

Subsampling factorization machine annealing

Yusuke Hama*

*Global R&D Center for Business by Quantum-AI Technology (G-QuAT),
National Institute of Advanced Industrial Science and Technology (AIST), 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan*

Tadashi Kadowaki 

*DENSO CORPORATION, 1-1-4, Haneda Airport, Ota-ku, Tokyo 144-0041, Japan
and Global R&D Center for Business by Quantum-AI Technology (G-QuAT),
National Institute of Advanced Industrial Science and Technology (AIST), 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan*



(Received 13 August 2025; accepted 19 November 2025; published 20 February 2026)

Quantum computing and machine learning are state-of-the-art technologies that have been investigated intensively in both academia and industry. The hybrid technology of these two ingredients is expected to be a powerful tool to solve complex problems in many branches of science and engineering such as combinatorial optimization problems and accelerate the creation of next-generation technologies. In this work, we develop an algorithm to solve a black-box optimization problem by improving Factorization Machine Annealing (FMA) such that the training of a machine learning model called factorization machine is performed not by a full dataset but by a subdataset that is sampled from a full dataset: Subsampling Factorization Machine Annealing (SFMA). According to such a probabilistic training process, the performance of FMA on exploring a solution space gets enhanced. As a result, SFMA exhibits balanced performance of exploration and exploitation, which we call exploitation-exploration functionality. We conduct numerical benchmarking tests to compare the performance of SFMA with that of FMA. Consequently, SFMA certainly exhibits the exploration-exploitation functionality and outperforms FMA in speed and accuracy. In addition, the performance of SFMA can be further improved by sequentially using two subsampling datasets with different sizes such that the size of the latter dataset is substantially smaller than the former. Such a substantial reduction not only enhances the exploration performance of SFMA but also enables us to run it with correspondingly low computational cost even for a large-scale problem. These results indicate the effectiveness of SFMA in a certain class of black-box optimization problems of significant size: the potential scalability of SFMA in solving large-scale problems with correspondingly low computational cost. The exploitation-exploration functionality of SFMA is expected to pave the way for solving various complex combinatorial optimization problems in the real world and designing next-generation technologies that become the basic building blocks for industrial advancements.

DOI: [10.1103/qgpg-6dmr](https://doi.org/10.1103/qgpg-6dmr)

I. INTRODUCTION

Optimization problems are ubiquitous in science and engineering (physics, chemistry, computer science, etc.) and have been studied extensively for long decades [1–6]. It includes various kinds of fundamental problems such as traveling salesperson problem, max-cut problem, and graph coloring problem, and the algorithms for solving these problems are expected to be the basic building blocks for advancing industrial technologies such as logistics and production planning [5,6].

In physics, combinatorial optimization problem has been investigated intensively by using a powerful and simple for-

malism called quadratic unconstrained binary optimization (QUBO), or equivalently, Ising model [1,3,5–11]. The canonical algorithms for the optimization based on QUBO (Ising model) include simulated annealing (SA), quantum annealing (QA), and/or adiabatic quantum computing [2,3,5–28]. On the other hand, in gate-based quantum computing variational algorithms so-called quantum approximate optimization algorithm and quantum alternating operator ansatz have been developed and utilized for solving combinatorial optimization problems [27,29–34]. Furthermore, owing to the advancement of both classical and quantum computers (quantum annealer and gate-based quantum computer), such physics techniques become able to be implemented on these machines and solve many types of optimization problems related to societal issues as well as the development of technologies, for instance, materials engineering, drug discovery, portfolio optimization, and protein folding [5,12,30,35–51]. Meanwhile, the development of machine learning techniques or algorithms (both classical and quantum) has been progressing [4,52–70], and such data-driven approaches have been playing significant roles in, for instance, simulations and designing in materials science

*Contact author: hama.yusuke@aist.go.jp

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

and engineering, i.e., materials informatics (MI) [71–80]. Due to such advancements, in recent years the hybrid techniques of quantum computing (quantum annealing) and machine learning have been developed, for example, so-called Factorization Machine Annealing (FMA) [81–94], variants of FMA [95,96], and Bayesian Optimization of Combinatorial Structures (BOCS) [85,86,97–99]. These methods have been applied to solve a problem called black-box optimization (BBO), which is an optimization problem where a functional form of an objective function is unknown. In most actual cases, the optimization problems are considered to be BBO since the objective functions (output variables) are generated from input variables through complex processes. For instance, the optimization of physical properties of target composite materials is BBO in general, because how the physical properties (objective functions) are generated from ingredients (input variables) are complex processes [39,76,81,89,92]. Both FMA and BOCS have been applied to materials simulations and designing and many successful results have been reported [81–85,87–89,91–93]. Since the advancement of the quantum computing and machine learning techniques is expected to continue, it is an important issue to further develop effective hybrid techniques or algorithms of these two ingredients. It is expected that such hybrid algorithms are going to be powerful tools to tackle a wide range of optimization problems in the real world.

Toward the above goal, in this work we develop an effective algorithm for BBO by improving FMA. In general, the key to effectively solve combinatorial optimization problems is to coherently perform exploration and exploitation in solution spaces. That is, we search candidate solutions in a wide range of solution spaces in the first half of optimization processes (exploration), while we concentrate on finding the best candidate solutions in the second half (exploitation) [2,100]. On the other hand, we consider that FMA has good performance on exploitation but not on exploration since it is basically a point-estimation approach with regard to machine learning parameters. By taking this limitation into account, we improve FMA by amplifying the performance of exploration, which is done by training a machine learning model with a subdataset that is sampled from a full dataset. We call this algorithm Subsampling Factorization Machine Annealing (SFMA). In SFMA, a machine learning model is trained probabilistically and the exploration performance of FMA gets enhanced. As a result, SFMA exhibits the balanced performance of exploration and exploitation: exploration-exploitation functionality. To numerically check whether SFMA truly possess the exploration-exploitation functionality and test its utility in numerous black-optimization problems, we perform numerical experiments on black-box optimization problems called lossy compression of data matrices [86,101,102]. These experiments are conducted by benchmarking SFMA against FMA. Consequently, SFMA certainly possesses the exploration-exploitation functionality and exhibits the faster convergence to the optimal solutions with higher accuracy than FMA. Moreover, its exploration performance can be further enhanced by serially utilizing two different subsampling datasets such that the size of the latter dataset is significantly smaller than the former. Namely, we are able to solve a large-scale problem by tailoring the size of a subdataset to

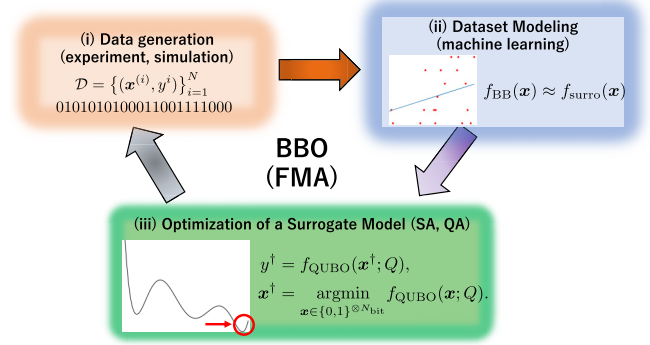


FIG. 1. Schematic of a procedure to conduct BBO. It is comprised of three steps: (i) data generation, (ii) dataset modeling, and (iii) optimization of a surrogate model.

be correspondingly small, which implies that SFMA can be run with correspondingly low computational cost. As a result, the exploitation performance (accuracy) of SFMA is amplified as well. This is the significant finding in our results and is expected to be a powerful tool for solving a complex problem in the real world.

The paper is structured as follows. In Sec. II, we briefly summarize the concept of BBO and the framework of FMA. In Sec. III, which is one of the main sections of this paper, we explain how to develop SFMA. Furthermore, we discuss the advantages of SFMA with respect to computational cost and scalability. In Sec. IV, which is the second main section of this paper, we demonstrate the numerical experiments on SFMA and analyze the results. Section V is devoted to the conclusions and outlook of this paper.

II. BLACK-BOX OPTIMIZATION AND FACTORIZATION MACHINE ANNEALING

To understand how Subsampling Factorization Machine Annealing is developed (described in detail in Sec. III), in this section, first we explain the concept of black-box optimization. After that, we explain the framework of FMA and its characteristic from a viewpoint of exploration and exploitation. In the following, we assume that the optimal solution to a black-box function is the global minimum.

A. Black-box optimization

The goal of BBO is to find the optimal solution to a black-box objective function that we denote by f_{BB} . Such an optimization problem is solved using a data-driven approach represented by a schematic in Fig. 1. BBO is conducted in three basic steps: (i) data generation, (ii) dataset modeling, and (iii) optimization of a surrogate model. Step (i) is performed by, for instance, experiments or numerical simulations. Step (ii) is equivalent to the construction of a machine learning model or a surrogate model using a dataset at hand. These three steps are performed iteratively. Hereinafter, let us call this sequential process BBO loop. In mathematical terms, suppose we have given N_0 data samples consisting of input (explanatory) variables $\mathbf{x}^{(i)}$ and output (objective) variables $y^{(i)} (= f_{BB}(\mathbf{x}^{(i)}))$ with $i =$

$1, \dots, N_0$, i.e., the generation of an initial dataset in step (i). We describe such a dataset by a form $\mathcal{D}_0 = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N_0} = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(i)}, y^{(i)}), \dots, (\mathbf{x}^{(N_0)}, y^{(N_0)})\}$. We assume $\mathbf{x}^{(i)}$ to be N_{bit} -component binary vectors, i.e., $\mathbf{x}^{(i)} = (x_1^{(i)}, \dots, x_l^{(i)}, \dots, x_{N_{\text{bit}}}^{(i)})$, where $x_l^{(i)} \in \{0, 1\}$. Next, in step (ii) we construct a surrogate model that we denote by $f_{\text{surrogate}}(\mathbf{x}; \mathcal{Q}^{(0)})$, where $\mathcal{Q}^{(0)}$ denotes a collection of trainable parameters obtained by $\mathcal{D}_0$ and a machine learning algorithm. Then, in step (iii) we find the binary optimal solution to $f_{\text{surrogate}}(\mathbf{x}; \mathcal{Q}^{(0)})$ and denote it by $\mathbf{x}^{\dagger(1)}$. After that, we add $(\mathbf{x}^{\dagger(1)}, y^{\dagger(1)})$ to $\mathcal{D}_0$, where $y^{\dagger(1)} = f_{\text{BB}}(\mathbf{x}^{\dagger(1)})$. The dataset at hand is augmented such that $\mathcal{D}_1 = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(N_0)}, y^{(N_0)}), (\mathbf{x}^{(N_0+1)}, y^{(N_0+1)})\}$, where $\mathbf{x}^{(N_0+1)} \equiv \mathbf{x}^{\dagger(1)}$, $y^{(N_0+1)} \equiv y^{\dagger(1)}$. We repeat the BBO loops $N_{\text{ite}} - 1$ ($N_{\text{ite}} \geq 2$) more times. As a result, we obtain the final dataset $\mathcal{D}_{N_{\text{ite}}} = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(N_{\text{tot}})}, y^{(N_{\text{tot}})})\}$, where $N_{\text{tot}} = N_0 + N_{\text{ite}}$. Then, we obtain an approximate optimal solution to $f_{\text{BB}}(\mathbf{x})$ that we denote by $(\tilde{\mathbf{x}}^*, \tilde{y}^*)$, where $\tilde{y}^* = \min\{y^{(1)}, \dots, y^{(N_{\text{tot}})}\}$. In the following, we denote the true optimal solution by $(\mathbf{x}^*, y^*)$ to distinguish it from $(\tilde{\mathbf{x}}^*, \tilde{y}^*)$.

Let us end this section by making a comment to avoid misunderstanding of BBO. In general, the goal of machine learning (supervised learning) is to train a model with a given dataset and use it to make a quantitative prediction for new input data. To achieve this goal, machine learning is processed not only with training data but also with validation data and test data in order to avoid overfitting. In contrast, in BBO we use a trained (surrogate) model not for a prediction task but for finding the global minimum of a black-box function under consideration. Therefore, we do not split the data $\mathcal{D}_a$ into the training and validation data to calculate a surrogate model. Instead, the dataset at hand is fully used as training data and a surrogate model is built with it.

