A Quantum Formulation for Clustering Edge Topologies

Simone Reale, Elisabetta Di Nitto, Giovanni Quattrocchi, Luciano Baresi
DEIB - Politecnico di Milano, Milano, Italy
{simone1.reale, elisabetta.dinitto, giovanni.quattrocchi, luciano.baresi}@polimi.it

***Abstract*—Edge topologies usually comprise many different computational nodes, each with limited resources. Moreover, they manage varying workloads that can access the system from different entry points. These key characteristics make the problem of selecting the nodes dedicated to a specific workload quite heavy. In this context, clustering, that is, organizing nodes into smaller, more manageable groups, is a critical first step to support effective decision making.**

This paper introduces Min-Distance Based Clustering (M-DBC), a novel quantum-based model for clustering edge topologies that utilizes quantum annealing. By recasting the clustering task as a quantum optimization problem, we can easily identify the appropriate clusters of edge nodes and nearby demand sources in a spatially aware manner. Our approach enables fast, periodic clustering executions, allowing the edge topology to adapt quickly to continuously changing conditions. We thoroughly evaluate the proposed method and demonstrate clear gains in scalability and execution times while maintaining almost optimal solution quality.

***Index Terms*—Quantum annealing, edge computing, topology clustering**

I. Introduction

Edge computing shifts data processing and storage from centralized data centers to the periphery of the network. By processing information near its source [1], one can reduce latency, conserve bandwidth, and enhance data privacy. Edge computing is designed to support emerging applications such as autonomous driving and industrial IoT, where rapid response times and local data processing are critical [2].

The execution of computation in edge environments is challenged by limited resources. Edge servers typically offer far less computational power and memory than traditional cloud data centers. Moreover, in many cases, they can exploit only small and limited amounts of energy to operate. The dynamic nature of workloads, coupled with user mobility and fluctuating network conditions, compounds the difficulty of ensuring that all tasks are executed under optimal conditions without violating the Quality of Service (QoS) requirements [3]. Poor resource allocation, misaligned application placement, or inefficient task scheduling can lead to performance degradation and increased latency, reducing the benefits of edge computing.

Proper techniques for identifying nodes capable of serving user demand, and for clustering demand sources around nearby nodes, can significantly mitigate the complexity and overhead associated with managing distributed edge environments. These techniques contribute to improving workload distribution and overall system response [4]–[6].

However, classical clustering algorithms are not fully suitable for addressing two important characteristics of edge computing. The first is the size of the topologies, which are made up of hundreds or even thousands of edge nodes and an even significantly higher number of source nodes connected to the former [1]. The second is that user mobility patterns are highly dynamic and determine continuous changes in how traffic is distributed across the edge topologies. These two aspects clash with the fact that classical iterative clustering approaches can become computationally expensive, especially when repeated frequently to adapt to changing conditions [7]. This implies delayed decisions that can degrade system performance, lead to higher latency, reduce quality of service, and violate service-level agreements [1], [8].

Quantum computing has recently emerged as a viable approach to tackle combinatorial optimization and other resource-intensive problems [9], [10]. In particular, Quantum Annealing stands out as a specialized metaheuristic technique that can show promising speed-ups compared to its classical counterparts in complex optimization problems [11], [12].

Only a limited number of studies have explored how quantum computing can be used in the context of edge computing to enhance real-time decision-making capabilities [13]–[16]. Yet, the literature on practical implementations of quantum-driven approaches for edge management and, in particular, edge clustering remains mostly unexplored.

This paper presents a novel quantum clustering formulation called *Min-Distance Based Clustering (M-DBC)* tailored to the edge computing context. It addresses the problem of identifying a subset of edge nodes to handle user demand, given the locations of both users and edge nodes. M-DBC balances proximity constraints and spatial distribution, and it is designed for quantum executions through a Quadratic Unconstrained Binary Optimization (QUBO) [17] representation.

We demonstrate the practical applicability of this model in a realistic edge computing scenario involving a fleet of taxis operating in an urban area and dynamically interacting with a 5G antenna network topology. A comprehensive evaluation using real-world datasets shows that our quantum-based approach performs similarly to an optimal approach in terms of solution quality while completing the task in significantly less time and scaling effectively with increasing demand. The experimental analysis also includes another quantum-based clustering approach, outperformed by M-DBC in terms of solution quality.

The remainder of the paper is organized as follows. Section II reviews related work. Section III provides an overview of the problem we consider. Section IV presents our M-DBC proposal. Section V presents the set up for our evaluation and Section VI the experimental results. Section VII discusses the findings, and Section VIII concludes the paper.

II. Related Work

As mentioned, our work relies on Quantum Annealing (QA). It operates utilizing two key Hamiltonians: the *problem Hamiltonian* H_P, whose ground state encodes the solution to the optimization problem, and an *initial Hamiltonian* H_S, which has a simple and easily prepared ground state. The system evolves under a time-dependent Hamiltonian that interpolates between H_S and H_P, governed by a schedule function $p(t)$. This function depends on time and satisfies the boundary conditions $p(t_0) = 1$ and $p(t_{\text{end}}) = 0$. The resulting time-dependent Hamiltonian is expressed as: $H(t) = p(t)H_S + (1 - p(t))H_P$. In QA, the goal is to guide the system toward the ground state of H_P as it evolves, typically making use of both quantum fluctuations and thermal relaxation effects. Through quantum tunneling, QA can escape local minima more efficiently than classical thermal methods such as simulated annealing, making it well-suited for complex optimization landscapes with many local minima and sharp energy barriers.

This section reviews both approaches that employ clustering in edge computing and early applications of quantum computing in the edge domain.

A. Clustering in edge computing

Edge computing exploits clustering for three main activities. First, **node deployment and placement** involve deciding where to deploy edge nodes and/or how to distribute the computation (components, services, functions) onto them. Zhang et al. [4] tackle the joint problem of edge server deployment and service placement by first clustering, using *K-Means* [18] and user demand patterns; then they solve an optimization problem for placement within the clusters. A similar approach based on machine learning is presented in [4]. Xiang et al. [19] instead adopt a game-theoretic approach to deploy edge servers in large-scale networks. They apply clustering to group candidate locations and deploy edge nodes at the centroids of these clusters. Baresi et al. [5], [20] exploit the *SLPA* algorithm [21] to group nearby edge nodes into clusters and manage resource allocation and function placement more effectively by partitioning the global problem. Vali et al. [6] introduce a custom clustering algorithm based on linear integer programming and a greedy procedure to determine the optimal locations of edge servers in a city-wide mobile edge network. They cluster user locations and place an edge server in the center of each cluster to place servers close to users.

Second, clustering similar tasks or nodes can simplify **task scheduling** and offloading decisions. For example, Long et al. [22] uses *K-Means* to group nearby nodes. The scheduling problem assigns tasks only considering the nodes in a single cluster, thus reducing the solution space of the problem. Pan et al. [23] introduce a custom multi-objective evolutionary algorithm to schedule workflows on mobile edge nodes. They first cluster tasks in different workflows according to similarities (e.g., common data or deadline sensitivity), and then make the scheduling of these task clusters evolve to optimize objectives like execution cost and energy consumption. Also Ullah et al. [24] use *K-Means* to group the incoming tasks with respect to the resources they need (CPU, I/O or network). The system then schedules each group of similar tasks at the different nodes. Having similar tasks on appropriate nodes improves load balancing and reduces the chance of node overloading.

