

Quick design of feasible tensor networks for constrained combinatorial optimization

Hyakka Nakada^{1,2}, Kotaro Tanahashi¹, and Shu Tanaka^{2,3,4,5}

¹Recruit Co., Ltd., Tokyo 100-6640, Japan

²Graduate School of Science and Technology, Keio University, Kanagawa 223-8522, Japan

³Department of Applied Physics and Physico-Informatics, Keio University, Kanagawa 223-8522, Japan

⁴Keio University Sustainable Quantum Artificial Intelligence Center (KSQAIC), Keio University, Tokyo 108-8345, Japan

⁵Human Biology-Microbiome-Quantum Research Center (WPI-Bio2Q), Keio University, Tokyo 108-8345, Japan

Quantum computers are expected to enable fast solving of large-scale combinatorial optimization problems. However, their limitations in fidelity and the number of qubits prevent them from handling real-world problems. Recently, a quantum-inspired solver using tensor networks has been proposed, which works on classical computers. Particularly, tensor networks have been applied to constrained combinatorial optimization problems for practical applications. By preparing a specific tensor network to sample states that satisfy constraints, feasible solutions can be searched for without the method of penalty functions. Previous studies have been based on profound physics, such as $U(1)$ gauge schemes and high-dimensional lattice models. In this study, we devise to design feasible tensor networks using elementary mathematics without such a specific knowledge. One approach is to construct tensor networks with nilpotent-matrix manipulation. The second is to algebraically determine tensor parameters. We showed mathematically that such feasible tensor networks can be constructed to accommodate various types of constraints. For the principle verification, we numerically constructed a feasible tensor network for facility location problem, to find much faster construction than conventional methods. Then, by performing imaginary time evolution, feasible solutions were always obtained, ultimately leading to the optimal solution.

Hyakka Nakada: hyakka_nakada@r.recruit.co.jp

1 Introduction

Combinatorial optimization is the process of identifying a set of discrete variables that minimizes or maximizes an objective function. Numerous real-world problems can be viewed as combinatorial optimization, which has a significant academic and industrial importance. In modern society, as the amount of data traffic increases due to technological advances, so does the size of the combinatorial optimization problems to be solved. Therefore, fast optimization solvers are widely researched, such as integer programming and metaheuristics. In recent years, quantum-gate-type computers have been attracting much attention and are expected to solve combinatorial optimization problems on such a large scale that they cannot be handled by classical computers.

However, the current quantum-gate hardware has a small number of quantum bits and errors during execution. Such hardware is called Noisy Intermediate-Scale Quantum (NISQ) devices [1] and solving practical problems remains challenging. This leads to motivating many quantum-inspired classical algorithms to appear for baseline, approximate, and alternative methodologies in classical computers. For example, methods using Tensor Networks (TNs) [2–4] have been proposed. TNs can approximate quantum simulations by limiting the coefficients of quantum states into the form of tensor products and utilizing singular value decomposition. Thus, it is possible to solve combinatorial optimization problems at a scale beyond the capabilities of current NISQ devices.

Recently, TN based solvers specialized for combinatorial optimization have been actively researched [5–7]. Other efforts have been reported,

including approaches such as searching for ground states by differentiable programming [8] and classically simulating Quantum Approximate Optimization Algorithm (QAOA) [9]. Particularly, in the weighted Max-Cut problem, a TN approach has been reported to consistently yield high approximation ratios and efficient execution on larger graphs, compared with other classical solvers [7]. Not only to mimic quantum-gate methods, but also to find better optimization algorithms, TNs are gaining much attention as a new solver for combinatorial optimization problems.

Many real-world problems have constraints and require solutions within the feasible solution space that satisfies these constraints. In the field of quantum or quantum-inspired algorithms, constrained combinatorial optimization is typically solved by the penalty function method [10–17]. In this method, violation terms for constraint conditions are added to the original cost Hamiltonian, allowing feasible solutions to be effectively searched for. However, several challenges arise such as difficulty in adjusting the penalty coefficients, an increase in computational cost due to interactions between many qubits, and inability to completely prohibit infeasible solutions.

To address these challenges, a lot of algorithms without the penalty function method have been proposed for quantum algorithms [18–22] and quantum-inspired algorithms [23–26]. One such method involves preparing TNs that describe the superpositioned state of feasible solutions and searching for the ground state with imaginary time evolution [23]. Under ideal imaginary time evolution, transitions to the states of infeasible solutions are forbidden. The second method employs a similar TN as a generative model and iteratively modifies the parameters of each tensor based on the energy expectation value [24]. This approach is an application of the Generator-Enhanced Optimization (GEO) [27] to constrained combinatorial optimization problems. Both methods efficiently optimize by structuring TNs to exclusively output states of feasible solutions. In other words, the capability to design a “feasible” TN is crucial for solving constrained combinatorial optimization problems.

In addition to special constraints such as cardinality ones [24, 26], other constraints have been

targeted. For example, TNs are applied to open-pit mining problem by introducing additional tensors that represent flags to indicate whether the constraint is satisfied or not [23]. Another method has been reported to handle arbitrary linear constraints as well as cardinality ones by introducing $U(1)$ gauge symmetry, which ensures the law of particle number conservation [24]. However, the first method requires an auxiliary tensor to connect the physical variables appearing in the constraints. This leads to an exponential increase in their tensor sizes in the case of global constraints, although the open-pit mining problem has only local constraints. Furthermore, the TNs have a higher dimensional structure than Matrix Product State (MPS). Although the second method can encode global linear constraints to an efficient MPS, it requires processing such as backtracking to ensure $U(1)$ gauge symmetry. This causes deriving “feasible” TNs is $\#P$ -hard when multiple constraints are imposed [24].

As mentioned above, the previous researches applying TNs to constrained combinatorial optimization mainly originate from physics schemes. In this study, we propose designing “feasible” TNs using only elementary mathematics, which leads to more user-friendly design of TNs and application to a wider range of constraints. Thanks to the user-friendly design, the extensibility of constraint conditions can be improved. To deal with real-world problems, tasks are often designed through trial and error, such as the addition of another new constraint, because the appropriate constraint conditions of the problem are usually not predetermined. However, in the previous methods [23, 24], it is necessary to redesign the fully “feasible” TNs from scratch to satisfy all constraints. Because the proposed method can quickly obtain them by simple mathematical manipulation, further utilization of TNs is expected in the field of combinatorial optimization.

Figure 1 provides an overview of the proposed methods. The first is a method that adopts a nilpotent matrix as a tensor. Specifically, we employ a nilpotent matrix to the power of the coefficient a_i in a constraint equation. Thus, “feasible” TNs can be obtained without any backtracking algorithms. This method is a modification of the conventional TNs conserving the number of particles, which enables the handling of a linear inequality or equality constraint. To consider the

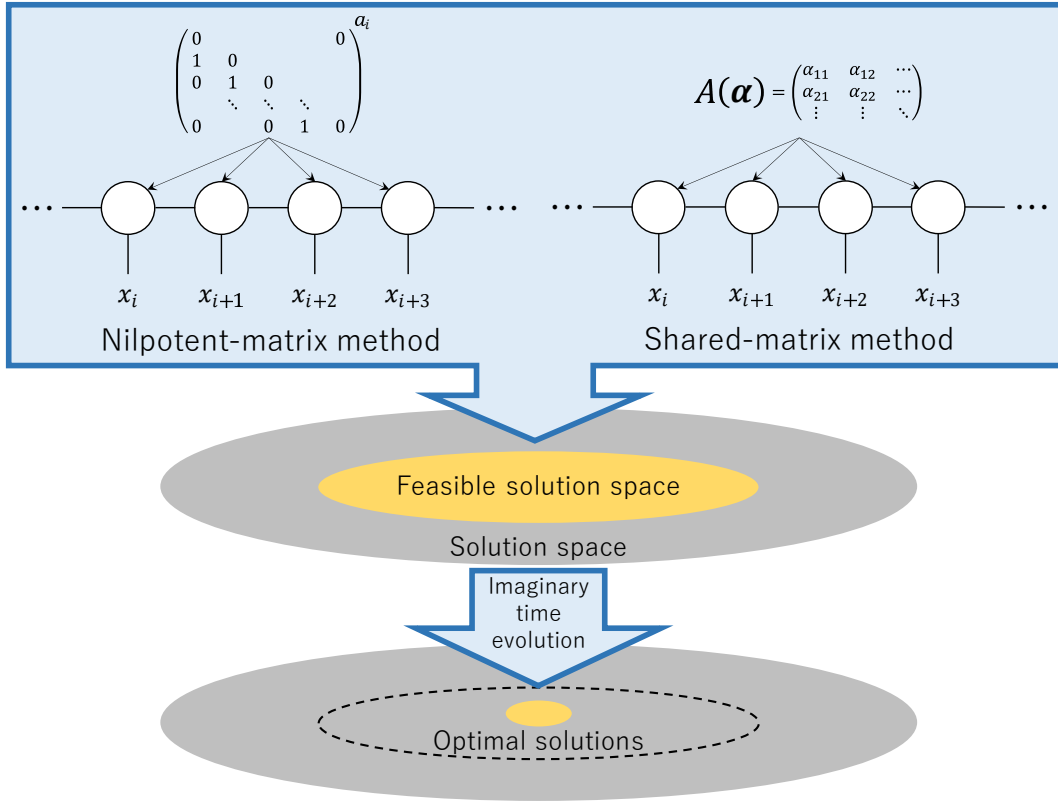


Figure 1: Overview of proposed method. Feasible solutions can be encoded by Tensor Networks (TNs) through nilpotent-matrix method and shared-matrix method. Then, imaginary time evolution is applied to TNs so as to find the optimal solutions. Optimization methods are not limited to imaginary time evolution.

presence of more complex constraints beyond a single linear constraint, as a second method, we have devised an approach that adopts shared matrices for the tensors of each site. Their parameters α are determined algebraically, without relying on the physics scheme of particle number conservation. In other words, this is a method to formulate and solve equations for parameters so that the coefficients of the states that satisfy the constraint are non-zero, while others are set to 0.

After preparing “feasible” TNs, optimization is performed. Here, the optimization method is not limited to imaginary time evolution. Alternatively, one can use such as density matrix renormalization group [2] or GEO method [27].

In this study, our purpose is creating “feasible” TNs for wider kinds of constraints and demonstrating optimization for the principle verification. By the above methods, we showed mathematically that “feasible” TNs can be constructed to accommodate various types of constraints, such as linear equality and inequality constraints, comparison constraints regarding the magnitude relationship between variables, and so

on. We numerically constructed a “feasible” TN for facility location problem, to find much faster construction than conventional methods. Then, by applying imaginary time evolution to the obtained TN, the optimal solutions can be sought. Feasible solutions were consistently obtained, and the optimal solutions were achieved after sufficient time evolution.

The proposed method can construct a “feasible” TN without the penalty function method. Moreover, the results of this study may not only contribute to the development of optimization methods using TNs but also lead to that using quantum gates. In recent years, a technique for converting TNs into equivalent quantum circuits has been proposed [28]. By combining such a technique and the proposed method, we can devise a new solver for constrained combinatorial optimization using quantum gates.

We describe the structure of this paper. First, in Section 2, we introduce the definition of “feasible” TNs and the previous studies. In Section 3 and 4, we discuss the theory of constructing feasible TNs using nilpotent-matrix method

and shared-matrix method, respectively. Then, in Section 5, we extend to multiple constraints. Section 6 details the costs of TNs and compare them with baselines. In Section 7, we solve facility location problem and explain the results while comparing with the baselines. Finally, we conclude in Section 8.

2 Preliminaries

2.1 Tensor network

TNs are mathematical expressions used to describe entangled quantum states in many-body systems [2–4]. Well-known TNs include MPS and Pair Entangled Projected State (PEPS) [29]. A quantum state can generally be represented by

$$|\psi\rangle = \sum_{\mathbf{x}} \psi_{x_1, x_2, \dots, x_N} |x_1, x_2, \dots, x_N\rangle. \quad (1)$$

Here, each x_i represents a physical variable that takes a bit value of $\{0, 1\}$. Thus, summation over $\mathbf{x}$ is performed in $\{0, 1\}^N$. $\psi_{x_1, x_2, \dots, x_N}$ represents the state coefficient. In TNs, this coefficient is restricted to the form of a product of tensors. The original quantum state shown in Eq. (1) requires memory of the order of 2^N . By reducing the size of the tensor, quantum states can be approximately simulated with less memory.

In an MPS, the state coefficient in Eq. (1) is modeled by

$$\psi_{x_1, x_2, \dots, x_N} = \text{tr} \left[\prod_{i=1}^N A^{[i]x_i} \right]. \quad (2)$$

Here, i represents the coordinates of each site in the one-dimensional lattice system and the left and right bonds of on-site matrices $A^{[i]x_i}$ must be determined so that the trace is well-defined.

