applied sciences

MDPI

Article

Petri Net Modeling for Ising Model Formulation in Quantum Annealing

Morikazu Nakamura *, Kohei Kaneshima and Takeo Yoshida

Computer Science & Intelligent Systems Group, Faculty of Engineering, University of the Ryukyus, Okinawa 903-0213, Japan; k208580@ie.u-ryukyu.ac.jp (K.K.); tyoshida@ie.u-ryukyu.ac.jp (T.Y.)
* Correspondence: morikazu@ie.u-ryukyu.ac.jp

Abstract: Quantum annealing is an emerging new platform for combinatorial optimization, requiring an Ising model formulation for optimization problems. The formulation can be an essential obstacle to the permeation of this innovation into broad areas of everyday life. Our research is aimed at the proposal of a Petri net modeling approach for an Ising model formulation. Although the proposed method requires users to model their optimization problems with Petri nets, this process can be carried out in a relatively straightforward manner if we know the target problem and the simple Petri net modeling rules. With our method, the constraints and objective functions in the target optimization problems are represented as fundamental characteristics of Petri net models, extracted systematically from Petri net models, and then converted into binary quadratic nets, equivalent to Ising models. The proposed method can drastically reduce the difficulty of the Ising model formulation.

Keywords: quantum annealing; adiabatic quantum optimization; Ising model; quadratic unconstrained binary optimization; Petri nets; binary quadratic net; model-based engineering; model-based optimization

Citation: Nakamura, M.; Kaneshima, K.; Yoshida, T. Petri Net Modeling for Ising Model Formulation in Quantum Annealing. *Appl. Sci.* **2021**, *11*, 7574. https://doi.org/10.3390/app11167574

Academic Editors: Luis Gomes and João Paulo Barros

Received: 3 July 2021
Accepted: 16 August 2021
Published: 18 August 2021

1. Introduction

Quantum annealing is a metaheuristic for combinatorial optimization in quantum mechanics research [1,2]. This new approach solves unconstrained optimization problems formulated as a Hamiltonian by evolving the time-dependent Schrödinger equation from a quantum mechanical superposition of all possible states to its ground state in physical systems [3]. A quantum annealing machine is a special-purpose quantum computer that solves combinatorial optimization problems. D-wave is the first commercial quantum annealing machine [4].

Combinatorial optimization contributes to a more effective and reasonable life by minimizing the total costs or maximizing the benefits under certain constraints. Although sophisticated solvers can efficiently solve small problems, many practical problems are computationally intractable. Such problems are formally characterized as NP-hard; that is, there are no polynomial-time algorithms to solve the problems. Many researchers in computer science and operations research believe that there are no polynomial-time algorithms for NP-hard problems and continue to develop efficient heuristic algorithms to solve larger problems [5].

Quantum annealing is a metaheuristic that does not restrict the target problems. Thus far, some combinatorial optimization problems, but not many, have been formulated as Ising or Quadratic Unconstrained Binary Optimization (QUBO) models and solved using annealing machines [6–9]. Theoretical and experimental studies have contributed to improving the performance of annealing processes [10–13]. Useful software tools have been developed to utilize quantum annealing machines [14]. Digital implementations of the quantum annealing process have also emerged because this new optimization technique has the potential to outperform traditional meta-heuristic algorithms. A graphics processing

unit (GPU) based quantum annealing simulator has shown high-performance optimization by utilizing spatial and temporal parallelism during the annealing process [15]. Therefore, we have high hopes for this new innovative optimizer.

However, obstacles to quantum annealing usability include the hardness of the parameter tuning and formulation of the target problems as Ising or QUBO models. The constraints of the original problem and objective functions are represented as a penalty function, minimized as an unconstrained optimization problem. Search processes can reach infeasible solutions, and to avoid such situations, it is necessary to carefully tune the parameters in pre-performing annealing process. The difficulty of the Ising model formulation is also an essential obstacle to the usability of quantum annealing. The new annealer requires users to create quadratic penalty functions, Ising, or QUBO models, for target optimization problems.

This study focuses solely on the latter problem, which is the difficulty of formulating the Ising and QUBO models. To overcome this problem, we propose a method for generating Ising or QUBO models from Petri net models representing the target optimization problems. Our proposal is a Petri net modeling approach in which, once we model the optimization problem with Petri nets, we can systematically obtain the Ising or QUBO formulation.

A Petri net is a mathematical and graphical modeling language for various systems, such as computer networks, manufacturing systems, transportation networks, biological systems, agricultural production processes, and other applications [16,17]. We only need the domain knowledge of the target systems for modeling because the rules and components of Petri nets are simple and intuitive. Petri net modeling is quite efficient for representing integer linear programming problems because Petri nets have mathematical forms, denoted as linear expressions [18]. The authors proposed an algorithm for generating mixed-integer linear programming (MILP) based on Petri net models for practical scheduling and optimal resource assignment problems [19,20]. Our algorithm realizes the automatic formulation of MILPs for given combinatorial optimization problems.

This study extends our MILP version to quantum annealing models, where quadratic functions of binary variables need to be formulated. As far as we know, this is the first study to apply Petri net modeling to the quantum annealing. In this paper, we introduce a new notation called binary quadratic nets, which denote Ising or QUBO models, respectively. Our method incrementally targets binary quadratic nets from problem-domain Petri net models that represent target optimization problems.

This paper is organized as follows: Section 2 summarizes the basic knowledge on combinatorial optimization problems, quantum annealing, Ising models, and Petri nets. Section 3 introduces a new class of Petri nets, binary quadratic nets, for representing Ising or QUBO models. Section 4 proposes a method for formulating Ising or QUBO models from problem-domain Petri nets and presents some examples. Finally, in Section 5, we present some concluding remarks and areas of future tasks.

2. Preliminaries

This section provides the basic concepts and definitions to understand the subsequent sections; containing combinatorial optimization, quantum annealing, Ising models, and Petri net fundamentals.

2.1. Combinatorial Optimization Problems

Combinatorial optimization is a branch of operations research and computer science; it is utilized frequently in a wide range of areas, such as artificial intelligence, bioinformatics, and VLSI circuit design [21,22].

A combinatorial optimization problem can be denoted by $CP = (X, \Sigma, Cstr, f)$, where $X = \{x_1, x_2, \ldots, x_n\}$ is a set of variables, $\Sigma = \{\Sigma_1, \Sigma_2, \ldots, \Sigma_n\}$ is a set of types for each variable, $Cstr$ is a set of constraints, and $f : \Sigma_1 \times \Sigma_2 \ldots \times \Sigma_n \to \mathbb{R}$ is an objective function such that Σ includes discrete types, such as integers, binary numbers, and Boolean values.

Here, $\mathbb{R}$ is the set of real numbers. The combinatorial optimization problem requires the optimum solution to be obtained with respect to f, out of all the feasible solutions, where feasibility means that the values assigned to the variables satisfy all of the constraints in *Cstr*.

Many combinatorial optimization problems are NP-hard; that is, no polynomial-time deterministic algorithms are known, to date, to solve such problems [5]. Sophisticated solvers, based on the branch-and-bound and branch-and-cut methods, can efficiently search for feasible solutions; however, they are still computationally insufficient for large-size instances of NP-hard problems.

Related to a given combinatorial optimization problem, we sometimes consider corresponding decision problemsin quantum annealing, for which rather than requiring just a yes or no for the existence of a feasible solution, we also need to obtain a feasible solution for the decision problem. This type of decision problem is called the search variant as compared to the yes–no type decision variant [5]. For example, we solve a search variant decision problem to obtain a viable schedule to complete all tasks by a given deadline in scheduling problems.

It should be noted that NP-hard decision problems, both search and decision variants, are equally difficult to solve for the corresponding optimization problem from the viewpoint of computational complexity. However, a search variant can be implemented more easily than an optimization problem for quantum annealing. Iterative solving of search variants of the decision problem leads to the optimum solution by varying the parameter related to the optimization, that is, the deadline for the scheduling problem. We use this technique in the job shop scheduling example in Section 4.

In the following sections, we treat four NP-hard combinatorial optimization problems: minimum vertex cover, graph partitioning, traveling salesman, and job-shop scheduling problems to enable readers to comprehensively understand our method and to show the usefulness of our Petri net modeling-based approach for quantum annealing.

2.2. Quantum Annealing and Ising Models

Quantum annealing is a metaheuristic for solving combinatorial optimization problems, where it tries to find a value $+1$ or -1 for each of the Ising variables $\boldsymbol{s} = (s_1, s_2, \ldots, s_N)$ such that the following Hamiltonian of the Ising models gives the lowest energy state:

$$H_P(\boldsymbol{s}) = \sum_{i=1}^{N} h_i s_i + \sum_{i<j} J_{i,j} s_i s_j, \tag{1}$$

where h_i is the magnetic field coefficient at site s_i, and $J_{i,j}$ is the interaction coefficient between s_i and s_j. Ising variables correspond to discrete variables in the metaheuristic. The lowest energy state, called the *ground state* of H_P, provides the optimal solution. Figure 1 depicts a two-dimensional array Ising model. A circle represents a spin and is connected to four neighbor spins. The arrows $\uparrow$ and $\downarrow$ represent the *up* spin (+1) and *down* spin (-1), respectively.