Third, **data placement and replication** are also key. Popular data must be replicated on different nodes to serve users quickly (i.e., low latency). This task can be formulated as a clustering problem by grouping users with similar content interests or contents based on access patterns. For example, Zhang et al. [25] propose a cluster-based caching strategy in which users are grouped using a custom spectral clustering algorithm based on social ties and physical locations. They assign each cluster of users to a nearby edge node and proactively cache content that is likely to be requested by those users. Chen et al. [26] introduce a weighted clustering approach based on *K-Means* for content popularity prediction that informs caching decisions. Instead of treating each content item in isolation, the method groups content items that exhibit similar popularity trends or usage contexts.

The reviewed literature highlights that most approaches adopt algorithms like *K-Means*, which are known to struggle in dynamic or large-scale environments [27].

B. Initial quantum-based solutions for the edge

Recent studies have begun investigating quantum-based approaches for managing edge topologies. Dong et al. [13] propose a quantum particle swarm optimization algorithm for task offloading. The integration of quantum principles and classical particle swarm optimization allows for a more efficient exploration of the search space, aiming to balance energy consumption and latency. Similarly, Bey et al. [16] present a quantum-inspired particle swarm optimization technique for service placement in edge environments based on IoT. The solution focuses on minimizing both service latency and resource usage through an efficient distribution of services on the different nodes. Ansere et al. [14] introduce a quantum machine learning framework that combines quantum circuits with classical neural networks to improve energy efficiency in mobile edge computing. Ansere et al. [15] also propose a quantum deep-reinforcement learning model for dynamic resource allocation in IoT-based edge systems; the use of quantum-enhanced policies helps adapt to varying network conditions. These studies demonstrate the potential of quantum computing, but none of them addresses clustering specifically.

III. Problem Overview

Figure 1 shows an edge topology as a pair of sets $\mathcal{ET} = \langle \mathcal{DP}, \mathcal{CSP} \rangle$. $\mathcal{DP}$ is the set of *demand points* (*dp*), that is, the

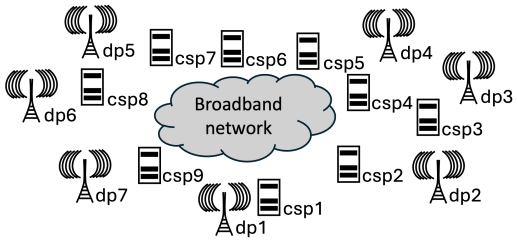

Fig. 1: Edge topology representation.

nodes from which requests are introduced in the system. $\mathcal{CSP}$ is the set of *candidate service points* (csps), which represent the nodes that could be used to serve incoming requests. Each node (either dp or csp) is characterized by its position in the reference space (it might be the geographical space or any other space where a notion of distance is defined). Moreover, csps are connected to a broadband network while dps can reach $csps$ at a close distance (e.g., through wireless channels). In addition, for simplicity, we assume that all requests generated by dps have a uniform weight and computational complexity, which means that each request requires roughly the same amount of resources.

The objective we want to achieve with a clustering approach is to select an optimal subset $\mathcal{ASP}$ of $\mathcal{CSP}$, which contains the *assigned service points* ($asps$) in charge of serving requests generated by the dps. This corresponds to a placement problem, where the goal is to determine the most effective deployment of nodes within the edge network. Each asp should be located close to dps to minimize response time and maximize efficiency. However, excessive concentration of $asps$ can lead to redundancy, leaving other regions underserved. The achievement of a proper balance between these two factors is the primary goal of this work. It requires an optimization approach that is inherently spatially aware, incorporating both proximity to demand and spatial separation constraints to ensure robust coverage. More formally, the goal of our optimization is to select a given number k of nodes in $\mathcal{ASP}$ in such a way that:

1) The selected $asps$ are spatially well distributed, maximizing their mutual distance to promote coverage across the reference space.
2) The selection is guided by an *importance* weighting factor that favors $csps$ located near those dps that generate a high volume of requests.
3) The set comprises exactly k nodes. .

After identifying the members of $\mathcal{ASP}$, the dps are clustered around them based on the chosen proximity criterion. If a dp were "close" to more than one asp, the asp is chosen randomly.

The $\mathcal{ASP}$ selection problem, in its most general form, occurs in multiple scenarios where resources should be strategically placed to maximize their effectiveness and can be framed within the research line related to facility location problems (FLP) [28], a fundamental branch of operations research. Such problems become intractable in closed-form formulations very quickly, since their asymptotic size grows proportionally to the cardinality of $\mathcal{DP}$ and $\mathcal{CSP}$ (e.g., in scenarios involving a large number of users and/or large-scale edge topologies). Nowadays, they are mainly approached with the help of heuristic algorithms that cannot effectively capture all the different facets of these complex multi-objective problems. This is why, in the near future, *quantum computing* can represent a viable option to tackle these problems. In addition to its natural inclination towards high-dimensionality data, it can also offer a way to explore the problem landscape more efficiently, offering different solutions that incorporate distinct trade-offs between set goals.

IV. Min-Distance Based Clustering (M-DBC)

The problem of selecting among $csps$ that can serve dps optimally could be approached using optimization-based methods or classical clustering methods.

P-Center [29] and *maximal coverage location* [30] are examples of optimization approaches that could be adapted to the problem at hand, but would not take into account the magnitude of demand characterized by the quantity of dps in a specific location. *P-Median* [31] would be intrinsically dependent on the size of $\mathcal{DP}$, thus limiting its ability to handle realistic edge topologies with numerous dps. Moreover, it would tend to overfit the specific $\mathcal{DP}$.

Analyzing the classical clustering algorithms, a plausible starting point could be the *K-Medoids* algorithm, which has also already been implemented as a QUBO model in [32]. The objective of K-Medoids is to identify cluster centers, called *medoids*. As such, it could be applied directly to the set of candidate service points ($\mathcal{CSP}$) with the goal of selecting k $csps$ to act as medoids, that is, $asps$, so that the sum of distances from the remaining unselected $csps$ to their nearest selected asp is minimized. However, the disadvantage of K-Medoids is that it lacks knowledge of dps, which is critical for effective service placement in edge computing.

To combine the strengths of optimization and classical clustering methods, we propose *Min-Distance Based Clustering (M-DBC)*, an approach that naturally adapts to changes in demand distribution without requiring explicit optimization over individual demand points. Our formulation is inherently spatially aware. It incorporates a factor that accounts for the *importance* of a csp relative to the distribution of nearby dps. It also includes a penalty for configurations where *asps* are positioned too closely –i.e., within a minimum operational distance– balanced by a complementary factor that encourages broader geographical *spread* of the *asps*. We build the mathematical foundations of M-DBC by describing it as a QUBO model suitable for running in a quantum context.

A. Distance Matrix and Normalization

Let $D \in \mathbb{R}^{n \times n}$ be the distance matrix, where D_{ij} represents the pairwise distance between csp_i i and csp_j, with $n = |\mathcal{CSP}|$. We normalize this matrix as follows:

$$\tilde{\Delta}_{ij} = 1 - \exp\left(-\gamma D_{ij}\right) \tag{1}$$

where $\gamma = \ln(2)/\bar{D}$ is a scaling parameter and $\bar{D}$ is the mean pairwise distance value. We call the matrix $\tilde{\Delta}$, resulting from all $\tilde{\Delta}_{i,j}$, *mean pairwise normalized distance*. By construction, all its values are in the range $[0, 1)$ (0 when $D_{i,j} = 0$, while it approaches 1 as $D_{i,j} \to \infty$).