Another example of a TN is the PEPS below.

$$|\psi\rangle = \sum_{\mathbf{x}} \text{tr}' \left[\prod_{i=1}^{N_1} \prod_{j=1}^{N_2} A^{[i,j]x_{i,j}} \right] |x_{1,1}, \dots, x_{N_1, N_2}\rangle.$$

Here, i, j represents the coordinates of each site in the two-dimensional lattice system. Although the MPS uses a regular trace, in the case of PEPS, a more general trace tr' is used. This tensor trace contracts dummy variables not only in the left and right bonds but also in the up and down bonds. The bonds of on-site matrices $A^{[i,j]x_{i,j}}$ must be determined so that the trace is well-defined.

2.2 Feasible TN for constrained combinatorial optimization

We consider a TN where at least one coefficient of a feasible solution state for given constraint C is a non-zero real value, and any coefficient of a state that violates the constraint is always 0. This TN is defined as a “feasible” TN for the constraint C .

Additionally, the special feasible TNs where the coefficients of any feasible solution state are non-zero are called “fully feasible” TNs. For finding the optimal solutions with imaginary time evolution, an initial TN must be fully feasible. Thus, this paper focuses on constructing a fully feasible MPS. That is, in Eq. (2), we aim to find an MPS that satisfies

$$\text{tr} \left[\prod_{i=1}^N A^{[i]x_i} \right] = \begin{cases} 0 & \text{for } \forall \mathbf{x} \notin X_C, \\ \text{Non-zero} & \text{for } \forall \mathbf{x} \in X_C, \end{cases} \quad (3)$$

where X_C represents the set of all the feasible solutions that satisfy the constraint C .

2.3 Previous studies on feasible TN construction

As previous studies on designing feasible TNs, two main methods are explained. The first method constructs an MPS with $U(1)$ gauge symmetry to conserve the number of particles [24]. This ensures that the charge N_{in} coming in each site, the charge N_{out} going out, and the on-site charge n are totally balanced. Such an MPS has the following forms,

$$A_{\alpha, \beta}^a = \left(A_{n_{\alpha}, n_{\beta}}^{n_a} \right)_{t_{n_{\alpha}}, t_{n_{\beta}}}^{t_{n_a}} \delta_{n + N_{\text{in}}, N_{\text{out}}}, \quad (4)$$

$$N_{\text{in}} = \sum_{i \in \mathcal{I}} n_i, N_{\text{out}} = \sum_{i \in \mathcal{O}} n_i,$$

where $t_n = 1, 2, \dots, d_n$ is the index of degeneracy, and d_n denotes the degree of degeneracy of the charge n . $\mathcal{I}, \mathcal{O}$ are the subscripts of the sites coming in and going out, respectively. α and β are dummy variables. In the case of a constant sum constraint, in other words cardinality constraint $\sum_{i=1}^N x_i = d$,

$$\begin{aligned}
A^{[i]0} &= \begin{cases} \begin{pmatrix} 0 & 1 & & 0 \\ & 0 & 1 & \\ & & \ddots & \ddots \\ 0 & & & 0 & 1 \end{pmatrix} & \text{for } i = 1, \dots, d, \\ \begin{pmatrix} 1 & & & 0 \\ & 1 & & \\ & & 1 & \\ & & & \ddots \end{pmatrix} & \text{for } i = d+1, \dots, N-d, \\ \begin{pmatrix} 0 & & & 1 \\ 0 & & 0 & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ 0 & & 1 & 0 \\ & & & 1 \end{pmatrix} & \text{for } i = N-d+1, \dots, N \end{cases} \\
A^{[i]1} &= \begin{cases} \begin{pmatrix} 1 & 0 & & 0 \\ & 1 & 0 & \\ & & \ddots & \ddots \\ 0 & & & 1 & 0 \end{pmatrix} & \text{for } i = 1, \dots, d, \\ \begin{pmatrix} 0 & & & 0 \\ 1 & 0 & & \\ & 1 & 0 & \\ & & \ddots & \ddots \\ 0 & & & 1 & 0 \end{pmatrix} & \text{for } i = d+1, \dots, N-d, \\ \begin{pmatrix} 1 & & & 0 \\ 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ 0 & & & 0 \end{pmatrix} & \text{for } i = N-d+1, \dots, N \end{cases} \quad (5)
\end{aligned}$$

are one of the feasible tensors so as to satisfy Eq. (4). The size of each matrix is $i \times (i+1)$ for $i = 1, \dots, d$, $d \times d$ for $i = d+1, \dots, N-d$, and $(N-i+2) \times (N-i+1)$ for $i = N-d+1, \dots, N$. Here, each element value was conveniently set to 1, but any real number is acceptable.

While several efforts have reported to connect between linear constraints and TNs [30–33], this method can be applied to arbitrary linear constraints. However, it is necessary to find any consistent set of all link charges in the constraints. Finding the appropriate TN becomes $\#P$ -hard. Therefore, the paper [24] stated that it is difficult to derive a fully feasible TN for multiple linear constraints.

Recently, an improved method has been reported to efficiently design fully feasible TNs without explicitly deriving all link charges [25]. However, as in the literature [24], backtracking

is used, which causes exponential computation for the number of the constraints. Thus, obtaining TNs is difficult when a large number of constraints are imposed. Handling general constraints other than linear ones is also difficult. In addition, there is a weakness in constraint extensibility. When another new constraint is added to existing constraints, it should be noted that redesigning fully feasible TNs from scratch is necessary to satisfy all constraints.

As another previous study, a TN for constraints among local sites has also been proposed [23]. In this method, auxiliary tensors that represent whether the constraints are satisfied or not are added. For example, the constraint $C : x_1 + x_2 + x_3 = 1$ is considered.

$$|\psi\rangle = \sum_{\mathbf{x}, \mathbf{y}} \prod_{i=1}^3 A_{y_i}^{[i]x_i} B_{y_1, y_2, y_3}^{[C]} |x_1, x_2, x_3\rangle. \quad (6)$$

Both the physical variable x_1, x_2, x_3 and the dummy variable y_1, y_2, y_3 are binary variables of $\{0, 1\}$. $B^{[C]}$ corresponds to the auxiliary tensor for the constraint condition C and is responsible for connecting the physical variables that are involved in this condition. All the elements of tensors are initially set to 0. Then, for the fully feasibility, $B_{0,0,1}^{[C]} = B_{0,1,0}^{[C]} = B_{1,0,0}^{[C]} = 1$ and $A_0^{[1]0} = A_1^{[1]1} = A_0^{[2]0} = A_1^{[2]1} = A_0^{[3]0} = A_1^{[3]1} = 1$ are encoded. In the paper [23], a PEPS for the open-pit mining problem is constructed using such tensor structure. Several researches have been reported to apply this method to other problems [34, 35].

Theoretically, introducing auxiliary tensors can handle a general type of constraints. However, the bond size of $B^{[C]}$ becomes exponentially large for the number of physical variables involved in the constraint C . Thus, this method is considered to be specialized for local constraints. When another new constraint is added to existing constraints, redesigning fully feasible TNs from scratch is necessary as with the methods in [24, 25]. In addition, introducing auxiliary tensors disables the computational advantages of MPS because higher dimensional lattices are required.

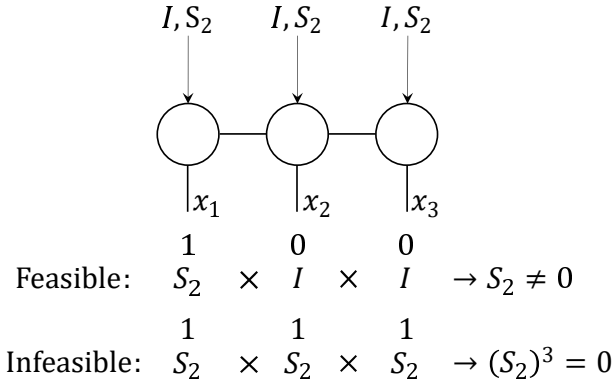


Figure 2: MPS constructed by nilpotent-matrix method (equally and positively weighted). A special constraint $x_1 + x_2 + x_3 \leq 2$ is considered. A nilpotent matrix S_2 becomes zero when multiplied more times than upper bound 2.

3 Feasible MPS construction by nilpotent-matrix method

We explain how to construct a fully feasible MPS using the nilpotent-matrix method. Target constraints are linear inequality or equality constraints with integer coefficients. Letting d be an integer greater than 0, one nilpotent matrix of exponent $d + 1$ can be expressed by

$$S_d \equiv \begin{pmatrix} 0 & & & 0 \\ 1 & 0 & & \\ 0 & 1 & 0 & \\ & \ddots & \ddots & \ddots \\ 0 & & 0 & 1 & 0 \end{pmatrix}. \quad (7)$$

This is a $(d + 1) \times (d + 1)$ matrix that becomes zero matrix when raised to the power of $d + 1$. Although we set 1 as a non-zero element, any real number can be set generally. In particular, if we set $\sqrt{1}, \sqrt{2}, \sqrt{3}, \dots$ in order from the top left of the matrix, it will be equivalent to the matrix representation of creation or annihilation operators. From Eq. (7), the nilpotent matrix S_d to the power of k is

$$(S_d)^k = \begin{cases} \begin{pmatrix} 0 & & & 0 \\ 1 & 0 & & \\ 0 & 1 & 0 & \\ & \ddots & \ddots & \ddots \\ 0 & & 0 & 1 & 0 \end{pmatrix} \\ d - k + 1 \end{cases} \quad (8)$$

where $k = 0, 1, \dots, d$. Thus, S_d has the property that the sub-diagonal elements with the value of 1 descend one step towards the lower left, each time the matrix is multiplied. As shown in Fig. 2, the nilpotent-matrix method uses this S_d matrix as a tensor to construct an MPS. For example, we consider the following MPS

$$\begin{aligned} A^{[1]0} &= v, A^{[1]1} = vS_d, \\ A^{[i]0} &= I, A^{[i]1} = S_d \text{ for } i = 2, 3, \dots, N-1, \\ A^{[N]0} &= v^T, A^{[N]1} = S_d v^T. \end{aligned} \quad (9)$$

Here, I denotes a identity matrix and $v \equiv (1 \ 1 \ \dots \ 1 \ 1)$ is a row vector of dimension $d + 1$ with all elements being 1. When the trace for the tensor product in Eq. (9) is calculated,

$$\psi_{x_1, x_2, \dots, x_N} = v (S_d)^{\sum_{i=1}^N x_i} v^T \quad (10)$$

is obtained. If $\sum_{i=1}^N x_i > d$, the tensor product $(S_d)^{\sum_{i=1}^N x_i}$ becomes a zero matrix and the right hand of Eq. (10) is 0. Otherwise, a sub-diagonal component with the value of 1 remains according to Eq. (8), and the product becomes non-zero. Thus, the MPS described above is found to be fully feasible for the inequality constraint $\sum_{i=1}^N x_i \leq d$ because Eq. (3) is satisfied. Figure 2 illustrates an example under $x_1 + x_2 + x_3 \leq 2$. While the conventional method described in the paper [24] uses a nilpotent matrix at some sites as shown in Eq. (5), we use the matrix at all sites, as proposed in [26]. Below, we demonstrate that the nilpotent-matrix method can be extended to linear inequality or equality constraints.

3.1 Feasible MPS for linear inequality constraint

In this section, we construct a fully feasible MPS for linear inequality constraints using the nilpotent-matrix method.

$$\sum_{i=1}^N a_i x_i \leq d. \quad (11)$$

First, let all coefficients a_i be non-negative integers, and let d be an integer greater than 0. If $d > \sum_{i=1}^N a_i$, Eq. (11) becomes a trivial equation. Therefore, we focus on the case where $d \leq \sum_{i=1}^N a_i$. A fully feasible MPS is defined as

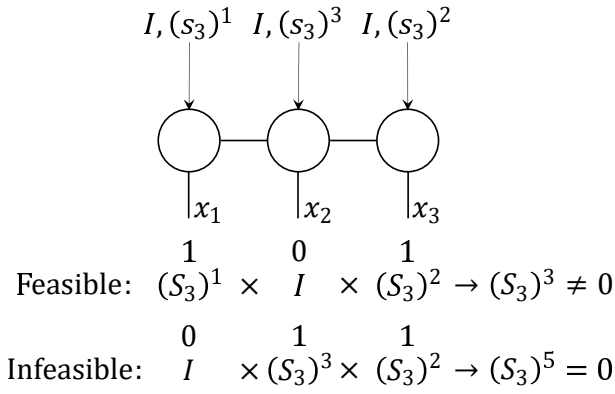


Figure 3: MPS constructed by nilpotent-matrix method (positively weighted). A special constraint $x_1 + 3x_2 + 2x_3 \leq 3$ is considered. A nilpotent matrix S_3 becomes zero when multiplied more times than upper bound 3.

follows

$$\begin{aligned} A^{[1]0} &= v, A^{[1]1} = v (S_d)^{a_1}, \\ A^{[i]0} &= I, A^{[i]1} = (S_d)^{a_i} \text{ for } i = 2, 3, \dots, N-1, \\ A^{[N]0} &= v^T, A^{[N]1} = (S_d)^{a_N} v^T. \end{aligned} \quad (12)$$

An example under the constraint $x_1 + 3x_2 + 2x_3 \leq 3$ is shown in Fig. 3.