B. Factorization machine annealing

FMA is the BBO algorithm where the surrogate model is factorization machine (FM) and the optimizer applied in step (iii) is an annealer such as SA and QA. By denoting $\mathbf{x}$ to be an N_{ite} -dimensional binary vector, the functional form of FM is given in terms of $\mathbf{x}$ by [81–94, 103]

$$f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta}) = w_0 + \sum_{i=1}^{N_{\text{bit}}} w_i x_i + \sum_{1 \leq i < j \leq N_{\text{bit}}} \left(\sum_{l=1}^k v_{i,l} v_{j,l} \right) x_i x_j, \quad (1)$$

where x_i (x_j) is the i th (j th) component of $\mathbf{x}$, $\boldsymbol{\theta} = (w_0, \dots, w_{N_{\text{bit}}}, v_{1,1}, \dots, v_{N_{\text{bit}},k})^T$ are trainable parameters of FM (FM parameters) that are real, and k is a hyperparameter that represents the expressivity of FM. By using the relation $x_i^2 = x_i$, we see that $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta})$ in Eq. (1) has the same mathematical structure with a QUBO function. That is, by writing matrix elements of QUBO by $Q_{i,j}$, we observe that $Q_{i,i} = w_i$ and $Q_{i,j} = \sum_{l=1}^k v_{i,l} v_{j,l}$ ($i \neq j$). Although $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta})$ in Eq. (1) includes the constant term w_0 , this does not affect optimization processes and annealers can be utilized. In FMA, the computational complexity to learn the coefficients $v_{i,l}$ (off-diagonal matrix elements of FM) is $\mathcal{O}(N_{\text{bit}}k)$, while the number of off-diagonal elements of a QUBO matrix is $N_{\text{bit}}(N_{\text{bit}} - 1)/2$, i.e., the computational cost to train off-

diagonal elements of QUBO matrices is $\mathcal{O}(N_{\text{bit}}^2)$. By tuning the hyperparameter k to be $N_{\text{bit}}/2 - 1$ or less than that, we become able to train FM functions with lower computational complexity than that for training general QUBO functions. This is one of the advantages of adopting FM as a surrogate model.

Let us end this section by discussing the performance of FMA from a viewpoint of the exploration and exploitation performance. FM models $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta})$ (FM parameters $\boldsymbol{\theta}$) are trained by minimizing a loss function [e.g., mean square error (MSE)] using a gradient-based optimization method such as stochastic gradient descent (SGD) and Adam. In such a scheme, $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta})$ are trained deterministically for a definite dataset: From this perspective, it can be considered that FM parameters are trained by a point-estimation approach. Such a way of training surrogate models is in contrast to the approach taken in BOCS. That is, in BOCS, the matrix elements of QUBO matrices (surrogate-model parameters) are sampled from a posterior probability distribution [85,86,97–99], which implies that surrogate models are nondeterministic or the uncertainty of trainable parameters are explicitly taken into account. When a given dataset is comprised of data points that belong to a vicinity of a local minimum, we consider that there is a possibility that FMA does not work properly. That is, even if we repeat running BBO loops, a series of FM models is trained in a way that the minimum point of every FM model could be equivalent to such a local minimum. In such a circumstance, we consider that an annealer keeps on selecting the best candidate solutions to be this local minimum point. This indicates that there is a possibility that the candidate solutions searched by FMA get trapped into a local minimum. We consider that such an aspect is a potential limitation of utilizing FMA.

III. SUBSAMPLING FACTORIZATION MACHINE ANNEALING

In this section, we explain how to create SFMA and its characteristics. After that, we discuss the advantages of SFMA over FMA and BOCS.

A. Sampled subdataset

SFMA is constructed by a subdataset that is sampled from a full dataset as shown schematically in Fig. 2, and this is the main difference from FMA. Let us denote a subdataset created in an a th BBO loop ($a = 1, \dots, N_{\text{ite}}$) by $\mathcal{B}_a = \{(\mathbf{x}^{(a,i)}, y^{(a,i)})\}_{i=1}^{|\mathcal{B}_a|}$, where $|\mathcal{B}_a|$ is the size of $\mathcal{B}_a$. It is controlled by a hyperparameter R ($0 < R < 1$) via a relation $|\mathcal{B}_a| = \lfloor R \cdot |\mathcal{D}_a| \rfloor$, where $\lfloor \dots \rfloor$ denotes the floor function and $|\mathcal{D}_a|$ is the size of the full dataset obtained in the a th BBO loop. When the FM function $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta}^{(a)})$ is trained by an optimization method (e.g., SGD and Adam) with $\mathcal{B}_a$, the values of $\boldsymbol{\theta}^{(a)}$ fluctuate or they are trained probabilistically. We comment on that such a probabilistic way of training surrogate models is similar to an approach taken in BOCS, which is performed by sampling model parameters according to posterior probability distributions [85,86,97–99]. Correspondingly, the best solution to $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta}^{(a)})$, which is $(\mathbf{x}^{\dagger(a)}, y^{\dagger(a)}) = (\mathbf{x}^{(a+1)}, y^{(a+1)})$, also fluctuates. Such a fluctuation is going to be a driving force

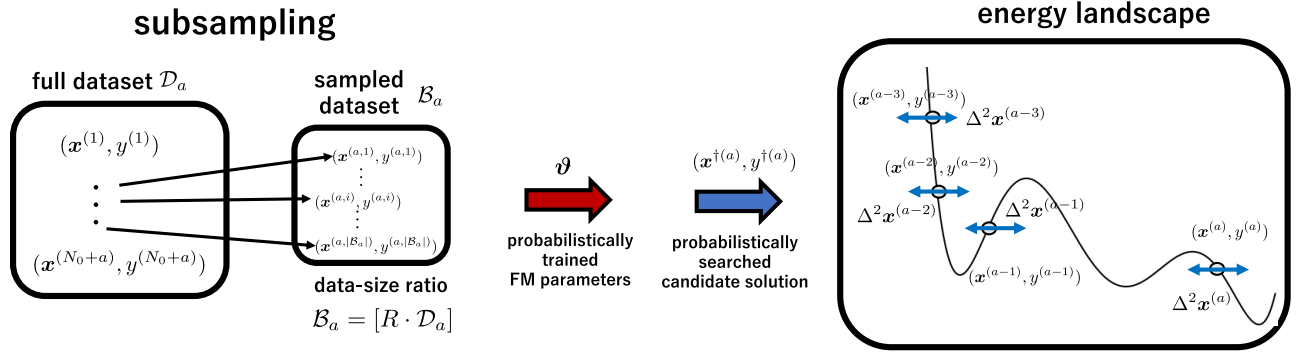


FIG. 2. (a) Illustration of how to create a sampled dataset $\mathcal{B}_a$. It is created by sampling the elements of $\mathcal{D}_a$ according to a probability distribution, and the relation between the size of $\mathcal{B}_a$ ($|\mathcal{B}_a|$) and that of $\mathcal{D}_a$ ($|\mathcal{D}_a|$): $|\mathcal{B}_a| = [R \cdot |\mathcal{D}_a|]$ ($0 < R < 1$). The data point $(x^{(a,i)}, y^{(a,i)})$ is an i th ($i = 1, \dots, |\mathcal{B}_a|$) element of $\mathcal{B}_a$. Owing to such a probabilistic construction of $\mathcal{B}_a$, the FM function $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta}^{(a)})$ is derived probabilistically (FM parameters $\boldsymbol{\theta}$ are trained probabilistically), and correspondingly, the a th best candidate solution $(\mathbf{x}^{\dagger(a)}, \mathbf{y}^{\dagger(a)}) (= (\mathbf{x}^{(a+1)}, \mathbf{y}^{(a+1)}))$ is selected from the solution space associated with a deviation. As a result, we become able to explore a wider range of the energy landscape of $f_{\text{BB}}(\mathbf{x})$ than a case of running FMA as indicated by deviations of the best candidate solutions denoted by symbols $\Delta^2 \mathbf{x}^{(\alpha)}$ ($\alpha = a-3, \dots, a$) and the exploration performance gets strengthened.

to explore a broader range of a solution space than an exploration range that can be achieved by FMA as schematically represented in Fig. 2. To support the argument above, we note that it has been investigated in Refs. [104,105] that the training of machine learning methods using subsampling, which can be called mini-batch methods as well, enables the training of parameters in more nondeterministic ways than those using large-scale subdatasets (large-batch methods). In other words, subsampling (mini-batch) methods are able to explore a broader range of trainable parameter spaces. As a result, the exploration performance of FMA gets strengthened and it is expected that SFMA exhibits the balanced performance of exploration and exploitation: exploration-exploitation functionality. Due to this exploration-exploitation functionality, it is expected that SFMA finds the optimal solution such that in the first half of optimization processes it exhibits good performance on exploring a wide range of solution space (exploration phase). This is because the dataset size $|\mathcal{B}_a|$ is small and the deviation of an FM function as well as its best solution are large. In the second half, it is expected to perform well on searching for the best candidate solutions (exploitation phase) since at this stage the dataset size at hand is sufficiently big, which indicates that the candidate solutions have been explored sufficiently. In Algorithm 1, we present a pseudocode for SFMA. The details of standardization given in Algorithm 1 are described in Appendix A.

B. Computational cost and scalability

Let us highlight the advantages of SFMA by making comparisons with FMA and BOCS. First and foremost, the significant advantage of SFMA is that it is implementable with lower computational cost than the other two algorithms. That is, when we utilize FMA or BOCS, the size of a dataset at hand gets larger as BBO iterations process, which indicates that the computational cost for machine learning becomes more expensive in the later training processes. In particular, the implementation of BOCS is highly demanding since the computational cost to calculate a posterior distribution per

sample is, for instance, $\mathcal{O}(p^3)$ or $\mathcal{O}(p \cdot |\mathcal{D}_a|^2)$ ($p = 1 + N_{\text{bit}} + N_{\text{bit}}(N_{\text{bit}} + 1)/2$), although it possess superior exploration performance [85,86,97–99]. Such an issue becomes prominent for a large-scale problem. On the other hand, SFMA has a potential to overcome this issue, which can be done in the following way. When dataset size as well as N_{bit} become substantially large, we take the value of R to be correspondingly small, which implies the R times reduction of the computational cost for training a surrogate model (FM model) than that using a full dataset. Furthermore, taking substantially small R is beneficial not only for saving the computational cost for training an FM model but also for solving a large-scale optimization problem with high accuracy. This is because the deviation of training an FM model enhances by taking R to be significantly small, and correspondingly, the best candidate

ALGORITHM 1. SFMA.

Input : Initial dataset $\mathcal{D}_0$, BBO iteration number N_{ite} , and ratio R .

Output: Final dataset $\mathcal{D}_{N_{\text{ite}}}$ and the optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$.

Procedure

- Generate a dataset $\mathcal{D}_1$ via a single BBO loop with $\mathcal{D}_0$.
- **for** $a \leftarrow 1$ **to** $N_{\text{ite}} - 1$ **do**
 1. Create a subsampled dataset $\mathcal{B}_a$ with $\mathcal{D}_a$ and a ratio R .
 2. Perform the standardization, $y^{(i)} \rightarrow \hat{y}_{\text{stand}}^{(i)}$, and construct a standardized sampled dataset $\hat{\mathcal{B}}_a$.
 3. Train $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta}^{(a)})$ with $\hat{\mathcal{B}}_a$.
 4. Run an annealing process and find the best solution $(\mathbf{x}^{\dagger(a)}, \mathbf{y}^{\dagger(a)})$.

return The final dataset $\mathcal{D}_{N_{\text{ite}}}$ and the optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$.

solution obtained by an annealer fluctuates substantially. This implies the further amplification of the exploration performance, and correspondingly, the exploitation performance is enhanced as well. In this regard, we consider that SFMA has scalability in solving large-scale problems. Note that the exploration performance of SFMA might be weaker than that of BOCS even for a substantially small R ; however, we are still able to explore a broad range of a solution space and SFMA is practically more feasible than BOCS according to the lower computational cost.

IV. NUMERICAL EXPERIMENTS

Let us demonstrate our numerical experiments on SFMA and examine whether it truly possesses the exploration-exploitation functionality or not and yields the convergence to the optimal solutions to the given problems with high accuracy. We perform these tasks by benchmarking SFMA against FMA. The problem we choose for our numerical experiments is called lossy compression of data matrices [86,101,102], which is an issue or a technique used in, for instance, image recognition, audio data processing, and edge computing. Moreover, lossy compression is related to other techniques such as Non-negative Matrix Factorization and Non-negative/Binary Matrix Factorization [106–108], which is utilized in various real-world problems, for instance, image processing and MI [86,101,106–108]. By showing the effectiveness of SFMA on such a ubiquitous problem, we consider that we can use it as an indicator to emphasize the utility of SFMA in numerous combinatorial optimization problems in the real world. In this section, first we explain the problem setting of lossy compression of data matrices. Next, we describe the numerical setup of our experiments. Finally, we discuss our numerical results.

