Shortest Codeword Problem driven by DQI

Rushil Shah*

Department of Physics, Massachusetts Institute of Technology

(Dated: May 22, 2026)

Abstract

Quantum optimization algorithms attempt to prove quantum supremacy over classical computation. Many of these algorithms solve certain instances of a general class of problems called constraint satisfication problems (CSPs). These problems are fundamental, so demonstrating an exponential speedup is necessary for the viability of quantum computation. Recently, a paper proposed an algorithm called DQI [5] which prepares a state that generates a polynomial of a CSP objective function. They provide an application to the max-XORSAT problem where they notice speedups in special cases. In this paper, we introduce the shortest codeword problem and demonstrate an equality with the max-XORSAT problem. We also take a look at some key features of the DQI algorithm which provide its functionality, specifically the usage of elementary symmetric polynomials. Using these features, we apply the DQI algorithm to the spiked shortest codeword problem [1] and analyze the results with a simulated annealing approach. We notice that DQI performs slightly worse than the classical simulated annealing algorithm on 10 random 6 by 12 generator matrices, likely a result of a smaller polynomial degree and suboptimal coefficient selection. Finally, we take a look at the limits of the DQI algorithm and how it can be applied to similar problems.

* rshah2@mit.edu

I. INTRODUCTION

One of the best ways to quantify quantum supremacy over classical computing is through asymptotic improvements in runtime for optimization problems. Throughout the development of the quantum computing field, there have been many attempts to identify exponential speedups in approximate optimization problems. Recently, there has been a new approach on quantum optimization called Decoded Quantum Interferometry, or DQI [5]. Unlike previous approaches, DQI utilizes the quantum fourier transform to reduce the optimization problem to a decoding problem. This procedure has yielded exponential speedup for problems with special structures of the parity check matrix of the decoding problem, most notably, the optimal polynomial intersection problem [4]. However, it was found that for other optimization problems using a less optimal decoder, DQI performs worse compared to other quantum algorithms like QAOA over 14 rounds for the max-3-XORSAT problem [5].

In this paper, we apply DQI to the shortest codeword problem. We construct a solution by modfying the results of the max-XORSAT problem and applying them to the shortest codeword problem. Using a classical simulation, we test the DQI algorithm on an example code space to understand its performance. We then propose a modification to the shortest codeword problem and attempt to generate a DQI state using the elementary symmetric polynomials mentioned in the DQI paper [5]. Comparing classical simulations of DQI on the modified codeword problem with a classical simulated annealing algorithm, we highlight the downsides of the DQI algorithm. Lastly, we provide some future directions and applications for the shortest codeword problem on DQI.

II. OVERVIEW OF DQI ON MAX-XORSAT

The max-XORSAT problem states,

Definition II.1. *Given a set of constraints $B \in \mathbb{F}_2^{m\times n}$, where $m > n$, and desired outcomes $v \in \mathbb{F}_2^m$, we want to find a $\boldsymbol{x} \in \mathbb{F}_2^n$ such that $B\boldsymbol{x} = v$. Since the system is overdetermined, we want to find an $\boldsymbol{x}$ that maximizes the number of satisfied $b_i \cdot \boldsymbol{x} = v_i$.*

We can write the objective function as,

$$f(x) = \sum_i (-1)^{b_i \cdot \mathbf{x} + v_i}, \tag{1}$$

where the intuition is that if $b_i \cdot \mathbf{x} = v_i$, the exponent is 0 or 2. So, when $b_i \cdot \mathbf{x} = v_i$, the objective function yields a 1, otherwise, it yields a -1. Therefore, the objective function maximizes the number of instances where $b_i \cdot \mathbf{x} = v_i$.

The DQI algorithm for the max-XORSAT problem can be found in Figure 3 of [5]. However, before moving onto the shortest codeword problem, we will go over some insights from the paper that will aid calculations later on.

