Minimizing Qubit Overhead in Error-Aware Distributed Quantum Computing

Ranjani G. Sundaram
Stony Brook University, NY

Himanshu Gupta
Stony Brook University, NY

***Abstract*—Scaling fault-tolerant quantum computing to practical advantage requires distributing quantum circuits across networks of heterogeneous processors. However, managing physical qubit overhead while satisfying strict logical error targets remains a critical bottleneck. We introduce the Error-Aware Distributed Quantum Computing (`EA-DQC`) problem, a unified framework for minimizing the physical footprint of distributed circuits under global error and per-processor capacity constraints. To drive our algorithm design and enable formal performance guarantees, we leverage an *aggregation model* that frames single-qubit allocation as a Generalized Assignment Problem (GAP), allowing us to prove the optimality of our main algorithm (`BPF`) in several hardware-specific cases. We also consider the generalized setting of multi-qubit block-encodings, which introduce *sub-additive costs* and non-monotonic scaling that break standard assignment frameworks, necessitating a density-aware algorithm (`BCF-Set`) to exploit synergistic resource efficiencies. We evaluate our framework on a diverse set of synthetic and benchmark circuits (QFT, QPE, GHZ) across heterogeneous network topologies. Our results show that our combinatorial heuristics achieve up to a 50% reduction in physical-qubit overhead compared to stochastic baselines such as Simulated Annealing. This work provides a rigorous bridge between quantum error correction and classical optimization, offering a scalable path for resource-efficient fault-tolerant distributed quantum architectures.**

I. Introduction

Quantum computing promises to solve specialized computational problems that are fundamentally intractable for classical systems. However, realizing this potential requires executing circuits with millions of high-fidelity logical operations. A significant hurdle remains: current physical error rates are several orders of magnitude too high for reliable computation, necessitating Quantum Error Correction (QEC). While QEC can suppress physical error to a tolerable logical error rate, it imposes a massive resource burden—a single logical qubit may require thousands of physical qubits to achieve the necessary reliability. Distributed Quantum Computing (DQC), which links multiple quantum processors (QPUs) into a unified network, provides a modular and scalable approach to achieving the required number of physical qubits.

The primary bottleneck to achieving practical DQC is the simultaneous management of physical-qubit overhead and the aggregate circuit error rate. These metrics dictate the economic and practical feasibility of any quantum architecture: the physical qubit count drives the massive hardware and cooling costs, while the total circuit error determines whether the output of a computation can be trusted.

Realizing error-aware DQC requires navigating a fundamental resource-error trade-off: higher error suppression demands more physical qubits, while minimizing resource usage risks exceeding the error threshold required for a successful computation. This challenge is compounded in distributed environments, where executing remote gates (operations spanning multiple QPUs) necessitates high-fidelity entanglement pairs (EPRs) generated over noisy, lossy channels.

In this work, we address this trade-off by introducing a framework that treats the target circuit error as a strict constraint rather than a secondary metric. The optimization objective is then to minimize the total physical qubit overhead across the network required to satisfy this error budget. This necessitates a comprehensive model that accounts for all noise sources, including native processor noise, QEC-induced logical error rates, inter-processor channel loss, and the performance of remote-entanglement generation protocols. Crucially, we account for the fact that the time required to generate entanglement directly affects qubit fidelity via decoherence. We present a formal definition of the Error-Aware DQC (`EA-DQC`) problem and develop a suite of algorithms designed to minimize resource overhead in heterogeneous quantum networks while strictly adhering to error-budget constraints.

Prior Work. Existing research in DQC has largely focused on algorithmic efficiency and communication minimization. Previous objectives have centered on reducing the number of required EPRs [1, 6, 9, 20, 21] or minimizing total circuit execution time [5, 19, 22]. While these approaches successfully address communication latency, they often rely on simplified noise models that overlook the resource costs of fault tolerance and the complexities of hardware heterogeneity. Complementary efforts have focused on error-correction methods for DQC, proposing schemes such as hierarchical surface codes [13] or distributed QEC architectures to mitigate noise [3]. In these works, error correction is treated as a fixed requirement rather than an optimization variable, leaving the complex trade-off between physical footprint and error budget largely unexplored.

Our work bridges this gap by introducing the `EA-DQC` problem—the first unified framework to optimize the equilibrium between physical qubit overhead and total circuit error across heterogeneous hardware.

This work was partly supported by the National Science Foundation under Award FET-2106447 and CNS-2504621.

Our Contributions. In this paper, we formally define and address the challenge of error-aware distribution of quantum circuits over heterogeneous networks, focusing on minimizing the total physical qubit footprint while strictly satisfying global error targets. Our specific contributions include:

1) *Formal Problem Characterization:* We present a generalized formulation of the `EA-DQC` problem. Our framework is intentionally agnostic to specific hardware noise models and QEC code families, ensuring its applicability across a wide range of emerging fault-tolerant protocols and hardware modalities.
2) *Aggregation Model as a Design Driver:* We propose a streamlined circuit error estimation model based on the *aggregation* of local qubit budgets. This model serves as the theoretical foundation for our optimization techniques, enabling us to bridge the gap between gate-level physics and combinatorial resource allocation.
3) *Algorithms and Optimality Proofs:* We develop a suite of combinatorial heuristics—`BPF`, `DPF`, and `BCF`—that leverage hardware noise hierarchies to optimize qubit placement. We prove that `BPF` yields an optimal solution for several critical cases: (i) two-processor networks, (ii) settings with flexible capacity margins, and (iii) topologies where processor capacities scale with their intrinsic fidelity.
4) *Multi-Qubit Synergistic Encodings:* We extend our framework to accommodate the complex regime of multi-qubit block-encodings, which introduce sub-additive physical costs and non-monotonic scaling that extend beyond standard linear assignment models, necessitating our density-aware `BCF-Set` algorithm to exploit synergistic efficiencies.
5) *Comprehensive Evaluation:* We evaluate our algorithms across diverse benchmarks (including QFT, QPE, and GHZ states) and heterogeneous network topologies. Our results demonstrate that our combinatorial approach achieves up to a 50% reduction in physical qubit overhead compared to a stochastic baseline, Simulated Annealing.

II. Background

Basics of Quantum Computing. The basic unit of information in quantum computing is called a *qubit*, which, unlike a classical bit, can exist in a superposition of states, $|0\rangle$ and $|1\rangle$. In general, a qubit is represented as a state vector $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ within a two-dimensional complex Hilbert space, $\mathbb{C}^2$, satisfying $|\alpha|^2 + |\beta|^2 = 1$. *Measurement* (in the computational basis) is an operation on a qubit that collapses its state to either $|0\rangle$ or $|1\rangle$ with probabilities $|\alpha|^2$ and $|\beta|^2$, respectively. Similar to Boolean gates in classical computing, quantum *gates* are unitary operations applied to qubits to change their state. In general, a quantum computation can be represented as a sequence of gates and measurements over a set of qubits. When multiple qubits interact, they can become *entangled*, meaning their measurement outcomes exhibit correlations that cannot be explained by treating the qubits independently. In particular, the maximally entangled *EPR* (also known as a *Bell pair*) is represented by $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and is widely used as a resource in quantum communication.

Quantum Network and Communication. We model a quantum network (QN) as a connected graph where nodes represent quantum processors and edges represent communication links. Each processor P_i is characterized by a physical qubit capacity c_i and a set of native physical (gate) error rates (PERs) $\{\zeta_{i,g}\}$. Quantum communication between remote processors P_i and P_j relies on the generation of high-fidelity EPR pairs. An EPR over a pair of "remote" nodes A and B is created by first generating EPRs over network *links* along a path connecting A and B, followed by a sequence of entanglement swapping operations. The stochastic nature of entanglement swapping and the noise inherent in optical links mean that EPR generation incurs latency and results in finite *EPR fidelity*. This fidelity directly impacts the error rate of any remote operation that consumes the EPR.

