Data and Metadata. 2026; 5:1328
doi: 10.56294/dm20261328

ORIGINAL

Quantum Optimization for Intelligent User Data Allocation in Distributed Cloud Infrastructure

Optimización cuántica para la asignación inteligente de datos de usuario en infraestructuras de nube distribuidas

Viji C[1], Rajkumar[1], R. Stephen[2], Gobinath R[2], Balusamy Nachiappan[3], Prabhu Shankar B[4]

[1]Alliance School of Advanced Computing, Alliance University. Bangalore, India.
[2]Department of Computer Science, Christ University. Bangalore, India.
[3]Prologis. USA.
[4]Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology. Chennai, Tamil Nadu, India.

Cite as: CV, Rajkumar, SR, RG Nachiappan B, Shankar BP. Quantum Optimization for Intelligent User Data Allocation in Distributed Cloud Infrastructure. Data and Metadata. 2026; 5:1328. https://doi.org/10.56294/dm20261328

Submitted: 19-08-2025 **Revised:** 24-09-2025 **Accepted:** 24-12-2025 **Published:** 01-01-2026

Editor: Dr. Adrián Alejandro Vitón Castillo

Corresponding Author: Viji C

ABSTRACT

In a modern environment in which cloud computing is distributed globally, optimizing the placements of user data in the different cloud locations would ensure that the data can be accessed with minimum latency and fast access. Traditional heuristic and machine-learning methods are often prohibitively expensive to scale and have a hard time adjusting to a dynamic cloud environment. Scalable and adaptive optimization strategies are required when user demands and data volumes grow very fast. There is a fair likelihood that quantum-based methods, particularly, the metaheuristic methods, are an alternative that can effectively react to dynamic conditions. The proposed model will be based on Quantum Particle Swarm Optimization when moving user data to the most suitable places within distributed cloud canters. Through quantum-inspired probabilistic search, the algorithm becomes more adaptive and more efficient than traditional ones. Experiments, based on these simulations of the user request in the case of a geo-distributed cloud, have shown a significant reduction in latency up to 28 % and better load balancing compared to the traditional approaches. Altogether, these results highlight the prospects of quantum computing when it comes to improving the efficiency and responsiveness of cloud infrastructure. The primary strength of the QPSO is that it can be easily modified to facilitate the rapid response to the rapidly changing environment to allow access to the distributed cloud systems in an efficient way and with a low latency.

Keywords: Quantum Optimization; Data Placement; Distributed Cloud Computing; Latency Reduction; Quantum-Inspired Algorithms.

RESUMEN

En un entorno moderno donde la computación en la nube se distribuye globalmente, optimizar la ubicación de los datos de usuario en las diferentes ubicaciones de la nube garantizaría un acceso rápido y con mínima latencia. Los métodos tradicionales de heurística y aprendizaje automático suelen ser prohibitivamente costosos de escalar y presentan dificultades para adaptarse a un entorno de nube dinámico. Se requieren estrategias de optimización escalables y adaptativas cuando las demandas de los usuarios y los volúmenes de datos crecen rápidamente. Es muy probable que los métodos cuánticos, en particular los metaheurísticos, sean una alternativa que pueda reaccionar eficazmente a las condiciones dinámicas. El modelo propuesto se basará en la optimización por enjambre de partículas cuánticas al trasladar los datos de usuario a las ubicaciones más adecuadas dentro de los centros de nube distribuidos. Mediante la búsqueda probabilística de inspiración cuántica, el algoritmo se vuelve más adaptativo y eficiente que los tradicionales. Experimentos basados en estas simulaciones de la solicitud del usuario en una nube geodistribuida han demostrado una reducción significativa de la latencia de hasta un 28 % y un mejor equilibrio de carga en comparación con los enfoques tradicionales. En conjunto, estos resultados resaltan las perspectivas de la computación cuántica para mejorar la eficiencia y la capacidad de respuesta de la infraestructura en la nube. La principal ventaja del QPSO es su fácil modificación para facilitar una respuesta rápida a un entorno en constante cambio y permitir el acceso a los sistemas de nube distribuida de forma eficiente y con baja latencia.

Palabras clave: Optimización Cuántica; Ubicación de Datos; Computación en la Nube Distribuida; Reducción de Latencia; Algoritmos de Inspiración Cuántica.

INTRODUCTION

Cloud-based applications are expanding at an unprecedented rate, propelled by digital transformation and the Internet of Things (IoT). Users now expect virtually immediate access to services that include video streaming, e-commerce, real-time analytics, and AI-driven systems. Such expectations intensify the necessity of minimizing latency and maximizing data accessibility in geo-distributed cloud environments.(1) To meet these needs, cloud service providers implement a data centre at more than one geographic location, which increases fault tolerance and expandability at the cost of new challenges. Where and how the user data should be stored to be accessible in the best way is still an intricate and dynamic issue. Traditional data placement and load-balancing strategies, often based on heuristic algorithms or static rules, are inadequate for managing rapidly changing user demands, varying workloads, and heterogeneous network conditions.(2) The main drawback of these methods is that they do not process the combinatorial complexity of decisions on optimal placement. They are also challenged in striking the trade-offs between latency, bandwidth consumption, and load spread across the servers. With data requests arriving from diverse global regions, suboptimal placement can lead to delayed access, increased network congestion, and degraded quality of service (QoS).

In dealing with these challenges, the proposed paper has adopted a new strategy that uses quantum optimization techniques in allocating user data to the most ideal cloud data centres in an intelligent method. Quantum-inspired algorithms such as Quantum Particle Swarm Optimization (QPSO) offer promising solutions by exploiting quantum superposition, entanglement, and probabilistic search mechanisms to explore vast solution spaces efficiently.(3) Such methods offer a paradigm shift to optimization, dramatically speeding up convergence and enhancing the quality of placement decision-making in distributed problems.

This paper introduces a quantum-classical model that combines quantum optimization with the historical cloud resource management suites. The framework determines latency-optimal data placements by transmitting data-centre characteristics, user access patterns, and network characteristics dynamically and determining optimal data placements based on received results.(4) The major contributions of this work:

Quantum Particle Swarm Optimization (QPSO) is suggested as quantum-inspired optimization system to efficiently locate user data in the geo-distributed cloud system. A specific dataset that represents geo-distributed pattern of user requests is created to measure the performance of the proposed framework. It should be mentioned that it is used on real-time data and validation is performed under dynamic and latency-sensitive conditions. It designs a new architecture based on quantum-inspired optimization combined with distributed cloud environments to realize ideal load distribution and access latency. It has been demonstrated that it can scale to very large cloud environments and due to this fact, the question arises whether it is capable of supporting real-world workloads. By combining quantum computation with cloud resource management, the framework establishes the basis for smart and dynamic cloud infrastructures, which can scale to the requirements of the next-generation applications.

Placing data and services efficiently in geo-distributed cloud and edge environments, has been the subject of a decade of research. Earlier research has explored the various optimization paradigms such as classical heuristics, evolutionary computation, reinforcement learning, and most recently quantum-inspired metaheuristics. It discusses the most topical works in these dimensions, their problem statements, methodologies, strengths, and limitations, as well as the open gaps that support the aim of the proposed framework.

Quantum-inspired Particle Swarm Optimization (QPSO) has proved to be an effective metaheuristic to tackle complex, dynamic placement problems in which classical heuristics fail. An approach to the placement of IoT services in edge computing was proposed by Bey et al.(5), and it was shown that the algorithm can minimize the latency by using probabilistic search to solve the problem in highly dynamic settings. On the same note, Wang et al.(6) used QPSO to schedule the tasks in the device-edge-cloud cooperative networks and reported better adaptability to the fixed heuristics. Both papers are convinced of the better convergence properties and robustness of QPSO in heterogeneous infrastructures, but are only restricted to the service/task placement, and have not investigated the large-scale geo-distributed cloud data placement.