A. Elementary Symmetric Polynomials

The main insight of DQI is that we are able to generate a state that raises the objective function to some power,

$$|P(f)\rangle = \sum_{k=0}^{\ell} u_k \sum_{x \in \mathbb{F}_2^n} P^{(k)} |x\rangle = \sum_{k=0}^{\ell} w_k |P^{(k)}\rangle, \tag{2}$$

where,

$$|P^{(k)}\rangle = H^{\otimes n} |\tilde{P}^{(k)}\rangle = H^{\otimes n} \frac{1}{\sqrt{\binom{m}{k}}} \sum_{\substack{\mathbf{y} \in \mathbb{F}_2^m \\ |\mathbf{y}| = k}} (-1)^{\mathbf{v} \cdot \mathbf{y}} |B^T \mathbf{y}\rangle. \tag{3}$$

as defined in (17) and (24) of [5]. To show that these equations are valid, we can follow the steps of (16) to (26) in [5] as they will be relevant in solving arbitrary constraint satisfication problems. As a quick review, we can write $P(f)$ as an ℓ-th degree polynomial,

$$P(f) = \sum_{k=0}^{\ell} \alpha_k f^k = \sum_{k=0}^{\ell} \alpha_k (f_1 + f_2 + \ldots + f_m)^k \tag{4}$$

$$= \sum_{k=0}^{\ell} \alpha_k \sum_{\substack{b \in \mathbb{F}_k^m \\ \sum b = k}} f_1^{b_1} f_2^{b_2} \ldots f_m^{b_m} = \sum_{k=0}^{\ell} u_k \sum_{|b| = k} f_1^{b_1} f_2^{b_2} \ldots f_m^{b_m} \tag{5}$$

$$= \sum_{k=0}^{\ell} u_k P^{(k)}(f_1, \ldots, f_m) \tag{6}$$

We perform the last simplification since $f = \pm 1$, $f^2 = 1$, so we can reduce all even values of b_i above to 1 and all odd values of b_i to 1. So, we can represent b as a bitstring where the hamming weight of b represents the number of odd powers it had originally.

B. Bounded Distance Decoding

A major limitation of DQI is the necessity of decoding. Decoding for DQI is necessary to uncompute the $|\mathbf{y}\rangle$ registers. To generate the $|B^T\mathbf{y}\rangle$ registers, the DQI algorithm produces the $|\mathbf{y}\rangle$ and performs matrix multiplication, creating entanglement between the registers. However, (3) requires only the $|B^T\mathbf{y}\rangle$ register, so uncomputation must occur to make the y registers all $|0\rangle$ states.

Performing this uncomputation is analogous to syndrome decoding where $|\mathbf{y}\rangle$ are the errors and $|B^T\mathbf{y}\rangle$ are the syndromes. However, decoding is difficult for arbitrary matrix patterns. Therefore, in order to make DQI efficient, determining an optimal decoder for a particular pattern is necessary. For the case of max-XORSAT, an LDPC decoder is used for a sparse matrix B.

III. SHORTEST CODEWORD PROBLEM

Definition III.1 (Shortest Codeword Problem)**.** *Suppose we have a code generator matrix $C \in \mathbb{F}_2^{m\times n}$. Each row of the matrix represents a codeword of length n. Determine a linear combination of these codewords, rows of C, with the smallest hamming weight.*

Representing the code space mathematically,

$$S_C = \{C^T\mathbf{x} | \mathbf{x} \in \mathbb{F}_2^m\}.$$

In order to minimize the hamming weight, we will sum $(-1)^{b_i}$ where $b_i = (C^T\mathbf{x})_i$ is the i-th bit of some linear combination bitstring, so we want to maximize the objective function,

$$f(x) = \sum_{i=1}^{n} (-1)^{C_i^T\cdot\mathbf{x}}.$$

This is analogous to the max-XORSAT objective function in (1) where $v = 0$. Therefore,

setting $v = 0$ from lines 19 to 23 in [5], we get the following polynomials and DQI states,

$$P^{(k)} = \sum_{\substack{|\mathbf{y}|=k \\ \mathbf{y}\in\mathbb{F}_2^n}} (-1)^{(C\mathbf{y})\cdot\mathbf{x}}, \tag{7}$$

$$|\tilde{P}^{(k)}\rangle = \frac{1}{\sqrt{\binom{n}{k}}} \sum_{\substack{|\mathbf{y}|=k \\ \mathbf{y}\in\mathbb{F}_2^n}} |Cy\rangle, \tag{8}$$

$$|P(x)\rangle = \sum_{k=0}^{\ell} w_k |P^{(k)}\rangle = \sum_{k=0}^{\ell} w_k H^{\otimes m} |\tilde{P}^{(k)}\rangle. \tag{9}$$