Quantum Error Correction (QEC) and Resource Overhead. QEC is essential for suppressing physical noise to levels required for fault-tolerant computation. By encoding one *logical qubit* into a *logical block* of *physical qubits* using a QEC code (e.g., Surface or LDPC codes), we can exponentially reduce the logical error rate. The protection level is quantified by the code distance (d); a larger d provides higher resilience but requires significantly more physical qubits ($s_q \propto d^2$ for surface codes). In this work, we assume logical qubits may utilize different protection levels (distances) to balance the trade-off between their physical footprint and their contribution to the total circuit error rate.

Executing Gates over Encoded Qubits. The distribution of a circuit involves mapping logical qubits to processors, encoding them using an appropriate QEC code family and protection level (code distance d), and executing the circuit gates fault-tolerantly. The specific method of gate execution over these encoded qubits determines the resulting error contribution:

- *Unary Gates:* Often implemented transversally (physically applied to each qubit in a block), these are generally the lowest-error operations. The logical error rate is primarily determined by the code distance and the processor's native physical (gate) error rate.
- *Local Binary Gates:* These gates are between two qubits on the same processor. Whether implemented transversally or via gate teleportation/lattice surgery, the error rate depends on the physical gate fidelities of the host processor and the code distances of both operand qubits.
- *Remote Binary Gates:* Executed between logical qubits on different processors, these operations require consuming a distributed EPR [7, 10, 11] via gate teleportation. The net error of a remote gate is significantly higher than a local one, as it aggregates errors from: (i) entanglement generation and link fidelity, (ii) local operations at both nodes, and (iii) classical communication (feedforward) latency.

Defining the Net Circuit Error Rate. The cumulative error rate ϵ of a distributed quantum circuit is an aggregate of the

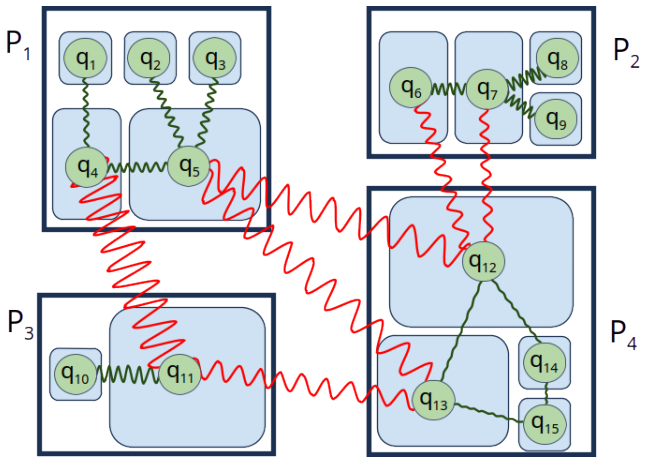

Fig. 1: An illustration of the EA−DQC problem as a qubit allocation task with heterogeneous physical costs and global error constraints. Logical qubits (green circles) are mapped to processors (black rectangles). The physical footprint (blue blocks) of each qubit represents the resource overhead (code distance) required to protect it against noise from local gates (green lines) and higher-error remote gates (red lines); larger wavy-line amplitudes denote increased physical error rates. The network-wide protection levels must "orchestrate" to keep the *aggregate* circuit error within the given budget while minimizing the total physical footprint.

logical errors of all constituent operations. Crucially, this error is a function of: (i) the *qubit-to-processor mapping*, which determines which binary gates become high-error remote operations; (ii) the *processor characteristics*, which define local gate noise; and (iii) the *chosen code distances*, which determine the degree of error suppression for each qubit at the cost of physical usage. The EA−DQC problem, addressed in this paper, seeks to optimize these parameters to meet a total error budget ϵ with minimal total physical qubits $\sum s_q$.

III. Problem Formulation and Related Work

In this section, we establish the necessary terminology and mathematical framework before formally defining the Error-Aware Distributed Quantum Computing (EA-DQC) problem.

Quantum Circuits and Qubit-Allocation Function. We model an abstract quantum circuit $\mathcal{C}$ defined over a set of qubits $\mathcal{Q} = \{q_1, q_2, \dots\}$ as a sequence of gates $\langle g_1, g_2, \dots \rangle$, where each gate g_t is either unary or binary. Executing $\mathcal{C}$ across a quantum network requires two key determinations: (i) a qubit allocation function $\eta : \mathcal{Q} \to P_i$ that maps each logical qubit to a specific processor, and (ii) a choice of encoding for each logical qubit that defines its physical logical block. Both the allocation function and the selected encodings must satisfy the constraint that the total physical qubit count at any individual processor does not exceed its physical capacity.

Error-Aware DQC (EA−DQC) Problem Formulation. Given a quantum circuit $\mathcal{C}$, a quantum network $\mathcal{N}$, and an error budget ϵ, the goal of the EA−DQC problem is to:

1) distribute the qubits of $\mathcal{C}$ across $\mathcal{N}$ by determining the qubit allocation function, and
2) determine the encoding (QEC code family and protection level) for each circuit qubit.

The objective is to minimize the total number of physical qubits across the network while ensuring that: (i) the cumulative circuit error rate does not exceed ϵ, and (ii) the physical qubit count on each processor remains within its specific capacity. See Fig. 1

More formally, let $\mu : \mathcal{Q} \to \{P_1, \dots, P_k\}$ represent the assignment of logical qubits to processors, and let $s_q(\mu)$ denote the number of physical qubits—including peak ancilla and buffer requirements for gate operations—required to encode qubit q on processor $\mu(q)$. The EA−DQC problem seeks to determine the assignment μ and the corresponding sizes $s_q(\mu)$ such that the expected error rate of the distributed circuit remains within the threshold ϵ, and the following capacity constraint is satisfied for all $P_i \in \mathcal{N}$:

$$\sum_{q:\mu(q)=P_i} s_q(\mu) \leq c_i. \tag{1}$$

This formulation is intentionally general, remaining agnostic to specific error models and QEC implementations.

Performance Metrics and Optimization Challenges. The primary metrics for distributed quantum circuits are execution time, physical qubit overhead, and circuit error. In this work, we assume a network of processors with comparable speeds (e.g., a single-species network), which renders the total execution time relatively insensitive to qubit placement.Consequently, our optimization objective is to minimize the total physical footprint while strictly adhering to a logical-error target and per-processor capacity limits. This focus is motivated by the fact that minimizing circuit error alone can lead to extreme resource inefficiency for marginal gains in accuracy. Notably, minimizing the physical footprint indirectly reduces the number of remote gates, as these gates typically require higher protection (and thus more physical qubits) to mitigate their higher error rates.

The EA−DQC problem is NP-hard, as it can be viewed as a significant generalization of the *Generalized Assignment Problem* (GAP). In a standard GAP, fixed assignment costs are given; however, in EA−DQC, the physical costs (assignment costs in GAP) are not natively well-defined for individual qubits. Instead, they are highly interdependent: the error budget and physical footprint of a single qubit depend on the global distribution and the requirements of remote gates. Consequently, the circuit error rate and total footprint are fully defined only when a complete set of assignments is specified. Our overall strategy is to leverage a simplified error model to drive algorithm design and performance guarantees, and subsequently extend these algorithms to arbitrary noise profiles and the synergistic complexities of multi-qubit block-encodings that further reduce physical overhead.

A complete specification of the EA−DQC problem thus necessitates a well-defined model for estimating circuit error as a function of the distribution and encoding parameters, which we discuss in the following section.