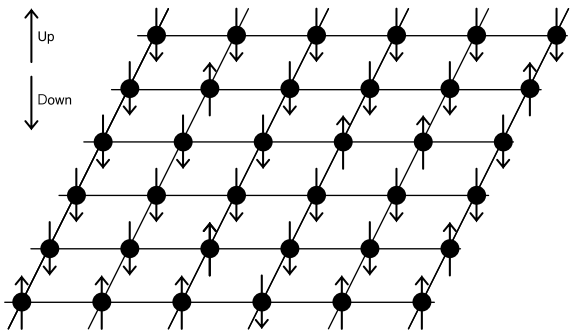

Figure 1. 2D Array Ising Model.

A Hamiltonian may include an objective function, where the lower energy in the terms of the function contributes to a smaller objective value in minimization problems. For maximization problems, we reverse the sign of the Ising model parameters for the objective function and then minimize it. Constraints in optimization problems should also be composed of sets of terms in the Hamiltonian. The lowest energy state of the terms leads to the satisfaction of the corresponding constraints. Therefore, the annealing process needs to obtain feasible solutions that satisfy all constraints with a good quality of the objective value by exploring the lowest energy state of the total Hamiltonian. Note that we can easily replace the type of decision variables from $\{+1, -1\}$ to $\{0, 1\}$, and vice versa.

The mechanism of the quantum annealing process is explained well in [2,4,13]. The following paragraph gives only a brief introduction to the optimization process based on quantum annealing.

An Ising model Hamiltonian, whose ground state encodes the solution of the target optimization problem, can be solved using adiabatic quantum optimization. First, we prepare the following time-dependent Hamiltonian:

$$H(t) = (1 - \frac{t}{T})H_0 + \frac{t}{T}H_P, \tag{2}$$

where H_P is the Hamiltonian corresponding to the target optimization problem as shown in Equation (1); H_0 is another Hamiltonian whose ground state is easily set up in the computing platform. The Hamiltonian varies from $H(0) = H_0$ to $H(T) = H_P$ according to the time progress from $t = 0$ to $t = T$. The adiabatic theorem states that if $H(0)$ is the ground state, then $H(T)$ is also the state under a sufficiently slow change of parameters in the Hamiltonian during the time progress. That is, the solution of the combinatorial optimization is obtained at $H(T)$.

2.3. Petri Net Fundamentals

A Petri net $PN = (P, T, F, \mathbf{M}_0)$ is a directed bipartite graph $N = (P, T, F)$ with initial marking $\mathbf{M}_0$ [16]. The bipartite graph has two types of vertex sets: a set of places $P = \{P_1, P_2, \ldots, P_{|P|}\}$ and a set of transitions $T = \{T_1, T_2, \ldots, T_{|T|}\}$. The arc function $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ defines the connection from places to transitions, and vice versa, where $\mathbb{N}$ is the set of natural numbers. The returned values of F indicate the number of removed tokens from the starting place when $(P \times T) \to \mathbb{N}$ or the number of generated tokens to the ending place when $(T \times P) \to \mathbb{N}$ upon the firing of the transition.

Instead of an arc function, a matrix representation is often used for the connection. Here, *Pre* and *Post* show the incident matrix of size $|P| \times |T|$ to indicate the connection from places to transitions and from transitions to places, respectively. It should be noted that $Pre(P_i, T_j) = F((P_i, T_j))$ and $Post(P_i, T_j) = F((T_j, P_i))$.

Tokens are located in places such that the token distribution on P represents the status of the modeled system. The vector $\mathbf{M}_k$ of size $|P|$ represents the number of tokens in each place $P_i \in P$ at step k. In addition, $\mathbf{M}_0$, called the *initial marking*, is the marking at step 0, which corresponds to the initial status of the system. Therefore, the Petri net $PN = (N, \mathbf{M}_0)$ represents the structure and initial states of the modeled system.

The transition $T_j \in T$ is *enabled* at step k only when $M_k(P_i) \geq Pre(P_i, T_j)$ for each P_i in ${}^\bullet T_j$, where ${}^\bullet T_j$ and $T_j^\bullet$ denote the set of input and output places of T_j, respectively. Similarly, ${}^\bullet P_i$ and $P_i^\bullet$ are the sets of input and output transitions of P_i, respectively. An enabled transition T_j can fire. Firing implies the occurrence of an event in the system. The firing of transition T_j removes $Pre(P_i, T_j)$ tokens from each $P_i \in {}^\bullet T_j$ and adds $Post(P_i, T_j)$ tokens to each $P_i \in T_j^\bullet$. Firing changes the token distribution, which indicates the change in status of the system by the corresponding event. The following mathematical form shows the change in status caused by the firing of transitions at step k:

$$\begin{aligned} \mathbf{M}_{k+1} &= \mathbf{M}_k - Pre \cdot \mathbf{X}_k + Post \cdot \mathbf{X}_k \\ &= \mathbf{M}_k + (Post - Pre) \cdot \mathbf{X}_k, \end{aligned} \tag{3}$$

where $\mathbf{X}_k$ is a vector of size $|T|$ showing the number of times each T_j fires at step k, and $\mathbf{X}_k$ is called the *firing count vector* at step k.

For a quantitative analysis of the dynamical behavior of a system, *time* was introduced to Petri nets [23]. The timing methods can be categorized into three types: firing duration (FD), holding duration (HD), and enabling duration (ED) [23]. The FD assigns time to transitions to represent the firing duration. The HD is referred to as place time Petri nets, where tokens are unavailable for firing for a particular period after being located in the place. In the last method, i.e., the ED, a transition cannot be fired for a given period after being enabled. Although other types can be allowed with our method, this paper focuses on timed Petri nets with the FD.

A timed Petri net is a six-tuple $TPN = (P, T, F, TS, FD, \mathbf{M}_0)$, where TS is the set of time values, and $FD : T \to TS$ is a function that returns the firing duration time of transition $T_j \in T$. In this study, we assume that $TS = \mathbb{N}$ and is a set of natural numbers. A timestamp is also attached to tokens to record the token generation time. In the timed Petri net, the transition T_j is enabled at time k when each input place P_i of T_j has more than or equal to $F(P_i, T_j)$ tokens, and its time stamp is no more than k. By firing T_j at time k, the marking is changed according to the same rule as the Petri net described above, except that we attach the time stamp $k + FD[T_j]$ to each output token.

A colored Petri net is an extension of Petri nets, where we can introduce values (colors) to tokens, guard functions to transitions for their firing conditions, and arc functions to output arcs of transitions to calculate the colors of the output tokens [24]. Therefore, tokens become much more informative and allow us flexible and efficient modeling.

This extended Petri net is denoted by $CPN = (\Sigma, P, T, F, V, C, G, E, \mathbf{M}_0)$, where P, T, and F represent a set of places, transitions, and arcs, respectively. Σ shows a set of color types, and $C : P \to \Sigma$ is a color function for the places. Only tokens with colors specified by $C(P_i)$ are located in place P_i. In addition, V is a set of arc variables, where $\mathrm{Type}(v) \in \Sigma$ for $v \in V$. Moreover, $\mathrm{Type}(v)$ is the type of variable v. Here, E denotes an arc function where $E((T_j, P_i))$ returns tokens on $C(P_i)$ for each output place P_i based on the binding values to each of the arc variables. The term G represents a guard function that returns a Boolean value to determine whether the corresponding transition is enabled. A marking $\mathbf{M}_k$ at step k is a mapping of multiple sets of $C(P_i)$ to each place. Transition T_j is enabled at marking $\mathbf{M}_k$ with binding b when for each input place P_i of T_j,

$$M_k(P_i) \geq E((P_i, T_j))(b) \tag{4}$$

where $E((P_i, T_j))(b)$ denotes the result of the arc function for arc (P_i, T_j) when we bind $b(v)$ for each variable v in $E((P_i, T_j))$. At marking $\mathbf{M}_k$, $\mathbf{M}_{k+1}$ is generated by firing T_j with binding b:

$$M_{k+1}(P_i) = M_k(P_i) - E((P_i, T_j))(b) + E((T_j, P_i))(b). \tag{5}$$

Colored and timed Petri nets are extremely powerful in the sense that complicated systems can be easily modeled. This fact contributes tremendously to modeling combinatorial optimization problems in our research.

Another essential benefit of using Petri net modeling is the well-systematized formulation of behavioral and structural properties. These have been studied in Petri net research [16,17,23]. The behavioral properties includes *reachability*, *boundedness*, *coverability*, *liveness*, and *persistence*. The structural properties, such as *precedence relation*, *structural boundedness* and *conservative*, are initial-marking independent and can be derived from the structure of the bipartite graph. These fundamental properties can be used for Ising model formulation from Petri net models.

