Journal of Computer Science

Review Article

A Survey of Quadratization Methods Used in QUBO Formulations and their Usage in Quantum and Digital Annealing

Geethika Prabhath[1], Anuradha Mahasinghe[2], Nalin Ranasinghe[1] and Kasun De Zoysa[1]

[1]*School of Computing, University of Colombo, Colombo, Sri Lanka*
[2]*Center for Mathematical Modeling, University of Colombo, Colombo, Sri Lanka*

Article history
Received: 21-03-2025
Revised: 16-06-2025
Accepted: 25-06-2025

Corresponding Author:
Geethika Prabhath
School of Computing, University of Colombo, Colombo, Sri Lanka
Email: anuradhamahasinghe@maths.cmb.ac.lk

Abstract: Quantum annealing is a metaheuristic method aimed at finding the global minimum of a pseudo-Boolean quadratic function. To solve a problem by a quantum annealer, it must be restated as a QUBO (quadratic unconstrained binary optimization) problem, the standard format acceptable to an annealer. However, there are many natural problems that turn out be of cubic or higher order, yet promising solutions can be obtained for many instances using quantum annealers, after converting to the QUBO format, using order reduction criteria. Accordingly, quadratization becomes a crucial preprocessing step in order to make use of present-day quantum annealing hardware that relies on coupling technology, which can only process interactions of at most degree two. A critical challenge is that quadratization methods often require many additional variables, degrading the performance of the annealers. A number of algorithmic and heuristic quadratization techniques have been proposed and used for annealing in the literature, affecting the performance of the annealers differently. We examine a number of such quadratization techniques, focusing on their efficiency, compactness, and applicability. We analyze the approaches motivated by diverse goals and highlight the transition from foundational approaches to more recent compact approaches. An evaluation of the current usage of different quadratization methods in the literature is carried out at the end of the survey. Further, we comment on the limitations on usage and show possible opportunities to enhance the performance of annealers.

Keywords: Scientific Computing, Optimization, Computational Complexity, Integrated Quantum Computer Systems

Introduction

Quantum Computing

Quantum Computing is a revolutionary computing paradigm harnessing the unique principles of quantum mechanics, superposition, entanglement, unitary evolution, and measurement, to process information. Quantum computers do not aim for a replacement of classical computers; rather, they can process information in a way that classical computing simply cannot, thus providing significant advantages over solving certain problems that are beyond the capability of classical computing.

From a practical viewpoint, there are two main approaches in utilizing quantum effects for computing, the pioneering gate based circuit model (Deutsch, 1989) and the adiabatic quantum computing model proposed about 25 years ago (Farhi et al., 2000, 2001). The circuit model, which is based on the Heisenberg-Born picture of quantum mechanics, embodies an evolution of the system with discrete steps, that is achieved by applying quantum gates that represent the unitary operators in finite-dimensional spaces. In contrast to that, the adiabatic model follows the Schrödinger picture of quantum mechanics that describes the evolution of a quantum system in a continuous framework, through the application of a Hermitian operator, which represents the energy Hamiltonian of the system (Mahasinghe et al., 2019). The aim of introducing the adiabatic framework was to obtain efficient solutions to instances of NP-complete problems, which had not shown much progress in the gate model.

Adiabatic Quantum Computing

Recall that adiabatic quantum computation (AQC) was introduced specifically to handle NP problem instances; it works by encoding an optimization problem in the form of a cost function in a time-dependent Hamiltonian. AQC incorporates the process of evolution of the initial problem Hamiltonian, whose ground state is easy to prepare, into a final Hamiltonian that encodes the solution to the problem through the ground state (Albash and Lidar, 2018). The Hamiltonian is guaranteed to preserve the ground state-hence will provide the best solution, if the evolution is carried out slowly enough.

Quantum Annealing

The sufficiently slow evolution of the energy Hamiltonian in the adiabatic paradigm of computing resembles the slow cooling of metals to achieve a stable, low-energy state, which has been the motivation for the well-known optimization heuristic simulated annealing (SA). Inspired by SA, quantum annealing was introduced in the early nineties (Apolloni et al., 1990) as its quantum counterpart and was developed further. The current state of annealing was formulated in Kadowaki and Nishimori (1988). Later, following the footsteps of the adiabatic algorithm, QA was seen as the practical pathway to achieve the quantum speedup, and the Canadian company D-wave systems started making commercial quantum annealers that provide promising solutions to many instances of NP problems. These machines are seeking the ground state of the generic Ising spin model, yet, from a computer science perspective, it is more convenient to deal with the QUBO model.

The conversion from Ising to QUBO is performed via a simple transformation, and it is straightforward. QA aims to surpass SA by leveraging special quantum mechanical fluctuations such as quantum tunneling when seeking for the global minima of an energy landscape, where QA can escape from a local minima through the energy barrier as depicted in Figure 1 (Rajak et al., 2023).

Quantum annealing is generally more noise-tolerant due to its ability to leverage environmental noise for certain computations, whereas universal quantum computing requires precise error correction to mitigate the effect of noise (Kapit and Oganesyan, 2017; Dickson et al., 2013). Also, QA is found to be resilient against dephasing errors (Yarkoni et al., 2022). These reasons have caused QA to draw the attention of the computer science community. Where leading quantum companies like IBM have only developed 1000+ error-corrected qubit QPUs so far, D-wave has 5000+ qubit QPUs for commercial usage.

Many optimization problems, such as traveling salesperson (Martonak et al., 2004), job-shop scheduling (Carugno et al., 2022), isomorphism (Calude et al., 2017), route planning (Hua et al., 2024), and graph coloring (Dinneen et al., 2019), prime factorizing (Jiang et al., 2018), and many other problems (Lucas, 2014), were encoded and solved through QA, leading to the solvability of real-life problems such as logistics optimization, machine learning, and financial modeling, route optimization.

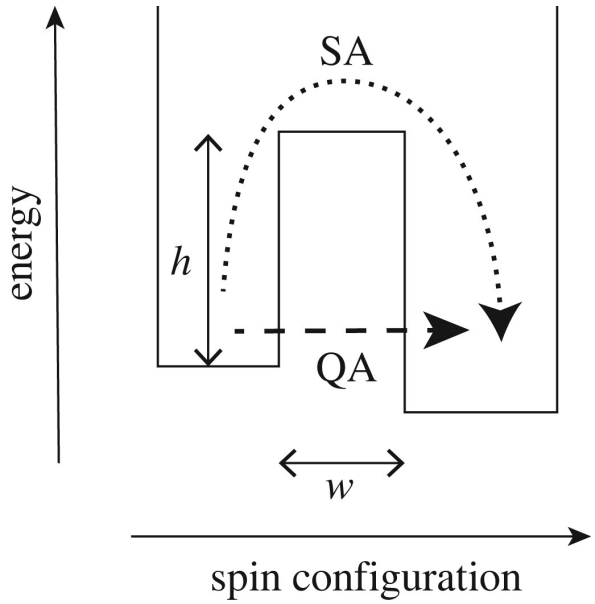

Fig. 1: Thermal fluctuations vs quantum tunneling in a system with local energy minima separated by an energy barrier

While classical optimization methods such as linear programming, constraint programming, or metaheuristics (simulated annealing) are widely used for combinatorial problems, they typically operate directly on the original problem formulation. Although approaches like simulated annealing use QUBO, it is not essential in classical methods. But in contrast, quantum annealing requires a transformation to QUBO (or Ising) form, which introduces unique challenges such as compact-quadratization and embedding. This distinction highlights the specialized role of QUBO as the standard interface for quantum hardware, necessitating dedicated preprocessing techniques not required in classical optimization. This contrast leads to more focused and dedicated studying in the annealing and QUBO realm, separately from the traditional optimization approaches.

D-Wave Machines

D-Wave quantum annealers are specialized quantum computers that draw optima or near-optima solutions to a cost function with binary variables (Lanting et al., 2014; Johnson et al., 2011). Using the theories of QA, it was the first to be commercially introduced as a quantum annealing device, with specialized hardware for solving Ising / QUBO problems (Ding et al., 2024). D-wave provides cloud access to its use of hardware commercially through their Leap service.

DOI: 10.3844/jcssp.2026.309.333

The qubits in quantum annealers are arranged in a particular way, which is called the network topology, and it becomes a critical factor when programming these devices. Earlier, D-wave used a Chimera graph for network topology, and they enhanced qubit connectivity by replacing it with Pegasus topology in their next-generation quantum annealers. The quantum processor unit uses a lattice that is divided into Pegasus unit cells, where each cell contains eight qubits arranged in a bipartite graph. This enhances the qubit connectivity by increasing the degree of each qubit to 15 and introducing odd couplers that connect qubits in adjacent rows or columns (Boothby et al., 2020).

This inherent hardware design, with the use of couplers, introduces a critical constraint to the utilization of currently available quantum annealing devices, which is the limitation of 2-body interactions. This happens to be a key reason why QA requires quadratization of higher-order terms in cost functions.

For the D-Wave quantum computer, the Hamiltonian may be represented as follows (D-WaveSystems, 2024):

$$\mathcal{H}_{\text{ising}} = -\frac{A(s)}{2}\underbrace{\left(\sum_i \hat{\sigma}_x^{(i)}\right)}_{\text{Initial Hamiltonian}} + \frac{B(s)}{2}\underbrace{\left(\sum_i h_i\hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right)}_{\text{Final Hamiltonian}} \quad (1)$$

where,

$$h_i = \frac{1}{4}\sum_{k=1}^{n^2}\left(q_{i,k} + q_{k,i}\right), \quad (2)$$

$$J_{i,j} = \begin{cases} \frac{q_{i,j}+q_{j,i}}{4} & \text{if } i << j, \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

In Eq. 1 , $\hat{\sigma}_z^{(i)}$ denotes that Pauli z-gate is acting on the ith qubit, and $\hat{\sigma}_x^{(i)}$ is described similarly. The terms $q_{i,k}$ in Equation 2 come from the QUBO problem, which can be expressed as the minimization of a quadratic objective function $f(x) = x^T Q x$, where x is a n-vector of binary variables and Q is an upper-triangular $n \times n$ matrix:

$$x^* = \min_x \sum_{i\leq j} x_i Q_{(i,j)} x_j, \text{ where } x_i \in \{0,1\}. \quad (4)$$

Accordingly, problems with terms above quadratic cannot be encoded in the energy function, requiring quadratization pre-processing for any higher-order interaction.

Quantum-Inspired Digital Annealing

Digital annealing is a computing paradigm inspired by QA, which was aimed at minimizing the decoherence and scaling issues encountered in QA. By incorporating merits from both quantum and general-purpose computing, Fujitsu (Aramon et al., 2019), in collaboration with the University of Toronto, developed digital annealers. Naturally, digital annealers require problems to be encoded in the form of QUBO. Therefore, quadratization becomes an essential step in solving cubic and higher-order problems using these devices (Sao et al., 2019).

Another Annealing device developed by Hitachi is the CMOS annealing machine. This is a similar quantum-inspired device, requiring quadratization for higher-order functions (Codognet, 2021).

Pseudo-Boolean Functions

Many real-world combinatorial optimization problems, decisions that tend to be inherently binary (yes/no, on/off, selected/not selected). Pseudo-Boolean functions (PBF) are ideal mathematical formulations for such optimization problems of Boolean variables expressed as multi-linear polynomials (Zielinski et al., 2023). They are central to optimization problems used to solve with the aforementioned solvers (quantum/classical) due to their ability to model cost/energy functions. In the case of QA, after formulating a PBF, they are mapped to a Hamiltonian in a matrix form, which should be in the Ising or QUBO form so that it can be embedded in the quantum hardware (Fernández-Villaverde and Hull, 2023).

The formula for a Pseudo-Boolean function is typically expressed as:

$$f(x_1, x_2, \ldots, x_n) = \sum_{S\subseteq\{1,2,\ldots,n\}} c_S \prod_{i\in S} x_i \quad (5)$$

where,

- $x_i \in \{0,1\}$ are binary variables,
- c_S are coefficients associated with subsets S of the set of variables $\{x_1, x_2, \ldots, x_n\}$, and,
- $\prod_{i\in S} x_i$ represents the product of all variables x_i in the subset S.

The degree of a pseudo-Boolean function is defined as the size of the largest subset S for which the coefficient $c_S \nvDash 0$. Pseudo-Boolean functions are important as their applicability lies in a wide variety of domains other than optimization problems. It must be noted that minimizing a pseudo-Boolean function is an NP-hard problem (Boros and Gruber, 2014). As the adiabatic model is specially designed to address the NP problems, annealers have provided promising solutions to this minimization problem.

Quadratic Unconstrained Binary Optimization

QUBO, formerly referred to as Unconstrained Binary Quadratic Programming (UBQP), is the integral format that modern annealers can work with. QUBO demonstrates real-world relevance in many domains such as portfolio optimization, scheduling, Max-Cut, etc. (Glover, 1975; Kochenberger and Glover, 2006;

Kochenberger et al., 2014). A QUBO can essentially be viewed as a 2nd degree PBF in the context of optimization. Its special feature is that a QUBO can formulate a whole problem into a function, whether it's unconstrained or not. The constraints are handled by adding penalties and reformulating the problem to an unconstrained version-still preserving the original solution space-to act as the input to the quantum annealer. When the degree of the function exceeds 2, the problem is known as a Higher-Order Unconstrained Binary Optimization (HUBO).

Quadratization

Quadratization is the process of transforming a (pseudo-Boolean in this context) function, which may include higher-order or linear terms, into a quadratic function while preserving its minima and maxima, generally at the expense of adding new auxiliary (ancillary) variables. As discussed before, this is crucial for quantum annealing because current hardware can only handle quadratic terms due to its architectural limitations. Quadratization is sometimes referred to with different terminologies such as order reduction, locality reduction, and degree reduction in literature. In the context of quantum annealing, we deal with the quadratization of HUBOs to convert them into QUBOs.

Given a pseudo-Boolean function $f : \mathbb{B}^n \to \mathbb{R}$ as in (1), where $\mathbb{R}$ denotes the set of reals, the following minimization problem:

$$\min_{x\in\mathbb{B}^n} f(x) \tag{6}$$

is to be reduced to a quadratic pseudo-Boolean function $g(x, w)$, where $w \in \mathbb{B}^m$ is a set of auxiliary (new) variables, such that the equality:

$$f(x) = \min_{w\in\mathbb{B}^m} g(x, w) \tag{7}$$

holds for all $x \in \mathbb{B}^n$. Then, such a g is the quadratization of f (Boros and Gruber, 2014).