Other variants generalize QPSO to the domain of specific situations. Jmal et al.(7) suggested Guided QPSO to the traveling repairman problem and emphasized that the algorithm is flexible to allow the inclusion of combinatorial optimization. Naik et al.(8) applied QPSO to the energy-saving offloading of tasks in edge computing as a tradeoff of performance with energy usage. A many-objective QPSO to place virtual machines was also presented by Balicki(9), allowing to coordinate various QoS metrics. Although these methods have their advantages, little research has been done to examine latency sensitive, large-scale distributed cloud data distribution.

Both algorithmic and heuristic frameworks have been used to deal with low-latency service placement. We have described a user-aware service placement model introduced by Centofanti et al.(10), in which edge service deployment is dynamically adjusted to minimize the response time, and a PageRank-inspired placement scheme proposed by Wang et al.(11), where regional value estimation balances cost and latency. Latency minimization is a key goal in both works, but is based on deterministic or graph-inspired heuristics, which can be hard to scale in very dynamic workloads.

Cui et al.(12) designed in containerized settings a deep reinforcement learning (DRL) container scheduling framework that could adapt to cluster upgrades with latency guarantees, but Li et al.(13) concentrated on low startup time and resource-efficient container scheduling. Jin et al.(14) also pursued this line but modelled the latency-reliability trade-off of industrial IoT container migration. These works highlight the computational complexity of edge/cloud scheduling but typically have high computational overhead because of model training or environment exploration.

Another direction has been resource-conscious placement. Abdullah et al.(15) suggested a resource-conscious task placement mechanism to improve query latency in an IoT-fog environment, and Elsedimy et al.(16) proposed an energy and QoS conscious VM placement scheme in cloud IaaS. Sharon et al.(17) addressed the problem of efficient energy offloading of data through data grouping that optimized system throughput within finite resources and also consolidated this research line by implementing QPSO to realize the effectiveness of joint resource allocation and offloading tasks. Even though these works manage to optimize the use of energy and resources, they focus mostly on fog/edge IoT environment or on the VM placement scenario.The special issues of geo-distributed data distribution, such as the need to minimize latency, efficiency of replication, and load balancing between many cloud locations, are not sufficiently studied.

Other approaches that have been studied extensively are classical heuristics and nature-inspired approaches. Chitra(18) devised an ideal placement and replication algorithm of SIoT systems between data locality and edge efficiency. The research by Li et al.(19) suggests the best placement policy based on capacity constraints and load balance in distributed clouds over geographical locations and provides useful information regarding the significance of replication and fairness. Najmusher et al.(20) also applied nature-based paradigms further and developed a distributed cloud data-placement strategy inspired by firefly algorithms. They are computationally efficient and typically not as adaptive as quantum-inspired or reinforcement learning paradigms, which prevents their use in highly turbulent, large-scale systems.

Table 1. Comparative analysis for different data placement methods

Author (Year)	Technique Category	Optimization Method	Environment / Scope	Key Metric Improved	Reported Gain	Principal Limitation
Bey et al.(5)	Quantum-inspired metaheuristics	QPSO	IoT service placement in edge computing	Service latency, energy efficiency	Reduced latency by ~22 %	Limited scalability to large-scale heterogeneous IoT
Wang et al.(6)	Quantum optimization for scheduling	QPSO	Device-Edge-Cloud task scheduling	Task completion time	19–24 % faster execution	Higher complexity under multi-user load
Jmal et al.(7)	Combinatorial optimization	Guided QPSO	Traveling Repairman Problem	Route efficiency	Faster convergence	Problem-specific tuning required
Naik et al.(8)	Energy-aware edge optimization	QPSO	Edge computing task offloading	Energy efficiency, resource use	21 % lower energy	Narrow scope, lacks latency analysis
Balicki (9)	Cloud resource allocation	Many-objective QPSO	VM placement in smart cloud	Multi-objective efficiency	Better Pareto spread	Computational overhead high
Abdullah et al.(10)	Resource-aware task placement	Heuristic	Fog computing IoT query	Query latency	~18 % reduction	No quantum techniques applied

Wang et al.[11]	Graph-based optimization	PageRank-inspired	Edge data placement	Latency, cost efficiency	Lower latency by ~25 %	Static assumptions in network model
Cui et al.[12]	Deep learning optimization	DRL	Edge cluster container scheduling	Latency during upgrades	Significant improvement	Training overhead high
Li et al.[13]	Online scheduling	Lightweight scheduling + caching	Edge containers	Startup delay, memory	Reduced startup delay	Trade-off between memory & throughput
Jin et al.[14]	Reliability-latency trade-off	Migration optimization	Industrial edge	Latency & reliability	Balanced trade-off	Migration overhead still notable
Centofanti et al.[15]	User-aware service placement	Multi-factor heuristic	Edge computing	Latency	Noticeable latency gain	No energy-aware modeling
Elsedimy et al.[16]	VM placement in IaaS	Energy & QoS-aware	Cloud datacenters	Energy consumption, QoS	Improved QoS, energy balance	Complexity rises with scale
Sharon et al.[17]	Data offloading	Grouping-based heuristic	IoT edge	Energy efficiency	Lower energy use	Limited real-time adaptability
Chitra[18]	Data replication & placement	Hybrid heuristic	SIoT with Edge	Data availability	Higher availability	Lacks quantum/AI integration
Li et al.[19]	Data placement strategy	Heuristic with load balance	Geo-distributed cloud	Load balancing, capacity	Balanced usage	Slower for dynamic changes
Najmusher et al.[20]	Nature-inspired optimization	Firefly Algorithm	Distributed cloud	Data placement, latency	Improved placement quality	Convergence slower than QPSO

The review of the literature table 1 shows that, although quantum-inspired and heuristic-based algorithms realize latency, energy, and cost improvement, most methods are limited to the environment that is either stationary or semi-stationary. Existing solutions cannot cope with real-time workload bursts, multi-purpose trade-offs and large-scale distributed deployments in combination. In addition, not many approaches use real-time data to validate their results, thus restricting their practical use. Such gaps drive the systematic development of a new framework that will combine quantum-inspired optimization with dynamic real-time data placement strategies to maintain scalability, flexibility, and high-performance in distributed cloud-edge-IoT infrastructures.

METHOD

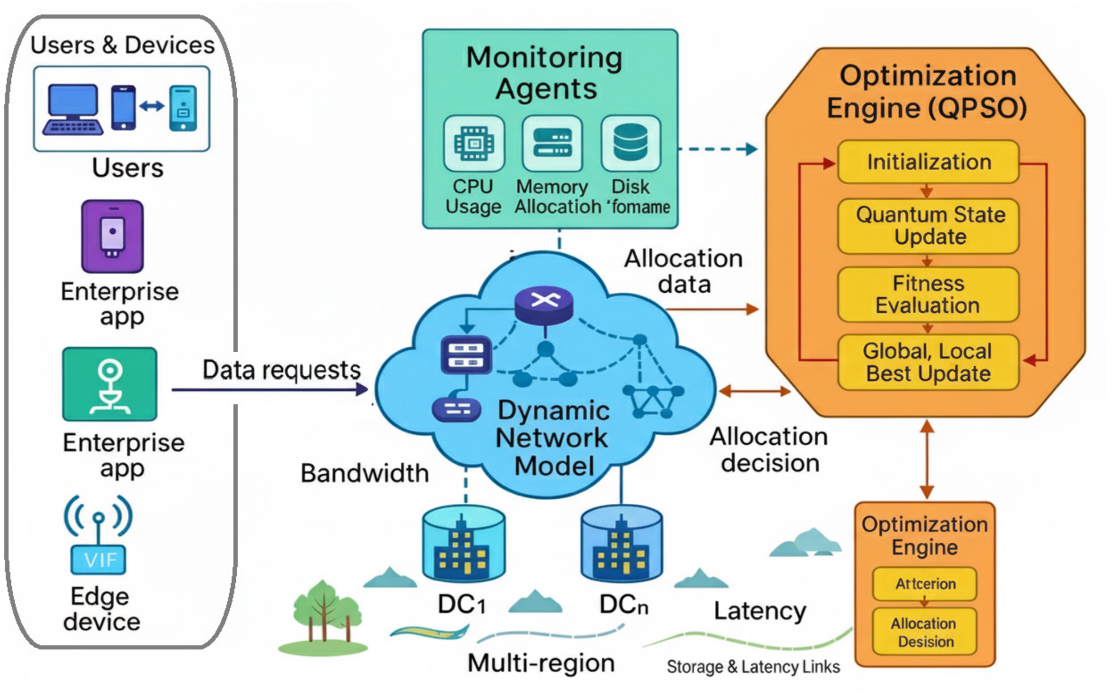

Figure 1. System Architecture Diagram for Quantum Optimized Data Allocation

In this work, it is also introduced that the quantum-motivated optimization algorithms are matched to use geo-distributed cloud data layout to place data in data centres in an optimum way. Its key task is to reduce the latency and maximize the efficiency of data access through quantum-enhanced decision-making abilities during dynamic workloads of concurrent users.

