Decomposed Quadratization: Efficient QUBO Formulation for Learning Bayesian Network

Yuta Shikuri
Tokio Marine Holdings, Inc.
Tokyo, Japan
shikuriyuta@gmail.com

Abstract

Algorithms and hardware for solving quadratic unconstrained binary optimization (QUBO) problems have made significant recent progress. This advancement has focused attention on formulating combinatorial optimization problems as quadratic polynomials. To improve the performance of solving large QUBO problems, it is essential to minimize the number of binary variables used in the objective function. In this paper, we propose a QUBO formulation that offers a bit capacity advantage over conventional quadratization techniques. As a key application, this formulation significantly reduces the number of binary variables required for score-based Bayesian network structure learning. Experimental results on 16 instances, ranging from 37 to 223 variables, demonstrate that our approach requires fewer binary variables than quadratization by orders of magnitude. Moreover, an annealing machine that implement our formulation have outperformed existing algorithms in score maximization.

1 Introduction

A Bayesian network is a probabilistic graphical model that represents a joint probability distribution among random variables in a directed acyclic graph (DAG) (Pearl 1988). One class of associated computational problems is learning the graph structure of a Bayesian network from data. There are two principal approaches to Bayesian network structure learning: constraint-based and score-based (Kitson et al. 2023). Constraint-based algorithms construct graphs through conditional independence tests. Score-based approaches aim to find a DAG with the highest possible score. In this study, we focus on score-based approaches.

Bayesian network structure learning is NP-hard (Chickering, Heckerman, and Meek 2003). Many approaches have been proposed to improve accuracy and reduce execution time. For a small number of variables, some algorithms such as (Cussens 2011) are capable of identifying a DAG with the maximum score. For high-dimensional data, the standard methodology involves the use of approximate approaches. The hill climbing search method over the space of DAGs (Bouckaert 1994) remains competitive despite its simplicity. Approximate searches over the space of topological ordering (Teyssier and Koller 2005) are also well known as competitive algorithms.

Quadratic unconstrained binary optimization (QUBO) is capable of modeling a wide range of combinatorial optimization problems, including image synthesis in the field of computer vision (Kolmogorov and Rother 2007) and various machine learning tasks such as neural networks (Sasdelli and Chin 2021) and decision trees (Yawata et al. 2022). QUBO can also be applied to score-based Bayesian network structure learning (O'Gorman et al. 2014). Note that constraint-based algorithms do not appear to be suitable for QUBO formulation. Recent developments in annealing machines have garnered significant attention for QUBO formulation. An annealing machine is a specialized hardware architecture designed to heuristically solve QUBOs (Yamamoto 2020). In particular, annealing machines based on semiconductor technology derived from simulated annealing have reportedly outperformed existing solvers in maximum cut problems (Matsubara et al. 2020).

The challenge of setting up a QUBO lies in reformulating the objective function into a quadratic polynomial using fewer binary variables. The greater the number of binary variables in a QUBO, the more difficult it generally becomes to solve. More practically, encoding a large number of binary variables onto the hardware circuit of an annealing machine is often infeasible. Quadratization is the common method of QUBO formulation (Heck 2018; Anthony et al. 2016; Boros, Crama, and Heck 2019). Algorithms have been developed to minimize the number of auxiliary variables for quadratization (Verma and Lewis 2020). However, quadratization still requires too many auxiliary variables for certain tasks, including Bayesian network structure learning. Consequently, there is a potential demand for a more efficient QUBO formulation method as an alternative to conventional quadratization.

In this study, we propose a QUBO formulation specifically designed to use fewer binary variables than quadratization. We demonstrate that this formulation offers significant advantages for learning Bayesian networks in terms of bit capacity. Figure 1 illustrates the overview of our approach. Experimental results on 16 instances, ranging from 37 to 223 variables, show that our formulation dramatically reduces the number of binary variables compared to (O'Gorman et al. 2014). These instances can be encoded on the circuit of

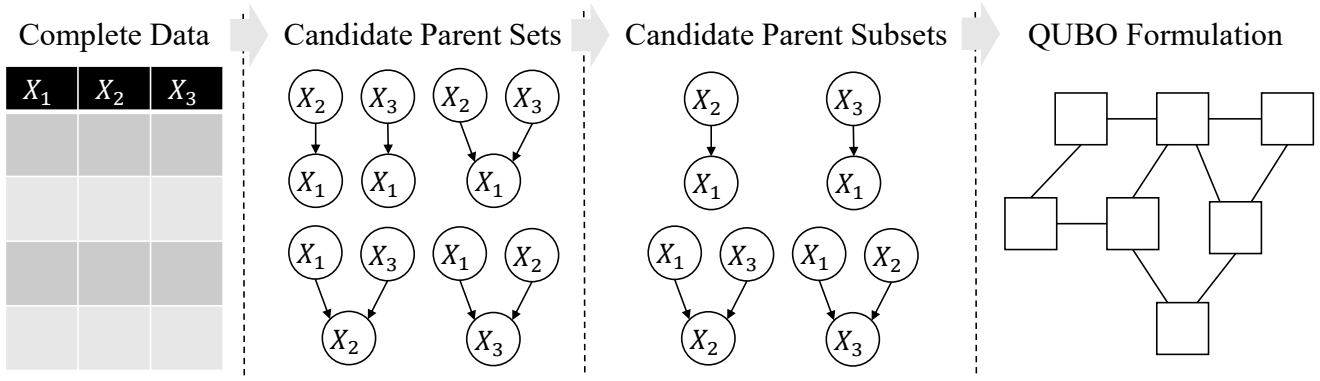

Figure 1: Overview of our approach to Bayesian network structure learning. We introduce the concept of candidate parent subsets to represent candidate parent sets as a quadratic polynomial.

the Fujitsu Digital Annealer, a fully-coupled annealing machine with 100K bit capacity (Nakayama et al. 2021). The scores achieved by the Digital Annealer using our formulation were greater than those of existing approaches.

2 Preliminary

We describe a QUBO formulation for learning Bayesian networks, incorporating candidate parent sets into the approach of (O'Gorman et al. 2014). For indexed symbols, we assign the zero index to the empty set and ensure no overlap (e.g., $W_{i0} = \emptyset$ and $\{W_{ij}\}_{j=0}^{\lambda_i}$ includes $\lambda_i + 1$ elements). Appendix A provides the list of symbols.