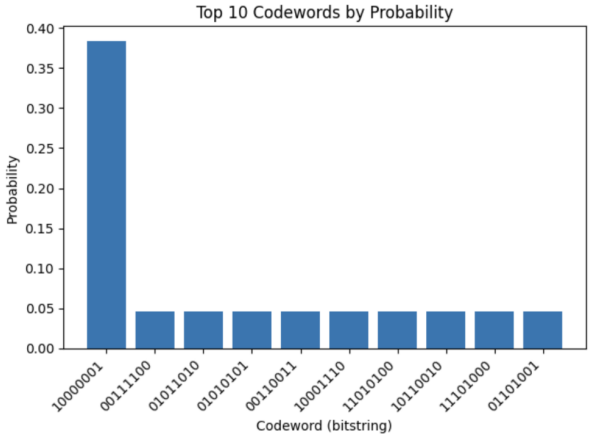
We aren't quite done yet because this setup allows for $\mathbf{x} = 0$ which is a trivial linear combination. Therefore, we must also set the amplitude of the $\mathbf{x} = 0$ state to be 0.

Let's see how DQI performs on the shortest codeword problem. Consider the code,

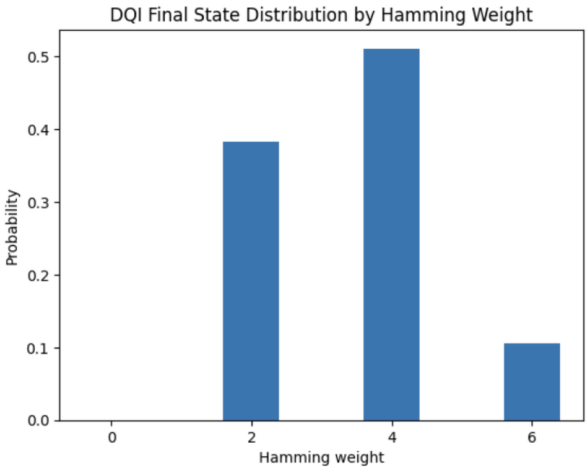
$$C = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

The shortest codeword, 10000001, occurs as a linear combination of all codewords in C. We perform this simulation classically as to avoid any decoding by splicing the bitstring on the syndrome registers. However, the Dicke state preparation is still performed quantum mechanically and then the amplitude vectors are extracted at which point decoding and the hadamard transform are conducted classically.

As shown in Figure 1a, the codeword measured with the highest probability is the shortest codeword. Furthermore, the codewords with hamming weight of 4 are all measured with an equal and significantly smaller probability. This matches up with the expected output. In Figure 1b, we see a distribution of the hamming weights of the codewords for the same code that seems unexpected. However, the peak for the hamming weight 4 codewords is a result of the large number of hamming weight 4 codewords compared to the singular hamming weight 2 codeword which is resolved if we were to increase the degree of the polynomial.



(a) Probability distribution of 10 most likely codewords



(b) Probability distribution of hamming weights

FIG. 1: DQI on Shortest Codeword Problem

IV. SPIKED SHORTEST CODEWORD PROBLEM

A. Setup

We will now tweak the shortest codeword problem by introducing a spike according to the function of hamming weight,

$$h(w) = \begin{cases} w, & w \neq n/4, \\ n, & w = n/4. \end{cases}$$

From [2], the simulated annealing approach tends to get stuck at the local minimum. So, we want to see if DQI is able to identify the minimum weight codewords despite the spike.

Using the idea of elementary symmetric polynomials introduced in [5], we want to design an objective function such that $f_i^2 = 1$ for all i. The easiest way would be to use a sum of $(-1)^n$ where n is some non negative integer. To encode the fact that $h(w) = n$ for $w = n/4$, we can include an extra $3n/4$ objective functions of the form $(-1)^0 = 1$ except

when $C^T x = n/4$ for which $(-1)^1 = -1$. So, formally, assuming C is an $m \times n$ matrix,

$$f_i(x) = \begin{cases} (-1)^{C_i^T \cdot \mathbf{x}}, & 1 \leq i \leq n, \\ (-1)^0, & n < i \leq 7n/4, |C^T x| \neq n/4, \\ (-1)^1, & n < i \leq 7n/4, |C^T x| = n/4. \end{cases}$$