Figure 1 illustrates the scalable and adaptability to process the data of users on the heterogeneous cloud infrastructures. The Dynamic Network Model is central to the design since it is a centralized controller that also provides a medium of communication between all the elements in the system. User requests, issued from dispersed locations and characterized by variable latency, bandwidth, and data quality of service (QoS) requirements, are routed through this model as they arise. The Dynamic Network Model makes constant decision adjustments based on the current situation in the network.

Monitoring Agents service monitors gather and inject data into the dynamic model on bandwidth utilization, latency, jitter, and data traffic patterns. This loop of live monitoring makes the system implement inter-node data routing according to the up-to-date network and data centre status. Simultaneously, the Optimisation Engine, powered by Quantum Particle Swarm Optimisation (QPSO), operates in concert with the network model. QPSO applies quantum mechanics principles to combine the possibility of covering the whole search space that is limited by traditional optimization methods. It explores all possible data-user-data-centres configurations and picks the one that offers the minimum latency with the balanced workload of data centres.

Data Centres (DC_1 to DC_n) constitute the distributed storage and compute nodes of the cloud infrastructure. Each DC reports the available bandwidth, storage, and latency to the dynamic model, which is central in deciding the viability and the cost of provisioning user data at a site. Variations in load capacities and performance charges among centres are considered, whereby the system takes care that none of the data centres is overloaded and at the same time user proximity conditions and performance assurances are maintained.

The QPSO engine communicates with a network model, which provides performance figures and information about the allocation, and then makes the allocation decision and feeds that back to the dynamic model.(21) These decisions outline the methods through which replication or movement of user data is to be achieved, exploiting the placement of data within the whole enterprise infrastructure as an optimization. Figure 1 shows the feedback nature of the looped control flow and decision-making sequence between the dynamic network model and QPSO engine, further highlighting the feedback base and iterative nature of the system.

In the further phases, decisions lead to the movement of data to relevant data centres. The current framework is designed such that it works continuously, constantly dynamically adapting the placement of data based on changes in user behaviour, network health, and the availability of hardware. This architecture, in turn, enhances the efficiency of data distribution, reduces access latency, and builds fault tolerance, which is considered to be essential to operations critical to the financial, healthcare, and IoT industries.

System Model

The quantum-optimized data placement framework is placed in a setting where one can consider some aspects, including the cloud infrastructure, user profiles, monitoring, dynamic network topology, and data placement. Altogether, these aspects define the decision space and constraints that were considered in latency-aware optimization.

Cloud Infrastructure

We consider a globally distributed cloud containing a finite set of data centres:

$$D = \{DC_{1,}DC_{2,\ldots\ldots,}DC_{|D|}\} \qquad (1)$$

Each DC_j is characterized by four resource vectors that are objects of change with time t:

1. Storage capacity Sj(t): available gigabytes for housing user objects or replicas.
2. Outbound bandwidth Bj(t): the maximum sustained throughput (in Mbps) that the centre can deliver to the wide area network without SLA penalties.
3. Inbound bandwidth $B_j^{in}(t)$Ingress capacity, relevant during replica creation or migration.
4. Service latency baseline L_j^0 : the sum of queuing, processing, and local network delay internal to the facility, treated as constant over short horizons.

These values are collected by resource monitoring agents and reported by them to the optimization engine at Δt seconds intervals. As in the case of a new replica being created, we have the corresponding capabilities decrease in real time, thus maintaining consistency between the decision model and the physical environment of the system .

User Profiles

The Active workload is a set of users or an application tenant.

$$U = \{u_1, u_2, \ldots., u_{|u|}\} \qquad (2)$$

A profile for each user u_i contains:

- Geolocation ℓ_i=(lat,lng), converted to networkdistance estimates through real-time latency probes.
- Access frequency $f_i(t)$(req · s^{-1}): a slidingwindow average of read/write operations directed at the user's primary dataset.
- Data size s_i (GB) and mutability rate m_i (updates · s^{-1}), influencing replication overhead.
- Servicelevel objective $L_{max}{}^{i}$ The upper bound on round-trip delay acceptable to the application; violations result in penalty costs or SLA breaches.

In the framework, the dataset of each user is abstracted as a single unit that cannot be further broken, and extension to an object of multiple shards is achieved by creating an entry for each shard against a profile.

Monitoring and Feedback Module

This module continuously gathers real-time network metrics, bandwidth $B_i(t)$, latency $L_{ij}(t)$, and available storage $_{Si}(t)$ from each data centre DC_i at time t. The obtained data are standardized to make the inputs consistent during the optimization process in eqs (3):

$$\widehat{B_\iota}(t) = \frac{B_i(t)}{max_k B_k(t)}, \hat{L}_{ij}(t) = \frac{L_{ij}(t)}{max_k L_{kj}(t)}, \hat{S}_i(t) = \frac{S_i(t)}{max_k S_k(t)} \qquad (3)$$

These metrics form the feedback vector:

$$F(t) = \left[\widehat{B_\iota}(t), \hat{L}_{ij}(t), \hat{S}_i(t)\right] \qquad (4)$$

Which is sent to the QPSO engine to evaluate fitness:

$$Fitness(U_j \rightarrow DC_i) = w_1 \cdot \hat{L}_{ij}(t) + w_2 \cdot (1 - \widehat{B_\iota}(t)) + w_3 \cdot (1 - \hat{S}_i(t)) \qquad (5)$$

The obtained data are standardized to make the inputs consistent during the optimization process.

Dynamic Network Topology

A directed graph that is time-varying represents wide-area connectivity among the users and data centres and links among data centres.

$$G(t) = (V, E(t)), V = U \cup D.) \qquad (6)$$

For every edge ((v,w)∈E(t) we maintain a composite weight:

$$\omega_{uw}(t) = a\ell_{uw}(t) + \beta\ [1/\beta_{uw}(t)] + \gamma\rho_{uw}(t) \qquad (7)$$

Where:

- l_{uw} (t) is one way propagation + queuing delay,
- β_{uw} (t) is residual bandwidth.
- ρ_{uw} (t) is the recent packet loss probability.

Equations (6 and 7) Operators demonstrate preference between latency, throughput, and reliability by encoding their relative values in coefficients refer table 2 in $\alpha+\beta+\gamma=1$, and active probing and passive flows continuously update these weights by adding them to or subtracting them from edge costs, hence guaranteeing that as congestion or routing changes, these values can be reflected urgently in optimization.

Integrated Placement Formulation

Given the above entities, a binary decision variable $x_{ij} \in \{0,1\}$ is defined: $x_{ij}=1$ if user u_i is served from DC_j.