2.1 Bayesian Network Structure Learning

A Bayesian network, which is a graphical model composed of a DAG and its parameters, represents a joint probability distribution. Each vertex of the DAG corresponds to a random variable. Edges and parameters characterize conditional probability distributions. Let $\mathcal{X} \equiv \{X_i\}_{i=1}^n$ denote the set of vertices, and Π_i denote the set of vertices with edges directed towards X_i. A topological ordering of a graph exists if and only if the graph is a DAG (Cormen et al. 1990).

Definition 2.1. A topological ordering $\prec$ is a binary relation between any two of vertices such that

- $X_i \prec X_j$ and $X_j \prec X_k \Rightarrow X_i \prec X_k$.
- $X_i \prec X_j \Rightarrow X_i \notin \Pi_j$.

The goal of score-based Bayesian network structure learning is to find a DAG with the maximum score. A score is the sum of local scores which only rely on the parent set of each vertex. Given complete data, the parent sets $\Pi_1, \cdots, \Pi_n$ are optimized to maximize the score under the constraint that the graph $\mathcal{G}$ corresponding to them is a DAG. This optimization is formulated as

$$\text{maximize} \sum_{1 \le i \le n} \log S_i(\Pi_i) \ \text{ subject to } \mathcal{G} \text{ is a DAG}, \quad (1)$$

where $S_i : 2^{\mathcal{X} \setminus \{X_i\}} \to \mathbb{R}$. For simplicity, we take $S_i(W) = S_i(\emptyset)$ when $|W|$ is greater than the maximum parent set size m. The identification of candidate parent sets facilitates narrowing the search space by the relation between parent sets and local scores (de Campos and Ji 2010).

Definition 2.2. The candidate parent sets of X_i are defined as $\{W \subseteq \mathcal{X} \setminus \{X_i\} \mid W' \subset W \Rightarrow S_i(W') < S_i(W)\}$.

Let $\{W_{ij}\}_{j=0}^{\lambda_i}$ denote the candidate parent sets of X_i, and $\mathcal{X}_i$ denote the union of them. Using the candidate parent sets, we can ignore the topological ordering of certain edges that do not produce cycles. Let $\mathcal{E}$ denote the set of edges on possible cycles.

2.2 QUBO Formulation

QUBO is a mathematical optimization problem of minimizing a quadratic polynomial with binary variables. A pseudo-Boolean function maps binary-valued inputs to a real value. Every pseudo-Boolean function can be uniquely represented as a multilinear polynomial given by

$$f(\boldsymbol{p}) \equiv \sum_{V \in \mathcal{F}} \pi(V) \prod_{i \in V} p_i, \quad (2)$$

where $\boldsymbol{p} \equiv (p_i)_{i=1}^I \in \{0,1\}^I$, $\pi : 2^{\{1,\cdots,I\}} \to \mathbb{R} \setminus \{0\}$, $\mathcal{F} \subseteq 2^{\{1,\cdots,I\}} \setminus \{\emptyset\}$, and we ignore the constant term. Quadratization is a major technique to convert a higher degree pseudo-Boolean function into a quadratic one using auxiliary variables $\boldsymbol{q} \equiv (q_i)_{i=1}^J \in \{0,1\}^J$.

Definition 2.3. The quadratization of f is a quadratic polynomial function $g : \{0,1\}^I \times \{0,1\}^J \to \mathbb{R}$ such that

$$f(\boldsymbol{p}) = \min_{\boldsymbol{q} \in \{0,1\}^J} g(\boldsymbol{p}, \boldsymbol{q}). \quad (3)$$

Rosenberg 1975 has proven that every pseudo-Boolean function can be transformed into a quadratic polynomial through the substitution of the product of two variables with an auxiliary variable and the addition of a penalty term.

2.3 QUBO Formulation for Structure Learning

Bayesian network learning can be uniquely represented by a multilinear polynomial over the state of edges and topological ordering. The state of $\boldsymbol{d} \equiv ((d_{ij})_{j=1}^n)_{i=1}^n$ is mapped

one-to-one to the set of edges, where $d_{ij} = 1$ if X_j is a parent of X_i; otherwise, $d_{ij} = 0$. The score component is

$$O(\boldsymbol{d}) \equiv \sum_{1\le i\le n} \sum_{W\subseteq \mathcal{X}_i} \pi_i(W) \prod_{1\le j\le n; X_j\in W} d_{ij}, \quad (4)$$

where $\pi_i(W) \equiv \sum_{W'\subseteq W}(-1)^{1+|W\setminus W'|}\log S_i(W')$. The higher-degree terms in the score component can be transformed into quadratic terms through quadratization. The state of $\boldsymbol{r} \equiv (r_{ij})_{(i,j)\in\mathcal{E}}$ corresponds to the topological ordering, where $r_{ij} = 0$ if the order of X_j is higher than X_i; otherwise, $r_{ij} = 1$. The linear ordering of vertices and the consistency of edges in definition 2.1 are captured by

$$C(\boldsymbol{d},\boldsymbol{r}) \equiv \sum_{(i,j),(j,k),(i,k)\in\mathcal{E}} \delta_1 R(r_{ij}, r_{jk}, r_{ik}) + \sum_{(i,j)\in\mathcal{E}} \delta_2\big(d_{ij}r_{ij} + d_{ji}(1-r_{ij})\big), \quad (5)$$

where $\delta_1, \delta_2 \in (0,\infty)$ and $R(r_{ij}, r_{jk}, r_{ik}) \equiv r_{ik} + r_{ij}r_{jk} - r_{ij}r_{ik} - r_{jk}r_{ik}$. There exist (δ_1, δ_2) such that $C(\boldsymbol{d},\boldsymbol{r}) = 0$ when the objective function $O(\boldsymbol{d}) + C(\boldsymbol{d},\boldsymbol{r})$ is minimized. Additionally, the graph corresponding to the state of $\boldsymbol{d}$ is a DAG if $C(\boldsymbol{d},\boldsymbol{r}) = 0$. Consequently, this QUBO formulation is equivalent to learning Bayesian networks.