A. Problem setting

The objective of this combinatorial optimization problem is to approximately decompose a target $N \times D$ matrix W in terms of M ($N \times K$ integer matrix) and C ($K \times D$ real matrix), i.e., $W \simeq MC$. The matrix elements of M take either 1 or -1 . By using a pseudoinverse of M , which is $(M^T M)^{-1} M^T$, the matrix C is approximately re-expressed as $C \simeq (M^T M)^{-1} M^T W$, and we obtain $W \simeq V(M, W)$ with $V(M, W) = M(M^T M)^{-1} M^T W$. To put it another way, our goal is to construct $V(M, W)$ as a function of M that is sufficiently close to the target matrix W . To do this, we introduce two functions of M defined by

$$\begin{aligned} f_W(M) &= W - V(M, W), \\ f_{\text{BB},W}(M) &= \|f_W(M)\|_F. \end{aligned} \quad (2)$$

The notation $\|\cdot\|_F$ in the above equation represents the Frobenius norm defined by $\|A\|_F = \sum_{i=1}^{n_r} \sum_{j=1}^{n_c} |a_{i,j}|^2$, where $a_{i,j}$ is an (i, j) th matrix element of A and n_r (n_c) is the number of rows (columns) of A . Our aim is to find a matrix M that makes the value of $f_{\text{BB},W}(M)$ as small as possible. Namely, the problem can be reformulated as the combinatorial optimization problem such that the black-box objective function is $f_{\text{BB},W}(M)$ and the optimal solution is $M^* = \text{argmin}_{M \in \{-1,1\}^{N \times K}} f_{\text{BB},W}(M)$. This optimization problem can

be reformulated to a binary optimization in the following way. Let us rewrite M into a form $M = [M_1^T, \dots, M_i^T, \dots, M_N^T]^T$, where $M_i = [m_{i,1}, \dots, m_{i,j}, \dots, m_{i,K}]^T$ with $m_{i,j}$ denoting the (i, j) th element of M . Since $m_{i,j} = \pm 1$, it can be regarded as a spin variable. Then, by introducing a new variable $x_{i,j}^M = (1 + m_{i,j})/2$, we obtain an N_{bit} -dimensional binary vector $\mathbf{x}^M = [x_{1,1}^M, \dots, x_{i,j}^M, \dots, x_{N,K}^M]^T \in \{0, 1\}^{N_{\text{bit}}}$ with $N_{\text{bit}} = N \cdot K$. Therefore, this optimization problem is recast into the black-box optimization problem such that the black-box function is $f_{\text{BB},W}(M)$ in Eq. (2) and the solution space is the $2^{N_{\text{bit}}}$ -dimensional binary solution space.

B. Numerical setup

Let us briefly explain the setup of our numerical experiments. For the details, see Appendix B. In our numerical experiments, we create ten types of W matrices (problem instances) that we denote by W_n ($n = 0, 1, \dots, 9$) and set $N_{\text{bit}} = 12, 16, 20$. The parameters (N, D, K) are taken to be $(6, 50, 2)$, $(8, 50, 2)$, and $(10, 50, 2)$ for $N_{\text{bit}} = 12, 16$, and 20 , respectively. These W matrices represent model parameters of the shrunk versions of the final fully connected layer of VGG16 convolutional neural network [109]. The repetition number N_{ite} is set to be $N_{\text{ite}} = C_{\text{FM}} \cdot N_{\text{bit}}^2 + 1$ with $C_{\text{FM}} = 2, 4, 6$. Meanwhile, we take the hyperparameter k [see Eq. (1)] to be $k = N_{\text{bit}}/2 - 1$. In such settings, the computational cost is smaller than that for training general QUBO functions. The sampled datasets $\mathcal{B}_a$ are constructed according to uniform sampling. Namely, the elements of $\mathcal{B}_a$ are sampled from $\mathcal{D}_a$ according to uniform distributions with the probabilities $1/|\mathcal{D}_a|$ and a fixed random seed. For doing so, we allow duplicate selections.

Finally, we discuss the quantities that are calculated in our experiments. We calculate N_{samp} samples of the datasets $\mathcal{D}_a$ that we denote by $\{\mathcal{D}_{a,\alpha}\}_{\alpha=1}^{N_{\text{samp}}}$, where $\mathcal{D}_{a,\alpha}$ is the dataset obtained in the α th round of the sampling. The reason why we perform such sampling is that the loss functions, which are chosen to be MSE, include fourth-degree terms in the FM parameters $v_{i,l}$ that are responsible for nonconvexities. Therefore, when the initial values of θ differ, in general, the final (updated) values of θ and the final dataset $\mathcal{D}_{N_{\text{ite}}}$ differ as well. Thus, the benchmarking of these two algorithms must be done by some statistical quantities. In our experiments, first we introduce a quantity defined by $y_{\min,\alpha}^{(a)} = \min_{l \in \{1, \dots, N_{\text{in}}+a\}} y_{\alpha}^{(l)}$, where $y_{\alpha}^{(l)} = f_{\text{BB}}(\mathbf{x}_{\alpha}^{(l)})$ with $\mathbf{x}_{\alpha}^{(l)}$ denoting an l th component of the input variable of $\mathcal{D}_{a,\alpha}$. $y_{\min,\alpha}^{(a)}$ describes the minimum value obtained up to the a th round of the BBO loop in the α th sampling. Next, we introduce two statistical quantities defined by $\bar{y}_{\min}^{(a)} = (1/N_{\text{samp}}) \sum_{\alpha=1}^{N_{\text{samp}}} y_{\min,\alpha}^{(a)}$ and $R_{\min,\text{success}}^{(a)} = N(y_{\min,\alpha}^{(a)} = y^*)/N_{\text{samp}}$. The quantity $\bar{y}_{\min}^{(a)}$ describes how the candidate solutions obtained by the BBO loops converge to the global minimum (minima) of the given problems as the iteration number a (or the dataset size $|\mathcal{D}_a|$) increases. Meanwhile, $N(y_{\min,\alpha}^{(a)} = y^*)$ is the number describing how many $y_{\min,\alpha}^{(a)}$ are equal to the optimal value y^* . In other words, it indicates how many samples out of N_{samp} trials succeeded in finding the optimal solution after BBO loops were performed $N_{\text{it},a} = |\mathcal{D}_a| - |\mathcal{D}_0|$ times. Thus, $R_{\text{success}}^{(a)}$ is the percentage of $y_{\min,\alpha}^{(a)}$ such that $y_{\min,\alpha}^{(a)} = y^*$, i.e., the success rate of SFMA

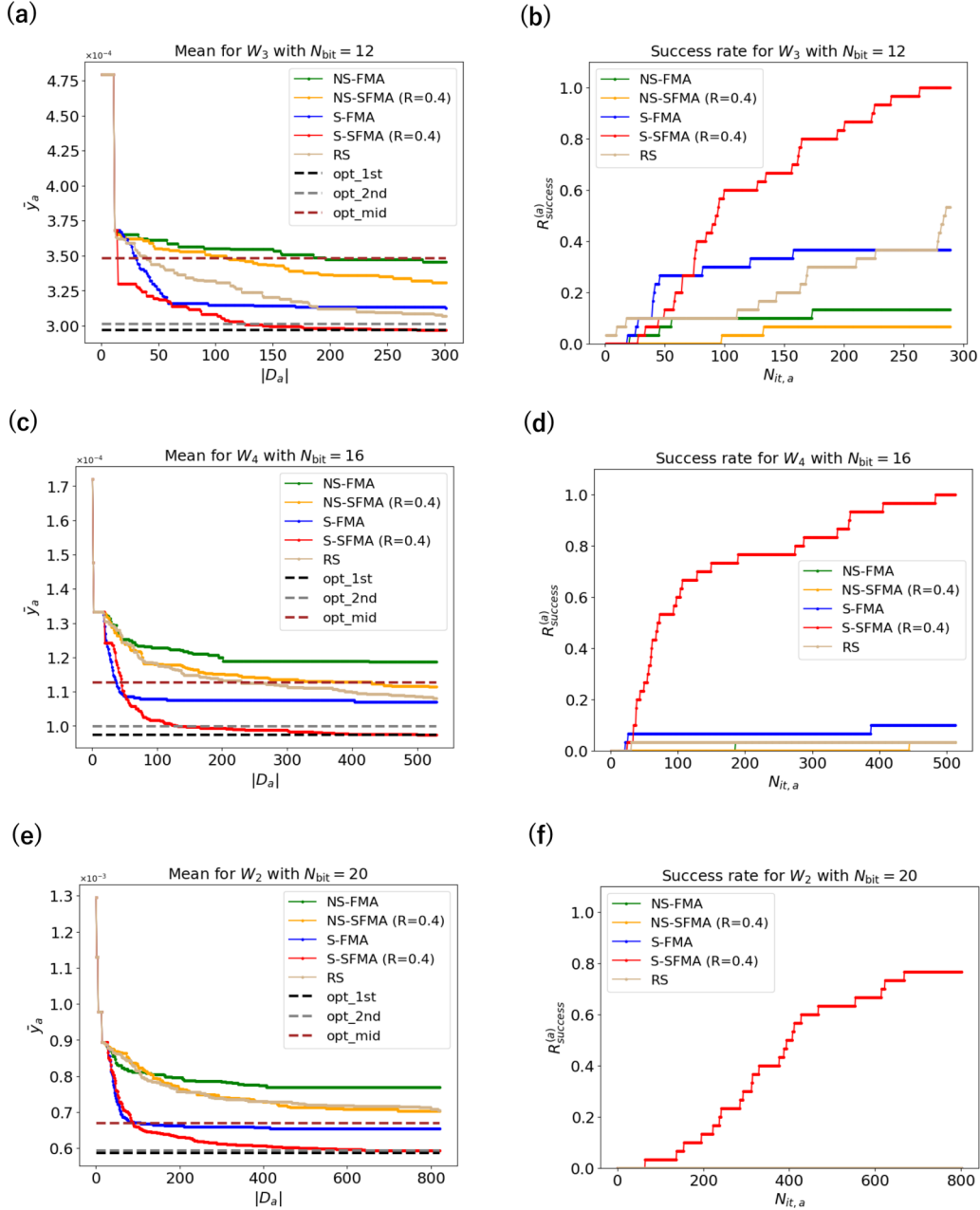


FIG. 3. Results of $\bar{y}_{\min}^{(a)}$ [panels (a), (c), and (e)] and $R_{\min, \text{success}}^{(a)}$ [panels (b), (d), and (f)]. Plots in panels (a) and (b), (c) and (d), and (e) and (f) display the results for W_3 with $N_{\text{bit}} = 12$, W_4 with $N_{\text{bit}} = 16$, and W_2 with $N_{\text{bit}} = 20$, respectively. The iteration number N_{ite} is taken to be $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$. The red (orange) and blue (green) curves represent the results of SFMA and FMA with (without) standardization, respectively. The results of RS are plotted by the tan curves.

(FMA). By utilizing $R_{\min, \text{success}}^{(a)}$, we introduce metrics for the exploration performance and the exploitation performance of the algorithms denoted by N_{conv} and $R_{\text{success}}^{\text{final}} = R_{\text{success}}^{(a=N_{\text{ite}})}$, respectively. The former represents the minimum number of $N_{\text{it}, a}$ such that $R_{\min, \text{success}}^{(a)} \geq 0.5$, whereas the latter describes the final value of $R_{\min, \text{success}}^{(a)}$. In our numerical experiments, all the above statistical quantities are computed by taking $N_{\text{samp}} = 30$. In addition, all these quantities are calculated by rounding them to 17 decimal places in order to avoid floating-point errors.