Although there exist a few quadratization approaches that introduce zero auxiliary variables (Okada et al., 2015), general methods always introduce new variables, and the minimization of those variables is a challenging task. In quantum annealing, each new variable requires an extra qubit in quantum processors. In the case of the D-Wave computer, the minor embedding step maps the variables from the quadratic cost function to the QPU topology (D-WaveConcepts, 2024). The number of qubits that can be utilized for a problem remains a large constraint due to the scarcity of qubits in the QPUs. Moreover, quadratization with the minimum number of auxiliary variables is itself shown to be an NP-hard problem (Boros and Gruber, 2014).

Despite numerous quadratization methods, a significant research gap exists in the systematic empirical comparison and principled selection of these methods, leading to an uncritical reliance on a few classical techniques in practical applications. This observation seemed to particularly hinder the performance in resource-constrained computing, such as quantum and digital annealing.

We present a summary of most of the promising quadratization methods through the Tables 1 and 2; we have stated exact bounds where applicable. Note that many of them remain theoretical as we are yet to see proper usages of most of the following techniques about to be discussed.

Quadratization and Properties

We introduce some of the properties of PBFs, quadratization, and target QUBOs, and the desirable outcomes we seek from general quadratization.

Submodularity

A desirable property for a target QUBO is being submodular after quadratization (Boros and Gruber, 2014). But this property cannot be easily maintained from a submodular higher-order function through the conversion to quadratic form. The importance of submodularity is that submodular functions are known to be solved in polynomial time (Schrijver, 2000; Iwata et al., 2001). It is sometimes considered the discrete counterpart of convexity.

A submodular function on a finite set V is a function $f : 2^V \to \mathbb{R}$ that assigns a real value to each subset of V. It satisfies the submodularity property:

$$f(Y) + f(Z) \geq f(Y \cap Z) + f(Y \cup Z)$$

for all subsets $Y, Z \subseteq V$ (Schrijver, 2000). A general result to find if a function is submodular is given below.

A quadratic pseudo-Boolean function is submodular if, and only if, all quadratic terms have non-positive coefficients (Boros and Hammer, 2002).

Compact Quadratization

Compact quadratization refers to reducing the number of auxiliary variables introduced during quadratization. Compact quadratizations are steadily studied, and they increase the efficiency of overall optimization (Boros et al., 2020). A good quadratization should minimize the number of auxiliary variables, which increase the dimensionality of the problem, and the number of quadratic terms that make the problem harder to embed on quantum hardware.

Compared to submodularity (which is particularly harder to maintain), reducing the auxiliary variable count is highly desirable, and it is a constantly researched topic in quadratization (Boros and Gruber, 2014). In our review, we consider a few approaches that aim to perform efficient compact quadratizations in the upcoming sections.

DOI: 10.3844/jcssp.2026.309.333

Symmetry

Symmetry plays an important role in quantum computing, quantum information, and quantum communication (Croke, 2015; Hashagen, 2018; Mahasinghe et al., 2015, 2014). A symmetric PBF's value only depends on the Hamming weight of the input. In other words, a function f is symmetric if it is invariant under any permutation of the coordinates of its variables. According to Anthony et al. (2016), a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ is symmetric if there exists a function $k\{0, 1, \ldots, \} \not\to$ such$\mathbb{R}$that:

$$f(x) = k\left(\sum_{j=1}^{n} x_j\right). \tag{8}$$

In simple terms, this implies that a symmetric function depends solely on the number of ones in the input.

Review of Quadratization Techniques

This review examines both general quadratization techniques applicable to arbitrary pseudo-Boolean functions (PBFs) and specialized methods tailored to structured functions or specific problem types. Except for gadget-based techniques, most methods discussed are applicable across domains that require second-degree PBFs. Our selection includes both classical methods with continued relevance in QA workflows and novel approaches that aim to improve beyond traditional techniques. We excluded minor variants and adaptations (Rosenberg, PTR, NTR-based methods reused without novelty) to maintain focus. For a comprehensive list of quadratization techniques, we refer the reader to Dattani's work (Dattani, 2019), which provides broad coverage of known methods but offers only brief technical descriptions of each. Our source collection involved tools like Google Scholar and Consensus AI, and citation tracing from key publications.

Unlike previous surveys that focus on mathematical formulations (Louveaux and Buchheim, 2018; Dattani, 2019), in contrast, this review emphasizes the motivations, usage trends, computational aspects in real-life usage, and problem-specific applicability of selected quadratization techniques. Rather than detailing each method's derivation and application, the focus is on synthesizing insights about how and why certain techniques have been introduced and adopted in practice and highlighting gaps in empirical evaluation and methodological guidance. While observing the computational applicability of them, the ultimate goal is to support the identification of quadratization strategies that can most effectively enhance QA performance.

So far, the literature on quadratization has primarily focused on two main approaches: term-wise quadratization, which handles individual positive and negative monomials (terms) separately, and substitution-based quadratization, which reformulates the entire function by introducing constraints, if not a 'penalty', to force quadratic relationships. While auxiliary variables are typically needed, we also cover methods that aim to reduce or eliminate them.

Comparison of Quadratization Methods

The quadratization methods reviewed in this work are highly diverse, ranging from classical substitutions and penalty-based formulations to more recent structure-specific and preprocessing approaches. Due to their differing applicability, goals, and mathematical structures, a fully unified comparison is not straightforward. We can, however, broadly categorize them as methods that introduce and do not introduce auxiliary variables.

We present a broad grouping of these methods based on their primary strategy:

- Classical Substitution-based and term-wise methods (Rosenberg, Ishikawa, Freedman)
- Recent advancements in classical approaches (Compact positive monomial reduction - Boros et al)
- Structure-aware methods for symmetric functions (Boros et al)
- Algorithmic approaches (Local Structure reduction - Schmidbauer et al, Direct quadratization in Ising space - Mandal et al)
- Preprocessing techniques for the reduction of auxiliary variables (Integer programming - Verma et al)
- Methods that introduce no auxiliary variables (Deduc-Reduc, Split-Reduc - Dattani et al)

Where applicable, one can compare these approaches based on standard criteria, such as the number of auxiliary variables introduced, preservation of submodularity, practical runtime, implementation notes, problem sparsity or density after quadratization, and the accuracy of the reduced problem as well as the optimized answer.

Quadratization With Auxiliary Variables

In this section, we review some of the most common and predominant types of quadratization techniques that have been researched over a few decades. Based on a couple of foundational methods, certain aspects have been improved over time.

Early Works and Foundational Methods

Substitution-Based Methods

Rosenberg's Substitution Method (1975)

One of the earliest and highly recognized methods for quadratization is Rosenberg's substitution, which falls under reduction by substitution method. This method substitutes auxiliary variables to replace higher-order terms with equivalent quadratic expressions, while

introducing a penalty term to enforce the actual values of the original function. For example, consider a PBF as follows:

$f(a,b,c) = abc + ab + c$

The substitution will pick the product ab and replace it with x for each occurrence of ab in the function, and will add the penalty term $MD(a,b,x)$. Here, M is chosen to be a large positive number so that, whenever $xy \models z$ and thus $D(xyz) > 0$, it is impossible for $\bar{f}$ to take the minimum:

$$\bar{f}(a,b,c,x) = xc + x + c + MD(a,b,x) \tag{9}$$

While $x, y, z \in \mathbb{B}$, it is defined that:

$$D(x,y,z) = xy - 2xz - 2yz + 3z \tag{10}$$

The final result and the number of new variables introduced will depend on which specific products one chooses; thus, it will ultimately depend on the PBF itself. It can be observed that by repeating the above technique, any higher order PBF can be formulated to the quadratic case (Rosenberg, 1975).

However, this method has several disadvantages. The very large coefficient in the penalty term makes the reformulations highly non-submodular, and it can introduce up to $(n-2)$ new variables, making it somewhat inefficient to use. An important thing to notice is that a single variable can be used across the entire function instead of considering each monomial.

Despite this, Rosenberg's reduction method seems to be the most used reduction method for many problems solved with QA and other applications. After Rosenberg's method, the most practically used methods will be discussed in the following section of Term-wise quadratization.

Term-wise Methods

Also referred to as Reduction by Minimum Selection, these methods consider each monomial term and apply quadratization for negative and positive monomials separately using independent auxiliary variables.

Kolmogorov and Zabih's Method (2004)

This method introduced first and is suitable for negative monomial quadratization in the context of graph cut optimizations but is limited to degree 3 monomials (Kolmogorov and Zabih, 2004).

Freedman and Drineas (2005) - Negative Monomial Reduction (NTR)

Later introduced, Freedman's approach provides an optimal quadratization for negative monomials requiring only one auxiliary variable. Which is considered the most ideal method for negative monomials to date. According to Freedman and Drineas (2005), for a degree n negative monomial,

$$N_n(x) = \min_{y\in\{0,1\}}(n-1)y - \sum_{i=1}^{n} x_i y \tag{11}$$

Gives the quadratization where y is the only added auxiliary variable. This method is a direct improvement on Kolmogorov and Zabih (2004)'s work, and has been used extensively to date. This Negative Monomial Reduction (NTR) reductions can be used to quadratize a function while maintaining perfect submodularity if all the monomials of the higher-order function are negative.

Ishikawa's Method (2011) - Positive Monomial Reduction (PTR)

This method provides a significant reduction in the number of added new variables over the Rosenberg (1975) method for positive monomials. It remains a foundational technique for later improvements in positive monomial reduction, specifically. According to Ishikawa (2011), the quadratization of a degree n positive monomial $P_n(x)$ is given as:

$$P_n(x) = \min_{y\in\{0,1\}^m} \sum_{i=1}^{m} y_i \left(c_{i,n}(-|x| + 2i) - 1\right) + \frac{|x|(|x|-1)}{2} \tag{12}$$

where $|x| = \sum_{i=1}^{n} x_i$, and $m = \lfloor \frac{n-1}{2} \rfloor$. The coefficients $c_{i,n}$ are defined as:

$$c_{i,n} = \begin{cases} 1, & \text{if } n \text{ is odd and } i = m, \\ 2, & \text{otherwise.} \end{cases}$$

This quadratization method will add $\left\lfloor \frac{n-1}{2} \right\rfloor$ new auxiliary variables and will have $\binom{n}{2}$ positive quadratic terms.

A downside of this method is that the reformulated function is highly non-submodular due to the number of positive terms. However, Ishikawa (2011) claims very good computational results compared to Rosenberg.

In later sections, we can observe that there have been improvements to this method for special cases of functions (Boros et al. 2020) and an overall general improvement with a logarithmic bound by Boros et al. (2020).

Recent Advances in Quadratization

We observe that recent research on quadratization has leaned towards Compact Quadratization and some special cases of PBF, such as Symmetric Functions, Parity Functions, Zero until k Functions, and Even/Odd Support Functions, etc., especially by Anthony et al. (2016); Boros et al. (2020). Preserving submodularity, establishing tighter bounds, balancing computational performance (Boros et al., 2020), and overall reducing the variable count for general cases (Boros et al., 2020) have been some other focuses. In the following section, we will look at some of the significant and key findings from the research about quadratization in special structures of PBF.

Zero-Until-k Functions

Boros et al. (2020) presents two lower bounds for Zero-until-k functions. Through Theorem 2 and Theorem 3 in their paper, they prove that there exists a lower bound of $\Omega(2^{\frac{n}{2}})$ for almost all PBFs and another lower bound of $m \geq \lceil log(k) \rceil - 1$ for all PBFs. The upper bound for Zero-until-k functions is the same as for general pseudo-Boolean functions, $O(2^{\frac{n}{2}})$ as previously established (Anthony et al., 2017).

Even(Odd) Support Functions

The lower bound $\lceil log(n) \rceil - 1$ is proven for Even(Odd) support Functions by Boros et al. (2020). Again, the upper bound for this function class is the same as previously established by Anthony et al. (2017), which is $O(2^{\frac{n}{2}})$.

Symmetric Functions

It was established by Anthony et al. (2016) that the lower bound for the number of variables required to quadratize symmetric functions is $\Omega(\sqrt{n})$. Improving upon that, Boros et al. (2020) states a quadratization in their paper for symmetric functions using $2\lceil\sqrt{n+1}\rceil$ auxiliary variables.

Theorem 1 (Boros et al., 2020)

Let $f(x_1, \ldots, x_n)$ be a symmetric pseudo-Boolean function such that $f(x) = r(|x|)$, with $r : \mathbb{N} \to \mathbb{R}$ and $r(k) = 0$ for $k > n$, by convention. Let $l = \lceil\sqrt{n} + 1\rceil$, and choose $M \in \mathbb{R}$ such that $M > |r(k)|$ for all $k \in \mathbb{Z}$. Then,

$$g(x,y,z) = \sum_{i=0}^{l-1}\sum_{j=0}^{l-1} r(il+j)y_i z_j + 2M\left(1 - \sum_{i=0}^{l-1} y_i\right)^2 + 2M\left(1 - \sum_{j=0}^{l-1} z_j\right)^2 + 2M\left(|x| - \left(l\sum_{i=0}^{l-1} i y_i + \sum_{j=0}^{l-1} j z_j\right)\right)^2 \quad (13)$$

is a quadratization of f using $2\lceil\sqrt{n}+1\rceil = O(\sqrt{n})$ auxiliary variables $y_i, z_i, i = 0, \ldots, l-1$.

At Least k-out-of-n

Boros et al. (2020) proves that At least k-out-of-n functions cannot be quadratized using less than $\lceil log(k) \rceil - 1$ auxiliary variables, providing a lower bound.

Theorem 2 (Boros et al., 2020)

For each integer $0 \leq k \leq n$, the At Least k-out-of-n function $f_{\geq}k$ admits a quadratization given by:

$$G_k(x,y,z) = \tfrac{1}{2}\left(A_k(x,y,z)\right)\left(A_k(x,y,z) - 1\right) + (1-z) \quad (14)$$

where $A_k(x,y,z)$ is a suitable auxiliary function. This quadratization requires:

$$l = \max(\lceil\log(k)\rceil, \lceil\log(n-k)\rceil) \leq \lceil\log(n)\rceil$$

auxiliary variables, consisting of $y \in \{0,1\}^{l-1}$ and an additional variable $z \in \{0,1\}$.

Exact k-out-of-n

For Exact k-out-of-n functions, a tighter lower bound is given by:

Theorem 3 (Boros et al., 2020)

$$\lceil\log(k)\rceil - 1 \quad (15)$$

as stated in Boros et al. (2020). A quadratization for the Exact k-out-of-n function is given by:

For each integer $0 \leq k \leq n$, the function:

$$G_k(x,y,z) = \tfrac{1}{2}A_k(x,y,z)\left(A_k(x,y,z) - 1\right) \quad (13)$$

is a quadratization of the Exact k-out-of-n function $f_{=k}$ using:

$$l = \max(\lceil\log(k)\rceil, \lceil\log(n-k)\rceil) \leq \lceil\log(n)\rceil$$

auxiliary variables, consisting of $y \in \{0,1\}^{l-1}$ and an additional variable $z \in \{0,1\}$.