Example. Fig. 2 shows an instance and solution of the EA−DQC problem. We are given a quantum circuit with an error budget of 0.5 and a quantum network comprising two

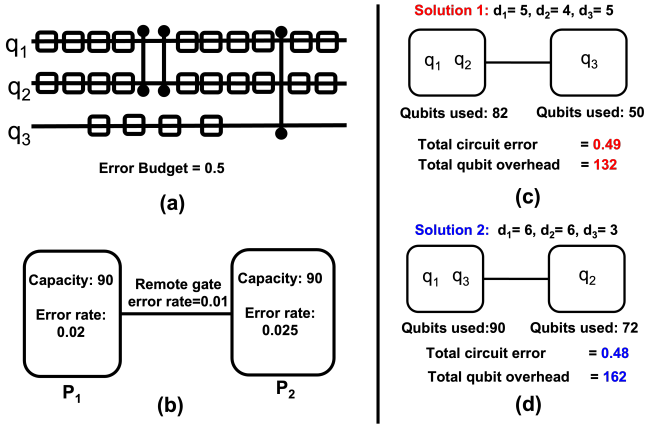

Fig. 2: EA−DQC Problem Example 1. **(a)** A quantum circuit with error budget $\epsilon = 0.5$, (b) A quantum network with two connected processors, (c) and (d) present two solutions to the problem instance.

processors, P_1 and P_2, each with a capacity of 90. Additionally, local gates in P_1 and P_2 incur error rates of 0.02 and 0.025, respectively, and remote gates executed across the processors incur an error rate of 0.01. Fig. 2(c) and (d) present two solutions. The first solution q_1 and q_2 to P_1, and q_3 to P_2, with code distances (d_i values) of 5, 4, and 5 to q_1, q_2, q_3 respectively. The second solution instead allocates q_1 and q_3 to P_1, and q_2 to P_2, and assign code distances of 6, 6, and 3. We observe that although the second allocation yields lower total error, it also incurs a significantly higher physical cost (i.e., the number of physical qubits used). (We approximate the number of physical qubits corresponding to code distance of d as $2d^2$, consistent with surface code scaling for d.)

A. Related Work

Distribution of Quantum Circuits (DQC). The DQC problem involves mapping a quantum circuit onto a distributed network to optimize a specific performance objective. Existing literature broadly falls into two categories based on the optimization target. The first category [1, 6, 9, 14, 15, 20, 21] focuses on minimizing communication overhead, typically by reducing the weighted number of EPRs required for remote gate execution. The second category [5, 8, 19, 22] prioritizes minimizing the total circuit execution time (makespan) by accounting for entanglement generation latency and operation scheduling. While these works provide essential insights into communication and timing, they largely rely on simplified noise models that assume perfect or fixed-fidelity qubits and operations, ignoring the net circuit error and massive physical resource costs associated with fault-tolerant encoding.

Error Correction in Distributed Architectures. Achieving fault tolerance in a network requires hardware-aware error correction strategies. Recent research has introduced specialized protocols for this environment; for instance, recent work on distributed QEC [3] investigates the use of physical qubits from multiple QPUs to form a single logical qubit, enabling processors with limited individual capacity to participate in error correction. Other architectural approaches, such as hierarchical surface codes [13], propose connecting small, modular quantum processing units (QPUs) into a larger unified system to achieve scalability. These works do not consider the distribution of quantum circuits.

Quantum Resource Allocation and Assignment Problems. From a combinatorial perspective, the mapping of logical qubits to a heterogeneous network of capacity-constrained processors shares similarities with the *Generalized Assignment Problem* (GAP) [17, 24]. As discussed before, while traditional GAP research focuses on static costs and linear constraints, the EA−DQC domain introduces unique challenges: assignment costs are derived from gate-level noise models, global error targets must be satisfied through local budgeting, and the multi-qubit setting moves the optimization beyond standard linear or quadratic assignment models into the realm of sub-additive, density-driven clustering. Prior work in quantum *routing and mapping* [4, 18] has primarily focused on minimizing SWAP gate overhead in monolithic chips; our work shifts this focus to the physical resource overhead of error-corrected computation and communication in distributed topologies.

IV. Modeling Circuit Error and Physical Resource Usage

The EA−DQC problem formulation assumes the existence of an error-estimation function defined as follows:

- Given a qubit allocation function η and the specific encoding of each logical qubit, the function $E(\eta, \text{enc})$ returns the estimated *circuit error rate* incurred during execution.

This function can also be utilized in reverse to determine the specific qubit encodings required to satisfy a target error threshold. In this work, we leverage the simplified model described below to drive algorithm design, then extend these solutions to more general error models; we subsequently discuss the general model.

Aggregation Model for Error and Usage Estimation. The *Aggregation Model* provides a streamlined estimation by approximating the total circuit error rate as the sum of individual logical qubit error rates. We estimate the error rate of a single logical qubit as the sum of the error rates of all gates applied to it, with two key simplifying assumptions: (i) to avoid double-counting, we attribute one-half of a binary gate's error rate to each of its participating qubits; and (ii) we initially treat all binary gates as local gates to eliminate the circular dependency between a qubit's error rate and its network-wide allocation.

Formally, the estimated error rate $E(\mathcal{C})$ of a circuit $\mathcal{C}$ is:

$$E(\mathcal{C}) = \sum_{q \in Q(\mathcal{C})} \left(\sum_{g \in U(q)} E(g) + \frac{1}{2} \sum_{g \in B(q)} E(g) \right) \quad (2)$$

where $Q(\mathcal{C})$ is the set of qubits in $\mathcal{C}$, $U(q)$ is the set of unary gates on qubit q, and $B(q)$ is the set of binary gates involving q. This model relies on the assumption that gate failure probabilities are sufficiently low and independent, such that the probability of simultaneous multi-gate failures is negligible.

Under this model, for a given qubit-processor pair (q, p), we determine the optimal encoding and the resulting physical resource usage through the following process:

1) *Error Budget Distribution:* The total circuit error budget ϵ is partitioned across all logical qubits. This allocation can be non-uniform, allowing more sensitive qubits a larger share of the budget.
2) *Determining Encoding:* For a qubit q, with an assigned budget ϵ_q, being mapped to processor p, we identify the optimal encoding (minimizing physical usage) that ensures q's error rate remains below ϵ_q. This is informed by the specific noise characteristics of processor p, and achieved as discussed in the next paragraph.
3) *Allocation and Refinement:* These parameters inform the global qubit allocation algorithm. Any remaining error budget violations at the circuit or qubit level are resolved via a post-processing step that selectively increases the protection levels for the affected qubits.

Determining Encoding from Qubit Error Budgets. We consider two strategies for mapping a qubit's error budget to a physical encoding:

1) *Holistic Code Selection (No-Split):* For a given qubit q, we identify the QEC code family and protection level (distance d) that satisfy the total qubit error budget ϵ_q with the minimum physical footprint. The qubit's aggregate error rate is estimated by summing the error rates of its constituent gates, which are derived from code-specific analytical models (e.g., see Eqn. 4 for surface codes).
2) *Gate-Level Budget Partitioning (Split):* In this method, the qubit's budget ϵ_q is distributed among its constituent gates. Assuming binary gates are local, the budget is partitioned proportionally based on the native gate error rates of the processor. The required protection level for the qubit is then determined by the gate that requires the greatest distance to remain within its sub-budget.

Split Strategy and Generalized Assignment Problem. The introduction of the per-qubit error budget in the *Split* strategy, in conjunction with the locality of the aggregation model, effectively reduces the `EA-DQC` allocation problem to a *Generalized Assignment Problem* (GAP). By anchoring the circuit's global error target to individual logical qubits, the allocation is transformed into an assignment of tasks with fixed, heterogeneous physical costs (m_{qp} values, defined in Lemma 1) to capacity-constrained processors. While GAP is a classic NP-hard problem, standard solvers are typically "bin-agnostic." Our primary heuristic, `BPF` (§V), distinguishes itself by leveraging domain-specific hardware rankings—prioritizing processors by their intrinsic noise profiles—a strategy that is absent in generic GAP formulations but essential for efficient quantum resource management.

General Model for Circuit Error Estimation. In a general setting, the net circuit error rate ϵ is a complex function of: (i) the qubit mapping $\mu(\cdot)$, which dictates the ratio of local to remote (teleported) gates; (ii) the chosen QEC code and protection-level for each logical qubit; and (iii) the heterogeneous error parameters of the processors and network links. Our high-level approach to solving the `EA-DQC` problem within this general model follows a two-step procedure:

1) First, we solve the problem instance using the *Aggregation Model* to establish a baseline solution.
2) Second, we apply a comprehensive circuit error estimation function to the baseline solution. If the resulting error exceeds the threshold ϵ, we perform a post-processing refinement (see Section V-B) to resolve the violation while minimizing the additional physical-qubit overhead.