3 Decomposed Quadratization

Searching over candidate parent sets is a key technique in Bayesian network structure learning. To deal with candidate parent sets in a QUBO formulation, we use auxiliary variables corresponding to the sets of primary variables. The auxiliary variables capture the score component and identify the state of primary variables. We decompose a quadratization into the optimization of auxiliary variables and the transformation that maps them to primary variables. This decompostion of quadratization is defined as

Definition 3.1. The decomposed quadratization of f is the pair of a function $g_1 : \{0,1\}^J \to \{0,1\}^I$ and a quadratic polynomial function $g_2 : \{0,1\}^J \to \mathbb{R}$ such that

$$f(\boldsymbol{p}) = \min_{\boldsymbol{q}\in\{0,1\}^J; \boldsymbol{p}=g_1(\boldsymbol{q})} g_2(\boldsymbol{q}). \quad (6)$$

Example 3.2. Let $g_1(q_1,q_2) = (q_1,q_2,q_2)$ and $g_2(q_1,q_2) = -q_1 - 3q_2 + 2q_1q_2$. Then (g_1,g_2) is a decomposed quadratization of $f(p_1,p_2,p_3) = -p_1 - 3p_2p_3 + 2p_1p_2p_3$.

Here we describe the behavior of decomposed quadratization. Considering the case where each p_i corresponds to q_i, the concept of decomposed quadratization encompasses quadratization. To capture the subsets of the terms in $\mathcal{F}$, we make a set $\emptyset \subset Q_i \subseteq \{1,\cdots,I\}$ correspond to a binary variable $q_i \in \{0,1\}$ one-to-one. Let $\bar{g}_1, \bar{g}_2$ denote

$$\bar{g}_1(\boldsymbol{q}) \equiv (\min\{1, \sum_{1\le j\le J; i\in Q_j} q_j\})_{i=1}^I, \quad (7)$$

$$\bar{g}_2(\boldsymbol{q}) \equiv \sum_{1\le i\le j\le J} \bar{\pi}(Q_i,Q_j) q_i q_j + h(\boldsymbol{q}), \quad (8)$$

where $\bar{\pi} : 2^{\{1,\cdots,I\}} \times 2^{\{1,\cdots,I\}} \to \mathbb{R}$, and $h : \{0,1\}^J \to [0,\infty)$is a quadratic polynomial. To discuss the behavior of $(\bar{g}_1,\bar{g}_2)$, we introduce theorem 3.5. Assumption 3.3 ensures the comprehensiveness of the terms in $\mathcal{F}$. Assumption 3.4 presents the inclusion relations among $\{Q_i\}_{i=1}^J$.

Assumption 3.3. $\mathcal{F} \subseteq \bigcup_{1\le i\le j\le J}\{Q_i \cup Q_j\}$.

Assumption 3.4. If there exists a pair of an index $1 \le i \le J$ and a set $\mathcal{W} \subseteq \{1,\cdots,J\}\setminus\{i\}$ with $|\mathcal{W}| \ge 2$ such that

$$Q_i \subseteq \bigcup_{j\in\mathcal{W}} Q_j \text{ and } Q_i \nsubseteq \bigcup_{j\in\mathcal{W}'} Q_j \text{ for any } \mathcal{W}' \subset \mathcal{W}, \quad (9)$$

then $Q_i = Q_j \cup Q_k$ for some $j,k \in \{1,\cdots,J\}\setminus\{i\}$.

Theorem 3.5. *Suppose that assumption 3.3 and assumption 3.4 hold. Then there exists $(\bar{\pi}, h)$ such that $(\bar{g}_1,\bar{g}_2)$ is a decomposed quadratization of f.*

Proof. See appendix B.1. □

To capture the inclusion relations, we consider designing a penalty function h that induces

$$Q_i \subseteq \bigcup_{j\in\mathcal{W}} Q_j \Rightarrow (1-q_i)\prod_{j\in\mathcal{W}} q_j = 0. \quad (10)$$

According to lemma 3.6, one element cannot be directly covered by more than three others. This prevents the direct incorporation of eq. (10) into a penalty. Assumption 3.4 indirectly addresses it by covering one element with two others. Lemma 3.7 restricts the formulation of penalties to capture this assumption.

Lemma 3.6. *For $K \in \{3,4,\cdots\}$, no quadratic polynomial function $\phi : \{0,1\}^{K+1} \to \mathbb{R}$ satisfies*

$$\phi(q_1,\cdots,q_K,0) = 0 \Leftrightarrow \prod_{1\le i\le K} q_i = 0. \quad (11)$$

Proof. See appendix B.2. □

Lemma 3.7. *Suppose that a quadratic polynomial function $\phi : \{0,1\}^3 \to \mathbb{R}$ satisfies eq. (11). Then the following holds:*

$$\phi(0,0,1) + \phi(1,1,1) \ne \phi(1,0,1) + \phi(0,1,1). \quad (12)$$

Proof. See appendix B.3. □

Quadratization is a specific form of decomposed quadratization that requires assumption 3.8 in addition to assumption 3.3 and assumption 3.4. This additional assumption of quadratization is disadvantageous for reducing the number of auxiliary variables in certain tasks such as example 3.9.

Assumption 3.8. $\{i\} \in \{Q_j\}_{j=1}^J$ for all $1 \le i \le I$.

Example 3.9. Let $\mathcal{F} = \{\{1,2,3,4\},\{1,2,3\},\{4\},\{3,4\}\}$. A decomposed quadratization requires 3 auxiliary variables: $Q_1 = \{1,2,3\}, Q_2 = \{4\}$, and $Q_3 = \{3\}$. A quadratization requires 6 auxiliary variables: $Q_1 = \{1\}, Q_2 = \{2\}, Q_3 = \{3\}, Q_4 = \{4\}, Q_5 = \{1,2\}$, and $Q_6 = \{3,4\}$.

Considering the biases and variable couplers in a QUBO, the auxiliary variables satisfy $J \ge \lceil -\frac{1}{2} + \frac{1}{2}\sqrt{1+8|\mathcal{F}|}\rceil$. We minimize the number of auxiliary variables to capture assumption 3.3 and assumption 3.4. An optimal solution of $\{Q_i\}_{i=1}^J$ is a subset of $\bigcup_{V\in\mathcal{F}} 2^V \setminus \{\emptyset\}$. Let $V_{\mathrm{close}}, V_{\mathrm{open}} \subseteq$

Algorithm 1: Search Space Reduction

1: $W_{\mathrm{open}} \leftarrow \mathcal{F}, V_{\mathrm{close}} \leftarrow \emptyset, V_{\mathrm{open}} \leftarrow \bigcup_{V \in \mathcal{F}} 2^V \setminus \{\emptyset\}$.
2: Merge $\{V \in V_{\mathrm{open}} \mid$ for some $W \in W_{\mathrm{open}}$, V is only an element in V_{open} such that $W \in \bigcup_{V' \in \{\emptyset\} \cup V_{\mathrm{close}}} \{V \cup V'\}\}$ into V_{close}.
3: Exclude $\bigcup_{V,V' \in V_{\mathrm{close}}} \{V \cup V'\}$ from W_{open}.
4: Exclude $V_{\mathrm{close}} \cup \{V \in V_{\mathrm{open}} \mid$ there does not exist $W \in W_{\mathrm{open}}$ such that $V \subseteq W\}$ from V_{open}.

