Optimal Box Contraction for Solving Linear Systems via Simulated and Quantum Annealing

Sanjay Suresh[1] and Krishnan Suresh[2*]

[1]Computer Science, University of Wisconsin, Madison, 1210 W. Dayton Street, Madison, 53706, WI, USA.
[2*]Mechanical Engineering, University of Wisconsin, Madison, 1513 University Avenue, Madison, 53706, WI, USA.

*Corresponding author(s). E-mail(s): ksuresh@wisc.edu;
Contributing authors: ssuresh27@wisc.edu;

Abstract

Solving linear systems of equations is an important problem in engineering. Many quantum algorithms, such as the Harrow-Hassidim-Lloyd algorithm and the box algorithm, have been proposed for solving such systems.
The focus of this article is on improving the efficiency of the box algorithm. The basic principle behind this algorithm is to transform the linear system into a series of quadratic unconstrained binary optimization (QUBO) problems, which are then solved on annealing machines.
The computational efficiency of the box algorithm is entirely determined by the number of iterations, which, in turn, depends on the box contraction ratio, typically set to 0.5. Here, it is shown through theoretical analysis that a contraction ratio of 0.5 is sub-optimal and that one can achieve a computational speed-up with a contraction ratio of 0.2. This is confirmed through numerical experiments where a computational speed-up between **20%** to **60%** is observed when the optimal contraction ratio is used.

Keywords: QUBO; Linear system of equations; Quantum annealing; Simulated annealing; D-WAVE; Quantum computing

1 Introduction

Solving least squares problems and linear systems of equations are of utmost importance in science and engineering. Many algorithms have been proposed to solve such problems on classical computers. Quantum computers have recently been proposed as an alternative since they can potentially accelerate the computation [1, 2]. In particular, the Harrow-Hassidim-Lloyd (HHL) algorithm is a landmark strategy for solving linear systems of equations on *gate-based quantum computers*. In theory, it offers an exponential speed-up over classical algorithms [3], and it has been further improved recently [4–9]. However, due to the accumulation of errors in current noisy intermediate-scale quantum (NISQ) computers [10], the HHL algorithm and its variants are limited, in practice, to 4×4 systems [9, 11, 12]. Furthermore, while extracting the final solution is impractical, one can always extract a feature of the solution vector. [13].

In parallel, *quantum annealing machines*, such as the D-Wave systems with several thousand qubits [14], have also been proposed for solving such problems since they are less susceptible to noise [15, 16]. The basic principle is to convert the least squares or linear system into a series of quadratic unconstrained binary optimization (QUBO) problems. For example, O'Malley and Vesselinov solved the least-squares problem using a finite-precision qubit representation [17]. Borle and Lomonaco conducted a theoretical and numerical analysis of this approach [18, 19].

This article focuses on solving linear systems of equations via the QUBO formulation. A linear system problem can always be converted into a minimization problem [18], and consequently, into a series of QUBO problems. However, current quantum annealing machines are only equipped with about 5600 qubits (at the time of writing), with additional restrictions on connectivity [14, 20–22]. This implies that (1) the size of the linear system is still limited, and/or (2) the solution can only be computed with limited precision. The first limitation can potentially be addressed through a hybrid

Gauss-Seidel strategy [23], where the linear system is reduced to a set of smaller but coupled subsystems, which are solved iteratively. The second limitation typically relies on the *iterative box algorithm* [24], the main focus of this article. Alternative iterative methods are proposed in [25] and [26].

In the box algorithm, the number of iterations strongly depends on the box contraction ratio, i.e., the ratio by which the box size reduces under certain conditions (see Section 2). This ratio is typically set to 0.5 [24]. It is shown here, through theoretical analysis, that a contraction ratio of 0.5 is sub-optimal, and a computational speed-up with a contraction ratio of (approximately) 0.2 can be achieved. This is confirmed through numerical experiments using simulated and quantum annealing.