Using this objective function, we can construct the symmetric polynomials. To simplify the expressions, we will denote α as the conditional exponent on f_i for $n < i \leq 7n/4$,

$$\begin{aligned} P^{(k)}_{\text{spiked}} &= \sum_{i_1 \ldots i_k} f_{i_1} \times \ldots \times f_{i_k} \\ &= P^{(k)} + (-1)^\alpha \sum_{1 \leq i_1 \ldots i_{k-1} \leq m} (-1)^{(C^T_{i_1} + \ldots + C^T_{i_{k-1}}) \cdot \mathbf{x}} + (-1)^{2\alpha} \sum_{1 \leq i_1 \ldots i_{k-2} \leq m} (-1)^{(C^T_{i_1} + \ldots + C^T_{i_{k-2}}) \cdot \mathbf{x}} + \ldots, \\ &= P^{(k)} + (-1)^\alpha \sum_{\substack{|y| = k-1, \\ y \in \mathbb{F}_2^n}} (-1)^{(Cy) \cdot \mathbf{x}} + (-1)^{2\alpha} \sum_{\substack{|y| = k-2, \\ y \in \mathbb{F}_2^n}} (-1)^{(Cy) \cdot \mathbf{x}} + \ldots, \end{aligned}$$

where $P^{(k)}$ is as defined in (7). To provide some insight into this equation, the first term represents the case where all the objective functions are the ones from the unperturbed shortest code word problem. Each term after represents an additional objective function for the spike. Looking closer at the summations, we notice that they are the DQI states of lower degrees. Therefore, we can rewrite the polynomial as the summation,

$$P^{(k)}_{\text{spiked}} = \sum_{i=0}^{k} (-1)^{(k-i)\alpha} P^{(i)}.$$

Therefore, our desired DQI state is,

$$|P^{(k)}_{\text{spiked}}\rangle = \sum_{x \in \mathbb{F}_2^m} \sum_{i=0}^{k} \binom{3n/4}{k-i} (-1)^{(k-i)\alpha} P^{(i)} |x\rangle,$$

where we ignore an overall normalization constant. Note that the binomial coefficient comes from the number of ways to choose $k - i$ of the $3n/4$ extra spiked objective functions. To simplify writing out the expressions, let's define the spike region as,

$$S = \{x : |C^T x| = n/4, x \in \mathbb{F}_2^m\}.$$

Breaking up the DQI state into the spiked region and unspiked region,

$$|P^{(k)}_{\text{spiked}}\rangle = \sum_{x\in S}\sum_{i=0}^{k}\binom{3n/4}{k-i}(-1)^{(k-i)}P^{(i)}|x\rangle + \sum_{x\notin S}\sum_{i=0}^{k}\binom{3n/4}{k-i}P^{(i)}|x\rangle,$$

$$|P^{(k)}_{\text{spiked}}\rangle = \sum_{x\in S}\sum_{i=0}^{k}\binom{3n/4}{k-i}(-1)^{(k-i)}P^{(i)}|x\rangle + \left(\sum_{x\in\mathbb{F}_2^m}\sum_{i=0}^{k}\binom{3n/4}{k-i}P^{(i)}|x\rangle - \sum_{x\in S}\sum_{i=0}^{k}\binom{3n/4}{k-i}P^{(i)}|x\rangle\right),$$

$$= \sum_{i=0}^{k}\binom{3n/4}{k-i}\sum_{x\in S}\left((-1)^{(k-i)}-1\right)P^{(i)}|x\rangle + \sum_{i=0}^{k}\binom{3n/4}{k-i}\sqrt{2^m\binom{n}{i}}|P^{(i)}\rangle, \tag{10}$$

where $|P^{(i)}\rangle$ is given by (9). We can simplify this a bit more since for all $x \in S$, $P^{(i)}$ is generated by the same value of the objective function, n. Therefore, we can define,

$$|P_S\rangle = \frac{1}{\sqrt{|S|}}\sum_{x\in S}|x\rangle.$$

In addition, $P^{(i)}$ is the value of the polynomial when the codeword has a hamming weight of $n/4$. So, plugging this definition into (10), our DQI state on the spiked potential will be,

