Towards Quantum Belief Propagation for LDPC Decoding in Wireless Networks

Srikar Kasi and Kyle Jamieson
{skasi,kylej}@cs.princeton.edu
Department of Computer Science, Princeton University

ABSTRACT

We present Quantum Belief Propagation (QBP), a Quantum Annealing (QA) based decoder design for Low Density Parity Check (LDPC) error control codes, which have found many useful applications in Wi-Fi, satellite communications, mobile cellular systems, and data storage systems. QBP reduces the LDPC decoding to a discrete optimization problem, then embeds that reduced design onto quantum annealing hardware. QBP's embedding design can support LDPC codes of block length up to 420 bits on real state-of-the-art QA hardware with 2,048 qubits. We evaluate performance on real quantum annealer hardware, performing sensitivity analyses on a variety of parameter settings. Our design achieves a bit error rate of 10^{-8} in 20 μs and a 1,500 byte frame error rate of 10^{-6} in 50 μs at SNR 9 dB over a Gaussian noise wireless channel. Further experiments measure performance over real-world wireless channels, requiring 30 μs to achieve a 1,500 byte 99.99% frame delivery rate at SNR 15-20 dB. QBP achieves a performance improvement over an FPGA based soft belief propagation LDPC decoder, by reaching a bit error rate of 10^{-8} and a frame error rate of 10^{-6} at an SNR 2.5–3.5 dB lower. In terms of limitations, QBP currently cannot realize practical protocol-sized (e.g., Wi-Fi, WiMax) LDPC codes on current QA processors. Our further studies in this work present future cost, throughput, and QA hardware trend considerations.

CCS CONCEPTS

• **Networks → Wireless access points, base stations and infrastructure**; • **Hardware → Quantum computation**.

KEYWORDS

Wireless Networks, Channel Coding, LDPC Codes, Belief Propagation, Quantum Annealing, Embedding, Quantum Computation.

1 INTRODUCTION

As the design of mobile cellular wireless networks continues to evolve, time-critical baseband processing functionality from the base stations at the very edge of the wireless network is being shifted and aggregated into more centralized locations (*e.g.,* Cloud/-Centralized-RAN [15, 44, 63]) or even small edge datacenters. A key component of mobile cellular baseband processing is the error correction code, a construct that adds parity bit information to the data transmission in order to correct the bit errors that interference and the vagaries of the wireless channel inevitably introduce into the data. In particular LDPC codes, first introduced by Gallager[25] in 1962 but (with few exceptions [49, 67, 77]) mostly ignored until the work of McKay *et al.* in the late 90s [48], have approached the Shannon rate limit [60]. Along with Turbo codes [6], LDPC codes stand out today because of their exceptional error correcting capability even close to capacity, but their decoding comprises a significant fraction of the processing requirements for a mobile cellular base station. LDPC codes are considered for inclusion in the 5G New Radio traffic channel [24], the DVB-S2 standard for satellite communications [55], and deep space communications [11, 12]. LDPC codes are also currently utilized in the most recent revisions of the 802.11 Wi-Fi protocol family [32]. Given the dominance of LDPC codes in today's wireless networks, the search for computationally efficient decoders and their ASIC/FPGA realization is underway.

Background: Quantum Annealing. This paper notes exciting new developments in the field of computer architecture hold the potential to efficiently decode LDPC codes: recently, *quantum annealer* (QA) machines previously only hypothesized [36, 52] have been commercialized, and are now available for use by researchers. QA machines are specialized, analog computers that solve NP-complete and NP-hard optimization problems in their Ising specification [9] on current hardware, with future potential for substantial speedups over conventional computing [53]. They are comprised of an array of physical devices, each representing a single *physical qubit* (quantum bit), that can take on a continuum of values, unlike classical information bits, which can only take on binary values. The user of the QA inputs a set of desired pairwise *constraints* between individual qubits (*i.e.,* a slight preference that two particular qubits should differ, and/or a strong preference that two particular qubits should be identical) and preferences that each individual qubit should take a particular classical value (0 or 1) in the solution the machine outputs. The QA then considers the entire set of constraints as a large optimization problem that is typically expressed as a quadratic polynomial of binary variables [36, 47]. A multitude of quantum annealing *trials* comprises a single machine *run*, with each anneal trial resulting in a potentially different solution to the

problem: a set of classical output bits, one per qubit, that best fits the user-supplied constraints on that particular trial.

Quantum-Inspired and Hybrid Algorithms. The growing interest in quantum computing has recently led to the emergence of several physics-based **quantum-inspired** algorithms (QIA) [4, 5, 28, 38] and quantum-classical **hybrid** algorithms (QCH) [33, 51, 66, 69]. QIA can be used to simulate quantum phenomena such as *superposition* and *entanglement* on classical hardware [54], where widely practiced QIA approaches (*e.g., digital annealing* [4, 50]) have solved combinatorial optimization problems with as many as 8,192 problem variables [50]. QCH algorithms broadly operate on a hybrid workflow between classical search heuristics and quantum queries, providing ways to use noisy intermediate-scale quantum computers [59] for optimizing problems with as many as 10,000 variables [17]. In this work, while we demonstrate a quantum annealing based LDPC decoder approach by realizing a small 700 variable problem, we also note that implementation of the same ideas using QIA and QCH methods is also a promising possibility.

This paper presents *Quantum Belief Propagation* (QBP), a new uplink LDPC decoder that takes a new look at error control decoding, from the fresh perspective of the quantum annealer. QBP is a novel way to design an LDPC decoder that sets aside traditional *belief propagation* (BP) decoding, instead *reduces* the first principles of the LDPC code construction in a highly-efficient way directly onto the physical grid of qubits present in the QA we use in this study, the D-Wave 2000-qubit (DW2Q) quantum adiabatic optimizer machine, taking into account the practical, real-world physical qubit interconnections. We have empirically evaluated QBP on the real DW2Q QA hardware. Results on the real-world quantum annealer show that QBP achieves a bit error rate of 10^{-8} in 20 μs and a 1,500 byte frame error rate of 10^{-6} in 50 μs at signal-to-noise ratio of 9 dB over a Gaussian noise channel. In comparison with FPGA-based soft BP LDPC decoders, QBP achieves the same 10^{-8} bit error rate and 10^{-6} frame error rate at an SNR 2.5–3.5 dB lower, even when the classical decoder is allowed a very large number of iterations (100). Currently, QBP cannot realize practical protocol-sized LDPC codes on state-of-the-art QA processors with 2,048 qubits. Our further studies present limitations and predicted future of QA (§9).

2 PRIMER: LDPC CODES

A binary (N, K) LDPC code is a linear block code described functionally by a sparse parity check matrix [25, 48]. It is said to be a (d_b, d_c)-regular code if every bit node participates in d_c checks, and every check has d_b bits that together constitute a *check constraint*. This section describes the conventional encoding and decoding schemes of LDPC codes. Let $\mathbf{H} = [h_{ij}]_{M \times N}$ be the LDPC check matrix. Each row in $\mathbf{H}$ represents a check node constraint whereas each column indicates which check constraint a bit node participates in. In the Tanner graph [67] of Figure 1, the nodes labeled c_i are check nodes and those labeled b_i are bit nodes, and a value of 1 at $h_{mn} \in \mathbf{H}$ represents an edge between c_m and b_n. Code *girth*, the length of the shortest cycle in the Tanner graph, is a crucial measure, as a low girth affects the independence of information exchanged between check and bit nodes, diminishing the code's performance [16, 46, 57].
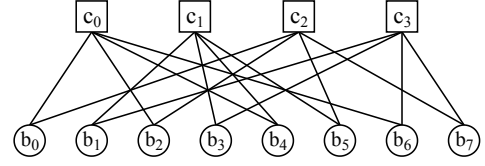

Figure 1: A *Tanner Graph* of an example LDPC code.

LDPC Encoder. Let u be a message of length K. The overall encoding process is summarized as follows:

(1) Convert $\mathbf{H}$ into *augmented form* $[\mathbf{P}|\mathbf{I}_{N-K}]$ by Gauss-Jordan elimination. (Here, $\mathbf{P}$ is obtained in the conversion process and $\mathbf{I}$ is the identity matrix.)
(2) Construct a generator matrix $\mathbf{G}$ as $[\mathbf{I}_K|\mathbf{P}^T]$.
(3) The encoded message c is constructed as $c = u\mathbf{G}$.

This way of encoding ensures that the modulo two bit-sum at every check node is zero [25].

LDPC Decoder. We describe the BP-based *min-sum* algorithm [76]. Let y be received information, $N(c_m)$ the set of bit nodes participating in check constraint c_m, and $M(b_n)$ the set of check nodes connected to bit node b_n.

Initialization. Initialize all the bit nodes with their respective *a priori* log-likelihood ratios (LLRs) as:

$$LLR_{b_n}(x_n) = \log\left(\frac{Pr(b_n = 0|y)}{Pr(b_n = 1|y)}\right) \quad \forall b_n \in N(c_m) \quad (1)$$

Step 1. For every combination $\{(m, n) \mid h_{mn} = 1\}$, initialize messages sent to check c_m from bit $b_n \in N(c_m)$ as:

$$Z_{b_n \to c_m}(x_n) = LLR_{b_n}(x_n) \quad (2)$$

Step 2. Every check node c_m then updates the message to be sent back, w.r.t every $b_n \in N(c_m)$ as:

$$Z_{c_m \to b_n}(x_n) = \prod_{b_{n'} \in N(c_m) \backslash b_n} \text{sgn}(Z_{b_{n'} \to c_m}) \cdot \min|Z_{b_{n'} \to c_m}| \quad (3)$$

Step 3. Each bit node b_n now updates the message to send back, w.r.t every $c_m \in M(b_n)$ as:

$$Z_{b_n \to c_m}(x_n) = LLR_{b_n}(x_n) + \sum_{c_{m'} \in M(b_n) \backslash c_m} Z_{c_{m'} \to b_n}(x_n) \quad (4)$$

To decode, each bit node computes:

$$Z_{b_n}(x_n) = LLR_{b_n}(x_n) + \sum_{c_m \in M(b_n)} Z_{c_m \to b_n}(x_n). \quad (5)$$

Decision Step. After Step 3, quantize $\hat{\mathbf{x}} = [\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_{N-1}]$ such that $\hat{x}_n = 0$ if $Z_{b_n}(x_n) \geqslant 0$, else $\hat{x}_n = 1$. $\hat{\mathbf{x}}$ are the decoded bits. If $\hat{\mathbf{x}}$ satisfies the condition enforced at encoding ($\hat{\mathbf{x}}\mathbf{H}^T = 0$), then $\hat{\mathbf{x}}$ is declared as the final decoded message. If it doesn't satisfy this condition, the BP algorithm iterates Steps 1–3 until a satisfactory $\hat{\mathbf{x}}$ is obtained. The decoder terminates at a predetermined threshold number of iterations.