2 Background

2.1 QUBO Formulation

Consider the following linear system of equations:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

where $\mathbf{A}$ is a $d \times d$ matrix. If $\mathbf{A}$ is positive-definite (assumed to be true in the remainder of the article), then solving Eq. (1) is equivalent to minimizing the potential energy:

$$\min_{\mathbf{x}} \Pi = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b} \tag{2}$$

Each real component x_j is represented using qubit variables to solve this on a quantum annealing machine. A well-known strategy is the two's complement radix representation [18, 19]; for example, a scalar variable x can be represented using m qubits as:

$$x = -q_1 2^{m-1} + \sum_{i=2}^{m} q_i 2^{i-2} \tag{3}$$

Since the number of qubits is often limited in a quantum machine, it is common to let $m = 2$, leading to:

$$x = -2q_1 + q_2 \tag{4}$$

Of course, this can only capture the numbers $\{-2, -1, 0, 1\}$. However, one can easily extend this to a wide range of real numbers through scaling L and offset c via (see Section 2.2 for further explanation):

$$x = c + L(-2q_1 + q_2) \tag{5}$$

This is often called the *box representation* [24]. Furthermore, one can easily generalize this to arbitrary dimension d via:

$$\mathbf{x} = \mathbf{c} + L(-2\mathbf{q}_1 + \mathbf{q}_2) \tag{6}$$

where $\mathbf{q}_1$ and $\mathbf{q}_2$ are qubit vectors of length d, i.e., a total of $2d$ qubits is used to capture $\mathbf{x}$. Thus, a d-dimensional system is associated with 4^d total states. This is illustrated schematically in Fig. 1 for $d = 2$.

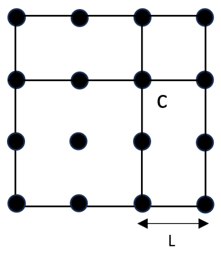


Fig. 1: The box representation for $d = 2$, where $\mathbf{c}$ is the current center of the box and L is the size of the box; the cross-lines centered around $\mathbf{c}$ help visualize the translation and contraction discussed below.

Since $\mathbf{x}$ is linear in $\mathbf{q}_1$ and $\mathbf{q}_2$, substituting Eq. (6) into Eq. (2) leads to a quadratic unconstrained *binary* optimization (QUBO) problem:

$$\min_{\mathbf{q}=\{\mathbf{q}_1,\mathbf{q}_2\}} \Pi = \frac{1}{2}\mathbf{q}^T\mathbf{Q}'\mathbf{q} + \mathbf{q}^T\mathbf{r} \tag{7}$$

where an entry $\mathbf{Q}'_{ij}$ represents coupling between the i^{th} and j^{th} qubit. Furthermore, since the qubit variables can only take the values 0 or 1, q can be replaced with q^2, since $0^2 = 0$ and $1^2 = 1$ [27]. Consequently, the linear term can be absorbed into the quadratic term [28], resulting in the standard form:

$$\min_{\mathbf{q}=\{\mathbf{q}_1,\mathbf{q}_2\}} \Pi = \frac{1}{2}\mathbf{q}^T\mathbf{Q}\mathbf{q} \tag{8}$$

where $\mathbf{Q}$ is symmetric.

2.2 Box Algorithm

The box algorithm is described in Algorithm 1; it exploits the QUBO formulation to solve Eq. (1) to high precision. During each iteration of the algorithm, the center $\mathbf{c}$ or scale L is updated as follows. In a particular iteration, when a QUBO problem in Eq. 8 is solved, if a lower potential energy state than the current state is reached, $\mathbf{c}$ is updated (referred to as a *translation*; see Fig 2a), else L is reduced by a **contraction ratio** β, typically, 0.5; see Fig 2b. The iteration is then continued until L is equal to or falls below the precision desired.

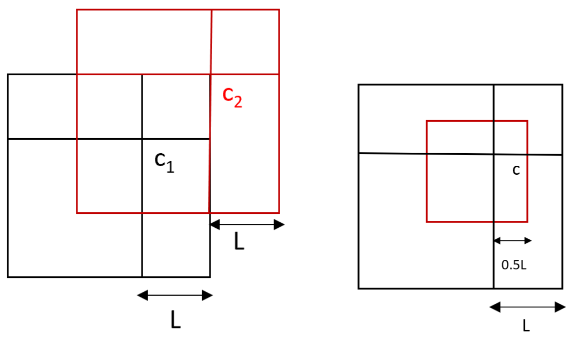


Fig. 2: Box algorithm: (a) translation, (b) contraction.