C. Results

In this section, we discuss our numerical experimental results given in Figs. 3–5 and Tables I and II. In these figures and tables, we denote the terms nonstandardized and standardized by using acronyms NS and S, respectively; for instance, we call nonstandardized FMA NS-FMA, while we call standardized SFMA S-SFMA. In addition to these four algorithms, we present the results of random search (RS) and use them as indicators to show how well SFMA as well as FMA converge to the optimal solution. For each N_{bit} , RS is performed by

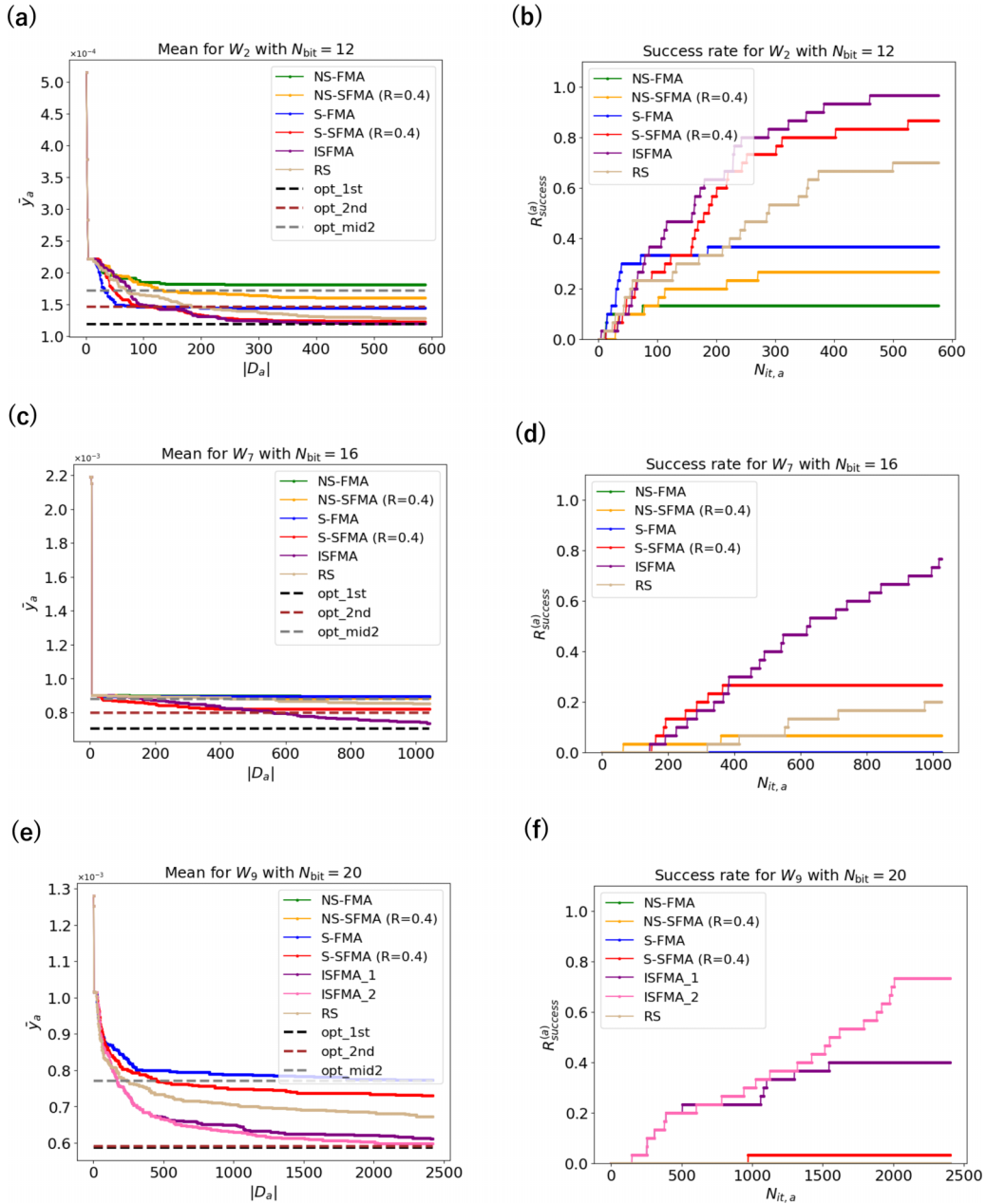


FIG. 4. Results of $\bar{y}_{\min}^{(a)}$ [panels (a), (c), and (e)] and $R_{\min, \text{success}}^{(a)}$ [panels (b), (d), and (f)] of the improved SFMA. Plots in panels (a) and (b), (c) and (d), and (e) and (f) are the results for W_2 with $N_{\text{bit}} = 12$ and $N_{\text{ite}} = 4N_{\text{bit}}^2 + 1$, W_7 with $N_{\text{bit}} = 16$ and $N_{\text{ite}} = 4N_{\text{bit}}^2 + 1$, and W_9 with $N_{\text{bit}} = 20$ and $N_{\text{ite}} = 6N_{\text{bit}}^2 + 1$, respectively. For $N_{\text{bit}} = 12$ and 16, the results of the improved SFMA (ISFMA) are presented by purple curves. On the other hand, for $N_{\text{bit}} = 20$, the two improved algorithms ISFMA₁ and ISFMA₂ are displayed by purple and pink curves, respectively. To make the comparisons between the performance of the improved SFMA and those of the other algorithms, we also plot the results of the nonstandardized FMA (green), the standardized FMA (blue), the nonstandardized SFMA (orange), the standardized SFMA (red), and RS (tan).

augmenting the initial dataset $|D_0|$ with N_{ite} data points, which are selected randomly according to N_{samp} different random seeds. In Figs. 3–5, we plot the results for $\bar{y}_{\min}^{(a)}$ and $R_{\min, \text{success}}^{(a)}$ as functions of the dataset size $|D_a|$ and the iteration number of BBO loops $N_{it, a}$, respectively: The plots of $\bar{y}_{\min}^{(a)}$ ($R_{\min, \text{success}}^{(a)}$) are labeled by a, c, and e (b, d, and f). For the plots of $\bar{y}_{\min}^{(a)}$, we also present three horizontal dashed lines with three colors, black, brown, and gray, in order to present how well SFMA and FMA are approaching the optimal solutions. The black, brown, and gray lines represent the output value of the first

optimal solution $y_{\text{opt}, 1\text{st}}$, that of the second optimal solution $y_{\text{opt}, 2\text{nd}}$, and the approximate optimal values of $f_{\text{BB}, W}(M)$ in Eq. (2) (y_{mid}), which is obtained by an algorithm shown in Ref. [86] (see Eqs. (1)–(5) in Ref. [86]), respectively. The two values $y_{\text{opt}, 1\text{st}}$ and $y_{\text{opt}, 2\text{nd}}$ are calculated by exhaustive search (brute force) methods. On the other hand, in Tables I and II, we present the tabular data on N_{conv} and $R_{\text{success}}^{\text{final}}$. The last columns denoted by the term frequency indicate how many times each algorithm demonstrated the best performance in convergence (the smallest value of N_{conv}) and accuracy (the

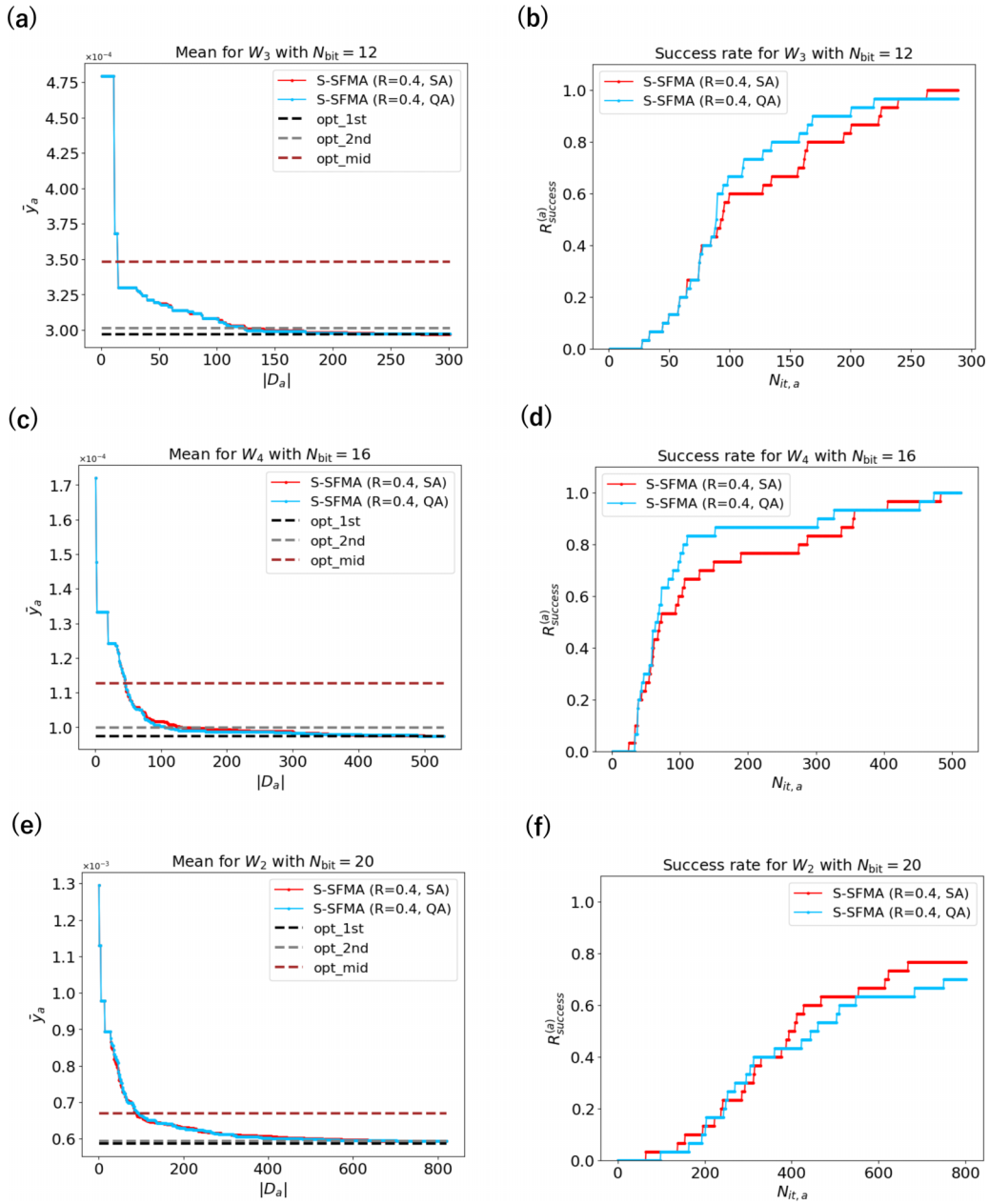


FIG. 5. Results of $\bar{y}_{\min}^{(a)}$ [panels (a), (c), and (e)] and $R_{\min, \text{success}}^{(a)}$ [panels (b), (d), and (f)] of the standardized SFMA with $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$. Plots in panels (a) and (b), (c) and (d), and (e) and (f) are the results for W_3 with $N_{\text{bit}} = 12$, W_4 with $N_{\text{bit}} = 16$, and W_2 with $N_{\text{bit}} = 20$, respectively. The red and light blue curves represent the results for the annealers selected to be SA and QA, respectively.

largest value of $R_{\text{success}}^{\text{final}}$ across the ten instances. To highlight which algorithm exhibits the best performance in convergence and accuracy, for each W_n we write the smallest value of

N_{conv} , the largest value of $R_{\text{success}}^{\text{final}}$, and the largest frequency by the bold texts. When $n_{\text{alg}} (= 1, \dots, 4)$ algorithms and/or RS become tied (n_{alg} algorithms and/or RS exhibit the same

TABLE I. Data on N_{conv} for $N_{\text{bit}} = 20$ and $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$. All the annealing processes have been performed by SA.

Instance Algorithm	W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	Frequency
S-SFMA	396	None	393	732	None	None	274	380	321	None	6
S-FMA	None	None	None	None	None	None	None	None	None	None	0
NS-SFMA	None	None	None	None	None	None	None	None	None	None	0
NS-FMA	None	None	None	None	None	None	None	None	None	None	0
RS	None	None	None	None	None	None	None	None	None	None	0

TABLE II. Data on $R_{\text{success}}^{\text{final}}$ for $N_{\text{bit}} = 20$ and $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$. All the annealing processes have been performed by SA.