V. Best Processor First (BPF) Algorithm

In this section, we describe our main proposed algorithm, referred to as `Best Processor First` (`BPF`), and show its performance guarantees in several special cases.

The `BPF` Algorithm. The fundamental intuition behind the `Best Processor First` (`BPF`) algorithm is that a logical qubit q, given a fixed error budget ϵ_q, will incur the minimum physical qubit overhead when mapped to the processor with the lowest native physical error rate (PER). Building on this, `BPF` adopts a greedy strategy by populating processors in increasing order of their PERs.

To maximize resource efficiency, the algorithm seeks to utilize each processor's physical capacity as fully as possible. Consequently, `BPF` iteratively employs a *Knapsack-style* dynamic programming approach [2] to optimally fill the current processor's capacity c_i before moving to the next. Conceptually, this mirrors the process of allocating high-reliability memory blocks in classical computing before utilizing noisier regions.

Algorithm 1: `BPF` Algorithm

Input: Circuit $(\mathcal{Q}, \mathcal{G})$, Network $\mathcal{N}$, Error Budget ϵ_{total}
Output: Qubit allocation η, Code distances $\{d_q\}$
Sort processors $\mathcal{N} = \{P_1 \dots P_k\}$ in increasing order of their (binary gate)^2 PERs, i.e., $\zeta_1 \leq \zeta_2 \leq \cdots \leq \zeta_k$
Let U be the set of unassigned qubits, initially equal to the set of all circuit qubits.
for $j = 1$ to k **do**
    Let $S \subseteq U$ be a subset of qubits that maximizes the number of physical qubits used in j, while satisfying P_j's capacity constraints.[1]
    For all $q \in S$, set $\eta(q) = P_j$.
    $U \leftarrow U \setminus S$
end for
return Qubit allocation function η, Encodings $\{d_q\}$

A. Provable Optimality in Special Cases

Under the *Aggregation Model*, the physical resource cost of a logical qubit can be evaluated independently of others by distributing binary gate errors equally among operands. In this context, we demonstrate that `BPF` is ***optimal*** (minimizes total physical qubits) in the following scenarios:

- *Two-Processor Networks:* When the network $\mathcal{N}$ consists of exactly two processors.

[1]This can be solved optimally using the pseudo-polynomial dynamic programming algorithm for the well-known Knapsack problem.

- *Proportional Capacity:* When a processor's capacity is directly proportional to its error-suppression efficiency.
- *Minimal Capacity Extension:* When capacities can be minimally extended to accommodate the physical footprint of a single qubit (while also granting the same extensions to the optimal).

To establish these results, we first prove a fundamental lemma regarding the scalability of physical costs across different processors. We assume that the qubit encodings are determined via the *Split* method (see Section IV).

Lemma 1: Let m_{ir} denote the physical cost of mapping qubit q_i (with a given error budget ϵ_i) to processor P_r. For any two circuit qubits q_i, q_j and any two processors P_r, P_s, the ratio of their physical costs is invariant of the qubit identity:

$$\frac{m_{ir}}{m_{is}} = \frac{m_{jr}}{m_{js}}. \tag{3}$$

PROOF: The proof follows from the analytical relationship between logical error rates and physical resources in common QEC codes (e.g., surface codes).

1) Using the *Split* method, the qubit error budget ϵ_q is partitioned into fixed gate-level budgets b (binary) and u (unary), which are independent of the target processor.
2) In most QEC families, the logical gate error scales as
$$E_L \approx C \cdot (\zeta_r/\tau)^{(d+1)/2}, \tag{4}$$
where ζ_r is the native PER of processor P_r, d is the code distance, and τ is an error threshold specific to a code family. Solving for the required distance d_{ir} to satisfy (binary)[2] budget b_i of a qubit q_i, we find $d_{ir} \propto \frac{\log(b_i/C)}{\log(\zeta_r/\tau_r)}$.
3) Since the physical cost m_{ir} is proportional to d_{ir}^2, we can express the cost as $m_{ir} = f(q_i) \cdot g(P_r)$, where f depends only on the qubit's gate-level budget and g depends only on the processor's noise characteristics. The ratio $m_{ir}/m_{is} = g(P_r)/g(P_s)$ is therefore independent of the qubit index i, proving the lemma.

■

Special Case 1: Two-Processor Networks. We demonstrate that the BPF algorithm is optimal for a network consisting of two processors, P_1 and P_2, where P_1 denotes the higher-efficiency (lower-error) processor.

Theorem 1: Given a circuit $(\mathcal{Q}, \mathcal{G})$ and a two-processor network $\mathcal{N} = \{P_1, P_2\}$, the BPF algorithm returns a qubit allocation η that minimizes the total physical qubit count.

PROOF: Let (G_1, G_2) be the sets of qubits assigned to P_1 and P_2 by BPF, and let (O_1, O_2) be the sets assigned in an optimal solution. Without loss of generality, we may assume $G_1 \cap O_1 = \emptyset$.[3] This assumption implies $G_1 \subseteq O_2$.

Because BPF employs a Knapsack-style approach to maximize the utilization of the more efficient processor P_1, the total physical cost of G_1 must exceed that of O_1 when evaluated on P_1. Specifically, $c_1(G_1) > c_1(O_1)$, where $c_r(S)$ denotes the total physical qubits required to encode the set S on P_r.

According to Lemma 1, there exists a constant scaling factor $x \geq 1$ such that for any qubit q, the cost on P_2 is proportional to the cost on P_1, i.e., $m_{q2} = x \cdot m_{q1}$. Since $G_1 \subseteq O_2$, it follows that $c_1(O_2) \geq c_1(G_1)$, or equivalently, $c_2(O_2)/x \geq c_1(G_1)$. Combined with our earlier derivation $c_1(G_1) > c_1(O_1)$, we obtain:

$$\frac{c_2(O_2)}{x} > c_1(O_1). \tag{5}$$

To reach a contradiction, consider a *complementary* allocation $(\tilde{O}_1, \tilde{O}_2)$ where $\tilde{O}_1 = O_2$ and $\tilde{O}_2 = O_1$. We claim the total cost of this allocation is strictly less than the optimal cost $C_{opt} = c_1(O_1) + c_2(O_2)$. The cost of the complementary solution is $C_{comp} = c_1(O_2) + c_2(O_1) = \frac{c_2(O_2)}{x} + x \cdot c_1(O_1)$. The inequality $C_{comp} < C_{opt}$ holds if:

$$\frac{c_2(O_2)}{x} + x \cdot c_1(O_1) < c_1(O_1) + c_2(O_2). \tag{6}$$

Rearranging terms and dividing by $(x-1)$ (assuming $x > 1$; $x = 1$ is a trivial case), we get $c_1(O_1) < c_2(O_2)/x$, which is exactly the condition established in Eqn. 5. Thus, $C_{comp} < C_{opt}$, contradicting the optimality of (O_1, O_2). ■

Special Case 2: Minimal Capacity Overflows. We now consider a scenario where we possess the flexibility to exceed the physical capacity of any processor by a small, fractional amount. In this setting, the Knapsack-based selection (Line 4 of Algorithm 1) is replaced by a simple greedy strategy that fills the current processor P_i arbitrarily until it utilizes at least its given capacity c_i. In practice, the capacity is extended minimally—by at most the cost of a single logical qubit on P_i. Below, we prove that this modified BPF algorithm is optimal, provided the baseline optimal solution is also permitted to utilize the same extended capacities.

Theorem 2: Consider the EA-DQC problem where processor capacities may be extended minimally. Given an input circuit $(\mathcal{Q}, \mathcal{G})$ and a network $\mathcal{N} = \{P_1, \ldots, P_k\}$, the modified BPF algorithm yields a total physical qubit count equal to that of an optimal solution, granted the same capacity extensions.