When the trace for the tensor product in Eq. (12) is calculated,

$$\psi_{x_1, x_2, \dots, x_N} = v (S_d)^{\sum_{i=1}^N a_i x_i} v^T \quad (13)$$

is obtained. If $\sum_{i=1}^N a_i x_i > d$, the tensor product $(S_d)^{\sum_{i=1}^N a_i x_i}$ becomes a zero matrix and the right hand of Eq. (13) is 0. Otherwise, a sub-diagonal component with the value of 1 remains according to Eq. (8), and the product becomes non-zero. That is, the above MPS is found to be fully feasible for the inequality constraint $\sum_{i=1}^N a_i x_i \leq d$ because Eq. (3) is satisfied.

Next, the constraint condition $\sum_{i=1}^N a_i x_i \leq d$ is extended to allow negative integer coefficients. By excluding a trivial domain, we assume that d is an integer such that $-\sum_{i \in \Delta_-} |a_i| \leq d \leq \sum_{i \in \Delta_+} a_i$. Here, Δ_+, Δ_- are the index sets where the coefficient a_i is non-negative and negative, respectively. The constraint condition is equivalent to

$$\sum_{i \in \Delta_+} a_i x_i + \sum_{i \in \Delta_-} |a_i| (1 - x_i) \leq d + \sum_{i \in \Delta_-} |a_i|. \quad (14)$$

From Eq. (14), by flipping the bit of the physical variable with a negative coefficient, the constraint

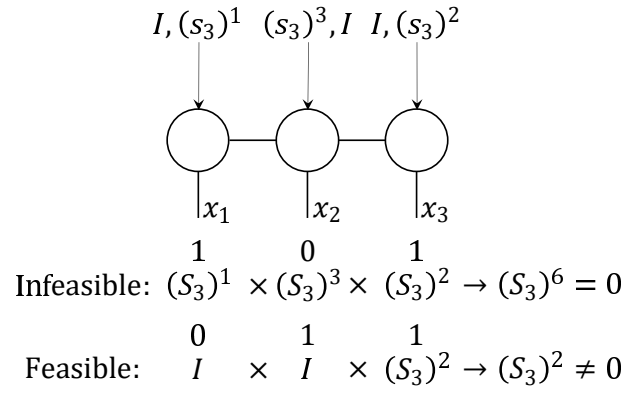


Figure 4: MPS constructed by nilpotent-matrix method (arbitrarily weighted). A special constraint $x_1 - 3x_2 + 2x_3 \leq 0$ is considered. This constraint is equivalent to $x_1 + 3(1 - x_2) + 2x_3 \leq 3$. A nilpotent matrix S_3 becomes zero when multiplied more times than upper bound 3.

condition can be converted to an equivalent condition with all non-negative coefficients. Therefore, a fully feasible MPS is defined as follows

$$\begin{aligned} A^{[1]0} &= v' (S_{d'})^{|a_1| - b_1}, A^{[1]1} = v' (S_{d'})^{b_1}, \\ A^{[i]0} &= (S_{d'})^{|a_i| - b_i}, A^{[i]1} = (S_{d'})^{b_i} \text{ for } i = 2, \dots, N-1, \\ A^{[N]0} &= (S_{d'})^{|a_N| - b_N} (v')^T, A^{[N]1} = (S_{d'})^{b_N} (v')^T. \end{aligned} \quad (15)$$

Here, we let $d' \equiv d + \sum_{i \in \Delta_-} |a_i| \geq 0$ and $b_i \equiv \max\{0, a_i\}$, $v' \equiv (1 \ 1 \ \dots \ 1 \ 1)$ is a row vector of dimension $d' + 1$ with all elements being 1. An example under the constraint $x_1 - 3x_2 + 2x_3 \leq 0$ is shown in Fig. 4.

When the trace for the tensor product in Eq. (15) is calculated,

$$\psi_{x_1, x_2, \dots, x_N} = v' (S_{d'})^{\sum_{i=1}^N a_i x_i + d' - d} (v')^T \quad (16)$$

is obtained. If $\sum_{i=1}^N a_i x_i > d$, the tensor product $(S_{d'})^{\sum_{i=1}^N a_i x_i + d' - d}$ becomes a zero matrix and the right hand of Eq. (16) is 0. Otherwise, a sub-diagonal component with the value of 1 remains according to Eq. (8), and the product becomes non-zero. That is, the above MPS is found to be fully feasible for the inequality constraint $\sum_{i=1}^N a_i x_i \leq d$ because Eq. (3) is satisfied.

Finally, the case of inequality constraints that consider not only the upper bound but also the lower bound ($d_1 \leq d_2$) is considered.

$$d_1 \leq \sum_{i=1}^N a_i x_i \leq d_2. \quad (17)$$

By excluding a trivial domain, we assume that d_1 and d_2 are integers that satisfy $-\sum_{i \in \Delta_-} |a_i| \leq$

$d_1 \leq d_2 \leq \sum_{i \in \Delta_+} a_i$. In this case, a fully feasible MPS is defined as follows

$$\begin{aligned} A^{[1]0} &= v_1 (S_{d'_2})^{|a_1|-b_1}, A^{[1]1} = v_1 (S_{d'_2})^{b_1}, \\ A^{[i]0} &= (S_{d'_2})^{|a_i|-b_i}, A^{[i]1} = (S_{d'_2})^{b_i} \text{ for } i = 2, \dots, N-1, \\ A^{[N]0} &= (S_{d'_2})^{|a_N|-b_N} (v_2)^T, A^{[N]1} = (S_{d'_2})^{b_N} (v_2)^T. \end{aligned} \quad (18)$$

Here, $d'_2 \equiv d_2 + \sum_{i \in \Delta_-} |a_i|$. $v_1 \equiv (0 \ \dots \ 0 \ 1)$ is a row vector of dimension $d'_2 + 1$ with the last component being 1 and the other components being 0. $v_2 \equiv (1 \ \dots \ 1 \ 0 \ \dots \ 0)$ is a row vector of dimension $d'_2 + 1$ with the last component being 0 and the other components being 1.

When the trace for the tensor product in Eq. (18) is calculated,

$$\psi_{x_1, x_2, \dots, x_N} = v_1 (S_{d'_2})^{\sum_{i=1}^N a_i x_i + d'_2 - d_2} (v_2)^T \quad (19)$$

is obtained. If $\sum_{i=1}^N a_i x_i > d_2$, the tensor product $(S_{d'_2})^{\sum_{i=1}^N a_i x_i + d'_2 - d_2}$ becomes a zero matrix and the right hand of Eq. (19) is 0. If $\sum_{i=1}^N a_i x_i < d_1$, the right hand is also 0 because the last component in $(S_{d'_2})^{\sum_{i=1}^N a_i x_i + d'_2 - d_2} (v_2)^T$ become 0 due to $\sum_{i=1}^N a_i x_i + d'_2 - d_2 < d'_1$. Otherwise, under $d_1 \leq \sum_{i=1}^N a_i x_i \leq d_2$, the right hand is non-zero because the last component holds the value of 1. Thus, the above MPS is found to be fully feasible for the inequality constraint $d_1 \leq \sum_{i=1}^N a_i x_i \leq d_2$ because Eq. (3) is satisfied.

3.2 Feasible MPS for linear equality constraint

The nilpotent-matrix method can be extended to linear equality constraints.

$$\sum_{i=1}^N a_i x_i = d. \quad (20)$$

Equation (20) is a special case of Eq. (17) under $d_1, d_2 \rightarrow d$. In other words, by setting $d_1, d_2 \rightarrow d$ in Eq. (18), a fully feasible MPS for the constraints of Eq. (20) is obtained.

$$\begin{aligned} A^{[1]0} &= (0 \ \dots \ 0 \ 1) (S_{d'})^{|a_1|-b_1}, \\ A^{[1]1} &= (0 \ \dots \ 0 \ 1) (S_{d'})^{b_1}, \\ A^{[i]0} &= (S_{d'})^{|a_i|-b_i}, A^{[i]1} = (S_{d'})^{b_i} \text{ for } i = 2, \dots, N-1, \\ A^{[N]0} &= (S_{d'})^{|a_N|-b_N} (1 \ 0 \ \dots \ 0)^T, \\ A^{[N]1} &= (S_{d'})^{b_N} (1 \ 0 \ \dots \ 0)^T. \end{aligned} \quad (21)$$

4 Feasible MPS construction by shared-matrix method

In the previous section, we constructed a fully feasible MPS using nilpotent matrices to ensure the law of particle number conservation. However, many real-world problems often involve more complex constraints than a single linear constraint. Thus, we also propose the following shared-matrix method. This method is expected to handle complex constraints because $U(1)$ gauge symmetry is not assumed.

4.1 Shared-matrix method

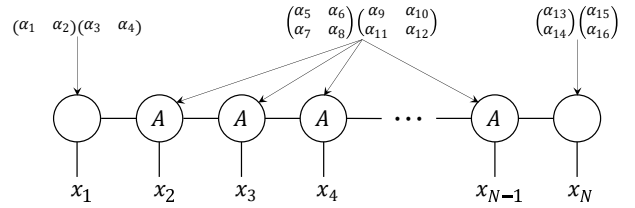


Figure 5: MPS constructed by shared-matrix method. By sharing tensors on several sites, their parameters can be quickly determined so as to obtain feasible MPSs.

As shown in Fig. 5, the shared-matrix method adopts shared matrices for the tensors of the MPS at sites except for both ends and determines the elements of each matrix inversely so that the output states become feasible solutions. That is, for an N -bit problem, we assume the tensors of the MPS as

$$\begin{aligned} A^{[1]0} &= (\alpha_1 \ \alpha_2), A^{[1]1} = (\alpha_3 \ \alpha_4), \\ A^{[i]0} &= \begin{pmatrix} \alpha_5 & \alpha_6 \\ \alpha_7 & \alpha_8 \end{pmatrix}, A^{[i]1} = \begin{pmatrix} \alpha_9 & \alpha_{10} \\ \alpha_{11} & \alpha_{12} \end{pmatrix} \text{ for } i = 2, \dots, N-1, \\ A^{[N]0} &= \begin{pmatrix} \alpha_{13} \\ \alpha_{14} \end{pmatrix}, A^{[N]1} = \begin{pmatrix} \alpha_{15} \\ \alpha_{16} \end{pmatrix} \end{aligned} \quad (22)$$

and determine the real values of matrix parameters α . Hereinafter, shared matrices are represented by $A^0 = A^{[i]0}, A^1 = A^{[i]1}$ for $i = 2, \dots, N-1$. Equations for the matrix parameters are designed so that the trace of the tensor product of Eq. (22) becomes non-zero for feasible solutions, and 0 for infeasible solutions. In other words, by finding the parameters to satisfy Eq. (3), a fully feasible MPS can be obtained. While a 2×2 shared matrix is used in Eq. (22), we could also adopt a larger size matrix. For example, a nilpotent matrix in Eq. (9) can be considered to be a $(d+1) \times (d+1)$ shared matrix. If a

small size matrix is adopted, an MPS has a moderate bond dimension, which leads to an efficient calculation during imaginary time evolution.

Generally, the matrix parameters of an MPS are difficult to be determined inversely for the target property because the number of them is intractable. However, by using a shared matrix, this number can be significantly reduced and these parameters are expected to be efficiently solved. The method may be applicable to problems with high periodicity and symmetry, because it is expected that feasible TNs can be described with small independent parameters. In the following sections, we construct fully feasible MPSs for several kinds of constraints by using shared-matrix method. For simplicity, we assume that each element of the matrix takes only the values 0 or 1 hereinafter.

4.2 Feasible MPS for many-to-one comparison constraint

A fully feasible MPS for the many-to-one comparison constraint

$$x_1, x_2, \dots, x_{N-1} \leq x_N \quad (23)$$

($N \geq 3$) is considered. This inequality constraint is often used in assignment problems such as facility location problems [36].

First, we determine the type of a shared matrix to be employed. If $x_i = 0$ on i th site ($i = 2, \dots, N-1$), the values of the other bits are unaffected. Additionally, if at least one or more values are taken as 1 at the bits on $i = 2, \dots, N-1$, the value of x_N should be 1. Therefore, we assume an identity matrix as the shared matrix A^0 and an idempotent matrix as A^1 . Considering these assumptions, we adopt the matrices in Appendix A,

$$A^0 = I \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, A^1 = P \equiv \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad (24)$$

as shared matrices. As a result, the product of the tensors on the sites $i = 2, \dots, N-1$ becomes

$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ if at least one or more bits have the value of 1 and $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ if all bits are 0. Thus, equations

that satisfy Eq. (3)

$$\begin{aligned} \alpha_1 \alpha_{13} + \alpha_2 \alpha_{14} &> 0, \\ \alpha_1 \alpha_{15} + \alpha_2 \alpha_{16} &> 0, \\ \alpha_1 \alpha_{13} &= 0, \\ \alpha_1 \alpha_{15} &> 0, \\ \alpha_3 \alpha_{13} + \alpha_4 \alpha_{14} &= 0, \\ \alpha_3 \alpha_{15} + \alpha_4 \alpha_{16} &> 0, \\ \alpha_3 \alpha_{13} &= 0, \\ \alpha_3 \alpha_{15} &> 0 \end{aligned} \quad (25)$$

are obtained. By calculating the parameters that satisfy Eq. (25),

$$\begin{aligned} A^{[1]0} &= \begin{pmatrix} 1 & 1 \end{pmatrix}, A^{[1]1} = \begin{pmatrix} 1 & 0 \end{pmatrix}, \\ A^{[N]0} &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}, A^{[N]1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{aligned} \quad (26)$$

are obtained. By taking the product of the tensors $A^{[i]}$ in Eq. (24) and (26), a fully feasible MPS is realized.