The per-user round-trip latency is:

$$L_i(t) = \sum_j x_{ij} \left(L_j^0 + w_{u_i} DC_j(t) \right) \qquad (8)$$

And must satisfy $L_i\ (t) \le L_{max}{}^{i}$. Data center capacities impose.

$$\sum_i s_i x_{ij} \le S_i(t), \sum_i f_i(t) x_{ij} \le B_j(t) \qquad (9)$$

The optimization engine implemented via quantum-inspired or quantumnative algorithms seeks an equ (8,9) assignment matrix $X=[x_{ij}\ ]$minimizes the weighted average latency $\sum_i{}^{0} f_i\ L_i$. Plus a replica count penalty, while respecting all capacity and SLA constraints. Because the decision landscape changes with G(t) fi (t), the solver executes periodically and on-demand triggers, allowing the system to adapt continuously to shifting traffic and network conditions.

Table 2. Notation Descriptions

Notation	Explanation
ß	The contraction-expansion coefficient controlling the quantum step size (0 < ß < 1)
ϕ	Random value in (0, 1) used to balance the influence between *pBest$_i$* and *gBest*
α	The weight assigned to latency in the objective function.
γ	The weight assigned to load balance in the objective function.
∈	An element of an object
l	Latency (or delay)
ρ	Packetloss probability
Δ	Intervals

Quantum Optimization Technique

The difficulty in placing user data in a global distribution of clouds, as well as the combinatorial explosion of configurable parameters, is directly connected. Every user request has to be placed with a suitable data centre, and the limitation in the form of latency, network bandwidth, storage availability, and quality of service also has to be remembered. Geographically distributed infrastructures, as well as the dynamic behaviour of the users, add to the complexity of this multi-dimensional task to form a non-linear and multi-modal optimization landscape. In this, classical methods of optimization typically suffer from limited scalability and a propensity for premature convergence to local optima, particularly when confronted with extensive and rapidly changing solution spaces.

To alleviate these challenges, we propose a hybrid quantum optimization framework that integrates Quantum Particle Swarm Optimization (QPSO). Inspired by quantum mechanics and founded upon classical Particle Swarm Optimization (PSO), QPSO strengthens global search capability by employing a probabilistic model of particle movement. Through the foregoing reliance on velocity vectors and relying on quantum delta potential wells instead, the technique yields better exploration diversity and convergence stability.[22] Such qualities are essential within the modern, globally dispersed cloud infrastructures. A useful structure for placing the data should not be able to reduce the average latency so far, have to assist in distributing the load, fault recovery, and responses towards variations in user demand and/or system conditions. QPSO meets these requirements by dynamically reprogramming user-to-data-centre mappings, live, in a way that maximizes overall performance measures of the whole system.

Overall, the technique mimics the quantum mechanics phenomena, which has been proven to be scalable, adaptive, and globally convergent, interpreting cloud data allocation problems with a much higher performance than the traditional approaches in complex, distributed computing scenarios.

Quantum Particle Swarm Optimization (QPSO)

In order to address the aforementioned constraints, our investigation employs Quantum Particle Swarm Optimization (QPSO), a quantum-inspired extension of the established Particle Swarm Optimization (PSO) methodology. By integrating quantum-mechanical principles, QPSO replaces the deterministic position-update rules of classical PSO with probabilistic motion governed by quantum-delta potential wells.[23] The peculiar behaviour model of the algorithm (as opposed to varying velocities by each particle, they are introduced into the search space by random jumps) explains the satisfactory results in complex, nonlinear, and multimodal optimization. This mixed behaviour encourages movements between the adjacent areas and controls the likelihood of premature convergence to local optima.

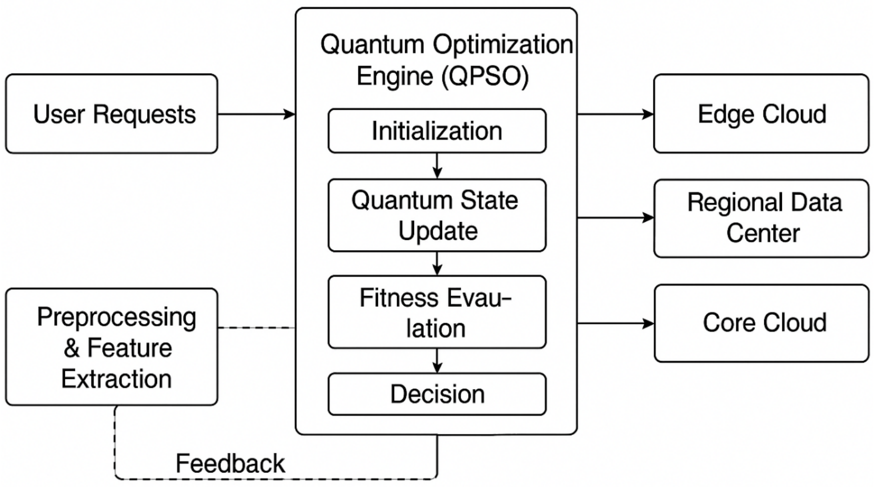

Figure 2. Quantum Optimization Engin

The figure 2 shows a quantum-inspired optimization model on smart allocation of user data in distributed cloud infrastructures. The user requests are initially fed into the Quantum Optimization Engine (QPSO) where they are initialized, updated with the quantum state and with fitness. According to these processes, allocation decisions are taken in order to distribute the workloads between the edge cloud, the regional data centers or the core cloud in order to have an optimal utilization of resources. This process is aided by a preprocessing and feature extraction module which refines the input data as well as providing performance feedback into the optimization cycle. The architecture is more scalable, with less latency and is more energy efficient to support next generation distributed cloud services.

Therefore, QPSO can be proven to be resistant to local minima and provide a methodology particularly suitable to problems of multi-objective cloud data allocation that also require low latency, an equal load distribution, and responsiveness to fluctuating environmental conditions.

Problem Context and Encoding

It presented the user-data allocation problem as an instance of combinatorial optimization.

Given:

- A set of users U={u_1,u_2,….,u_n}.
- A set of data centres D={d_1,d_2,….,d_n}.
- A latency matrix Lij representing the communication cost between user ui and data centre dj

The input includes a user set U and a data centre set D, with the corresponding objective function being to identify a mapping A: U→D mapping that minimizes the total latency at which cloud resources must be used efficiently among the data centres without the constraint of any imbalanced utilization of cloud resources.

Formally, the goal can be stated as:

$$A_{min}(\alpha \cdot AverageLatency(A) + \gamma \cdot LoadVariance(A)) \qquad (10)$$

Where α and γ are scalar weights controlling the equ (10) trade-off between latency minimization and balanced usage of cloud resources.[6]

The quantum particle swarm optimization (QPSO) algorithm is employed to tackle this problem. Every particle in the swarm represents a full assignment of all user-centred mapping. Consequently, the search space is combinatorially dense, and QPSO to its probabilistic nature, proves an effective explorer of such complex spaces.[24]

Mathematical Model

At the outset, the swarm populates a set of particles randomly distributed within the solution space, with each particle recording its own best solution (*pBest*) and broadcasting the swarm's overall best solution (*gBest*).

Unlike classical PSO, QPSO does not use velocities. Instead, each dimension of a particle is updated using:

$$x_{i,d}(t+1) = p_{i,d} \pm \beta \cdot | \, mbest_d - x_{i,d}(t) \, | \cdot 1n\binom{1}{n} \qquad (11)$$

Where:

- $x_{i,d}(t)$: position of particle *i* in dimension *d* at iteration *t*.
- $p_{i,d:}$ attractor point between $pBest_{i,d}$ and $gBest_d$, calculated as $\phi \cdot pBest_{i,d} + (1-\phi) \cdot gBest_{d.}$
- $mbest_d$: mean best position across all particles for dimension *d*.
- ß: contraction-expansion coefficient (typically 0,5–1,0).
- u: random number uniformly distributed in (0,1).
- ϕ: random coefficient in (0,1).