$\bigcup_{V \in \mathcal{F}} 2^V \setminus \{\emptyset\}$ such that $V_{\mathrm{close}} \cap V_{\mathrm{open}} = \emptyset$, V_{close} is a subset of an optimal solution, and $V_{\mathrm{close}} \cup V_{\mathrm{open}}$ contains an optimal solution. Since V_{close} is a subset of an optimal solution, we can ignore the terms in $\mathcal{F}$ that can be represent by the union of some two elements in V_{close}. Let $W_{\mathrm{open}} \subseteq \mathcal{F}$ be the terms that still cannot be ignored. We provide algorithm 1 to find an optimal solution. The mechanism of this algorithm is as follows:

Row 2: If there is only one way to represent W, then V is included in an optimal solution.

Row 3: We can ignore elements in W_{open} represented by pairs of elements in V_{close}.

Row 4: If V is not a subset of any element in W_{open}, then it is not included in an optimal solution.

Here we minimize $\sum_{i=1}^{\nu} v_i$ under the constraint that assumption 3.3 and assumption 3.4 hold for

$$\{Q_i\}_{i=1}^{J} = V_{\mathrm{close}} \cup \{V_i \mid v_i = 1\}_{i=1}^{\nu}, \tag{13}$$

where $\boldsymbol{v} \equiv (v_i)_{i=1}^{\nu} \in \{0,1\}^{\nu}$ and $\{V_i\}_{i=1}^{\nu} \equiv V_{\mathrm{open}}$. By expressing the constraints as linear inequalities, the search for an optimal solution can be formulated as an integer linear programming problem of $\boldsymbol{v}$. See appendix C.1. This formulation, with some conversion, can be adapted to minimize the number of terms. See appendix C.2.

4 Efficient QUBO for Structure Learning

In this section, we propose the decomposed quadratization of O to encode candidate parent sets efficiently. Here we introduce the concept of candidate parent subsets.

Definition 4.1. The candidate parent subsets of X_i are defined as the set containing elements $U, U' \subseteq \mathcal{X}_i$ such that $U \cup U' = W$, for any W in the candidate parent sets of X_i.

Let $\{U_{ij}\}_{j=0}^{\mu_i}$ denote the candidate parent subsets of X_i. To encode candidate parent subsets in a QUBO, we make a candidate parent subset U_{ij} correspond to a binary variable $u_{ij} \in \{0,1\}$ one-to-one. The state of $\boldsymbol{d}$ is identified by

$$D(\boldsymbol{u}) \equiv ((\min\{1, \sum_{1 \le k \le \mu_i; X_j \in U_{ik}} u_{ik}\})_{j=1}^{n})_{i=1}^{n}, \tag{14}$$

where $\boldsymbol{u} \equiv ((u_{ij})_{j=1}^{\mu_i})_{i=1}^{n}$. A parent set is captured by the union of at most two elements included in candidate parent subsets. Since each vertex has just one parent set, we can disregard the inclusion relations among candidate parent subsets as mentioned in assumption 3.4. From definition 4.1, the following proposition equivalent to assumption 3.3 holds.

Proposition 4.2. *$\mathcal{F}_i \equiv \{W_{ij}\}_{j=1}^{\lambda_i} \subseteq \bigcup_{1 \le j \le k \le \mu_i} \{U_{ij} \cup U_{ik}\} \subseteq 2^{\mathcal{X}_i} \setminus \{\emptyset\}$ for all $1 \le i \le n$.*

The main distinction of our approach from section 2.3 is that the objective function is represented not by edges among vertices but by candidate parent subsets. To enforce exactly one parent set per vertex (i.e., $\sum_{1 \le j \le \mu_i} u_{ij} \le 2$), we use the constraint characterized by a penalty coefficient $\xi \in (0, \infty)$ and auxiliary variables $\boldsymbol{z} = (z_i)_{i=1}^{n} \in \{0,1\}^n$. Then the score component is

$$\bar{O}(\boldsymbol{u}, \boldsymbol{z}) \equiv \sum_{1 \le i \le n} \sum_{1 \le j \le k \le \mu_i} \bar{\pi}_i(U_{ij}, U_{ik}) u_{ij} u_{ik} + \sum_{1 \le i \le n} \xi \Big(z_i - z_i \sum_{1 \le j \le \mu_i} u_{ij} + \sum_{1 \le j < k \le \mu_i} u_{ij} u_{ik} \Big), \tag{15}$$

where $\bar{\pi}_i(U, U') \equiv (-1)^{1+|\{U\} \cup \{U'\}|} (\log S_i(U \cup U') - \log S_i(U) - \log S_i(U') + \log S_i(\emptyset))$. Using proposition 4.2, our formulation requires equal to or fewer binary variables than the quadratization of O. For candidate parent subsets in quadratization, it is essential to capture the relation between them and the edges as follows:

$$(1 - u_{ij}) \prod_{1 \le k \le n; X_k \in U_{ij}} d_{ik} = 0. \tag{16}$$

Lemma 3.6 demonstrates that such relation cannot be captured by a quadratic polynomial when a candidate parent subset includes more than three elements. Consequently, we benefit from the use of decomposed quadratization with candidate parent subsets. The following lemma provides a lower bound for the penalty coefficient ξ.

Lemma 4.3. *Let ξ_0 be defined as*

$$\xi_0 \equiv -3 \min_{1 \le i \le n} \min_{0 \le j \le k \le \mu_i} \bar{\pi}_i(U_{ij}, U_{ik}). \tag{17}$$

Suppose that $\xi > \xi_0$ holds and the state of $\boldsymbol{d}$ aligns with candidate parent sets. Then $(D, \bar{O})$ is a decomposed quadratization of O.