3 CLASSICAL BP DECODER LIMITATIONS

The goal of most classical BP LDPC decoders is an efficient hardware implementation that maximizes throughput, thus driving a need to minimize data errors. A variety of architectures for the classical hardware implementation of LDPC decoders have been developed [26, 29, 62], and in practice, depending on the problem of interest and the hardware resource availability, the decoders are implemented either in serial, partly-parallel, or fully parallel architectures on FPGA/ASIC hardware. Although existing decoders do reach theoretically supported line speeds of, *e.g.* Wi-Fi [30], Wi-MAX [31], and DVB-S/S2 [23], they make throughput compromises, in particular, reducing decoding precision (such as using low precision LLR bitwidth, limiting iterations, or using reduced-complexity algorithms [26]). Therefore, the goal of maximizing throughput requires making the most efficient trade-offs among the following:

(1) To achieve high throughput, a high degree of **decoding parallelism** is required, demanding more resources in the silicon hardware implementation.

(2) Accurate decoding results require high **LLR bit precision** (*ca.* $8 - 10$), along with a precise decoding algorithm, again demanding more hardware resources.

(3) The iterative nature of the BP algorithm impedes throughput by requiring **numerous serial iterations** before reaching the best, final result. Thus a trade-off between iteration limit and throughput must be made.

These tradeoffs induce network designers to compromise between decoder operation line rate and precision, within the available limited silicon hardware resources. *Block RAMs* (BRAMs) are the fundamental array storage resources in FPGAs, where state-of-the-art BRAMs have a read and a write port with independent clocks, implying that a single BRAM can perform a maximum of two read/write operations in parallel [2, 74]. Therefore, to realize a high degree of parallelism required in protocol sized LDPC codes, many BRAMs must be used in parallel to access the BP LLRs. Furthermore to meet FPGA device timing constraints, today's dual-ported support for BRAMs limits the size of a single data access to 2,048 bits and the number of BRAMs accessible in a single clock cycle to 1,024 [37, 68, 74]. This limitation results in the maximum degree of achievable parallelization in current top-end Xilinx FPGAs, which corresponds to a 2,048 (1,024 × 2) LDPC code block length. However, practical block lengths reach up to 1,944 bits in Wi-Fi, 2,304 bits in Wi-MAX, and 64,800 bits in DVB-S2 protocol standards [23, 30, 31].

A Xilinx FPGA Resource Study. Using the Xilinx synthesis tool *Vivado HLS*, we have implemented a min-sum algorithm based decoder for a $\frac{1}{2}$-rate, 1944 block-length, (3, 6)-regular LDPC code, on the Xilinx Virtex Ultrascale 440 (*xcvu440*, the most resourceful Xilinx FPGA) with 8-bit LLR precision. The resource measurement metric in FPGAs is generalized to a *Configurable Logic Block (CLB)*. Each CLB in the Ultrascale architecture contains eight six-input LUTs[1], and 16 flip-flops along with arithmetic carry logic and multiplexers [73]. Our implementation of this fully parallel LDPC decoder covers ≈ 72% (229,322/316,220) of the CLBs in the device, the upper limit of reliability in terms of resource utilization. Furthermore, our

---

[1]Most recent FPGAs are equipped with six-input LUTs, which is equivalent to 1.6× the resources of a four-input LUT [26, 73].
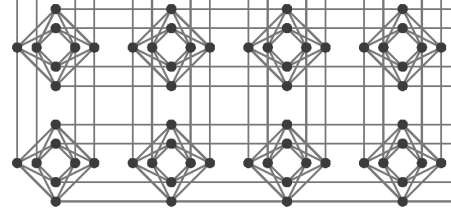

Figure 2: A portion of the Chimera qubit connectivity graph of the DW2Q QA, showing qubits (nodes in the figure) grouped by *unit cell*. The edges in the figure are *couplers*.

HLS implementation of a (4,8)-regular LDPC code of block length 2048 bits (fully parallel decoder with 8-bit LLR precision) does not fit into that FPGA.

4 PRIMER: QUANTUM ANNEALERS

Quantum Annealing is a heuristic approach to solve combinatorial optimization problems and can be understood at a high level as solving the same class of problems as the more familiar *simulated annealing* [42] techniques. QA takes advantage of the fundamental fact that any process in nature seeks a minimum energy state to attain stability. Given a discrete optimization problem as input, a QA *quantum processor unit* (QPU) internally frames it as an energy minimization problem and outputs its ground state as the solution.

Quantum Annealing Fundamentals. In the QA literature, qubits are classified into two types: *physical* and *logical*. A physical qubit is a qubit that is directly available physically on the QA hardware, while a logical qubit is a set of physical qubits. It is often the case that the QA hardware lacks a *coupler* between a particular pair of physical qubits that the user would like to correlate. To construct such a relationship, it is general practice to use intermediate couplers to make several physical qubits behave similarly, as explained below in §4.2, a process known as *embedding*. The set of these similarly behaving embedded physical qubits is then referred to a *logical qubit*. The process of evolution of quantum bits to settle down at the ground state in the DW2Q QA is called an *anneal*, while the time taken for this evolution is called the *annealing time*. The strength of the preference given to each single qubit to end up in a particular 0 or 1 state is a *bias*, while the strength of each coupler is called *coupler strength*. Moreover the strength of the couplers that are used to make physical qubits behave similarly as in the aforementioned embedding process, are called *JFerros*.

Quantum Annealing Hardware. The QA processor hardware is a network of interlaced radio-frequency superconducting quantum interference device flux qubits fabricated as an integrated circuit, where the local longitudinal fields (*i.e.,* biases) of the devices are adjustable with an external magnetic field and the interactions (*i.e.,* couplers) between pairs of devices are realized with a tunable magnetic coupling using a programmable on-chip control circuitry [35, 40]. The interconnection diagram of the DW2Q QA hardware we use in this study is a quasi-planar bi-layer *Chimera* graph. Fig. 2 shows a 2×4 portion of the 16×16 QA's Chimera graph: each set of eight physical qubits in the figure is called a Chimera *unit cell*, whereas each edge in the figure is a *coupler*.

The Annealing Process. QA processors simulate systems in the two-dimensional transverse field Ising model described by the time-dependent Hamiltonian:

$$H(s) = -A(s) \sum_i \sigma_i^x + B(s)H_P, \tag{6}$$

$$H_P = \sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z. \tag{7}$$

where $\sigma_i^{x,z}$ are the Pauli matrices acting on the ith qubit, h_i and J_{ij} are the problem parameters, $s = t/t_a$ where t is the time and t_a is the annealing time. $A(s)$ and $B(s)$ are two monotonic signals such that at the beginning of the anneal (*i.e., t = 0*), $A(0) >> B(0) \approx 0$ and at the end of the anneal (*i.e., t = t_a*), $B(1) >> A(1) \approx 0$. The annealing processor initializes every qubit in a *superposition* state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ that has no classical counterpart, then gradually evolves this Hamiltonian from time $t = 0$ until $t = t_a$ by introducing quantum fluctuations in a low-temperature environment. The time-dependent evolution of these signals A and B is essentially the annealing algorithm. During the annealing process, the system ideally stays in the local minima and probabilistically finds the global minimum energy configuration of the problem Hamiltonian H_P at its conclusion [3, 20].

4.1 QA Problem Forms

QA processors can be used to solve the class of *quadratic unconstrained binary optimization* (QUBO) problems in their equivalent Ising specification [9, 39], which we define here. The generalized Ising/QUBO form is:

$$E = \sum_i h_i q_i + \sum_{i<j} J_{ij} q_i q_j. \tag{8}$$

Ising form solution variables $\{q_i\}$ take values in $\{-1, +1\}$, and in QUBO form they take values in $\{0, 1\}$. The linear coefficient h_i is the bias of q_i, whereas the quadratic coefficient J_{ij} is the strength of the coupler between q_i and q_j. Coupler strengths can be used to make the qubits agree or disagree. For instance, let us consider an example Ising problem:

$$E = J_{12}q_1q_2, \quad (q_1, q_2 \in \{-1, +1\}). \tag{9}$$

Case I: $J_{12} = +1$. The energies for qubit states $(q_1, q_2) = (-1, -1)$, $(-1, +1)$, $(+1, -1)$, and $(+1, +1)$ are $+1$, -1, -1, and $+1$ respectively. Hence a strong **positive coupler strength** obtains a minimum energy of -1 when the two qubits **are opposites** of each other.
Case II: $J_{12} = -1$. The energies for qubit states $(q_1, q_2) = (-1, -1)$, $(-1, +1)$, $(+1, -1)$, and $(+1, +1)$ are -1, $+1$, $+1$, and -1 respectively. Hence a strong **negative coupler strength** obtains a minimum energy of -1 when the two qubits **agree** with each other.

4.2 Embedding of Logical Qubits

To visualize the relationship between logical and physical qubits, let us consider another example problem:

$$E = J_{12}q_1q_2 + J_{23}q_2q_3 + J_{13}q_1q_3. \tag{10}$$

Figure 3(a) is the direct graphical representation of this example problem. However, observe that a three-node, fully-connected graph structure does not exist in the Chimera graph (*cf.* Figure 2).
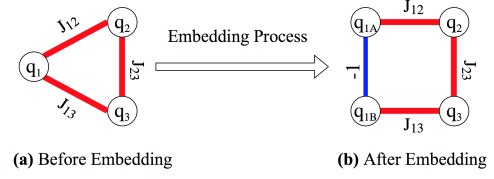

(a) Before Embedding   **(b)** After Embedding

Figure 3: The embedding process of Eq. 10, where the logical qubit q_1 in (a) is mapped onto two physical qubits q_{1A} and q_{1B} as in (b) with a JFerro of -1; here q_{1A} and q_{1B} agree.

Hence, the standard solution is to *embed* one of the logical qubits into a physical realization consisting of two physical qubits, as Figure 3(b) shows, such that we can construct each required edge in Figure 3(a). Here, logical qubit q_1 is mapped to two physical qubits, q_{1A} and q_{1B} with a JFerro of -1 to make q_{1A} and q_{1B} agree with each other.

5 DESIGN

In this section we first detail Quantum Belief Propagation's reduction of the LDPC decoding problem into a quadratic polynomial (QUBO) form (§5.1), and then present QBP's graph embedding model (QGEM) design on real QA hardware (§5.2).

5.1 QBP's LDPC to QUBO Reduction

Our QUBO reduction (§5.1.1) is a linear combination of two functions we have created: **(1)** an *LDPC satisfier* function (§5.1.2), and **(2)** a *distance* function (§5.1.3). During an anneal, the LDPC satisfier function leaves all the valid LDPC codewords in the zero energy level while raising the energy of all invalid codewords by a magnitude proportional to the LDPC code girth (§2). QBP's distance function distinguishes the true solution of the problem among all the valid LDPC codewords by separating them by a magnitude depending on the distance between the individual codeword and the received information (with channel noise).

System Model. Let $\boldsymbol{y} = [y_0, y_1, \ldots, y_{N-1}]$ be the received information corresponding to an LDPC-encoded transmitted message $\boldsymbol{x} = [x_0, x_1, \ldots, x_{N-1}]$. Let V be the set of all check constraints c_i of this LDPC encoding. Furthermore, let the final decoded message be the final states of the qubits $[q_0, q_1, \ldots, q_{N-1}]$ respectively, and let any $q_{e_i} \forall$ $i > 0$ be an *ancillary qubit* used for calculation purposes. Any given binary string is said to be a *valid codeword* when it checks against a given parity check matrix, and an *invalid codeword* otherwise.