3. Binary Quadratic Nets

This section introduces a new Petri net class called *binary quadratic nets* to represent the Ising and QUBO models. The Ising model is a mathematical model in statistical mechanics. The model is a graph in which the vertices correspond to spins, and the edges

interact between spins. Petri nets are also suitable for expressing Ising models because their fundamental components, places, and transitions can naturally represent the states of the spins and their interactions. A token in a place can show the corresponding spin state when we introduce colors $\{-1, +1\}$ to tokens. For QUBO models, the color type of tokens becomes $\{0, 1\}$.

In our Petri net modeling-based Ising model formulation, Petri net models representing target optimization problems were converted into the corresponding binary quadratic nets. In other words, our proposed method is a net transformation from problem-domain Petri nets into binary quadratic nets. Binary quadratic nets are entirely equivalent to Ising or QUBO models.

3.1. Formal Definition

Definition 1. *(Binary Quadratic Net) A binary quadratic net is a colored Petri net denoted by* $BQN = (\hat{\Sigma}, \hat{P}, \hat{T}, \hat{F}, \hat{C}, \hat{w})$*:*

$$\hat{\Sigma} \in \{\{-1, +1\}, \{0, 1\}\} \tag{6}$$

$$\hat{P} = \{\hat{p}_1, \hat{p}_2, ..., \hat{p}_n\} \tag{7}$$

$$\hat{T} = \{\hat{t}_{i,j} \mid \hat{p}_i, \hat{p}_j \in \hat{P}, \{\hat{p}_i, \hat{p}_j\} = \hat{t}^{\bullet}_{i,j} = {}^{\bullet}\hat{t}_{i,j}, i < j\} \tag{8}$$

$$\hat{F} : (\hat{P} \times \hat{T}) \cup (\hat{T} \times \hat{P}) \to \begin{cases} 1 & \text{if } (\hat{p}_i, \hat{t}_{i,j}) \text{ and } (\hat{t}_{i,j}, \hat{p}_j), \hat{t}_{i,j} \in \hat{T} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$\hat{C}(\hat{p}) \in \hat{\Sigma}, \forall \hat{p} \in \hat{P} \tag{10}$$

$$\hat{w} : \hat{P} \cup \hat{T} \to \mathbb{R} \tag{11}$$

There is one token in each place, where a place corresponds to a spin variable in the Ising models and a binary variable in the QUBO models, respectively. For simplicity, we denote the value of a variable associated with place $\hat{p}_i$ by $M(\hat{p}_i)$, that is, $M(\hat{p}_i) \in \{-1, +1\}$ in the Ising models or $M(\hat{p}_i) \in \{0, 1\}$ in the QUBO models. Note that in the QUBO model, $M(\hat{p}_i) = 0$ indicates that the place $\hat{p}_i$ includes a token with color 0.

Current quantum annealing platforms have various physical topologies, chimera graphs for D-waves, and three-dimensional lattices for CMOS annealers [25]. Thus, we need to embed the logical Ising model to a specific graph topology to run the annealing process. However, we do not treat the embedding process in this study because we can use existing embedding algorithms for each platform. Moreover, Fujitsu Digital Annealer allows fully connected topologies. For reference, Figure 2 depicts the binary quadratic net model for the 2D array Ising model shown in Figure 1.

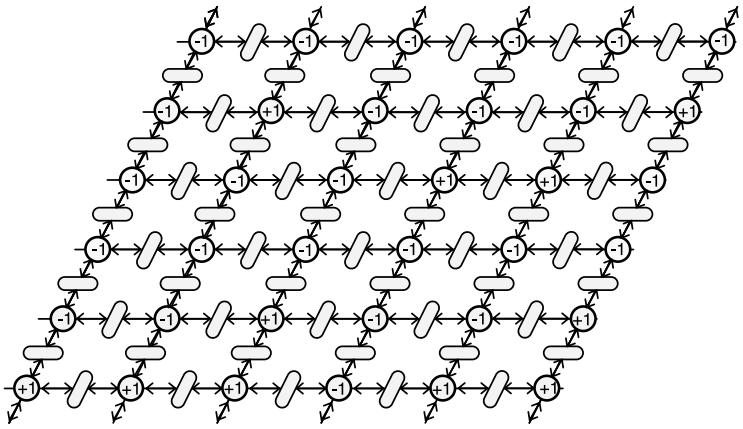

Figure 2. Binary Quadratic Net for 2D-Array Ising Model shown in Figure 1.

To combine binary quadratic nets and Ising models, we introduce a measure to represent the state of binary quadratic nets with a marking **M**. We define this as an energy function of binary quadratic nets corresponding to the Hamiltonian in the Ising models.

Definition 2. *(Energy Function of Binary Quadratic Net) For a binary quadratic net* $BQN = (\hat{\Sigma}, \hat{P}, \hat{T}, \hat{F}, \hat{C}, \hat{w})$ *with a marking* **M**, *the energy function is defined as follows:*

$$H_{BQN}(\mathbf{M}) = \sum_{\hat{p}_i \in \hat{P}} w(\hat{p}_i) M(\hat{p}_i) + \sum_{\hat{t}_{i,j} \in \hat{T}} w(\hat{t}_{i,j}) M(\hat{p}_i) M(\hat{p}_j). \tag{12}$$

The first summed terms show the energy derived from token existence at each place. The second summed term represents the energy from the interaction between tokens on both sides of each transition. The weight parameters $w(\hat{p}_i)$ and $w(\hat{t}_{i,j})$ are defined for $\hat{p}_i$ and $\hat{t}_{i,j}$, respectively. The energy function in Definition 2 is equivalent to the Ising model shown in Equation (1), and the energy function is uniquely defined from the corresponding binary quadratic net. Although it is straightforward, we simply summarize this fact as a proposition for the readability of the remaining parts.

Proposition 1. *For an Ising model, we have an equivalent binary quadratic net BQN in the sense that the energy function* $H_{BQN}(\mathbf{M})$ *is equivalent to the Hamiltonian of the Ising model.*

In our binary quadratic nets, transitions represent interactive relations between tokens on both sides. There may be numerous interaction types. We choose interaction types carefully depending on the target optimization problems. In the Appendix, we summarize the primitive interaction for QUBO and the Ising model, respectively. Table A1 shows the interaction primitives $I_i^{qubo}, i = 0, 1, \ldots, 15$, for two binary variables in $\{0, 1\}$, where $(0, 0), (0, 1), (1, 0)$, and $(1, 1)$ express all possible combinations of the two binary variables. In addition, $I_1^{qubo}, I_7^{qubo}, I_8^{qubo}, I_9^{qubo}$ correspond to the well-known logical functions AND, XOR, OR, and NOR, respectively. Moreover, $I_0^{qubo}, I_{15}^{qubo}$ indicate inconsistency and a tautology, respectively. Table A2 lists the interactions for the Ising model converted from those in Table A1 by applying the following relation.

$$M(\hat{p}_i)^{ising} = 2M(\hat{p}_i)^{qubo} - 1, \tag{13}$$

$$M(\hat{p}_i)^{qubo} = (M(\hat{p}_i)^{ising} + 1)/2, \tag{14}$$

where $M(\hat{p}_i)^{ising}$ and $M(\hat{p}_i)^{qubo}$ denote the marking in the Ising and QUBO models, respectively.

3.2. Binary Quadratic Net Examples

As we describe in Proposition 1, binary quadratic net models are equivalent to the Ising models. We show the binary quadratic net models for well-known graph partitioning and minimum vertex cover problems. These problems are straightforwardly modeled with binary quadratic nets and formulated as marking problems in such nets.

Example 1. *Minimum Vertex Cover: For a given undirected graph* $G = (V, E)$, *with vertex set V and edge set E, a vertex cover set satisfies the condition such that every edge in E is incident on a vertex in the cover set. The problem is to find the minimum vertex cover set. The problem is known as NP-hard [5].*

We choose the QUBO model for the problem, where we generate the place set and transition set of the binary quadratic net under the one-to-one correspondence with the vertex set and the edge set, respectively. We express a vertex cover by a marking, where we place a token with a color of 1 to show that the corresponding vertex is in the vertex cover, and where a color of 0 is not in the cover. To satisfy the feasibility of the vertex cover, we need to avoid a token with a color of 0 in both input places of each transition because the situation does not cover the corresponding edge. We formulate the feasibility of the condition corresponding to I_8^{qubo} *in Table A1 as follows:*

$$H_{constraint}(\mathbf{M}) = \sum_{i<j, \hat{p}_i^\bullet \cap \hat{p}_j^\bullet \neq \emptyset} (1 - M(\hat{p}_i))(1 - M(\hat{p}_j)), \tag{15}$$

To minimize the objective function, we attempt to reduce the number of color 1 tokens in $\mathbf{M}$*. Therefore, the following penalty function is suitable because only color 1 tokens increase the penalty.*

$$H_{cost}(\mathbf{M}) = (\sum_{i=1}^{|P|} M(\hat{p}_i)) \tag{16}$$

Based on the superposition principle, we have the total formulation by combining the binary quadratic nets corresponding to (15) and (16):

$$H_{total}(\mathbf{M}) = A \cdot H_{constraint}(\mathbf{M}) + B \cdot H_{cost}(\mathbf{M}) \tag{17}$$

where $H_{constraint}(\mathbf{M})$ *counts the number of transitions, both of which have a 0 color token. In addition,* $H_{cost}(\mathbf{M})$ *denotes the number of places colored with 1 to minimize the vertex cover. Moreover, A and B show the parameters for a trade-off between the constraint and the objective function. The formulation is equivalent to the QUBO model in [6].*