The minimum distance allowed $d_{\min}$ is a parameter that can be used to model the operating range of *csps* and is normalized on the same scale of the distance matrix as:

$$\tilde{d}_{\min} = 1 - \exp(-\gamma d_{min}) \tag{2}$$

This kind of normalization ensures that at the mean value of the distance matrix $\bar{D}$, the transformation reaches 50% of its full range ($\tilde{\Delta} = 0.5$), while maintaining a good resolution on distances smaller than $\bar{D}$, allowing these distances to be easily distinguishable. Obtaining a good normalization of the range of values is essential for a model that seeks to penalize distances between *csps* that are below a certain threshold.

B. Optimization Model

Given binary decision variables z_i each stating whether csp_i has been selected, the general objective function

$O(z)|z \in \{0,1\}^n$ to be minimized comprises four terms: P (Min-Distance Penalty), S (Spread Reward), I (Importance Reward), and C (Constraint on the cardinality of $\mathcal{ASP}$):

$$O(z) = P + S + I + C \tag{3}$$

1) Min-Distance Penalty Term (P): penalizes pairs of selected *csps* that are closer than $d_{\min}$.

$$P = \sum_{i<j} z_i z_j \cdot \pi \cdot (\tilde{d}_{\min} - \tilde{\Delta}_{ij}) \tag{4}$$

where π is a penalty weight term defined as:

$$\pi = \left(\frac{c_p \cdot \bar{\Delta}}{k}\right) \cdot \frac{N_{spread}}{N_{penalty} + 10^{-6}} \tag{5}$$

with:

- $\bar{\Delta}$ being the mean pairwise normalized distance.
- $k = |\mathcal{ASP}|$ being the number of *csps* that must be selected at the end of the optimization process.
- $N_{penalty}$ being the number of *csp* pairs with distances below $\tilde{d}_{\min}$.
- N_{spread} being the number of *csp* pairs with distances above $\tilde{d}_{\min}$.
- c_p being a tuning coefficient; modifying its relative magnitude enables explicit manipulation of the tradeoff between term P and term S.
- 10^{-6} serving to avoid division by zero while computing π.

2) Spread Term (S): encourages selected *csps* to be far apart.

$$S = \sum_{i<j} z_i z_j \cdot (-\rho \cdot \tilde{\Delta}_{ij}) \tag{6}$$

where ρ is a spread term defined as:

$$\rho = \frac{c_s \cdot \bar{\Delta}}{\sqrt{n}} \tag{7}$$

with n being the cardinality of the set $\mathcal{CSP}$ and c_s a coefficient similar to c_p.

Since $\rho > 0$ and $\tilde{\Delta}_{ij}$ increase with distance, minimizing this term favors pairs (i, j) with larger $\tilde{\Delta}_{ij}$.

3) Importance Bias Term (I): encourages selection of *csps* with high importance scores.

$$I = \sum_i z_i \cdot (-\lambda \cdot w_i) \tag{8}$$

where λ is a non-negative weight controlling the influence of the importance score with a range in $[0, 10]$ found empirically, and w_i is the *importance value* of csp_i. The definition of importance values is application-specific. For example, importance can reflect the average distance from *csp* to nearby *dps*, the expected request load, the available computing capacity, or a combination of such factors. The definition of importance values in the context of our experiments is reported in Section V-B).

4) Constraint Violation Term (C): enforces the selection of exactly k *csps*.

$$C = \lambda_c \left(\sum_i z_i - k\right)^2 \tag{9}$$

where λ_c is a tunable Lagrange multiplier coefficient.

C. QUBO formulation

To use a quantum annealer as a solver, M-DBC should be mapped on a common QUBO formulation [17]:

$$\min_{z\in\{0,1\}^n} \mathcal{O}_{QUBO}(z) = \sum_{i<j} Q_{ij} z_i z_j + \sum_i Q_{ii} z_i + \sum_i c_i z_i + \chi \tag{10}$$

where χ represents a constant term, Q_{ij} represents the quadratic interaction between z_i and z_j, Q_{ii} represents the coefficient of the $z_i^2 (= z_i)$ term, and c_i is the coefficient of the purely linear z_i term.

P contributes to the off-diagonal QUBO coefficients:

$$Q_{ij}^{(P)} = \pi \cdot (\tilde{d}_{\min} - \tilde{\Delta}_{ij}) \quad (\text{for } i < j) \tag{11}$$

S contributes to the off-diagonal QUBO coefficients:

$$Q_{ij}^{(S)} = -\rho \cdot \tilde{\Delta}_{ij} \quad (\text{for } i < j) \tag{12}$$

I is purely linear and contributes to the c_i vector:

$$c_i^{(I)} = -\lambda \cdot w_i \tag{13}$$

C contributes to all terms in (10). More specifically, expanding the quadratic penalty in (9):

$$C = \lambda_c \left(\sum_i z_i^2 + \sum_{i\neq j} z_i z_j - 2k \sum_i z_i + k^2\right)$$

To map this to the QUBO form (10), we use the binary property $z_i^2 = z_i$ and identify coefficients:

$$Q_{ii}^{(C)} = \lambda_c, \quad Q_{ij}^{(C)} = 2\lambda_c \text{ (for } i < j), \quad c_i^{(C)} = -2k\lambda_c$$

Note the presence of a factor 2 in the coefficient $Q_{ij}^{(C)}$, given the necessity of representing symmetrical contributions in the expansion of the square: imposing $i < j$ we would, in fact, neglect the contributions of $z_i z_j$ with $i > j$, which we are, instead, considering with the factor 2.

We assemble the final QUBO coefficients:

- Off-diagonal coefficients ($i < j$):

$$Q_{ij} = Q_{ij}^{(P)} + Q_{ij}^{(S)} + Q_{ij}^{(C)} = \pi(\tilde{d}_{\min} - \tilde{\Delta}_{ij}) - \rho\tilde{\Delta}_{ij} + 2\lambda_c$$

- Diagonal coefficients ($i = j$): $Q_{ii} = Q_{ii}^{(C)} = \lambda_c$)
- Linear coefficients: $c_i = c_i^{(I)} + c_i^{(C)} = -\lambda \cdot w_i - 2k\lambda_c$

Of course, in the minimization of $O_{QUBO}(z)$ the constant term χ is ignored.

V. Validation set-up

The objective of the validation is threefold: (i) to confirm the effectiveness of the M-DBC model introduced in Section IV, (ii) to assess the suitability of quantum annealers as an execution platform for the model, and (iii) to compare the performance of M-DBC with that of two other state-of-the-art approaches, one executed on a classical solver and another both on classical and quantum solvers.

The analysis is performed on the use case of an edge network in Beijing (Section V-A) with the importance values w_i defined in Section V-B. The metrics used in our experiments are presented in Section V-C). The approaches used for comparison with M-DBC (Section V-D) have been selected for their ability to address the problem at hand and for the variety of execution environments they can cover.

A. The Beijing taxis use case

The use case presented here addresses the need for modern taxis to access services provided by an edge computing platform. In this scenario, taxis represent the set of dps that require access to software services, which are hosted on computational resources located at mobile network antenna sites, represented as $csps$. Our goal is to apply clustering approaches to identify a subset of $csps$ that minimizes the service delay experienced by taxis. To realistically model taxi mobility and antenna distribution within the target area, we rely on data from the following two main datasets:

- **Taxi Trajectories** [33] contains the GPS positions of 10,357 taxis sampled, on average, every 177 seconds, during the period of February 2 to February 8, 2008, within Beijing. It contains a total of 15 million data points.
- **Antenna Locations** [34] provides information about the location of 5G antennas in Beijing.

