto Z_n \end{bmatrix}$$

where $Z_1, \dots, Z_n$ are Pauli matrices on their respective qubits.

The problem Hamiltonian $H_p$ is given by:

$$H_p = A_1 \left(a_1 I_n \otimes I_{n-1} \otimes \dots \otimes I_2 \otimes Z_1 + \dots + a_n Z_n \otimes I_{n-1} \otimes \dots \otimes I_1\right) +$$

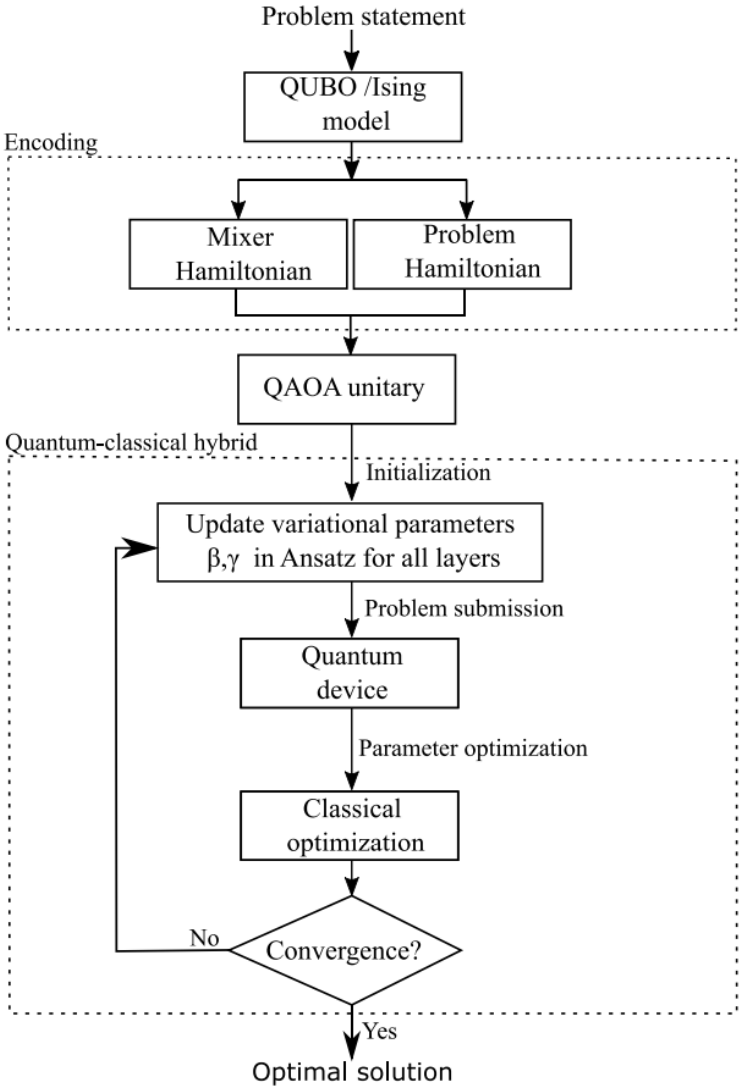



Figure 1: QAOA Flowchart

$$A_2\left(b_{1,2}I_n \otimes I_{n-1} \otimes \ldots \otimes I_3 \otimes Z_2 \otimes Z_1 + \cdots + b_{n-1,n}Z_n \otimes Z_{n-1} \otimes I_{n-2} \otimes \cdots \otimes I_1\right)$$

where $A_1$ and $A_2$ are normalization factors, $a_1, \ldots, a_n$, $b_{1,2}, \ldots, b_{n-1,n}$ are coefficients.

The mixer Hamiltonian $H_m$ is:

$$H_m = I_n \otimes I_{n-1} \otimes \cdots \otimes I_2 \otimes X_1 + I_n \otimes I_{n-1} \otimes \ldots \otimes X_2 \otimes I_1 + \cdots + X_n \otimes I_{n-1} \otimes \cdots \otimes I_1$$

The unitary operators are defined as:

$$U(\beta) = e^{-i\beta H_m}, \quad U(\gamma) = e^{-i\gamma H_p}$$

The QAOA ansatz state is constructed as:

$$|\phi(\beta,\gamma)\rangle = U(\beta)_{k-1}U(\gamma)_{k-1}\ldots U(\beta)_0 U(\gamma)_0 |\phi_0\rangle$$

where $|\phi\rangle$ is the final state and $|\phi_0\rangle$ is the initial superposition state $\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)^{\otimes n}$.

Finally, the objective function is:

$$\langle \phi(\beta_{\text{opt}}, \gamma_{\text{opt}}) | H_p | \phi(\beta_{\text{opt}}, \gamma_{\text{opt}}) \rangle$$

# Results and Comparison

Random graph models such as Erdos-Renyi and a few standard graphs were generated using NetworkX (Python library) and used for analyzing the performance of the two techniques- quantum annealing and METIS. With regards to the QA implementation on the DWave system, the limitations imposed by embedding were taken into consideration. In order to express the problem's QUBO formulation in a form compatible with the hardware topology of the DWave system, an embedding has to be carried out. This involves encoding the decision variables as chains of multiple physical qubits on a Quantum Processing Unit (QPU) such that they act as a single local qubit.