Consider the graph shown in Figure 3 as a problem instance of the vertex cover problem. We can formulate the problem instance as a marking problem in the corresponding binary quadratic net in Figure 4. Marking $\mathbf{M} = (1, 0, 1, 0, 0, 1, 0, 1)$ *is a feasible solution to the instance.*

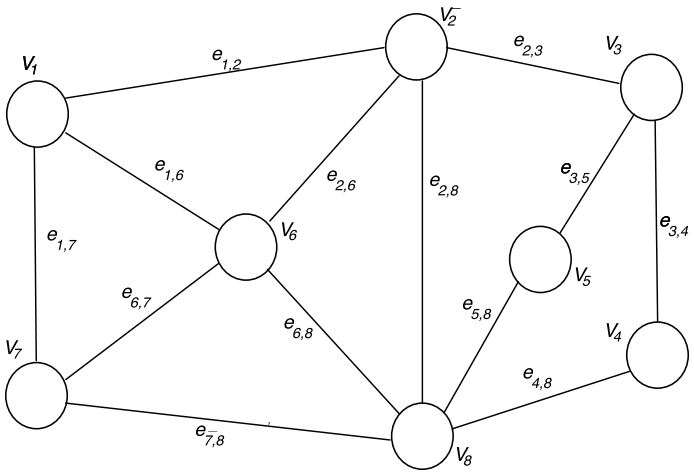

Figure 3. Example of Undirected Graphs.

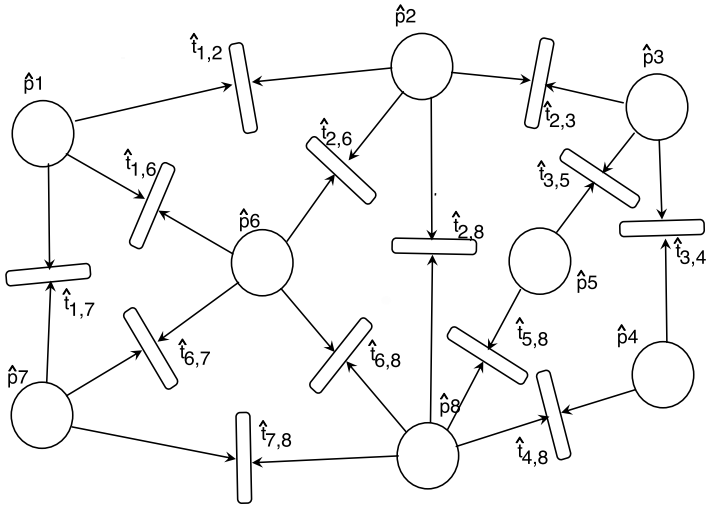

Figure 4. Binary Quadratic Net for Vertex Cover Problem Instance shown in Figure 3.

Example 2. *Graph Partitioning: Graph partitioning is an optimization problem, which is explained as follows: Let us consider an undirected graph $G = (V, E)$ with vertex set V and edge set E, where $|V|$ is the number of vertices. The problem is partitioning V into two subsets whose sizes are equal to $|V|/2$, minimizing the number of edges between the subsets. The problem is known as NP-hard [5].*

We reduce the problem to a marking problem in binary quadratic nets, where each vertex in the given graph corresponds to a place in the binary quadratic net, and the marking such that $M(\hat{p}_i) \in \{-1, +1\}, \forall \hat{p}_i \in \hat{P}$ shows the partition. The objective function minimizes the number of edges connecting the two partitioned groups. The transitions have a one-to-one correspondence with the edges. To minimize the objective function, we want to reduce the different color tokens in pairs of places that share the output transition. We design the objective function with I_6^{Ising} in Table A2 as follows:

$$H_{cost}(\mathbf{M}) = \sum_{i<j, \hat{p}_i^\bullet \cap \hat{p}_j^\bullet \neq \emptyset} \frac{1 - M_0(\hat{p}_i) M_0(\hat{p}_j)}{2} \tag{18}$$

Here, I_6^{Ising} outputs 1 only when both places have different color tokens, that is, XOR.

Second, we consider the constraint of the graph-partitioning problem and the equality in the vertex size of both partitioned groups. Concerning $\{-1, +1\}$ logic, we can design the following energy function for the constraint because minimizing the function leads to a satisfaction of the constraint:

$$H_{constraint}(\mathbf{M}) = (\sum_{i=1}^{|P|} M(\hat{p}_i))^2 \tag{19}$$

The following penalty function shows the total energy function for graph partitioning and is equivalent to the Ising model presented in [6].

$$H_{total}(\mathbf{M}) = A \cdot H_{constraint}(\mathbf{M}) + B \cdot H_{cost}(\mathbf{M}) \tag{20}$$

Figure 5 shows the binary quadratic net for the graph partitioning example, where the subnet composed of the places and the transitions connected by the solid arcs correspond to $H_{cost}(\mathbf{M})$, and the other subnet with all places and transitions connected by the dotted arcs are added based on $H_{constraint}(\mathbf{M})$.

One of the feasible solutions is such that $M(\hat{p}_1) = M(\hat{p}_2) = M(\hat{p}_6) = M(\hat{p}_7) = +1$ and $M(\hat{p}_3) = M(\hat{p}_4) = M(\hat{p}_5) = M(\hat{p}_8) = -1$.

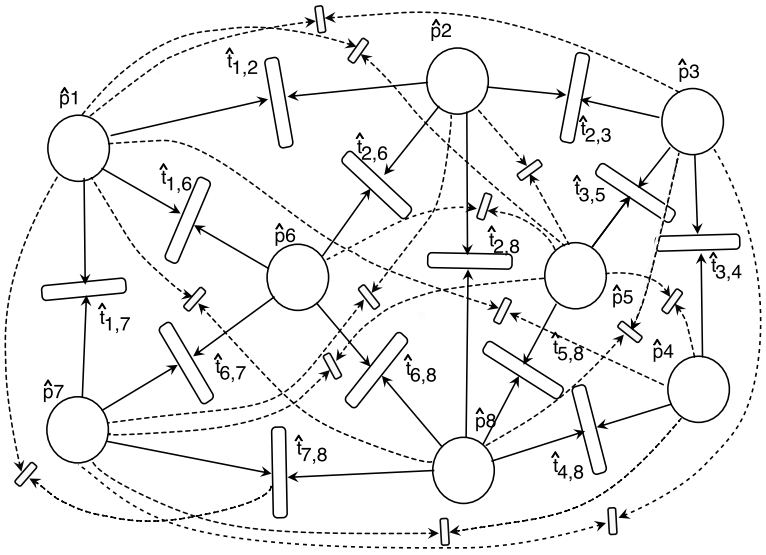

Figure 5. Binary Quadratic Net of Graph Partitioning Problem for Instance shown in Figure 3.

4. Binary Quadratic Net Construction from Problem Domain Petri Nets

In the previous section, we modeled combinatorial optimization problems directly, a minimum vertex cover problem, and a graph partitioning problem with binary quadratic nets and formulated them as marking problems. However, they are exceptional cases. In general, we can meet the conceptual gap between combinatorial optimization problems and target binary quadratic nets. Note that the problem is more serious in the direct formulation of the Ising or QUBO models.

Our approach attempts to minimize the gap by converting problem-domain Petri net models, represented by timed Petri nets and colored Petri nets, into the corresponding binary quadratic nets. This section proposes a method for constructing target binary quadratic nets from problem-domain Petri net models.

4.1. Incremental Construction Based on Superposition Principle

A binary quadratic net can be composed incrementally by combining binary quadratic subnets, each corresponding to a constraint or an objective function in the original optimization problem. This superposition principle of the net structure and weight values simplifies the binary quadratic net construction. The composition is straightforward, where we add the weight values on the places and transitions if the same places and transitions in the subnets are to be combined; otherwise, add new places and transitions with their weight values. In the previous section, we observed this process in the graph partitioning example (Example 2). Even though we did not consider the weight values, we combined the two subnets. The following definition formally represents the binary quadratic net composition.