Parity Functions

According to Corollary 4 in Boros et al. (2020), the Parity function belongs to the class of Even (Odd) support functions. Thus, its lower bound for the number of auxiliary variables is given by:

$\lceil\log(n)\rceil - 1.$

A quadratization technique for Parity functions has been introduced in the same article, providing the following upper bound:

Theorem 3 (Boros et al., 2020)

Let $l = \lceil\log(n)\rceil$. The Parity function $\pi_n(x)$ admits the following quadratizations:

If $niseven, the function$:

$$g_e(x,y) = \left(|x| - n + 2^l - \sum_{i=1}^{l-1} 2^i y_i - 1\right)^2 \quad (16)$$

is a quadratization of $\pi_n(x)$.

If n is odd, the function:

$$g_o(x,y) = \left(|x| - n + 2^l - \sum_{i=1}^{l-1} 2^i y_i\right)^2 \quad (17)$$

is a quadratization of $\pi_n(x)$.

Both $g_e(x,y)$ and $g_o(x,y)$ use $\lceil\log(n)\rceil - 1$ auxiliary variables.

All the above-discussed methods for quadratizing a PBF have only been specifically proven for the special structures. It remains an open question whether the lower bound of a general (not necessarily symmetric) quadratization is also logarithmic, and raises concerns about computational performance due to the large coefficients introduced by reformulations having a logarithmic number of auxiliary variables (Boros et al., 2020). Furthermore, there has been no proof for a better upper bound for general pseudo-Boolean functions than $O(2^{\frac{n}{2}})$ (for Zero-until-k and Even(odd) support functions) proposed by Anthony et al. (2017). Fig. 2 shows the connection between the special classes of PBFs, which were studied by Boros et al. (2020).

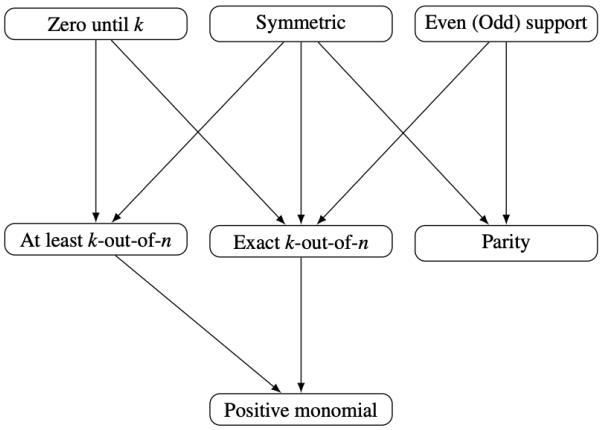

Fig. 2: Relation between the considered classes of functions

Logarithmic Bounds for Positive Monomials

Even though a satisfactory quadratization with only one new auxiliary variable for negative monomials was discovered early on, the case for positive monomials (not necessarily symmetric) is surprisingly more challenging. Traditionally, it required $O(n)$ number of new variables until Boros et al. (2020) claimed that they had found a remarkable logarithmic upper bound for the number of auxiliary variables. This finding was a significant step up from Ishikawa's method, which required $m = \left\lfloor \frac{n-1}{2} \right\rfloor$ variables. Moreover, they have proven that the lower bound for quadratizing a positive monomial is $m = \lceil log(n) \rceil - 1$ number of new variables that exactly matches the upper bound. The bounds were derived from zero until k and even (Odd) support functions. The quadratization for the positive monomials has been established as follows:

Theorem 5 (Boros et al., 2020)

Let $l = \lceil \log(n) \rceil$. Then:

$$g(x,y) = \tfrac{1}{2}\left(|x| + 2^l - n - \textstyle\sum_{i=1}^{l-1} 2^i y_i\right) \times \left(|x| + 2^l - n - \textstyle\sum_{i=1}^{l-1} 2^i y_i - 1\right) \quad (18)$$

is a quadratization of the positive monomial:

$P_n(x) = \prod_{i=1}^{n} x_i$

using $\lceil \log(n) \rceil - 1$ auxiliary variables.

This method could potentially increase the computational efficiency in term-wise quadratizations due to the lower number of auxiliary variables. However, the challenge of using this method to quadratize an arbitrary PBF most efficiently remains a task that depends on the function itself.

Table 1 summarizes the values of the lower and upper bounds described by the compact quadratization methods discussed above. Note that these are not the only quadratization methods proposed for specific problem instances. The earlier discussed methods are noteworthy and significant, as the cited authors have contributed extensively to the study of quadratization.

In the following sections, we will explore some noteworthy alternative quadratization techniques, not necessarily covering broader classes, but approaches quadratization through leveraging other structural characteristics of PBFs.

Table 1: Summary of lower and upper bounds for special function classes used in quadratization

Function	Lower bound	Upper bound
Symmetric	$\boldsymbol{\Omega(\sqrt{n})}$ for some functions	$O(\sqrt{n}) = 2\lceil \sqrt{n} + 1 \rceil$
Zero until k	$\boldsymbol{\Omega(2^{\frac{n}{2}})}$ for some functions $\lceil \log(k) \rceil - 1$ for all functions	$O(2^{\frac{n}{2}})$
At least k-out-of-n	$\lceil \log(k) \rceil - 1$	$\max(\lceil \log(k) \rceil, \lceil \log(n-k) \rceil)$
Even (Odd) support	$\lceil \log(n) \rceil - 1$ for all functions	$O(2^{\frac{n}{2}})$
Exact k-out-of-n	$\lceil \log(n) \rceil - 1$	$\max(\lceil \log(k) \rceil, \lceil \log(n-k) \rceil)$
Parity	$\lceil \log(n) \rceil - 1$	$\lceil \log(n) \rceil - 1$
Positive monomial	$\lceil \log(n) \rceil - 1$	$\lceil \log(n) \rceil - 1$

Algorithmic Quadratization Techniques

The following quadratization techniques take an algorithmic approach by the design of structural methodologies that can optimize specific problem settings more efficiently and take a solver-friendly manner.

Some of the authors have specifically considered QA while developing these algorithms, (Schmidbauer et al., 2024; Verma and Lewis, 2020; Mandal et al., 2020), which is worthwhile to note. Practical experiments on quadratization time (Schmidbauer et al., 2024) comparing the existing and new algorithms have been carried out, making a significant contribution to the field.

Local Structure Reduction (LSR) Method

In the paper by Schmidbauer et al. (2024), a novel method for quadratization, along with research on its time complexity, is introduced. It has been acknowledged in the paper that this transformation is crucial to do in a computationally feasible time to practically use quantum annealing hardware. The paper not only presents a new algorithm but also provides a thorough analysis of its performance, comparing it with a few other traditional methods in terms of quadratization time. Which is a highly noteworthy contribution. The authors propose representing the PBF as a multi-graph, where nodes represent variables, and edges represent interactions between variables and monomials. The multiplicity of an edge between two nodes indicates how many monomials contain that particular pair of variables.

The traditional monomial-based reduction has 3 variants: Sparse, Medium, and Dense. The denser the variable pair selection is, the more compact the QUBO is.

1. Dense: Choosing the variable pair that appears most often among all monomials.
2. Medium: Choosing the variable pair that appears most often among all highest degree monomials.
3. Sparse: Choosing the first variable pair of a monomial with the highest degree.

Algorithm 1 by Schmidbauer et al. (2024) follows two main stages, and it is summarized as follows;

Stage 1: Graph-based Reduction: The algorithm starts by constructing a graph representation of the PBF as said above. We then iteratively select variable pairs with high multiplicity and replace them with auxiliary variables, updating the graph. This method uses a Percentile-based selection strategy to choose which variable pairs to reduce. This allows more control over the density of the resulting QUBO. Having control over the density matters as unnecessary interactions can slow down the annealing process.

Stage 2: Independent Monomial-based Reduction Where monomials that do not share variable pairs with other monomials are handled separately afterward.

The authors provide runtime complexity and experiment results analysis comparing monomial-based term-wise methods and their LSR method. They tested random PBF with degree 4 with varying densities. When it comes to speed-up, they claim that at 39 variables in the input function, the Dense selection type seemingly required more than a day to quadratize a function on traditional methods. While the new algorithm with $q = 1.0$, which is comparable with dense selection, needed about 10 seconds for the same input polynomial. Figure 3 shows the overall experiment's quadratization time against the number of variables for degree 4 PBFs with varying selection (Schmidbauer et al., 2024).

Algorithm 1: Local Structure Reduction (LSR) (Schmidbauer et al., 2024)

Input: PBF f, percentile q *(deg(f) > 2, q ∈ [0,1])*

Output: Quadratic PBF f', penalty PBF p

$h \leftarrow 1$

$p \leftarrow 0$

$G = (V, E), R \leftarrow G_f = (V_f, E_f), R_f$

Stage 1: Graph-based reduction

while G contains multi-edges $(\exists i, j \in V : \beta(i, j) > 1)$ **do**

  $\{i, j\} \leftarrow$ `Choose_Random_Element`$(R(\hat{\beta}_q))$

  $f \leftarrow$ `Replace_Var_Pair`(G, f, x_i, x_j, y_h)

  $G, R \leftarrow$ `Update_Graph_Data`(G, R, i, j, h)

  $p \leftarrow p + p(x_i, x_j, y_h)$

  $h \leftarrow h + 1$

end while

Now we have $V_i, j \in V : \beta(i, j) \leq 1$

Stage 2: Independent monomial-based reduction

for $m \in f$ with $\deg(m) > 2$ **do**

  while $\deg(m) > 2$ **do**

    $f, p \leftarrow$ `Multi_Reduce`(f, p, m)

  end while

end for

return f, p

procedure `Update_Graph_Data`(G, R, i, j, h)

  $Z \leftarrow \{z : (i, j, z) \in E\}$

  $N \leftarrow \{h\}$, $E_{\text{removed}} \leftarrow \{\}$, $E_{\text{added}} \leftarrow \{\}$

  for $z \in Z$ **do**

    for $k \in m_z \wedge k \Vdash h$ **do**

      $N \leftarrow N \cup \{k\}$

      $E_{\text{removed}} \leftarrow E_{\text{removed}} \cup \{(k, i, z)\} \cup \{(k, j, z)\}$

      $E_{\text{added}} \leftarrow E_{\text{added}} \cup \{(k, h, z)\}$

    end for

  end for

  $M \leftarrow \{(\beta(n, k) : n \in N, k \in \{i, j\}\}$

  $E \leftarrow (E \setminus (E_{\text{removed}} \cup E^{i,j})) \cup E_{\text{added}}$

  $R \leftarrow$ `Update_R`(G, R, N, M, i, j, h)

  return G, R

end procedure

DOI: 10.3844/jcssp.2026.309.333

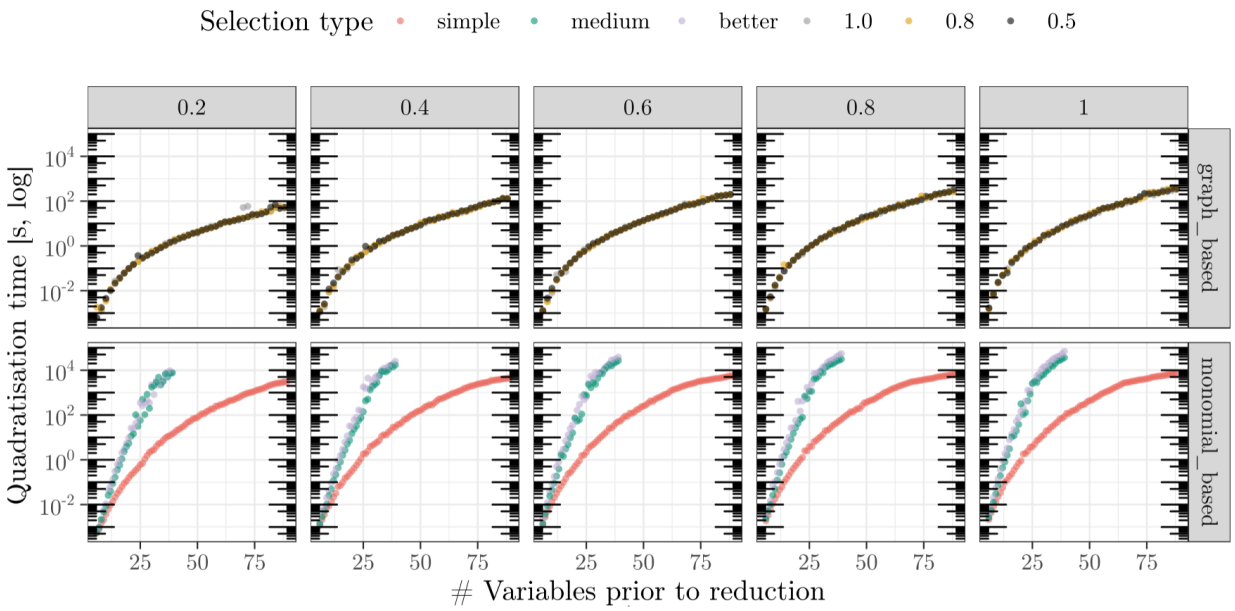

Fig. 3: Time in seconds (log) (y-axis) for the quadratization of a $\deg(f) = 4$ function f vs problem size (x-axis: Number of variables). New LSR algorithm (top row) compared to the existing monomial-based (bottom row)

Quadratization in Ising Space

A method that offers significant improvements for quantum annealing applications (and Ising model solvers overall) has been developed by Mandal et al. (2020), which can directly work in both QUBO and the Ising space as well. Many combinatorial optimization problems in real life, presented as HUBOs, are primarily quadratized by approaches operating in the Binary Space. These methods may not be the most ideal for problems formulated with the Ising space. The Reason is, there exists a necessity for a transformation process from the Ising model to the QUBO model to apply quadratization, which points to a major drawback where a sparse problem in Ising space is not necessarily sparse in Boolean space and vice versa (Mandal et al., 2020).

The authors offer 2 algorithms: a hash table-based one and one using a bipartite graph (both greedy approaches), which can quadratize any HOBO over Boolean or Ising space. Algorithm 2 starts with a hash table that keeps track of all variable pairs and their corresponding expressions. The algorithm repeatedly identifies the most frequently occurring pair and replaces it with a new auxiliary variable in a greedy manner. This process repeats and reduces the complexity of expressions until only simple quadratic terms remain. Finally, to impose the constraints between the auxiliary variables and the quadratic terms, either Binary or Ising polynomials are invoked.

The next algorithm, Algorithm 3 works by constructing a dynamic weight bipartite graph where LHS represents quadratic terms, and the RHS side represents high-degree monomials. The connections between them indicate dependencies. The algorithm then iteratively replaces the LHS vertex with the maximum sum of edge weights with a new auxiliary variable and updates the graph by introducing the quadratic terms involving the new variable in the graph. This process continues until all higher-order expressions are simplified into quadratic terms.