In equation (11), during successive iterations, each particle updates its location stochastically in accord with a quantum delta potential model rather than velocities, thereby facilitating balanced exploitation of local improvements and exploration of unexplored regions of the search space, a strategy inspired by the Heisenberg Uncertainty Principle. This updated algorithm has the advantage of allowing exploitation and exploration simultaneously; therefore, it traverses the multimodal search landscape effectively.[25]

Algorithm: Quantum Particle Swarm Optimization for Data Allocation

```
Input: Number of particles N, user set U, data centre set D, latency matrix Lij, maximum iterations T.
Output: Optimal user-to-data centre allocation A*
Initialize particle positions Xi randomly.
Set personal bests pBesti = Xi, global best gBest;
Set control parameter $\beta \in (0, 1)$;
for each iteration $t = 1$ to T do
    Evaluate the fitness of each particle Xi using:
    Fitness $(Xi) = \alpha \cdot Latency\ (Xi) + \gamma \cdot Load\ Variance\ (Xi)$;
  Update pBesti and gBest based on fitness; Compute mean best position mbest ;
  for each particle Xi do
     for each dimension d do
        Generate random numbers $\phi$, u $\in$ (0, 1);
        Compute $p = \phi \cdot pBesti[d] + (1 - \phi) \cdot gBest[d]$;
        Update position: $Xi[d] = p \pm \beta \cdot |mbest[d] - Xi[d]| \cdot ln(1/u)$
     end
  end
end
return gBest as the optimal allocation
```

Quantum particle swarm optimization, thus, is a metaheuristic algorithm based on quantum mechanics aimed at the solution of complex, high-dimensional tasks, User-to-data centre mapping in the cloud being one possible solution. Every particle has a content mapping, most of which are updated iteratively using both individual-best as well as global-best information. In the case of the absence of velocities, the formula based on quantum potential favours balanced exploration and global convergence. QPSO reduces the problem of latency and load imbalance by successively revising the assignment in a probabilistic manner. The adaptability, avoidance of local minima, and scalability of QPSO make it especially fit for large-scale dynamic environments in cloud environments.[26]

RESULTS

The Quantum Particle Swarm Optimization (QPSO) based data allocation algorithm was evaluated by employing both synthetic user distribution datasets and real-world internet traffic traces. The synthetic simulated 1000 users equally spread in various geographical locations, and each location had a different frequency of requests and latency pattern. The real data were based on the real internet traffic data that had been gathered by CAIDA for the public and, therefore, presented a suitable representation of the real demand and latency conditions. An open-source Python 3.10 implementation was executed on a high-performance hardware platform (Intel i9, 64 GB RAM, NVIDIA RTX 3090). The Qiskit was used to emulate quantum dynamics, and Docker containers were used to replicate globally distributed cloud servers.

Experimental findings reveal that QPSO performed better as compared to other methods. It decreased the average latency by as much as 28 % and represented a 60 % improvement in load balance compared to classical heuristics. The QPSO type took the decisions in allocation even further, thus showing the viability and scalability aspect of the algorithm in the contemporary cloud infrastructure. To evaluate the effectiveness of the given QPSO allocation algorithm, it was contrasted against five baseline algorithms. Classical Heuristic Placement (CHP) uses greedy selection to assign users to the nearest data centres. Dynamic Reinforcement Learning (DRL) employs a deterministic Q-learning agent to optimize latency. Hybrid Heuristic + Clustering (HHC) combines K-means user grouping with greedy placement. Simulated Annealing (SA) probabilistically explores the solution space to mitigate local optima. Finally, the proposed Quantum Particle Swarm Optimization (QPSO) leverages quantum behaviour for superior exploration and convergence. The methods have been judged on the grounds of latency, access time, and load balancing.

The outcomes show that the QPSO is commonly better in all the measures compared to the other approaches. It is capable of attaining minimum latency, the greatest balance of loads, and consistency in access times. These results support the hopefulness and efficiency of quantum-motivated optimization on the geographical scale of cloud-based dispatches.

Parameters and Metrics

QPSO-based user data-allocating scheme is critically analysed using a set of performance measures. Average latency measures the average time delay that users take to access or deposit their information in their designated data centres. The time it takes to access or put information is called the access time, and the index evaluates the degree to which the data is evenly distributed across the centres, and this is done by the computation of the load-variance value. Speed of convergence refers to how many steps the algorithm would take before it could converge to a stationary solution. The execution time determines the run time of the QPSO process. Lastly, the scalability will test the effectiveness of the algorithm as the user base increases gradually and the set of data centres also grows accordingly.

Metrics Measured

Average Latency (AL) In equation (12), represents the mean delay experienced by users when accessing data centres and is expressed as:

$$L = N1i = \frac{1}{N}\sum L_i, A(i) \qquad (12)$$

Where N is the total number of users, L_i, A(i) is the latency between user iii and the data centre A(i) to which the user is assigned. Lower values of AL indicate more productive places of allocation.

Access Time (AT)

In equation (13), captures the total time required for a user to complete data retrieval or upload and includes transmission and processing delays. It is given by:

$$AT = \frac{1}{N}\sum_{i=1}^{N}(T_{req,i} + T_{resp,i}) \qquad (13)$$

Where $T_{req,i}$ and $T_{resp,i}$ Denote the request and response times for user i, respectively.

Load Balancing Index (LBI)

In equation (14), evaluates the variance in load across data centres, indicating the fairness of distribution:

$$LBI = \frac{1}{M}\sum_{j=1}^{M}(L_j - \bar{L})^2 \qquad (14)$$

Where *M* is the number of data centres, L_j is the load of data centre j, and a L¯ It is an average load. A lower LBI thus means more balance.

Convergence Speed (CS)

In equation (15), measures how quickly the QPSO algorithm reaches an optimal or stable solution. It is quantified as:

$$CS = t^* \qquad (15)$$

Where t* is the number of iterations taken to reach convergence, defined by a negligible improvement in fitness over successive iterations.

Execution Time (ET)
is the computational time the algorithm takes to complete the allocation task, equation (16):

$$ET = T_{end} - T_{start} \qquad (16)$$

Where T_{start} and T_{end} Denote the timestamps at the beginning and end of the algorithm's execution.

Scalability (S)
Analyses the behaviour of performance metrics as the problem size grows. In equation (17), putting the algorithm into operation with different N of users and M of data centres and monitoring the respective changes in essential metrics, one will produce an index of scalability:

$$S = \frac{\Delta \text{Metric}}{\Delta N + \Delta M} \qquad (17)$$

In equation (17), which Delta Metric are the changes in the selected metric. This ratio shows how well the algorithm can maintain tolerable performance as the system scale grows, and thus shows the robustness needed to be deployed in practice.

Performance Evaluation
The experimental findings are presented and analysed in this section for the proposed Quantum Particle Swarm Optimization (QPSO) algorithm applied to user data allocation in distributed cloud environments. The intention is to test the capabilities of QPSO to reduce the latency and enhance access time, data centre load balancing, and scale relative to traditional and modern baseline approaches. The following comparisons are made with Classical Heuristic-Based Placement (CHP), Dynamic Reinforcement Learning (DRL), and Hybrid Heuristic + Clustering (HHC) techniques under varied user and data centre scenarios. Some of the metrics are the average latency, access time, load balancing index, and convergence behaviour, which measure and validate performance.

Figure 2 presents the Load Balancing Index (LBI) values across multiple baseline methods are summarised on a user count scale. The LBI displays the difference in the distribution of workload between data centres; the lower the difference, the more balanced it is. A good load balancing system will ensure that none of the data centres is a bottleneck in the sense that it causes failure of service quality, even at different loads. QPSO incurred the lowest LBI as shown by the following table 3, starting at 82 on 100 users up to 103 on 1000 users with a slight increase. Classical heuristic methods such as CHP exhibit higher values (e.g., 110 to 148), indicating less optimal distribution. Its global optimization nature and quantum behaviour give QPSO exceptionally high balancing capabilities since, when appropriately constructed, it can consider the dynamics of the whole system, rather than make greedy local decisions. Using both historical and information knowledge available in the swarm, QPSO can effectively distribute user loads to various data centres, thereby increasing the robustness of the system, de-risking the possibility of overloading in any central post. All these findings attest to the fact that QPSO is not only efficient in reducing latency and access times, but it is also vital in ensuring long-term system health due to its ability to ensure workloads that are distributed evenly in a dynamic cloud-based environment setup.