Definition 3. *(Binary Quadratic Net Composition) For two given binary quadratic nets* $BQN_h = (\hat{\Sigma}, \hat{P}_h, \hat{T}_h, \hat{F}_h, \hat{C}_h, \hat{w}_h)$ *and* $BQN_k = (\hat{\Sigma}, \hat{P}_k, \hat{T}_k, \hat{F}_k, \hat{C}_k, \hat{w}_k)$ *such that the model types of both nets, Ising or QUBO, are the same, the new binary quadratic net* $BQN = (\hat{\Sigma}, \hat{P}, \hat{T}, \hat{F}, \hat{C}, \hat{w})$ *is composed based on the following superposition principle:*

$$\hat{\Sigma} \in \{\{-1, +1\}, \{0, 1\}\} \tag{21}$$

$$\hat{P} = \hat{P}_h \cup \hat{P}_k \tag{22}$$

$$\hat{T} = \hat{T}_h \cup \hat{T}_k \tag{23}$$

$$\hat{F} : (\hat{P} \times \hat{T}) \cup (\hat{T} \times \hat{P}) \to \begin{cases} 1 & (\hat{p}_i, \hat{t}_{i,j}) \text{ and } (\hat{p}_j, \hat{t}_{i,j}), \hat{t}_{i,j} \in \hat{T}, \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

$$\hat{C} : \hat{P} \to \begin{cases} \{-1, +1\} & \text{Ising Model} \\ \{0, 1\} & \text{QUBO Model} \end{cases} \tag{25}$$

$$\hat{w} : \hat{P} \cup \hat{T} \to \begin{cases} w_h(x) + w_k(x) & x \in \hat{P}_h \cup \hat{T}_h \text{ and } x \in \hat{P}_k \cup \hat{T}_k, \\ w_h(x) & x \in \hat{P}_h \cup \hat{T}_h \text{ and } x \notin \hat{P}_k \cup \hat{T}_k, \\ w_k(x) & x \in \hat{P}_k \cup \hat{T}_k \text{ and } x \notin \hat{P}_h \cup \hat{T}_h, \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

The following proposition is a straightforward but essential property for validating our method.

Proposition 2. *(Superposition Property) Let BQN be composed from* BQN_h *and* BQN_k*. The following properties hold.*

$$H_{BQN}(\mathbf{M}) = H_{BQN_h}(\mathbf{M}) + H_{BQN_k}(\mathbf{M}), \quad (\textit{Additivity}), \tag{27}$$

$$H_{A \cdot BQN}(\mathbf{M}) = A \cdot H_{BQN}(\mathbf{M}), \quad (\textit{Homogeneity}), \tag{28}$$

where A is a scalar, and $H_{A\cdot BQN}(\mathbf{M})$ is the energy function of BQN, such that we replace the weight function $\hat{w}$ with $A \cdot \hat{w}$.

Proof. In Definition 3, the energy function is defined by the net structure and weight function $\hat{w}$. Based on the composition rules in Equations (22), (24), and (25), all structural properties of BQN_h and BQN_k are transformed into BQN. Based on the rule for the weight function (26), we can confirm that $\hat{w}(\hat{p}_i)$ and $\hat{w}(\hat{t}_{i,j})$ can be divided into $\hat{w}_h(\hat{p}_i) + \hat{w}_k(\hat{p}_i)$ and $\hat{w}_h(\hat{t}_{i,j}) + \hat{w}_k(\hat{t}_{i,j})$ for the common place $\hat{p}_i$ and transition $\hat{t}_{i,j}$, respectively. Therefore, the additivity in Equation (27) holds.

The homogeneity property with a degree of 1 is given by Definition 2 if we replace the weight function $\hat{w}$ with $A \cdot \hat{w}$. □

Owing to the additivity and homogeneity properties in Proposition 2, we have the following corollary:

Corollary 1. *Binary quadratic nets can be composed by the incremental application of Definition 3, in which we can scale the weight function $\hat{w}$ in subnets with a constant factor.*

In our approach, we construct a target binary quadratic net based on the incremental compositions of binary quadratic subnets. Each subnet is converted from a property of the Petri net model representing the optimization problems. We call the Petri net *problem-domain Petri net*. The properties of the problem-domain Petri net models are expressed with marking or firing sequences. To focus on markings or firing sequences, we employ marking-based or firing-based constructions. In the following subsections, we assume that binary quadratic nets are QUBO models unless otherwise stated, but can be converted into Ising models by using the conversion rule (13).

4.2. Marking-Based Construction

Let us denote a problem-domain Petri net for a target optimization problem by $N = (P, T, F)$ with $P = \{P_1, P_2, \ldots, P_n\}$ and $T = \{T_1, T_2, \ldots, T_m\}$. Marking $M_k(P_j)$ represents a set of tokens in place P_j at time step k. In addition, $\mathbf{M}_k$ is a vector of size $|P|$, showing a token distribution on the Petri net at time step k. The marking trajectory of a Petri net model, $\mathbf{M}_0, \mathbf{M}_1, \ldots, \mathbf{M}_K$ denotes the status changes triggered by the firing of transitions.

In the marking-based construction of binary quadratic nets, we represent a marking trajectory of the problem domain Petri net by the place set of the target binary quadratic net. If each place P_i in the problem-domain Petri net has at most one token at any time and the maximum step is K, we initially prepare the following place set of the target binary quadratic net.

$$\hat{P} = \{\hat{p}_i^k | P_i \in P, k = 0, 1, 2, \ldots, K\} \tag{29}$$

If $M_k(P_i)$ is a natural number, that is, more than one token or a color token with a natural number value is possible in each place in P, we need to prepare more places because each place in the binary quadratic nets has one token with color in $\{0, 1\}$ or $\{-1, +1\}$.

$$\hat{P} = \{\hat{p}_{i,n}^k | P_i \in P, n = 0, 1, 2, \ldots, N, k = 0, 1, 2, \ldots, K\}, \tag{30}$$

where N is the possible maximum value of $M_k(P_i)$. At the same time, we require a one-hot constraint because only one place between $\hat{p}_{i,n}^k$ for $n = 0, 1, \ldots, N$ for each i and k should be marked with 1, and the others should be marked with 0:

$$\sum_{n=0}^{N} M(\hat{p}_{i,n}^k) = 1, \forall i, k \tag{31}$$

As the energy function representation, we need the following:

$$H_{onehot}(\mathbf{M}) = (\sum_{n=0}^{N} M(\hat{p}_{i,n}^{k}) - 1)^2, \forall i, k \tag{32}$$

Note that we can obtain the corresponding binary quadratic net from an energy function.

4.2.1. Boundedness

The boundedness is an essential characteristic to ensure avoiding an overflow in the system behavior. In the Petri net theory, we can express this characteristic as follows.

$$M_k(P_i) \leq U_i, \forall i, k \tag{33}$$

where U_i is the upper bound for P_i. We can convert the constraint into a binary quadratic net and represent it as the energy function under the one-hot constraint (32):

$$H_{boundedness}(\mathbf{M}) = \sum_{k=0}^{K+1} \sum_{i=1}^{|P|} (\sum_{n=0}^{N} nM(\hat{p}_{i,n}^{k}) - U_i)^2, \tag{34}$$

Function (34) is sufficient for the equality constraint $M_k(P_i) = U_i$, but not for the upper bound. Therefore, we improve the function by introducing ancilla places, $\hat{u}_{i,m}, i = 1, 2, \ldots, |P|$, $m = 0, 1, \ldots, U_i$. This technique is well known in the Ising model formulation [6,14].

$$H_{boundedness}(\mathbf{M}) = \sum_{k=0}^{K+1} \sum_{i=1}^{|P|} (\sum_{n=0}^{N} nM(\hat{p}_{i,n}^{k}) - \sum_{m=0}^{U_i} m\hat{u}_{i,m})^2 + (\sum_{m=0}^{U_i} \hat{u}_{i,m} - 1)^2 \tag{35}$$

The upper bound constraint appears in numerous optimization problems. The knapsack constraint is a well-known example of this constraint for the knapsack place. We can also express the boundedness of specific places P_i by removing the other places from the function.

4.2.2. Invariant

The boundedness shown in Equation (33) denotes inequality constraints. The invariance leads to equality constraints based on markings. Note that the invariance is different from the structural invariance of the net theory. The behavioral invariance requires that the total weighted sum of the tokens be equal among the firing sequences. The sum may be the cost required for the resource to operate the system. The following constraint shows that the weighted sum becomes W for all k.

$$\sum_{i=1}^{|P|} h_i M_k(P_i) = W, k = 0, 1, \ldots, K. \tag{36}$$

We can convert the constraint into a binary quadratic net and represent it as the energy function under the one-hot constraint (32):

$$H_{invariant}(\mathbf{M}) = \sum_{k=0}^{K+1} (\sum_{i=1}^{|P|} \sum_{n=0}^{N} h_i nM(\hat{p}_{i,n}^{k}) - W)^2 \tag{37}$$

4.3. Firing-Based Construction

A firing count vector $\mathbf{X}_k$ represents the firing counts of each transition in a Petri net model at step k, where $X_k(T_j)$ denotes the firing counts of transition T_j at time step k. Elements of $\mathbf{X}_k$ are usually one of $\{0, 1\}$; that is, transitions can fire only once at each time step. We call this single firing restriction *single-server semantics* in Petri net theory. However,

any natural number values (allowing more than one) are also possible with *infinite server semantics*. For the single-server semantics, we generate a place set $\hat{P}$ in the target binary quadratic net, where $M(\hat{p}_i^k)$ corresponds to $X_k(T_i)$:

$$\hat{P} = \{\hat{p}_i^k | T_i \in T, k = 0, 1, 2, \ldots, K\} \tag{38}$$

Similarly, we can extend the single-firing semantics to the N times firing semantics. Note that an infinite number of firing times is possible mathematically but impossible practically. Thus, we restrict the maximum number reasonably to N.

$$\hat{P} = \{\hat{p}_{i,n}^k | T_i \in P, n = 0, 1, 2, \ldots, N, k = 0, 1, 2, \ldots, K\}, \tag{39}$$

As the energy function representation, we need the same one-hot constraint (32).