We use the Taxi Trajectories dataset to estimate a realistic distribution of taxi positions when requests are issued to an edge-based taxi application running on the edge topology. Each of these positions is treated as a demand point (dp). Since the dataset includes a large number of GPS points per taxi, and assuming that requests are issued only from a subset of these positions, we created a pipeline to filter and split the data. The goal is to generate plausible dp distributions over time and space. The process consists of the following steps:

1) **Time Splitting.** To consider a manageable amount of data, we divided the dataset into *time splits* of T hours each. In our experiments, we used a subset of these time splits $TS = 10$. The duration T of each split has been set to 4 hours for a total observation period of 40 hours.
2) **Taxi Selection.** In each time split, TX taxis are randomly selected. In our main experiments, we used $TX = 30$, generating an average of 2212 dsp for each split, while in the scalability experiments, we increased this number to a maximum of 15230.
3) **Selecting dps.** For each selected taxi within a time split, we choose a number of GPS positions from the dataset to represent the locations of edge requests. This number is drawn from a Poisson distribution $Poisson(\lambda_{taxi})$ where λ_{taxi} is the average request rate per taxi per hour. If the available data are insufficient, new positions are generated by interpolation. In our experiments, we used $\lambda_{taxi} = 20$

B. Use case-specific importance values

In Section IV we have introduced the concept of importance values as application-specific terms. For our use case, they are computed taking into account the density of demand via w_j^{radius}, while considering the antennas serving demand not covered by other antennas via w_j^{unique}:

$$w_j = \frac{w_j^{\text{radius}} + \zeta \cdot w_j^{\text{unique}}}{\max_{m \in \mathcal{CSP}} \left(w_m^{\text{radius}} + \zeta \cdot w_m^{\text{unique}} \right)} \tag{14}$$

where w_j^{unique} counts taxis that are uniquely covered by antenna j, w_j^{radius} counts the number of taxis within a given radius r from each antenna, capturing the localized density of demand:

$$w_j^{\text{radius}} = \frac{|\{i \in \mathcal{DP} \mid d_{ij} \leq r\}|}{\max_{k \in \mathcal{CSP}}(|\{i \in \mathcal{DP} \mid d_{ik} \leq r\}|)} \tag{15}$$

where d_{ij} is the distance between taxi i and antenna j. Finally, ζ in Equation 14 is a parameter that balances the two terms and is comprised in the range $[0, 1]$.

C. Evaluation Metrics

To assess the effectiveness of the proposed solutions, we consider a range of evaluation metrics that capture *service placement*, *demand coverage*, *spatial distribution*, *demand fairness*, and the average and standard deviation of *execution times*. These metrics comprehensively evaluate the set of selected service points and their ability to serve demand points effectively; the objective was to evaluate the solution with a test infrastructure unbiased with respect to specific services offered by the edge platform to evaluate the intrinsic optimality of the resulting specific edge topology.

Service Placement. To evaluate the quality of clusters, we considered two metrics. First, the *Average Distance to Service Point (ADSP)* measures how close *dps* are to the nearest *asp* on average:

$$ADSP = \frac{1}{|\mathcal{DP}|} \sum_{i=1}^{|\mathcal{DP}|} d(x_i, s_i) \tag{16}$$

where x_i is the location of a *dp*, s_i is its assigned asp, and $d(x_i, s_i)$ is the distance between them. A lower ADSP indicates better centrality. Second, the *Maximum Distance to Service Point (MDSP)* ensures that no *dp* is extremely far from the closest *asp*, representing worst-case performance:

$$MDSP = \max_i d(x_i, s_i) \tag{17}$$

where x_i is the location of a dp and s_i is its assigned asp.

Demand Coverage. Coverage assesses how well the solution ensures that requests generated by *dps* are served within a specific range r (mapped in our model to $d_{\min}$). A dp (x_i) is said to be *covered* if its distance to the nearest asp is less than or equal to r. The metric $Coverage(r)$ expresses the percentage of covered dps on the total (higher is better).

Let $\mathcal{DP}_{\text{cov}}(r) = \{x_i \in \mathcal{DP} \mid \min_{s_j \in ASP} d(x_i, s_j) \leq r\}$.

$$Coverage(r) = \frac{|\mathcal{DP}_{\text{cov}}(r)|}{M} \times 100 \tag{18}$$

Spatial Distribution. We consider three spatial metrics to evaluate the geographical spread of the locations of selected service points. First, metric *Convex Hull Area* computes the total area covered by the convex hull of the selected service points. A small area may indicate an over concentration in a single region.

Second, metric *Service Points Density per unit of measure* assesses whether the *asps* are distributed efficiently, that is, the operating ranges of $asps$ are not overlapping:

$$Density = \frac{\#\text{ of } asps}{\text{Convex Hull Area}} \tag{19}$$

A high density suggests redundancy, whereas a low density may indicate gaps in coverage.

Third, the *Standard Deviation of Assignment Distances (σ_{assign})* measures the variability in distances between demand points and their assigned service points. A lower value indicates more consistent service distances among all demand points. Let $\overline{ADSP}$ be the mean distance calculated in Eq. 16.

$$\sigma_{asp} = \sqrt{\frac{1}{|\mathcal{DP}|} \sum_{i=1}^{|\mathcal{DP}|} (d(x_i, s_i) - \overline{ADSP})^2} \tag{20}$$

Demand Fairness. We evaluate how demand is distributed among the $asps$ using two metrics.

First, metric *Maximum Demand per Service Point* measures whether some *asps* are overloaded with a large number of *dps*, leading to an imbalanced demand distribution.

$$MaxDemand = \max_{j=1\ldots k} |\mathcal{DP}(s_j)| \tag{21}$$

where $\mathcal{DP}(s_j)$ is the set of dps assigned to asp s_j.

Second, metric *Demand-to-Service Point Ratio Standard Deviation* computes the standard deviation in the number of demand points assigned to each service point:

TABLE I: Execution environments and clustering approaches.

Exec. env. / approaches	KM	IBP-M	M-DBC
Classical solver (C)	✓	✓	
Simulated Annealing (SA)	✓		✓
Hybrid (H)	✓		✓
Quantum Annealing (QA)			✓

$$\sigma_{demand} = \sqrt{\frac{1}{|\mathcal{ASP}|} \sum_{i=1}^{|\mathcal{ASP}|} (n_i - \bar{n})^2} \tag{22}$$

where n_i is the number of demand points assigned to service point i, and $\bar{n}$ is the average demand per service point. A high variance suggests an imbalanced allocation of demand points to service points.

D. Competitor approaches

We compare M-DBC with K-Medoids [32] and a variant of P-Median [31] we have developed specifically to adapt the approach to the problem at hand.

K-Medoids (KM in the following) has been selected for the following reasons:

1) It represents a standard, well-understood, classical clustering approach applicable to the $\mathcal{CSP}$ set.
2) It focuses solely on the spatial distribution and representativeness of the *candidate service points* themselves, completely ignoring the locations and distribution of the *dps*. This helps quantify the benefit gained by our approach, which, instead, explicitly incorporates demand information.
3) The existence of a QUBO formulation [32] offers the possibility of a direct comparison using annealing-based solvers.