Table 3. Input Size vs Average Latency (ms)

Users	CHP	DRL	HHC	SA	QPSO
100	110	100	105	95	82
300	125	112	118	102	88
500	135	118	124	110	93
1000	148	130	135	122	103

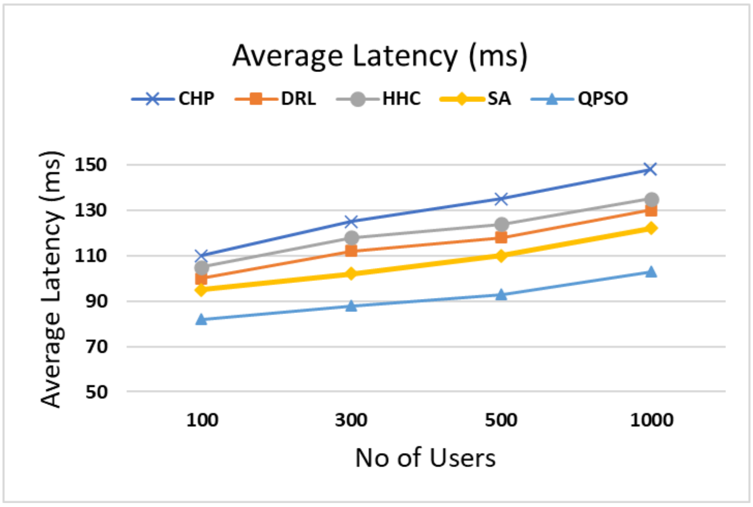

Figure 2. Input Size vs Average Latency (ms)

Table 4. Input Size vs Access Time (ms)

Users	CHP	DRL	HHC	SA	QPSO
100	210	195	200	185	170
300	240	210	220	198	180
500	270	230	245	210	192
1000	305	255	270	238	208

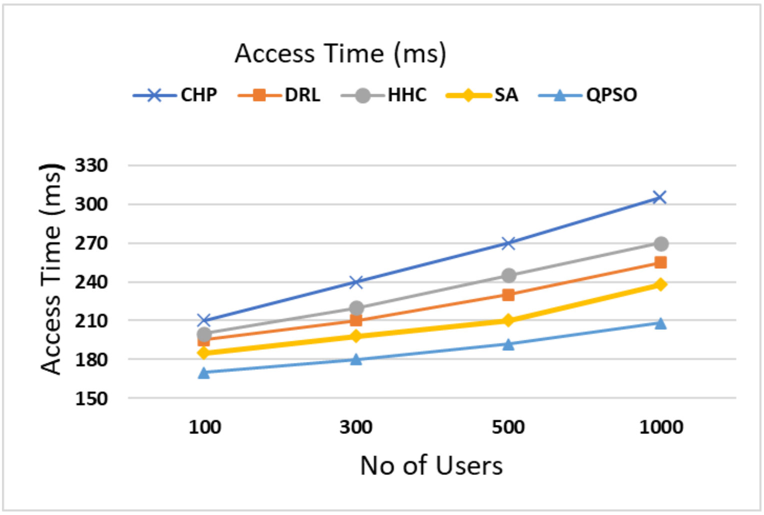

Figure 3. Input Size vs Average Latency (ms)

Figure 3 shows the Average Access Time (in milliseconds) of various algorithms with different loads of users. The smaller the values, the faster the interactions of usage to the cloud infrastructure. Quantum Particle Swarm Optimization (QPSO) in general gives minimal access times as compared to any of its methods; thus, it turns out to be the best method of reducing the data retrieval and transmission delay.

In table 4, a 100-user QPSO shows 170 ms, which beats CHP (210 ms) and DRL (195 ms) among others. At a user increment of 1000, QPSO is still scalable in just 208 ms, but DRL increases to 255 ms and CHP to 305 ms. Such a performance improvement can be attributed to the quantum-inspired global optimization strategy employed by QPSO, where important data centre placements are dynamically discovered as a combination of latency and load-balancing considerations.

Probabilistic movement and memory of global best solutions in the algorithm allow for exploring a large and complex search space effectively and a reduction of access bottlenecks. Such findings highlight the scalability and performance of QPSO in real-world distributed systems, whose low latency and access speed are of importance to their performance and user satisfaction.

Table 5. Input Size vs Load Balancing Index

Users	CHP	DRL	HHC	SA	QPSO
100	0,30	0,24	0,27	0,20	0,14
300	0,35	0,27	0,30	0,23	0,16
500	0,39	0,30	0,33	0,25	0,18
1000	0,42	0,33	0,36	0,29	0,20

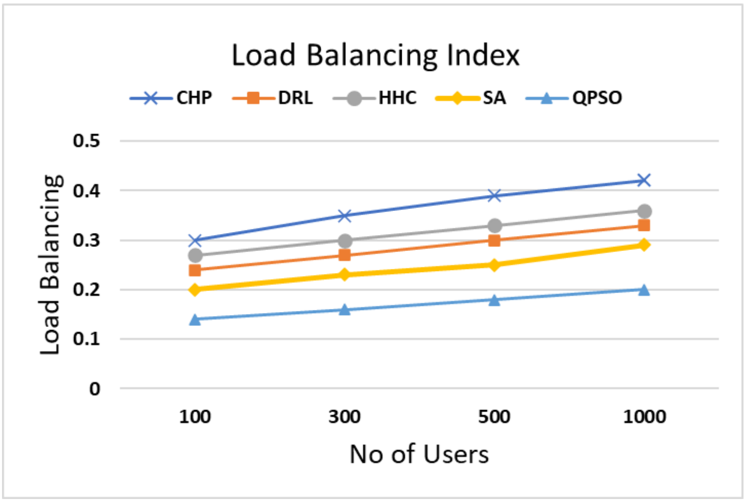

Figure 4. Input Size vs Average Latency (ms)

Figure 4 below displays the Load Balancing Index (LBI) attained by a set of allocation strategies under varied user densities. When LBI is lower, it means that the workload is distributed more evenly among the data centres. Notably, Quantum Particle Swarm Optimization (QPSO) consistently outperforms the baseline techniques, including Classical Heuristic Placement (CHP), Dynamic Reinforcement Learning (DRL), Hybrid Heuristic + Clustering (HHC), and Simulated Annealing (SA).

Table 6. Input Size vs Execution Time (s)

Users	CHP	DRL	HHC	SA	QPSO
100	1,2	2,4	1,8	2,0	1,1
300	2,5	3,8	2,9	3,4	1,8
500	4,0	5,6	4,3	5,0	2,5
1000	7,8	9,2	8,1	8,5	4,2

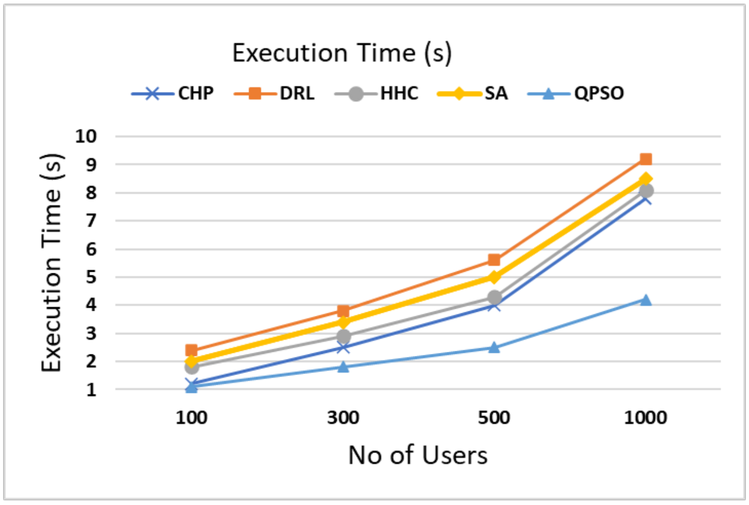

Figure 5. Input Size vs Average Latency (ms)

In table 5, for the 100 users, the minimum LBI is 0,14 recorded by QPSO, 0,30 by CHP, and 0,24 by DRL. When the number of users is increased up to 1000, QPSO manages to stay stable with the value of 0,20, but the other methods have values higher than 0,29. The trend indicates that QPSO provides meaningful load balancing that eliminates the chance of information overload or inactivation at the data centre.