5.1.1 QBP's objective function. QBP's QUBO objective function comprises two terms, an *LDPC satisfier* function $\sum_{\forall c_i \in V} L_{sat}(c_i)$ to prioritize solutions that satisfy the LDPC check constraints (*i.e.,* $L_{sat}(c_i) = 0$), and a *distance* function $\sum_{j=0}^{N-1} \Delta_j$ to calculate candidate solutions' proximity to the received information. The entire QUBO function is a weighted linear combination of these two terms:

$$\min_{q_i} \left\{ W_1 \sum_{\forall c_i \in V} L_{sat}(c_i) + W_2 \sum_{j=0}^{N-1} \Delta_j \right\} \tag{11}$$

Here, W_1 is a positive weight used to enforce LDPC-satisfying constraints, while the positive weight W_2 increases the success probability of finding the ground truth [34].
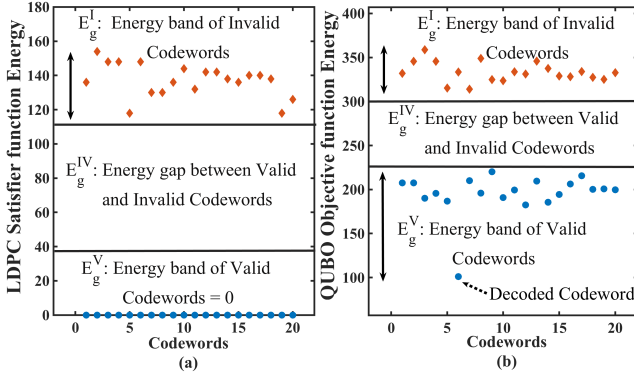
Figure 4: (a) LDPC satisfier function creating an energy gap between valid and invalid codewords. **(b)** QBP's objective function seperating the energy bands of both the valid and invalid LDPC codewords, to correctly decode.

The overall mechanism is depicted in Fig. 4 with real data: computing the energy values of 20 valid and 20 invalid codewords drawn at random. In Fig. 4(a), we see an *energy gap* (whose magnitude is denoted E_g^{IV}) that our LDPC satisfier function creates between valid and invalid codewords. Note that E_g^{IV} is directly proportional to the girth (§2) of the LDPC code (*i.e.*, if the girth of the code is low, there exists an invalid codeword which fails lesser number of check constraints, thus implying a low energy gap E_g^{IV}). Increasing W_1 in Eq. 11 increases this energy gap, thus eliminating invalid codewords as potential solutions. We observe in Fig. 4(b) that the distance function distinguishes the actually-transmitted codeword from other valid (but not transmitted) codewords that would otherwise also land in the ground energy state. The distance function works by separating the energy levels of both the valid E_g^V and the invalid E_g^I codewords by a factor proportional to the design parameter W_2. We explore experimentally in §7 the impact of wireless channel SNR and the QA dynamic range on the best choice of W_1 and W_2.

5.1.2   LDPC satisfier function. The only LDPC encoding constraint is that the modulo-two bit-sum at every check node is zero, *i.e.*, that the sum be even. For each check node c_i we define the function:

$$L_{sat}(c_i) = \left(\left(\sum_{\forall j: h_{ij} = 1} q_j \right) - 2L_e(c_i) \right)^2 \forall c_i \in V, \quad (12)$$

The LDPC constraint is satisfied at check node c_i, if and only if $L_{sat}(c_i) = 0$. Here $L_e(c_i)$ is a function of ancillary qubits $\{q_{e_i}\}$ (defined in §5.1). We formulate L_e to use minimal number of ancillary qubits with the following minimization:

$$L_e(c_i) = \sum_{s=1}^{t} (2^{s-1} \cdot q_{e_{s+k}}) \quad (13)$$

$$t = \min_{n \in \mathbb{Z}} \{ 2^{n+1} - 2 \geq d(c_i) - (d(c_i) \mod 2) \}. \quad (14)$$

where $d(c_i)$ is the degree of c_i (*i.e.*, the number of bits in check constraint c_i). In Eq. 13, the value of k in $L_e(c_i)$ is the largest index of the ancillary qubit used while computing $L_e(c_{i-1})$, ensuring each ancillary qubit is only used once. This formulation of L_e is

Table 1: Ancillary qubits required versus check node degree.

Check node degree $d(c_i)$:	3	4–7	8–15	16–31
Ancillary qubits required:	1	2	3	4

the binary encoding of integers in the range $[0, 2^t - 1]$, where a single integer corresponds to a single ancillary qubit configuration. The number of ancillary qubits required per check node is given in Table 1. Upon expansion of Eq. 12, $L_{sat}(c_i)$ introduces both biases and couplers to the objective QUBO and hence require embedding on the Chimera graph.

5.1.3   Distance function. We define a distance Δ_i that computes the proximity of the qubit q_i to its respective received information y_i as:

$$\Delta_i = (q_i - Pr(q_i = 1 | y_i))^2. \quad (15)$$

In Eq.15, the probability that q_i should take a one value given the received soft information y_i, can be computed using the likelihood information obtained from the soft demapping of received symbols, for various modulations and channels [75]. For instance, for a BPSK-modulated $(0 \rightarrow -1, 1 \rightarrow +1)$ information transmitted over an AWGN channel with noise variance σ^2, this probability is given by $1/(1 + e^{-2y_i/\sigma^2})$.

Hence, we observe that Δ_i is lesser for the $q_i \in \{0, 1\}$ that has a greater probability of being the transmitted bit. Upon expansion of Eq. 15, we note that the distance function introduces only biases to the QUBO problem and hence do not require embedding due to the absence of coupler terms.

5.2   Embedding on Annealer Hardware

Section 4 above has described the process of embedding problems onto the QA in general terms. In this section, we explain how we embed QBP's QUBO reduction onto the Chimera graph of the DW2Q QA hardware. QBP's embedding design can make use of an arbitrarily-large hardware qubit connectivity, supporting LDPC code block lengths up to 420 bits on state-of-the-art DW2Q QA.

Let us assign 2D coordinate values $U(x, y)$ to each unit cell in the Chimera graph with the bottom-left most unit cell as the origin $U(0, 0)$. Here we define terminology:

- A Chimera unit cell $U(a, b)$ is said to be a *neighbor* of $U(x, y)$ if and only if $|x - a| + |y - b| = 1$, and let $\Lambda(x, y)$ denote the set of all neighbors of $U(x, y)$.
- An *intra-cell* embedding is an embedding where both participating qubits lie in the same Chimera unit cell.
- An *inter-cell* embedding is an embedding where one of the qubits belongs to $U(x, y)$, and the other participating qubit belong to a unit cell in $\Lambda(x, y)$.

QGEM: QBP's Graph Embedding Model. We structure our embedding scheme into two levels, *Level I* (§5.2.1) and *Level II* (§5.2.2). QBP's graph embedding model *(QGEM)* first maps the check constraints (*i.e.*, $L_{sat}(c_i)$) by constructing the Level-I embedding for all the available Chimera unit cells, and next it accommodates more check constraints via the Level-II embedding, using the idle qubits that were left out during the Level-I embedding. QGEM makes use of the entire qubit computational resources available in the DW2Q QA hardware leaving no qubit idle in the machine.
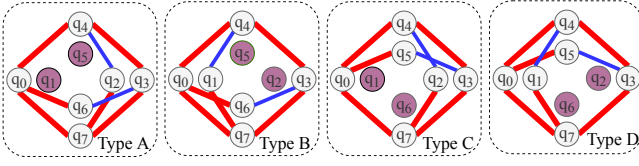
Figure 5: QBP's unit cell schemas for *Level-I* Chimera Graph embedding. Here (q_a, q_b, q_c, q_{e_1}) of Eq. 16 can be interpreted as (q_0, q_4, q_7, q_3) respectively in each schema. Idle qubits are shown in a darker shade. Embeddings are *thin-blue* lines and *thick-orange* lines are QUBO problem couplers.

In the Level-I embedding, QGEM represents a single check constraint (*i.e.*, each $L_{sat}(c_i)$) of at most degree three on a single Chimera unit cell using one of the four schemas presented in Fig. 5, which we refer to as *Types A–D*. Each of these schemas uses six qubits for a degree-three check constraint, leaving two qubits in the unit cell idle. Based on the coordinate location $U(x,y)$ of the unit cell, QGEM chooses a single schema for a single Chimera unit cell in a fashion that creates a specific pattern of idle qubits in the Chimera graph, then leverages this pattern to accommodate more check constraints as explained in §5.2.2. Next, QGEM places the check constraints that share a common bit closest to each other, then embeds the qubits representing this shared bit to make them agree, as described in Fig. 3 (§4.2). Specifically, if a check constraint c_i is placed in $U(x_0, y_0)$, then QGEM places the check constraints that share common bits with c_i in $\Lambda(x_0, y_0)$ and embeds the qubits representing such commonly shared bits via an inter-cell embedding (see dotted lines in Fig. 6(a)).

In the Level-II embedding, QGEM represents a single check constraint in an *ensemble* of nine Chimera unit cells using the pattern of idle qubits that the Level-I embedding leaves. The placement of each of these ensembles in the Chimera graph follows a similar fashion as in Level-I embedding (*i.e.*, placing the ensembles whose Level-II check constraints share bits close to each other).

We detail the overall working of QBP's graph embedding model more fully with a running example. Consider a $(2, 3)$-regular LDPC code: as the degree of each check node is three, let us assume that $[x_a, x_b, x_c]$ are the three bits participating in one of the check constraints c_i. Let $[q_a, q_b, q_c]$ be the *bit-node-representing* qubits used at the decoder to extract $[x_a, x_b, x_c]$ respectively. From Eqs. 12 and 13, the LDPC satisfying constraint of this check node is:

$$L_{sat}(c_i) = \left(q_a + q_b + q_c - 2q_{e_1}\right)^2 \tag{16}$$

5.2.1 Level-I Embedding. Upon expansion of Eq. 16, we observe that the quadratic terms (*i.e.*, qubit-pairs) requiring a coupler connectivity are $\{ (q_a, q_b), (q_a, q_c), (q_a, q_{e_1}), (q_b, q_c), (q_b, q_{e_1}), (q_c, q_{e_1}) \}$. QBP's Level-I embedding for the example in Eq. 16 can be visualized by interpreting (q_a, q_b, q_c, q_{e_1}) as equivalent to (q_0, q_4, q_7, q_3) respectively in Figs. 5 and 6. QBP realizes the above required coupler connectivity in four schemas presented in Fig. 5. We next demonstrate the Type A schema.