P-Median has been chosen for its extensibility that has allowed us to introduce in its formulation the concept of importance values that we have already used for M-DBC, biasing the general model toward the selection of the most important $csps$. We call the new variant *Importance Based P-Median* (IBP-M). In its formulation, we use a demand-service distance matrix $F = [f_{ij}]$, where f_{ij} is the distance between dp_i and asp_j, and a set of importance values w_j for each $csp_j \in \mathcal{CSP}$. The objective is to minimize the total assignment cost, that is, the sum of the distances between each dp and its asp, weighted using the importance values w_j defined in Section V-B:

$$D_{\text{tot}} = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{f_{ij}}{w_j^{\phi}} x_{ij} \tag{23}$$

where $x_{ij} \in \{0, 1\}$ indicates whether dp_j is assigned to asp_i and ϕ is a tunable exponent that controls the influence of the importance values (e.g., $\phi = 1$ for direct weighting, $\phi > 1$ to amplify the effect of importance values).

IBP-M allows us to i) favor antennas located in high-demand regions rather than treating all locations equally, ii)

adapt to different urban layouts by dynamically adjusting to the spatial distribution of taxi calls, and iii) maintain a balance between distance minimization and demand concentration, ensuring a practical deployment strategy. This model is expected to serve as a valuable benchmark that represents a classical optimum for our problem. For this reason, we use it as a target reference to evaluate the quality of our quantum-based model. However, it does not scale. In fact, it requires optimization variables not only for selecting the N candidate service points (z_i) but also for assigning each of the M demand points to a selected service point (x_{ij}). This results in a model complexity, represented by the number of variables, that scales with $O(N+NM)$, with M that tends to be very large in real scenarios. In contrast, our M-DBC formulation defines the core optimization problem primarily on the selection variables z_i, leading to a QUBO model whose size scales with $O(N^2)$, where N is typically significantly smaller (at least one order of magnitude) than M. Therefore, given an edge infrastructure that tends to be stable over time (and, therefore, a fixed N), the number of variables for IBP-M tends to grow linearly and rapidly with the growth of the number of demand points, while this number remains constant for M-DBC. This implies that IBP-M becomes computationally intractable for scenarios with a very large number of demand points, which is common in realistic, dynamic edge environments (e.g., tens or hundreds of thousands of users/devices).

During the analysis, to allow the IBP-M baseline to execute within a reasonable timeframe, we decided to keep the number of demand points per time split relatively small (e.g., typically in the order of hundreds to a few thousand). In such datasets, IBP-M, by directly optimizing the assignment of every demand point, can achieve very good performance, particularly on metrics like ADSP. However, this may represent an overfitting to the specific and limited set of demand points present in the sample, potentially lacking robustness to variations in demand patterns.

Our goal for M-DBC is not necessarily to strictly outperform IBP-M on every metric within this limited, potentially overfitted scenario. Instead, we aim for M-DBC to achieve *comparable* high-quality results thus demonstrating the utility of a quantum-based approach for the considered problem, while offering the crucial advantage of scalability with the growth of the number of demand points, where IBP-M would be computationally infeasible. The results presented in Section VI should be interpreted in light of this scalability trade-off.

KM and IBP-M have been compared with our M-DBC approach using the execution environments shown in Table I. IBP-M has been executed only on a classical solver, PuLP. Given its scalability problems, we did not consider adapting its implementation to a quantum environment. KM is available both as a classical algorithm (using the scikit Python library) and as a QUBO formulation [32]. The latter is presented in a simulated annealing and in a hybrid context (we have performed the experiments using the parameters in [32]). We tried to execute it also on a pure quantum annealer, but could not obtain a correct execution because we experienced a high number of chain breaks. Finally, M-DBC has been conceived as a quantum-based approach and, therefore, has not been executed in the classical context. Quantum, hybrid, and simulated annealing executions utilized the D-Wave Ocean SDK [35].

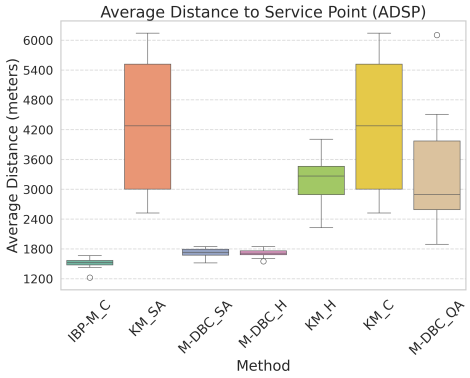

Fig. 2: Box plot of ADSP across runs.

In the following, the different combinations of approaches and execution environments are identified by concatenating the acronym of each approach with the capital letter suffixes that in Table I indicate the execution environments.

Finally, to execute the experiments, the clustering objective for all the methods was to select $k = 20$ *asps* from the set of *csps*. Moreover, we set $d_{min} = r = 4000\,\mathrm{m}$ unless otherwise specified, $c_p = 6.0, c_s = 1, \lambda = 3.0, \lambda_c = 1.0$, and $\phi = 0.2$ (for IBP-M), while for the QUBO implementation of K-Medoids (KM_SA/H) presented in [32] we used the configuration of parameters suggested by the authors.

VI. Experiments

Our evaluation focuses on two main aspects: i) the quality of generated solutions and ii) the scalability of the approaches. The implementation of M-DBC, along with the replication package for the experiments, is available at [36].

A. Solution Quality

To evaluate solution quality in a setup where IBP-M_C could deliver significant results, we restricted the analysis to individual dataset splits, ensuring that the input size remained manageable for IBP-M_C.

Service Placement. Figure 2 shows the results for the metric ADSP averaged over the results obtained in each split. As expected, IBP-M_C provides the best performance, achieving the lowest average distances with minimal variability among runs. This was anticipated, as IBP-M_C serves as the reference for optimal clustering quality, being based on an assignment model that directly minimizes total distances, as discussed in Section V-D. Both M-DBC_SA and M-DBC_H approaches achieve results close to IBP-M_C, demonstrating their ability to approximate high-quality clustering while maintaining scalability. In contrast, KM_C and KM_SA, which ignore the demand distribution, perform significantly worse,

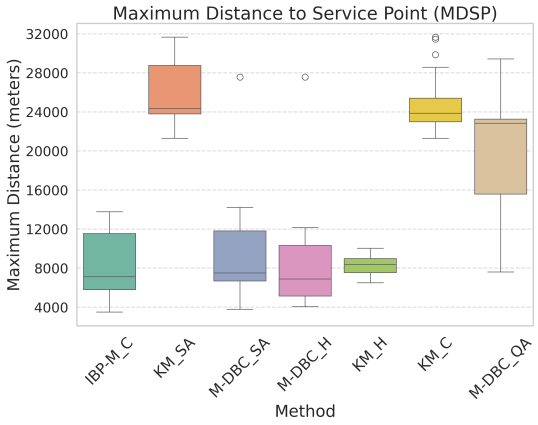

Fig. 3: Box plot of MDSP across runs.

showing much higher average distances and greater variability. M-DBC_QA shows a clear performance drop compared to M-DBC_SA and M-DBC_H, suggesting that pure quantum annealing, under the tested conditions, is not yet able to match hybrid and simulated annealing performance.