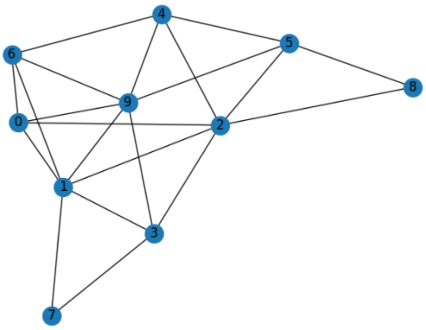
## Quantum Annealing

Two samplers were considered-the DWaveSampler,which directly embeds the concerned QUBO onto the QPU to sample for its minimum energy state and the LeapHybridSampler,which implements state-of-the-art classical algorithms together with intelligent allocation of the QPU to parts of the problem where it benefits the most. The number of decision variables required is n (number of nodes) x k (number of partitions). This saturates direct QPU-based sampling (takes considerable time to run/gives infeasible results) at around 100 variables, which could be, for example, 25 nodes and 4 partitions.

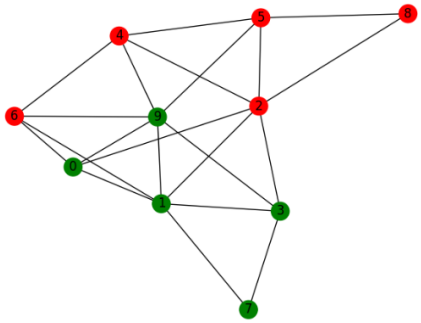
| $k$ | $n$ | No. of edges | DWaveSampler | METIS |
|---|---|---|---|---|
| 2 | 10 | 21 | 8 | 8 |
| 2 | 15 | 46 | 19 | 22 |
| 2 | 20 | 95 | 43 | 36 |
| 2 | 25 | 152 | 68 | 77 |
| 2 | 30 | 213 | 99 | 82 |
| 4 | 10 | 21 | 15 | 15 |
| 4 | 15 | 46 | 38 | 34 |
| 4 | 20 | 95 | 74 | 59 |
| 4 | 25 | 152 | - | 107 |

Table 1

Table 1 gives a comparison of the number of cut-edges obtained by METIS and the DWaveSampler on Erdos-Renyi graphs of varying sizes (n) and for different number of partitions(k). The probability parameter $p$ specified for the random Erdos-Renyi graph was taken to be 0.5 as this roughly gives the maximum number of 'effective' edges and hence maximum 'effective' connectivity. The word 'effective' is used since a graph with same number of nodes and more than half the total number of possible edges can be tackled by taking its complement and applying the graph-partitioning algorithms to it.The DWaveSampler produces solutions of a quality comparable to those produced by METIS and gives roughly the same number of cut-edges within a margin of 15% of the total number of edges. The following figure illustrates the 2-way partitioning of a graph with 10 nodes as obtained via quantum annealing:



Erdos Renyi Graph with 10 nodes



Bipartitioned graph

k-Concurrent graph partitioning on large graphs requires the use of the hybrid classical-quantum LeapHybridSampler since a direct QPU embedding exhausts space and the ability to infer values of logical variables from the states of the respective chain of physical qubits. Results of the application of METIS and the LeapHybridSampler on graphs of size 250, 500 and 1000. The parameter $p$ is set to 0.5 just as before.

| $k$ | $n$ | No. of edges | LeapHybridSampler | METIS |
|---|---|---|---|---|
| 2 | 100 | 2430 | 1053 | 1072 |
| 2 | 250 | 15565 | 7158 | 7199 |
| 2 | 500 | 62395 | 29311 | 29535 |
| 4 | 100 | 2430 | 1640 | 1660 |
| 4 | 250 | 15565 | 10810 | 10966 |

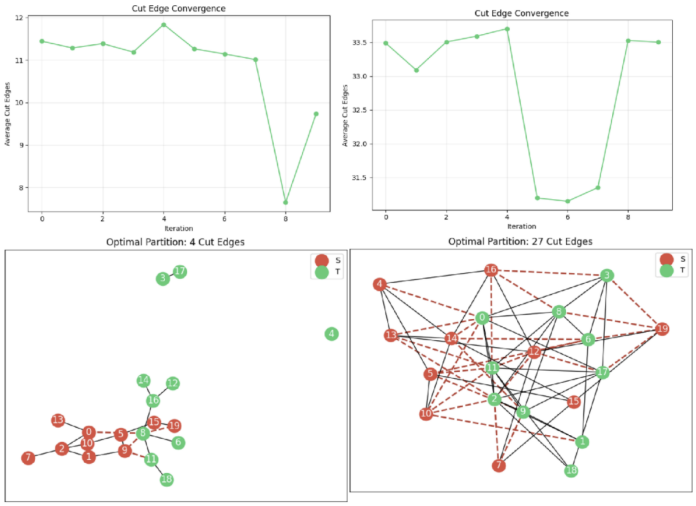
Table 2

It can be seen that the Leap Hybrid Sampler consistently outperforms METIS in partitioning larger graphs giving as low as 200 cut-edges fewer than the corresponding METIS solution, although the number of imbalanced partitions is nonzero in the former case. A partition is classified as balanced if its number of nodes is within 10% of $n/k$. Table 3 shows the number of cut edges obtained for a simpler case of the Erdos-Renyi graph to enable a comparison with QAOA.