The box algorithm (see Algorithm 1) is now described in detail since it will be relevant for the remainder of the article. Observe in the algorithm that:

- Lines 2-7: Various quantities are initialized: (1) the center $\mathbf{c}$ is initialized to $\mathbf{0}$, (2) the length L is set to 1 (see remark below), (3) the qubit vectors $\mathbf{q}_1$ and $\mathbf{q}_2$ are created, (4) the number of translations N_t, and contractions N_t, are initialized to 0, (5) the contraction ratio β is initialized to 0.5 and (5) the potential energy is initialized to 0.
- Line 9: The unknown vector $\mathbf{x}$ is represented via the qubits and the current $\mathbf{c}$ and L .
- Line 10: The potential energy Π is formulated.
- Line 11: The minimum value of Π and the corresponding qubit values are determined either via simulated annealing or quantum annealing.
- Lines 12-15: If a lower energy state is found, the center is translated and N_t, the number of translations, is incremented (see remark below).
- Lines 16-18: Else, N_c, the number of contractions, is incremented and L is reduced by a factor of β.
- Line 20: The algorithm terminates if L is less than or equal to the desired tolerance, or if the number of total iterations $(N_t + N_c)$ is greater than an allowable $N_{\text{allowable}}$

Remark on Line 3: Although L is initialized to 1, the box algorithm is robust in that it converges for any reasonable value of L [24]. In other words, the box-algorithm can find solutions outside the initial box but may require numerous (initial) translation steps, i.e., a good choice for L will lead to faster convergence.

Remark on Line 12: The box algorithm is more stable, and unnecessary translations can be avoided if a small buffer is added by checking if $\Pi^* < \hat{\Pi}(1 + 10^{-8})$.

Algorithm 1 Box Algorithm

```
1:  procedure BoxAlg($\mathbf{A}$, $\mathbf{b}$, $\epsilon$,$N_{allowable}$ )
2:      $\mathbf{c} \leftarrow \mathbf{0}$                                  ▷ Center of length $d$
3:      $L \leftarrow 1$                                  ▷ Initialize box size
4:      $\mathbf{q}_1, \mathbf{q}_2 \leftarrow$ Qubits($d$)                  ▷ Create qubit arrays of length d
5:      $N_c = N_t = 0$                                  ▷ Translation and contraction steps set to 0
6:      $\beta = 0.5$                                  ▷ Contraction ratio
7:      $\hat{\Pi} = 0$                                  ▷ Initial potential energy
8:      repeat                                  ▷ Until convergence
9:          $\mathbf{x} \leftarrow \mathbf{c} + L(-2\mathbf{q}_1 + \mathbf{q}_2)$                  ▷ Symbolic expression
10:         $\Pi \leftarrow \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}$                  ▷ Construct QUBO
11:         $\Pi^*, \mathbf{q}_1^*, \mathbf{q}_2^* \leftarrow$ minimize($\Pi$)                  ▷ Solve QUBO
12:         if $\Pi^* < \hat{\Pi}$ then                  ▷ A lower energy state has been found.
13:             $\mathbf{c} \leftarrow \mathbf{c} + L(-2\mathbf{q}_1^* + \mathbf{q}_2^*)$          ▷ Translation of box
14:             $N_t = N_t + 1$                          ▷ Update translation counter
15:             $\hat{\Pi} = \Pi^*$                          ▷ Update the lowest potential energy
16:         else
17:             $L \leftarrow \beta L$                          ▷ Reduce box size
18:             $N_c = N_c + 1$                          ▷ Update contraction counter
19:         end if
20:     until $(L \leq \epsilon)$ or $(N_c + N_t > N_{allowable})$          ▷ Termination
21: end procedure                                  ▷ Output: solution $\mathbf{c}$
```

A typical convergence of the box algorithm in 2D is illustrated in Fig 3. Observe that the number of QUBO problems one must solve is equal to the total number of iterations ($N = N_t + N_c$). The objective of this article is to reduce N by finding an optimal value for β. In particular, in the next section, it is demonstrated that the default value of $\beta = 0.5$ recommended in the literature is sub-optimal.

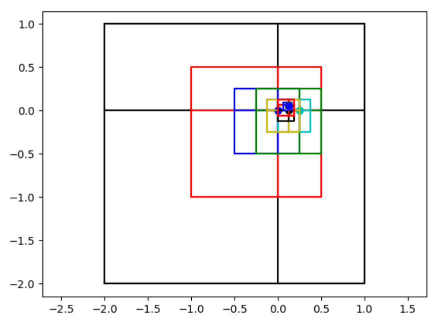


Fig. 3: Typical box convergence in 2D.