Figure 3 reports the results for metric MDSP aggregated in all the 10 splits. IBP-M_C again establishes the best performance, with the lowest maximum distances and moderate variability. M-DBC_SA and M-DBC_H closely follow, confirming their robustness not only in average but also in worst-case scenarios. KM-based approaches (KM_C and KM_SA) perform poorly, exhibiting very high maximum distances, indicating their inability to ensure acceptable service quality for all demand locations. Also, for MDSP, M-DBC_QA shows inferior results compared to its hybrid and simulated annealing counterparts, confirming the observations made on ADSP.

Table II reports detailed MDSP results for three randomly selected dataset splits, providing additional insights beyond the aggregated box plots. As expected, IBP-M_C consistently achieves the best results in terms of average MDSP across all splits, confirming its role as a reference model for clustering quality. In Split 1, IBP-M_C achieves an average MDSP of 1531.59 meters. M-DBC_SA and M-DBC_H show competitive results, achieving average MDSP values close to IBP-M_C across all splits while often providing better maximum MDSP values. For example, in Split 1, M-DBC_SA limits the maximum MDSP to 3774.20 meters, significantly lower than IBP-M_C. Moreover, M-DBC approaches maintain good distance performance despite covering larger convex hull areas, confirming their ability to create well-distributed topologies while preserving proximity between users and service points. In Split 1, M-DBC_SA limits the maximum MDSP to 3774.20 meters, significantly better than IBP-M_C's 6154.95 meters. Similarly, in Split 3, M-DBC_SA and M-DBC_H maintain maximum distances around 10000 meters, slightly better than IBP-M_C. This indicates that M-DBC formulations are not only close in average quality but can outperform IBP-M_C in worst-case scenarios.

Demand Coverage. Table III reports the average coverage results among all dataset splits. M-DBC_H, M-DBC_SA, and IBP-M_C achieve very similar coverage values, all above 95%. M-DBC_H shows the highest mean coverage, closely followed by M-DBC_SA and IBP-M_C, with minor differences and low variability. In contrast, M-DBC_QA exhibits a clear drop in performance, achieving around 79.6% coverage, highlighting the current limitations of pure quantum annealing.

Spatial Distribution. Table III reports the results for the spatial distribution metrics. M-DBC_H, M-DBC_SA, and IBP-M_C show very similar behavior in terms of Convex Hull Area, all covering around 700–750 km^2, confirming that the service points selected by M-DBC maintain a spatial spread comparable to the reference model IBP-M_C. In contrast, KM_C and KM_SA cover much smaller areas (around 300 km^2), suggesting a concentration of service points in limited regions and reflecting their poor coverage performance. Interestingly, KM_H achieves a Convex Hull Area of approximately 943 km^2, comparable or even larger than M-DBC variants, indicating a more balanced spatial spread in this case.

The density values further confirm these trends: KM_C and KM_SA exhibit much higher densities due to clustering in small areas, while M-DBC variants, IBP-M_C, and KM_H maintain balanced, lower densities.

The standard deviation of the ASP distances highlights additional differences. M-DBC_H, M-DBC_SA, and IBP-M_C show low variability (around 800–900 meters), indicating a uniform spatial distribution of service points. KM_H also achieves better results than other KM-based methods, with a moderate standard deviation (1876 meters), whereas KM_C and KM_SA display extremely high variability, suggesting uneven and inefficient service point placement. M-DBC_QA deviates from the other M-DBC variants, showing a significantly higher standard deviation and more compact Convex Hull Area, consistent with its worse performance across the other quality metrics.

Demand Fairness. Regarding demand fairness, all methods based on demand-aware clustering (M-DBC variants and IBP-M_C) achieve balanced results, with similar maximum demand per ASP and comparable demand ratio standard deviations. KM_SA achieves very good fairness, obtaining the lowest demand ratio standard deviation (23.18), slightly better than M-DBC and IBP-M_C. This suggests that despite its limitations in coverage and spatial distribution, KM_SA is able to distribute demand more evenly across service points. KM_C, instead, shows higher variability, and KM_H presents moderate results.

B. Execution Time

Table IV presents the execution time of our experiments concerning the classical solvers and simulated annealing approaches. The experiments were conducted on a DELL Latitude 5440 equipped with a 13th Gen Intel(R) Core(TM) i7 CPU and 16 GB of DDR5 RAM. As shown in the table, IBP-M_C exhibits a strong dependency on the number of considered demand points. Although it completes in approximately 42 seconds for 2122 demand points and around 488 seconds

TABLE II: MDSP across different methods and splits. Best overall values per column are bolded for each split.

Split	Method	Min MDSP (m)	Avg MDSP (m)	Max MDSP (m)	Convex Hull Area (km^2)
Split 1	IBP-M_C	22.30	**1531.59**	6154.95	488.19
	KM_C	**18.45**	3169.20	23724.87	202.34
	KM_H	**18.45**	3416.96	6896.61	**637.24**
	KM_SA	**18.45**	3169.20	23724.87	202.34
	M-DBC_H	99.05	1696.94	5859.30	577.51
	M-DBC_QA	**18.45**	2700.54	22775.72	425.64
	M-DBC_SA	**18.45**	1723.99	**3774.20**	553.84
Split 2	IBP-M_C	**16.94**	**1483.27**	**6936.75**	763.66
	KM_C	**16.94**	2729.67	31449.82	217.07
	KM_H	93.62	3131.88	7794.06	**1017.67**
	KM_SA	**16.94**	2729.67	31449.82	217.07
	M-DBC_H	**16.94**	1679.24	12137.71	852.93
	M-DBC_QA	38.58	2132.77	7604.10	717.27
	M-DBC_SA	**16.94**	1657.59	12137.71	864.86
Split 3	IBP-M_C	18.46	**1475.86**	9818.36	629.01
	KM_C	28.71	4948.39	29864.95	169.29
	KM_H	40.36	2910.88	**7458.30**	**824.43**
	KM_SA	28.71	4948.39	29864.95	169.29
	M-DBC_H	28.71	1605.53	9899.01	635.71
	M-DBC_QA	**18.40**	3059.60	23218.02	494.31
	M-DBC_SA	28.71	1558.49	10818.34	632.97

TABLE III: Coverage, Spatial Distribution and Demand Fairness Metrics (Mean $\pm$ Std Dev)

Method	Coverage (%)	Convex Hull Area (km^2)	Density (ASPs/km^2)	Std Dev ASP Distance (m)	Max Demand per ASP	Demand Ratio Std Dev
M-DBC_H	**96.68** $\pm$ 1.81	703.3 $\pm$ 167.0	0.0305 $\pm$ 0.0069	**807.2** $\pm$16.5	99.0 $\pm$ 29.3	25.39 $\pm$ 9.71
M-DBC_SA	96.19 $\pm$ 2.27	746.8 $\pm$ 180.6	0.0288 $\pm$ 0.0077	869.3 $\pm$ 321.3	99.7 $\pm$ 29.3	26.12 $\pm$ 8.12
M-DBC_QA	79.58 $\pm$ 7.90	489.2 $\pm$ 144.9	0.0591 $\pm$ 0.0187	4454.8 $\pm$ 2346.7	109.1 $\pm$ 45.4	26.88 $\pm$ 8.54
IBP-M_C	95.75 $\pm$ 1.67	741.2 $\pm$ 184.0	0.0297 $\pm$ 0.0073	892.1 $\pm$ 88.5	107.8 $\pm$ 36.8	25.47 $\pm$ 6.76
KM_C	65.99 $\pm$ 10.38	318.9 $\pm$ 112.8	0.0807 $\pm$ 0.0261	5780.1 $\pm$ 1541.8	131.1 $\pm$ 51.2	29.28 $\pm$ 11.02
KM_SA	68.00 $\pm$ 11.90	298.8 $\pm$ 101.5	0.0868 $\pm$ 0.0242	5619.0 $\pm$ 1624.4	**93.1** $\pm$ 35.5	**23.18** $\pm$ 8.48
KM_H	54.59 $\pm$ 12.49	**943.0** $\pm$ 205.7	**0.0231** $\pm$ 0.0052	1876.3 $\pm$ 418.0	98.5 $\pm$ 33.3	27.72 $\pm$ 6.80

TABLE IV: Execution Time Comparison.