4.3.1. Resource Conflict

If transitions T_i and T_j have a common input place and conflict with a single token, $X_k(T_i) = X_k(T_j) = 1$ cannot be allowed. Therefore, we need the following energy function:

$$H_{conflict}(\mathbf{M}) = \sum_{k=0}^{K} \sum_{(T_i, T_j) \in C} M(\hat{p}_i^k) M(\hat{p}_j^k), \tag{40}$$

where C is a set of conflict transitions. In addition, C can be extracted from the problem-domain Petri net.

In timed Petri net models, we need to consider the firing duration. Let $N = (P, T, F, TS, FD)$ with $P = \{P_1, P_2, \ldots, P_n\}$ and $T = \{T_1, T_2, \ldots, T_m\}$ be a timed Petri net, where $FD : T \to \mathbb{N}$ is a function that returns the firing duration. We then extend the resource conflict in the stepwise firing into the timed firing.

$$H_{conflict}(\mathbf{M}) = \sum_{(T_i, T_j, k, h) \in C^{timed}} M(\hat{p}_i^k) M(\hat{p}_j^h), \tag{41}$$

$$C^{timed} = \{(T_i, T_j, k, h) | \forall (T_i, T_j) \in C, h \leq k + FD(T_i) \text{ or } k \leq h + FD(T_j)\}, \tag{42}$$

where C^{timed} is the timed conflict set such that conflict transitions cannot fire until the firing duration is complete. We can obtain C^{timed} from the given timed Petri net.

4.3.2. Firing Count

In some applications, we must specify the number of firing occurrences for each transition.

$$\sum_{k=0}^{K} X_k(T_i) = FC_i, \forall i \tag{43}$$

where FC_i is the specified number of firings of T_i.

We can convert the constraint into a binary quadratic net and represent it as the energy function under the one-hot constraint (32):

$$H_{firings}(\mathbf{M}) = \sum_{i=1}^{|T|} (\sum_{k=0}^{K} \sum_{n=0}^{N} n M(\hat{p}_{i,n}^k) - FC_i)^2 \tag{44}$$

Note that $M(\hat{p}_{i,n}^k)$ in the binary quadratic net corresponds to $X_k(T_i)$ in the problem-domain Petri net.

Assuming that each transition T_i should fire exactly once during $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_K$, the function (43) can be represented as follows:

$$H_{firings}(\mathbf{M}) = \sum_{i=1}^{|T|}(\sum_{k=0}^{K} M(\hat{p}_i^k) - 1)^2 \tag{45}$$

In practical cases, this constraint is commonly used.

4.3.3. Precedence Relation

Let us assume again that each transition T_i should fire exactly once during $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_K$. We consider the precedence relation between the firing of transitions. If T_i precedes structurally to T_j, that is, $T_i^\bullet \subseteq {}^\bullet T_j$ and ${}^\bullet T_i \not\subseteq T_j^\bullet$, the following penalty function is required.

$$H_{precedence}(\mathbf{M}) = \sum_{k=0}^{K} \sum_{(T_i,T_j)\in Prec, h\leq k} M(\hat{p}_i^k)M(\hat{p}_j^h), \tag{46}$$

where *Prec* is the set of precedence relations. Here, *Prec* can be extracted from the problem-domain Petri net. Similar to the firing conflict, we consider timed Petri nets by introducing the firing duration.

$$H_{precedence}(\mathbf{M}) = \sum_{k=0}^{K}(\sum_{(T_i,T_j)\in Prec, h\leq k+FD(T_i)} M(\hat{p}_i^k)M(\hat{p}_j^h)) \tag{47}$$

4.4. Application Example 1 (Marking-Based Construction): Traveling Salesman Problems

We consider the traveling salesman problem as an example of a marking-based construction. Let $N = (P, T, F, TS, FD)$ with $P = \{P_1, P_2, \ldots, P_n\}$ and $T = \{T_{i,j} | \forall (P_i, P_j) \in P \times P\}$ be a timed Petri net, where $FD : T \to \mathbb{N}$ is a function that returns the firing duration. Each place P_i corresponds to a city, and the transition $T_{i.j}$ denotes the path from P_i to P_j. The firing of $T_{i,j}$ indicates the movement from P_i to P_j. The initial marking $\mathbf{M}_0$ includes only one token at P_1.

Because the salesman visits each city only once from the problem definition,

$$\sum_{k=0}^{|P|-1} M_k(P_i) = 1, \forall P_i \in P. \tag{48}$$

In addition, because the salesman should be at one place for each step,

$$\sum_{i=1}^{|P|} M_k(P_i) = 1, k = 0, 1, \ldots, |P| - 1. \tag{49}$$

The objective function is to minimize the total time to visit all cities and return to the starting place.

$$distance = \sum_{i=1}^{|P|}\sum_{j=1}^{|P|}(FD(T_{i,j}) \cdot \sum_{k=0}^{|P|-1} M_k(P_i)M_{k+1}(P_j)) \tag{50}$$

By converting the constraints and objective function in the problem-domain Petri net into the binary quadratic nets, we have

$$H_{visitingOnce}(\mathbf{M}) = \sum_{i=1}^{|P|}(\sum_{k=0}^{|P|-1} M(\hat{p}_i^k) - 1)^2 \tag{51}$$

$$H_{singleness}(\mathbf{M}) = \sum_{k=0}^{|P|-1}(\sum_{i=1}^{|P|} M(\hat{p}_i^k) - 1)^2 \tag{52}$$

$$H_{distance}(\mathbf{M}) = \sum_{i=1}^{|P|}\sum_{j=1}^{|P|}(FD(T_{i,j}) \cdot \sum_{k=0}^{|P|-1} M(\hat{p}_i^k)M(\hat{p}_j^{k+1})). \tag{53}$$

The total binary quadratic net is as follows:

$$H_{total}(\mathbf{M}) = A \cdot H_{visitingOnce}(\mathbf{M}) + B \cdot H_{singleness}(\mathbf{M}) + C \cdot H_{distance}(\mathbf{M}), \tag{54}$$

where A, B, and C denote the scale factors of the corresponding subnets. The formulation is equivalent to the QUBO model in [6].

4.5. Application Example 2 (Firing-Based Construction): Job-Shop Scheduling Problems

We implemented our method by incorporating CPNTools [24], SNAKES[26], and PyQUBO [14]. As an example, we modeled a simple job-shop scheduling problem with three jobs, four tasks per job, and three shared resources. Figure 6 shows the model drawn by the GUI software of CPNTools. Three sequential systems,

$$p0, t0, p1, t1, p2, t2, p3, t3, p4,$$
$$p5, t4, p6, t5, p7, t6, p8, t7, p9,$$
$$p10, t8, p11, t9, p12, t10, p13, t11, p14,$$

represent the jobs, and places $m0, m1, m2$ represent the resources. Our software reads the model, represents the problem-domain Petri net model with SNAKES, and then extracts the necessary information to convert it into the target binary quadratic net.

The following are the energy functions of the binary quadratic subnets corresponding to the precedence relation constraint between tasks in each job, the resource conflict constraint for each shared resource, and the firing count constraint for each task.

$$H_{precedence}(\mathbf{M}) = \sum_{k=0}^{MaxTime}(\sum_{(T_i,T_j)\in Prec, h\le k+FD(T_i)} M(\hat{p}_i^k)M(\hat{p}_j^h)) \tag{55}$$

$$H_{conflict}(\mathbf{M}) = \sum_{(T_i,T_j,k,h)\in C^{timed}} M(\hat{p}_i^k)M(\hat{p}_j^h), \tag{56}$$

$$H_{firings}(\mathbf{M}) = \sum_{i=1}^{|T|}(\sum_{k=0}^{MaxTime} M(\hat{p}_i^k) - 1)^2. \tag{57}$$

The precedence set $Prec$ in Equation (55) is easily constructed by extracting the precedence relation in each sequential system (job). We can extract the timed conflict set C^{timed} in Equation (56) from the connection between the transitions and resource places and the firing duration FD. $MaxTime$ is the only parameter we need to set before the optimization process. This value indicates the delivery time deadline. The total binary quadratic net is as follows:

$$H_{total}(\mathbf{M}) = A \cdot H_{precedence}(\mathbf{M}) + B \cdot H_{conflict}(\mathbf{M}) + C \cdot H_{firings}(\mathbf{M}) \tag{58}$$

Note that this formulation is used to obtain a feasible solution; however, we can extend this to the optimization by incorporating it with a binary search to find the minimum

MaxTime. This formulation is equivalent to the QUBO model in [27]. Figure 7 shows the schedule obtained through the annealing process using a Fujitsu Digital Annealer.