3 Box Algorithm Analysis

3.1 Contraction steps

Observe that each time the box contracts, the length of the box reduces by a factor of β, until $L \leq \epsilon$. Consequently, the total number of contractions, independent of the number of translations and the linear system being solved, is given by

$$N_c \sim \log_\beta \epsilon \tag{9}$$

3.2 Translation steps

Although the number of contraction steps is independent of β, the number of translations depends on β. The objective is to determine an upper bound $\hat{N}_t$ and average estimate $\overline{N}_t$ in terms of β and ϵ.

3.2.1 1D Box Algorithm

Consider one-dimensional problems that require only 2 qubits. Let L_i be the length of the box *before* the i-th contraction, i.e., $L_1 = 1$, $L_2 = \beta$, $L_3 = \beta^2$, etc. Further, at the start of the algorithm, $c = 0$, and c gets updated as follows:

$$c \leftarrow c + L_i(-2q_1 + q_2) \tag{10}$$

Let n_i be the number of translation steps *before* the i^{th} contraction. The objective is to find an upper bound for n_i.

Without loss of generality, it is assumed that the solution x^* lies within the range $(-2, 1)$ for theoretical analysis, and that, during each iteration, the algorithm finds the lowest energy solution of that particular iteration. Consider the possible sequence of translations, starting at $c = 0$, before the first contraction. If there is no translation, then $n_1 = 0$, else, c gets updated to $\{-2, -1, 1\}$, as per Eq (10). Suppose $c = -1$, corresponding to $q_1 = 1$, $q_2 = 1$. The solution x^* must lie in the range $(-1.5, -0.5)$. To prove this, assume the contradictory, i.e., assume that $x^* < -1.5$. Since x^* is the global minimum, Π (being a quadratic function) must be symmetric about x^*. This implies that $\Pi(-2) < \Pi(-1)$, since -2 is closer to x^* than -1. However, if $\Pi(-2) < \Pi(-1)$, the box would have translated to $c = -2$ and not $c = -1$ in the previous step, which is a contradiction. The same logic holds for if $x > -0.5$. As a result, x^* must lie in the range $(-1.5, -0.5)$. Similarly, if $c = -2$, or $c = 1$, x^* must lie in the range $(-2, -1.5)$ or $(0.5, 1)$ respectively.

No further translation is possible since it would require c to translate to an inferior solution. In other words, the box must contract in the next iteration. In summary, $n_1 \leq 1$.

After this, the box will contract, and $L_2 = \beta$, and x^* must lie in the range $[c - 1/2, c + 1/2]$, where c is the updated center. Therefore, the next sequence of translations can move the center by at most 0.5 units where each translation is at least $L_2 = \beta$, as per Eq. (10). Note that the center can also translate by -2β per Eq. (10), However, since an upper bound for N_t is being sought, only the worst-case scenario need to be considered. Therefore, the maximum number of translations is given by $n_2 \leq \frac{0.5}{\beta}$. After this, the box must contract, resulting in $L_3 = \beta^2$.

After the contraction, the solution x^* must lie in the range $[c - \frac{\beta}{2}, c + \frac{\beta}{2}]$, where c is the updated center. The next sequence of translations can move the center by at most

$\beta/2$ units and each translation is at least $L_3 = \beta^2$, as per Eq. (10). Consequently, the maximum number of translations $n_3 \leq (\beta/2)/\beta^2 = \frac{0.5}{\beta}$. After this, the box contracts to $L_4 = \beta^3$.

Repeating this logic for all N_c contractions, it follows that $n_1 \leq 1$ and $n_2, n_3, \ldots, n_{N_c} \leq \frac{1}{2\beta}$. Therefore, the total number of translation steps has the following upper bound:

$$\hat{N}_t = \sum_{i=1}^{N_c} n_i = 1 + \frac{1}{2\beta} + \frac{1}{2\beta} + \cdots + \frac{1}{2\beta} = 1 + \frac{N_c - 1}{2\beta} \tag{11}$$

It is safe to assume that $\beta \leq 0.5$ (see below for a justification).

$$\hat{N}_t = \frac{N_c}{2\beta} \tag{12}$$

Combining this with Eq. (9), an upper bound on the total iterations is given by:

$$\hat{N} = \hat{N}_t + N_c = \left(1 + \frac{1}{2\beta}\right) \log_\beta \epsilon \tag{13}$$

To find the optimal value of β, the derivative of $\hat{N}$ in Eq. 13 with respect to β is set to 0. Solving this numerically, $\beta^* \sim 0.232$, independent of ϵ. This is illustrated in Fig. 4. For this optimal value, one can observe a 32% reduction in the maximum box iterations, compared to the default value of $\beta = 0.5$. Further, observe that the number of translations increases for $\beta > 0.5$, justifying the earlier assumption.