Algorithm 2: HOBO to QUBO Transformation (Mandal et al., 2020)

Input: A higher order binary optimization (HOBO) over $(z_1, z_2, ..., z_n) \in \mathcal{B}^n$ where $\mathcal{B}$ is either $\{-1, 1\}$ or $\{0, 1\}$.

Output: A QUBO equivalent to the given HOBO problem.

1: Sort the variables and create a data structure storing (key, value) pairs where the key is the set of all quadratic terms in HOBO and the value is the set of monomials of degree at least 3.

2: **while** keys remain in the table **do**

3: Select the key with the most associated values and replace it with a new variable.

4: Update the table by adding new keys and values for the new variable.

5: Remove all degree-3 terms that involve the replaced key.

6: Delete the key if all its values have been removed.

7: Store the new variable and its corresponding quadratic term.

8: **end while**

9: Apply the appropriate quadratic transformation to ensure correctness.

10: **return** the QUBO equivalent of the given HOBO problem.

Algorithm 3: HOBO to QUBO 2 (Mandal et al., 2020)

Input: A higher order binary optimization (HOBO) over $(z_1, z_2, ..., z_n) \in \mathcal{B}^n$ where $\mathcal{B}$ is either $\{-1, 1\}$ or $\{0, 1\}$.

Output: A QUBO equivalent to the given HOBO problem.

1: Sort the indices of variables and construct a weighted bipartite graph.

2: Assign all possible quadratic terms as left nodes and all monomials in the HOBO as right nodes.

3: Create edges between left and right nodes if a monomial contains the quadratic term.

4: Set edge weights as the degree of the monomial minus 1.

5: **while** there exists an edge in the graph **do**

6: Replace the quadratic term with the variable having the largest sum of edge weights.

7: Remove all degree-3 terms that involved the quadratic term.

8: Remove quadratic terms that no longer have edges.

9: Update the graph after adding quadratic terms involving the new variable.

10: Store the variable and its corresponding quadratic term in a mapping.

11: **end while**

12: Apply the quadratic polynomial transformation for each auxiliary variable.

13: **return** The QUBO equivalent to the given HOBO problem.

According to their experiment results, the algorithms show an improvement in direct usage on Ising problems than in the usage on transformed (to QUBO) problems. This method is a significant improvement where problems are formulated with the Ising model. The elimination of the unnecessary transformation overhead and saving the sparsity of the original problem should be highlighted with respect to QA applications.

Integer Programming (IP) Approach to Minimize Auxiliary Variable Count

A quadratization approach tailored for fourth-degree pseudo-boolean polynomials has been presented by Verma and Lewis (2020). They offer a preprocessing step to minimize the number of introduced auxiliary variables to the quadratized function based on exact integer programming model. It is worthwhile to note that the authors have taken QUBO formulations specifically used in quantum annealers like D-Wave systems into account while developing the approach.

The same authors Verma et al. (2021) present an optimized transformation tested on cubic-to-quadratic Max 3-SAT problems that reduces both the number of auxiliary variables and the penalty coefficient M. The core idea of this method is to use an Integer programming approach to select the best quadratic substitutions globally from the entire functions, rather than considering each term individually, which helps to determine an optimal set of auxiliary variables to minimize the problem size. Also, the lower bound of the penalty Coefficient M is determined using the nonlinear Constraint Programming. This work is an improvement upon their previous work (Verma and Lewis, 2020). The authors analyze Max 3-SAT problems, formulating at most cubic terms under four cases:

1. No negations $(x_i \vee x_j \vee x_k)$:
$$g(X) = x_i + x_j + x_k - x_ix_j - x_ix_k - x_jx_k + x_ix_jx_k \quad (19)$$
2. One negation $(x_i \vee x_j \vee \neg x_k)$:
$$g(X) = 1 - x_k + x_ix_k + x_jx_k - x_ix_jx_k \quad (20)$$
3. Two negations $(x_i \vee \neg x_j \vee \neg x_k)$:
$$g(X) = 1 - x_jx_k + x_ix_jx_k \quad (21)$$
4. Three negations $(\neg x_i \vee \neg x_j \vee \neg x_k)$:
$$g(X) = 1 - x_ix_jx_k \quad (22)$$

The SAT problems are considered NP-Complete, and if there exists a deterministic polynomial time algorithm for solving SAT, every NP problem can be solved in polynomial time. Therefore, the importance of optimizing the SAT problem should be emphasized in the computer science domain.

The following lemma proposed by the authors are used to quadratize any max 3-SAT problems with minimum additional variables and a minimized penalty coefficient.

Lemma 1

In order to minimize the number of auxiliary variables during the transformation, the quadratic subterm x_ix_j will always be utilized if it dominates concerning the frequency count in all the cubic terms in which it appears, i.e.,

$$(f(x_ix_j) > f(x_ix_k)) \quad \& \quad (f(x_ix_j) > f(x_jx_k))$$
$$\forall (x_ix_j) \subset (x_ix_jx_k)$$

such that $(x_ix_jx_k) \subset C$, where $f()$ represents the function that counts the occurrence of each quadratic subterm across all cubic terms and C denotes the set of cubic terms.

$$M_{LB} = \max\left(\forall x_ix_j = y_{ij}\left(\sum_k a_{ijk}, \quad -\sum_k a_{ijk}\right)\right) \quad (23)$$

The authors aim to transform these cubic terms into quadratic terms using a minimum number of auxiliary variables by solving the integer program proposed in Verma and Lewis (2020) and with the use of Lemma 1. Then the lower bound on the penalty coefficient M is identified using Equation 23. The summarized Algorithm 4 is given below.

The authors claim that computational experiments using SATLIB datasets demonstrated that their approach

achieved a near 100%.

Algorithm 4: Minimal QUBO for Max 3-SAT (Verma et al., 2021)

procedure MinimalQUBO()

1: $n \leftarrow$ number of variables

2: $m \leftarrow$ number of clauses

3: $S \leftarrow$ set of all terms in the objective function

4: $C \leftarrow$ set of all cubic terms in the objective function

5: $IP \leftarrow$ integer program to identify the minimum number of auxiliary variables after applying Lemma 1

6: $LB \leftarrow$ lower bound of penalty coefficient obtained from Equation (1)

7: $Y \leftarrow$ set of auxiliary variables

8: **top**:

9: Convert each clause i into a cubic penalty term $g(X)$

10: Objective Function $obj \leftarrow \sum_{i=1}^{m} g(X)_i$

11: **loop**:

12: **if** (Degree of term i in obj) == 3 **then**

13: $C \leftarrow C \cup (\text{Term } i)$

14: **end if**

15: $Y \leftarrow IP(C)$

16: $M^* \leftarrow LB(C, Y)$ using Equation (1)

17: $Q \leftarrow$ Transformed QUBO using sets C, Y and M^*

18: Optimize Q using Path Relinking Tabu Search routine

19: **end procedure**

Algebraic Quadratization Techniques

In contrast to the algorithmic approaches, the following quadratization methods lean towards being logical and algebraic. However, they still take a different approach or seek to leverage special structural qualities of the PBFs, as in previous methods. Especially in the case where a function has a certain limited degree or a limited number of variables present. Since a larger number of real-world problems can map to a cubic or a quartic setting within a function, the following methods specialize in order reduction of such occurrences, potentially gaining an advantage over arbitrary methods.

Variable Count-Based Quadratization

Dattani and Chau (2019) presents a specialized compact quadratization approach for quadratizing any function of exactly four variables with a minimal overhead of auxiliary variable count. This method relies on substitution and does not introduce penalty functions or a term-wise quadratization. The method provides an exact quadratization framework specifically for functions with four binary variables, claiming that the function can be quadratized using only one auxiliary variable. This approach exploits the structural properties of 4-variable functions to achieve minimal auxiliary count while preserving function equivalence. This can be considered the most compact quadratization for functions in this class. Several important things to note are that this technique gives an exact quadratic reformulation without approximation errors and shows more control over the coefficient ranges compared to penalty-based techniques.

The methodology categorizes functions into four cases depending on the coefficient configuration, considering the ordering and the magnitude. For each of these cases, the paper provides four different lemmas for quadratizations. For a function of binary variables $b_i \in \{0,1\}$ with real-valued coefficients α:

$$\Omega = \alpha_{1234} b_1 b_2 b_3 b_4 + \alpha_{123} b_1 b_2 b_3 + \alpha_{124} b_1 b_2 b_4 + \alpha_{134} b_1 b_3 b_4 + \alpha_{234} b_2 b_3 b_4 \quad (24)$$

Lemma 2

Suppose $\alpha_{1234} \geq 0$. If $\alpha_{ijk} \geq -\frac{\alpha_{1234}}{2}$ for all ijk, or $-\alpha_{1234} \leq \alpha_{123} \leq -\frac{\alpha_{1234}}{2} \leq 0 \leq \alpha_{234} \leq \alpha_{134} \leq \alpha_{124}$, or both $-\alpha_{1234} \leq \alpha_{123} \leq -\frac{\alpha_{1234}}{2} \leq \alpha_{124} \leq -\alpha_{1234}$, then Equation 24 is perfectly quadratized by:

$$\left(3\alpha_{1234} + \sum_{ijk} \alpha_{ijk}\right) b_a + \alpha_{1234} \sum_{ij} b_i b_j + \sum_{k \neq ij} \alpha_{ijk} b_i b_j - \sum_i \left(2\alpha_{1234} + \sum_{jk, i \neq jk} \alpha_{ijk}\right) b_i b_a \quad (25)$$

Lemma 3

If $\alpha_{1234} \leq 0$ and $\alpha_{ijk} \leq 0$, then Equation 24 is perfectly quadratized by:

$$\left(\alpha_{1234} \left(\sum_i b_i - 3\right) + \sum_{ijk} \alpha_{ijk} \left(\sum_{l \in ijk} b_l - 2\right)\right) b_a \quad (26)$$

Lemma 4

If $\alpha_{1234} \geq 0$, $\alpha_{123} \leq -\alpha_{1234}$, and $-\frac{\alpha_{1234}}{2} \leq \alpha_{124} \leq \alpha_{134} \leq 0 \leq \alpha_{234}$, then Equation 24 is perfectly quadratized by:

$$\alpha_{1234} - \sum_i (\alpha_{12i} + \alpha_{1234}) b_i + \sum_i \alpha_{i34} b_a + \sum_{ijk, i,j \neq 1,2} \alpha_{ijk} b_i b_j + \alpha_{1234} b_3 b_4 - \sum_{i=p,q,p,q=1,2 \text{ or } 3,4} \left(\sum_{j=r,s,r,s=3,4 \text{ or } 1,2} \alpha_{pqj} - \alpha_{irs}\right) b_i b_a \quad (27)$$

Lemma 5

If $\alpha_{1234} \geq 0$, $\alpha_{123} \leq -\frac{\alpha_{1234}}{2} \leq \alpha_{124} \leq \alpha_{134} \leq \alpha_{234} \leq 0$, and $\alpha_{123} + \alpha_{124} \leq -\alpha_{1234}$, then Equation 24 is perfectly quadratized by:

$$\alpha_{1234} - \sum_i (\alpha_{12i} + \alpha_{1234}) b_i$$

$$+\sum_i \alpha_{i34} b_a + \sum_{ijk} \alpha_{ijk} b_i b_j + \alpha_{1234} b_3 b_4$$
$$-\sum_{i=p,q,p,q=1,2 \text{ or } 3,4} \left(\sum_{j=r,s,r,s=3,4 \text{ or } 1,2} \alpha_{pqj} - \alpha_{irs}\right) b_i b_a \quad (28)$$
$$-\sum_i \alpha_{12i}(b_1 + b_2) - \sum_i \alpha_{i34}(b_3 + b_4 - 1 - b_i)$$
$$-\alpha_{1234}(b_3 + b_4 - 1) b_a \quad (29)$$

One can broaden the applicability of this approach to any kind of function (with 4 variable subsets), leveraging bit-flipping techniques from Ishikawa (2011), which can transform a function into a suitable form by utilizing another 2 lemmas by Dattani and Chau (2019).

This technique does not seem suitable for larger problems with more variables. Nonetheless, one can apply this method if it is possible to decompose larger functions into multiple 4-variable subsets, which in return could introduce more complexity and redundancy. There can be a function that can be overly complex to decompose or nearly impossible. In such cases, it is far more efficient to use another approach. If a function can be decomposed with reasonable complexity or for small-scale modular functions, this technique can be highly effective. It can be particularly useful for Quantum Annealing hardware for small problems due to the extremely compact quadratization it provides. Citing Dattani and Chau (2019), "Nevertheless, we do not rule out the possibility that other quadratization formulas involving only one auxiliary variable can exist: it may just be that we have not yet found them.", shows great potential for future research in quadratization for specific structures like this.

Quadratization Without Auxiliary Variables

Deduc-Reduc (Deduction-Reduction) Method

The 'Deduc-reduc' method, which was proposed by Tanburn et al. (2015), aims to eliminate the number of 4-qubit and 3-qubit terms in the AQC Hamiltonian without adding auxiliary qubits. This method is highly relevant, especially for QA, as it inherently leverages structural properties in physical problem Hamiltonians and applies logical reductions and substitutions without any cost of added auxiliary variables. Noticeably, it maintains the exact ground state of the original problem as well unlike approximation methods.

The methodology involves identifying variable relationships like $x_1, x_2 = 0$ from higher-order terms like $2x_1x_2x_3x_4$ and prior knowledge of functions such as $x_1 + x_2 + x_3 = 1$, then simplifying by substitution and adding penalty terms, resulting in quadratic (or lower) terms. The general formula for the method is:

$$H'(x) = q(x)g(x) + r(x) + \lambda(f(x) - g(x))^2 \quad (30)$$

Where:

- $H(x) = q(x)f(x) + r(x)$ is the original Hamiltonian.
- $f(x) = g(x)$ is a deduction that holds for all ground states.
- λ is chosen such that $|q(x)| \leq \lambda$ for all states x.

However, identifying deductions cannot always be performed naturally using prior knowledge of the problem; it must be brute-forced. Which in turn can introduce exponential scaling (Dattani, 2019). But in general, this method can technically be used in QUBO problems according to Tanburn et al. (2015). If the method can be implemented in a QA problem context, it will avoid the formulation of denser Hamiltonians and could be easily embedded onto the sparse connectivity of hardware architectures like D-Wave's Chimera or Pegasus graphs.