Proof. See appendix B.4. □

The constraint for definition 2.1 is also captured by candidate parent subsets. Let $\bar{C}(\boldsymbol{u}, \boldsymbol{r})$ denote the right side of eq. (5) where d_{ij} is replaced by

$$\sum_{1 \le k \le \mu_i; X_j \in U_{ik}} u_{ik}, \tag{18}$$

where this substitution does not use additional auxiliary variables, but may increase the number of terms. The objective function in our approach is $\bar{H}(\boldsymbol{u}, \boldsymbol{r}, \boldsymbol{z}) \equiv \bar{O}(\boldsymbol{u}, \boldsymbol{z}) + \bar{C}(\boldsymbol{u}, \boldsymbol{r})$. Figure 2 illustrates this formulation. The following theorem guarantees that optimal networks can be obtained using our QUBO formulation.

Theorem 4.4. *Let δ_0 be defined as*

$$\delta_0 \equiv \max_{1 \le i \le n, 0 \le j \le \lambda_i} (\log S_i(W_{ij}) - \log S_i(\emptyset)). \tag{19}$$

Suppose that $\xi > \xi_0$ and $\delta_2 > (n-2)\delta_1 > (n-2)\delta_0$ hold. Then $\bar{C}(\boldsymbol{u}, \boldsymbol{r}) = 0$ holds when $\bar{H}(\boldsymbol{u}, \boldsymbol{r}, \boldsymbol{z})$ is minimized.

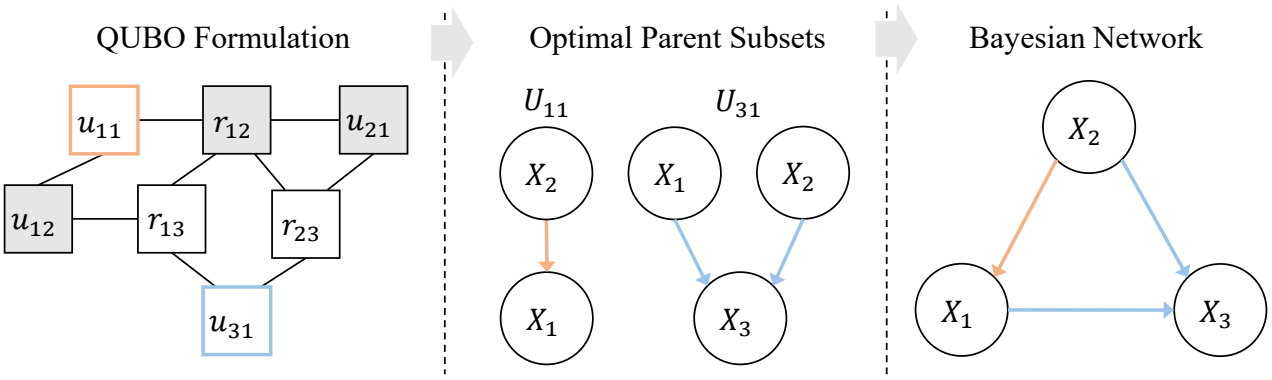

Figure 2: Example of the QUBO formulation with candidate parent subsets. An undirected edge between two binary variables represents that the objective function contains their product term. Gray signifies that the state of a binary variable is 0 and white is 1. Each color corresponds to the edge in the DAG. Let $n = 3, m = 2, \lambda_1 = 3, \lambda_2 = 1, \lambda_3 = 1, \Pi_1 = \{X_2\}, \Pi_2 = \emptyset, \Pi_3 = \{X_1, X_2\}, W_{11} = \{X_2\}, W_{12} = \{X_3\}, W_{13} = \{X_2, X_3\}, W_{21} = \{X_1, X_3\}, W_{31} = \{X_1, X_2\}, \mu_1 = 2, \mu_2 = 1, \mu_3 = 1, U_{11} = \{X_2\}, U_{12} = \{X_3\}, U_{21} = \{X_1, X_3\}$, and $U_{31} = \{X_1, X_2\}$.

Algorithm 2: Candidate Parent Subset Reduction

1: $W_{\text{open}} \leftarrow \mathcal{F}_i, V_{\text{close}} \leftarrow \emptyset, V_{\text{open}} \leftarrow \bigcup_{V \in \mathcal{F}_i} 2^V \setminus \{\emptyset\}$.
2: **while** V_{close} has changed since the last evaluation, **do**
3: Exclude $\{V \in V_{\text{open}} \mid V \subseteq W$ for at most one element $W \in W_{\text{open}}\}$ from V_{open}.
4: Execute the row from 2 to 4 in algorithm 1.
5: Merge $W_{\text{open}} \setminus \bigcup_{V,V' \in V_{\text{close}} \cup V_{\text{open}}} \{V \cup V'\}$ into V_{close}, and exclude it from W_{open}.
6: **end while**
7: Merge $W_{\text{open}} \setminus \bigcup_{V \in V_{\text{open}}, V' \in V_{\text{close}}} \{V \cup V'\}$ into V_{open}.

Proof. See appendix B.5. □

The number of required binary variables is $\sum_{1 \leq i \leq n} (\mu_i + 1) + |\mathcal{E}|$. We find candidate parent subsets to minimize μ_i under the constraint to capture proposition 4.2. An optimal solution of candidate parent subsets of X_i is a subset of $\bigcup_{V \in \mathcal{F}_i} 2^V \setminus \{\emptyset\}$. Since assumption 3.4 is not required, algorithm 2 can reduce the search space of candidate parent subsets. This algorithm works as follows:

Row 3: We temporarily ignore elements that contribute to represent at most one element included in W_{open}.

Row 5: Considering the removed elements at row 3, an optimal solution includes the elements in W_{open} that cannot be represented by pairs of elements in $V_{\text{close}} \cup V_{\text{open}}$.

Row 7: From a perspective similar to row 5, elements in W_{open} that are represented solely by pairs of elements in V_{open} may be included in an optimal solution.

Additionally, we can exclude the constraint that captures assumption 3.4 from the integer linear programming problem.

5 Experimental Results

In this section, we investigate the performance of our approach for reducing the required number of binary variables. We also demonstrate the score maximization using some solvers with our QUBO formulation. Version 9.1.2 of the Gurobi Optimizer [1] was used to solve the integer linear programming problems. The fourth-generation Fujitsu Digital Annealer was employed for score maximization. All experiments, except for those using the Digital Annealer, were conducted on a 64-bit Windows machine with an Intel Xeon W-2265 @ 3.50 GHz and 128 GB of RAM. The code was implemented in the Julia programming language 1.5.3 version. As benchmarks, we adopted 5 simulated datasets from each of the 8 discrete networks: alarm ($n = 37$), barley ($n = 48$), hailfinder ($n = 56$), hepar2 ($n = 70$), win95pts ($n = 76$), pathfinder ($n = 109$), munin1 ($n = 186$), and andes ($n = 223$) from the Bayesian network repository [2]. In addition, we used 5 times non-recoverable extracted datasets from each of the 3 datasets: chess ($n = 75$), mushroom ($n = 117$), and connect ($n = 129$) from the Frequent Item Set Mining Dataset Repository [3]. The sample size of each dataset is 1000.