Construction. We construct the required-and-available coupler connectivity using the QA's direct physical couplers (*e.g.*, q_0 to q_4 in Type A, Fig. 5), and realize the required-but-unavailable coupler
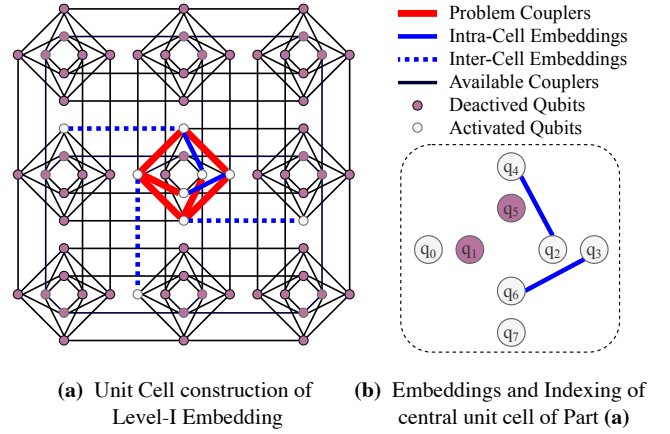

(a) Unit Cell construction of Level-I Embedding

(b) Embeddings and Indexing of central unit cell of Part (a)

Figure 6: QBP's Level-I Chimera Graph Embedding.

connectivity $\{(q_0, q_3), (q_4, q_7)\}$, using two intra-cell embeddings (*e.g.* q_2 to q_4 in Type A, Fig. 5).

Placement. Let us assume that QGEM chooses the above Type A schema for one of the Chimera unit cells whose placement is shown in Fig. 6(a). We note that the example LDPC code is $(2, 3)$-regular, and so every bit node participates in two check constraints. This implies that each bit-node-representing qubit (*i.e.*, excluding ancillary qubits) must be present in two Chimera unit cells since in the Level-I embedding, we represent a check constraint in a single Chimera unit cell. QGEM thus represents the other check constraint of each of these bit-node-representing qubits $\{q_0, q_4, q_7\}$ in a neighbor unit cell connected via an inter-cell embedding as depicted in Fig. 6(a), thus making the physical qubits involved in the embedding agree. QGEM repeats this construction over the entire Chimera graph, mapping each check constraint to an appropriate physical location in the QA hardware. QGEM selects the schema type to use (see Fig. 5) for each unit cell in a way that the two idle qubits of the Level-I unit cell schemas form the pattern as shown in Fig. 7(a).

5.2.2 Level-II Embedding. Let us continue with the example of Eq. 16. The overview of QBP's Level-II embedding is presented in Fig. 7. Here, in Level-II, the mapping of bits in the check constraint of Eq. 16 to physical qubits is (q_a, q_b, q_c, q_{e_1}) map to (q_A, q_B, q_C, q_E) respectively. In the Fig. 7, qubits $q_{A_i} \ \forall \ i \in [0, 3]$ represent q_A, $q_{B_i} \ \forall \ i \in [0, 2]$ represent q_B, $q_{C_i} \ \forall \ i \in [0, 2]$ represent q_C, and the qubits $q_{E_i} \ \forall \ i \in [0, 3]$ represent q_E, as they are embedded together as shown in Fig. 7(b). The pattern in the figure now allows us to realize all the required coupler connectivity of the example in Eq. 16 as depicted in Fig. 7(a). Similar to our Level-I placement policy, QGEM repeats this construction over the entire Chimera graph, mapping each Level-II check constraint to an appropriate physical location in the QA.

6 IMPLEMENTATION

We implement QBP on the DW2Q QA: our decoder targets a $(2, 3)$-regular maximum girth LDPC code of block length 420 bits. In the DW2Q, a *solver* is a resource that runs the input problem. We implement QBP remotely via the *C16-VFYC* hardware solver API,
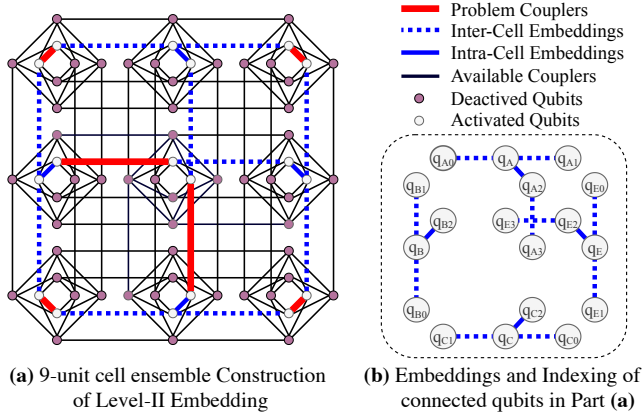
(a) 9-unit cell ensemble Construction of Level-II Embedding

(b) Embeddings and Indexing of connected qubits in Part **(a)**

Figure 7: QBP's Level-II Chimera Graph Embedding. In Fig. 7(b), (q_A, q_B, q_C, q_E) represent (q_a, q_b, q_c, q_{e_1}) of Eq. 16.

using the Python client library. This solver first maps the implementation of the problem at hand directly onto the DW2Q's QPU hardware, then determines the final states of the few (15 on our particular DW2Q) defective qubits via post-processing on integrated conventional silicon [22]. Since post-processing problem size is two orders of magnitude smaller than overall problem size, post-processing parallelizes with annealer computation and therefore does not factor into overall performance.

DW2Q readout fidelity is greater than 99%, and the chance of QPU programming error is less than 10% for problems that use all the available QA hardware resources [19]. However, we increase readout fidelity and decrease the chance of programming error via the standard method of running multiple anneals for every LDPC decoding problem, where each anneal reads the solution bit-string once. In our evaluation, we further quantify the unavoidable *intrinsic control errors* (§9) that arise due to the quantization effects and the flux noise of the qubits [19]. Our end-to-end evaluation results capture all the above sources of QA imprecision.

7 EVALUATION

Our experimental evaluation is on the DW2Q QA, beginning with our experimental methodology description (§7.1). We measure performance over a variety of DW2Q parameter settings (chosen in §7.2), and in both simulated wireless channels, and realistic trace-driven wireless channels. End-to-end experiments (§7.3) compare head-to-head against FPGA-based soft belief propagation decoding.

7.1 Experimental Methodology

Let us define an *instance I* as an LDPC codeword. Our evaluation dataset consists 150 instances with an overall 2×10^4 message bits. We conduct 10^4 anneals for each instance and note the distribution of the solutions returned along with their occurrence frequency. If N_s^I is the number of different solutions returned for an instance I, we rank these solutions in increasing order of their energies as $R_1, ..., R_{N_s^I}$ with R_1 being the rank of the minimum energy solution. All the N_s^I solutions can be treated as identically independent random variables, as each anneal is identical and independent.

7.1.1 BER Evaluation. Let R_{min} be the rank of the minimum energy solution in a particular population sample of the entire solution distribution, of size N_a ($<10^4$) anneals. We compute the expected *number of bit errors N_B^I* of an instance I over N_a anneals as:

$$E[N_B^I|N_a] = \sum_{i=1}^{N_s^I} \Pr(R_{min} = R_i|I, N_a) \cdot N_B^I(R_i|I, N_a), \quad (17)$$

where the probability of R_{min} being $R_i \; \forall \; i \in [1, N_s^I]$ for an instance I, over performing N_a anneals is computed using the cumulative distribution function $F(\cdot)$ of observed solutions in 10^4 anneals as [41]:

$$\Pr(R_{min} = R_i|I, N_a) = (1 - F(R_{i-1}))^{N_a} - (1 - F(R_i))^{N_a}, \quad (18)$$

Hence we compute the bit error rate (BER) of an instance I with K information bits upon performing N_a anneals as:

$$\text{BER} = E[N_B^I|N_a]/K. \quad (19)$$

7.1.2 FER Evaluation. Frame Construction: We construct a frame of length N_F using data blocks of length N_B, so we require N_F/N_B such blocks, where each block is an instance. If N_{ins} is the number of available instances, we can construct a single frame by combining any N_F/N_B instances among the available N_{ins} instances. Thus the total number of distinct frames we construct for our *frame error rate* (FER) evaluation is $\binom{N_{ins}}{N_F/N_B}$.
FER Calculation: A frame is error-free iff all the code blocks in the frame has zero errors, just as if it has a cyclic redundancy check appended. We compute the probability of a particular k^{th} frame being error-free ($Pr(F_{ef}^k)$) as:

$$\Pr\left(F_{ef}^k\right) = \prod_{I=1}^{N_F/N_B} \left\{ \sum_{\forall i} \Pr\left(R_{min} = R_i|I, N_a, N_B^I(R_i) = 0\right) \right\} \quad (20)$$

Then we compute the overall frame error rate (FER) as:

$$\text{FER} = \left[\sum_{k=1}^{\binom{N_{ins}}{N_F/N_B}} \left\{ 1 - \Pr\left(F_{ef}^k\right) \right\} \right] / \binom{N_{ins}}{N_F/N_B} \quad (21)$$

7.1.3 Wireless Trace-driven Evaluation. We collected channel traces from an aerial drone server communicating with a ground client in an outdoor environment, using the Intel 5300 NIC wireless chip at the client [27]. In realistic wireless transmissions, code blocks are transmitted over multiple OFDM symbols, where subcarriers within an OFDM symbol typically experience a diverse range of channels. In our performance evaluation over experimental channels, we compute the per-subcarrier SNR information through *channel state information* (CSI) readings, and distribute a corresponding Gaussian noise over bits individually for every subcarrier. Next we demodulate and interleave the data symbols and perform QBP's decoding. Hence we use the distance function (§5.1.3) for this evaluation with σ^2 equal to the noise variance experienced by y_i's subcarrier.

7.1.4 QA versus FPGA Throughput Evaluation. Consider a data frame with N_K message bits. Let us assume that QBP decodes this frame on the QA for a T_c *compute time*, and soft BP decodes the same frame on an FPGA with clock frequency f_{clk}, for N_{it} iterations. Let
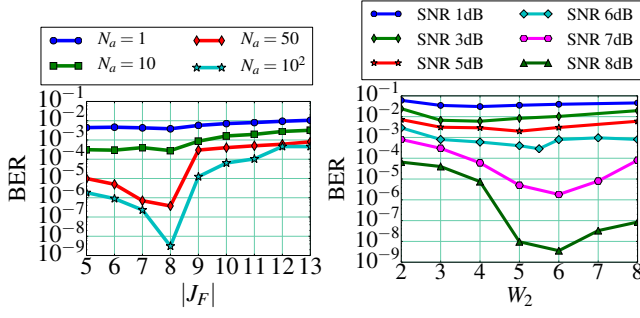
Figure 8: *Left.* Choosing *JFerro* strength to minimize BER. *Right.* Effect of W_2 on BER at various channel SNRs. The magnitude of W_2 that minimizes BER is proportional to SNR.

Figure 9: Choosing anneal time T_a. Figure depicts the probability of **not** finding the ground truth across distribution of problem instances. $T_a = 1\ \mu s$ is sufficient to achieve a high probability of finding ground truth.

$N_{clk/it}$ be the number of FPGA clock cycles the soft BP requires to complete an iteration. The actual throughput QBP achieves is then $(1 - FER_{QA}) \cdot N_K/T_c$, and the actual FPGA soft BP-based throughput is then $(1 - FER_{FPGA}) \cdot N_K \cdot f_{clk}/(N_{it} \cdot N_{clk/it})$.

The values of $N_{clk/it}$ and f_{clk} depend on the decoder implementation architecture (*i.e.,* serial or parallel) and FPGA hardware type. In order to make a throughput comparison between QA and FPGAs, we evaluate the QA throughput versus the best silicon realization (*i.e.,* a fully-parallel decoder, $N_{clk/it} = 1$) throughput on the highest specification Xilinx FPGA, for a range of FPGA clock frequencies and highlight the design-dependent operating-time regions (§7.3).