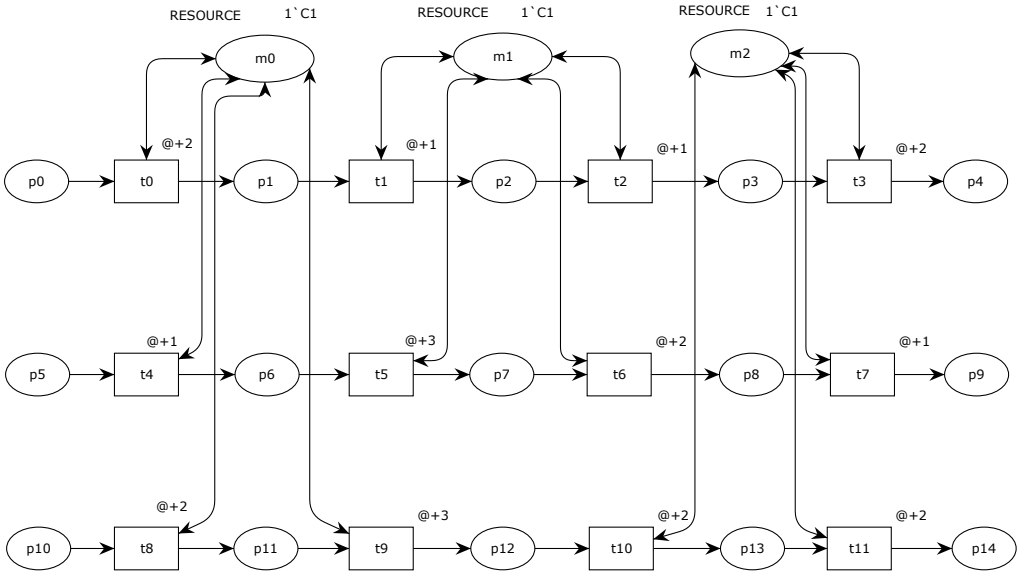

Figure 6. Colored Timed Petri Net Model for Job-Shop Scheduling created by using CPNTools [24].

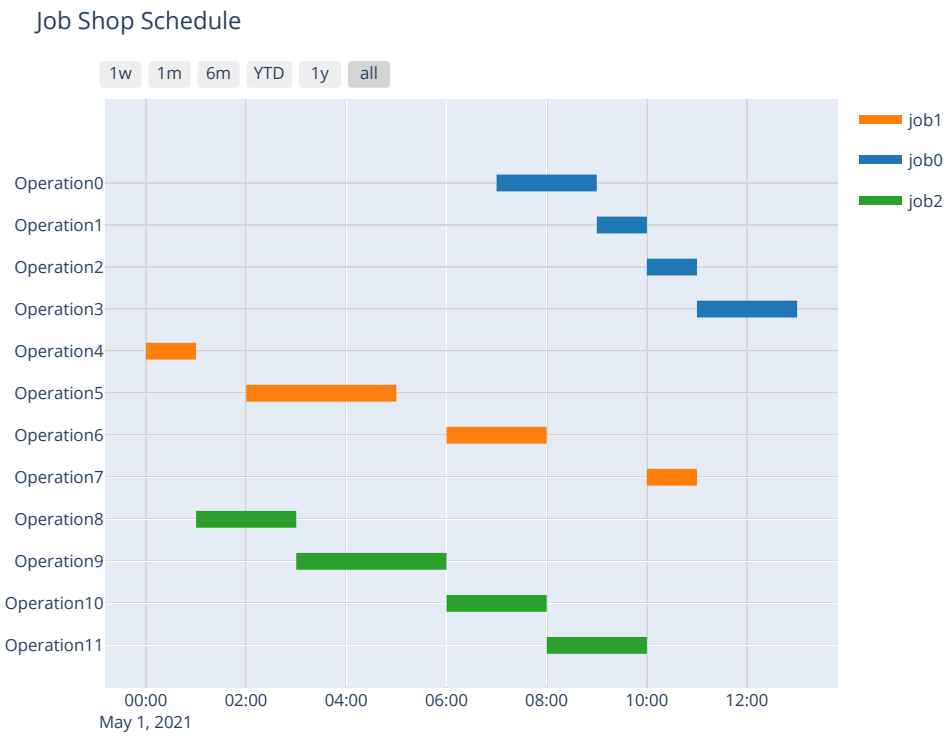

Figure 7. Job-Shop Schedule obtained with the Fujitsu Digital Annealer.

5. Conclusions

This paper proposes a Petri net modeling approach to Ising model formulation for quantum annealing. A Petri net is an outstanding method for modeling target optimization problems because it can represent the structure and behavior of problem instances using simple rules; the color and time extension provides powerful modeling capability. Moreover, for Ising model formulation, we can use various properties extracted systematically from Petri net models. These properties are well-defined in the Petri net theory.

We introduced a new class of colored Petri nets, binary quadratic nets, which are equivalent to the Ising models. Thus, our modeling approach has the following steps:

Step 1: Model the target combinatorial optimization problem with a problem-domain Petri net such as a timed and colored Petri net;
Step 2: Extract constraints and objective functions as properties from problem-domain Petri net models and construct a binary quadratic net incrementally;
Step 3: Convert the binary quadratic net to the specified format of the corresponding quantum annealing machine;
Step 4: Invoke the annealing process with the converted Hamiltonian.

Although our method requires users to model their optimization problems with problem-domain Petri nets, this process can be carried out in a relatively straightforward manner if we know the target problem and the simple Petri net modeling rules. Therefore, we can drastically reduce the difficulty of the Ising model formulation.

We implemented our method with Python incorporated using well-known Petri net tools, CPNTools, and SNAKES. We can systematically generate the Ising models for optimization problems, such as scheduling, vehicle routing, and portfolio optimization, once we model the target optimization problems with Petri nets.

In the future, we aim to introduce a mechanism to reduce the Ising variables and their interaction densities because they can affect the performance of quantum annealing. Petri net reachability analysis can contribute to this process as a pre-processing step for quantum annealing. The binary quadratic net model can also be used to analyze the quantum annealing process by attaching an additional subnet that simulates the annealing process. This tool may contribute to parameter tuning for annealing, which is an essential task used to expand the quantum annealing technology, mentioned in the Introduction.

Author Contributions: Conceptualization, M.N.; methodology, M.N. and K.K.; software, K.K.; validation, T.Y. and M.N.; formal analysis, M.N. and K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Interaction Primitives

Table A1 shows the interaction primitives $I_i^{qubo}, i = 0, 1, \ldots, 15$ for two binary variables in $\{0, 1\}$, where $(0, 0), (0, 1), (1, 0)$, and $(1, 1)$ express all possible combinations of the two binary variables. In addition, $I_1^{qubo}, I_7^{qubo}, I_8^{qubo}, I_9^{qubo}$ correspond to AND, XOR, OR, and NOR, respectively. Moreover, $I_0^{qubo}, I_{15}^{qubo}$ are the inconsistency and tautology, respectively. The others also show important logical functions.

Table A2 shows the interaction primitives for the Ising model converted from Table A1 by using the following relation.

$$M(\hat{p}_i)^{ising} = 2M(\hat{p}_i)^{qubo} - 1 \tag{A1}$$

$$M(\hat{p}_i)^{qubo} = (M(\hat{p}_i)^{ising} + 1)/2 \tag{A2}$$

Table A1. Interaction Primitives in QUBO Models.

	(0, 0)	**(0, 1)**	**(1, 0)**	**(1, 1)**	**Energy Function**
I_0^{qubo}	0	0	0	0	0
I_1^{qubo}	0	0	0	1	$M(\hat{p}_i)M(\hat{p}_j)$
I_2^{qubo}	0	0	1	0	$M(\hat{p}_i)(1-M(\hat{p}_j))$
I_3^{qubo}	0	0	1	1	$M(\hat{p}_i)$
I_4^{qubo}	0	1	0	0	$(1-M(\hat{p}_i))M(\hat{p}_j)$
I_5^{qubo}	0	1	0	1	$M(\hat{p}_j)$
I_6^{qubo}	0	1	1	0	$M(\hat{p}_i)+M(\hat{p}_j)-2M(\hat{p}_i)M(\hat{p}_j)$
I_7^{qubo}	0	1	1	1	$M(\hat{p}_i)+M(\hat{p}_j)-M(\hat{p}_i)M(\hat{p}_j)$
I_8^{qubo}	1	0	0	0	$1-M(\hat{p}_i)-M(\hat{p}_j)+M(\hat{p}_i)M(\hat{p}_j)$
I_9^{qubo}	1	0	0	1	$1-M(\hat{p}_i)-M(\hat{p}_j)+2M(\hat{p}_i)M(\hat{p}_j)$
I_{10}^{qubo}	1	0	1	0	$1-M(\hat{p}_j)$
I_{11}^{qubo}	1	0	1	1	$1-M(\hat{p}_j)+M(\hat{p}_i)M(\hat{p}_j)$
I_{12}^{qubo}	1	1	0	0	$1-M(\hat{p}_i)$
I_{13}^{qubo}	1	1	0	1	$1-M(\hat{p}_i)+M(\hat{p}_i)M(\hat{p}_j)$
I_{14}^{qubo}	1	1	1	0	$1-M(\hat{p}_i)M(\hat{p}_j)$
I_{15}^{qubo}	1	1	1	1	1

x and y in $(x,y), x,y \in \{0,1\}$, show $M(\hat{p}_i)$ and $M(\hat{p}_j)$, respectively. A value of 1 and 0 in each cell represents a preferable and an un-preferable interaction, respectively.