5.1 QUBO Formulation

We show that our formulation reduces the number of binary variables required in a QUBO.

Settings. The objective function was $\bar{H}(\boldsymbol{u}, \boldsymbol{r}, \boldsymbol{z})$. The score function was the BDeu score (Buntine 1991), where the equivalent sample size was set to 1. See appendix D.1. The identification of candidate parent sets was performed exactly. Candidate parent sets were identified by simply enumerating all the elements in $\{W \subseteq \mathcal{X} \setminus \{X_i\} \mid |W| \leq m\}$ and comparing their local scores when the inclusion relations between them was in place. The maximum parent set size m for each instance with more than 75 variables was set to the largest value that allowed the identification process to be completed within 48 [h]. The number of candidate parent

[1] https://www.gurobi.com/
[2] https://www.bnlearn.com/bnrepository/
[3] http://fimi.uantwerpen.be/data/

Instance	m	O'Gorman et al.		IP only		IP + algorithm 2	
		Number	Time [s]	Number	Time [s]	Number	Time [s]
alarm	4	3982 ± 346	272 ± 61	$\mathbf{1373 \pm 46}$	61 ± 29	$\mathbf{1373 \pm 46}$	$\mathbf{14 \pm 2}$
barley	4	1826 ± 279	271 ± 149	$\mathbf{995 \pm 143}$	$\mathbf{11 \pm 2}$	$\mathbf{995 \pm 143}$	$\mathbf{11 \pm 2}$
hailfinder	4	2374 ± 31	52 ± 24	$\mathbf{1913 \pm 6}$	$\mathbf{4 \pm 1}$	$\mathbf{1913 \pm 6}$	$\mathbf{4 \pm 1}$
hepar2	4	3629 ± 423	51 ± 16	$\mathbf{2616 \pm 133}$	3 ± 0	$\mathbf{2616 \pm 133}$	$\mathbf{2 \pm 0}$
chess	2	7040 ± 167	$\mathbf{63 \pm 2}$	$\mathbf{6964 \pm 165}$	89 ± 4	$\mathbf{6964 \pm 165}$	103 ± 4
chess	3	80116 ± 3014	$\mathbf{3229 \pm 151}$	26355 ± 1403	7268 ± 492	$\mathbf{23662 \pm 1103}$	5567 ± 376
win95pts	2	5150 ± 39	$\mathbf{27 \pm 2}$	$\mathbf{5028 \pm 42}$	33 ± 3	$\mathbf{5028 \pm 42}$	37 ± 4
win95pts	3	37705 ± 1935	1223 ± 90	9220 ± 398	1036 ± 117	$\mathbf{9025 \pm 333}$	$\mathbf{890 \pm 105}$
pathfinder	2	10831 ± 382	$\mathbf{178 \pm 8}$	$\mathbf{10725 \pm 379}$	259 ± 30	$\mathbf{10725 \pm 379}$	284 ± 20
pathfinder	3	107725 ± 5329	$\mathbf{11695 \pm 834}$	38245 ± 3242	30707 ± 4322	$\mathbf{34448 \pm 3016}$	21364 ± 3773
mushroom	2	18050 ± 508	$\mathbf{318 \pm 13}$	$\mathbf{17930 \pm 506}$	681 ± 32	$\mathbf{17930 \pm 506}$	513 ± 23
connect	2	15155 ± 334	$\mathbf{137 \pm 9}$	$\mathbf{14978 \pm 340}$	228 ± 17	$\mathbf{14978 \pm 340}$	240 ± 12
connect	3	179788 ± 6242	$\mathbf{9897 \pm 602}$	64533 ± 4352	30381 ± 4267	$\mathbf{58247 \pm 4053}$	20106 ± 2661
munin1	2	25180 ± 266	$\mathbf{699 \pm 20}$	$\mathbf{25001 \pm 268}$	953 ± 75	$\mathbf{25001 \pm 268}$	1079 ± 67
andes	2	29502 ± 171	$\mathbf{40 \pm 1}$	$\mathbf{28810 \pm 182}$	249 ± 37	$\mathbf{28810 \pm 182}$	250 ± 35
andes	3	92838 ± 3735	2182 ± 289	$\mathbf{31162 \pm 136}$	504 ± 132	$\mathbf{31162 \pm 136}$	$\mathbf{217 \pm 8}$

Table 1: Number of binary variables and execution time for the QUBO formulation. The mean and standard deviation of 5 trials are presented. All the candidate parent subsets not proven optimal within 60 [s] were feasible solutions. The bold values represent the smallest number of binary variables and the shortest execution times among all approaches.

sets and the execution time for their identification are found in appendix D.2. The integer linear programming problem to find optimal candidate parent subsets was formulated as appendix C.1. The time limit for each variable was 60 [s]. The execution time for the QUBO formulation includes that of algorithm 2, setting up the integer linear programming problem, solving it, and incorporating the solution into the objective function.

Baselines. We compare our formulation with the approach proposed by (O'Gorman et al. 2014). Each vertex requires $\lfloor \frac{1}{4}(|\mathcal{X}_i| - 1)^2 \rfloor + 2$ auxiliary variables when $m_i = 3$, and $\frac{1}{2}|\mathcal{X}_i|(|\mathcal{X}_i| - 1) + 3$ when $m_i = 4$, where $m_i \equiv \max_{0 \leq j \leq \lambda_i} |W_{ij}|$. To investigate the merit of algorithm 2, we found candidate parent subsets using the integer linear programming problem with $W_{\mathrm{open}} = \mathcal{F}_i, V_{\mathrm{open}} = \bigcup_{V \in \mathcal{F}_i} 2^V \setminus \{\emptyset\}$, and $V_{\mathrm{close}} = \emptyset$. We refer to this as "IP only" and our approach with algorithm 2 as "IP + algorithm 2". The candidate parent sets to formulate these baselines were the same as those in the previously described setting. The execution time for each baseline does not include the time spent on identifying candidate parent sets.