Metric	IBP-M_C			KM_C	KM_SA	M-DBC_SA
$\lvert\mathcal{DP}\rvert$	2122	7575	14187	all cases	all cases	all cases
Execution time	42.39 $\pm$ 0.18 s	488.22 $\pm$ 17.46 s	Timeout 15 min	12.4 $\pm$ 0.06 s	65.3 $\pm$ 2.4 s	28.9 $\pm$ 1.2 s

for 7575 demand points, it exceeds the set timeout limit of 15 minutes when the number of demand points rises above 14,000. This clearly demonstrates that IBP-M_C does not scale well with increasing demand size.

In contrast, the execution times of KM_C, KM_SA, and M-DBC_SA are effectively independent of the number of demand points, highlighting the scalability of these approaches. M-DBC_SA, in particular, maintains a stable execution time at around 29 seconds across all tested scenarios, making it suitable for large-scale dynamic edge environments.

Hybrid approaches are not included in the table, as their execution time is fixed to 5 seconds, after which the best solution found is retrieved. For pure quantum executions, the reported time to solution includes the model creation time, QPU programming time, and annealing time, with model creation being the dominant factor (in the order of seconds), while QPU programming and annealing occur in microseconds.

The D-Wave system provides a detailed breakdown of QPU timings. For example, in our M-DBC_QA runs, representative values indicate a **QPU access time** (`QPU_ACCESS_TIME`) ranging approximately from **290 ms to 305 ms** for a single call to the QPU. This total access time is made up of the following main components:

- `QPU_PROGRAMMING_TIME`: Approximately 18.6 ms (consistent across runs).
- `QPU_SAMPLING_TIME`: Ranges from 273 ms to 287 ms. This time is for collecting all samples (typically 1000 in our configuration) from the QPU. The breakdown per individual sample is as follows:

Component per Sample	**Duration**
`QPU_ANNEAL_TIME_PER_SAMPLE`	20.0 µs
`QPU_READOUT_TIME_PER_SAMPLE`	230–245 µs
`QPU_DELAY_TIME_PER_SAMPLE`	$\sim$20.6 µs

The `TOTAL_POST_PROCESSING_TIME` (on QPU side) is negligible, typically in the range of tens to a hundred microseconds. This detailed timing data under-

scores that while the quantum annealing process itself (QPU_ANNEAL_TIME_PER_SAMPLE) is extremely fast, the overheads associated with programming the QPU, repeated readouts for sampling, and system access latencies constitute the bulk of the QPU interaction time for a single problem submission.

VII. Discussion

This section analyzes the experimental results presented in Section VI, interpreting their implications for clustering edge topologies. We focus on the performance of the proposed M-DBC formulation across different solvers (Hybrid, Simulated Annealing, Quantum Annealing) and compare it against the classical IBP-M and K-Medoids baselines.

*a) **Performance Analysis of M-DBC Results**:* The results provide valuable insights into the performance of our M-DBC model using different optimization approaches:

Hybrid (M-DBC_H): the D-Wave hybrid CQM solver consistently delivered high-quality solutions among nearly all metrics. It achieved low ADSP, the best (lowest) median MDSP, excellent coverage, wide spatial spread (large convex hull, low density), and good demand fairness (low max demand and ratio std dev). This demonstrates the practical effectiveness of current hybrid quantum-classical techniques for solving complex, constrained optimization problems like M-DBC at a relevant scale.

Simulated Annealing (M-DBC_SA): SA proved to be a very strong classical heuristic for the M-DBC QUBO. It achieved results comparable to M-DBC_H in ADSP, coverage, and spatial distribution (area/density). Its MDSP was slightly higher and more variable than M-DBC_H, and its demand fairness metrics were similar. This confirms that SA can find high-quality solutions for M-DBC, serving as a robust baseline for the QUBO formulation.

Quantum Annealing (M-DBC_QA): the pure quantum annealing approach, using the parameters specified, yielded suboptimal results compared to hybrid and SA methods for M-DBC on most key metrics (ADSP, MDSP, coverage, spatial area). Although still outperforming the variants of K-Medoids in some aspects (e.g. ADSP), its performance indicates challenges that include the difficulty of embedding the specific M-DBC QUBO structure into the QPU using the minorminer tool included in the D-WAVE Ocean SDK [35], noise impact, or the need for more customized parameter tuning (annealing schedules, chain strengths adjusted specifically for M-DBC). It is useful to note that in our analysis we did not adapt the parameters to each specific dataset split, so it is probable that with an ad-hoc setup for each split, better results can be obtained. The most relevant limiting factor is surely related to the frequency of chain breaks, also with relatively high chain strength parameter, due probably to the presence of long qubit chains in the embedding.

A. Comparative Analysis: M-DBC vs. Baselines

When comparing the M-DBC family with the baselines:

Importance Based P-Median (IBP-M_C): The comparison between the results shown by M-DBC_H/SA and IBP-M_C highlights the trade-off between the two: IBP-M_C minimizes the average, while M-DBC offers a more balanced solution considering the spread and worst-case distances.

K-Medoids: the K-Medoids objective, even when solved effectively by the hybrid engine (KM_H), fails to adequately capture the core requirements of placing service points near demand points in the edge context (low ADSP, high coverage). This strongly validates the necessity of a custom formulation like M-DBC, which explicitly incorporates demand proximity, importance, and coverage aspects into its objective function, leading to superior overall performance when run on the same hybrid platform (M-DBC_H vs. KM_H).

VIII. Conclusions

This work presents and validates the M-DBC formulation as an effective approach to edge topology clustering, offering a good balance between cluster quality properties, worst-case guarantees, spatial distribution, and demand fairness. M-DBC, when used with hybrid quantum-classical methods (M-DBC_H) or simulated annealing (M-DBC_SA), clearly outperforms traditional K-Medoids and offers a more robust and spatially aware alternative to classical P-Median methods like IBP-M_C. While quantum annealing (M-DBC_QA) still needs improvement, the results highlight the effectiveness of M-DBC for building efficient edge computing systems.

M-DBC can be applied to various dynamic edge-computing scenarios where rapid user demand shifts and mobility require frequent reconfiguration of service clusters. For example, in emerging 6G networks [37], AI-based services are expected to migrate toward user hotspots via micro-data centers [38]. M-DBC supports this by enabling fast re-clustering with execution times independent of demand size. Similar needs arise in vehicular edge computing [39], [40], where nodes must adapt to changing traffic to maintain low-latency services. In industrial IoT [41], [42], M-DBC helps dynamically reassign resources to meet fluctuating demands without manual control, ensuring stable performance.

Future work may explore capacity-augmented variants of P-Median models and introduce a pre-processing step that clusters demand points to improve scalability for edge computing. Additionally, validating solutions in realistic simulation environments would help assess their practical applicability.