Split Reduction Method

The split-reduc method (Okada et al., 2015) offers an alternative to deduc-reduc (Dattani, 2019), targeting scenarios where logical deductions are not evident as they are unlikely to appear in every general case. It is good to note that this method has been proposed by the authors specifically to leverage quantum annealing hardware more efficiently. Split-reduction approaches the quadratization by iteratively splitting the Hamiltonian into multiple sub-Hamiltonians to reduce multiple qubit terms. But this method will increase the number of objective functions to find the solution to the original problem. However, adding auxiliary qubits improves the method. This flexibility allows for a trade-off between resource efficiency and computational complexity. Hybrid solvers, for example, D-wave systems, can benefit from the Hamiltonians by handling them separately. Depending on the problem and the hardware, split-reduc provides greater flexibility to the usage of the available quantum hardware.

The split-reduc method relies on two cost functions to guide the splitting process:

Cost Function $\mathcal{C}(\mathcal{H})$

This function determines whether the Hamiltonian H needs further splitting. It is defined as:

$$\mathcal{C}(H) = n + \sum_t \mathcal{R}(\text{order}(t) - 2) \leq Q \quad (31)$$

Where:

- n is the number of original qubits.
- $\mathcal{R}$ is the ramp function, which gives the maximum number of auxiliary qubits needed to reduce a term t to quadratic order.
- Q is the qubit capacity of the quantum device.

If $\mathcal{C}(H) \leq Q$, the Hamiltonian can be implemented on the device without further splitting.

Cost Function $C_H(x_i)$

This function determines which variable x_i to split next. It is defined as:

$$C_H(x_i) = \sum_t [I_{x_i,t} \cdot \mathcal{R}(\text{order}(t) - 2 + 1)] \quad (32)$$

Where:

- $I_{x_i,t}$ is an indicator function that is 1 if x_i appears in term t, and 0 otherwise.
- $\mathcal{R}(\text{order}(t) - 2)$ is the maximum number of auxiliary qubits needed to reduce term t to quadratic order.

The variable x_i with the highest $C_H(x_i)$ is chosen for splitting, as it appears in the most higher-order terms.

The authors provide theoretical calculations and estimates for Ramsey number experiment, which, by using their method (Okada et al., 2015), can greatly reduce the runtime and minimize the extra qubit usage compared to the original experiments run on quantum annealers (Bian et al., 2013).

ELC Quadratization Method

Excludable Local Configurations quadratization method first introduced by Ishikawa (2014) for reducing higher-order terms in Markov Random Field (MRF) at the cost of zero auxiliary variables. This method works by identifying excludable local configurations in the energy/cost function. An ELC is a specific assignment of values to a subset of variables that cannot be part of the global minimizer. By modifying the function to increase the energy for these configurations, the higher-order terms can be eliminated without affecting the global minimum.

Below is the modification of the energy function to eliminate higher-order terms:

$$E'(x) = E(x) + |\alpha_C| \prod_{i \in C} (u_i x_i + (1 - u_i)(1 - x_i)) \quad (33)$$

Where:

- $E(x)$ is the original energy function.
- α_C is the coefficient of the higher-order term involving the variables in clique $\mathcal{C}$.
- u_i is the value of the ELC for variable x_i.
- $\psi(x) = |\alpha_C| \prod_{i \in C} (u_i x_i + (1 - u_i)(1 - x_i))$ is the added term that increases the energy only when the variables take the values specified by the ELC u.

A major drawback of this method is the fact that there are no known methods to find an ELC; hence, even Ishikawa (2014) uses a brute force approach, which can scale exponentially with the number of variables. Since ELCs do not always exist in every PBF so this method is not suitable for any arbitrary case.

Table 2: Comparative Summary of Quadratization Techniques: Auxiliary Variable Bounds, Submodularity, Use-Case Suitability, and Limitations

Method	Auxiliary Variables (Bound)	Submodular	Where It's Advantageous	Limitations & Comments
Rosenberg Substitution	Up to $(n-2)$	No	General-purpose for arbitrary PBFs; foundational and easy to implement	Large penalty terms; not efficient for large problems; produces dense QUBOs
Freedman-Drineas (NTR)	1 per term	Yes	Negative monomials; optimal for preserving submodularity	Limited to negative monomials; not applicable to general PBFs
Ishikawa (PTR)	$\lfloor \frac{n-1}{2} \rfloor$	No	Positive monomials	Still non-submodular; may lead to explosion in quadratic terms
Symmetric Functions	$\lceil \sqrt{n} + 1 \rceil$	Depends	Useful for symmetric PBFs like exact-k, parity, at-least-k	Requires structure-specific handling; not general-purpose
Logarithmic PTR	$\lceil \log(n) \rceil - 1$	No	Positive monomials; tighter bounds than Ishikawa's	Submodularity not preserved; structural assumptions needed
Local Structure Reduction (LSR)	Variable	Depends	Faster in quadratizing; dedicated for QUBO	May be less impactful on smaller problems
Ising-Space Quadratization	$O(n)$	Depends	Suitable for native Ising Hamiltonians (better than converting to QUBO); avoids Boolean conversion	Greedy approach; still requires many auxiliary variables in worst case
IP Optimization	Empirically minimized ($O(n)$)	Depends	Minimize auxiliary variable overhead; very successful in Max 3-SAT problems	Only a pre-processing step; not tested on general cases
4-variable Algebraic Method	1	Depends	Highly efficient for easily decomposable 4-variable functions	Only applies to size-4 subsets; not scalable
Deduc-Reduc Method	0	Depends	Advantageous when logical constraints are identifiable	Requires deductive rules or brute-force; not always available
Split-Reduc Method	0 (many sub-Hamiltonians)	Depends	Ideal when hardware constraints limit ancilla budget	Increases number of problems to solve; can be computationally heavy depending on resources
ELC Method	0	Depends	Suitable for MRFs or sparse structured PBFs	Needs forbidden config detection; brute-force may be required

Gadget-Based Quadratization and Gadgets

Classical quadratization methods, such as Rosenberg's substitution and Ishikawa's method, focus on combinatorial optimization problems involving pseudo-Boolean functions, where these techniques are universal across various domains and solvers-classical and quantum. Gadget-based quadratization methods offer fundamentally different approaches that are specifically suitable for quantum systems. They have been designed to preserve quantum and physical properties such as the energy spectrum that is native to quantum systems (physics simulations, quantum chemistry, etc). These quadratization methods operate within Ising space or with the Pauli representation. The goal of these techniques is to ensure the same ground state of the higher-order function in the quadratic case. This goal is approached by introducing auxiliary qubits and enforcing constraints through energy penalties. The following section will only introduce some of the techniques in the literature. Dattani (2019) has done an in-depth comparative analysis of the Hamiltonian-based techniques as well as all the other classical quadratization methods.

Perturbative Gadgets

Perturbative gadgets can approximate higher-order interactions by the process of introducing auxiliary qubits with specific energy penalties. These methods are considered to be more efficient but may cause errors. Several research on the usage of Perturbative gadgets has been carried out (Chen, 2011; Kempe et al., 2006; Oliveira and Terhal, 2005; Bravyi et al., 2008; Jordan and Farhi, 2008; Cao et al., 2015).

(3 → 2) Gadgets

Transforming 3-local interactions into 2-local interactions using auxiliary qubits and perturbative expansions.

1-By-1 Gadgets

This approach steps-by-step reduces the order of the formulated Hamiltonian, decreasing the locality, one order at each step.

Subdivision Gadgets

Reducing higher-order terms by dividing them into further smaller terms in a step.

Direct Gadgets

This method avoids iterative reduction and instead, performs the quadratization in a single step.

Non-Perturbative Gadgets

Unlike Perturbative gadgets, Non-Perturbative gadgets ensure exact quadratization without approximations. These methods require a relatively higher number of auxiliary variables. But they are more suitable for high-precision applications.

Ocko and Yoshida (2011); Subaşı and Jarzynski (2016); Nagaj (2010, 2012) have proposed a few different techniques for different Hamiltonian forms. Some of these techniques leverage additional resources beyond qubits, such as qutrits and ququits.

Discussion

Computational Aspects

Computational Overhead

While quadratization facilitates compatibility with current quantum hardware, it introduces computational overhead, affecting the efficiency of solving and transformation. Most research on quadratization focuses on transformation techniques and their theoretical properties, but only a few address the computational impact of this transformation. While some studies have compared their novel approaches to some traditional methods (Schmidbauer et al., 2024), a comprehensive comparison of computation overhead among the prominent and most used techniques for different problems is lacking.

One of those studies that examine computational overhead in solvability after quadratization is Valiante et al. (2021), which provides a detailed empirical evaluation of the effects of locality reduction. The following content of this chapter is based on this work. Using the publicly available `Chook` package, authors generated degree-3 and degree-4 benchmark problems and tested them on Microsoft Azure Quantum's k-local solvers. They compared native k-local problems with their quadratized 2-local versions.

Quadratization increased variable count by 3x for 3-local and 6x for 4-local problems. Interestingly, density slightly decreased post-reduction. However, the Time-to-Solution (TTS) increased significantly-quadratized problems often failed to solve even with a 5x timeout, unlike the original k-local instances, which had 100% success. Due to the inability to find a proper scaling, the authors surmise that the higher the locality is, the higher the overhead in solving the 2-local reductions will be.

They also found broader, less symmetric coupler distributions in quadratized problems, increasing coefficient variability and complicating the energy landscape, making them harder for quantum annealers to solve efficiently (Figure 4). The benchmark experiment has been performed with two different values of the parameter `timeout`. The data show that solving the 2-local versions of the problems is extremely difficult. They were unable to do a scaling analysis as the majority of the problems could not be solved.

DOI: 10.3844/jcssp.2026.309.333

Schmidbauer et al. (2024) compares their LSR method's computational time against standard term-wise methods. They observed that their approach greatly reduces the time taken to quadratize a certain function in the tested scenarios. Unfortunately, this kind of comparison regarding computational overhead is largely lacking in this domain.

In conclusion, the current state-of-the-art quadratization methods introduce unnecessary, highly increased overhead in the computational aspect of solving the problem, hinting that newer quadratization techniques should be studied that will overcome or minimize this negative effect.

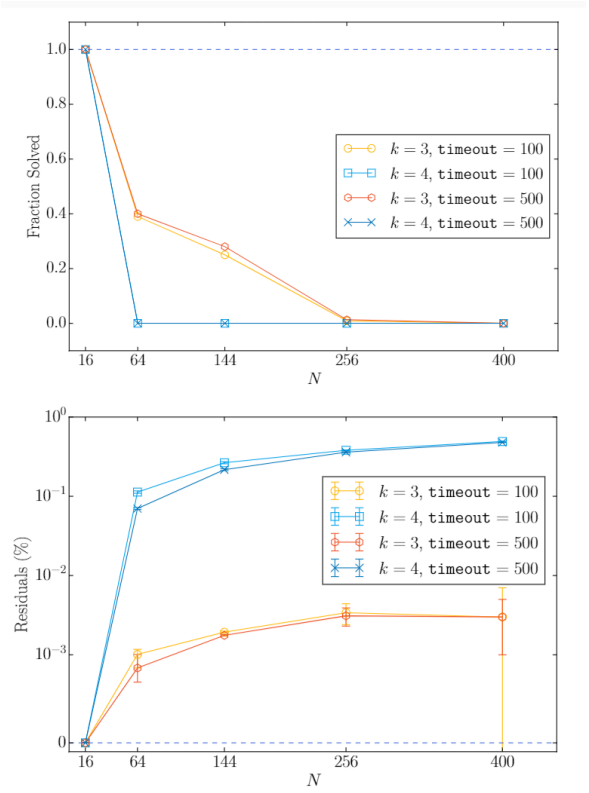

Fig. 4: Fraction solved (top) and residuals (bottom) of 2-local problems obtained by reducing k-local instances with k = 3 and k = 4. The dashed line represents the reference for the ideal cases.

Algorithm Implementations

There are a number of algorithmic implementations made using different programming languages for the automation of quadratization, with adjustable parameters for the strength of quadratization.

Ocean

The leading quantum annealing company, D-Wave, handles automatic quadratization for any HUBO formulations through their `make\_quadratic()` function from the `dimod` package provided by the Ocean software. It takes a polynomial representation (HUBO), a strength parameter to control constraint enforcement, and optionally a variable type (BINARY or SPIN for Ising Model) and an existing Binary Quadratic Model (BQM) to modify. According to the Ocean software (D-WaveReformulation, 2024) documentations, they have implemented two approaches based on Reduction by Substitution and Reduction by Minimum Selection.

PyQUBO

A Python library, which allows users to create and work with QUBO or Ising models. It's compatible with D-Wave Ocean, hence it can be integrated with its workflow. The `compile()` function will detect any higher-order interactions and automatically reduce them to quadratic terms. According to Zaman et al. (2021), the quadratization technique used is the penalty-based substitution method suggested by Rosenberg.

QUBOVert

Another Python library that supports formulating and solving QUBO and Ising model problems. They handle automatic quadratization through their `to\_qubo()` function if it were applied to any problem with higher order terms. Although the exact method is not provided in their documentation, it's mentioned that quadratization is done by adding auxiliary variables.

QUBO.jl

A library implemented with Julia for working with QUBO and Ising models. `ToQUBO.jl` provides an interface for automated quadratization. By leveraging Julia's multiple dispatch paradigm, users are allowed to implement their quadratization methods as well. According to Xavier et al. (2023), currently, two methods have been implemented, using PTR and NTR.

Critical Analysis

We have reviewed a broad selection of works other than specifically on quadratization, including approximately 35 papers on quantum and classical problem solving (which utilized any kind of quadratization in the process). Table 3 depicts the real-world examples for research and each quadratization approach that was chosen or referred to in the studies. Although this is a non-exhaustive list, it captures the essence of the usage ratio of different methods. Note that some recent methods we covered from previous sections are not available because we were unable to find any usage. The search was carried out through the Google Scholar search engine. This section is dedicated to presenting insights by analyzing the usage of quadratization observed across these studies. The problems in these studies belong to different domains such as quantum physics, mathematics, chemistry, scheduling, computing, and theoretical studies. According to our selection criteria, we only chose

DOI: 10.3844/jcssp.2026.309.333

problems where quadratization was needed as a part of the problem-solving process.

Method Usage Trends

The evolution of quadratization techniques reveals a shift from general-purpose transformations (e.g., Rosenberg's method, Ishikawa's, Freedman's formulation) toward more problem-specific and hardware-efficient methods (Mandal et al., 2020; Okada et al., 2015). While we can gradually see the new methods have been referred to by particularly newer works like Bishwas et al. (2024) and Couzinié et al. (2024), it is evident that traditional methods still dominate the usage. (We will refer to Rosenberg (1975)'s reduction method, Ishikawa (2011)'s PTR method, and Freedman and Drineas (2005); Kolmogorov and Zabih (2004)'s NTR method as the traditional methods.) Figure 5 depicts the methods that have been used to solve problems where quadratization was needed and performed.