Evaluation. The comparison between IP + algorithm 2 and IP only in table 1 demonstrates that optimizing candidate parent subsets with algorithm 2 is advantageous in terms of both the number of binary variables and execution time when the parent set size is larger. The quadratization for some instances with $m = 3$ required binary variables over the 100K bit capacity of the Digital Annealer. The decomposed quadratization in our approach significantly reduced the required number of binary variables compared to the quadratization approach. The execution time of our formulation was disadvantageous for most larger networks, primarily due to the increasing number of terms. Table 2 shows that incorporating the candidate parent subsets into the QUBO introduced a significant bottleneck.

5.2 Score Maximization

We demonstrate score maximization using the QUBO formulation by IP + algorithm 2.

Settings. The time limit for each trial was set to 3600 [s]. The coefficients in the penalty terms were $\delta_1 = 1.1\delta_0, \delta_2 = 1.1(n-2)\delta_1$, and $\xi = 1.1\xi_0$. We used two heuristic solvers and one exact method: the Digital Annealer (DAQ), classical simulated annealing (SAQ), and the Gurobi Optimizer (GOQ). The inequality constraint is placed outside the objective function in both DAQ and GOQ. This formulation is strictly categorized as binary quadratic programming. See appendix D.4. The parameters for DAQ and GOQ were set to their default values. The annealing schedule of SAQ was geometric, starting with $T_0 = 100$. See appendix D.3. Additional bits for a minor embedding (Choi 2008, 2010; Eppstein 2009) were not required because the Digital Annealer is a fully-coupled type. Note that the short description of minor embedding can be found in appendix D.6.

Baselines. The baseline algorithms are only score-based. We adopted three approximate approaches and one exact algorithm: the hill climbing search method (HCS), the simulated annealing over ordering space (SAO), the acyclic selection ordering-based search (ASO), and the GOBNILP software [4] (GOB). The HCS and SAO are approximate approaches described in (Scutari, Graafland, and Gutiérrez 2019). The ASO and GOB are known as competitive algorithms. The tabu list length in HCS was set to 10, and the state of 10 randomly selected possible edges was changed

[4] https://www.cs.york.ac.uk/aig/sw/gobnilp/

Instance	m	Time [s]
alarm	4	$1 \pm 0, 3 \pm 1, 10 \pm 1$
barley	4	$0 \pm 0, 0 \pm 0, 11 \pm 2$
hailfinder	4	$0 \pm 0, 0 \pm 0, 4 \pm 1$
hepar2	4	$0 \pm 0, 0 \pm 0, 2 \pm 0$
chess	2	$2 \pm 0, 0 \pm 0, 100 \pm 4$
chess	3	$406 \pm 49, 2554 \pm 199, 2608 \pm 195$
win95pts	2	$1 \pm 0, 0 \pm 0, 36 \pm 4$
win95pts	3	$45 \pm 5, 398 \pm 82, 446 \pm 54$
pathfinder	2	$3 \pm 0, 0 \pm 0, 281 \pm 19$
pathfinder	3	$1407 \pm 272, 1679 \pm 184, 18278 \pm 3349$
mushroom	2	$23 \pm 1, 0 \pm 0, 490 \pm 22$
connect	2	$5 \pm 0, 0 \pm 0, 235 \pm 12$
connect	3	$2472 \pm 436, 4129 \pm 148, 13505 \pm 2114$
munin1	2	$12 \pm 1, 0 \pm 0, 1067 \pm 67$
andes	2	$1 \pm 0, 0 \pm 0, 249 \pm 35$
andes	3	$6 \pm 1, 2 \pm 1, 209 \pm 8$

Table 2: Execution time for IP + algorithm 2. The execution time for each of the following three steps is displayed. Left : algorithm 2 and preparation of the interger linear programming problem. Middle : finding candidate parent subsets by solving the problem. Right : incorporating the candidate parent subsets into the objective function of the QUBO. Note that the execution time for IP only is shown in appendix D.5.

Instance	m	DAQ	GOQ	HCS	SAO	GOB
alarm	4	0	0	0	0	**5**
barley	4	0	0	0	0	**5**
hailfinder	4	0	0	0	0	**5**
hepar2	4	0	0	0	0	**5**
chess	2	**5**	0	0	0	0
chess	3	**5**	0	0	0	0
win95pts	2	1	0	0	0	**4**
win95pts	3	2	0	0	**3**	0
pathfinder	2	**4**	0	0	1	0
pathfinder	3	**5**	0	0	0	0
mushroom	2	**5**	0	0	0	0
connect	2	**5**	0	0	0	0
connect	3	**2**	1	1	1	0
munin1	2	**5**	0	0	0	0
andes	2	**5**	0	0	0	0
andes	3	**5**	0	0	0	0

Table 3: Results of score maximization. The number of trials achieving the highest score is displayed. The highest score refers to the shortest execution time when the scores are the same. For the four instances above the line, the GOB identified optimal networks within a few seconds. While the GOQ identified an optimal network in only one trial of the barley instance, the execution time was longer than the GOB. The solutions for the win95pts with $m = 2$ and the connect with $m = 3$ were not proven to be optimal. The SAQ and ASO never won in any of the trials. The bold values show the greatest number of wins among all solvers.

upon termination of the greedy search. The annealing schedule of SAO was same as that of SAQ. The ASO was the version proposed in (Scanagatta et al. 2015). The version of GOB was the pygobnilp1.0 that relies on the Gurobi Optimizer. We used common candidate parent sets in our formulation, except for HCS. To ensure a fair comparison, the time limit for each of SAO, ASO, and GOB was set at 3600 [s] plus the execution time required for the QUBO formulation by IP + algorithm 2. The time limit for HCS was further extended by the time needed to identify candidate parent sets. We ignored the time to transfer the encoding information online to the Digital Annealer environment. The HCS and ASO were restarted repeatedly during the time limit.

Evaluation. Table 3 displays the results of score maximization. For the smaller instances, GOB identified optimal networks. In most other cases, the scores achieved by DAQ exceeded those of the baselines. If the long execution time required for the QUBO formulation can be reduced in future work, the difference will increase further. The Digital Annealer effectively highlighted the strengths of our formulation compared to the results from SAQ and GOQ using the same QUBO formulation.