4.3 Feasible MPS for domain-wall encoding constraint

Next, a fully feasible MPS is constructed for the domain-wall type constraint

$$x_1 \leq x_2 \leq \dots \leq x_N \quad (27)$$

($N \geq 3$), which is the comparison constraint similar to Subsection 4.2. The technique of designing bit variables to satisfy this constraint is called domain-wall encoding. This encoding is frequently used in quantum annealing because the number of interactions between bits is suppressed [37].

First, we determine the type of a shared matrix to be employed. If at least one or more values are set to 1 at the bits on $i = 2, \dots, N-1$, the value of x_N should be 1. Additionally, if the bits on $2 \leq i \leq N-2$ are set to 1, the value of x_{i+1} should be 1. Therefore, we assume an idempotent matrix as the shared matrix A^0 , and $A^1 A^0 = 0$. Considering these assumptions, we adopt the matrices in Appendix A,

$$A^0 = Q \equiv \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, A^1 = P \equiv \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad (28)$$

as shared matrices. As a result, the product of the tensors on the sites $i = 2, \dots, N-1$ becomes

$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$ if all bits are 0 and $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ if one domain wall exists or all bits are 1, and a zero matrix otherwise. Thus, equations that satisfy Eq. (3)

$$\begin{aligned} \alpha_2\alpha_{13} + \alpha_2\alpha_{14} &> 0, \\ \alpha_2\alpha_{15} + \alpha_2\alpha_{16} &> 0, \\ \alpha_1\alpha_{13} &= 0, \\ \alpha_1\alpha_{15} &> 0, \\ \alpha_4\alpha_{13} + \alpha_4\alpha_{14} &= 0, \\ \alpha_4\alpha_{15} + \alpha_4\alpha_{16} &= 0, \\ \alpha_3\alpha_{13} &= 0, \\ \alpha_3\alpha_{15} &> 0 \end{aligned} \quad (29)$$

are obtained. By calculating the parameters that satisfy Eq. (29),

$$\begin{aligned} A^{[1]0} &= \begin{pmatrix} 1 & 1 \end{pmatrix}, A^{[1]1} = \begin{pmatrix} 1 & 0 \end{pmatrix}, \\ A^{[N]0} &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}, A^{[N]1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{aligned} \quad (30)$$

are obtained. By taking the product of the tensors $A^{[i]}$ in Eq. (28) and (30), a fully feasible MPS is realized.

4.4 Feasible MPS for degree-reduction constraint

Next, a fully feasible MPS is constructed for the constraint

$$\prod_{i=1}^{N-1} x_i = x_N \quad (31)$$

($N \geq 3$), which aims at dimension reduction. Using this reduction, it is possible to transform a higher-order function into a lower-order function [38]. For example, an original problem expressed in higher order binary optimization can be transformed into quadratic order binary optimization so as to be solved by quantum annealing [39].

First, we determine the type of a shared matrix to be employed. If $x_i = 1$ on i th site ($i = 2, \dots, N-1$), the values of the other bits are unaffected. Additionally, if at least one or more values are set to 0 at the bits on $i = 2, \dots, N-1$, the value of x_N should be 0. Therefore, we assume an identity matrix as the shared matrix A^1 and an idempotent matrix as A^0 . Considering

these assumptions, we adopt the matrices in Appendix A,

$$A^0 = R \equiv \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, A^1 = I \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (32)$$

as shared matrices. As a result, the product of the tensors on the sites $i = 2, \dots, N-1$ becomes $\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$ if at least one or more bits have the value of 0 and $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ if all bits are 1. Thus, equations that satisfy Eq. (3)

$$\begin{aligned} \alpha_1\alpha_{13} + \alpha_2\alpha_{13} &> 0, \\ \alpha_1\alpha_{15} + \alpha_2\alpha_{15} &= 0, \\ \alpha_1\alpha_{13} + \alpha_2\alpha_{14} &> 0, \\ \alpha_1\alpha_{15} + \alpha_2\alpha_{16} &= 0, \\ \alpha_3\alpha_{13} + \alpha_4\alpha_{13} &> 0, \\ \alpha_3\alpha_{15} + \alpha_4\alpha_{15} &= 0, \\ \alpha_3\alpha_{13} + \alpha_4\alpha_{14} &= 0, \\ \alpha_3\alpha_{15} + \alpha_4\alpha_{16} &> 0 \end{aligned} \quad (33)$$

are obtained. By calculating the parameters that satisfy Eq. (33),

$$\begin{aligned} A^{[1]0} &= \begin{pmatrix} 1 & 0 \end{pmatrix}, A^{[1]1} = \begin{pmatrix} 0 & 1 \end{pmatrix}, \\ A^{[N]0} &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}, A^{[N]1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{aligned} \quad (34)$$

are obtained. By taking the product of the tensors $A^{[i]}$ in Eq. (32) and (34), a fully feasible MPS is realized.

4.5 Remarks on other constraints

The shared-matrix method can be applied to other constraints. Firstly, it can handle the negation of Eq. (20), that is $\sum_{i=1}^N a_i x_i \neq d$. As detailed in Appendix C, the constraint can be encoded into only shared matrices with 2×2 size. This not-equality is a well-known constraint for disjunctive programming [40]. Usually, by utilizing Big M method [41], the constraint is converted to the equivalent linear constraints $\sum_{i=1}^N a_i x_i - d \geq 1 - Gb$ and $\sum_{i=1}^N a_i x_i - d \leq -1 + G(1-b)$ for linear programming. Here, G is a sufficiently large constant and b is a binary variable. However, our formalism can naturally encode the constraint into a simple MPS without Big

M method. Similarly, the congruent-type constraints (i.e. $\sum_{i=1}^N a_i x_i \not\equiv 0 \pmod{2}, 0 \pmod{3}, \dots$) can also be handled, as detailed in Appendix C.

As shown in several examples above, the shared-matrix method has a potential to derive fully feasible MPSs with a compact tensor size even for global constraints. In addition, this method can construct such MPSs even for non-linear constraints or not-equal constraints.

5 Feasible MPS synthesis

Next, we explain the design of an MPS that generates states satisfying all constraints by synthesizing TNs. For example, although the method described in Section 3 is originally designed for a single linear constraint, this MPS synthesis allows it to handle multiple linear constraints. Thus, when another new constraint is added to existing constraints, redesigning fully feasible TNs from scratch is not necessary.

5.1 Feasible MPS for uncorrelated constraints

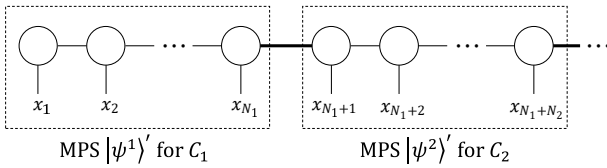


Figure 6: Feasible MPS for uncorrelated constraints. A fully feasible MPS for all constraints can be constructed by connecting MPSs for each constraint in series. That is, matrix products of the tensors on the sites linked by bold bonds are performed.

In this section, we consider a problem with multiple constraints C_k ($k = 1, \dots, L$), and assume that the constraints are independent of each other. That is, there is no intersection between the index sets $D_k = \{i_1, i_2, \dots, i_{N_k}\}$ of the physical variables appearing in each constraint C_k . From this assumption of independence, by appropriately arranging the order of the physical variables, the sets of physical variables appearing in each constraint C_k can be enumerated in ascending order: $D_1 = \{1, 2, \dots, N_1\}$, $D_2 = \{N_1 + 1, N_1 + 2, \dots, N_1 + N_2\}$, $\dots$, $D_L = \{\dots, \sum_{k=1}^L N_k\}$. In addition, the remaining $x_{1+\sum_{k=1}^L N_k}, \dots, x_N$ are the variables that do not appear in the constraints.

We assume that a fully feasible MPS has been already obtained,

$$|\psi^k\rangle' = \sum_{\mathbf{x}_{D_k}} \psi_{\mathbf{x}_{D_k}}^k |\mathbf{x}_{D_k}\rangle = \sum_{\mathbf{x}_{D_k}} \text{tr} \left[\prod_{i \in D_k} \tilde{A}_k^{[i]x_i} \right] |\mathbf{x}_{D_k}\rangle \quad (35)$$

for each constraint C_k . Here, $\mathbf{x}_{D_k} = (x_{\min D_k}, x_{\min D_k+1}, \dots, x_{\max D_k})$, and a prime symbol attached in the wave function is used to represent a subsystem. In addition, a tilde symbol attached in the tensors is used to represent a partial constraint. A fully feasible MPS for all constraints can be constructed by connecting the above MPS for each constraint in series, as shown in Fig. 6.

$$A^{[i]x_i} = \begin{cases} \tilde{A}_k^{[i]x_i} & \text{for } i \in D_k, \\ 1 & \text{for } i = \sum_{k=1}^L N_k + 1, \dots, N. \end{cases} \quad (36)$$

This is proved as follows. When the trace for the tensor product in Eq. (36) is calculated,

$$\begin{aligned} \text{tr} \left[\prod_{i=1}^N A^{[i]x_i} \right] &= \text{tr} \left[\left(\prod_{i \in D_1} \tilde{A}_1^{[i]x_i} \right) \times \dots \times \left(\prod_{i \in D_L} \tilde{A}_L^{[i]x_i} \right) \times 1 \right] \\ &= \text{tr} \left[\prod_{i \in D_1} \tilde{A}_1^{[i]x_i} \right] \times \dots \times \text{tr} \left[\prod_{i \in D_L} \tilde{A}_L^{[i]x_i} \right] \\ &= \prod_{k=1}^L \psi_{\mathbf{x}_{D_k}}^k \end{aligned} \quad (37)$$

is obtained. Here, we used the property that $\prod_{i \in D_k} \tilde{A}_k^{[i]x_i}$ is not a matrix but a scalar. At the end of Eq. (37), $\psi_{\mathbf{x}_{D_k}}^k$ is non-zero if $\mathbf{x}_{D_k}$ satisfies the constraint C_k and 0 otherwise. Thus, $\text{tr}[\prod_{i=1}^N A^{[i]x_i}]$ satisfies Eq. (3), and is proved to be a fully feasible MPS.

5.2 Feasible MPS for correlated constraints

In this section, we consider a problem with multiple constraints C_k ($k = 1, \dots, L$), and assume that the constraints are not independent. We assume that a fully feasible MPS has been already obtained,

$$|\psi^k\rangle = \sum_{\mathbf{x}} \psi_{\mathbf{x}}^k |x_1, \dots, x_N\rangle = \sum_{\mathbf{x}} \text{tr} \left[\prod_{i=1}^N \tilde{A}_k^{[i]x_i} \right] |x_1, \dots, x_N\rangle \quad (38)$$

for each constraint C_k . A fully feasible MPS for all constraints can be constructed by connecting the above MPS for each constraint in parallel, as shown in Fig. 7.

$$A^{[i]x_i} = \tilde{A}_1^{[i]x_i} \otimes \tilde{A}_2^{[i]x_i} \dots \otimes \tilde{A}_L^{[i]x_i} \quad \text{for } i = 1, 2, \dots, N. \quad (39)$$

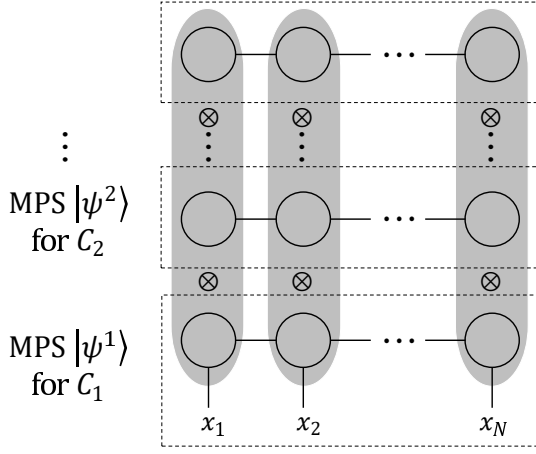


Figure 7: Feasible MPS for correlated constraints. A fully feasible MPS for all constraints can be constructed by connecting MPSs for each constraint in parallel. That is, Kronecker products of the tensors inside shaded areas are performed.