Table 3: References to Quadratization Methods in Selected Papers (Non-Exhaustive)

Quadratization Method	Referenced Papers
Penalty Based Rosenberg's Method (Rosenberg, 1975)	Jiang et al. (2018); Chang et al. (2019); Cruz-Santos et al. (2019); Jones et al. (2020); Mahasinghe et al. (2021); Arya et al. (2022); Salehi et al. (2022); Mahasinghe and Jayasinghe (2022); Mosca and Verschoor (2022); Mato et al. (2022); Domino et al. (2022); Jun and Lee (2023a); Jun and Lee (2023b); Fernández-Villaverde and Hull (2023); Sharma et al. (2023); Gilbert et al. (2023); Pichugina et al. (2024); Troy (2024); Wronski and Dzierzkowski (2024); Dobrynin et al. (2024); Grange et al. (2024)
PTR & NTR (Ishikawa, 2011; Freedman & Drineas, 2005)	Li (2016); Cruz-Santos et al. (2018); Sharma et al. (2023); Fernández-Villaverde and Hull (2023); Pichugina et al. (2024); Key and Freinberger (2024); Dobrynin et al. (2024)
`dimod.make_quadratic()` (D-Wave, 2024)	Mengoni et al. (2020); Copenhaver et al. (2020); Pelofske et al. (2022); Djidjev (2023); Dukalski et al. (2023); Uotila (2024); Uotila (2025)
Quadratization in Ising Space (Mandal et al., 2020)	Domino et al. (2022); Bishwas et al. (2024)
Deduc-Reduc Method (Tanburn et al., 2015)	Couzinié et al. (2024)
PyQUBO	Kaseb et al. (2024)
Qubovert	Chermoshentsev et al. (2021)

We observe that some papers did not perform actual quadratization but instead suggested possible methods in case quadratization was required. Additionally, we noticed that in relatively recent papers, even when quadratization was not applied, alternative methods beyond the traditional ones were considered. Figure 6 shows the overall proportions of methods referred to in studies. As quadratization is often overlooked, observing what methods researchers are aware of extracts useful information for our particular study.

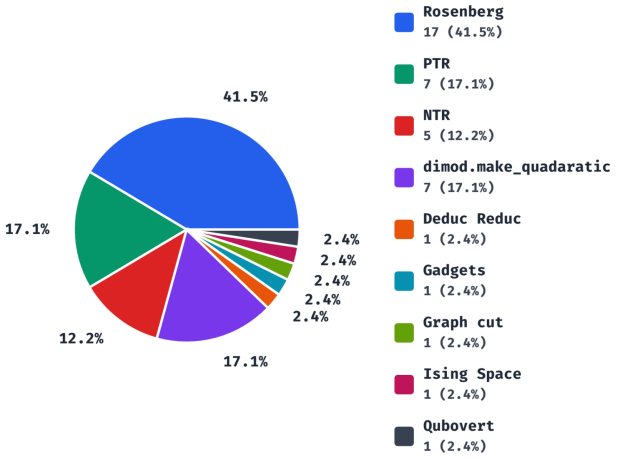

Fig. 5: Overall actual usages of known quadratization from the reviewed works

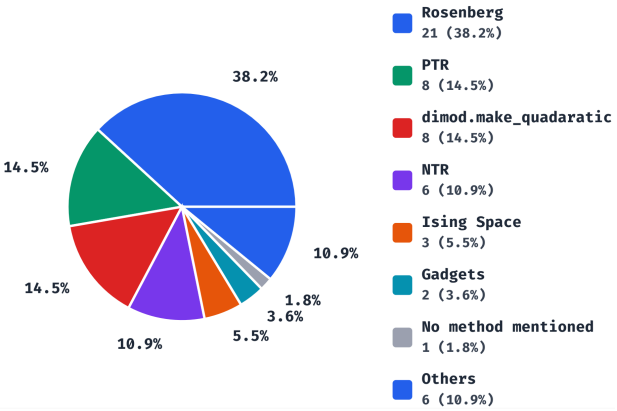

Fig. 6: Overall mentions of known quadratization methods from the reviewed works

Figure 5 shows that from 2018 to 2025, traditional quadratization methods have remained dominant. Despite the improved logarithmic bound presented by Boros et al. (2020), we found no instances of its use in the surveyed works. The Ocean software's `make\_quadratic` function was commonly used, but it relies on classic substitution and reduction techniques, reinforcing the continued use of traditional approaches. We also noted the use of other open-source tools in Julia and Python, though they similarly default to well-established methods.

Most studies do not justify their choice of quadratization method. The only clear rationale we observed was hardware compatibility, such as with D-Wave's Pegasus topology (Fernández-Villaverde and Hull, 2023). This lack of justification may arise from limited awareness of newer methods or a perception that

quadratization is peripheral to the main research goals. Only a few acknowledged the potential of more compact techniques, but still none applied them. Some authors even noted that expensive quadratization can negate potential quantum advantage (Uotila, 2025). Out of 35 studies using only traditional methods, Figure 7 compares those that mentioned alternative techniques, though without implementing them.

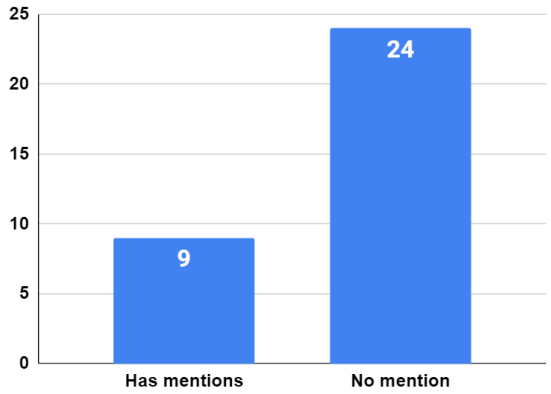

Fig. 7: Percentage of papers that acknowledged other existing techniques than the traditional ones

This suggests that within the research community, these fore-called traditional methods have become the de facto standard for quadratization, often applied without formal acknowledgment.

Domain-Specific Preferences

We were unable to extract clear insights on quadratization method usage across different domains, largely due to the lack of rationale provided for method selection. This made it difficult to associate specific methods with specific problem types.

That said, a few studies in quantum chemistry avoided manual traditional methods by using automated tools like Qubovert, `make\_quadratic`, or referring to newer techniques such as deduc-reduc and Ising-space quadratization (Copenhaver et al., 2020; Couzinié et al., 2024; Chermoshentsev et al., 2021; Bishwas et al., 2024). However, this data is insufficient to establish any domain-specific trends. Most other studies in quantum physics, mathematics, and computing continued to rely on traditional methods.

Experimental vs Theoretical Usage

Table 4 compresses the usage of different quadratization methods from the instances we analyzed within theoretical and experimental studies. What we can observe through this is that in practical works, researchers have utilized automated tools for order reduction needs, possibly due to the ease of use and convenience. We can see that non-traditional methods have been considered in both scenarios. Many other approaches could have been experimented with in theoretical studies, such as Split-Reduction and 4-variable quadratization studies carried out by Okada et al. (2015); Dattani (2019).

Table 4: Usage of quadratization methods in theoretical and practical aspects

Practical	Theoretical
PTR	PTR
NTR	NTR
Rosenberg	Rosenberg
dimod.make_quadratic	Ising Space
Deduc Reduc	Gadgets
Qubovert	-
PyQUBO	-
DADK	-
Ising Space	-
Gadgets	-
Symmetric Functions	-

Lack of Usage of Recent Approaches

As seen in the usage trends analyzed earlier, many recent quadratization techniques, particularly those leveraging structural properties or novel formulations-remain unused in practical problem-solving. This raises important questions about their performance across different hardware platforms and problem types. Papers rarely perform multiple experiments to evaluate alternatives (Jones et al., 2020). It remains unclear whether better results could have been achieved had more recent approaches, such as those in Dattani and Chau (2019); Verma and Lewis (2020), been tested.

Lack of Computational Comparisons and Practical Implementations in Quadratization

There is a clear lack of systematic comparisons across problem types, and few automated tools offer support for multiple techniques. This scarcity can discourage users who are less familiar with the underlying mathematics. Given the commercial availability of QA devices like D-Wave, accessible and flexible implementations are essential for broader adoption.

Several studies (Mengoni et al., 2020; Chermoshentsev et al., 2021; Kaseb et al., 2024) rely on automatic quadratization via standard libraries, typically using Minimum Selection or Substitution. However, these default choices are not optimal for all problem types, especially when the number of auxiliary variables becomes a performance concern. Tools that support the Ising model could benefit from integrating Ising-space quadratization, which avoids the need for intermediate QUBO transformation.

Quadratization Techniques in Real-Life

From the previous observations, we will now analyze the real-world applicability, a comparative evaluation, and the efficiency of the discussed methods.

Out of the work that has been reviewed, we categorize only the following to have actual adequate usages in real-life problem-solving cases ;

- Rosenberg's Substitution
- NTR - Freedman & Drineas
- PTR - Ishikawa

The above methods have proven to be useful, although with questionable impact on the performance. General applicability justifies their wide usage. Note that the Make Quadratic method is omitted since it uses the above 3 methods under the abstractions.

Even though Rosenberg (1975)'s method provides clear quadratization, Ishikawa (2011) claims that due to the large coefficient M, the method becomes impractical to use in real cases. Even though there is no better bound for the number of introduced variables in term-wise methods such as Ishikawa's PTR, with the combination of NTR, and because of smaller coefficients (quantum annealers' limited hardware precision to specify the values of Ising model parameters desires this), the claim is that the substitution methods could be outperformed. Verma et al. (2021)'s preprocessing techniques to minimize the coefficient and the number of auxiliary variables introduced could potentially lead the Rosenberg's substitution to outperform its traditional form. But we have yet to see any proper generalized application for this (besides Max 3-SAT instances).

While some studies prove that term-wise methods are applicable in real quantum calculations, such as prime factorization (Jiang et al., 2018), the minimum multi-cut problem (Cruz-Santos et al., 2019), eigenvalue problem (Jun and Lee, 2023b) in relatively smaller-scale with the correct bounds, some (Key and Freinberger, 2024; Copenhaver et al., 2020) raise a problem of quadratization affecting the quality of the solution due to the decrease in problem sparsity. Ultimately, limiting the scale of solvable problems is a major downside of these traditional methods.

The same observation can be seen with Rosenberg's method as well. Studies in quantum annealing, like the discrete logarithm problem (Mahasinghe and Jayasinghe, 2022), NP problems (Mahasinghe et al., 2021), Machine-Learning (Troy, 2024), music composition (Arya et al., 2022), molecular unfolding (Mato et al., 2022), etc., have chosen Rosenberg due to its algorithmic convenience, and have even raised concerns about the growing variable count as well. While the methods work with smaller-scale problems, the performance loss increases rapidly with the increasing degree. Gilbert et al. (2023) claims that these processing steps multiply the required physical qubits by 581% for 4th degree problems in D-Wave systems. Jones et al. (2020) refers to the Rosenberg method as the 'most general but worst scaling method'. These scaling problems could potentially be addressed by the non-general methods that are mentioned below.

Negative Term Reduction introduced by Freedman and Drineas (2005) remains the best technique to be performed on negative monomials, which also preserves the submodularity. Also, we found no example for the usage of the 'better' logarithmic bound PTR reduction method over Ishikawa's, hinting that we currently have no proper solid claims for the comparative effectiveness of the substitution and term-wise methods. As a summary, the advancement of substitution remains limitedly tested, and the term-wise advancements remain theoretical.

We see a concerning drop in quality after quadratization of larger-scale problems. This even led to researchers proposing Higher-Order Ising solver models as well (Bybee et al., 2023). At this current stage, quantum annealing would become infeasible for natural higher-order problem solving due to this bottleneck.

We observe the following methods as promising alternatives or improvements, but underutilized approaches that could potentially address the above issues in scaling. They have not yet gained much attention to provide enough data to determine their efficiency in real use cases.

- Ising Space Quadratization
- Deduc-Reduc
- Split-Reduc
- Local Structure Reduction (LSR)
- 4-Variable Function Quadratization

All the above-mentioned methods are promising candidates for specialized quantum annealing scenarios. Support for Ising problems, elimination of auxiliary variables, and fast optimized quadratization are highly useful in resource-sensitive workflows.

However, Couzinié et al. (2024) showed that by using the deduc-reduc method, they were not able to completely quadratize the problem, thus making it unable to be directly run on quantum hardware. This depicts that while these new methods are promising in certain aspects, they might ultimately fail to deliver satisfying results by themselves. Unfortunately for the rest of the aforementioned methods, we could find no applications.

In comparison to deduc-reduc, split-reduc provides greater versatility but at the cost of increased computational overhead with the generation of multiple separate Hamiltonians. Deduc-reduc is more efficient with problems that have inherent variable relationships. Split-reduc is far more suitable for problems lacking such structures, making it a complementary approach. It would be nice to see actual benchmarking results to solidify these insights.

As the authors claim (Mandal et al., 2020), the most optimal method that should be chosen for Ising problems would be their approach. However, the majority of the practical experiments are carried out in the QUBO if not

binary domain. Still, case studies exist (Xia et al., 2017) that converted Ising problems to the binary domain, resulting in an exponential number of auxiliary variables through the Rosenberg method. A future direction would be to re-evaluate these using the Ising quadratization method.

The LSR method is suitable for larger-scale problems over standard methods due to its lower computational overhead. While it achieves comparable results in terms of new variables to state-of-the-art methods, it can be made more flexible and can handle quadratizing larger problems. This work by Schmidbauer et al. (2024) is one of the rarest studies that ran actual comparative experiments on methods.

Since these methods carry more limitations than the general-purpose ones, they can be combined and used in a hybrid manner that will provide more optimal results.

Recent Advancements that Influence the Effectiveness of Quadratization

Some recent breakthroughs in quantum computing bear an influence on the effectiveness and the requirement of quadratization. For example, D-Wave has announced their latest 'Advantage2' Quantum Computer (D-Wave Systems Inc., 2024) with 4400+ qubits with a newer qubit connectivity topology 'Zephyr'. Zephyr is a significant advancement over their previous topologies, Pegasus and Chimera. It features qubits of degree 20 and native K_4 and $K_8, 8$ subgraphs (Boothby et al., 2021). This breakthrough may reduce the need to apply heuristics for the selection of optimal quadratization, and the traditional methods could be more applicable in general case scenarios due to the higher qubit count and a better coupling technology (Volpe et al., 2025)

Another approach to solving optimization problems is the Quantum Approximate Optimization Algorithm (QAOA) (Farhi et al., 2014; Bennett et al., 2024). This leverages the universal gate-based model and is continuously being developed with hybrid variational algorithms. This way of optimization vastly reduces the need for quadratization, as QAOA natively supports higher-order functions (Campbell and Dahl, 2022).