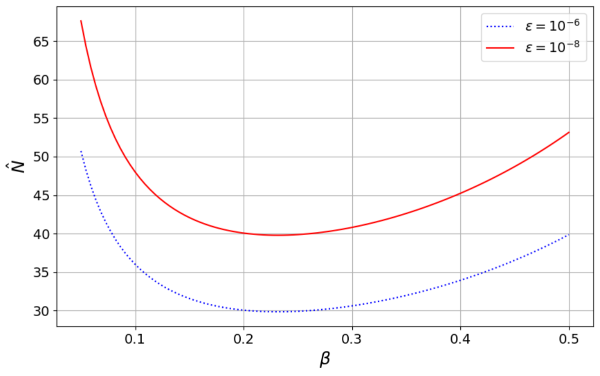


Fig. 4: Upper bound $\hat{N}$ on the number of box iterations.

Instead of the upper bound, one can also consider the average number of translations $\overline{N}_t$. Three different scenarios exist during each translation $(-1,\ -2,\ 1)$. If one assumes that there is an equal probability of translating in each of these directions (a very simplistic model), then the expected translation is given by $(|-1|+|-2|+|+1|)/3 = 4/3$ (as opposed to 1 in the worse case). Consequently

$$\overline{N}_t = \frac{3N_c}{8\beta} \tag{14}$$

Since the number of contractions remains the same:

$$\overline{N} = \overline{N}_t + N_c \approx \left(1 + \frac{3}{8\beta}\right) \log_\beta \epsilon \tag{15}$$

Fig. 5 illustrates $\overline{N}$ vs β. Taking the derivative of $\overline{N}$ with respect to β and setting it equal to 0, one can show that $\beta^* \approx 0.21$. For this optimal value, one can observe a 44% reduction in the average box iterations, compared to the default value of $\beta = 0.5$.

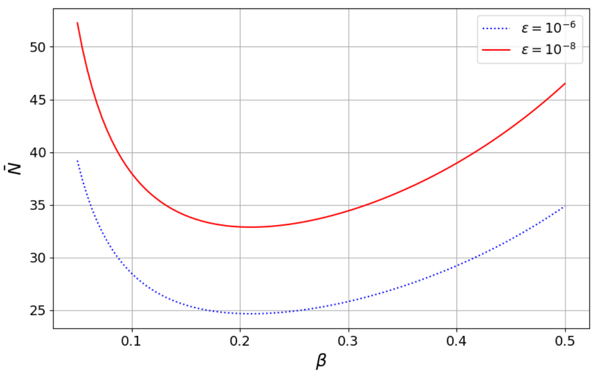


Fig. 5: Average number of box iterations $\overline{N}$.

3.2.2 Multi-Dimensional Problems

In this section, it is shown that the previous results in one dimension also hold in higher dimensions. It is assumed that the solution lies in the d-dimensional hyper-rectangle $(-2, 1)^d$, and that during each iteration, the algorithm finds the lowest energy solution within the desired precision.

For a d-dimensional problem, let $n_{i,k}$ represent the number of translations in the k^{th} dimension before the i^{th} contraction. Since the dimensions are independent, $n_{i,k}$ are also independent.

In order to prove this, consider the following hypothetical scenario: Before the i^{th} contraction, let the center translate in the first dimension until it cannot translate anymore in this direction. Then let it translate in dimension 2, and so on. Once dimension d is reached, this process is repeated until the box no longer can translate in any dimension. Let $n_{i,k}$ be the total number of times the box contracted in each dimension. By the premises of the box-algorithm, the box must now contract.

Following the logic from the previous section, it follows that $\forall i \geq 2, \forall k, n_{i,k} \leq \frac{1}{2\beta}$. However, the maximum number of translations is dictated by one or more of the dimensions. Therefore, let $n_i = \max_k n_{i,k}$. As a result, the upper bound on translation is given by

$$\hat{N}_t = \sum_{i=1}^{N_c-1} n_i = \frac{N_c}{2\beta} \tag{16}$$

Consequently,

$$\hat{N} = \left(1 + \frac{1}{2\beta}\right) \log_\beta \epsilon \tag{17}$$

irrespective of the number of dimensions. This is later confirmed in the next section through numerical experiments. Similar arguments can be made for the average case.