Algorithm \ Instance	W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	Frequency
S-SFMA	24/30	4/30	23/30	15/30	7/30	4/30	25/30	17/30	24/30	1/30	10
S-FMA	5/30	0/30	0/30	0/30	0/30	2/30	1/30	3/30	0/30	0/30	0
NS-SFMA	0/30	0/30	0/30	0/30	0/30	0/30	0/30	0/30	0/30	0/30	0
NS-FMA	0/30	0/30	0/30	0/30	0/30	0/30	0/30	0/30	0/30	0/30	0
RS	0/30	0/30	0/30	1/30	1/30	0/30	0/30	0/30	0/30	0/30	0

smallest values of N_{conv} or the highest values of $R_{\text{success}}^{\text{final}}$, we assign a value $1/n_{\text{alg}}$ [or $1/(n_{\text{alg}} + 1)$] to the frequency of each algorithm except for the cases when the values of N_{conv} do not exist as indicated by the term none and when $R_{\text{success}}^{\text{final}} = 0$. In addition to Figs. 3–5 and Tables I and II, in Appendix C we demonstrate the analyses of variances and standard deviations of $y_{\text{min}}^{(a)}$. In the following, we discuss the results by presenting a section for each figure.

1. Results for $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$ via SA

In Fig. 3, we show the numerical results for W_3 with $N_{\text{bit}} = 12$ [panels (a) and (b)], W_4 with $N_{\text{bit}} = 16$ [panels (c) and (d)], and W_2 with $N_{\text{bit}} = 20$ [panels (e) and (f)]. All these results are obtained by choosing an annealer to be SA with $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$. We explain the setup for the execution of SA in Appendix B 2. We present these results since they clearly exhibit the theoretically expected behaviors across the ten W matrices. The results for the rest of the W matrices are shown in Appendix D. Let us explain from the result for the means $\bar{y}^{(a)}$ [Figs. 3(a), 3(c), and 3(e)]. We observe in the figure that the means calculated by the standardization approach the first optimal solution faster than those calculated without the standardization, which implies the effectiveness of the standardization. Such a feature implies the verification of our consideration on the properness of the standardization. For the rest of the discussion, let us focus on the results of standardized SFMA and FMA and analyze their behaviors. Basically, we observe the same characteristics across all N_{bit} . That is, in the first half of the BBO processes the red curves (S-SFMA) take the equivalent or larger values than those of the blue curves (S-FMA) in certain ranges. On the other hand, in the second half S-SFMA exhibits the lower values at all $|\mathcal{D}_a|$ and approaches the first optimal solution with significantly higher accuracy. Correspondingly, in Figs. 3(b), 3(d), and 3(f) we observe that in the first half the values of $R_{\text{min,success}}^{(a)}$ calculated by S-SFMA take the equivalent or smaller values in certain ranges whereas in the second half they are larger at all $N_{\text{ite},a}$. We interpret such contrastive characteristics of the SFMA and FMA in the following way. In the first half, SFMA exhibits the larger deviation in searching the optimal solution since it is in the exploration phase with higher performance than the exploration performance of FMA. In the second half, the candidate solutions are explored sufficiently by SFMA and it exhibits higher exploitation performance. As a result, SFMA finds the optimal solution more efficiently and effectively than FMA. This consideration is supported by Tables I and II, which present the data on N_{conv} and $R_{\text{success}}^{\text{final}}$ for $N_{\text{bit}} = 20$,

respectively. From these tables, we clearly observe that SFMA shows the advantages over the other algorithms in both speed (convergence) and accuracy.

2. Results for $N_{\text{ite}} = 4N_{\text{bit}}^2 + 1, 6N_{\text{bit}}^2 + 1$ via SA

Next, we discuss the results in Fig. 4. As indicated by Fig. 3, Figs. 8–13 in Appendix D, and Tables I and II, the ways the algorithms converge to the optimal solutions do not show the similar behaviors with respect to W_i . Specifically, S-SFMA converge to the optimal solutions with high accuracy (large $R_{\text{success}}^{\text{final}}$) for certain cases including the ones shown in Fig. 3, whereas it does not (small $R_{\text{success}}^{\text{final}}$) for some cases. Furthermore, the values of N_{conv} of S-SFMA does not exist for some instances. The convergence performance as well as the accuracy of SFMA can be improved by, for instance, increasing the value of N_{ite} and/or taking the value of R to be smaller than 0.4. By controlling the values of N_{ite} and R coherently, SFMA can be improved in various ways and we investigate them in the following. In Fig. 4, we present the results for W_2, W_7 , and W_9 with $N_{\text{bit}} = 12, 16$, and 20 , respectively: the results for other W matrices are discussed in Appendix E. For $N_{\text{bit}} = 12$ and 16 , the parameters N_{ite} and R are taken to be $N_{\text{ite}} = 4N_{\text{bit}}^2 + 1, R = 0.1$. We call these algorithms ISFMA. For $N_{\text{bit}} = 20$, we make two types of improvements. The first one is done by taking $N_{\text{ite}} = 6N_{\text{bit}}^2 + 1, R = 0.1$ and the second one is done by sequentially running two different types of SFMA. In the first part, we perform SFMA with $N_{\text{ite}} = N_{\text{bit}}^2$ and $R = 0.1$, whereas in the second part we run it by taking $N_{\text{ite}} = 5N_{\text{bit}}^2$ and $R = 0.01$. For convenience, we call the former algorithm ISFMA₁ whereas the second one ISFMA₂. We benchmark them against the other four different algorithms (standardized and nonstandardized FMA and SFMA with $R = 0.4$) and RS in convergence and accuracy. These algorithms are performed by augmenting the datasets whose sizes are equal to $2N_{\text{bit}}^2 + 1$ (the results shown in Figs. 8–13) with additional datasets obtained by $2N_{\text{bit}}^2$ ($4N_{\text{bit}}^2$) times of BBO loops for $N_{\text{bit}} = 12$ and 16 ($N_{\text{bit}} = 20$). Let us explain from the cases of $N_{\text{bit}} = 12$ [panels (a) and (b)] and 16 [panels (c) and (d)]. For both cases, we observe that ISFMA (purple curve) exhibits the smallest values of N_{conv} and the highest values of $R_{\text{success}}^{\text{final}}$.

Finally, we discuss the result for $N_{\text{bit}} = 20$ [panels (e) and (f)]. What we observe from the figures is that ISFMA₁ (purple curve) and ISFMA₂ (pink curve) approach the optimal solution faster than any other four algorithms and RS. In the following, let us focus on the results of ISFMA₁ and ISFMA₂ and analyze their characteristics more in detail. Note that both

of these algorithms are performed such that the second parts of BBO calculations are carried out using the same dataset obtained by running SFMA with $N_{\text{ite}} = N_{\text{bit}}^2$ and $R = 0.1$ (the first part of BBO calculation). The results demonstrate that in the first half the mean values $\bar{y}^{(a)}$ of ISFMA₁ are smaller than those of ISFMA₂ in certain ranges. In the second half, the mean values $\bar{y}^{(a)}$ of ISFMA₂ are always smaller than those of ISFMA₁. Correspondingly, in the first half, the success rates $R_{\text{min,success}}^{(a)}$ of ISFMA₂ are equal to or smaller in certain ranges, whereas in the second half the values of $R_{\text{min,success}}^{(a)}$ of ISFMA₂ become equivalent or larger at all $N_{\text{ite},a}$. The final value of the success rate $R_{\text{success}}^{\text{final}}$ of ISFMA₂ is equal to 22/30 whereas that of ISFMA₁ is 12/30, which implies that the accuracy of ISFMA₂ is about twice as large as that of ISFMA₁. We consider that such a significant difference between these two algorithms has been generated due to how the FM models have been trained in the second part. That is, in ISFMA₂ the FM models have been trained by taking $R = 0.01$, which is ten times smaller than the value of R taken in ISFMA₁. We consider that such a way of training has enabled ISFMA₂ to explore a significantly wider range of the solution space, which leads the enhancement of the exploration performance of ISFMA₂ and the exploitation performance as well. Moreover, ISFMA₂ exhibits the value of N_{conv} while ISFMA₁ does not. As a result, both the performance of exploration and exploitation of ISFMA₂ are significantly higher than the other algorithms. We regard this result as the reflection of the potential scalability of SFMA discussed in Sec. III B.

3. Results for $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$ via QA

Finally, let us discuss the results in Fig. 5. These are the results for $\bar{y}^{(a)}$ and $R_{\text{min,success}}^{(a)}$ computed by running S-SFMA via both SA (red) and QA (light blue). Note that the red curves are the same ones presented in Fig. 3. The numerical setup for running QA is explained in Appendix B 3. We observe in Fig. 5 that the results obtained by the two annealers quantitatively demonstrate almost the same behaviors. In our experiments, we have not observed the explicit outperformance of QA over SA in speed and/or accuracy (quantum advantage). Meanwhile, for instance, in Refs. [23,24], it has been highlighted that for certain optimization problems QA might be more advantageous than SA. In addition, it has been studied in Ref. [110] that a quantum computer exhibits a superpolynomial quantum advantage over a classical computer in finding an approximate solution to combinatorial optimization problems. When N_{bit} is sufficiently larger than the values taken in our experiments, the complexity of a Hilbert space, which is a resource of quantum computing, becomes more enhanced, and we expect that there is a possibility that we observe the supremacy of QA such that it demonstrates advantages in speed and/or accuracy over SA. We would like to address such an issue in our future work.

V. CONCLUSION AND OUTLOOK

In this work, we have developed the algorithm that enables us to solve black-box optimization problems effectively and

efficiently by devising the training processes: SFMA. FMA is one of the methodologies for solving black-box optimization problems; however, it has a potential limitation such that its performance on exploring candidate solutions may be insufficient since FM models are trained by a point-estimation approach. From this perspective, we have improved FMA by strengthening its exploration performance. Specifically, we have trained FM models by sampled subdatasets. Since FM models are trained in such probabilistic ways, we become able to explore broader ranges of solution spaces, which implies the amplification of the exploration performance of FMA. As a result, SFMA exhibits the exploration-exploitation functionality. We have conducted the numerical experiments to verify whether SFMA truly exhibits the exploration-exploitation functionality or not and to benchmark it against FMA in speed (convergence) and accuracy. Consequently, we have verified that SFMA certainly exhibits the exploration-exploitation functionality and is superior to FMA in both speed and accuracy. Furthermore, we have observed the potential scalability of SFMA by setting the value of R to be correspondingly small according to the problem size. We expect that the advantages of SFMA, low computational cost and scalability, are going to be powerful tools for tackling complex problems in the real world.

Let us further highlight the superiority of SFMA with respect to the computational cost. In this work, we have amplified the exploration performance by subsampling. Meanwhile, we are able to strengthen the exploration performance in other ways. For instance, we can accomplish this by running FMA and BOCS alternatively. Specifically, first we run BOCS and later we execute FMA. Moreover, BOCS can be replaced with another stochastic algorithm. Although we can take such alternative approaches, we consider that using the subsampling method is more beneficial. This is because the computational cost for running stochastic algorithms like BOCS typically increases significantly as the dataset size and/or the dimension of a solution space become larger. On the other hand, SFMA can be performed with low computational cost even for a large dataset by setting the value of R to be correspondingly small, although its exploration performance might be weaker than other stochastic algorithms. Another approach to accomplish establishing the algorithms possessing the exploration-exploitation functionalities is, for instance, to enhance the exploitation performance of stochastic algorithms. We would like to address this problem in our future work by examining various stochastic algorithms. Finally, we would like to point out that the subsampling approach works for training other types of machine learning models. With this taken into account, the investigation of the machine learning model trained by a sampled subdataset that demonstrates the best performance on a given problem would also be an interesting future direction. In relation to this, the investigation of other sampling methods would be another important theme to explore. These could be, for instance, the creation of subdatasets using clustering methods and that comprised of small and large values of objective functions. Furthermore, the optimization of the hyperparameter R would be another interesting direction.

In this work, we have developed the effective algorithm for black-box optimization problems by devising dataset modeling, i.e., step (ii) in Fig. 1. By developing powerful method for each of the other two basic steps and incorporating them into SFMA coherently, we expect that we become able to run SFMA more effectively and efficiently. Furthermore, the development of powerful algorithms for BBO that are carried out by gate-based quantum optimization algorithms [27,29,31,33,34] would be another interesting and important direction.

Recently, intensive research and development in both machine learning (artificial intelligence) and quantum technology including both hardware engineering and algorithmic development has been progressing. It is expected to continue toward the development of algorithms that are capable of addressing complex problems in the real world and large-scale (both classical and quantum) devices that enable us to leverage such algorithms. We expect that our results pave the way for accomplishing these goals and lead to the creation of next-generation technologies that are going to be the basic building blocks for accelerating industrial developments.