Here, $\otimes$ represents Kronecker product. This is proved as follows. When the trace for the tensor product in Eq. (39), is calculated,

$$\begin{aligned}
 \text{tr} \left[\prod_{i=1}^N A^{[i]x_i} \right] &= \text{tr} \left[\prod_{i=1}^N \left(\tilde{A}_1^{[i]x_i} \otimes \cdots \otimes \tilde{A}_L^{[i]x_i} \right) \right] \\
 &= \text{tr} \left[\left(\prod_{i=1}^N \tilde{A}_1^{[i]x_i} \right) \otimes \cdots \otimes \left(\prod_{i=1}^N \tilde{A}_L^{[i]x_i} \right) \right] \\
 &= \text{tr} \left[\prod_{i=1}^N \tilde{A}_1^{[i]x_i} \right] \times \cdots \times \text{tr} \left[\prod_{i=1}^N \tilde{A}_L^{[i]x_i} \right] \\
 &= \prod_{k=1}^L \psi_{\mathbf{x}}^k
 \end{aligned} \tag{40}$$

is obtained. Here, we used the mixed-product property and spectrum property in Appendix B. At the end of Eq. (40), $\psi_{\mathbf{x}}^k$ is non-zero if $\mathbf{x}$ satisfies the constraint C_k and 0 otherwise. Thus, $\text{tr}[\prod_{i=1}^N A^{[i]x_i}]$ satisfies Eq. (3), and is proved to be a fully feasible MPS.

Note that as shown in Eq. (39), the size of the tensor increases exponentially with the number of non-independent constraints C_k . Therefore, for general problems involving many constraints, it is desirable to use the shared-matrix method or the technique in Subsection 5.1 as much as possible so as to reduce the operations of taking Kronecker product. For example, the many-to-one comparison constraint (23) consists of $N - 1$ inequality constraints that are not independent. If the MPS is designed by Eq. (39), the size of the tensor will be 2^{N-1} . On the other hand, using

the shared-matrix method, the MPS can be efficiently constructed by 2×2 matrices as shown in Eq. (24).

5.3 Additional tensors for unconstrained variables

For physical variables that do not appear in given constraints, whether the constraints are satisfied or not is independent of their values. Thus, one of natural choices for tensors on the corresponding site is an identity matrix. Their sizes are appropriately determined so that a tensor product can be well-defined.

We let the physical variable x_i do not appear in the constraint. Additionally, we let the tensors A_{i-1}, A_{i+1} be already given on $i - 1, i + 1$ th sites and their sizes be $n \times m, m \times l$, respectively. In this case, an $m \times m$ identity matrix should be adopted as a tensor on i th site. Such formalism is ill-defined unless the column size of A_{i-1} corresponds to the row size of A_{i+1} . However, the tensors constructed by the nilpotent-matrix method or shared-matrix method naturally satisfy this condition.

6 Cost comparison

A comparison of methods for generating fully feasible TNs is summarized in Table 1. While the methods in [24, 25] and the nilpotent-matrix method are specialized for linear constraints, the method in [23] can handle various types of local constraints. The shared-matrix method can handle arbitrary constraints as long as tensor parameters to satisfy Eq. (3) can be appropriately derived.

Next, we compare the algorithms for obtaining fully feasible TNs. In the paper [24], all link charges for given constraints are enumerated by backtracking. Therefore, under multiple linear constraints, this process is $\sharp$ P-hard. Alternatively, the improved method [25] enumerates quantum regions, which represent the regions of a feasible charge space. In that process, backtracking is also required. The algorithm in [23] checks whether each bit string satisfies the constraints or not and adds auxiliary tensors to ensure sampling feasible states. The proposed nilpotent-matrix method utilizes fixed nilpotent matrices. In the case of multiple constraints, Kronecker product

Table 1: Cost comparison. Here, L , N , d , and k denote the number of constraints, the number of physical variables, the sum constant in cardinality constraint $\sum_i x_i = d$, and the size of shared matrices, respectively. d'_2 is defined in Subsection 3.1. d'_2 corresponds to d in the case of cardinality constraints.

	[24]	[25]	[23]	Nilpotent-matrix	Shared-matrix
Constraint	Linear	Linear	Local	Linear	Any ^a
Algorithm	Backtracking	Backtracking	Adding tensor	Matrix manip.	Algebraic calc.
Dimension	One	One	Higher	One	One
Log size cost	$O(L \log d)^b$	$O(\log N) + O(L)$	$O(N)$	$O(L \log d'_2)$	$O(\log k)$

^a On condition that Eq. (3) is satisfied.

^b In the special case of multiple cardinality constraints.

described in Section 5 can generate fully feasible tensors. In the shared-matrix method, the TNs are obtained by solving the equations that the parameters of the shared matrix must satisfy. Thus, the proposed algorithms require not backtracking but rather elementary mathematics, such as matrix manipulation and algebraic computation.

The structures of the obtained TNs are more than two-dimensional only for the method in [23], and are one-dimensional for the others. This is because, as shown in Eq. (6), the method requires not only on-site tensors but also auxiliary tensors that span several sites appearing in the constraints. Thus, the proposed methods have a simpler structure and gain computational advantages by utilizing an MPS.

We discuss the cost of TNs. The maximum size of their tensors is an important factor in determining the amount of memory required to run a TN analysis. The tensor size is exponential with respect to the number of constraints L in the previous methods [24, 25] and the nilpotent-matrix method, and exponential with respect to the number of physical variables N in the method [23], respectively. On the other hand, the shared-matrix method has a size of $O(\log k)$. Here, the size of the shared matrix k is used. In the several examples of the constraints in Section 4, k is constant at 2 regardless of the values of L or N . Thus, while the nilpotent-matrix method has no advantages over the previous methods with respect to a tensor size, the shared-matrix method potentially has a significant advantage. Note that as shown in Eq. (39), if Kronecker product is used for synthesizing shared matrices, the total tensor size increases exponentially with the number of producting. The tensor size introduced above is only one form of costs. Many other costs can be considered, such as the com-

putational cost of determining the tensor-network parameters. Definition of them will be the subject of future work.

In addition, the extensibility of the constraint conditions can be improved. When another new constraint is added to existing constraints, the constraint is easily encoded by Kronecker product as described in Section 5. Thus, redesigning fully feasible TNs from scratch is not necessary unlike the previous methods [23–25]. However, note that such a MPS synthesis increases the size of tensors exponentially with respect to the number of correlated constraints.

7 Experiment

As an application of the proposed methods, we construct a fully feasible MPS for facility location problem and perform an optimal solution search by using imaginary time evolution. Because many complex global constraints exist, such a problem is considered as a good example for the principle verification and is difficult for the previous TN-based methods to handle in principle. In addition, the purpose of this study is to increase the number of available constraints that can be handled by TNs. Note that we do not claim any performance advantages over conventional non-TN based solvers in facility location problem.

7.1 Fully feasible MPS for facility location problem

In facility location problem [36], $x_{i,j}$ is a binary variable that takes 1 if j th customer is assigned to i th facility and 0 otherwise, and y_i is a variable that takes 1 if i th facility is opened and 0 otherwise. As the constraints, a group of conditions

$$x_{i,j} \leq y_i \quad \text{for } i = 1, \dots, M, j = 1, \dots, N, \quad (41)$$

$$x_{1,j} + x_{2,j} \cdots + x_{M,j} = 1 \text{ for } j = 1, \dots, N \quad (42)$$

is imposed. Here, M is the total number of facility location candidates and N is that of customers. Equation (41) is a many-to-one comparison constraint and Eq. (42) is a linear equality constraint. Therefore, a fully feasible MPS can be constructed by combining the nilpotent-matrix method and shared-matrix method.

As fully feasible MPSs, there are several ways to construct them based on the order of encoding constraint conditions. That is, the forms of the MPSs varies depending on whether Eq. (41) or (42) is encoded first. First, we explain the former case. The order of the variables are rearranged as $x_{1,1}, \dots, x_{1,N}, y_1, \dots, x_{M,1}, \dots, x_{M,N}, y_M$. Then, they are renamed as new physical variables $z_1, z_2, \dots, z_{MN+M}$ for a simple representation of the MPS. Concretely, for the constraint (41), we redefine the index set of the physical variables appearing in each constraint condition $C_i : x_{i,j} \leq y_i$ for $j = 1, 2, \dots, N$ as D_i : $D_1 = \{1, 2, \dots, N+1\}$, $D_2 = \{N+2, \dots, 2N+2\}$, $\dots$, $D_M = \{MN+M-N, \dots, MN+M\}$ so as to describe the sites of the MPS in a serial order.

The tensor of the feasible MPS for the condition C_i is

$$\begin{aligned} \tilde{A}_0^{[u_i+1]0} &= \begin{pmatrix} 1 & 1 \end{pmatrix}, \tilde{A}_0^{[u_i+1]1} = \begin{pmatrix} 1 & 0 \end{pmatrix}, \\ \tilde{A}_0^{[u_i+j]0} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \tilde{A}_0^{[u_i+j]1} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ for } j = 2, \dots, N, \\ \tilde{A}_0^{[u_i+N+1]0} &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tilde{A}_0^{[u_i+N+1]1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{aligned} \quad (43)$$

according to Subsection 4.2. Here, $u_i \equiv (N+1)(i-1)$. Because C_i ($i = 1, \dots, M$) are independent of each other, the fully feasible MPS for Eq. (41)

$$\begin{aligned} A_0^{[u_i+j]0} &= \tilde{A}_0^{[u_i+j]0}, A_0^{[u_i+j]1} = \tilde{A}_0^{[u_i+j]1} \\ \text{for } i &= 1, \dots, M, j = 1, \dots, N+1 \end{aligned} \quad (44)$$

is obtained using the tensors of Eq. (43).

Then, the second constraint (42) is encoded into this MPS. Each constraint condition $C'_j : x_{1,j} + x_{2,j} \cdots + x_{M,j} = 1$ is not independent from the constraint (41). Therefore, it is necessary to construct a fully feasible MPS for facility location problem by taking Kronecker product as detailed in Subsection 5.2. Note that because we are now using the new physical variables z , each constraint condition is re-expressed as $C'_j : z_{u_1+j} + z_{u_2+j} + \cdots + z_{u_M+j} = 1$.

The tensors of the fully feasible MPS for the condition C'_j are

$$\begin{aligned} A_j^{[k]0} &= 1, A_j^{[k]1} = 1 \text{ for } k = 1, \dots, j-1, \\ A_j^{[j]0} &= \begin{pmatrix} 0 & 1 \end{pmatrix}, A_j^{[j]1} = \begin{pmatrix} 1 & 0 \end{pmatrix}, \\ A_j^{[u_i+j]0} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, A_j^{[u_i+j]1} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \text{ for } i = 2, \dots, M-1, \\ A_j^{[u_i+k]0} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, A_j^{[u_i+k]1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ \text{for } i &= 2, \dots, M-1, k = 1, \dots, j-1, j+1, \dots, N+1, \\ A_j^{[u_M+j]0} &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}, A_j^{[u_M+j]1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \\ A_j^{[u_M+k]0} &= 1, A_j^{[u_M+k]1} = 1 \text{ for } k = j+1, \dots, N+1 \end{aligned} \quad (45)$$

according to Eq. (21) and Subsection 5.3. Substituting the tensors in Eq. (44) and (45) into Eq. (39) gives

$$A^{[i]z_i} = A_0^{[i]z_i} \otimes \cdots \otimes A_N^{[i]z_i} \text{ for } i = 1, \dots, MN+M. \quad (46)$$

This is one of the fully feasible MPSs for the constraints (41) and (42) in facility location problem.

Next, we explain another fully feasible MPS by first encoding the constraint (42). The order of the variables are rearranged as $x_{1,1}, x_{2,1}, \dots, x_{M,1}, \dots, x_{1,N}, \dots, x_{M,N}, y_1, \dots, y_M$. Then, they are renamed as new physical variables $z_1, z_2, \dots, z_{MN+M}$. Concretely, for the constraint (42), we redefine the index set of the physical variables appearing in each constraint condition $C_j : \sum_{i=1}^M x_{i,j} = 1$ as $D_1 = \{1, 2, \dots, M\}$, $D_2 = \{M+1, \dots, 2M\}$, $\dots$, $D_N = \{MN-M+1, \dots, MN\}$.

The tensors of the feasible MPS for the condition C_i are

$$\begin{aligned} A_0^{[w_j+1]0} &= \begin{pmatrix} 0 & 1 \end{pmatrix}, A_0^{[w_j+1]1} = \begin{pmatrix} 1 & 0 \end{pmatrix} \text{ for } j = 1, \dots, N, \\ A_0^{[w_j+i]0} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, A_0^{[w_j+i]1} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \\ \text{for } i &= 2, \dots, M-1, j = 1, \dots, N, \\ A_0^{[w_j+M]0} &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}, A_0^{[w_j+M]1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ for } j = 1, \dots, N, \\ A_0^{[w_{N+1}+i]0} &= 1, A_0^{[w_{N+1}+i]1} = 1 \text{ for } i = 1, \dots, M \end{aligned} \quad (47)$$

according to Eq. (21) and Subsection 5.3. Here, $w_j \equiv M(j-1)$.