7.2 Parameter Sensitivity Analysis

In this section, we determine DW2Q QA's optimal system parameters, including JFerro ($|J_F|$), annealing time (T_a), number of anneals (N_a), and the design parameter W_2 for evaluating QBP's overall end-to-end system performance (§7.3).

7.2.1 Choice of Embedding Coupler Strength $|J_F|$. In the QA literature, the coupler strength of an embedding is termed JFerro (§4). As the fundamental purpose of embeddings is to make qubits agree, a large, negative JFerro is required in order to ensure the embedding is effective (§4.1). However, as the supported range for coupler strengths in DW2Q QA is $[-1, 1]$, it is general practice to normalize all QUBO coefficients with respect to $|J_F|$ to bring all the coupler strengths into this supported range $[-1, 1]$.

Consider a QUBO problem with coupler strengths in the range $[A, B]$. Then $|J_F|$ must be greater than $\max(|A|, |B|)$ to prioritize embeddings over problem couplers, and moderate enough to distinguish the new normalized coupler strengths $\left[\frac{A}{|J_F|}, \frac{B}{|J_F|}\right]$ as the range lessens. We perform our JFerro sensitivity analysis at a moderate SNR of 8 dB. We use a relatively high anneal time ($T_a = 299\ \mu s$), to ensure minimal disturbance from the time limit, and we choose our QUBO design parameters $W_1 = 1.0$ and $W_2 = 6.0$, experiments show that all other values of W_1 and W_2 results in similar trends for the JFerro sensitivity. Fig. 8 (*left*) depicts QBP's BER performance at various $|J_F|$ strengths. The BER curve of $N_a = \{50, 10^2\}$ anneals clearly depict that $|J_F| = 8.0$ minimizes BER, while for $N_a = \{1, 10\}$ anneals BER is barely minimized at $|J_F| = 8.0$, as the effect of $|J_F|$ is slight because of fewer anneals. Hence heretofore we set $|J_F| = 8.0$ for further evaluation.
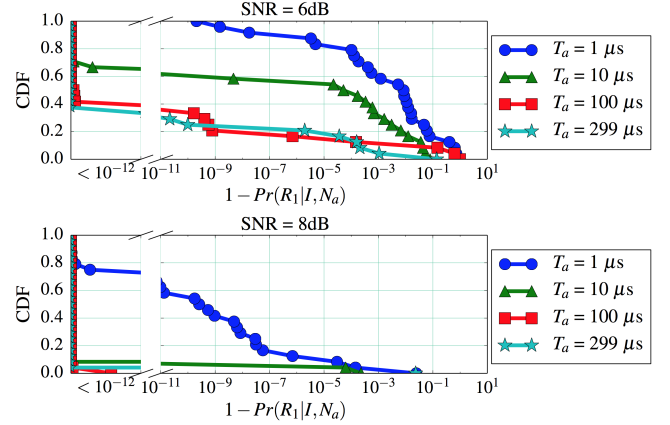
7.2.2 Choice of design parameter W_2. QBP's LDPC satisfier function (Eq. 12) introduces coupler strengths (*i.e.,* quadratic coefficients) greater than one, and hence must be normalized to bring all the problem coupler strengths into the supported $[-1, 1]$ range. Hence we set $W_1 = 1.0$ and consider the choice of W_2, the parameter that determines sensitivity to the received bits, in order to identify the correct codeword. We find the optimal value for W_2 dynamically with the wireless channel SNR, to balance between the received soft information and the LDPC code constraints.

We perform our W_2 sensitivity analysis at $|J_F| = 8.0$ (§7.2.1), $W_1 = 1.0$ (§7.2.2), and use a high anneal time ($T_a = 299\ \mu s$), to ensure minimal disturbance from the time limit. Fig. 8 (*right*) depicts QBP's BER performance at various SNRs while varying W_2. In the figure we observe that the magnitude of W_2 that minimizes BER, increases with increase in channel SNR. Hence QBP chooses W_2 at the time of data reception. As an incoming frame arrives, the receiver uses the packet preamble to estimate SNR, and then looks up the best W_2 for decoding in a lookup table.

7.2.3 Choosing the annealing time T_a. We perform our annealing time sensitivity analysis using $|J_F| = 8.0$ (§7.2.1) and $W_1 = 1.0$ (§7.2.2). We choose W_2 as above (§7.2.2) and perform $N_a = 10$ anneals (any number of anneals results in similar trends). Fig. 9 presents the probability of **not** finding the minimum energy solution over the cumulative distribution across problem instances. We find that an anneal time as low as one μs yields a high probability of finding the ground truth, hence we consider $T_a = 1\ \mu s$.

Heretofore we quantify QBP's performance over total *compute time T_c*, where $T_c = N_a \cdot T_a$. Fig. 10 depicts the combined result of the overall calibrations presented in (§7.2). Specifically, Fig. 10 shows the probability of **not** finding the minimum energy solution across the cumulative distribution of problem instances at wireless channel SNR 6 dB over various choices of W_2 and computing times (T_c). The figure shows that the best choice of W_2 results in a relatively low probability of **not** finding the ground truth, as well as the benefits of increasing compute time up to 100 μs.
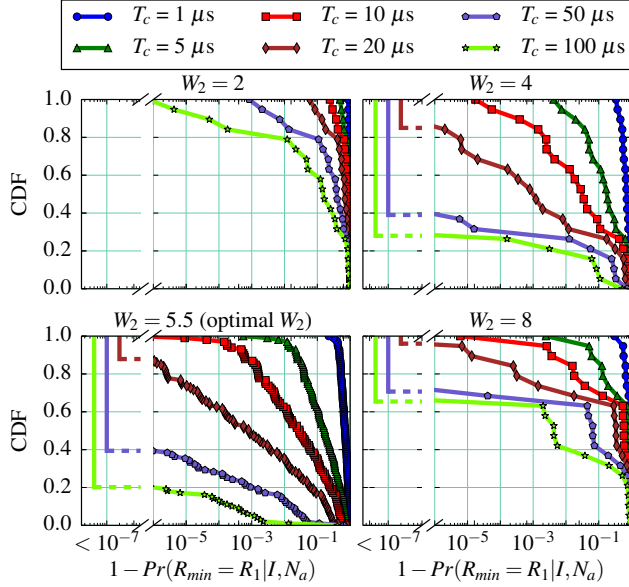
Figure 10: The effect of calibrations in (§7.2) at SNR 6 dB, depicting the probability of **not** finding the minimum energy state at $|J_F| = 8.0$. All plots share common x-y axes, and the distribution is across problem instances. The bottom–left plot corresponds to the best W_2 at SNR 6 dB (see Fig. 8 *right*).

7.3 System Performance

This section reports the QBP's end-to-end performance under the above system and design parameter choices (§7.2).

7.3.1 AWGN Channel Performance. We first evaluate over a Gaussian wireless channel at SNRs in the range 1–11 dB, comparing head-to-head against soft BP decoders operating within various iteration count limits.

Bit error rate performance. In Fig. 11(a), we investigate how average end-to-end BER behaves as the wireless channel SNR varies. At regions of channel SNRs less than 6 dB, QBP's performance lags that of conventional soft BP decoders operating at 20 and 100 iterations, and differences in QBP's performance at various QA computing times are barely distinguishable. This is because the optimal choice of W_2 at low SNRs is low (§7.2.2), thus making the probability of finding the ground truth low for a QA. However as we meet SNRs greater than 6 dB, we observe QBP's BER curves quickly drop down, reaching a BER of 10^{-8} at SNR 7.5–8.5 dB only, whereas conventional soft BP decoders acheive the same BER at an SNR of 10.5–11 dB. This is because the optimal choice of W_2 at high SNRs is high (§7.2.2), thus separating the ground truth and the rest with a high energy gap, making the true transmitted message easier to distinguish. Our QBP LDPC decoder acheives a performance improvement over a conventional silicon based soft BP decoder by reaching a BER of 10^{-8} at an SNR 2.5–3.5 dB lower.

Across problem instances. In Fig. 11(b), we investigate how bit errors are distributed among individual LDPC problem instances in the same parameter class. The figure shows that when the QBP decoder fails due to too-low QA compute time, bit error rates are

rather uniformly distributed across different problem instances. Conversely, increasing the computing time to 10–100 μs, the decoder drives BER low, so most instances have zero bit errors, and BER variation reduces. The result shows that $\{0, 28, 56, 73, 92, 98\}$ percent of instances under QBP's decoding are below the BER achieved by soft BP at QA compute times $\{1,5,10,20,50,100\}$ μs respectively.

Frame error rate performance. We investigate QBP's FER performance under frame sizes N_F of 420, and 12,000 bits. In Fig. 11(c), we observe a shallow FER error floor for SNRs less than 6 dB, noting the dependence of that error floor value on the frame length. When we meet an SNR of 8–9 dB, QBP acheives an FER of 10^{-6} with low dependence on the frame length and QA compute time, while soft BP achieves the same BER at an SNR 2–3 dB higher.

Throughput Analysis. An FPGA-based LDPC decoder is bounded by a maximum operating clock frequency (f_{clk}^{max}), the frequency beyond which the FPGA signal routing fails. Let us define the code block *solution time* T_{st} as the inverse of the minimum possible time to obtain a decoded solution (*i.e.,* T_a^{-1} for QA and $f_{clk} \cdot N_{clk/it}^{-1}$ for an FPGA). Fig.11(d) reports the throughputs. The figure shows that as the channel SNR increases, the throughput gap between QA ($N_a = 10$) and FPGAs ($N_{it} = 15$) tends toward a constant value whose magnitude is essentially the gap between the processing throughputs of QA and FPGAs, as the value of (1−FER) §7.1.4 tends toward one. The results imply that the QA can achieve a throughput improvement over the fastest FPGAs implementing a fully parallel decoder, when either the annealing time only improves roughly by 40×, or when the annealing time improves by 5× in combination with a 5.4× increase of qubit resources in the QA.

Fig.11(d) compares QBP against soft BP for a small code of 420 bits, thus f_{clk}^{max} achieved (56 MHz) is high enough for the FPGA to reach a throughput better than DW2Q QA. However, the value of f_{clk}^{max} significantly reduces as code block lengths increase, due to higher complexity of the decoder. Our FPGA implementation (fully parallel decoder, 8-bit LLR precision) of a (2,3)-regular LDPC code of block length 2048 bits achieves an f_{clk}^{max} of 17 MHz, while a (4,8)-regular similar LDPC code does not fit into that FPGA.

7.3.2 Trace-driven Channel Performance. Here we demonstrate QBP's performance in real world trace driven channels (§7.1.3).

Bit error rate performance. Fig. 12(a) depicts QBP's BER performance in trace-driven channels. For a given compute time, we observe the BER distribution across problem instances, and its dependency on the channel SNR. For channel average-SNRs in the range 5–10 dB, we observe that a few instances lie at a high BER of 10^{-2}, thus driving the mean BER high. As we step up to higher average SNRs greater than 10–15 dB, BER goes down very rapidly over increase in QA compute time for greater than 90% of problem instances, since there is less probability that channel subcarriers experience very low SNRs in this scenario.