ACKNOWLEDGMENTS

This work was done for Council for Science, Technology and Innovation (CSTI), Cross-ministerial Strategic Innovation Promotion Program (SIP), “Promoting Application of Advanced Quantum Technologies to Social Challenges” (Project management agency: QST) under Grant No. JPJ012367.

DATA AVAILABILITY

The data files presented in this paper can be obtained at Zenodo [111].

APPENDIX A: STANDARDIZATION METHOD

We explain how to construct the normalized sampled subdatasets $\hat{\mathcal{B}}_a$ in Algorithm 1. The subdatasets $\hat{\mathcal{B}}_a$ are formulated by the standardized output variables defined by [4,54,55,57,86]

$$\hat{y}_{\text{stand}}^{(i)} = \frac{y^{(i)} - \bar{y}}{\sqrt{\Delta^2 y} \times N_{\text{bit}}}. \quad (\text{A1})$$

The quantities $\bar{y}$ and $\Delta^2 y$ in the above equation are defined by

$$\bar{y} = \frac{1}{N_{\mathcal{G}}} \sum_{y \in \mathcal{G}} y, \quad \Delta^2 y = \frac{1}{N_{\mathcal{G}}} \sum_{y \in \mathcal{G}} (y - \bar{y})^2, \quad (\text{A2})$$

where $\mathcal{G}$ is an ensemble of pairs of input and output variables constructed randomly from a full dataset and $N_{\mathcal{G}}$ is the number of such pairs included in $\mathcal{G}$. In our numerical experiments, we take $N_{\mathcal{G}} = 5N_{\text{bit}}$. The reason why we use the standardized output variables $\hat{y}_{\text{stand}}^{(i)}$ in Eq. (A1) instead of the bare output variables $y^{(i)}$ to train FM models is because when the values of $y^{(i)}$ are considerably small, there are possibilities that annealing processes do not work effectively. Let us discuss more in detail for both SA and QA. In case of SA, when output values $y^{(i)}$ are too small this effectively corresponds to taking high inverse temperatures, which means that the annealing processes tend to show random behaviors. Such randomness destabilizes the convergence to the optimal solutions, which implies that the annealing processes do not work effectively. We consider that such destabilization of convergence is prominent for large N_{bit} since the dimension of a solution space is significantly big. Next, let us discuss the effectiveness of the standardization on QA. We consider that the standardization is also effective on QA because of the following reason. When bare output variables are substantially small, then the energy gaps between the ground state energies and the first excited energies are considerably small as well. In such cases, the performance of QA gets degraded significantly. By performing the standardization, the magnitudes of such energy gaps are effectively enhanced, and correspondingly, the performance of QA is amplified. Therefore, the standardization is also effective for performing QA with high accuracy.

APPENDIX B: IMPLEMENTATION OF SFMA

In this section, we describe the implementation of SFMA in our numerical experiments. First, we explain the numerical procedure of SFMA, which is represented schematically in Fig. 6. After that, we explain the setups for running SA and QA. All the classical computations in our numerical experiments are performed on a MacBook Air (Sonoma

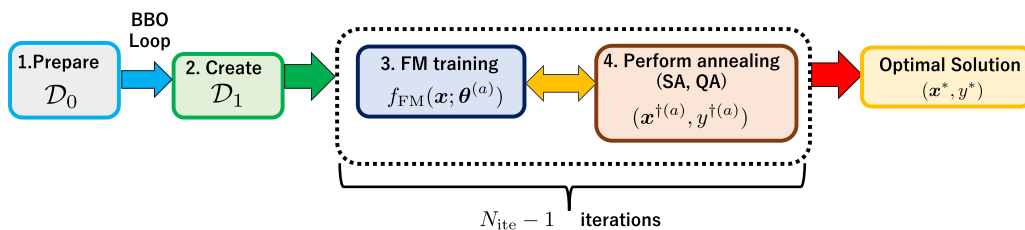


FIG. 6. Numerical procedure of SFMA comprised of four basic steps. First, we prepare an initial dataset $\mathcal{D}_0$. Second, we create $\mathcal{D}_1$. Third, we train an FM model $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta}^{(a)})$. Fourth, we run annealing (SA or QA) to search for the optimal solution to $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta}^{(a)})$ denoted by $(\mathbf{x}^{\dagger(a)}, \mathbf{y}^{\dagger(a)})$. Steps three and four are performed iteratively $N_{\text{ite}} - 1$ times.

14.2) that consists of Apple M2 (8-core CPU) and 24 GB of memory.

1. Numerical procedure

The numerical procedure of SFMA is comprised of four basic steps. First, we prepare an initial dataset $\mathcal{D}_0$ with $|\mathcal{D}_0| = N_{\text{bit}}$ for each problem instance generated by randomly selecting N_{bit} sets of input and output variables using a fixed random seed. Second, for each problem instance we create a full dataset $\mathcal{D}_1$ by running a single BBO loop. We perform this in two ways, the training of FM models with and without standardization. The reason we start running SFMA not with $\mathcal{D}_0$ but with $\mathcal{D}_1$ is because we conduct the benchmarking tests in a way that we compare the performance of SFMA and that of FMA after the initial dataset $\mathcal{D}_0$, which is the common given dataset, is used for both cases. In the rest, SFMA is performed $N_{\text{ite}} - 1$ times. Since each iteration is performed by the same way in terms of the basic three steps in Fig. 1, we describe the following description as the explanation of the procedure of running the a th ($a = 2, \dots, N_{\text{ite}}$) BBO loop. Third, we train the FM models $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta}^{(a)})$ with the datasets $\mathcal{D}_{a-1}$. We do this by optimizing MSE using Adam with all the samples of the datasets (full batch). For every calculation, the learning rate η and the epoch number N_{epoch} are taken to be $\eta = 0.01$ and $N_{\text{epoch}} = 200$, respectively. We note that for each iteration, each component of the FM parameters $\boldsymbol{\theta}^{(a)}$ is trained by initially generating a random number obeying a normal distribution with its mean and variance equal to 0 and $\Delta^2 \bar{y}$ given in Eq. (A2), respectively. We take the variance of the normal distribution to be $\Delta^2 \bar{y}$ since the FM model parameters $\boldsymbol{\theta}^{(a)}$ have the same orders with the black-box functions. In other words, we also rescale $\boldsymbol{\theta}^{(a)}$ as we do in the standardization: In contrast, in cases of the nonstandardized SFMA and FMA we do not perform rescaling. Such a normal distribution is constructed by a fixed random seed. All these machine learning calculations are carried out using PyTorch version 2.4.1 and Python version 3.12.7. Fourth, we run annealing processes (SA or QA) using D-Wave Ocean SDK version 8.1.0.

2. Setup for SA

Let us describe the setup for SA. The SA calculations are performed using the library called `neal`. `SimulatedAnnealingSampler`, which is supported by D-Wave Ocean SDK [112]. We set two parameters, `num_reads` and `num_sweeps`, and a fixed random seed. For convenience, hereinafter let us denote `num_reads` and `num_sweeps` by n_{reads} and n_{sweeps} , respectively. The parameter n_{reads} represents the repetition number of a definite single annealing procedure and n_{sweeps} describes the number of Markov chain Monte Carlo steps. In our experiments, we take $(n_{\text{reads}}, n_{\text{sweeps}}) = (10, 100)$ and the rest of the parameters are set to their default values. We note that we have verified that the annealing processes under these values have enabled the acquisitions of the optimal solutions to $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta}^{(0)})$ (FM function obtained by $\mathcal{D}_0$) trained in both cases, with and without the standardization procedure. Thus, the SA processes work properly for both cases under these values of n_{reads} and n_{sweeps} .

3. Setup for QA

Let us explain the setup for running QA. The quantum annealing calculations have been performed by the quantum device called `Advantage_system 6.4`, which is supported by D-Wave Systems, Inc. [113]. In order to implement the FM models on it, the logical qubits are mapped onto the physical qubits, which is so-called minor embedding. All the minor-embedding processes are carried out by utilizing the function called `EmbeddingComposite`, which is provided in D-Wave Ocean SDK [112]. In addition, as similar to the cases of SA, a definite QA process is run multiple times independently, which is characterized by a parameter `num_reads`: In the following, we denote it by n_{reads} . For every experiment, we take $n_{\text{reads}} = 50$ and the rest of the parameters are set to their default values. Furthermore, we have numerically verified that QA properly finds the optimal solutions to $f_{\text{FM}}(\mathbf{x}; \boldsymbol{\theta}^{(0)})$ under this parameter setting.

APPENDIX C: ANALYSES OF STANDARD DEVIATIONS

In Sec. IV C, we have discussed the results for the means $\bar{y}^{(a)}$ and the success rates $R_{\text{min,success}}^{(a)}$. In addition, let us analyze our results of variances defined by

$$\sigma^2 = \frac{1}{N_{\text{samp}}} \sum_{\alpha=1}^{N_{\text{samp}}} (y_{\text{min},\alpha}^{(a)} - \bar{y}_{\text{min}}^{(a)})^2. \quad (\text{C1})$$

In the following, we investigate the behaviors of the associated standard deviations $\sqrt{\sigma^2}$ given by Eq. (C1) by comparing to those of the success rates $R_{\text{min,success}}^{(a)}$. In Fig. 7, we display the results of S-SFMA and S-FMA by red and blue curves, respectively. The plots in panels (a) and (b), (c) and (d), and (e) and (f) denote the results for W_3 with $N_{\text{bit}} = 12$, W_4 with $N_{\text{bit}} = 16$, and W_2 with $N_{\text{bit}} = 20$, respectively. For every calculation, we take the BBO iteration number to be $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$. In panels (a), (c), and (e), we display the plots of the means $\bar{y}_{\text{min}}^{(a)}$ with shaded regions representing 95% confidence intervals defined by $[\bar{y}^{(a)} - 1.96\sqrt{\sigma^2/N_{\text{samp}}}, \bar{y}^{(a)} + 1.96\sqrt{\sigma^2/N_{\text{samp}}}]$. On the other hand, we present the results for $R_{\text{min,success}}^{(a)}$ in panels (b), (d), and (f). Note that $\bar{y}_{\text{min}}^{(a)}$ and $R_{\text{min,success}}^{(a)}$ shown here are the same ones in Fig. 3. From the results in panels (a), (c), and (e), we observe that, for all N_{bit} , when the dataset sizes $|\mathcal{D}_a|$ increase and get closer to N_{tot} , the standard deviations become sufficiently small or approach zero for the cases of S-SFMA. In contrast, the standard deviations remain finite for the cases of S-FMA. Correspondingly, the results in panels (b), (d), and (f) demonstrate that for the cases of S-SFMA, the values of $R_{\text{min,success}}^{(a)}$ increase as $N_{\text{it},a}$ get larger and become equal to or almost equal to 30/30 at $N_{\text{it},a} = N_{\text{ite}}$. For the cases of S-FMA, the values of $R_{\text{min,success}}^{(a)}$ get larger as $N_{\text{it},a}$ increase; however, the final values $R_{\text{min,success}}^{\text{final}}$ are equal to or smaller than 11/30. As a result, the standard deviations are zero (sufficiently small) when we achieve the optimal solutions in all (almost all) N_{samp} ($=30$) trials whereas they take moderate values when the optimal solutions have been obtained in the portions of the N_{samp} trials as expected.