Conclusion

This survey has reviewed the landscape of quadratization methods used in converting higher-order pseudo-Boolean functions into quadratic form, a crucial step for enabling optimization via quantum annealing and many other applications. We examined both traditional and modern techniques, including general-purpose reductions, structure-specific methods, and pre-processing for compact quadratizations with provable variable bounds. While advances have been made, a clear gap remains between the rich body of introduced novel methods and their practical deployment.

A key observation from our Critical Analysis is that despite the numerous array of quadratization methods proposed over the decades, most practical works still rely on a small handful of classical techniques, especially Rosenberg's, Ishikawa's, and Freedman's methods, often without any stated justification. This uncritical selection could be deemed problematic, particularly in quantum applications where inefficient quadratizations can severely hinder performance due to limited qubit counts and embedding challenges. Furthermore, our analysis revealed that researchers seldom compare multiple methods for the same problem, and virtually no empirical rationale is found to address the choice of one method over another.

This lack of methodological rigor underlines the need for a unified framework for quadratization method selection. Such a framework should consider structural properties of the pseudo-Boolean function (symmetry, submodularity, variable count, order), hardware constraints, and performance trade-offs. Our future work aims to address this gap through systematic experimentation on known and new problem instances using underexplored quadratization techniques. This would not only allow comparative benchmarking but could also uncover context-specific heuristics or rules-of-thumb.

We also propose the potential in using AI to assist the quadratization process, not to replace existing methods (as there exists no large datasets of problem-quadratization-performance to train on) but to help guide their selection or tuning. While still in early stages, AI could support tasks like identifying function structure or recommending suitable methods based on prior examples. As a future direction, quantum annealing can also be incorporated with solving continuous functions through Integer programming and discretizations, which would be highly beneficial, other than solving combinatorial optimizations. Another promising direction would be to combine these methods and use them in a planned hybrid manner in a way that the methods complement each other's weaknesses. From a mathematical perspective, more compact and efficient quadratization techniques are in dire need. Although this direction might not seem easily tackled, we still have not overcome the bottleneck caused by this transformation.

Finally, current software tools (such as D-Wave's Ocean SDK) default to legacy methods and do not yet incorporate compact or structure-aware quadratizations. Enhancing these tools to support a broader range of methods, as well as pre-processing steps that could reduce the resulting function complexity, would be a significant step forward in making quantum annealing more and accessible.

In conclusion, quadratization remains a bottleneck in realizing the full potential of quantum annealing. To bridge the gap between theory and practice, a data-driven

approach is necessary-one that acknowledges the diversity of methods available and selects or designs them accordingly.

Acknowledgment

Geethika Prabhath acknowledges the support received from the Quantum Computing Student Network Sri Lanka via Colombo Quantum Computing Meetup 2025.

Funding Information

The authors acknowledge the financial support received from the University of Colombo School of Computing.

Authors Contributions

Geethika Prabhath: Writing - Original Draft, Formal Analysis, Visualization, Writing - Review & Editing.

Anuradha Mahasinghe: Supervision, Conceptualization, Writing - Review & Editing.

Nalin Ranasinghe: Supervision, Conceptualization, Writing - Review & Editing.

Kasun De Zoysa: Supervision, Writing - Review & Editing.

Ethics

The authors declare that they have no ethical issues to report regarding the present study.

References

Albash, T., & Lidar, D. A. (2018). Adiabatic quantum computation. *Reviews of Modern Physics*, *90*(1), 015002. https://doi.org/10.1103/revmodphys.90.015002

Anthony, M., Boros, E., Crama, Y., & Gruber, A. (2016). Quadratization of symmetric pseudo-Boolean functions. *Discrete Applied Mathematics*, *203*, 1-12. https://doi.org/10.1016/j.dam.2016.01.001

Anthony, M., Boros, E., Crama, Y., & Gruber, A. (2017). Quadratic reformulations of nonlinear binary optimization problems. *Mathematical Programming*, *162*(1-2), 115-144. https://doi.org/10.1007/s10107-016-1032-4

Apolloni, B., Cesa-Bianchi, N., & Falco, D. (1990). A numerical implementation of "quantum annealing". *Stochastic Processes, Physics and Geometry: Proceedings of the Ascona-Locarno Conference*, 97-111.

Aramon, M., Rosenberg, G., Valiante, E., Miyazawa, T., Tamura, H., & Katzgraber, H. G. (2019). Physics-Inspired Optimization for Quadratic Unconstrained Problems Using a Digital Annealer. *Frontiers in Physics*, *7*, 00048. https://doi.org/10.3389/fphy.2019.00048

Arya, A., Botelho, L., Cañete, F., Kapadia, D., & Salehi, Ö. (2022). Music Composition Using Quantum Annealing. *Quantum Physics*. https://doi.org/10.48550/arXiv.2201.10

Bennett, T., Noakes, L., & Wang, J. (2024). Non-Variational Quantum Combinatorial Optimisation. *Proceeding of the International Conference on Quantum Computing and Engineering (QCE)*, 31-41. https://doi.org/10.1109/qce60285.2024.00014

Bian, Z., Chudak, F., Macready, W. G., Clark, L., & Gaitan, F. (2013). Experimental Determination of Ramsey Numbers. *Physical Review Letters*, *111*(13), 130505. https://doi.org/10.1103/physrevlett.111.130505

Bishwas, A. K., Pitchai, A., & Som, A. (2024). Molecular unfolding formulation with enhanced quantum annealing approach. https://doi.org/10.48550/arXiv.2403.00507

Boothby, K., Bunyk, P., Raymond, J., & Roy, A. (2020). Next-generation topology of D-Wave quantum processors. *Quantum Physics*. https://doi.org/10.48550/arXiv.2003.00133

Boothby, K., King, A. D., & Raymond, J. (2021). *Zephyr topology of D-Wave quantum processors*. https://doi.org/10.48550/arXiv.2003.00133

Boros, E., Crama, Y., & Rodríguez-Heck, E. (2020). Compact quadratizations for pseudo-Boolean functions. *Journal of Combinatorial Optimization*, *39*(3), 687-707. https://doi.org/10.1007/s10878-019-00511-0

Boros, E., & Gruber, A. (2014). On quadratization of pseudo-boolean functions. *Optimization and Control*. https://doi.org/10.48550/arXiv.1404.6538

Boros, E., & Hammer, P. L. (2002). PseudoBoolean optimization. *Discrete Applied Mathematics*, *123*(1-3), 218 01 00341-9. https://doi.org/10.1016/S0166-218X(01)00341-9

Bravyi, S., DiVincenzo, D. P., Loss, D., & Terhal, B. M. (2008). Quantum Simulation of Many-Body Hamiltonians Using Perturbation Theory with Bounded-Strength Interactions. *Physical Review Letters*, *101*(7), 070503. https://doi.org/10.1103/physrevlett.101.070503

Bybee, C., Kleyko, D., Nikonov, D. E., Khosrowshahi, A., Olshausen, B. A., & Sommer, F. T. (2023). Efficient optimization with higher-order Ising machines. *Nature Communications*, *14*(1), 6033. https://doi.org/10.1038/s41467-023-41214-9

Calude, C. S., Dinneen, M. J., & Hua, R. (2017). QUBO formulations for the graph isomorphism problem and related problems. *Theoretical Computer Science*, *701*, 54-69. https://doi.org/10.1016/j.tcs.2017.04.016

Campbell, C., & Dahl, E. (2022). QAOA of the Highest Order. *Proceeding of the International Conference on Software Architecture Companion (ICSA-C)*, 141-146. https://doi.org/10.1109/icsa-c54293.2022.00035

Cao, Y., Babbush, R., Biamonte, J., & Kais, S. (2015). Hamiltonian gadgets with reduced resource requirements. *Physical Review A*, *91*(1), 012315. https://doi.org/10.1103/physreva.91.012315

Carugno, C., Ferrari Dacrema, M., & Cremonesi, P. (2022). Evaluating the job shop scheduling problem on a D-wave quantum annealer. *Scientific Reports*, *12*(1), 6539. https://doi.org/10.1038/s41598-022-10169-0

Chang, T. H., Lux, T. C. H., & Tipirneni, S. S. (2019). Least-squares solutions to polynomial systems of equations with quantum annealing. *Quantum Information Processing*, *18*(12), 374. https://doi.org/10.1007/s11128-019-2489-x

Chermoshentsev, D. A., Malyshev, A. O., Esencan, M., Tiunov, E. S., Mendoza, D., Aspuru-Guzik, A., Fedorov, A. K., & Lvovsky, A. I. (2021). Polynomial unconstrained binary optimisation inspired by optical simulation. *Quantum Physics*. https://doi.org/10.48550/arXiv.2106.13167

Codognet, P. (2021). Constraint Solving by Quantum Annealing. *50th International Conference on Parallel Processing Workshop*, 1-10. https://doi.org/10.1145/3458744.3473364

Copenhaver, J., Wasserman, A., & Wehefritz-Kaufmann, Birgit. (2020). Using quantum annealers to calculate ground state properties of molecules. *The Journal of Chemical Physics*, *152*(2), 024107. https://doi.org/10.1063/1.5123532

Couzinié, Y., Nishiya, Y., Nishi, H., Kosugi, T., Nishimori, H., & Matsushita, Y. (2024). Annealing for prediction of grand canonical crystal structures: Implementation of n-body atomic interactions. *Physical Review A*, *109*(3). https://doi.org/10.1103/physreva.109.032416

Croke, S. (2015). PT-symmetric Hamiltonians and their application in quantum information. *Physical Review A*, *91*(5), 052113. https://doi.org/10.1103/physreva.91.052113

Cruz-Santos, W., Venegas-Andraca, S. E., & Lanzagorta, M. (2018). A QUBO Formulation of the Stereo Matching Problem for D-Wave Quantum Annealers. *Entropy*, *20*(10), 786. https://doi.org/10.3390/e20100786

Cruz-Santos, W., Venegas-Andraca, S. E., & Lanzagorta, M. (2019). A QUBO Formulation of Minimum Multicut Problem Instances in Trees for D-Wave Quantum Annealers. *Scientific Reports*, *9*(1), 17216. https://doi.org/10.1038/s41598-019-53585-5

Dattani, N. (2019). *Quadratization in discrete optimization and quantum mechanics*. https://doi.org/10.48550/arXiv.1901.04405

Dattani, N., & Chau, H. T. (2019). All 4- variable functions can be perfectly quadratized with only 1 auxiliary variable. *Discrete Mathematics*. https://doi.org/10.48550/arXiv.1910.13583

Deutsch, David Elieser. (1989). Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, *425*(1868), 73-90. https://doi.org/10.1098/rspa.1989.0099

Dickson, N. G., Johnson, M. W., Amin, M. H., Harris, R., Altomare, F., Berkley, A. J., Bunyk, P., Cai, J., Chapple, E. M., Chavez, P., Cioata, F., Cirip, T., deBuen, P., Drew-Brook, M., Enderud, C., Gildert, S., Hamze, F., Hilton, J. P., Hoskinson, E., +… Rose, G. (2013). Thermally assisted quantum annealing of a 16-qubit problem. *Nature Communications*, *4*(1), 1903. https://doi.org/10.1038/ncomms2920

Ding, J., Spallitta, G., & Sebastiani, R. (2024). Effective prime factorization via quantum annealing by modular locally-structured embedding. *Scientific Reports*, *14*(1), 3518. https://doi.org/10.1038/s41598-024-53708-7

Dinneen, M. J., Mahasinghe, A., & Liu, K. (2019). Finding the chromatic sums of graphs using a D-Wave quantum computer. *The Journal of Supercomputing*, *75*(8), 4811-4828. https://doi.org/10.1007/s11227-019-02761-5

Djidjev, H. N. (2023). Quantum Annealing with Inequality Constraints: The Set Cover Problem. *Advanced Quantum Technologies*, *6*(11), 2300104. https://doi.org/10.1002/qute.202300104

Dobrynin, D., Renaudineau, A., Hizzani, M., Strukov, D., Mohseni, M., & Strachan, J. P. (2024). Energy landscapes of combinatorial optimization in Ising machines. *Physical Review E*, *110*(4), 045308. https://doi.org/10.1103/physreve.110.045308

Domino, K., Kundu, A., Salehi, Ö., & Krawiec, K. (2022). Quadratic and higher-order unconstrained binary optimization of railway rescheduling for quantum computing. *Quantum Information Processing*, *21*(9), 337. https://doi.org/10.1007/s11128-022-03670-y

Duan, Q.-H., & Chen, P.-X. (2011). Realization of universal adiabatic quantum computation with fewer physical resources. *Physical Review A*, *84*(4). https://doi.org/10.1103/physreva.84.042332

Dukalski, M., Rovetta, D., van der Linde, S., Möller, M., Neumann, N., & Phillipson, F. (2023). Quantum computer-assisted global optimization in geophysics illustrated with stack-power maximization for refraction residual statics estimation. *GEOPHYSICS*, *88*(2), V75-V91. https://doi.org/10.1190/geo2022-0253.1

D-Wave Systems Inc. (2024). *Advantage2 Prototype: A Leap Forward in Quantum Computing*.

D-WaveConcepts. (2024). Ocean software documentation - concepts. *Ocean SDK Documentation*.

D-WaveReformulation. (2024). Reformulating a problem for the D-Wave system. *D-Wave System Documentation*.

D-WaveSystems. (2024). *Getting started with the D-Wave system*.

Farhi, E., Goldstone, J., & Gutmann, S. (2014). A quantum approximate optimization algorithm. https://doi.org/10.48550/arXiv.1411.4028

Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A., & Preda, D. (2001). A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem. *Science*, *292*(5516), 472-475. https://doi.org/10.1126/science.1057726

Farhi, E., Goldstone, J., Gutmann, S., & Sipser, M. (2000). Quantum computation by adiabatic evolution. *Quantum Physics*. https://doi.org/10.48550/arXiv.quant-ph/0001106

Fernández-Villaverde, J., & Hull, I. (2023). Dynamic Programming on a Quantum Annealer: Solving the RBC Model. *SSRN Electronic Journal*, 49. https://doi.org/10.2139/ssrn.4475924

Freedman, D., & Drineas, P. (2005). Energy Minimization via Graph Cuts: Settling What is Possible. *Proceeding of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 939-946. https://doi.org/10.1109/cvpr.2005.143

Gilbert, V., Rodriguez, J., Louise, S., & Sirdey, R. (2023). Solving Higher Order Binary Optimization Problems on NISQ Devices: Experiments and Limitations. *Computational Science - ICCS 2023*, *13983*, 224-232. https://doi.org/10.1007/978-3-031-36030-5_18

Glover, F. (1975). Improved Linear Integer Programming Formulations of Nonlinear Integer Problems. *Management Science*, *22*(4), 455-460. https://doi.org/10.1287/mnsc.22.4.455

Grange, C., Lavignac, M., Pozzoli, V., & Bourreau, E. (2024). *Quadratic versus polynomial unconstrained binary models for quantum optimization illustrated on railway timetabling*. https://doi.org/10.48550/arXiv.2411.10062

Hashagen, A.-L. K. (2018). *Symmetry methods in quantum information theory*.