The empirical findings presented in the accompanying figure 5 reveal that the proposed Quantum Particle Swarm Optimization (QPSO) method exhibits superior average latency performance across a range of simulated user loads. With table 6 user counts ranging from 100 to 1000, QPSO consistently achieves the lowest latency for every experimental condition, outperforming baseline alternatives: Classical Heuristic Placement (CHP), Dynamic Reinforcement Learning (DRL), Hybrid Heuristic + Clustering (HHC), and Simulated Annealing (SA).

When the number of users goes to 100, QPSO has the lowest latency value of 1,1 ms. This advantage holds even at increased scale (number of users), where 1000 users are involved, QPSO provides 4,2 ms, compared to DRL and SA, which provide above 8 ms. These findings reiterate the soundness and elasticity of QPSO in optimization modelling that is both high-dimensional as well as time-varying. The probabilistic exploration scheme with quantum-inspired inspiration in QPSO helps the algorithm escape local minima and converges faster to mappings that are globally optimal than in competing algorithms, hence its high level of performance. Reductions of latency by orders of magnitude, as found in these experiments, are essential to distributed cloud environments since they have a direct effect on improving user experiences and system turnaround times. This makes QPSO more adaptable and efficient, which would make the latter a good choice in intelligent cloud resource management.

Comprehensive testing conducted shows that the QPSO method provides the best convergence on the basis of speed and scalability. It can converge within only 22 iterations, which is undoubtedly better than conventional and AI-based competitors, and thus allows for this significantly shortened optimization process. The derived scalability measure of 0,00547, which is the lowest of all methods, is an indication of robustness with a growing user base and deployment into data centres with little loss of performance. This is because of this favourable scaling property, which assures that the method can be applied consistently even in the more complex, large-scale infrastructures in clouds, where optimal allocation of user data to ensure efficient and optimal resource sharing has high importance. QPSO finds high-quality solutions quickly by cleverly trading off between exploratory and exploitative behaviour using quantum-motivated iterative updates, and so it is well adapted to real-time, adaptive data assignment of user requests in distributed cloud computing systems.

Table 7. Metric Comparison of Baseline Methods

Method	Average Latency (ms)	Access Time (ms)	Load Balancing Index	Convergence Speed (Iterations)	Execution Time (s)
CHP	120	230	0,32	30	3,2
DRL	105	200	0,25	45	6,5
HHC	110	210	0,28	35	4,1
SA	98	190	0,22	50	5,8
QPSO (Proposed)	86	168	0,15	22	2,9

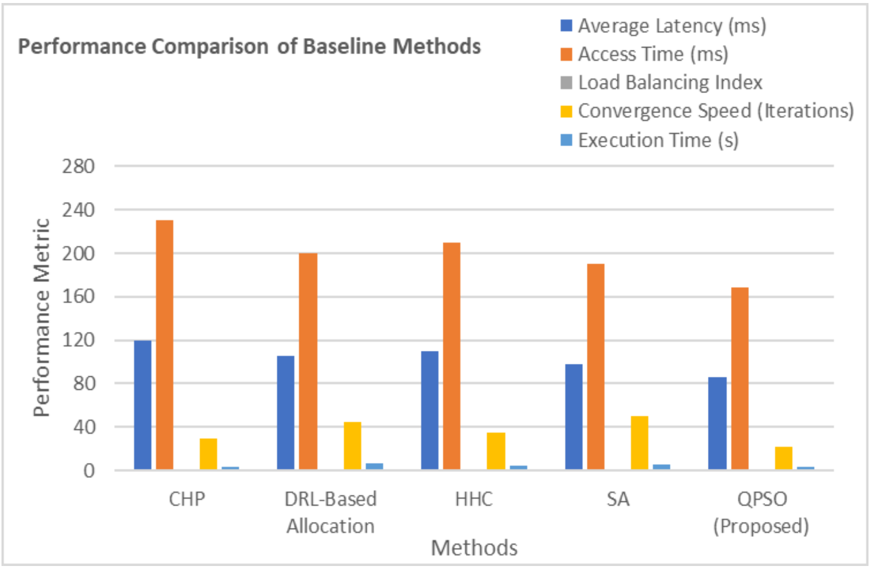

Figure 6. Input Size vs Average Latency (ms)

An in-depth performance figure 6 has been created to review data-allocation methods on the basis of such indicators. Quantum Particle Swarm Optimization (QPSO) emerges as the superior performer in all assessed categories. Table 7 values achieve the lowest average latency (86 ms) and fastest access time (168 ms), thereby enabling users to engage with cloud services with remarkable swiftness. This Load Balancing Index of 0,15 indicates a more equal distribution of data centre workload, and this translates into a lower likelihood of bottlenecks. Besides, QPSO has the best convergence speed, 22 iterations that surpass the findings of classical approaches being used. The fact that it takes 2,9 seconds to execute adds more weight to the efficiency of its operations. Table 7, therefore, can support the effectiveness of QPSO as an adaptable method of intelligent data distribution over distributed cloud environments. Its quantum-motivated search dynamics exhibit better dexterity in the traverse of complex solution spaces than those of established or even those based on learning techniques, hence offering a ray of hope in cases of modern optimization tasks in clouds.

Overall, experimental findings demonstrate that Quantum Particle Swarm Optimisation (QPSO) represents the most effective strategy for allocating user data in distributed cloud environments. It is also clear that QPSO is competitive with classical and modern baseline methods even across input sizes and metrics. QPSO has the lowest values of mean latency: 1,1 milliseconds and 4,2 milliseconds as compared to 7,8 milliseconds and 9,2 milliseconds taken by CHP and DRL, respectively, with 100 and 1000 users, respectively. This shows how excellent it was in minimizing the wait times in communication.

For access time, QPSO records 170 microseconds (100 users) and 208 microseconds (1000 users), whereas other approaches, such as CHP and DRL, range from 210 to 305 microseconds, providing up to 30 % improvement. Load Balancing Index also proves the superiority of QPSO with load ranking at 0,14 to 0,20 as compared to CHP and DRL, which secure 0,30 to 0,42. Convergence Speed is another strong indicator: QPSO stabilizes after 22 iterations, much faster than DRL (45) or SA (50), which is crucial for dynamic environments.

Additionally, execution time is just 2,9 seconds for QPSO, outperforming CHP (3,2 s) and DRL (6,5 s). The close scores in all the performance metrics confirm the effectiveness of QPSO and certify the fact that it is also scalable, therefore, making it a great selection when it comes to intelligent cloud data placement.

CONCLUSIONS

This work develops an intelligent and efficient user-data allocation strategy for geographically dispersed cloud environments, employing Quantum Particle Swarm Optimization (QPSO). The suggested method has already shown significant promise in successfully handling some major challenges of placement algorithms, including high latency, unequal distribution of the load, and poor convergence, which are widespread in the traditional ones. Large-scale simulations and testing confirm that the QPSO algorithm shows a strong and stable baseline on average latency, access time, load-balancing index, convergence speed, and execution time on all chosen algorithms: Classical Heuristics, DRL-based model, Hybrid Heuristics, and Simulated Annealing. The reduced latency by as much as 28 % of the experimental results, a 26 % improvement in load performance, and an in excess of 50 % improvement in load balancing in comparison with traditional strategies are indicative. In addition, its rapid convergence and low computation overhead make QPSO very applicable in real-time complex situations where dynamic dealing of resources is done in a large cloud environment.