Then, the first constraint (41) is encoded into this MPS. Each constraint condition $C'_i : x_{i,j} \leq y_i$ for $j = 1, \dots, N$ is not independent from the constraint (42). Therefore, it is necessary to construct a fully feasible MPS by taking Kronecker product as detailed in Subsection 5.2. Note

that because we are now using the new physical variables as $\mathbf{z}$, each constraint condition is re-expressed as $C'_i : z_{w_j+i} \leq z_{w_{N+1}+i}$ for $j = 1, \dots, N$. The tensors of the fully feasible MPS for C'_i are

$$\begin{aligned}
A_i^{[k]0} &= 1, A_i^{[k]1} = 1 \text{ for } k = 1, \dots, i-1, \\
A_i^{[i]0} &= \begin{pmatrix} 1 & 1 \end{pmatrix}, A_i^{[i]1} = \begin{pmatrix} 1 & 0 \end{pmatrix}, \\
A_i^{[w_j+i]0} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, A_i^{[w_j+i]1} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ for } j = 2, \dots, N-1, \\
A_i^{[w_j+k]0} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, A_i^{[w_j+k]1} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \\
&\text{for } j = 2, \dots, N-1, k = 1, \dots, i-1, i+1, \dots, M, \\
A_i^{[w_N+i]0} &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}, A_i^{[w_N+i]1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \\
A_i^{[w_{N+1}+k]0} &= 1, A_i^{[w_{N+1}+k]1} = 1 \text{ for } k = i+1, \dots, M
\end{aligned} \tag{48}$$

according to Subsection 4.2 and 5.3. Substituting the tensors in Eq. (47) and (48) into Eq. (39) gives

$$A^{[i]z_i} = A_0^{[i]z_i} \otimes \dots \otimes A_M^{[i]z_i} \text{ for } i = 1, \dots, MN+M. \tag{49}$$

This is another fully feasible MPS for the constraints (41) and (42) in facility location problem.

We compare the two types of fully feasible MPSs. Their sizes are found to be different as shown in Eq. (46) and (49). The tensor in Eq. (46) is Kronecker product of $N+1$ matrices and the size is at most 2^{N+1} , while that in Eq. (49) is 2^{M+1} . This shows the amount of required memory varies depending on the order of encoding constraints. Therefore, one can choose the memory-efficient MPS based on the size difference between M and N . Nevertheless, in our formalism, the tensor size increases exponentially with respect to M or N because Kronecker products are used.

7.2 Numerical test on fully feasible MPS construction for facility location problem

Figure 8 shows numerical results on required times for constructing fully feasible MPSs. In other words, this is the time to determine all of MPS parameters for facility location constraints (41) and (42), with varying the total number of facility location candidates M and total number of customers N . Here, $M = 1, 2, 3, 4$ and $N = 10, 20, 30, 40, 50$. The baseline is the Cons-training method [25], which is the modified and improved version of the method [24]. These are the methods to construct fully feasible MPSs

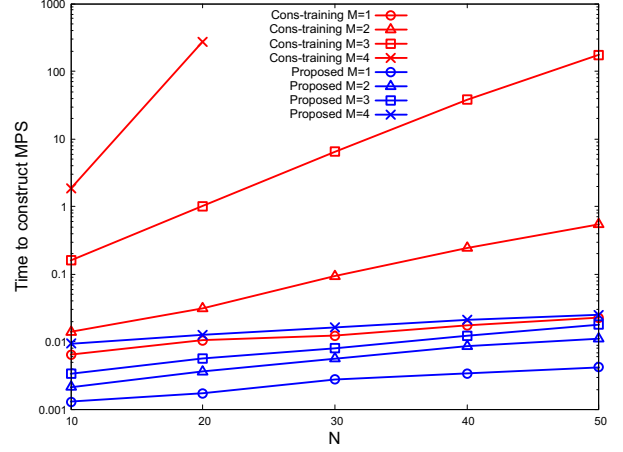


Figure 8: Comparison of required times for constructing fully feasible MPSs. The unit of the vertical axis is seconds. The solid lines shall guide the eye. In the Cons-training method, several conditions under which memory overflow occurred are not plotted.

for linear equality and inequality constraints, as mentioned in Subsection 2.3. Because Eqs. (41) and (42) are linear, the Cons-training method can be applied. This method was implemented with ConstrainTNet [25], to obtain the fully feasible MPS. For our method, the MPS was constructed from Eq. (49), which requires a more moderate tensor size than Eq. (46) because M is smaller than N . The simulations were performed using ITensor library [42] in the Julia environment.

The value of `maxQNs` in ITensor was set to 400 so that MPSs could carry a sufficiently large quantum number for the Cons-training method. After performing constructing MPSs 100 times at each pair of (M, N) , the averaged time was estimated and plotted with N as the abscissa. Refer to Appendix D for the M -dependency. For the Cons-training method, increasing M up to 4 caused calculation failures over $N = 30$ due to memory overflow. From Eqs. (41) and (42), the number of constraint equations rises to $(M+1)N$, which leads to the huge increase in time required to construct MPSs. On the other hand, for our method, the required time was moderate. This may be because the shared-matrix method can encode the whole of Eq. (41) into 2×2 tensors.

In addition, the other previous method [23] requires case divisions for all variables, which makes MPS construction even more intractable due to the combinatorial explosion. Specifically, in $(M, N) = (3, 30)$, there are 2^{93} cases. In this way, our method has efficiency superiority over

conventional TN-based methods of encoding constraints into MPSs for facility location problems.

7.3 Numerical test on optimization of facility location problem

We performed searching for optimal solutions using the fully feasible MPS obtained in Subsection 7.1. We applied imaginary time evolution to the MPS and evaluated the acquisition probabilities of the feasible solutions and optimal solutions. The constraint conditions are given by Eq. (41) and (42), and the cost Hamiltonian to be minimized is

$$\hat{H} = \sum_{i=1}^M \sum_{j=1}^N E_{i,j} \hat{x}_{i,j} + \sum_{i=1}^M F_i \hat{y}_i. \quad (50)$$

For $M = 2, 3, 4, N = 30, 40, 50$, we determined the cost coefficients randomly and generated 500 instances to perform the evaluation. Specifically, in Eq. (50), $E_{i,j}, F_i$ were generated as an integer from the uniform distribution on $[1, 3]$, and the states after imaginary time evolution were sampled. The expected energy was calculated by substituting the state solution into Eq. (50). We also checked whether the state was a feasible solution or an optimal solution to estimate the average probability measuring them. The results are shown in Figs. 9 and 10, respectively.

The optimal solutions can be easily pre-estimated by the following algorithm. First, enumerate the combinations of $\mathbf{y}$ by brute force. Define the index set where $y_i = 1$ as $I_{y=1}$. For each $j = 1, 2, \dots, N$, search for the bit string of $x_{i,j}$ with the smallest cost among $x_{i,j}$ for $i \in I_{y=1}$. This gives a suboptimal solution for the fixed values of $\mathbf{y}$. The optimal solution is with the minimum cost among these suboptimal solutions. Because the total number of facility location candidates M is relatively small in this experiment, the above brute force algorithm can be easily executed.

Next, we explain the details of the optimization using the proposed method. We constructed an MPS $|\psi\rangle$ using Eq. (49). For the initial state $|\psi\rangle$, we performed imaginary time evolution

$$|\psi_t\rangle = \frac{e^{-t\hat{H}} |\psi\rangle}{|e^{-t\hat{H}} |\psi\rangle|} \quad (51)$$

and sampled the final state after evolution.

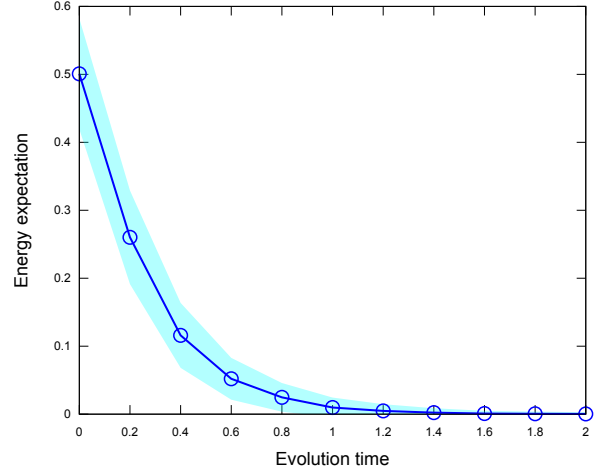


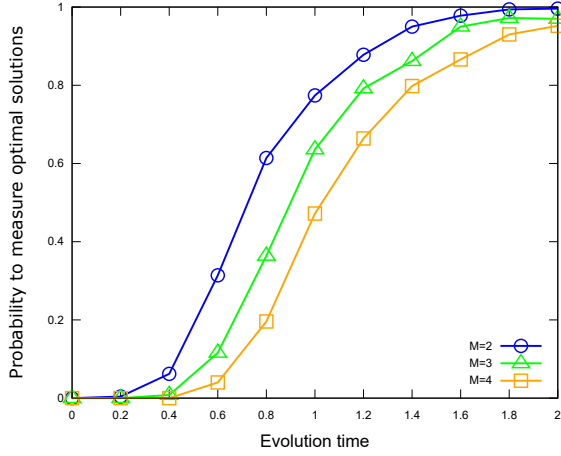
Figure 9: Change of energy expectation values during imaginary time evolution. The total number of facility location candidates M and total number of customers N are fixed at 3 and 40, respectively. The solid lines shall guide the eye. A light-blue shaded area shows the standard deviations.

In Eq. (51), the evolution time t was set in increments of 0.2 in the range $[0, 2]$. Note that because this experiment focuses on a linear form Hamiltonian, as shown in Eq. (50), it is sufficient to perform imaginary time evolution on each site locally and singular value decomposition [2] is not necessary. Thus, the evolution time does not require being divided into steps. In other words, $e^{-t\hat{H}} |\psi\rangle$ is calculated at once and then normalized. If the evolution time t is long, the value of the exponential function in Eq. (51) becomes large, leading to intractable calculation. Thus, a cutoff value was imposed: the upper limit was set to 22026 ($\simeq e^{10}$). The states were sampled with ITensor's `sample` module.

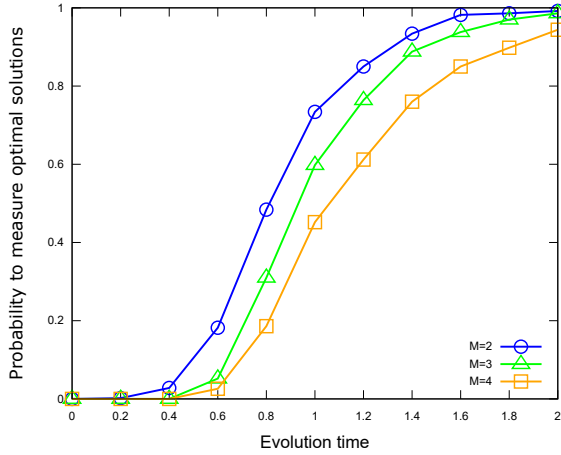
We confirmed that feasible solutions were always obtained. This is trivial because the MPS of Eq. (49) is feasible. The expectation values were min-max normalized by

$$\epsilon_\psi = \frac{\langle \hat{H} \rangle - E_{\min}}{E_{\max} - E_{\min}}. \quad (52)$$

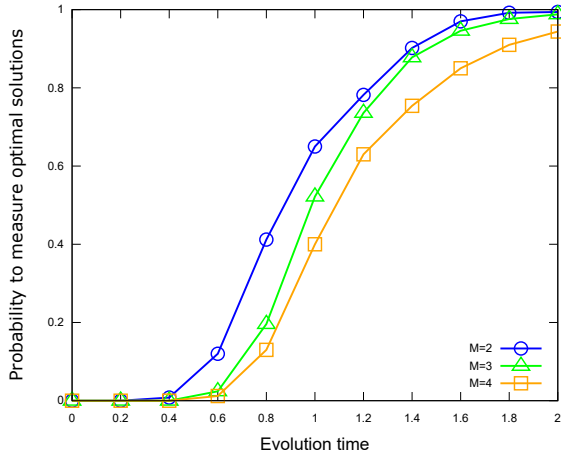
Here, $E_{\min}$ and $E_{\max}$ denote the minimum and maximum energy in the feasible solution space, respectively. These values were calculated by the above brute force algorithm. As shown in Fig. 9, the energy expectation values were confirmed to converge to zero, when M and N were set to 3 and 40, respectively.



(a) $N = 30$.



(b) $N = 40$.



(c) $N = 50$.

Figure 10: Probability of measuring optimal solutions for a fixed total number of customers during imaginary time evolution. (a), (b), and (c) show each plot when the total number of customers N is fixed at 30, 40, and 50, respectively. The solid lines shall guide the eye.