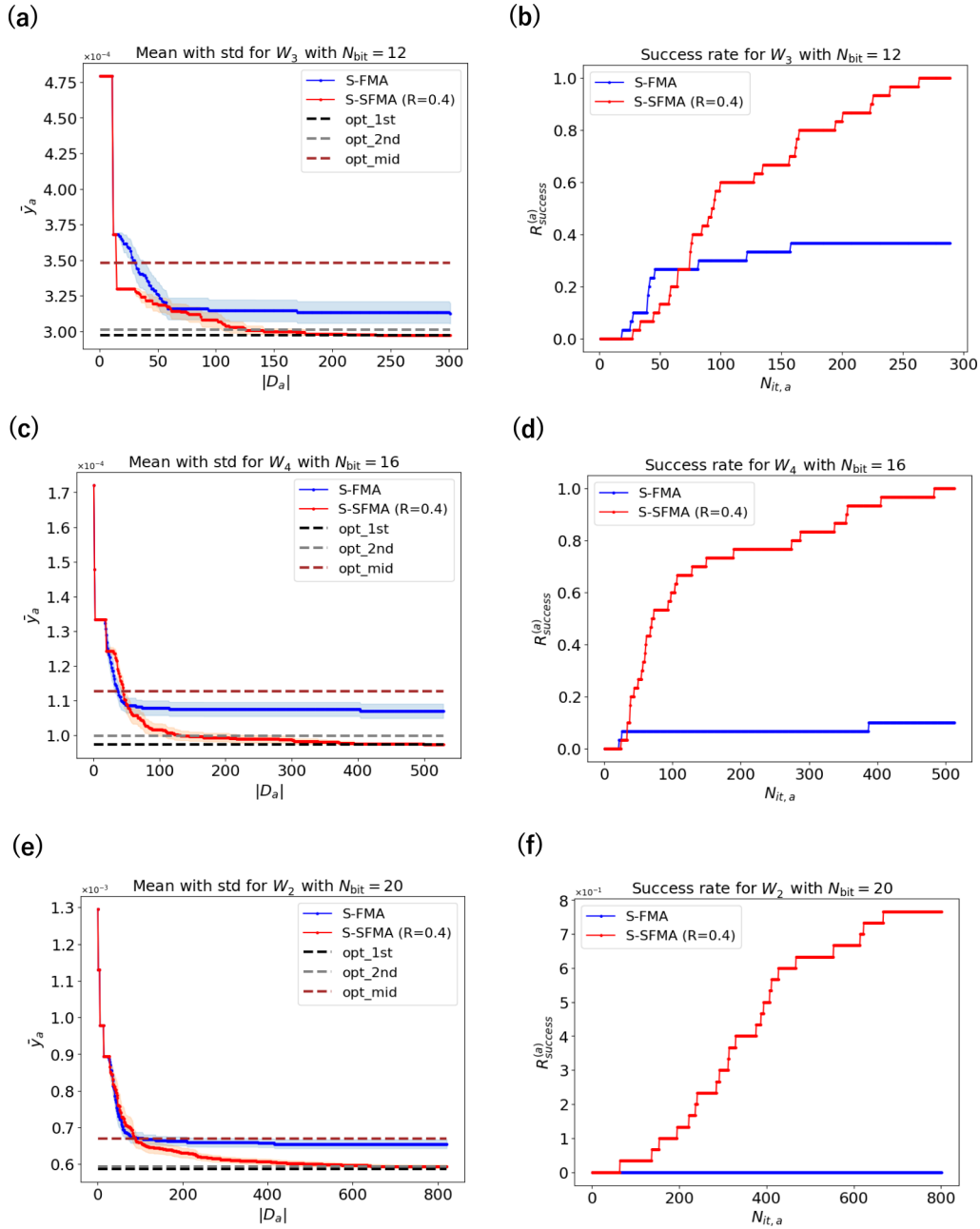


FIG. 7. Results of $\bar{y}^{(a)}$ associated with standard deviations $\sqrt{\sigma^2}$ [panels (a), (c), and (e)] and $R_{\text{min,success}}^{(a)}$ [panels (b), (d), and (f)] calculated by S-SFMA (red) and S-FMA (blue). The plots in panels (a) and (b), (c) and (d), and (e) and (f) are the results for W_3 with $N_{\text{bit}} = 12$, W_4 with $N_{\text{bit}} = 16$, and W_2 with $N_{\text{bit}} = 20$, respectively. N_{ite} is taken to be $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$ for every case.

APPENDIX D: RESULTS FOR THE OTHER W MATRICES

In this Appendix, we explain the results for $\bar{y}^{(a)}$ and $R_{\text{min,success}}^{(a)}$ for the other nine W matrices that have not been presented in Sec. IV C. The plots in Figs. 8, 10, and 12 (Figs. 9, 11, and 13) are the results for $\bar{y}^{(a)}$ ($R_{\text{min,success}}^{(a)}$) with $N_{\text{bit}} = 12, 16$, and 20 and $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$, respectively. According to the results presented in Figs. 8–13 and those in Sec. IV C, overall SFMA outperforms the other algorithms

in convergence (SFMA demonstrates the smallest values of N_{conv}) across the ten problem instances. Moreover, SFMA demonstrates the advantages in accuracy (SFMA exhibits the largest values of $R_{\text{success}}^{\text{final}}$) in ten instances for every N_{bit} . From these results, we consider that the advantage of SFMA becomes more prominent as N_{bit} increases, particularly, in accuracy. Furthermore, RS becomes less effective when N_{bit} increases.

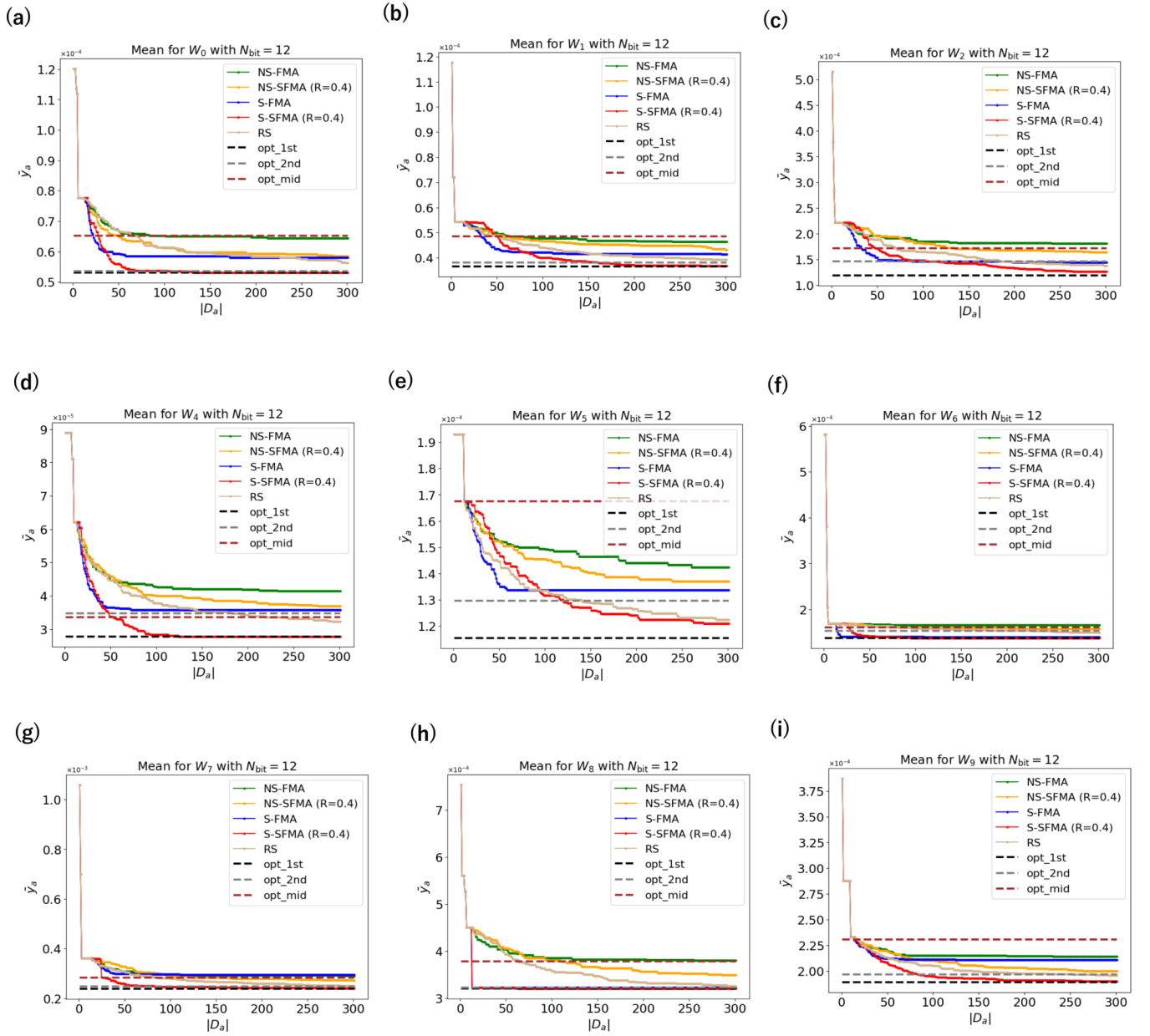


FIG. 8. Results for $\bar{y}_a^{(a)}$ with $N_{\text{bit}} = 12$ and $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$. Plots in panels (a)–(i) are the ones for W_n with $n = 0, 1, 2, 4, 5, 6, 7, 8$, and 9 , respectively.

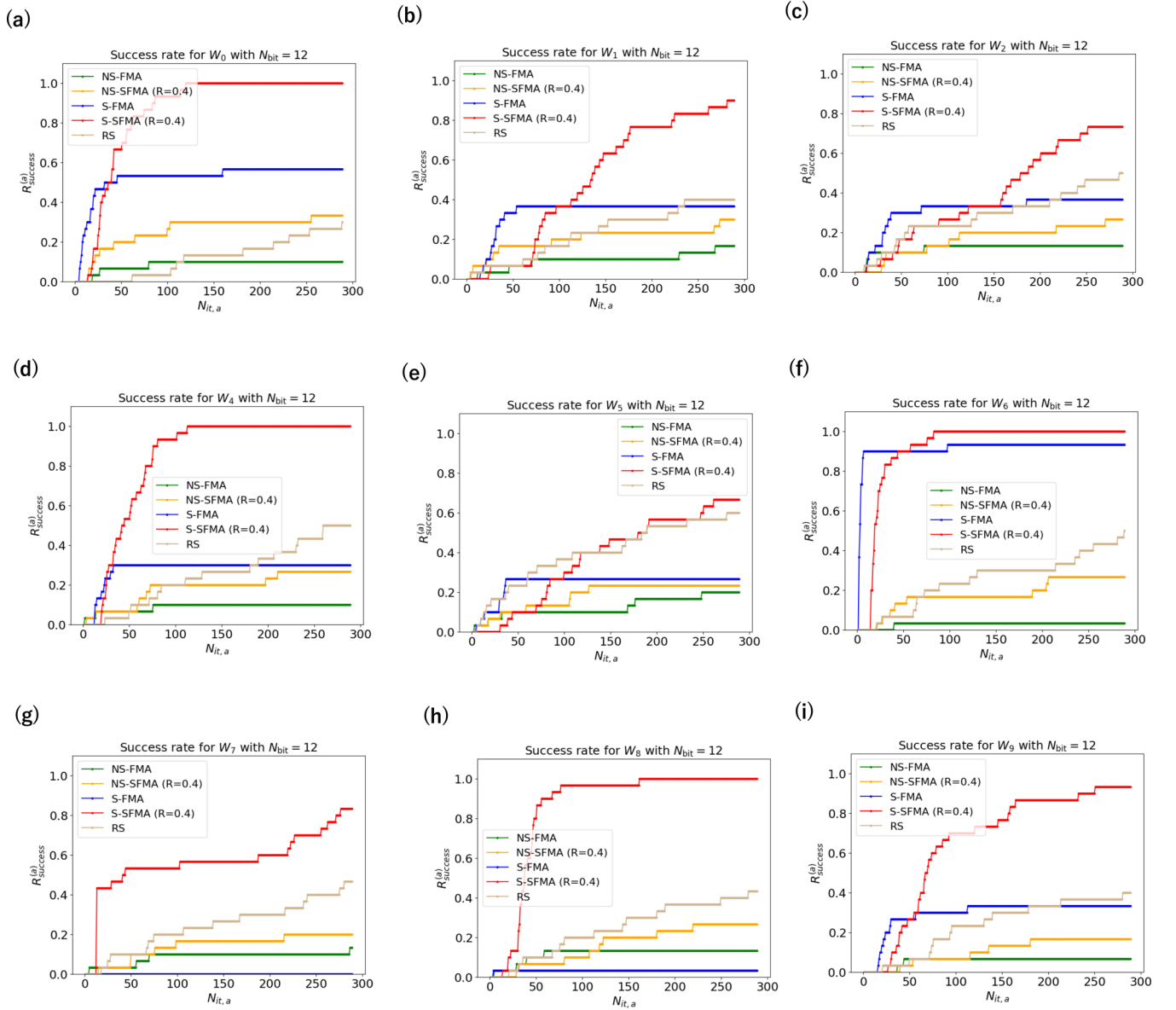


FIG. 9. Results for $R_{\min, \text{success}}^{(a)}$ with $N_{\text{bit}} = 12$ and $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$. Plots in panels (a)–(i) are the ones for W_n with $n = 0, 1, 2, 4, 5, 6, 7, 8$, and 9, respectively.