Across problem instances. Drilling down into individual problem instances at a particular average SNR in the range 10–15 dB, we observe in Fig. 12(b) that more than 75% of the problem instances lie below the 10^{-8} BER at computing times 20–100 μs, while exhibiting an error floor spanning two orders of BER between 10^{-4} and 10^{-2} when the QA computing time is set to 1 μs (far less than general practices).

(a) Average BER, AWGN channel.

(b) CDF, AWGN channel.

(c) Average FER, AWGN channel.

Code block solution time (T_{st} in $(\mu s)^{-1}$): $T_{st} = T_a^{-1}$ (QA). $T_{st} = f_{clk} \cdot N_{clk/it}^{-1}$ (FPGA), $N_{clk/it} = 1$ (fully parallel decoder).

(d) Throughput comparison of QBP versus soft BP decoders for a (2,3)-regular code of block length 420 bits. In the figure, the hatched area is the operating-time region of QA, the colored (solid filled) area is the throughput gap between QA ($N_a = 10$) and FPGAs ($N_{it} = 15$), the dotted vertical line is the f_{clk}^{max} (56 MHz) achieved by our FPGA implementation (8-bit LLR precision, fully parallel decoder), and the dark horizontal line is the upper bound of FPGA throughput (N_{it}=15) imposed by f_{clk}^{max}. The data points from top to bottom of the QBP line in the figure correspond to $N_a = \{1,5,10,20,50,100\}$ anneals respectively in all the plots.
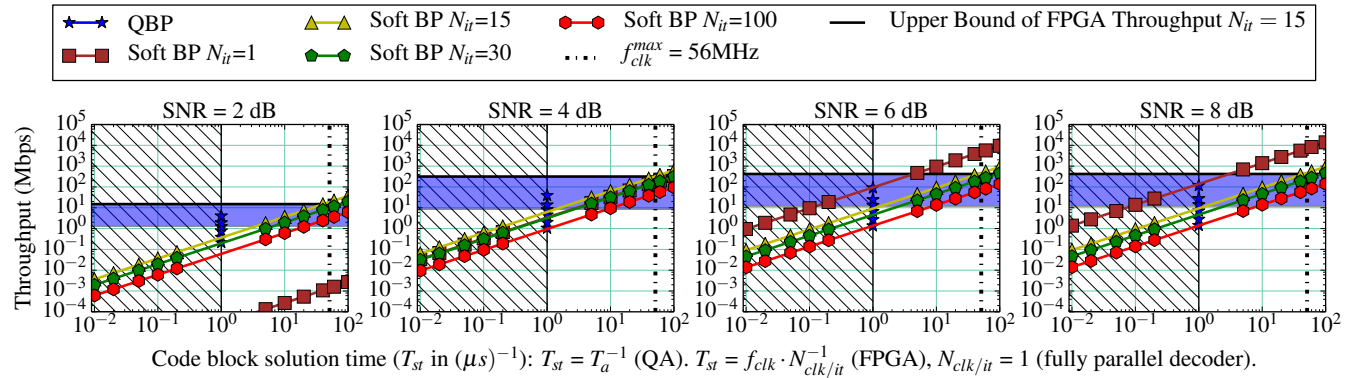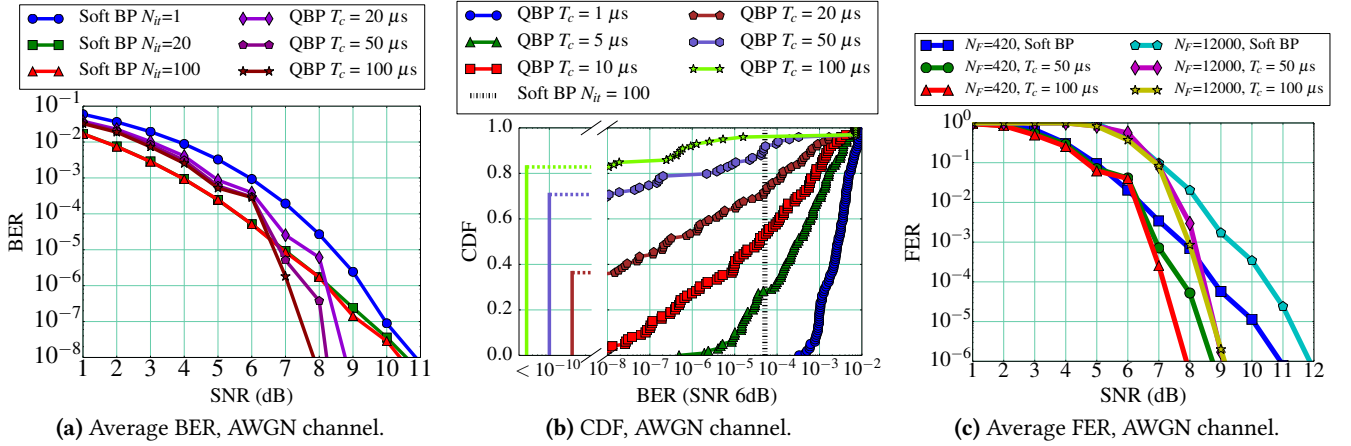
Figure 11: Quantum Belief Propagation's system performance in an AWGN channel. CDF in Fig. 11(b) is across individual LDPC problem instances. In Fig. 11(c), the frame size N_F is bits and the Soft BP iterations are 100. In Fig.11(d), all plots share common x-y axes.

Frame error rate performance. Fig. 13 depicts QBP's trace-driven channels' FER performance at various channel average-SNRs. Each box in the figure represents 10 different channel traces, where we compute FER by constructing $2 \times 10^2, 5 \times 10^6$ distinct frames (as mentioned in §7.1.2) for each channel trace when $N_F = 420$ and 12,000 bits respectively. We observe that FER exhibits an error floor when the average channel SNRs are less than 10–15 dB. FER drastically drops down for channel SNRs greater than 15 dB.

8 RELATED WORK

Bian *et al.* [7] present discrete optimization problem solving techniques tailored to QA, solving the LDPC decoding problem by dividing the Tanner graph into several *sub-regions* using min-cut heuristics, where a different QA run solves each sub-region. Bian *et al.* coordinate solutions of each run to construct the final decoded message. Conversely, QBP's approach differs with [7] both with respect to QUBO formulation and QA hardware embedding. The Bian *et al.* QUBO design does not adapt to both the wireless channel

noise (distance function §5.1.3) and the binary encoding minimization of the ancillary qubits (LDPC satisfier function §5.1.2). From embedding perspective, QBP can solve up to 280 check constraints in a single anneal while Bian *et al.* solves up to only 20 check constraints on an earlier QA with 512 qubits (which extends to 60–80 check constraints on the current QA with 2,048 qubits). Bian *et al.* evaluate over a binary symmetric channel (each sub-region run with $T_a = 20\mu s$) with crossover probabilities in the range of 8–14%, unrealistically high for practical wireless networks, nonetheless experiencing that only 4% out of 10^4 anneals had no bit errors, lower-bounding their BER by 10^{-3}. Lackey proposes techniques for solving Generalized BP problems by sampling a Boltzmann distribution [43], but does not venture into a performance evaluation. It is also possible to use the QBP's QUBO design (§5.1) as an input to D-Wave's built-in greedy search embedding tool [14], but this approach scales up to only 60 (2,3)-regular LDPC check constraints, which limits the LDPC code block length to an impractical 90 encoded bits.
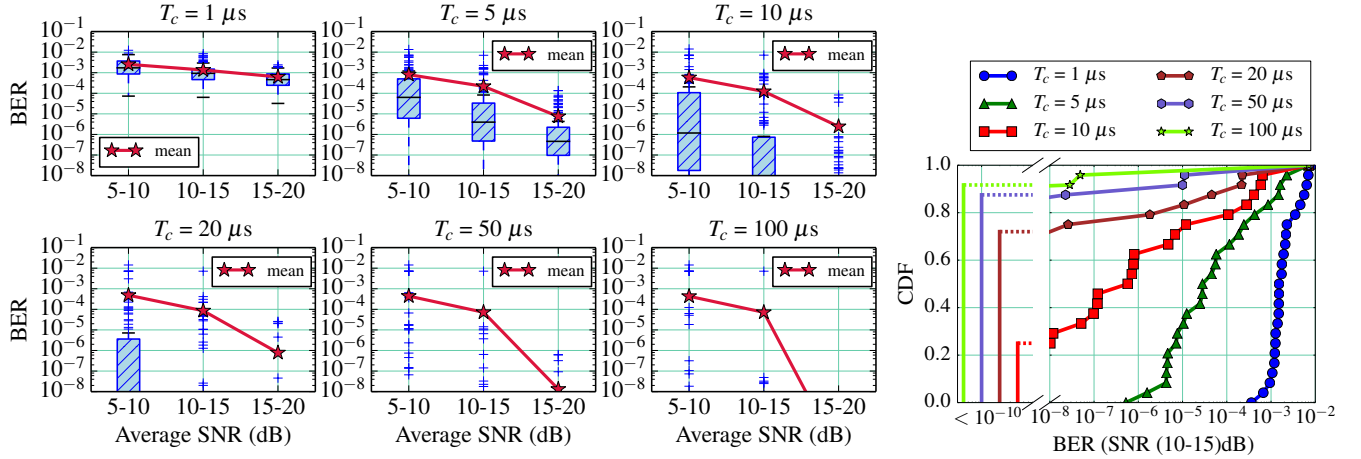
(a) BER of LDPC problem instances at different SNRs and QA compute times T_c in trace driven channels. The missing boxes in the figure are below 10^{-8} BER.

(b) CDF across individual LDPC problem instances in a trace driven channel.

Figure 12: Quantum Belief Propagation's overall experimental trace-driven channels' system performance. In Fig. 12(a), boxes' lower/upper whiskers and lower/upper quartiles represent $10^{th}/90^{th}$ and $25^{th}/75^{th}$ percentiles respectively.
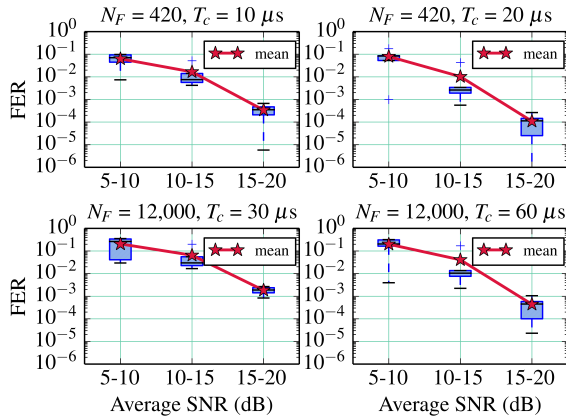

Figure 13: QBP's FER performance in trace driven channels. The unit of frame size N_F in the figure is bits. In the figure, boxes' lower/upper whiskers and lower/upper quartiles represent $10^{th}/90^{th}$ and $25^{th}/75^{th}$ percentiles repectively.