The probability of measuring optimal solutions was plotted, as shown in Fig. 10. The proba-

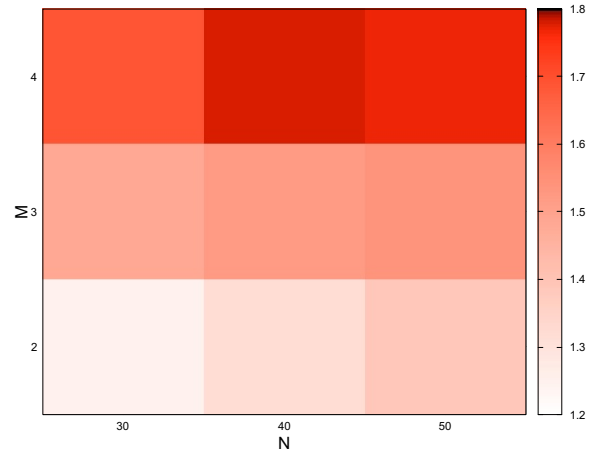


Figure 11: Evolution-time heatmap for optimality. For each pair of the total number of facility location candidates M and total number of customers N , we estimated the time such that the probability of measuring optimal solutions exceeds 0.9. Darker colors mean longer evolution time.

bility was found to approach 1 along imaginary time evolution. It was observed that the proposed method can search only for the feasible solutions and can obtain optimal solutions.

In addition, as the total number of facility location candidates M increases, the probability of measuring optimal solutions tends to decrease, as shown in Fig. 10. Figure 11 shows the evolution-time heatmap for optimality, with respect to the total number of facility location candidates M and total number of customers N . For each (M, N) , we estimated the time $t_{\text{threshold}}$ such that the probability of measuring optimal solutions exceeds 0.9 in the following procedure. From Fig. 10, $t_{\text{threshold}}$ was roughly determined by the intersection point with the probability 0.9. The imaginary time evolution was then performed with varying times in increments of 0.01 around the point, to estimate the minimum time at which the probability would be exceeded. As shown in Fig. 11, $t_{\text{threshold}}$ largely tends to increase towards the right and up. This phenomenon is thought to be reflected by the optimization difficulty. Because the solution space expanded as the problem size increased, the convergence of optimization was considered to require longer evolution time. Similar trends were also observed when the values of M were fixed, as detailed in Appendix E.

Finally, a comparison with the previous methods is explained. The CBC solver [43] was used through Python library PuLP [44]. In all in-

stances, the optimal solutions were obtained by the CBC solver, which is reasonable because this task is a moderate-size optimization. Thus, in the facility location problem, our method is considered to have no advantage over the conventional methods of mathematical programming. In the future, we would like to explore problems where such advantages can be realized.

All reported experiments have been run on Sonoma 14.5 and 4.05 GHz Apple M3 CPU with 16 GB of memory. Note that we used only a single core without parallelization.

8 Discussion and conclusion

Recently, several methods have been proposed to solve constrained combinatorial optimization problems using TNs. By preparing a specific TN to sample states that satisfy constraints, feasible solutions are efficiently searched for without using the penalty function methods. Such a TN is referred to as a feasible TN. These previous studies have been mainly based on profound physics, such as $U(1)$ gauge schemes and high-dimensional lattice models. In this study, we devise to design feasible TNs using elementary mathematics without such a specific knowledge.

The nilpotent-matrix method is a technique for constructing fully feasible MPSs using nilpotent matrices. The method allows us to find MPSs specialized for a linear constraint. In addition, MPS synthesis in Section 5 enables the method to handle multiple constraints. The previous methods in [24, 25] are also specialized for linear constraints and require a backtracking process to ensure $U(1)$ gauge symmetry. On the other hand, the proposed method requires not backtracking but simple matrix manipulation. When another new constraint is added to existing constraints, the constraint is easily encoded by Kronecker product as described in Section 5. Because redesigning fully feasible TNs from scratch is not necessary unlike the previous methods, the extensibility of the constraint conditions can be improved.

The shared-matrix method is another proposal and aims at constructing fully feasible MPSs by sharing tensor matrices across multiple sites. In this method, the matrix parameters are determined so that the trace of the MPS is non-zero for feasible solution states and 0 otherwise. Be-

cause the use of shared matrices significantly reduces the number of parameters, the fully feasible MPSs can be determined by simple algebraic analysis. Their moderate tensor size, which corresponds to that of the shared matrix, may reduce the amount of memory required to run a TN analysis. The shared-matrix method allows us to find the fully feasible MPSs for comparison constraints (many-to-one and domain-wall encoding cases) and constraints used in degree reduction. In addition, not-equal constraints can be handled. Our method is the first to explicitly derive fully feasible TNs for these constraints. The previous method in [23], which is specialized for local constraints, is difficult to apply to such global constraints. If the other methods in [24, 25] are applied to comparison constraints, the number of linear constraints has the same order as that of physical variables N . This causes an exponential increase in the tensor size. Moreover, these methods are also difficult to apply to degree-reduction constraints or not-equal constraints because they are originally specialized for linear constraints. Thus, the shared-matrix method has a potential to encode various types of constraints that are difficult to handle with the previous method.

In summary, the proposed methods have advantages of user-friendly design of TNs and application to a wider range of constraints. For the principle verification, we numerically constructed a feasible TN for facility location problem, to find much faster construction than conventional methods. Then, by performing imaginary time evolution, feasible solutions were always obtained, ultimately leading to the optimal solution.

We discuss future issues. Firstly, regarding the nilpotent-matrix method, the MPS obtained by this method requires a tensor of size at most $d + \sum_{i \in \Delta_-} |a_i| + 1$ for linear inequality/equality constraints with coefficients a_i and constant d . If the value of d is large, or if the absolute value of the negative coefficient is large, an increase of the tensor size may lead to intensive memory usage. Thus, more efficient encoding is required in such cases. In addition, the method of constructing fully feasible MPSs for multiple constraints by Kronecker product makes the tensor size exponentially large with respect to the number of constraints. Therefore, more efficient encoding is also required.

Although we predominantly focused on facility

location problem, the application to other problems is an important issue for the proof of the versatility and superiority. Bridging to quantum gates is also important. In recent years, a technique for converting TNs into equivalent quantum circuits has been proposed [28]. By encoding our feasible MPSs into variational quantum circuits [45], such as Variational Quantum Eigensolver (VQE) [46] or QAOA [47], a new gate-based method for constrained combinatorial optimization will be realized.

Acknowledgements

This work was partially supported by JSPS KAKENHI (Grant Number JP23H05447), the Council for Science, Technology, and Innovation (CSTI) through the Cross-ministerial Strategic Innovation Promotion Program (SIP), “Promoting the application of advanced quantum technology platforms to social issues” (Funding agency: QST), JST (Grant Number JPMJPF2221), and JST CREST (Grant Number JPMJCR19K4). The authors wish to express their gratitude to the World Premier International Research Center Initiative (WPI), MEXT, Japan, for their support of the Human Biology-Microbiome-Quantum Research Center (Bio2Q).

References

- [1] John Preskill. “Quantum computing in the NISQ era and beyond”. *Quantum* **2**, 79 (2018).
- [2] Jacob C Bridgeman and Christopher T. Chubb. “Hand-waving and interpretive dance: an introductory course on tensor networks”. *Journal of Physics A: Mathematical and Theoretical* **50**, 223001 (2017).
- [3] Román Orús. “Tensor networks for complex quantum systems”. *Nature Reviews Physics* **1**, 538–550 (2019).
- [4] Kouichi Okunishi, Tomotoshi Nishino, and Hiroshi Ueda. “Developments in the Tensor Network — from Statistical Mechanics to Quantum Entanglement”. *Journal of the Physical Society of Japan* **91**, 062001 (2022).
- [5] Yuichiro Minato. “Tensor Network Based HOBOSolver”. *arXiv:2407.16106* (2024).
- [6] Shoya Yasuda, Shunsuke Sotobayashi, and Yuichiro Minato. “HOBOTAN: Efficient Higher Order Binary Optimization Solver with Tensor Networks and PyTorch”. *arXiv:2407.19987* (2024).
- [7] Aitor Morais, Eneko Osaba, Iker Pastor, and Izaskun Oregui. “Comparative Analysis of Classical and Quantum-Inspired Solvers: A Preliminary Study on the Weighted Max-Cut Problem”. *arXiv:2504.05989* (2025).
- [8] Jin-Guo Liu, Lei Wang, and Pan Zhang. “Tropical Tensor Network for Ground States of Spin Glasses”. *Physical Review Letters* **126**, 090506 (2021).
- [9] Danylo Lykov, Roman Schutski, Alexey Galda, Valerii Vinokur, and Yuri Alexeev. “Tensor Network Quantum Simulator With Step-Dependent Parallelization”. In *Proceedings of the 2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 582-593 (2022).
- [10] Dimitri P. Bertsekas. “Constrained optimization and Lagrange multiplier methods”. *Academic press* (1982).
- [11] David G. Luenberger and Yinyu Ye. “Linear and Nonlinear Programming (Third edition)”. *Springer* (2008).
- [12] Andrew Lucas. “Ising formulations of many NP problems”. *Frontiers in Physics* **2** (2014).
- [13] Shu Tanaka, Ryo Tamura, and Bikas K. Chakrabarti. “Quantum Spin Glasses, Annealing and Computation”. *Cambridge University Press* (2017).
- [14] Kota Takehara, Daisuke Oku, Yoshiki Matsuda, Shu Tanaka, and Nozomu Togawa. “A Multiple Coefficients Trial Method to Solve Combinatorial Optimization Problems for Simulated-annealing-based Ising Machines”. In *Proceedings of the 2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*, 64-69 (2019).
- [15] Kensuke Tamura, Tatsuhiko Shirai, Hosho Katsura, Shu Tanaka, and Nozomu Togawa. “Performance Comparison of Typical Binary-Integer Encodings in an Ising Machine”. *IEEE Access* **9**, 81032-81039 (2021).
- [16] Kotaro Tanahashi, Shinichi Takayanagi, Tomomitsu Motohashi, and Shu Tanaka. “Appli-

- cation of Ising Machines and a Software Development for Ising Machines”. *Journal of the Physical Society of Japan* **88**, 061010 (2019).
- [17] Mashiyat Zaman, Kotaro Tanahashi, and Shu Tanaka. “PyQUBO: Python Library for Mapping Combinatorial Optimization Problems to QUBO Form”. *IEEE Transactions on Computers* **71**, 838-850 (2022).
- [18] Stuart Hadfield, Zhihui Wang, Bryan O’Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. “From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz”. *Algorithms* **12**, 34 (2019).
- [19] Zhihui Wang, Nicholas C. Rubin, Jason M. Dominy, and Eleanor G. Rieffel. “XY mixers: Analytical and numerical results for the quantum alternating operator ansatz”. *Physical Review A* **101**, 012320 (2020).
- [20] Andreas Bärttschi and Stephan Eidenbenz. “Grover Mixers for QAOA: Shifting Complexity from Mixer Design to State Preparation”. In Proceedings of the **2020 IEEE International Conference on Quantum Computing and Engineering (QCE)**, 72-82 (2020).
- [21] Atsushi Matsuo, Yudai Suzuki, Ikko Hamamura, and Shigeru Yamashita. “Enhancing VQE Convergence for Optimization Problems with Problem-Specific Parameterized Quantum Circuits”. *IEICE Transactions on Information and Systems* **E106.D**, 1772-1782 (2023).
- [22] Hyakka Nakada, Kotaro Tanahashi, and Shu Tanaka. “Inductive Construction of Variational Quantum Circuit for Constrained Combinatorial Optimization”. *IEEE Access* **13**, 73096-73108 (2025).
- [23] Tianyi Hao, Xuxin Huang, Chunjing Jia, and Cheng Peng. “A Quantum-Inspired Tensor Network Algorithm for Constrained Combinatorial Optimization Problems”. *Frontiers in Physics* **10** (2022).
- [24] Javier Lopez-Piqueres, Jing Chen, and Alejandro Perdomo-Ortiz. “Symmetric tensor networks for generative modeling and constrained combinatorial optimization”. *Machine Learning: Science and Technology* **4**, 035009 (2023).
- [25] Javier Lopez-Piqueres and Jing Chen. “Constraining tensor networks”. *arXiv:2405.09005* (2024).
- [26] Markus Bachmayr, Michael Götze, and Max Pfeffer. “Particle number conservation and block structures in matrix product states”. *Calcolo* **59**, 22 (2022).
- [27] Javier Alcazar, Mohammad Ghazi Vakili, Can B. Kalayci, and Alejandro Perdomo-Ortiz. “GEO: Enhancing Combinatorial Optimization with Classical and Quantum Generative Models”. *Nature Communications* **15**, 2761 (2024).
- [28] Manuel S. Rudolph, Jing Chen, Jacob Miller, Atithi Acharya, and Alejandro Perdomo-Ortiz. “Decomposition of matrix product states into shallow quantum circuits”. *Quantum Science and Technology* **9**, 015012 (2023).
- [29] Frank Verstraete and J. Ignacio Cirac. “Renormalization algorithms for Quantum-Many Body Systems in two and higher dimensions”. *arXiv:cond-mat/0407066* (2004).
- [30] Jacob D. Biamonte, Jason Morton, and Jacob Turner. “Tensor network contractions for $\sharp\text{sat}$ ”. *Journal of Statistical Physics* **160**, 1389-1404 (2015).
- [31] Stefanos Kourtis, Claudio Chamon, Eduardo R. Mucciolo, and Andrei E. Ruckenstein. “Fast counting with tensor networks”. *SciPost Physics* **7**, 060 (2019).
- [32] Gleb Ryzhakov and Ivan Oseledets. “Constructive tt-representation of the tensors given as index interaction functions with applications”. In Proceedings of the **11th International Conference on Learning Representations (ICLR)** (2023).
- [33] Jin-Guo Liu, Xun Gao, Madelyn Cain, Mikhail D Lukin, and Sheng-Tao Wang. “Computing solution space properties of combinatorial optimization problems via generic tensor networks”. *SIAM Journal on Scientific Computing* **45**, A1239-A1270 (2023).
- [34] Alejandro Mata Ali, Iñigo Perez Delgado, Marina Ristol Roura, and Aitor Moreno Fdez. de Leceta. “Polynomial-time Solver of Tridiagonal QUBO and QUDO problems with Tensor Networks”. *arXiv:2309.10509* (2024).