$$|P^{(k)}_{\text{spiked}}\rangle = \sum_{i=0}^{k}\binom{3n/4}{k-i}\sqrt{2^m\binom{n}{i}}|P^{(i)}\rangle - 2\sum_{\substack{i=0,\\ k-i\text{ odd}}}^{k}\binom{3n/4}{k-i}\sqrt{|S|}P_S^{(i)}|P_S\rangle,$$

$$|P_{\text{spiked}}\rangle = \sum_{k=0}^{\ell}w_k\left(\sum_{i=0}^{k}\binom{3n/4}{k-i}\sqrt{2^m\binom{n}{i}}|P^{(i)}\rangle - 2\sum_{\substack{i=0,\\ k-i\text{ odd}}}^{k}\binom{3n/4}{k-i}\sqrt{|S|}P_S^{(i)}|P_S\rangle\right), \tag{11}$$

$$= \sum_{k=0}^{\ell}\sqrt{\binom{n}{k}}\left(\sum_{i=k}^{\ell}w_i\binom{3n/4}{i-k}\right)\sqrt{2^m}|P^{(k)}\rangle - 2\sum_{k=0}^{\ell}\left(\sum_{\substack{i=k,\\ i-k\text{ odd}}}^{\ell}w_i\binom{3n/4}{i-k}\right)\sqrt{|S|}P_S^{(i)}|P_S\rangle,$$

(12)

$$= \sum_{k=0}^{\ell}v_k|P^{(k)}\rangle - 2\sqrt{|S|}\sum_{k=0}^{\ell}\tilde{v}_k|P_S\rangle, \tag{13}$$

where

$$v_k = \sqrt{2^m}\sqrt{\binom{n}{k}}\sum_{i=k}^{\ell}\sqrt{\binom{3n/4}{i-k}}w_i,\ \tilde{v}_k = P_S^{(k)}\sum_{\substack{i=k,\\ i-k\text{ odd}}}^{\ell}\binom{3n/4}{i-k}w_i,$$

and $P_S^{(k)}$ is the value of the k-th degree symmetric polynomial evaluated at the spiked hamming weight. Note that we flipped the summations to get from (11) to (12). With the right choice of w_k, the new DQI state should make some physical sense because we can

think of it as preparing a regular DQI state, including all of the code word linear combinations, and subtracting some amplitude from the states which represent code words with a hamming weight of $n/4$.

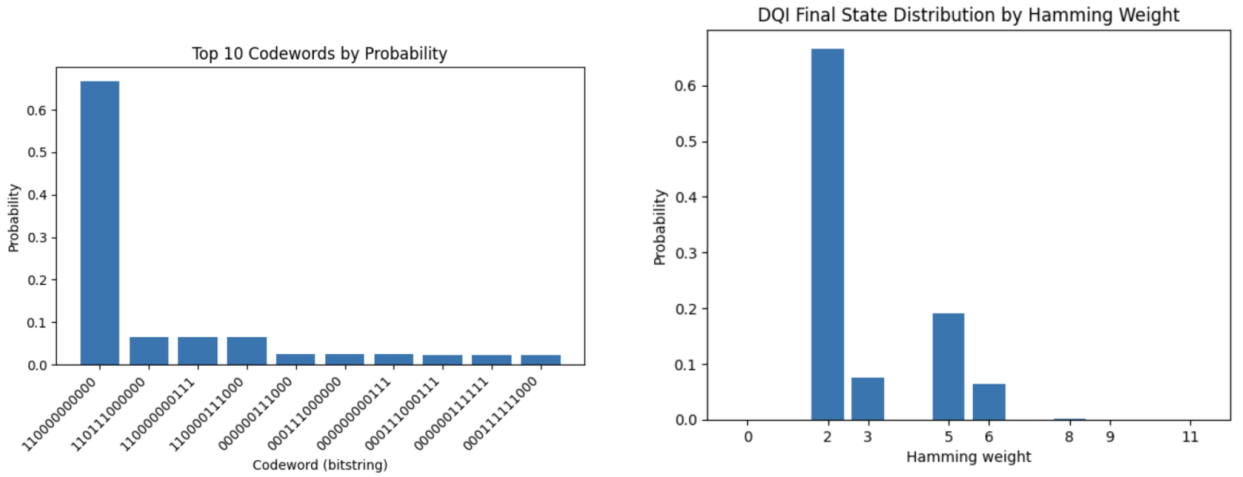
Preparing this state is simple classically since we can manipulate the amplitude vector using conditionals. To generate this state in a quantum circuit, $|P^{(k)}\rangle$ can be performed by the procedure listed in the DQI paper [5]. To prepare the $|P_S^{(k)}\rangle$ state, we can use an approach similar to Grover's algorithm. We can create a superposition over all possible bitstrings of length m. Then, using an oracle that tells us whether a state creates a codeword of hamming weight $n/4$, we flip the sign of all states $|x\rangle$ if $|C^T x| = n/4$. Finally, we can perform our amplification step by applying the Hadamard, phase flipping $|0\rangle^{\otimes m}$, and Hadamard again. Applying the phase flipping and the amplification enough will create a uniform amplitude distribution for all x bitstrings satisfying the oracle and minimizing the amplitudes of the bitstrings not satisfying the oracle.