IX. Acknowledgments

This work has been partially supported by projects CN-HPC-S10, QUASAR (grant 2022T2E39C - PRIN 2022), and MICS (grant PE00000004). We also thank D-WAVE, which has kindly offered free execution time on their quantum annealing resources.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[2] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[3] S. Bolettieri, R. Bruno, and E. Mingozzi, "Application-aware resource allocation and data management for mec-assisted iot service providers," *Journal of Network and Computer Applications*, vol. 181, p. 103020, 2021.

[4] X. Zhang, Z. Li, C. Lai, and J. Zhang, "Joint edge server placement and service placement in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 261–11 274, 2022.

[5] L. Baresi, D. Y. X. Hu, G. Quattrocchi, and L. Terracciano, "Neptune: a comprehensive framework for managing serverless functions at the edge," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 19, no. 1, pp. 1–32, 2024.

[6] A. A. Vali, S. Azizi, and M. Shojafar, "RESP: A recursive clustering approach for edge server placement in mobile edge computing," *ACM Transactions on Internet Technology*, vol. 24, no. 3, pp. 1–25, 2024.

[7] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[8] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[9] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[10] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[11] C. McGeoch, *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*. Morgan & Claypool Publishers, 2014.

[12] S. Reale and E. D. Nitto, "Quantum graph pursuit: Analysis of the advantages and challenges of a quantum dynamic combinatorial optimization model from a software developer perspective," in *2024 IEEE International Conference on Quantum Software*, 2024, pp. 24–34.

[13] S. Dong, Y. Xia, and J. Kamruzzaman, "Quantum particle swarm optimization for task offloading in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 8, pp. 9113–9122, 2022.

[14] J. A. Ansere, D. T. Tran, O. A. Dobre, H. Shin, G. K. Karagiannidis, and T. Q. Duong, "Energy-efficient optimization for mobile edge computing with quantum machine learning," *IEEE Wireless Communications Letters*, vol. 13, no. 3, pp. 661–665, 2023.

[15] J. A. Ansere, E. Gyamfi, V. Sharma, H. Shin, O. A. Dobre, and T. Q. Duong, "Quantum deep reinforcement learning for dynamic resource allocation in mobile edge computing-based iot systems," *IEEE Transactions on Wireless Communications*, vol. 23, no. 6, pp. 6221–6233, 2023.

[16] M. Bey, P. Kuila, B. B. Naik, and S. Ghosh, "Quantum-inspired particle swarm optimization for efficient iot service placement in edge computing systems," *Expert Systems with Applications*, vol. 236, p. 121270, 2024.

[17] F. Glover, G. Kochenberger, R. Hennig, and Y. Du, "Quantum bridge analytics i: a tutorial on formulating and using qubo models," *Annals of Operations Research*, vol. 314, no. 1, pp. 141–183, Jul 2022.

[18] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[19] B. Xiang, J. Elias, F. Martignon, E. Di Nitto, and D. Niyato, "Game theoretic resource planning and request scheduling in mobile edge computing networks," in *2023 IFIP Networking Conference (IFIP Networking)*, 2023, pp. 1–9.

[20] L. Baresi and G. Quattrocchi, "Towards vertically scalable spark applications," in *Euro-Par 2018: Parallel Processing Workshops: Euro-Par 2018 International Workshops, Turin, Italy, August 27-28, 2018, Revised Selected Papers 24*. Springer, 2019, pp. 106–118.

[21] J. Xie, B. K. Szymanski, and X. Liu, "Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process," in *2011 IEEE 11th International Conference on Data Mining Workshops*, 2011, pp. 344–349.

[22] S. Long, C. Wang, W. Long, H. Liu, Q. Deng, and Z. Li, "An efficient task scheduling algorithm in the cloud and edge collaborative environment," *Chinese Journal of Electronics*, vol. 33, no. 5, pp. 1296–1307, 2024.

[23] L. Pan, X. Liu, Z. Jia, J. Xu, and X. Li, "A multi-objective clustering evolutionary algorithm for multi-workflow computation offloading in mobile edge computing," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1334–1351, 2023.

[24] I. Ullah and H. Y. Youn, "Task classification and scheduling based on k-means clustering for edge computing," *Wireless Personal Communications*, vol. 113, no. 4, pp. 2611–2624, 2020.

[25] Y. Zhang, W. Zhang, H. Hao, and K. Zhang, "Cluster caching strategy based on user characteristics in edge networks," in *Proc. of the 24th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2023, pp. 36–41.

[26] C. Qi, W. Wei, Y. Feng, T. Meixia, and Z. Zhaoyang, "Content caching oriented popularity prediction: A weighted clustering approach," *IEEE Communications Letters*, vol. 25, no. 12, pp. 3916–3920, 2021.

[27] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Zomaya, S. Foufou, and A. Bouras, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 267–279, 2014.

[28] R. Manzini and E. G. and, "Optimization models for the dynamic facility location and allocation problem," *International Journal of Production Research*, vol. 46, no. 8, pp. 2061–2086, 2008.

[29] A. Suzuki and Z. Drezner, "The p-center location problem in an area," *Location Science*, vol. 4, no. 1, pp. 69–82, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0966834996000125

[30] L. Mrkela and Z. Stanimirović, "A bi-objective maximal covering location problem: a service network design application," in *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2020, pp. 1–7.

[31] M. S. Daskin and K. L. Maass, *The p-Median Problem*. Cham: Springer International Publishing, 2015, pp. 21–45.

[32] C. Bauckhage, N. Piatkowski, R. Sifa, D. Hecker, and S. Wrobel, "A qubo formulation of the k-medoids problem." in *LWDA*, 2019, pp. 54–63.

[33] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 316–324. [Online]. Available: https://doi.org/10.1145/2020408.2020462

[34] OpenCellID, "OpenCellID: The world's largest open database of cell towers," 2025, accessed: January 2, 2025. [Online]. Available: https://opencellid.org

[35] D-Wave Systems, "D-wave ocean software development kit," https://github.com/dwavesystems/dwave-ocean-sdk, 2024, gitHub repository.

[36] S. Reale, G. Quattrocchi, L. Baresi, and E. D. Nitto, "M-dbc implementation and replication package," https://github.com/SimoneReale/QuantumTopologyClustering, 2025, accessed: 2025-04-04.

[37] M. A. Khan, I. Ahmad, and F. Shabbir, "Survey on ai-enabled resource management for 6g networks," *Computer Networks*, vol. 235, p. 109787, 2023.

[38] M. A. Khan, M. A. Imran, and Q. H. Abbasi, "Towards 6g internet of things: Recent advances, use cases, and open challenges," *Internet of Things*, vol. 19, p. 100560, 2022.

[39] M. K. Hasan, N. Jahan, M. Z. A. Nazri, S. Islam, M. A. Khan, A. I. Alzahrani, N. Alalwan, and Y. Nam, "Federated learning for computational offloading and resource management of vehicular edge computing in 6g-v2x network," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 3827–3847, 2024.

[40] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile networks and applications*, vol. 26, pp. 1145–1168, 2021.

[41] M. Sharma, A. Tomar, and A. Hazra, "Edge computing for industry 5.0: Fundamental, applications and research challenges," *IEEE Internet of Things Journal*, 2024.

[42] S. Bolettieri, R. Bruno, and E. Mingozzi, "A latency sensitive and agile iiot architecture with optimized edge computing," *Journal of Network and Computer Applications*, vol. 212, p. 103741, 2025.