PROOF: We compare the total cost of the assignment generated by the BPF algorithm against a theoretical optimal assignment. For each qubit q, let m_{qk} represent its physical cost when mapped to the least efficient (highest-error) processor P_k. For every other processor P_i, we define the *deflation rate* x_i as the ratio of physical costs between P_k and P_i:

$$x_i = \frac{m_{qk}}{m_{qi}}. \tag{7}$$

Note that $x_i \geq 1$ for all $i < k$, and by Lemma 1, x_i is invariant of the qubit q.

Let $c'_1, c'_2, \ldots, c'_{k-1}$ be the total physical qubits utilized in the first $k-1$ processors by the BPF algorithm (where $c'_i \geq c_i$). The total physical cost of the BPF solution is the sum of the costs on the efficient processors and the cost of the remaining qubits on P_k:

$$C_{\texttt{BPF}} = \sum_{i=1}^{k-1} c'_i + \left(\sum_{q \in \mathcal{Q}} m_{qk} - \sum_{i=1}^{k-1} x_i c'_i \right). \tag{8}$$

[2] We assume the error budget partitioning for q_i is structured such that the binary gate budget b_i dictates the required code distance and consequently, the physical cost of q_i over any processor; thus, Algorithm 3 uses binary PERs to determine the processor order.

[3] Any qubits common to both G_1 and O_1 contribute identical costs and consume identical capacity in both solutions; thus, we need only consider the residual capacity and the remaining qubits.

Now, consider an arbitrary optimal solution O that utilizes c_i'' capacity on processor P_i, such that $c_i'' \leq c_i'$ for $1 \leq i \leq k-1$. The total cost of O is:

$$C_{opt} = \sum_{i=1}^{k-1} c_i'' + \left(\sum_{q \in \mathcal{Q}} m_{qk} - \sum_{i=1}^{k-1} x_i c_i'' \right)$$
$$= \sum_{q \in \mathcal{Q}} m_{qk} + \sum_{i=1}^{k-1} c_i''(1 - x_i).$$

Since $x_i \geq 1$, the term $(1 - x_i)$ is non-positive. Because the BPF algorithm fills the efficient processors more than the optimal solution ($c_i' \geq c_i''$), it follows that:

$$\sum_{i=1}^{k-1} c_i''(1 - x_i) \geq \sum_{i=1}^{k-1} c_i'(1 - x_i). \tag{9}$$

Therefore, $C_{opt} \geq C_{\texttt{BPF}}$. Since C_{opt} cannot be strictly less than $C_{\texttt{BPF}}$, the BPF algorithm is optimal. ∎

Special Case 3: Proportional Capacities. A particularly illustrative case occurs when processor capacities are scaled relative to their *deflation rates*, representing a balance between a processor's physical size and its error-suppression efficiency.

Theorem 3: If the capacity c_j of each processor P_j is proportional to its deflation rate (i.e., $c_j = \alpha x_j$ for some constant $\alpha > 0$), then the BPF algorithm yields an optimal solution. Here, $x_j = m_{qk}/m_{qj}$ as in Theorem 2's proof.

B. Generalization Beyond the Aggregation Model

The Aggregation Model treats remote binary gates as local, thereby providing a mathematical foundation for optimality. However, in distributed settings, the error contribution of a binary gate is a dynamic property dependent on the final mapping η. We now extend the BPF algorithm to handle these interdependencies.

Iterative Incorporation of Remote Gates. To accurately estimate costs during allocation, BPF utilizes a *pessimistic initialization* strategy. We initially map all unallocated qubits to the least efficient processor P_k (ignoring its capacity). This ensures that all binary gates involving unassigned qubits are initially treated as high-error remote operations, providing a conservative upper bound on the required code distance d_q.

As the algorithm iterates from P_1 to P_{k-1}, qubits are moved from the pessimistic pool in P_k to the current processor P_j. This triggers a re-evaluation: gates between the newly assigned qubit and previously allocated qubits on P_j are downgraded from remote to local. This "budget recovery" allows for a potential reduction in d_q and physical footprint s_q. Consequently, we update the physical costs of all remaining qubits after each processor is filled (Algorithm 2, Line 6). To finally resolve any remaining error budget or capacity violations, we employ a post-processing refinement step described in the next paragraph (also see Algorithm 2).

Arbitrary Error Models. In the most general setting, the net circuit error $\mathcal{E}$ may be a non-linear function of the individual qubit parameters. In such a general setting, to resolve any remaining error-budget or capacity violations, we employ the following generalized post-processing step:

Algorithm 2: Generalized BPF Algorithm

1: **Input:** Circuit $(\mathcal{Q}, \mathcal{G})$, Network $\mathcal{N}$, Error Budget ϵ_{total}
2: **Output:** Qubit allocation η, Code distances $\{d_q\}$
3: Sort $\mathcal{N} = \{P_1 \dots P_k\}$ by weighted error rates $\zeta_1 \leq \zeta_2 \leq \dots \leq \zeta_k$
4: $\eta(q) \leftarrow P_k, \forall q \in \mathcal{Q}$ {Pessimistic Initialization}
5: **for** $j = 1$ to $k-1$ **do**
6: Update m_{qj}, the physical cost of a qubit q on processor p_j, for all $q \in U$ based on current η and remote gate costs
7: $S \leftarrow$ Knapsack$(U, c_j, \{m_{qj}\})$
8: $\forall q \in S : \eta(q) \leftarrow P_j, U \leftarrow U \setminus S$
9: **end for**
10: /* Post Processing Step */
11: **while** $\mathcal{E}(\eta, \{d_q\}) > \epsilon_{total}$ **do**
12: Find $q = \text{argmax}_{q'}(\Delta\mathcal{E}/\Delta s_{q'})$
13: $d_q \leftarrow d_q + 1$
14: **if** $s_q >$ AvailableCapacity$(\eta(q))$ **then**
15: $\eta(q) \leftarrow P_k$ {Migrate to overflow processor}
16: **end if**
17: **end while**
18: **return** $\eta, \{d_q\}$

Post-processing Step. (See Algorithm 2)

- Compute the net circuit error rate using the given circuit error model.
- If the circuit error rate is larger than the circuit error budget, then we calculate the sensitivity $\mathcal{S}_q = \Delta\mathcal{E}/\Delta s_q$ for each qubit, representing the reduction in circuit error per additional physical qubit. We greedily increase the distance of the most sensitive qubit until the budget is met.
- We repeat the above process until the total circuit error is below the circuit error budget, while also moving some qubits to the worst processor as needed.

VI. Best Combination First (BCF) and Densest Processor First (DPF) Algorithms

Here, we present two additional heuristics for EA-DQC.

Best Combination First (BCF) Algorithm. The Best Combination First (BCF) algorithm provides an alternative greedy perspective that prioritizes *mapping affinity* over hardware ordering. While BPF is a processor-centric approach—filling the most reliable hardware first—BCF is a qubit-centric strategy. In each iteration, it evaluates all possible (qubit, processor) pairs (q, P_i) to identify the single assignment (q^*, P_i^*) that yields the minimum marginal increase to the global physical qubit count $\sum s_q$.

This approach may be well-suited to heterogeneous networks and circuits in which logical qubits have significantly different gate compositions. In such cases, specific processors may be better optimized for certain gate types, creating a natural affinity for specific qubits within the circuit. This granular evaluation allows BCF to exploit potential local optima

that a sequential processor-filling strategy might overlook, particularly when remote gate overheads vary significantly across the network topology. We evaluate both algorithms to characterize the trade-off between the provable optimality of BPF in restricted settings and the heuristic flexibility of BCF in heterogeneous scenarios.