4 Numerical Experiments

Here, several numerical experiments are carried out to validate the theoretical analysis. The experiments rely heavily on simulated annealing (SA) since quantum annealing (QA) is expensive today. However, a limited number of QA experiments are also carried out. For SA, D-Wave's Neal annealer is used; for hybrid QA, D-Wave's LeapHybridSampler is used, and for (pure) QA, DWaveSampler, with EmbeddingComposite, is used. For all three methods, 20 samples were used.

The QUBO problems were constructed using the pyQUBO package [28]. The Python code used in generating the results in this section is available from the GitHub link provided towards the end of the article.

4.1 Positive Definite Matrices

Here, random d-dimensional positive definite matrices $\mathbf{A}$ are generated. Further, $\mathbf{x}$ must lie within $[-2, 1]^d$, $\mathbf{x}$ is first generated within this range, and then the corresponding right-hand-side $\mathbf{b}$ is constructed. The corresponding Python code is given below:

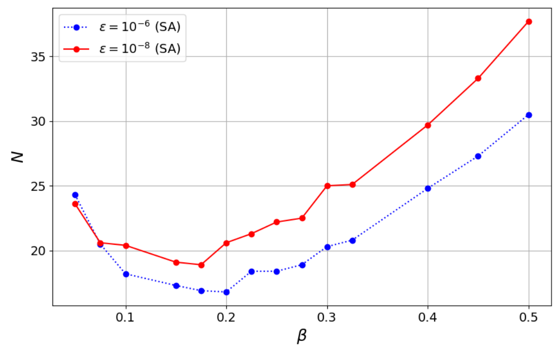
```python
B = np.random.rand(d, d)
A = d*np.eye(d)-(B + B.transpose())/2
x = np.array([random.uniform(-2, 1) for _ in range(d)])
```

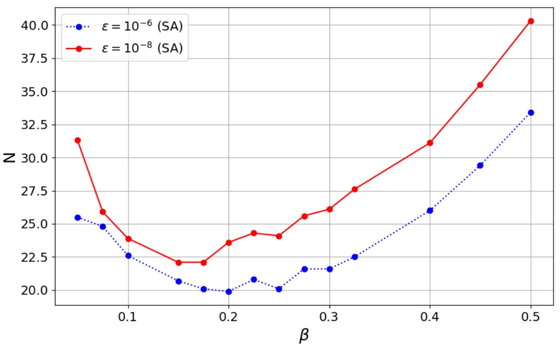
```
b = A.dot(xExact)
```

Listing 1: Generating d-dimensional positive definite matrices and right-hand side.

To capture the average behavior of the box algorithm, ten instances of $\mathbf{A}$ and $\mathbf{b}$, for $d = 2, d = 10$ and $d = 20$ are created. Finally, for each instance, the box algorithm (see Algorithm 1) is used to solve for $\mathbf{x}$ for $\epsilon = 10^{-6}$ and $\epsilon = 10^{-8}$, for various values of β. All experiments in this section are carried out using SA. The results are summarized in Fig. 6. Observe the following:

- All three graphs exhibit a minima around $\beta = 0.2$, independent of the dimension d and desired accuracy ϵ.
- The number of iterations is (nearly) independent of the dimension of the problem.
- The number of iterations is closer to the theoretical prediction $\overline{N}$ in Fig. 5 than to the upper bound prediction $\hat{N}$ in Fig. 4.





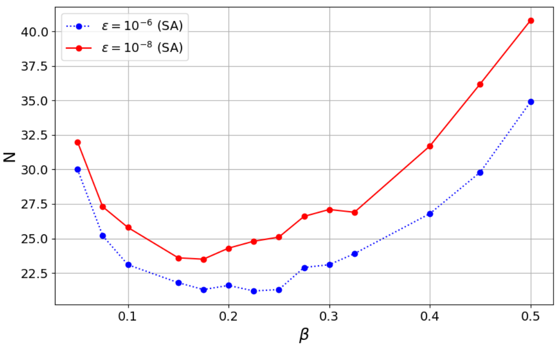


Fig. 6: Observed N vs β averaged over ten $d \times d$ problems: (a) d = 2, (b) d = 10, and (c) d = 20.