QA machines have been recently used to successfully optimize problems in several adjacent domains including Networks [39, 72], Machine Learning [1, 56], Scheduling [71], Fault Diagnosis [8, 58], and Chemistry [61]. Efficient embedding methods for mapping fully-connected QUBOs on to QA hardware graphs have also been discussed [13, 70] which support up to 64 variables on DW2Q QA.

9 LOOKING FORWARD

QA hardware trend predictions. For the past decade, the number of physical qubits in D-Wave's QPU has been steadily doubling each year and this trend is expected to continue [20]. Fig. 14 presents a predicted extrapolation of quantum annealer qubit and coupler counts into the future. The figure shows that at these rates, an

annealer processing chip with one million qubits could be available roughly by the year 2037. Let us envision future QAs with a processor topology that is either a Chimera or a supergraph of Chimera (*e.g.,* Pegasus [18]) with N_Q available qubits, which enables QBP to decode block lengths of at most $5N_Q/24$ bits in a single anneal. Thus in a QA with $N_Q = \{10^4, 10^5, 10^6\}$ qubits, we forecast QBP to be able to decode LDPC codes of block lengths up to {2,083, 20,833, 208,333} bits respectively in a single anneal with peak processing throughputs reaching {0.694, 6.94, 69.4} Gbps respectively, while most classical fully parallel decoders do not implement block lengths exceeding 2,048 bits due to signal routing and clock frequency constraints [26].

Limitations of QA. The lack of all-to-all qubit connectivity in today's QPUs limits the size of the problems the QA can practically solve, implying that the requirement of embedding is a major impediment to leveraging QA for practical applications. Furthermore, the process of transferring the computation and running on real analog QA device introduces a source of noise distinct from communication channel noise called *intrinsic control error* or *ICE*, which arises due to the flux noise and the quantization effects of the qubits. ICE effects in the QA alter both the problem biases ($h_i \rightarrow h_i \pm \delta h_i$) and couplers ($J_{ij} \rightarrow J_{ij} \pm \delta J_{ij}$), leading the QA to solve a slightly modified input problem in each anneal. Although the errors δh_i and δJ_{ij} are currently in the order of 10^{-2}, they may degrade the solution quality of some problems whose minimum energy state is not sufficiently separated from the other states in the energy landscape of the input problem Hamiltonian [19]. From a design perspective, ideally the qubits that are embedded together must all agree and end up in a similar final state at the end of the annealing process, otherwise the embedding chain is said to be *broken*: typically broken chains lead to performance degradation, and they are more likely to occur when the number of qubits embedded together in a particular chain is large (> 10). QBP's embedding design include chains of length two–four, and five–nine for Level I and Level II
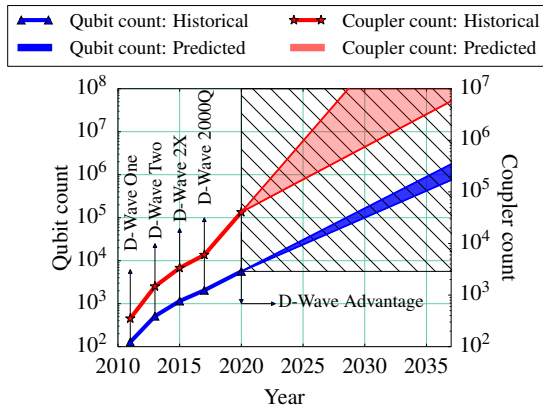
Figure 14: D-Wave QA's hardware resource counts over time. Historical data is in the years 2011–2020. The blue filled (darker) and the red filled (lighter) areas are the predicted qubit and coupler counts respectively, whose upper/lower boundaries are extrapolations of the most recent 2017–2020/2015–2017 qubit-coupler growths respectively. Annotations in the figure are the QA processor titles in the respective years.

Figure 15: *Top.* The plot shows the consumer price index (CPI) of classical computers and peripherals over time with 1997 as the base year. *Bottom.* The plot shows the predicted *price per hour* (PPH) of quantum annealers over time. The larger data point is the actual 2015–2020 QA price, which is conservatively assumed to remain the same until the QA technology matures in a predicted 17 years.

embeddings (§5.2) respectively. Further, there exist post-processing techniques such as *majority vote*, *weighted random*, and *minimize local energy* that can be used to improve the performance of broken chains [21]. QBP's embedding results in a low fraction of broken chains ($\approx 2\%$), and we use the majority voting technique in those cases to find the variables involved.

Cost considerations. QA technology is currently a cloud-based system and currently costs USD $2,000 for an hour of QPU access time, which is approximately $17.5M for an year. As the evolution of the technology is currently at an early stage (2011–), we consider the next 15 years for the technology to mature to the market. As usage becomes more widespread in future years, we hypothesize that QA prices will decrease with the same trend as classical compute prices have done since the late 20th century. Fig. 15 (top) shows the consumer price index (CPI) of classical computers and peripherals over time [45], while Fig. 15 (bottom) shows a similar predicted trend for QA *price per hour* (PPH). Figs. 14 and 15 imply that, at these rates QA technology is expected to deliver a machine with more than 10^6 qubits on a single annealer processing chip at the prices of $730, $235, $130, $82, and $68 per hour of QPU access time, by the years 2040, 2045, 2050, 2055, and 2059, respectively. This represents an approximate projected cost of $6.4M, $2M, $1.1M, $700K, and $600K per year, by the above respective years.

Timing considerations. Currently, the DW2Q has a 30–50 ms pre-processing time, 6–8 ms programming time, and 0.125 ms solution readout time per anneal, which are beyond the processing times available for wireless technologies (3-10 ms) [39], with supported annealing times in the range [1 μs, 2 ms]. Given the large amount of cost, embedding, and timing overheads of today's annealers, QBP currently cannot be deployed for use in practical applications. While approaches [7, 21] that decompose large-scale optimization problems can be used to study more problem variables, they suffer from requiring additional factors of the aforementioned machine
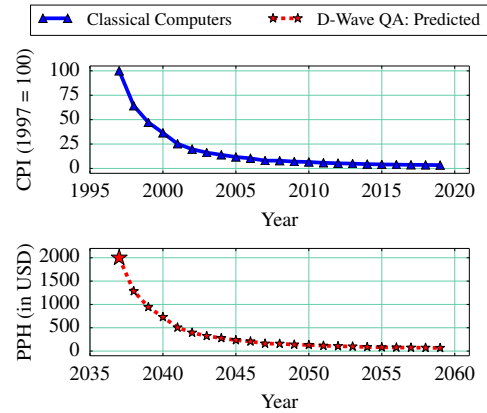
overhead times for each extra anneal. The historical trend is encouraging, with the DW2Q having a 5× annealing time improvement over the circa-2011 D-Wave One [10].

10 CONCLUSION AND FUTURE WORK

QBP is a novel QA-based uplink LDPC decoder that makes efficient use of the entire QA hardware to achieve new levels of performance beyond state-of-the-art BP decoders. Further efforts are needed to generalize QBP's graph embedding to large-scale LDPC codes with higher check bit degrees. The techniques we propose here may in the more distant future come to be relevant to practical protocol settings, while application of the aforementioned Cloud/Centralized-RAN architecture has also been proposed for small cells [64, 65]: opening the possibility to its future application to managed Wi-Fi local-area networks. Investigating the QA technology for problems such as network security, downlink precoding, scheduling, and other uplink channel codes such as Polar and Turbo codes is potential future work direction.

REFERENCES

[1] Steven H Adachi and Maxwell P Henderson. 2015. Application of quantum annealing to training of deep neural networks.

[2] Alexandru Amaricai and Oana Boncalo. 2017. Design Trade–Offs for FPGA Implementation of LDPC Decoders. In *Field*, George Dekoulis (Ed.). IntechOpen, Rijeka, Chapter 5, 105. https://doi.org/10.5772/66085

[3] Mohammad H Amin. 2015. Searching for quantum speedup in quasistatic quantum annealers. *Physical Review A* 92, 5 (2015), 052323.

[4] Maliheh Aramon, Gili Rosenberg, Elisabetta Valiante, Toshiyuki Miyazawa, Hirotaka Tamura, and Helmut G Katzgraber. 2019. Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. *Frontiers in Physics* 7 (2019), 48.

[5] Juan Miguel Arrazola, Alain Delgado, Bhaskar Roy Bardhan, and Seth Lloyd. 2019. Quantum-inspired algorithms in practice. arXiv:arXiv:1905.10415

[6] C. Berrou, A. Glavieux, and P. Thitimajshima. 1993. Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Proceedings of ICC '93 - IEEE International Conference on Communications*, Vol. 2. IEEE, Geneva, Switzerland, 1064–1070.

[7] Zhengbing Bian, Fabian Chudak, Robert Israel, Brad Lackey, William G Macready, and Aidan Roy. 2014. Discrete optimization using quantum annealing on sparse Ising models. *Frontiers in Physics* 2 (2014), 56.

[8] Zhengbing Bian, Fabian Chudak, Robert Brian Israel, Brad Lackey, William G Macready, and Aidan Roy. 2016. Mapping constrained optimization problems to quantum annealing with application to fault diagnosis. *Frontiers in ICT* 3 (2016), 14.

[9] Zhengbing Bian, Fabian Chudak, William G Macready, and Geordie Rose. 2010. The Ising model: teaching an old problem new tricks.

[10] Sergio Boixo, Troels F Rønnow, Sergei V Isakov, Zhihui Wang, David Wecker, Daniel A Lidar, John M Martinis, and Matthias Troyer. 2014. Evidence for quantum annealing with more than one hundred qubits. *Nature physics* 10, 3 (2014), 218–224.

[11] CCSDS Blue Book. 2020. Radio Frequency and Modulation Systems–Part 1 Earth Stations and Spacecraft.

[12] CCSDS Orange Book. 2014. Erasure Correcting Codes for Use in Near-Earth and Deep-Space Communications.

[13] Tomas Boothby, Andrew D King, and Aidan Roy. 2016. Fast clique minor generation in Chimera qubit connectivity graphs. *Quantum Information Processing* 15, 1 (2016), 495–508.

[14] Jun Cai, William G Macready, and Aidan Roy. 2014. A practical heuristic for finding graph minors.

[15] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann. 2015. Cloud RAN for Mobile Networks—A Technology Overview. *IEEE Communications Surveys Tutorials* 17, 1 (2015), 405–426. https://doi.org/10.1109/COMST.2014.2355255

[16] Shashi Kiran Chilappagari, Dung Viet Nguyen, Bane Vasic, and Michael W Marcellin. 2008. Girth of the Tanner graph and error correction capability of LDPC codes. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*. IEEE, IL, USA, 1238–1245.

[17] D-Wave Hybrid Solver Service. Website.

[18] D-Wave Next-Generation QPU Topology. Website.

[19] D-Wave Quantum Processing Unit. Website.

[20] D-Wave Systems Technology Information. Website.

[21] D-Wave *qbsolv* embedding tool. Website.

[22] D-Wave Virtual Full-Yield Chimera Solver. Website.

[23] ETSI. 2009. ETSI Standard EN 302 307: Digital Video Broadcasting; Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2).

[24] TS ETSI. 2018. 138 212 V15. 2.0 Technical Specification–5G, NR, Multiplexing and channel coding.