Algorithm 3: Best Combination First (BCF)

Input: Circuit $(\mathcal{Q}, \mathcal{G})$, Network $\mathcal{N}$, Error Budget ϵ_{total}
Output: Qubit allocation η, Code distances $\{d_q\}$
$U \leftarrow \mathcal{Q}$ {Set of unassigned logical qubits}
$RemCap_i \leftarrow c_i$ for all $i \in \{1 \dots k\}$ {Track remaining processor capacity}
while $U \neq \emptyset$ **do**
  BestPair $\leftarrow$ null, MinCost $\leftarrow \infty$
  for each $q \in U$ **do**
    for each $P_i \in \mathcal{N}$ **do**
      Estimate physical cost $s_{q,i}$ (incorporating current remote gate estimates)
      if $s_{q,i} \leq RemCap_i$ **and** $s_{q,i} <$ MinCost **then**
        MinCost $\leftarrow s_{q,i}$
        BestPair $\leftarrow (q, P_i)$
      end if
    end for
  end for
  if BestPair is null **then**
    break {Exit if no valid mappings remain}
  end if
  $(q^*, P_i^*) \leftarrow$ BestPair
  $\eta(q^*) \leftarrow P_i^*$
  $RemCap_{i^*} \leftarrow RemCap_{i^*} - s_{q^*,i^*}$
  $U \leftarrow U \setminus \{q^*\}$
  {Optional: Re-calculate costs for U to reflect the new allocation of q^*}
end while
Do "Post Processing Step" from Algorithm 2
return η, $\{d_q\}$

Densest Processor First (DPF) Algorithm. The Densest Processor First (DPF) algorithm is inspired by the maximum density packing problem. Rather than ordering processors solely by their native PERs (BPF) or evaluating individual qubit-to-processor affinities (BCF), DPF prioritizes hardware based on its *hosting density*. We define density as the ratio of logical qubits assigned to a processor relative to the total physical capacity consumed by those qubits.

In each iteration, the algorithm performs a trial allocation for every available processor in the network. For each processor P_i, it solves a local Knapsack problem to determine the subset of unassigned qubits $S_i \subseteq U$ that can be hosted within its capacity c_i while satisfying the error budget. The processor that yields the highest density—calculated as $|S_i|/c'_i$ (where c'_i is the used capacity)—is selected and populated. This strategy is designed to exhaust the most resource-efficient hardware units first, ensuring that we maximize the "logical yield" of each physical node. DPF is particularly effective in networks where a processor might have a slightly higher PER but a significantly larger capacity, allowing it to aggregate logical qubits more efficiently than a small, ultra-low-noise node.

Performance Guarantees. We note that the above BCF and DPF heuristics can perform arbitrarily badly for the simple case of two processors.

VII. Multi-Qubit Encodings and Risk-Aware Allocation

While single-qubit error correction—where each logical qubit is isolated in its own code block—offers simplicity, it often suffers from low code rates (the ratio of logical to physical qubits). In contrast, *multi-qubit encoding* maps a group of k logical qubits into a unified block of n physical qubits. This approach can significantly reduce the physical-to-logical overhead and, in some code families, enable faster intra-block gates. However, this increased resource efficiency introduces significant engineering risks:

- *Catastrophic Failure Correlation*: Unlike single-qubit codes, where an uncorrectable error affects only one unit, a failure in a multi-qubit block leads to the simultaneous loss of all k logical qubits.
- *Uniform Protection Inefficiency*: Multi-qubit blocks typically apply a global distance d to all k qubits. This results in over-protection (and thus wasted resources) if only a subset of qubits in the block requires a high level of protection.
- *Classical Decoding Latency*: The syndrome search space in multi-qubit codes is significantly larger, potentially increasing the latency of the classical feedback loop.

We formulate a generalized version of the EA-DQC problem, referred to as ERA-DQC (Risk-aware EA-DQC), to incorporate these trade-offs. We focus on the interplay between catastrophic failure risk and resource efficiency, deferring the modeling of decoder latency to future work.

ERA-DQC Problem Formulation. In the ERA-DQC setting, each processor P_i is characterized not only by its native PER but also by a failure function $f_i(n)$, which models the probability of a damaging physical event on a block of n physical qubits over the circuit's lifetime.

Given: A circuit $\mathcal{C}$, a network $\mathcal{N}$, and the following constraints:

- ϵ: The global circuit error budget.
- π: The *risk threshold*—the maximum acceptable probability that a fraction r of the total logical qubits collapse simultaneously.

Objective: Determine a qubit allocation function η and a block-encoding scheme that minimizes total physical qubits used such that:

1) The total circuit error rate remains below ϵ.
2) The probability $P(\text{collapsed logical qubits} \geq r|\mathcal{Q}|) \leq \pi$, where $|\mathcal{Q}|$ is the number of circuit qubits.

Risk Model and Bounds. To satisfy the global risk threshold π, we enforce a conservative local bound across all encoding blocks. We define a *block failure* as any event where a fraction r or more of the k logical qubits within a single block B_i

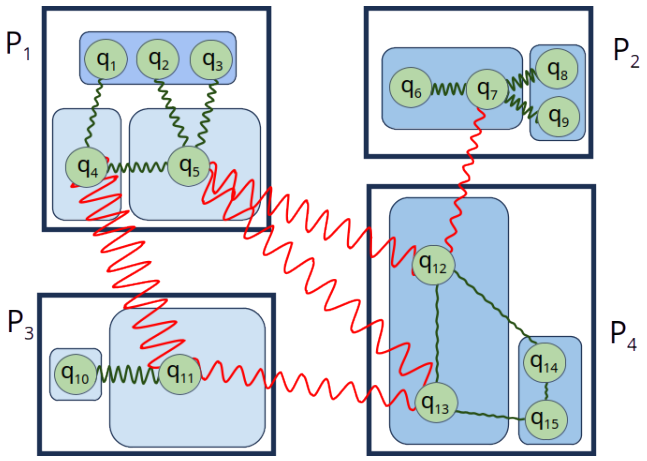

Fig. 3: An illustration of the ERA-DQC problem. Similar to Fig. 1, we encode logical qubits with different levels of protection and allocate them efficiently to processors. In this setting, sets of qubits can be encoded together within the same code block at the same protection level, thereby reducing physical cost. However, while cost-efficient, multi-qubit encodings may have a higher risk of logical qubit failure.

collapse. By ensuring that the probability of such a failure for any individual block is strictly less than π/M (where M is the total number of blocks in the allocation), the *Union Bound* guarantees that the probability of *at least one* block in the circuit failing remains below π.

For a $[\![n, k, d]\!]$ code block, we model the risk using the Chernoff bound. We first estimate the probability p of a single logical qubit failing within the block based on the physical error rate $f_i(n)$ and code distance d:

$$p \approx \binom{n/k}{(d+1)/2} \left(f_i(n)\right)^{\frac{d+1}{2}}$$

The probability that the number of logical errors X in the block exceeds the threshold $r \cdot k$ is then bounded as:

$$P(X \geq r \cdot k) \leq \left[\left(\frac{p}{r}\right)^r \left(\frac{1-p}{1-r}\right)^{1-r} \right]^k$$

Beyond the Linear Assignment Model of GAP. While the single-qubit aggregation model allows for a reduction to GAP, the transition to multi-qubit encodings moves the problem beyond the scope of linear assignment. This shift is necessitated by a synergistic complexity grounded in the domain-specific reality of fault-tolerant quantum block-encoding. In this multi-qubit regime, assignment costs are no longer independent; instead, they exhibit *sub-additivity*, where the marginal cost of adding a qubit to an existing block is often lower than the cost of a standalone assignment. This synergy, combined with the discrete, non-monotonic scaling of physical overhead required to satisfy the risk threshold π/M, motivates our subsequent approaches—in particular, the BCF-Set algorithm.

To solve the ERA-DQC problem, we adapt our core algorithms to navigate the trade-off between the high code rates of multi-qubit blocks and the stringent risk constraints.