Table A2. Interaction Primitives in Ising Models.

	(−1, −1)	**(−1, +1)**	**(+1, −1)**	**(+1, +1)**	**Energy Function**
I_0^{Ising}	0	0	0	0	0
I_1^{Ising}	0	0	0	1	$\frac{1}{4}(M(\hat{p}_i)+1)(M(\hat{p}_j)+1)$
I_2^{Ising}	0	0	1	0	$\frac{1}{4}(M(\hat{p}_i)+1)(-M(\hat{p}_j)+1)$
I_3^{Ising}	0	0	1	1	$\frac{1}{2}(M(\hat{p}_i)+1)$
I_4^{Ising}	0	1	0	0	$\frac{1}{4}(-M(\hat{p}_i)+1)(M(\hat{p}_j)+1)$
I_5^{Ising}	0	1	0	1	$\frac{1}{2}(M(\hat{p}_j)+1)$
I_6^{Ising}	0	1	1	0	$\frac{1}{2}(1-M(\hat{p}_i)M(\hat{p}_j))$
I_7^{Ising}	0	1	1	1	$\frac{1}{4}(M(\hat{p}_i)+M(\hat{p}_j)-M(\hat{p}_i)M(\hat{p}_j)+3)$
I_8^{Ising}	1	0	0	0	$\frac{1}{4}(-M(\hat{p}_i)-M(\hat{p}_j)+M(\hat{p}_i)M(\hat{p}_j)+1)$
I_9^{Ising}	1	0	0	1	$\frac{1}{2}(M(\hat{p}_i)M(\hat{p}_j)+1)$
I_{10}^{Ising}	1	0	1	0	$\frac{1}{2}(1-M(\hat{p}_j))$
I_{11}^{Ising}	1	0	1	1	$\frac{1}{4}(M(\hat{p}_i)-M(\hat{p}_j)+M(\hat{p}_i)M(\hat{p}_j)+3)$
I_{12}^{Ising}	1	1	0	0	$\frac{1}{2}(1-M(\hat{p}_i))$
I_{13}^{Ising}	1	1	0	1	$\frac{1}{4}(-M(\hat{p}_i)+M(\hat{p}_j)+M(\hat{p}_i)M(\hat{p}_j)+3)$
I_{14}^{Ising}	1	1	1	0	$\frac{1}{4}(-M(\hat{p}_i)-M(\hat{p}_j)-M(\hat{p}_i)M(\hat{p}_j)+3)$
I_{15}^{Ising}	1	1	1	1	1

x and y in $(x,y), x,y \in \{-1,+1\}$, show $M(\hat{p}_i)$ and $M(\hat{p}_j)$, respectively. A value of 1 and 0 in each cell represents a preferable and an un-preferable interaction, respectively.

References

1. Kadowaki, T.; Nishimori, H. Quantum annealing in the transverse Ising model. *Phys. Rev. E* **1998**, *58*, 5355–5363. [CrossRef]
2. Farhi, E.; Goldstone, J.; Gutmann, S.; Lapan, J.; Lundgren, A.; Preda, D. A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem. *Science* **2001**, *292*, 472–475. [CrossRef]
3. Morita, S.; Nishimori, H. Mathematical foundation of quantum annealing. *J. Math. Phys.* **2008**, *49*, 125210. [CrossRef]
4. Johnson, M.W.; Amin, M.H.S.; Gildert, S.; Lanting, T.; Hamze, F.; Dickson, N.; Harris, R.; Berkley, A.J.; Johansson, J.; Bunyk, P.; et al. Quantum annealing with manufactured spins. *Nature* **2011**, *473*, 194–198. [CrossRef]
5. Garey, M.R.; Johnson, D.S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*; W. H. Freeman & Co.: New York, NY, USA, 1990.
6. Lucas, A. Ising formulations of many NP problems. *Front. Phys.* **2014**, *2*, 5. [CrossRef]
7. Grant, E.; Humble, T.S.; Stump, B. Benchmarking Quantum Annealing Controls with Portfolio Optimization. *Phys. Rev. Appl.* **2021**, *15*, 014012. [CrossRef]
8. Stollenwerk, T.; O'Gorman, B.; Venturelli, D.; Mandrà, S.; Rodionova, O.; Ng, H.; Sridhar, B.; Rieffel, E.G.; Biswas, R. Quantum Annealing Applied to De-Conflicting Optimal Trajectories for Air Traffic Management. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 285–297. [CrossRef]
9. Ikeda, K.; Nakamura, Y.; Humble, T.S. Application of Quantum Annealing to Nurse Scheduling Problem. *Sci. Rep.* **2019**, *9*, 12837. [CrossRef]
10. Chen, H.; Lidar, D.A. Why and When Pausing is Beneficial in Quantum Annealing. *Phys. Rev. Appl.* **2020**, *14*, 014100. [CrossRef]
11. Graß, T. Quantum Annealing with Longitudinal Bias Fields. *Phys. Rev. Lett.* **2019**, *123*, 120501. [CrossRef]
12. Brady, L.T.; Baldwin, C.L.; Bapat, A.; Kharkov, Y.; Gorshkov, A.V. Optimal Protocols in Quantum Annealing and Quantum Approximate Optimization Algorithm Problems. *Phys. Rev. Lett.* **2021**, *126*, 070505. [CrossRef]
13. Hauke, P.; Katzgraber, H.G.; Lechner, W.; Nishimori, H.; Oliver, W.D. Perspectives of quantum annealing: methods and implementations. *Rep. Prog. Phys.* **2020**, *83*, 054401. [CrossRef] [PubMed]
14. Tanahashi, K.; Takayanagi, S.; Motohashi, T.; Tanaka, S. Application of Ising Machines and a Software Development for Ising Machines. *J. Phys. Soc. Jpn.* **2019**, *88*, 061010. [CrossRef]
15. Waidyasooriya, H.M.; Hariyama, M. A GPU-Based Quantum Annealing Simulator for Fully-Connected Ising Models Utilizing Spatial and Temporal Parallelism. *IEEE Access* **2020**, *8*, 67929–67939. [CrossRef]
16. Murata, T. Petri nets: Properties, analysis and applications. *Proc. IEEE* **1989**, *77*, 541–580. [CrossRef]
17. David, R.; Alla, H. *Discrete, Continuous, and Hybrid Petri Nets*, 2nd ed.; Springer Publishing Company, Incorporated: Heidelberg, Germany, 2010.
18. Richard, P. Modelling integer linear programs with petri nets*. *RAIRO-Oper. Res.* **2000**, *34*, 305–312. [CrossRef]
19. Nakamura, M.; Tengan, T.; Yoshida, T. A Petri Net Approach to Generate Integer Linear Programming Problems. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2019**, *E102.A*, 389–398. [CrossRef]
20. Porco, A.V.; Ushijima, R.; Nakamura, M. Automatic Generation of Mixed Integer Programming for Scheduling Problems Based on Colored Timed Petri Nets. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2018**, *E101.A*, 367–372. [CrossRef]
21. Hoos, H.H.; Stützle, T. 1-Introduction. In *Stochastic Local Search*; Hoos, H.H., Stützle, T., Eds.; The Morgan Kaufmann Series in Artificial Intelligence; Morgan Kaufmann: San Francisco, CA, USA, 2005; pp. 13–59. [CrossRef]
22. Gendreau, M.; Potvin, J.Y. Metaheuristics in Combinatorial Optimization. *Ann. Oper. Res.* **2005**, *140*, 189–213. [CrossRef]
23. van der Aalst, W.M.P. Petri net based scheduling. *Oper. Res. Spektrum* **1996**, *18*, 219–229. [CrossRef]
24. Jensen, K.; Kristensen, L.M.; Wells, L. Coloured Petri Nets and CPN Tools for modeling and validation of concurrent systems. *Int. J. Softw. Tools Technol. Transf.* **2007**, *9*, 213–254. [CrossRef]
25. Oku, D.; Terada, K.; Hayashi, M.; Yamaoka, M.; Tanaka, S.; Togawa, N. A Fully-Connected Ising Model Embedding Method and Its Evaluation for CMOS Annealing Machines. *IEICE Trans. Inf. Syst.* **2019**, *E102.D*, 1696–1706. [CrossRef]
26. Pommereau, F. SNAKES: A Flexible High-Level Petri Nets Library (Tool Paper). In *Application and Theory of Petri Nets and Concurrency*; Devillers, R., Valmari, A., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 254–265.
27. Venturelli, D.; Marchand, D.J.J.; Rojo, G. Quantum Annealing Implementation of Job-Shop Scheduling. *arXiv* **2016**, arXiv:1506.08479.