4.2 1D Poisson Problem

For the next experiment, a 6 × 6 matrix **A** that arises from a finite difference formulation of 1D Poisson problem [29] is constructed:

```
A = np.array([[6,-6,0,0,0,0],[-6,12,-6,0,0,0],
[0,-6,12,-6,0,0],[0,0,-6,12,-6,0],
[0,0,0,-6,12,-6],[0,0,0,0,-6,12]])
xExact = np.array([-np.pi/9, np.pi/11, -np.pi/20,
np.pi/8,  0.05*np.pi, -np.pi/5 ])
b = A.dot(xExact)])
```

Listing 2: Generating a 6-dimensional finite difference matrix and right hand side.

The results for SA, hybridQA, and QA are summarized in Figure 7. Observe that:

- The overall behavior of N vs. β is consistent with the theory.
- The hybridQA results precisely match that of SA for the three sampled points, suggesting that D-Wave probably relied entirely on CPU for this scenario.
- QA performed poorly compared to SA or hybridQA. This is consistent with the observations in [19]. However, even in this case, a 50% improvement in performance can be observed for $\beta = 0.2$, compared to $\beta = 0.5$.

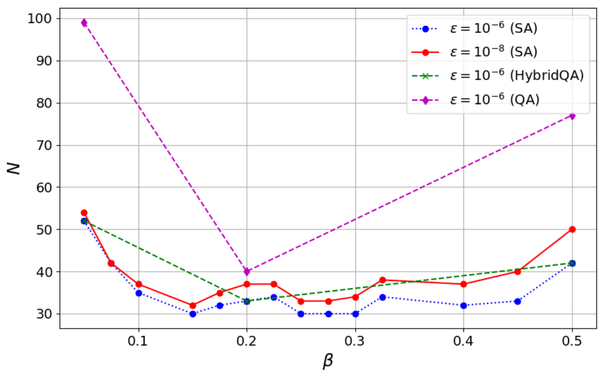


Fig. 7: Observed N versus β for a 1D Poisson problem.

5 Conclusions

The box algorithm is a popular method for solving linear systems of equations via the QUBO formulation. In this article, a theoretical analysis of the box algorithm was carried out that suggested that a computational speed-up can be easily achieved by making a simple modification to the algorithm. Specifically, the theory suggests that a 43% computational speed-up can be obtained by reducing the box contraction ratio from 0.5 to 0.2. This was confirmed through numerical experiments where a computational speed-up between 20% to 60% was observed.

While the article focused on linear systems, the strategy can be extended to least squares systems and other direct methods for solving linear systems via the QUBO formulation [25]. Further, the analysis here was restricted to 2-qubit representation of the scalar variable. The extension to the more general case needs to be investigated. Finally, while current quantum annealing can only solve small linear systems of equations [19], the rapid improvement in quantum technology holds much promise.

Compliance with ethical standards

The authors declare that they have no conflict of interest.

Data Availability

The Python code pertinent to this article is available at https://github.com/UW-ERSL/QUBOBoxContraction.

Acknowledgments

The authors would like to thank the Graduate School of UW-Madison for the Vilas Associate grant.

References

[1] G. Tosti Balducci, B. Chen, M. Möller, M. Gerritsma, R. De Breuker, Review and perspectives in quantum computing for partial differential equations in structural mechanics. Frontiers in Mechanical Engineering p. 75 (2022)

[2] Y. Wang, J.E. Kim, K. Suresh, Opportunities and challenges of quantum computing for engineering optimization. Journal of Computing and Information Science in Engineering **23**(6) (2023)

[3] A.W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations. Physical Review Letters **103**(15), 150502 (2009)

[4] A. Ambainis, Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations. arXiv preprint arXiv:1010.4458 (2010)

[5] A.M. Childs, R. Kothari, R.D. Somma, Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. SIAM Journal on Computing **46**(6), 1920–1950 (2017)

[6] X. Liu, H. Xie, Z. Liu, C. Zhao, Survey on the improvement and application of HHL algorithm. Journal of Physics: Conference Series **2333**(1), 012023 (2022)

[7] P.C. Costa, D. An, Y.R. Sanders, Y. Su, R. Babbush, D.W. Berry, Optimal scaling quantum linear-systems solver via discrete adiabatic theorem. PRX Quantum **3**(4), 040303 (2022)

[8] D. Babukhin, Harrow-hassidim-lloyd algorithm without ancilla postselection. Physical Review A **107**(4), 042408 (2023)