Modifying BPF and DPF. For the BPF and DPF algorithms, the primary modification occurs within the subroutine responsible for packing an individual processor P_i. We replace the standard single-qubit Knapsack approach with a *Risk-Aware Block Packing* heuristic:

1) *Priority Sorting:* All unallocated logical qubits are sorted in descending order of their required code distance d. This ensures that qubits with the most demanding protection requirements are considered first.
2) *Block Construction:* The algorithm selects the first k qubits from the sorted list to form a candidate multi-qubit block $[\![n, k, d]\!]$. We then solve for the minimum physical footprint n such that:
   - The probability of a block failure (collapse of $\geq rk$ logical qubits) is strictly bounded by the local risk threshold π/M.
   - The physical-to-logical ratio n/k is minimized, ensuring the most efficient use of the processor's capacity while respecting the safety constraint.
3) *Iterative Filling:* This process repeats, moving down the sorted list of qubits until the remaining physical capacity c_i of processor P_i is exhausted.

Modifying BCF (Best Combination First). The BCF heuristic is extended to support *Flexible Block Integration*. When evaluating the cost of assigning a logical qubit q to a processor P_i, the algorithm considers two primary allocation modes:

1) *Singleton/New Block:* Initializing a new encoding block for q, as performed in the standard single-qubit case.
2) *Block Merging:* Merging q into an existing multi-qubit block already allocated to P_i. This requires recalculating the physical footprint n for the expanded block ($k \to k+1$) to ensure the risk constraint π/M is still satisfied for the larger group.

The algorithm greedily selects the configuration (either a new block or an expansion of an existing one) that yields the smallest marginal increase in total physical qubit consumption across the network.

Limitations of Prior Approaches. A potential drawback of the modified BPF and DPF algorithms is the focus on filling a single processor completely before considering others. In the context of multi-qubit encoding, this "local greediness" may lead to suboptimal block formations, in which high-risk qubits are forced into inefficient blocks simply to meet the current processor's capacity limits. Similarly, in the modified BCF, mapping decisions are made one qubit at a time to minimize the absolute marginal increase in physical qubits (Δn). However, in multi-qubit encoding, due to a non-linear relationship, a single qubit can trigger a jump in the required code distance for an entire block—in which case, a sequential approach may commit to a "cheap" individual assignment that results in a globally inefficient encoding density.

BCF-Set Algorithm. The Best Set-Combination First (BCF-Set) algorithm addresses these limitations by adopting a *block-greedy* approach. Rather than asking which processor to fill first or which single qubit is cheapest to place, BCF-Set searches for the specific (P_i, S_i) pair—where S_i is a subset of unassigned qubits—that maximizes the *logical yield* (minimizes n/k). By evaluating the density of a group

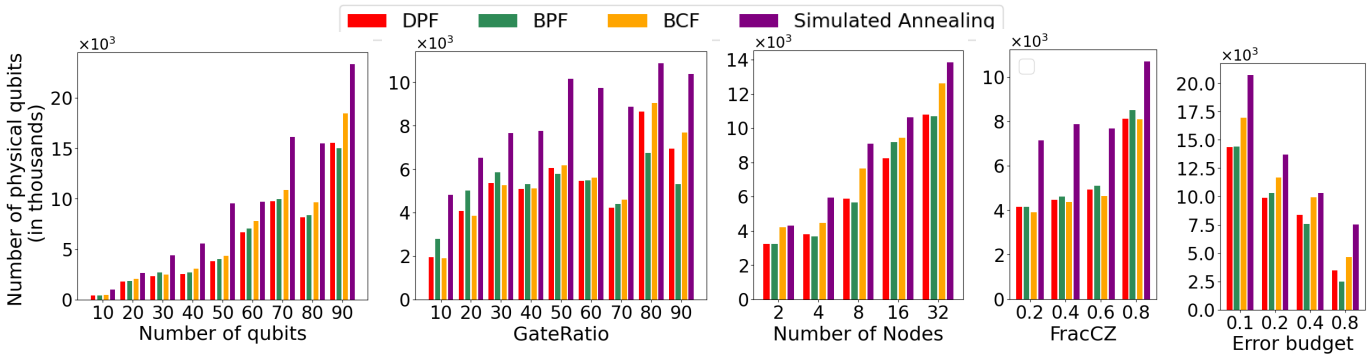

Fig. 4: Number of physical qubits used by various algorithms.

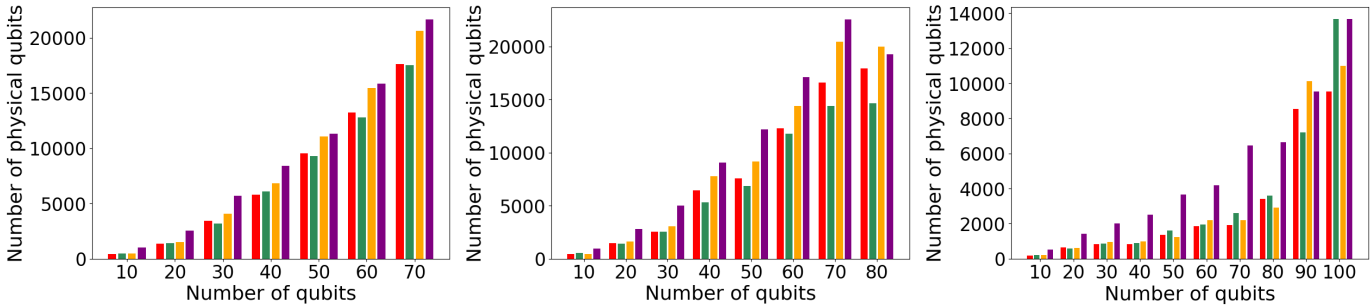

Fig. 5: Number of physical qubits used by various algorithms for benchmark circuits (a) QFT, (b) QPE, (c) GHZ.

rather than the cost of an individual assignment, `BCF-Set` identifies optimal sets of logical qubits that capitalize on the discrete scaling of quantum error correction. This ensures that processors are populated with their most efficient encoding configurations before resources are committed.

Specifically, `BCF-Set` identifies the (P_i, S_i) pair that minimizes the ratio $n_i/|S_i|$ as follows: In each iteration, the algorithm generates a candidate set S_i for every processor P_i. These sets are constructed by tentatively adding logical qubits—prioritizing those with the smallest marginal increase in physical overhead—and tracking the configuration that achieves the lowest local density. The algorithm then selects the global (P_i, S_i) pair with the minimum density across the entire network, ensuring that high-quality hardware is reserved for the most efficient multi-qubit encodings.

`BCF-Set` **Algorithm.** While unassigned qubits remain:

1) For every available processor P_i:
   a) Generate a candidate set of logical qubits S_i that minimizes the density ratio $n_i/|S_i|$, where n_i is the minimum physical footprint required to ensure:
      - The probability of $r\%$ of S_i collapsing is below the local risk threshold π/M.
      - The aggregate error of the qubits in S_i remains within their combined error budget.
   b) S_i is constructed by iteratively adding the unassigned qubit that yields the *lowest resulting density*. Since density is non-monotonic, we record the set size that achieves the absolute minimum density before the processor's capacity is reached.
2) *Global Selection:* Pick the (P_i, S_i) pair that achieves the minimum density across the entire network.

VIII. Evaluation

Algorithms Compared. For the single-qubit allocation problem (`EA-DQC`), we compare our proposed heuristics—`BPF`, `DPF`, and `BCF` (as described in §V–VI)—against a *Simulated Annealing (`SA`)* baseline. The `SA` algorithm initiates with a random allocation and iteratively explores neighboring states to minimize the physical footprint; we incorporate a post-processing step at each iteration to enforce capacity and error constraints. For the multi-qubit problem (`ERA-DQC`), we compare the modified (block-merging) versions of these heuristics against the `BCF-Set` algorithm.

Benchmarks and Parameters. We evaluate the algorithms using random quantum circuits with a default size of $|Q| = 50$ logical qubits, 50 gates per qubit, and a binary gate fraction of 0.5. The default total circuit error budget (ϵ) is 0.4. For the multi-qubit case, we assume a default risk probability $\pi/M = 0.01$ and a collapse threshold $r = 0.08$. Structured benchmarks include the *Quantum Fourier Transform (QFT)*, *Quantum Phase Estimation (QPE)*, and *GHZ state generation*, obtained from the Munich Quantum Toolkit [16].