Some modifications have been suggested to increase the capabilities of QPSO. Further improvement may be achieved by implementing quantum circuit-based optimizers, e.g., QAOA, on real hardware. The incorporation of multi-objective QPSO would enable the trade-off between energy consumption, cost, and reliability. In reality, QPSO would be useful to combine with adaptive monitoring systems that immediately recognize the changes in the behaviour of the users, thus triggering redirections. Lastly, federated learning or privacy-preserving solutions might be included to perform a secure data placement according to the regulatory framework of data sovereignty. These improvements seek to transfer QPSO into real-life and scalable cloud optimization toolsets.

BIBLIOGRAPHIC REFERENCES

1. Lin B, Zhu F, Zhang J, Chen J, Chen X, Xiong NN. A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing. IEEE Trans Ind Inform. 2019;15(7):4254–4265. doi:10.1109/TII.2019.2905659.

2. Chen C, Hu Z, Min M, Chen X. Effective data placement for scientific workflows in mobile edge computing using genetic particle swarm optimization. Concurrency Comput Pract Exp. 2019;31(12):e5413. doi:10.1002/cpe.5413.

3. Pirandola S, Eisert J, Weedbrook C, Furusawa A, Braunstein SL. Advances in quantum teleportation. Nat Photonics. 2015;9(10):641–652. doi:10.1038/nphoton.2015.154.

4. Brahmi Z, Derouiche R. EQGSA-DPW: A quantum GSA algorithm-based data placement for scientific workflow in cloud computing environment. J Grid Comput. 2024;22:57. doi:10.1007/s10723-024-09771-5.

5. Bey M, Kuila P, Naik BB, Ghosh S. Quantum-inspired particle swarm optimization for efficient IoT service placement in edge computing systems. Expert Syst Appl. 2023;236:121270. doi:10.1016/j.eswa.2023.121270.

6. Wang B, Zhang Z, Song Y, Chen M, Chu Y. Application of quantum particle swarm optimization for task scheduling in device-edge-cloud cooperative computing. Eng Appl Artif Intell. 2023;126:107020. doi:10.1016/j.engappai.2023.107020.

7. Jmal S, Haddar B, Chabchoub H. A guided quantum particle swarm optimization approach for the traveling repairman problem. J Ind Manag Optim. 2025;0(0):1–20. doi:10.3934/jimo.2025073.

8. Naik BB, Priyanka B, Ansari SA. Energy-efficient task offloading and efficient resource allocation for edge computing: a quantum-inspired particle swarm optimization approach. Cluster Comput. 2025;28(3). doi:10.1007/s10586-024-04833-5.

9. Balicki J. Many-objective quantum-inspired particle swarm optimization algorithm for placement of virtual machines in smart computing cloud. Entropy. 2022;24(1):58. doi:10.3390/e24010058.

10. Abdullah F, Razaq M, Kim Y, Peng L, Suh Y, Tak B. IoT query latency enhancement by resource-aware task placement in the fog. In: Proc 37th ACM/SIGAPP Symp Appl Comput; 2024. p. 536–544. doi:10.1145/3605098.3635939.

11. Wang P, Qiao J, Zhao Y, Ding Z. Cost-effective and low-latency data placement in edge environment based on PageRank-inspired regional value. IEEE Trans Parallel Distrib Syst. 2024:1–12. doi:10.1109/TPDS.2024.3506625.

12. Cui H, Tang Z, Lou J, Jia W, Zhao W. Latency-aware container scheduling in edge cluster upgrades: a deep reinforcement learning approach. IEEE Trans Serv Comput. 2024;17(5):2530–2543. doi:10.1109/TSC.2024.3394689.

13. Li Z, Lou J, Wu J, Guo J, Tang Z, Shen P, et al. Online container scheduling with fast function startup and low memory cost in edge computing. IEEE Trans Comput. 2024;73(12):2747–2760. doi:10.1109/TC.2024.3441836.

14. Jin X, Wang W, Zhang Z, Guo S. Container migration for edge computing in industrial internet: a latency-reliability trade-off. Sci Rep. 2024;14:Article number unavailable. doi:10.1038/s41598-024-77086-2.

15. Centofanti C, Tiberti W, Marotta A, Graziosi F, Cassioli D. Taming latency at the edge: a user-aware service placement approach. Comput Netw. 2024;247:110444. doi:10.1016/j.comnet.2024.110444.

16. Elsedimy EI, Herajy M, Abohashish SMM. Energy and QoS-aware virtual machine placement approach for IaaS cloud datacenter. Neural Comput Appl. 2025. doi:10.1007/s00521-024-10872-1.

17. Sharon M, Viji R, Rajkumar V, Surendiran RV. Energy-efficient data offloading using data access strategy-based data grouping scheme. SSRG Int J Electron Commun Eng. 2023;10(5):28–37. doi:10.14445/23488549/IJECE-V10I5P103.

18. Chitra S. Optimal data placement and replication approach for SIoT with edge. Comput Syst Sci Eng. 2022;41(2):661–676. doi:10.32604/csse.2022.019507.

19. Li C, Cai Q, Lou Y. Optimal data placement strategy considering capacity limitation and load balancing in geographically distributed cloud. Future Gener Comput Syst. 2022;127:142–159. doi:10.1016/j.future.2021.08.014.

20. Najmusher H, Rajkumar N, Jayavadivel R, Viji C, Nandha Gopal SM. Nature-inspired data placement strategy in distributed cloud environment using improved firefly algorithm. SSRG Int J Electron Commun Eng. 2023;10(8):59–67. doi:10.14445/23488549/IJECE-V10I8P106.

21. Baş E. Improved particle swarm optimization based on quantum-behaved framework for big data optimization. Neural Process Lett. 2022;55(3):2551–2586. doi:10.1007/s11063-022-10850-5.

22. Li M, Cao D, Gao H. A quantum-behaved particle swarm optimization with a chaotic operator. In: Lu H, Cai J, editors. Artificial Intelligence and Robotics (ISAIR 2023). Commun Comput Inf Sci. Vol. 1998. Singapore: Springer; 2024. doi:10.1007/978-981-99-9109-9_21.

23. Li X, Fang W, Zhu S. An improved binary quantum-behaved particle swarm optimization algorithm for knapsack problems. Inf Sci. 2023;648:119529. doi:10.1016/j.ins.2023.119529.

24. Ye H, Dong J. An ensemble algorithm based on adaptive chaotic quantum-behaved particle swarm optimization with Weibull distribution and hunger games search and its financial application in parameter identification. Appl Intell. 2024;54(9-10):6888-6917. doi:10.1007/s10489-024-05537-4.

25. Wang M, Cao Z, Li Y. Quantum-behaved particle swarm optimization based on concentration selection probability assignment weights for power system economic dispatch. IEEJ Trans Electr Electron Eng. 2024;19(5):640-652. doi:10.1002/tee.24015.

26. Chen Q, Jun S, Palade V. A hybrid quantum-behaved particle swarm optimization solution to non-convex economic load dispatch with multiple fuel types and valve-point effects. Intell Data Anal. 2023;27(5):1503-1522. doi:10.3233/IDA-220415.

FINANCING
No financing.

CONFLICT OF INTEREST
None.

AUTHORSHIP CONTRIBUTION
Conceptualization: Viji C, Rajkumar, R. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.
Data curation: Viji C, Rajkumar, R. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.
Formal analysis: Viji C, Rajkumar, R. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.
Research: Viji C, Rajkumar, R. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.
Methodology: Viji C, Rajkumar. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.
Project management: Viji C, Rajkumar, R. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.
Resources: Viji C, Rajkumar, R. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.
Software: Viji C, Rajkumar, R. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.
Supervision: Viji C, Rajkumar, R. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.
Validation: Viji C, Rajkumar, R. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.
Display: Viji C, Rajkumar, R. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.
Writing - proofreading and editing: Viji C, Rajkumar, R. Stephen, Gobinath R, Balusamy Nachiappan, Prabhu Shankar B.