- [35] Alejandro Mata Ali, Iñigo Perez Delgado, and Aitor Moreno Fdez. de Leceta. “Traveling Salesman Problem from a Tensor Networks Perspective”. [arXiv:2311.14344](#) (2023).
- [36] Reza Zanjirani Farahani, Maryam Steadie-Seifi, and Nasrin Asgari. “Multiple criteria facility location problems: A survey”. [Applied Mathematical Modelling](#) **34**, 1689-1709 (2010).
- [37] Nicholas Chancellor. “Domain wall encoding of discrete variables for quantum annealing and QAOA”. [Quantum Science and Technology](#) **4**, 045004 (2019).
- [38] Salvador E. Venegas-Andraca, William Cruz-Santos, Catherine McGeoch, and Marco Lanzagorta. “A cross-disciplinary introduction to quantum annealing-based algorithms”. [Contemporary Physics](#) **59**, 174-197 (2018).
- [39] Nike Dattani. “Quadratzation in discrete optimization and quantum mechanics”. [arXiv:1901.04405](#) (2019).
- [40] Egon Balas. “Disjunctive programming”. [Springer](#) (2018).
- [41] Wayne L. Winston. “Operations research: applications and algorithm”. [Thomson Learning, Inc.](#) (2004).
- [42] Matthew Fishman, Steven R. White, and E. Miles Stoudenmire. “The ITensor Software Library for Tensor Network Calculations”. [SciPost Physics Codebases](#), 4 (2022).
- [43] J. Forrest, T. Ralphs, S. Vigerske, Haroldo G. Santos, J. Forrest, L. Hafer, B. Kristjansson, and et al. “coin-or/Cbc: Release releases/2.10.12”. [Zenodo](#) (2024).
- [44] S. Mitchell, A. Kean, A. Mason, M. O’Sullivan, A. Phillips, and F. Peschiera. “Optimization with PuLP — PuLP 2.4 documentation”. <https://coin-or.github.io/pulp/>
- [45] M. Cerezo, Andrew Arrasmith, Ryan Babush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and et al. “Variational quantum algorithms”. [Nature Reviews Physics](#) **3**, 625-644 (2021).
- [46] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, ManHong Yung, Xiao-Qi Zhou, Peter J. Love, Alán AspuruGuzik, and Jeremy L. O’Brien. “A variational eigenvalue solver on a photonic quantum processor”. [Nature Communications](#) **5**, 4213 (2014).
- [47] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. “A Quantum Approximate Optimization Algorithm”. [arXiv:1411.4028](#) (2014).

A Typical examples of shared matrix

Here is an example of a 2×2 matrix used as a shared matrix. The first is an identity matrix

$$I \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

It is used when the value of a physical variable does not affect whether given constraint is satisfied or not. The second is idempotent matrices

$$\begin{aligned} P &\equiv \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \\ Q &\equiv \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \\ R &\equiv \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}. \end{aligned}$$

For example, an idempotent matrix is used in the case where if a certain value is taken on more than one sites, the constraint satisfaction does not depend on how many times this value is taken. Thirdly,

as a non-idempotent matrix, a nilpotent matrix

$$S_1 \equiv \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

exists. For example, it is used in the case where if a certain value is taken on more than two sites, the constraint breaks.

B Kronecker product property

Kronecker product has several properties. The first is the mixed product property, which states for any matrices A, B, C, D such that the products AC and BD are defined,

$$(A \otimes B)(C \otimes D) = AC \otimes BD$$

holds. Secondly, the spectral property states for any square matrices A, B ,

$$\text{tr}[A \otimes B] = \text{tr}[A] \text{tr}[B]$$

is satisfied.

C Feasible MPS for negation of Eq. (20)

The shared-matrix method is applied to the negation of Eq. (20), that is,

$$\sum_{i=1}^N a_i x_i \neq d.$$

Here, the coefficients a_i are integer.

Firstly, the equally-weighted case $\sum_{i=1}^N x_i \neq d$ is considered. In addition, $d > N/3$ is assumed. By using a shared matrix $A = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$,

$$\begin{aligned} A^{[1]0} &= \begin{pmatrix} 1 & 0 \end{pmatrix} I, A^{[1]1} = \begin{pmatrix} 1 & 0 \end{pmatrix} A, \\ A^{[i]0} &= I, A^{[i]1} = A \text{ for } i = 2, 3, \dots, N-1, \\ A^{[N]0} &= I \begin{pmatrix} 1 \\ 0 \end{pmatrix}, A^{[N]1} = A \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{aligned}$$

are assumed for the feasible MPS. Thus,

$$\psi_{x_1, x_2, \dots, x_N} = \left(A \sum_{i=1}^N x_i \right)_{1,1} = \cos \left(\alpha \sum_{i=1}^N x_i \right)$$

is obtained. To eliminate infeasible solutions, the parameter of the shared matrix is set to $\alpha = \pi/2d$. Because $0 \leq \alpha \sum_{i=1}^N x_i < 3\pi/2$ is satisfied thanks to $d > N/3$, the cosine value becomes zero if and only if $\sum_{i=1}^N x_i = d$. Thus, Eq. (3) is ensured and the feasible MPS is obtained. For $d \leq N/3$, rearranging $1 - x_i \rightarrow x_i$ changes the constraint into $\sum_{i=1}^N x_i = N - d$. Here, $N - d > N/3$ is satisfied and the above formulation can be applied.

Next, we consider the general case where $\sum_{i=1}^N a_i x_i$ is arbitrarily weighted. If the coefficient a_i is positive, as explained in Subsection 3.1, A is replaced by A^{a_i} . On the other hand, if the coefficient a_i is negative, A is set to I and I is replaced with A^{a_i} . Whereas the tensor size depends highly on d for $\sum_{i=1}^N a_i x_i = d$ as shown in Section 3, the negation $\sum_{i=1}^N a_i x_i \neq d$ always requires only 2×2 tensors.

The above formulation can be applied to the constraint

$$\sum_{i=1}^N a_i x_i = \text{even.}$$

In this case, the parameter of the shared matrix is set to $\alpha = \pi/2$. In addition, by changing the phase

$$A^{[1]0} = \begin{pmatrix} 0 & 1 \end{pmatrix}, A^{[1]1} = \begin{pmatrix} 0 & 1 \end{pmatrix} A,$$

to extract the sine instead of the cosine, the feasible MPS for $\sum_{i=1}^N a_i x_i = \text{odd}$ is obtained. In this way, the shared-matrix method can handle the congruent-type constraints (i.e. $\sum_{i=1}^N a_i x_i \not\equiv 0 \pmod 3, 0 \pmod 4, \dots$).

D M -dependency on time required to prepare MPSs

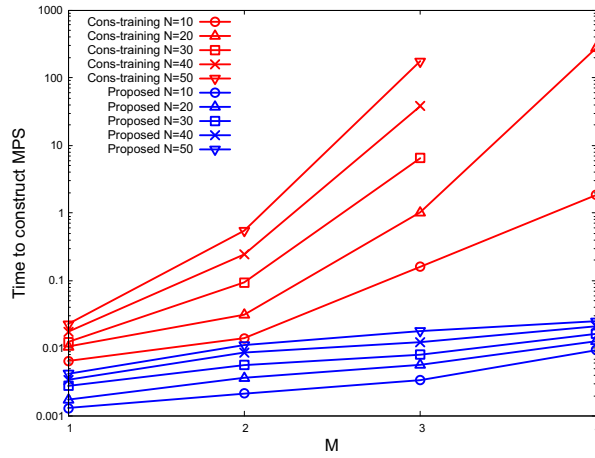
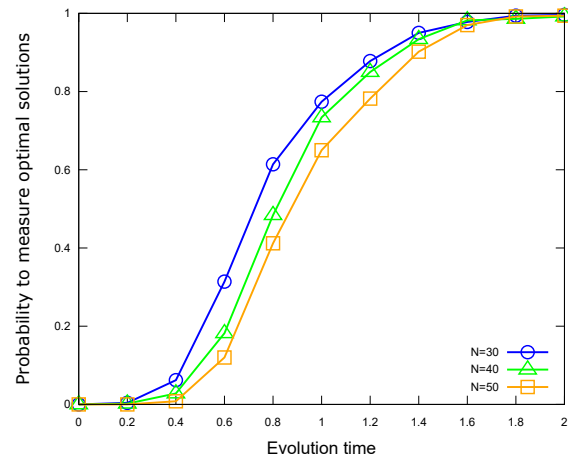


Figure 12: Comparison of required times for constructing fully feasible MPSs, for a fixed total number of customers. The unit of the vertical axis is seconds. In the Cons-training method, several conditions under which memory overflow occurred are not plotted.

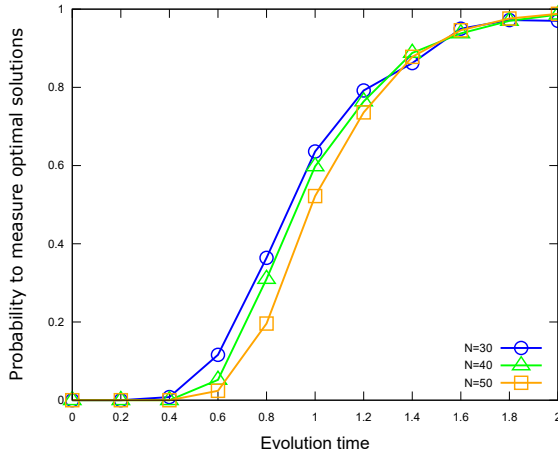
Regarding the MPS construction for facility location problem as discussed in Subsection 7.2, we explain the dependence of the total number of facility location candidates M on the results. As shown in Fig. 12, in the case of the Cons-training method, an increase trend larger than the exponential function can be seen. For our method, the slope along M tends to be larger than that along N . Because we used Kronecker product as described in Section 5, the size of tensors increases exponentially with respect to M . This would make the computational complexity sensitive to the value of M . The cause of the phenomenon will be investigated in the future. Nevertheless, our method is expected to be efficient for constructing the fully feasible MPSs of facility location problem.

E N -dependency on probability to measure optimal solutions

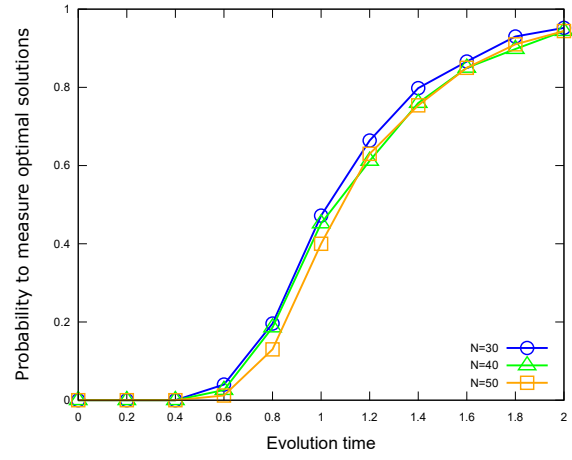
Regarding the optimization of facility location problem as discussed in Subsection 7.3, we explain the dependence of the total number of facility location candidates M on the results. The probability of measuring optimal solutions is plotted, as shown in Fig. 13. The probability was found to approach 1 along imaginary time evolution. Additionally, as the total number of facility location candidates M increases, the probability of measuring optimal solutions tends to decrease. This phenomenon is thought to be reflected by the optimization difficulty, as mentioned in Subsection 7.3.



(a) $M = 2$.



(b) $M = 3$.



(c) $M = 4$.

Figure 13: Probability measuring optimal solutions for a fixed total number of facility location candidates during imaginary time evolution. (a), (b), and (c) show each plot when the total number of facility location candidates M is fixed at 2, 3, and 4, respectively. The solid lines shall guide the eye.