Hua, R., Di Lorenzo, D., Chinesta, F., & Codognet, P. (2024). Quantum Annealing Solutions for Drone Route Planning Problems. *Proceeding of the International Conference on Quantum Computing and Engineering (QCE)*, 600-610. https://doi.org/10.1109/qce60285.2024.00076

Ishikawa, H. (2011). Transformation of General Binary MRF Minimization to the First-Order Case. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(6), 1234-1249. https://doi.org/10.1109/tpami.2010.91

Ishikawa, H. (2014). Higher-Order Clique Reduction without Auxiliary Variables. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 1362-1369. https://doi.org/10.1109/cvpr.2014.177

Iwata, S., Fleischer, L., & Fujishige, S. (2001). A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, *48*(4), 761-777. https://doi.org/10.1145/502090.502096

Jiang, S., Britt, K. A., McCaskey, A. J., Humble, T. S., & Kais, S. (2018). Quantum Annealing for Prime Factorization. *Scientific Reports*, *8*(1), 17667. https://doi.org/10.1038/s41598-018-36058-z

Johnson, M. W., Amin, M. H. S., Gildert, S., Lanting, T., Hamze, F., Dickson, N., Harris, R., Berkley, A. J., Johansson, J., Bunyk, P., Chapple, E. M., Enderud, C., Hilton, J. P., Karimi, K., Ladizinsky, E., Ladizinsky, N., Oh, T., Perminov, I., Rich, C., … Rose, G. (2011). Quantum annealing with manufactured spins. *Nature*, *473*(7346), 194-198. https://doi.org/10.1038/nature10012

Jones, E. B., Graf, P., Kapit, E., & Jones, W. (2020). K-spin Hamiltonian for quantum-resolvable Markov decision processes. *Quantum Machine Intelligence*, *2*(2), 12. https://doi.org/10.1007/s42484-020-00026-6

Jordan, S. P., & Farhi, E. (2008). Perturbative gadgets at arbitrary orders. *Physical Review A*, *77*(6), 062329. https://doi.org/10.1103/physreva.77.062329

Jun, K., & Lee, H. (2023a). HUBO and QUBO models for prime factorization. *Scientific Reports*, *13*(1), 10080. https://doi.org/10.1038/s41598-023-36813-x

Jun, K., & Lee, H. (2023b). HUBO formulations for solving the eigenvalue problem. *Results in Control and Optimization*, *11*, 100222. https://doi.org/10.1016/j.rico.2023.100222

Kadowaki, T., & Nishimori, H. (1998). Quantum annealing in the transverse Ising model. *Physical Review E*, *58*(5), 5355-5363. https://doi.org/10.1103/physreve.58.5355

Kapit, E., & Oganesyan, V. (2021). Noise-tolerant quantum speedups in quantum annealing without fine tuning. *Quantum Science and Technology*, *6*(2), 025013. https://doi.org/10.1088/2058-9565/abd59a

Kaseb, Z., Möller, M., Vergara, P. P., & Palensky, P. (2024). Power flow analysis using quantum and digital annealers: a discrete combinatorial optimization approach. *Scientific Reports*, *14*(1), 23216. https://doi.org/10.1038/s41598-024-73512-7

Kempe, J., Kitaev, A., & Regev, O. (2006). The Complexity of the Local Hamiltonian Problem. *SIAM Journal on Computing*, *35*(5), 1070-1097. https://doi.org/10.1137/s0097539704445226

Key, F., & Freinberger, L. (2024). A Formulation of Structural Design Optimization Problems for Quantum Annealing. *Mathematics*, *12*(3), 482. https://doi.org/10.3390/math12030482

Kochenberger, G. A., & Glover, F. (2006). *A Unified Framework for Modeling and Solving Combinatorial Optimization Problems: A Tutorial*. *82*, 101-124. https://doi.org/10.1007/0-387-29550-x_4

Kochenberger, G., Hao, J.-K., Glover, F., Lewis, M., Lü, Z., Wang, H., & Wang, Y. (2014). The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, *28*(1), 58-81. https://doi.org/10.1007/s10878-014-9734-0

Kolmogorov, V., & Zabih, R. (2004). What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(2), 147-159. https://doi.org/10.1109/tpami.2004.1262177

Lanting, T., Przybysz, A. J., Smirnov, A. Yu., Spedalieri, F. M., Amin, M. H., Berkley, A. J., Harris, R., Altomare, F., Boixo, S., Bunyk, P., Dickson, N., Enderud, C., Hilton, J. P., Hoskinson, E., Johnson, M. W., Ladizinsky, E., Ladizinsky, N., Neufeld, R., Oh, T., … Rose, G. (2014). Entanglement in a Quantum Annealing Processor. *Physical Review X*, *4*(2), 021041. https://doi.org/10.1103/physrevx.4.021041

Li, M. (2016). *Guaranteed Parameter Estimation of Discrete Energy Minimization for 3D Scene Parsing*.

Louveaux, Q., & Buchheim, C. (2018). Linear and quadratic reformulations of nonlinear optimization problems in binary variables. *4OR - A Quarterly Journal of Operations Research*, *17*, 221-222. https://doi.org/10.1007/s10288-018-0392-4

Lucas, A. (2014). Ising formulations of many NP problems. *Frontiers in Physics*, *2*, 00005. https://doi.org/10.3389/fphy.2014.00005

Mahasinghe, A., Fernando, V., & Samarawickrama, P. (2021). QUBO formulations of three NP problems. *Journal of Information and Optimization Sciences*, *42*(7), 1625-1648. https://doi.org/10.1080/02522667.2021.1930657

Mahasinghe, A., Hua, R., Dinneen, M. J., & Goyal, R. (2019). Solving the Hamiltonian Cycle Problem using a Quantum Computer. *Proceedings of the Australasian Computer Science Week Multiconference*, 1-9. https://doi.org/10.1145/3290688.3290703

Mahasinghe, A., Izaac, J. A., Wang, J. B., & Wijerathna, J. K. (2015). Phase-modified CTQW unable to distinguish strongly regular graphs efficiently. *Journal of Physics A: Mathematical and Theoretical*, *48*(26), 265301. https://doi.org/10.1088/1751-8113/48/26/265301

Mahasinghe, A., & Jayasinghe, Y. (2022). An initial step toward a quantum annealing approach to the discrete logarithm problem. *SECURITY AND PRIVACY*, *5*(4), e234. https://doi.org/10.1002/spy2.234

Mahasinghe, A., Wang, J. B., & Wijerathna, J. K. (2014). Quantum walk-based search and symmetries in graphs. *Journal of Physics A: Mathematical and Theoretical*, *47*(50), 505301. https://doi.org/10.1088/1751-8113/47/50/505301

Mandal, A., Roy, A., Upadhyay, S., & Ushijima-Mwesigwa, H. (2020). Compressed Quadratization of Higher Order Binary Optimization Problems. *Proceeding of the Data Compression Conference (DCC)*, 383. https://doi.org/10.1109/dcc47342.2020.00090

Martonak, R., Santoro, G. E., & Tosatti, E. (2004). Quantum annealing of the travelingsalesman problem. *Phys. Rev. E*, *70*(057701), 057701. https://doi.org/10.1103/PhysRevE.70.057701

Mato, K., Mengoni, R., Ottaviani, D., & Palermo, G. (2022). Quantum molecular unfolding. *Quantum Science and Technology*, *7*(3), 035020. https://doi.org/10.1088/2058-9565/ac73af

Mengoni, R., Ottaviani, D., & Iorio, P. (2020). Breaking RSA security with a low noise D-Wave 2000Q quantum annealer: Computational times, limitations and prospects. *arXiv*. https://doi.org/10.48550/arXiv.2005.02268

Mosca, M., & Verschoor, S. R. (2022). Factoring semi-primes with (quantum) SAT-solvers. *Scientific Reports*, *12*(1), 7752. https://doi.org/10.1038/s41598-022-11687-7

Nagaj, D. (2010). Fast universal quantum computation with railroad-switch local Hamiltonians. *Journal of Mathematical Physics*, *51*(6), 062331. https://doi.org/10.1063/1.3384661

Nagaj, D. (2012). Universal two-body-Hamiltonian quantum computing. *Physical Review A*, *85*(3), 032330. https://doi.org/10.1103/physreva.85.032330

Ocko, S. A., & Yoshida, B. (2011). Nonperturbative Gadget for Topological Quantum Codes. *Physical Review Letters*, *107*(25), 250502. https://doi.org/10.1103/physrevlett.107.250502

Okada, E., Tanburn, R., & Dattani, N. (2015). Reducing multi-qubit interactions in adiabatic quantum computation without adding auxiliary qubits. part 2: The "split-reduc" method and its application to quantum determination of ramsey numbers. *Https://Arxiv.Org/Abs/1508.04816*.

Oliveira, R., & Terhal, B. M. (2008). The complexity of quantum spin systems on a two-dimensional square lattice. *Quantum Information and Computation*, *8*(10), 900-924. https://doi.org/10.26421/qic8.10-2

Pelofske, E., Hahn, G., O'Malley, D., Djidjev, H. N., & Alexandrov, B. S. (2022). Quantum annealing algorithms for Boolean tensor networks. *Scientific Reports*, *12*(1), 8539. https://doi.org/10.1038/s41598-022-12611-9

Pichugina, O., Tan, Y., & Beck, C. (2024). Deriving compact QUBO models via multilevel constraint transformation. *Online Documentation*. https://doi.org/10.48550/arXiv.2404.03610

Rajak, A., Suzuki, S., Dutta, A., & Chakrabarti, B. K. (2023). Quantum annealing: an overview. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *381*(2241), 20210417. https://doi.org/10.1098/rsta.2021.0417

Rosenberg, I. G. (1975). Reduction of bivalent maximization to the quadratic case. *Cahiers Du Centre d'Etudes de Recherche Operationnelle* , *17*, 71-74.

Salehi, Ö., Glos, A., & Miszczak, J. A. (2022). Unconstrained binary models of the travelling salesman problem variants for quantum optimization. *Quantum Information Processing*, *21*(2), 67. https://doi.org/10.1007/s11128-021-03405-5

Sao, M., Watanabe, H., Musha, Y., & Utsunomiya, A. (2019). Application of digital annealer for faster combinatorial optimization. *Fujitsu Scientific and Technical Journal*, *55*(2), 45-51.

Schmidbauer, L., Lobe, E., Schaefer, I., & Mauerer, W. (2024). It's quick to be square: Fast quadratisation for quantum toolchains. *Quantum Physics*. https://doi.org/10.48550/arXiv.2411.19934

Schrijver, A. (2000). A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time. *Journal of Combinatorial Theory, Series B*, *80*(2), 346-355. https://doi.org/10.1006/jctb.2000.1989

Sharma, A., Burns, M., & Huang, M.-C. (2023). Combining cubic dynamical solvers with make/break 27 Prabhath et al / Journal of Computer Science2025, 14 (3): Page Numbers DOI: heuristics to solve SAT. *26th International Conference on Theory and Applications of Satisfiability*, 25-38.

Subaşı, Y., & Jarzynski, C. (2016). Nonperturbative embedding for highly nonlocal Hamiltonians. *Physical Review A*, *94*(1), 012342. https://doi.org/10.1103/physreva.94.012342

Tanburn, R., Okada, E., & Dattani, N. (2015). Reducing multi-qubit interactions in adiabatic quantum computation without adding auxiliary qubits. part 1: The "deduc-reduc" method and its application to quantum factorization of numbers. *arXiv*. https://doi.org/10.48550/arXiv.1508.04816

Troy, W. (2024). Sparks of quantum advantage and rapid retraining in machine learning. *Quantum Physics*. https://doi.org/10.48550/arXiv.2407.16020

Uotila, V. (2024). Tensor Decompositions and Adiabatic Quantum Computing for Discovering Practical Matrix Multiplication Algorithms. *Proceeding of the International Conference on Quantum Computing and Engineering (QCE)*, 390-401. https://doi.org/10.1109/qce60285.2024.00053

Uotila, V. (2025). Left-deep join order selection with higher-order unconstrained binary optimization on quantum computers. *Frontiers in Computer Science*. https://doi.org/10.3389/fcomp.2025.1649354

Valiante, E., Hernandez, M., Barzegar, A., & Katzgraber, H. G. (2021). Computational overhead of locality reduction in binary optimization problems. *Computer Physics Communications*, *269*, 108102. https://doi.org/10.1016/j.cpc.2021.108102

Verma, A., & Lewis, M. (2020). Optimal quadratic reformulations of fourth degree Pseudo-Boolean functions. *Optimization Letters*, *14*(6), 1557-1569. https://doi.org/10.1007/s11590-019-01460-7

Verma, A., Lewis, M., & Kochenberger, G. (2021). Efficient QUBO transformation for higher degree pseudo Boolean functions. *Optimization and Control*. https://doi.org/10.48550/arXiv.2107.11695

Volpe, D., Orlandi, G., & Turvani, G. (2025). Improving the Solving of Optimization Problems: A Comprehensive Review of Quantum Approaches. *Quantum Reports*, *7*(1), 3. https://doi.org/10.3390/quantum7010003

Wronski, M., & Dzierzkowski, L. (2024). Base of exponent representation matters -- more efficient reduction of discrete logarithm problem and elliptic curve discrete logarithm problem to the QUBO problem. *Quantum Information and Computation*, *24*(7 & 8), 541-564. https://doi.org/10.26421/qic24.7-8-1

Xavier, P. M., Ripper, P., Andrade, T., Garcia, J. D., Maculan, N., & Neira, D. E. B. (2023). QUBO.jl: A julia ecosystem for quadratic unconstrained binary optimization. *Optimization and Control*. https://doi.org/10.48550/arXiv.2307.02577

Yarkoni, S., Raponi, E., Bäck, T., & Schmitt, S. (2022). Quantum annealing for industry applications: introduction and review. *Reports on Progress in Physics*, *85*(10), 104001. https://doi.org/10.1088/1361-6633/ac8c54

Zaman, M., Tanahashi, K., & Tanaka, S. (2022). PyQUBO: Python Library for Mapping Combinatorial Optimization Problems to QUBO Form. *IEEE Transactions on Computers*, *71*(4), 838-850. https://doi.org/10.1109/tc.2021.3063618

Zielinski, S., Nüßlein, J., Stein, J., Gabor, T., Linnhoff-Popien, C., & Feld, S. (2023). Pattern QUBOs: Algorithmic Construction of 3SAT-to-QUBO Transformations. *Electronics*, *12*(16), 3492. https://doi.org/10.3390/electronics12163492