B. Results

Consider the example code generated by,

$$C = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Just by looking at the code, the codeword with the smallest hamming weight is the first row in the generator matrix. This is because no codewords have overlapping 1s, so creating any linear combination will create a new codeword with more 1s. However, because of the spiked potential, we expect that the state with hamming weight of $12/4 = 3$ will have a lower probability of appearing. Therefore, when performing optimization on this code, we would prefer a state with a hamming weight of 5 (the first word with any other word) over a state with a hamming weight of 3 (all the other words). Performing a classical simulation of DQI on this sample code, we notice, as depicted in Figure 2b, that we observe a large peak for hamming weight 2 codewords and a significantly smaller peak at hamming weight 5 codewords, implying that DQI is able to suppress the probabilities of

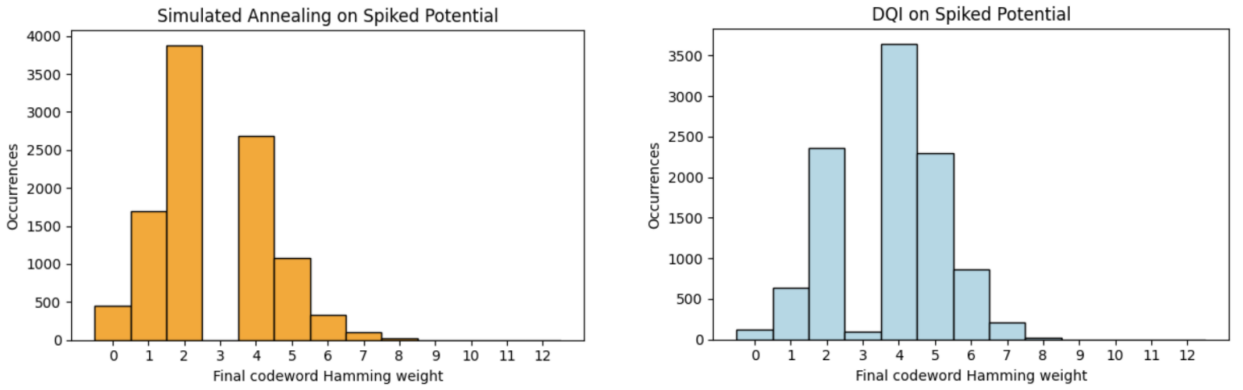


(a) Probability distribution of 10 most likely codewords

(b) Probability distribution of hamming weights

FIG. 2: DQI on the Spiked Shortest Codeword Problem

hamming weight 3 codewords. Similarly, in Figure 2a, we see that the probability of detecting the codeword of hamming weight 2 is significantly higher than other codewords, implying that DQI is able to find a local minima.



(a) Hamming weight SA distribution

(b) Hamming weights DQI distribution

FIG. 3: Spiked Shortest Codeword Problem w/ 10 random 6x12 Matrices

To properly analyze the effectiveness of DQI for the spiked shortest codeword problem, we compare its performance to a classical simulated annealing approach following the metropolis acceptance criteria. According to [2], a simulated annealing algorithm would get stuck at the local minimum at hamming weights of $n/4+1$ after a polynomial

number of steps since the spike would be too large to overcome for any singular bitflip. As shown in Figure 3, DQI and SA both appear to get stuck at the local minima of hamming weight 4 codewords. In fact, simulated annealing seems to perform slightly better than DQI.