[9] N. Baskaran, A.S. Rawat, A. Jayashankar, D. Chakravarti, K. Sugisaki, S. Roy, S.B. Mandal, D. Mukherjee, V. Prasannaa, Adapting the harrow-hassidim-lloyd algorithm to quantum many-body theory. Physical Review Research **5**(4), 043113 (2023)

[10] J. Preskill, Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018)

[11] W. Ji, X. Meng, Demonstration of quantum linear equation solver on the ibm qiskit platform. 2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES) pp. 308–311 (2020)

[12] M. Zhang, L. Dong, Y. Zeng, N. Cao, Improved circuit implementation of the hhl algorithm and its simulations on qiskit. Scientific Reports **12**(1), 13287 (2022)

[13] B.D. Clader, B.C. Jacobs, C.R. Sprouse, Preconditioned quantum linear system algorithm. Physical Review Letters **110**(25), 250504 (2013)

[14] S.W. Shin, G. Smith, J.A. Smolin, U. Vazirani, How quantum is the d-wave machine? arXiv preprint arXiv:1401.7087 (2014)

[15] P. Hauke, H.G. Katzgraber, W. Lechner, H. Nishimori, W.D. Oliver, Perspectives of quantum annealing: Methods and implementations. Reports on Progress in Physics **83**(5), 054401 (2020)

[16] S. Yarkoni, E. Raponi, T. Bäck, S. Schmitt, Quantum annealing for industry applications: Introduction and review. Reports on Progress in Physics (2022)

[17] D. O'Malley, V.V. Vesselinov, B.S. Alexandrov, L.B. Alexandrov, Nonnegative/binary matrix factorization with a d-wave quantum annealer. PloS one

13(12), e0206653 (2018)

[18] A. Borle, S.J. Lomonaco, in *WALCOM: Algorithms and Computation: 13th International Conference, WALCOM 2019, Guwahati, India, February 27–March 2, 2019, Proceedings 13* (Springer, 2019), pp. 289–301

[19] A. Borle, S.J. Lomonaco, How viable is quantum annealing for solving linear algebra problems? arXiv preprint arXiv:2206.10576 (2022)

[20] A.D. King, J. Raymond, T. Lanting, R. Harris, A. Zucca, F. Altomare, A.J. Berkley, K. Boothby, S. Ejtemaee, C. Enderud, et al., Quantum critical dynamics in a 5,000-qubit programmable spin glass. Nature **617**(7959), 61–66 (2023)

[21] F.F. Flöther, Early quantum computing applications on the path towards precision medicine. arXiv preprint arXiv:2403.02733 (2024)

[22] V. Kumar, N. Baskaran, V. Prasannaa, K. Sugisaki, D. Mukherjee, K. Dyall, B. Das, Computation of relativistic and many-body effects in atomic systems using quantum annealing. Physical Review A **109**(4), 042808 (2024)

[23] G.G. Pollachini, J.P. Salazar, C.B. Góes, T.O. Maciel, E.I. Duzzioni, Hybrid classical-quantum approach to solve the heat equation using quantum annealers. Physical Review A **104**(3), 032426 (2021)

[24] S. Srivastava, V. Sundararaghavan, Box algorithm for the solution of differential equations on a quantum annealer. Physical Review A **99**(5), 052355 (2019)

[25] O.M. Raisuddin, S. De, Feqa: Finite element computations on quantum annealers. Computer Methods in Applied Mechanics and Engineering **395**, 115014 (2022)

[26] C.C. Chang, A. Gambhir, T.S. Humble, S. Sota, Quantum annealing for systems of polynomial equations. Scientific reports **9**(1), 10258 (2019)

[27] S.W. Park, H. Lee, B.C. Kim, Y. Woo, K. Jun, in *2021 International Conference on Information and Communication Technology Convergence (ICTC)* (IEEE, 2021), pp. 1363–1367

[28] M. Zaman, K. Tanahashi, S. Tanaka, Pyqubo: Python library for mapping combinatorial optimization problems to qubo form. IEEE Transactions on Computers **71**(4), 838–850 (2021)

[29] R. Conley, D. Choi, G. Medwig, E. Mroczko, D. Wan, P. Castillo, K. Yu, Quantum optimization algorithm for solving elliptic boundary value problems on d-wave quantum annealing device. Quantum Computing, Communication, and Simulation III **12446**, 53–63 (2023)