## QAOA

We tested our own QAOA algorithm on Pennylane's simulator. We faced problems with the convergence of the cost and so, by extension, the number of cut edges. We do however, observe decent results with even a few iterations, the downside being there is no real improvement with increasing the number of iterations. It can thus be concluded that there is some problem with the parameter optimization pipeline. The results for two cases we tested to compare with QA and METIS are given in Table 3. Also given below is the plot of the number of cut edges varying with the number of iterations, which highlights our main issue with QAOA.

## Comparison:

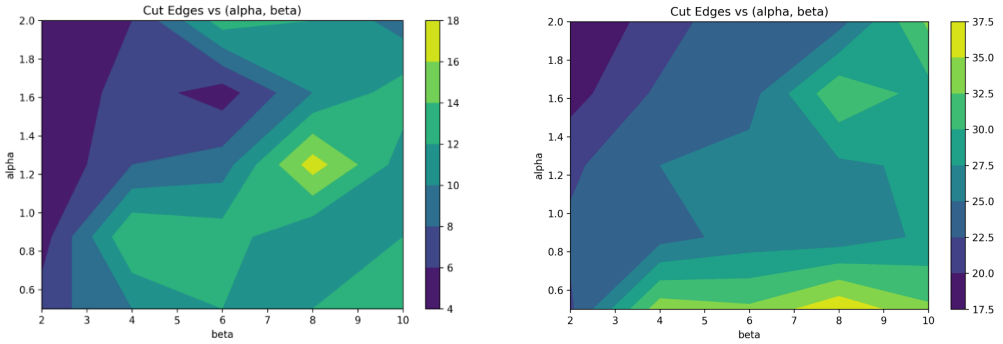



Number of cuts vs Iterations and the obtained partitioned graph for the cases in Table 3.

| k | n | No. of edges | DWaveSampler | METIS | QAOA |
|---|---|---|---|---|---|
| 2 | 20 | 23 | 4 | 4 | 4 |
| 2 | 20 | 67 | 22 | 23 | 27 |

Table 3

The values of the parameters $\alpha, \beta, \gamma$ were chosen by performing a parameter sweep over the $\alpha, \beta$ values, keeping $\gamma$ constant a fixed large value (=5), since we would most definitely want to avoid the partitions where a node has been assigned to more than one partitions. Figure 3 illustrates a heat map indicating the variation of the number of cut-edges with $\alpha$ and $\beta$:




Erdos Renyi Graph with 20 nodes and 23 edges  Erdos Renyi Graph with 20 nodes and 67 edges

A similar performance was observed in the case of partitioning of graphs associated with the density matrix calculations for the molecules- Phenyl Dendrimer and Peptide 1aft N in [2]. An attempt was made to reproduce results for these molecules, but we could only obtain the classical results through our METIS implementation. In the case of QAOA, the run time for the molecules was too high to be feasible, most probably due to the failings of the classical optimization. For the case of quantum annealing, the Embedding Composite method was unable to find a suitable embedding for the 730 node Dendrimer graph which motivates the need to decide a manual embedding. An alternative could be to apply quantum annealing or QAOA in place of the exact solver for

the subproblems in METIS. The graphs could be coarsened enough to be manageable for the existing QPUs, and can be partitioned easily, serving as the starting point for the next refined graph, as shown in [5].
Hence we have shown that quantum annealing is a suitable alternative to traditional GP tools.

# References


[1] *Catherine C. McGeoch*
Adiabatic Quantum Computation and Quantum Annealing, Springer.

[2] *Ushijima-Mwesigwa H., Negre C., Mniszewski S.*
Graph Partitioning using Quantum Annealing on the D-Wave System. Available at: https://arxiv.org/ftp/arxiv/papers/1705/1705.03082.pdf

[3] *Karypis G., Kumar V.*
A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. Available at: https://www.cs.utexas.edu/~pingali/CS395T/2009fa/papers/metis.pdf

[4] *Glover G., Kochenberger G., Du Y.*
A Tutorial on Formulating and Using QUBO Models. Available at: https://arxiv.org/abs/1811.11538

[5] *Ushijima-Mwesigwa H., Negre C., Mniszewski S.*
Multilevel Combinatorial Optimization Across Quantum Architectures. Available at: https://arxiv.org/pdf/1910.09985

[6] D-Wave Systems, D-Wave Solver Documentation. Available at: https://docs.dwavesys.com/docs/latest/c_gs_2.html

[7] *Bhatia D.*
Max-flow Min-cut Theorem in Quantum Computing. DOI: https://doi.org/10.1016/j.physa.2024.129990