Network Topology and Heterogeneity. Following the methodology in [19, 22], we model a quantum network distributed over a 100×100 km^2 area using the *Waxman model* [23]. The default network size is 8 nodes, with a maximum of 32. To simulate hardware heterogeneity, native PERs for processors are sampled uniformly from the range $[0.001, 0.01]$. Processor capacities are distributed so that the first $k-1$ processors host a total of $100 \times |Q|$ memory units, while the final processor is assigned unlimited capacity to ensure feasibility. Network parameters, including BSM success rates and latencies, are consistent with the models in [20].

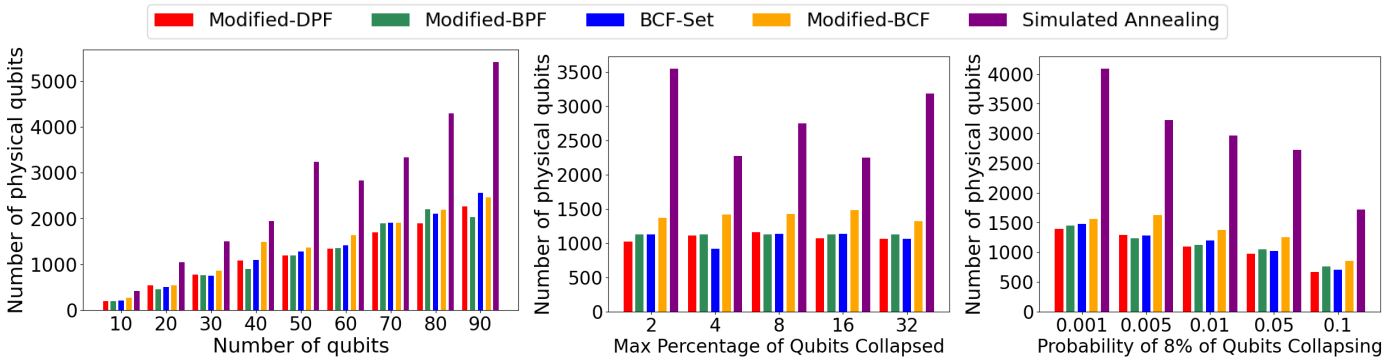

Fig. 6: Number of physical qubits used by various multi-qubit encoding algorithms.

Results and Observations. In the single-qubit setting, the performance hierarchy is generally BPF $\approx$ DPF $>$ BCF $>$ SA. On average, the processor-centric approaches BPF and DPF achieve a physical footprint reduction of approximately 10% relative to the qubit-centric BCF and up to 50% relative to the SA baseline. In some cases, e.g., for high gate ratios (see Fig. 4(b)) and large QPE instances (see Fig. 5(b), BPF significantly outperforms DPF too. Overall, the observations indicate that, in heterogeneous networks, prioritizing the use of high-fidelity "premium" processors—as BPF does—minimizes the global footprint more effectively than prioritizing the placement of high-risk qubits. The randomized search of SA fails to navigate tight capacity constraints effectively, often yielding suboptimal mappings that require higher protection levels.

Multi-Qubit Synergy. In the multi-qubit regime (ERA-DQC), the performance hierarchy shifts: BCF-Set, Modified-DPF, and Modified-BPF achieve comparable performance, while Modified-BCF performs approximately 10% worse. See Fig. 6. This indicates that while processor-centric strategies remain highly effective, the introduction of the block-greedy "set" strategy—which evaluates the density of a group rather than an individual assignment—enables qubit-centric heuristics to overcome their previous limitations and match the performance of the best-performing configurations. By identifying optimal multi-qubit encodings, these algorithms leverage the sub-additive costs of block encoding. Notably, all four heuristics outperform the SA baseline by a significant margin, confirming that the combinatorial approach is far more robust than stochastic search for navigating the non-monotonic efficiency jumps inherent in multi-qubit error correction.

IX. Conclusion and Future Work

We introduced a systematic framework for optimizing physical resources in fault-tolerant distributed quantum computing. Our combinatorial strategies, including the BPF approach, leverage intrinsic noise hierarchies to outperform stochastic baselines and achieve optimal solutions in several hardware-specific cases. We also showed that multi-qubit block-encodings introduce synergistic complexities requiring density-aware optimization. Future work will explore full hardware heterogeneity across disparate qubit modalities, the latency trade-offs of varying code rates, and the application of these techniques to quantum sensor networks [12, 25, 26].

References

[1] P. A. Martinez and C. Heunen. Automated distribution of quantum circuits via hypergraph partitioning. *Phys Rev. A*, 2019.

[2] R. Andonov et al. Unbounded knapsack problem: Dynamic programming revisited. *European J. of Opr. Res.*, 123(2), 2000.

[3] S. Babaie and C. Qiao. Towards distributed quantum error correction for distributed quantum computing. In *QCNC*, 2025.

[4] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah. On the qubit routing problem. In *TQC*, 2019.

[5] D. Cuomo et al. Optimized compiler for distributed quantum computing. *ACM Trans. on Quantum Computing*, 2023.

[6] O. Daei, K. Navi, and M. Zomorodi-Moghadam. Optimized quantum circuit partitioning. *Intl. J. of Theoretical Phys.*, 2020.

[7] X. Fan et al. Optimized distribution of entanglement graph states in quantum networks. *IEEE TQE*, 2025.

[8] D. Ferrari et al. Compiler design for distributed quantum computing. *IEEE Transactions on Quantum Engineering*, 2021.

[9] R. G. Sundaram, H. Gupta, and CR Ramakrishnan. Efficient distribution of quantum circuits. In *DISC*, 2021.

[10] M. Ghaderibaneh et al. Efficient quantum network communication using optimized entanglement swapping trees. *IEEE Transactions on Quantum Engineering*, 2022.

[11] Mohammad Ghaderibaneh et al. Generation and distribution of GHZ states in quantum networks. In *IEEE QCE*, 2023.

[12] Mark Hillery, Himanshu Gupta, and Caitao Zhan. Discrete outcome quantum sensor networks. *Physical Review A*, 2023.

[13] Y. Li et al. Hierarchical surface code for network quantum computing with modules of arbitrary size. *Phys. Rev. A*, 2016.

[14] Y. Mao et al. Probability-aware qubit-to-processor mapping in distributed quantum computing. In *QuNET*, 2023.

[15] E. Nikahd et al. Automated window-based partitioning of quantum circuits. *Phys. Scripta*, 2021.

[16] Nils Q. et al. Mqt bench: Benchmarking software and design automation tools for quantum computing. *Quantum*, 2023.

[17] G. Ross and R. Soland. A branch and bound algorithm for the generalized assignment problem. *Mathematical Prog.*, 1975.

[18] M. Y. Siraichi et al. Qubit allocation as a combination of subgraph isomorphism and token swapping. *OOPSLA*, 2019.

[19] R. Sundaram et al. DQC-QR: Distributing and routing quantum circuits with minimum execution time. *ACM TQC*, 2025.

[20] R. G. Sundaram et al. Distribution of quantum circuits over general quantum networks. In *QCE*. IEEE, 2022.

[21] R. G. Sundaram and H. Gupta. Distributing quantum circuits using teleportations. In *QSW*. IEEE, 2023.

[22] R. G. Sundaram and H. Gupta. Dynamic distribution of quantum circuits with minimum execution time. In *QCNC*. IEEE, 2025.

[23] B. M. Waxman. Routing of multipoint connections. *IEEE journal on selected areas in communications*, 1988.

[24] M. Yagiura et al. An ejection chain approach for the generalized assignment problem. *INFORMS J. on Computing*, 2004.

[25] C. Zhan et al. Optimizing initial state of detector sensors in quantum sensor networks. *ACM TQC*, 2024.

[26] Caitao Zhan and Himanshu Gupta. Quantum sensor network algorithms for transmitter localization. In *IEEE QCE*, 2023.