However, this result is likely not due to a limitation of DQI but rather the method by which the simulation took place. The simulation used a polynomial of degree 2. Choosing a higher degree would increase the probability of choosing smaller hamming weights. Furthermore, the coefficients determined in (13) may not be optimal. The w_i coefficients come from (67) of [5] which are calculated assuming the max-LINSAT instance.

V. DISCUSSION AND CONCLUSION

In this paper, we took a look at the application of DQI to the shortest codeword problem by providing an equality to the max-XORSAT problem. We observed the solution on an example code and verified its validity. Then, we used the idea of elementary symmetric polynomials proposed in the original paper to solve a spiked shortest codeword problem using DQI. Finally, we compared the results with a classical simulated annealing approach and provided some explanations for the relatively poor performance of DQI.

There is still significant work to be done to demonstrate exponential speedups for the shortest codeword problem. The general shortest codeword problem is an **NP-Hard** problem, so solving a general codeword instance is hard for DQI. To combat this issue, the algorithm in [5] utilizes LDPC codes to demonstrate speedup in specific instances of max-XORSAT. Similarly, for the shortest codeword problem, having a sparse generator matrix for a LDPC code would allow for efficient decoding using belief propagation.

Another avenue of approach to show exponential speedup for the shortest codeword problem would be to find a code generator matrix with a specific pattern. For example, the Reed-Solomon code, generated by a Vandermonde matrix, has an efficient decoder as used in the OPI problem of [5]. Finding similar code structures would allow exponential speedups for the shortest codeword problem.

Devising similar toy problems can also be a tool to find instances of exponential speedup. The strategy that we used in the paper to generate the objective function and corresponding DQI state can be applied to alternative modifications of the shortest code-

word problem. For instance, Farhi and Goldstone, [2], provide a different cost function,

$$h(z_0, z_1, \ldots, z_n) = \sum_{i=1}^{n} z_0(1 - z_i) + (1 - z_0) = z_0(n - w) + (1 - z_0),$$

where w is the hamming weight of $z = z_1, \ldots, z_n$. To solve this cost function, we could propose an objective function,

$$f_i(z_0, \mathbf{x}) = \sum_i (-1)^{z_0(C_i^T \mathbf{x}+1)} + (-1)^{1-z_0}.$$

By following the steps from section IV A, we can generate a DQI state for this objective function. This hamming weight function is interesting because there are 2^n states where $z_0 = 0$ and $h(z) = 1$, but only a small subset of codewords where $z_0 = 1$ and $w = n$ to make $h(z) = 0$. Therefore, to detect the state where $h(z) = 0$, we would expect the degree of the polynomial to be high in order to substantially separate the probability of $h(z) = 1$ states with the $h(z) = 0$ states, but this might not be the case.

Lastly, an interesting application of the shortest codeword problem, and generally the DQI algorithm, is for optimizing CNOT counts in circuits [3]. This paper frames this optimization problem as a decoding problem, $Hx = s^T$ where $|x|$ is minimized. We can turn this into a shortest codeword problem where $x = H^{-1}s^T$ where H^{-1} is our code generator matrix. Therefore, if H is invertible, DQI could be a potential candidate for developing CNOT circuits for quantum compilers.

---

[1] Tameem Albash and Daniel A. Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1), January 2018.

[2] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. Quantum adiabatic evolution algorithms versus simulated annealing, 2002.

[3] Timothée Goubault de Brugière, Marc Baboulin, Benoît Valiron, Simon Martiel, and Cyril Allouche. Decoding techniques applied to the compilation of cnot circuits for nisq architectures. *Science of Computer Programming*, 214:102726, February 2022.

[4] Simons Institute. Optimization by decoded quantum interferometry — quantum colloquium.

[5] Stephen P. Jordan, Noah Shutty, Mary Wootters, Adam Zalcman, Alexander Schmidhuber, Robbie King, Sergei V. Isakov, and Ryan Babbush. Optimization by decoded quantum interferometry, 2025.