6 Conclusion

We proposed a QUBO formulation tailored to score-based Bayesian network structure learning. The essence of this approach lies in reducing the number of required binary variables through decomposed quadratization with candidate parent subsets. We also provided an algorithm to efficiently find optimal candidate parent subsets. Experimental results demonstrated that our approach significantly reduced the number of binary variables compared to the previous work based on quadratization. Additionally, our formulation using the Digital Annealer achieved improved BDeu scores over existing methods for medium-sized instances. We expect that our approach can be more effectively applied to larger-scale structure learning problems in the future development of annealing processors.

Limitations. We have identified three limitations in our approach. First, the execution time to incorporate the candidate parent subsets into the objective function is a potential drawback for larger-scale problems. Secondly, capturing a parent set with two bits may be disadvantageous for a 1-bit inversion search, as transitioning from one candidate parent set to another necessitates an inversion of up to four bits. Lastly, our formulation could potentially have a negative impact by increasing the number of terms in a QUBO. In particular, a minor embedding requires additional bits as the number of terms increases.

Future Works. Along with further investigation addressing the above limitations, we will explore the application of decomposed quadratization to tasks beyond Bayesian network structure learning. Decomposed quadratization can serve as a promising alternative to conventional quadratization for a wide range of tasks involving multilinear polynomials with higher-degree terms.

Acknowledgments

We acknowledge Fujitsu Limited for their support in providing access to the fourth-generation Fujitsu Digital Annealer.

References

Anthony, M.; Boros, E.; Crama, Y.; and Gruber, A. 2016. Quadratic reformulations of nonlinear binary optimization problems. *Mathematical Programming*, 162(1): 115–144.

Besag, J. 1975. Spatial interaction and the statistical analysis of lattice systems. *The Journal of the Royal Statistical Society: Series D*, 24: 179–195.

Boros, E.; Crama, Y.; and Heck, E. R. 2019. Compact quadratizations for pseudo-Boolean functions. *Journal of Combinatorial Optimization*.

Bouckaert, R. R. 1994. Properties of Bayesian Belief Network Learning Algorithms. In *Proceedings of the 10th International Conference on Uncertainty in Artificial Intelligence*, 102–109.

Buntine, W. L. 1991. Theory refinement of Bayesian networks. In *Proceedings of the 7th International Conference on Uncertainty in Artificial Intelligence*, 52–60.

Chickering, D. M.; Heckerman, D.; and Meek, C. 2003. Large-sample learning of Bayesian networks is NP-hard. In *Proceedings of the 19th International Conference on Uncertainty in Artificial Intelligence*, 124–133.

Choi, V. 2008. Minor-embedding in adiabatic quantum computation: 1. The parameter setting problem. *Quantum Information Processing*, 7: 193–209.

Choi, V. 2010. Minor-embedding in adiabatic quantum computation: 2. Minor-universal graph design. *Quantum Information Processing*, 10: 343–352.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 1990. *Introduction to Algorithms*. MIT Press.

Cussens, J. 2011. Bayesian network learning with cutting planes. In *Proceedings of the 27th International Conference on Uncertainty in Artificial Intelligence*, 153–160.

de Campos, C. P.; and Ji, Q. 2010. Properties of Bayesian Dirichlet Scores to Learn Bayesian Network Structures. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 431–436.

Eppstein, D. 2009. Finding Large Clique Minors is Hard. *Graph Algorithms and Applications*, 13(2): 197–204.

Heck, E. R. 2018. *Linear and quadratic reformulations of nonlinear optimization problems in binary variables*. Ph.D. thesis, Universite de Liege, Liege, Belgium.

Kitson, N. K.; Constantinou, A. C.; Guo, Z.; Liu, Y.; and Chobtham, K. 2023. A survey of Bayesian Network structure learning. *Artificial Intelligence Review*.

Koller, D.; and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Kolmogorov, V.; and Rother, C. 2007. Minimizing nonsubmodular functions with graph cuts – a review. *Journal of IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29: 1274–1279.

Matsubara, S.; Takatsu, M.; Miyazawa, T.; Shibasaki, T.; Watanabe, Y.; Takemoto, K.; and Tamura, H. 2020. Digital Annealer for High-Speed Solving of Combinatorial optimization Problems and Its Applications. In *Proceedings of the 25th Asia and South Pacific Design Automation Conference*, 667–672.

Nakayama, H.; Koyama, J.; Yoneoka, N.; and Miyazawa, T. 2021. *Description: Third Generation Digital Annealer Technology*. Fujitsu Limited. https://www.fujitsu.com/jp/documents/digitalannealer/researcharticles/DA_WP_EN_20210922.pdf.

O'Gorman, B.; Ortiz, A. P.; Babbush, R.; Guzik, A. A.; and Smelyanskiy, V. 2014. Bayesian Network Structure Learning Using Quantum Annealing. *The European Physical Journal Special Topics*, 225(1).

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Rosenberg, I. G. 1975. Reduction of Bivalent Maximization to the Quadratic Case. *Journal of Cahiers du Centre d'Etudes de Recherche Operationnelle*, 17: 71–74.

Sasdelli, M.; and Chin, T.-J. 2021. Quantum Annealing Formulation for Binary Neural Networks. In *Proceedings of the 2021 International Conference on Digital Image Computing Techniques and Applications*, 1–10.

Scanagatta, M.; de Campos, C. P.; Corani, G.; and Zaffalon, M. 2015. Learning Bayesian Networks with Thousands of Variables. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 1864–1872.

Scutari, M.; Graafland, C. E.; and Gutiérrez, J. M. 2019. Who Learns Better Bayesian Network Structures: Accuracy and Speed of Structure Learning Algorithms. *Journal of Approximate Reasoning*, 115: 235–253.

Teyssier, M.; and Koller, D. 2005. Ordering-Based Search: A Simple and Effective Algorithm for Learning Bayesian Networks. In *Proceedings of the 21th International Conference on Uncertainty in Artificial Intelligence*, 584–590.

Verma, A.; and Lewis, M. 2020. Optimal Quadratic Reformulations of Fourth Degree Pseudo-Boolean Functions. *Journal of Optimization Letters*, 14: 1557–1569.

Yamamoto, K. 2020. *Research on Annealing Processors for Large-Scale Combinatorial Optimization Problems*. Ph.D. thesis, Graduate School of Information Science and Technology Hokkaido University.

Yawata, K.; Osakabe, Y.; Okuyama, T.; and Asahara, A. 2022. QUBO Decision Tree: Annealing Machine Extends Decision Tree Splitting. In *Proceedings of the 13th International Conference on Knowledge Graph*.