[25] Robert Gallager. 1962. Low-density parity-check codes. *IRE Transactions on Information Theory* 8, 1 (1962), 21–28.

[26] Peter Hailes, Lei Xu, Robert G Maunder, Bashir M Al-Hashimi, and Lajos Hanzo. 2016. A survey of FPGA-based LDPC decoders. *IEEE Communications Surveys & Tutorials* 18, 2 (2016), 1098–1122.

[27] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. 2011. Tool release: Gathering 802.11 n traces with channel state information. *ACM SIGCOMM Computer Communication Review* 41, 1 (2011), 53–53.

[28] Kuk-Hyun Han and Jong-Hwan Kim. 2002. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE transactions on evolutionary computation* 6, 6 (2002), 580–593.

[29] Dale E Hocevar. 2004. A reduced complexity decoder architecture via layered decoding of LDPC codes. In *IEEE Workshop on Signal Processing Systems*. IEEE, TX, USA, 107–112.

[30] IEEE. 2012. IEEE Standard 802.11: Wireless LAN Medium Access and Physical Layer Specifications.

[31] IEEE. 2012. IEEE Standard 802.16: Air Interface for Broadband Wireless Access Systems.

[32] IEEE. 2012. IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. https://doi.org/10.1109/IEEESTD.2012.6178212

[33] Hirotaka Irie, Haozhao Liang, Shinya Gongyo, Tetsuo Hatsuda, et al. 2020. Hybrid Quantum Annealing via Molecular Dynamics.

[34] Hiroshi Ishikawa. 2009. Higher-order clique reduction in binary graph cut. In *IEEE CVPR*. IEEE, FL, USA, 2993–3000.

[35] Mark W Johnson, Mohammad HS Amin, Suzanne Gildert, Trevor Lanting, Firas Hamze, Neil Dickson, R Harris, Andrew J Berkley, Jan Johansson, Paul Bunyk, et al. 2011. Quantum annealing with manufactured spins. *Nature* 473, 7346 (2011), 194.

[36] Tadashi Kadowaki and Hidetoshi Nishimori. 1998. Quantum annealing in the transverse Ising model. *Physical Review E* 58, 5 (1998), 5355.

[37] Ryan Kastner, Janarbek Matai, and Stephen Neuendorffer. 2018. Parallel Programming for FPGAs. arXiv:1805.03648 [cs.AR]

[38] Helmut G Katzgraber, Simon Trebst, David A Huse, and Matthias Troyer. 2006. Feedback-optimized parallel tempering Monte Carlo. *Journal of Statistical Mechanics: Theory and Experiment* 2006, 03 (2006), P03018.

[39] Minsung Kim, Davide Venturelli, and Kyle Jamieson. 2019. Leveraging Quantum Annealing for Large MIMO Processing in Centralized Radio Access Networks. In *Proceedings of the ACM Special Interest Group on Data Communication* (Beijing, China) *(SIGCOMM '19)*. Association for Computing Machinery, New York, NY, USA, 241–255. https://doi.org/10.1145/3341302.3342072

[40] Andrew D King, Juan Carrasquilla, Jack Raymond, Isil Ozfidan, Evgeny Andriyash, Andrew Berkley, Mauricio Reis, Trevor Lanting, Richard Harris, Fabio Altomare, et al. 2018. Observation of topological phenomena in a programmable lattice of 1,800 qubits. *Nature* 560, 7719 (2018), 456.

[41] John FC Kingman. 1975. Random discrete distributions. *Journal of the Royal Statistical Society: Series B (Methodological)* 37, 1 (1975), 1–15.

[42] P. J. M. Laarhoven and E. H. L. Aarts. 1987. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, USA.

[43] Brad Lackey. 2018. A belief propagation algorithm based on domain decomposition.

[44] Y. Lin, L. Shao, Z. Zhu, Q. Wang, and R. K. Sabhikhi. 2010. Wireless network cloud: Architecture and system requirements. *IBM Journal of Research and Development* 54, 1 (2010), 4:1–4:12.

[45] Long Term Price Trends of Computers and Peripherals: U.S. Bureau of Labor Statistics. Website.

[46] Jin Lu and Josée MF Moura. 2006. Structured LDPC codes for high-density recording: large girth and low error floor. *IEEE transactions on magnetics* 42, 2 (2006), 208–213.

[47] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014), 5. https://doi.org/10.3389/fphy.2014.00005

[48] David JC MacKay. 1999. Good error-correcting codes based on very sparse matrices. *IEEE Trans. on Information Theory* 45, 2 (1999), 399–431.

[49] Grigorii A Margulis. 1982. Explicit constructions of graphs without short cycles and low density codes. *Combinatorica* 2, 1 (1982), 71–78.

[50] S. Matsubara, M. Takatsu, T. Miyazawa, T. Shibasaki, Y. Watanabe, K. Takemoto, and H. Tamura. 2020. Digital Annealer for High-Speed Solving of Combinatorial optimization Problems and Its Applications. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, Beijing, China, 667–672.

[51] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. 2016. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics* 18, 2 (2016), 023023.

[52] C C McGeoch. 2014. Adiabatic quantum computation and quantum annealing: Theory and practice. *Synthesis Lectures on Quantum Computing* 5, 2 (2014), 1–93.

[53] Catherine C. McGeoch and Cong Wang. 2013. Experimental Evaluation of an Adiabatic Quantum System for Combinatorial Optimization. In *Proceedings of the ACM International Conference on Computing Frontiers* (Ischia, Italy) *(CF '13)*. Association for Computing Machinery, New York, NY, USA, Article 23, 11 pages. https://doi.org/10.1145/2482767.2482797

[54] Oscar Montiel, Yoshio Rubio, Cynthia Olvera, and Ajelet Rivera. 2019. Quantum-Inspired Acromyrmex Evolutionary Algorithm. *Scientific reports* 9, 1 (2019), 1–10.

[55] Alberto Morello and Vittoria Mignone. 2006. DVB-S2: The second generation standard for satellite broad-band services. *Proc. of the IEEE* 94, 1 (2006), 210–227.

[56] Alex Mott, Joshua Job, Jean-Roch Vlimant, Daniel Lidar, and Maria Spiropulu. 2017. Solving a Higgs optimization problem with quantum annealing for machine learning. *Nature* 550, 7676 (2017), 375–379.

[57] A. Orlitsky, R. Urbanke, K. Viswanathan, and J. Zhang. 2002. Stopping sets and the girth of Tanner graphs. In *Proceedings IEEE International Symposium on Information Theory,*. IEEE, Lausanne, Switzerland, 2.

[58] Alejandro Perdomo-Ortiz, Joseph Fluegemann, Sriram Narasimhan, Rupak Biswas, and Vadim N Smelyanskiy. 2015. A quantum annealing approach for

fault detection and diagnosis of graph-based systems. *The European Physical Journal Special Topics* 224, 1 (2015), 131–148.

[59] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (2018), 79.

[60] Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell Systems Technical Journal* 27, 3 (1948), 379–423.

[61] Michael Streif, Florian Neukart, and Martin Leib. 2019. Solving Quantum Chemistry Problems with a D-Wave Quantum Annealer. In *Quantum Technology and Optimization Problems*, Sebastian Feld and Claudia Linnhoff-Popien (Eds.). Springer International Publishing, Cham, 111–122.

[62] Y. Sun, M. Karkooti, and J. R. Cavallaro. 2006. High Throughput, Parallel, Scalable LDPC Encoder/Decoder Architecture for OFDM Systems. In *2006 IEEE Dallas/CAS Workshop on Design, Applications, Integration and Software.* IEEE, TX, USA, 39–42.

[63] Karthikeyan Sundaresan. 2013. Cloud-Driven Architectures for next Generation Small Cell Networks. In *Proceedings of the Eighth ACM International Workshop on Mobility in the Evolving Internet Architecture* (Miami, Florida, USA) *(MobiArch '13).* Association for Computing Machinery, New York, NY, USA, 3–4. https://doi.org/10.1145/2505906.2511048

[64] Karthikeyan Sundaresan, Mustafa Y. Arslan, Shailendra Singh, Sampath Rangarajan, and Srikanth V. Krishnamurthy. 2016. FluidNet: A Flexible Cloud-based Radio Access Network for Small Cells. *IEEE/ACM Trans. on Networking* 24, 2 (April 2016), 915–928. https://doi.org/10.1109/TNET.2015.2419979

[65] Karthikeyan Sundaresan, Mustafa Y. Arslan, Shailendra Singh, Sampath Rangarajan, and Srikanth V. Krishnamurthy. 2016. FluidNet: A Flexible Cloud-Based Radio Access Network for Small Cells. *IEEE/ACM Trans. Netw.* 24, 2 (April 2016), 915–928.

[66] Ryan Sweke, Frederik Wilde, Johannes Meyer, Maria Schuld, Paul K. Fahrmann, Barthelemy Meynard-Piganeau, and Jens Eisert. 2019. Stochastic gradient descent for hybrid quantum-classical optimization. arXiv:1910.01155 [quant-ph]

[67] R Tanner. 1981. A recursive approach to low complexity codes. *IEEE Transactions on information theory* 27, 5 (1981), 533–547.

[68] J. Teubner, R. Mueller, and G. Alonso. 2010. FPGA acceleration for the frequent item problem. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010).* IEEE, CA, USA, 669–680.

[69] Tony T Tran, Minh Do, Eleanor G Rieffel, Jeremy Frank, Zhihui Wang, Bryan O'Gorman, Davide Venturelli, and J Christopher Beck. 2016. A hybrid quantum-classical approach to solving scheduling problems. In *Ninth annual symposium on combinatorial search.* AAAI, NY, USA, 98–106.

[70] Davide Venturelli, Salvatore Mandrà, Sergey Knysh, Bryan O'Gorman, Rupak Biswas, and Vadim Smelyanskiy. 2015. Quantum Optimization of Fully Connected Spin Glasses. *Phys. Rev. X* 5 (Sep 2015), 031040. Issue 3. https://doi.org/10.1103/PhysRevX.5.031040

[71] Davide Venturelli, Dominic J. J. Marchand, and Galo Rojo. 2015. Quantum Annealing Implementation of Job-Shop Scheduling. arXiv:1506.08479 [quant-ph]

[72] Chi Wang, Huo Chen, and Edmond Jonckheere. 2016. Quantum versus simulated annealing in wireless interference network optimization. *Scientific reports* 6 (2016), 25797.

[73] Xilinx UltraScale Architecture User Guide. Website.

[74] Xilinx Vivado Design Suite User Guide. Website.

[75] Raman Yazdani and Masoud Ardakani. 2011. Efficient LLR calculation for non-binary modulations over fading channels. *IEEE transactions on communications* 59, 5 (2011), 1236–1241.

[76] Jianguang Zhao, Farhad Zarkeshvari, and Amir H Banihashemi. 2005. On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes. *IEEE transactions on communications* 53, 4 (2005), 549–554.

[77] Victor Vasilievich Zyablov and Mark Semenovich Pinsker. 1975. Estimation of the error-correction complexity for Gallager low-density codes. *Problemy Peredachi Informatsii* 11, 1 (1975), 23–36.