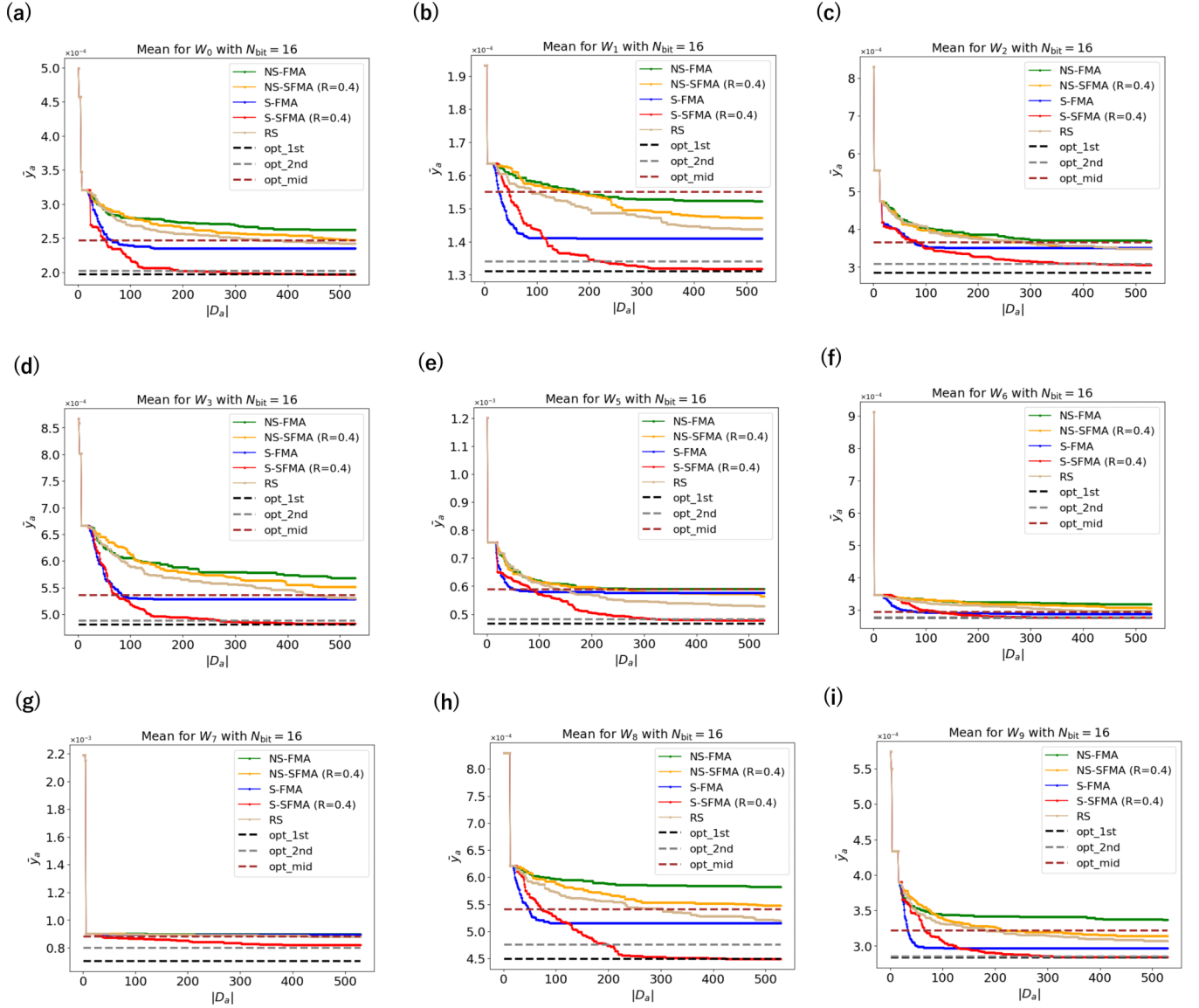


FIG. 10. Results for $\bar{y}^{(a)}$ with $N_{\text{bit}} = 16$ and $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$. Plots in panels (a)–(i) are the ones for W_n with $n = 0, 1, 2, 3, 5, 6, 7, 8$, and 9, respectively.

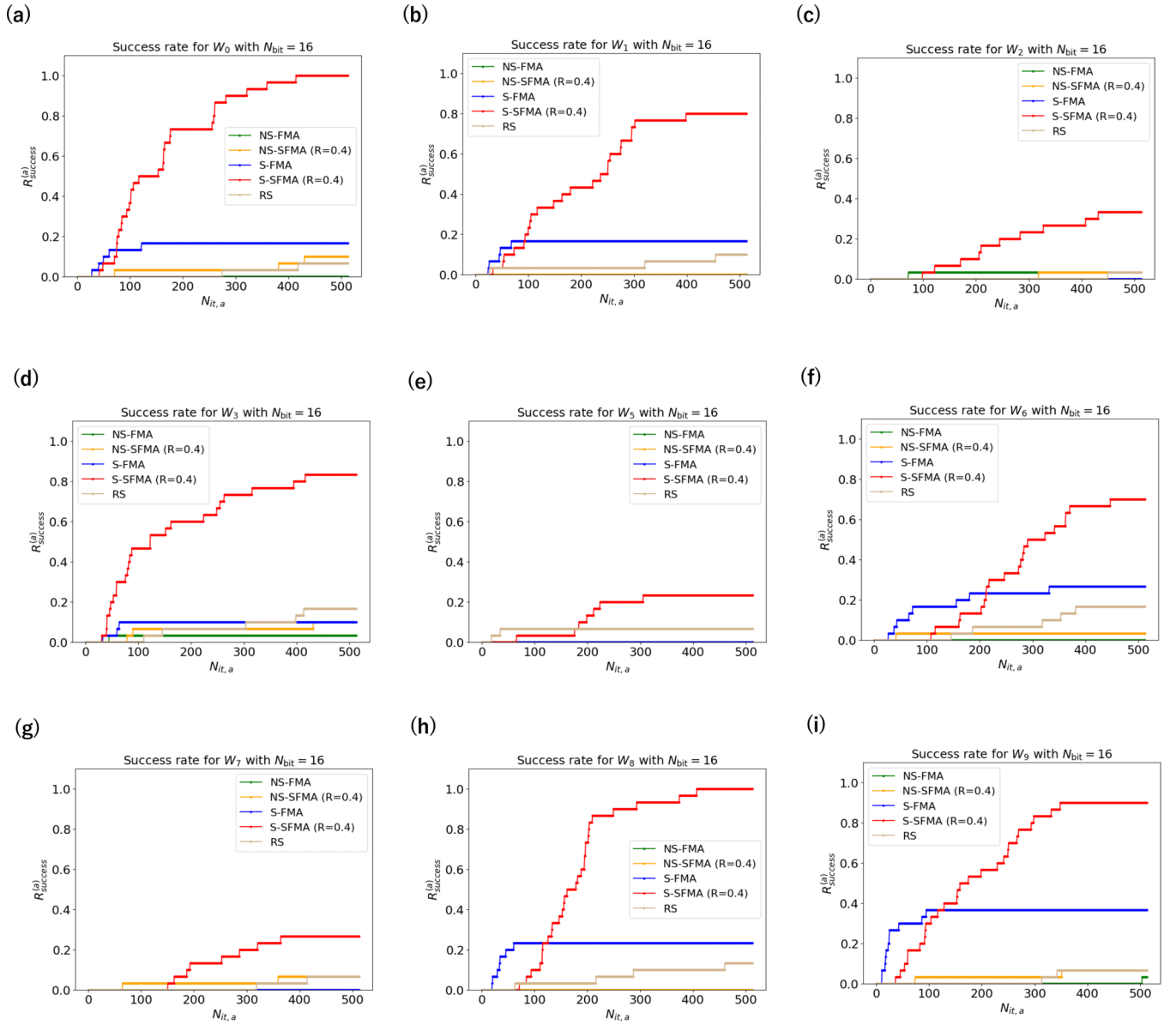


FIG. 11. Results for $R_{\min, \text{success}}^{(a)}$ with $N_{\text{bit}} = 16$ and $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$. Plots in panels (a)–(i) are the ones for W_n with $n = 0, 1, 2, 3, 5, 6, 7, 8$, and 9, respectively.

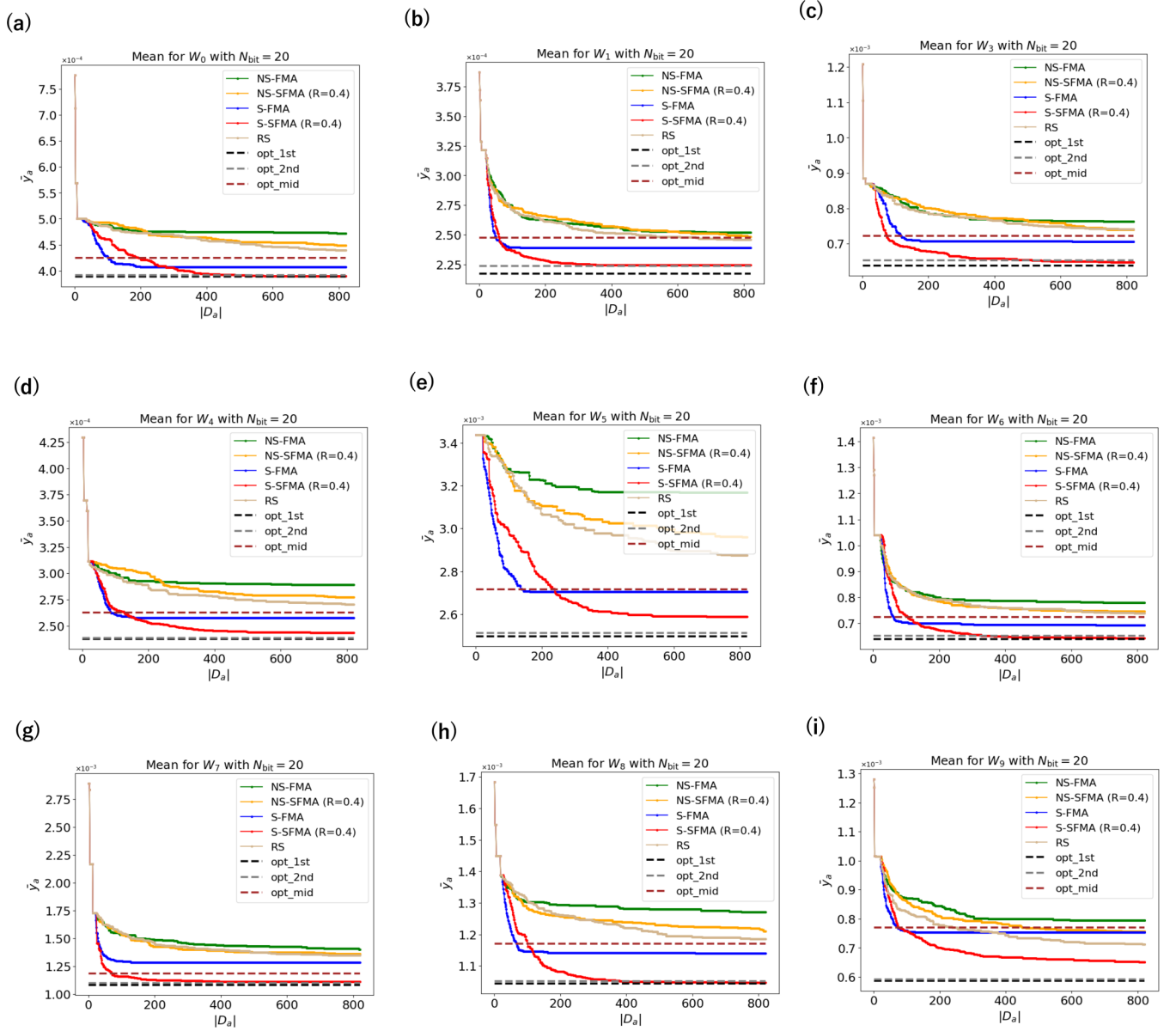


FIG. 12. Results for $\bar{y}^{(a)}$ with $N_{bit} = 20$ and $N_{ite} = 2N_{bit}^2 + 1$. Plots in panels (a)–(i) are the ones for W_n with $n = 0, 1, 3, 4, 5, 6, 7, 8$ and 9 , respectively.

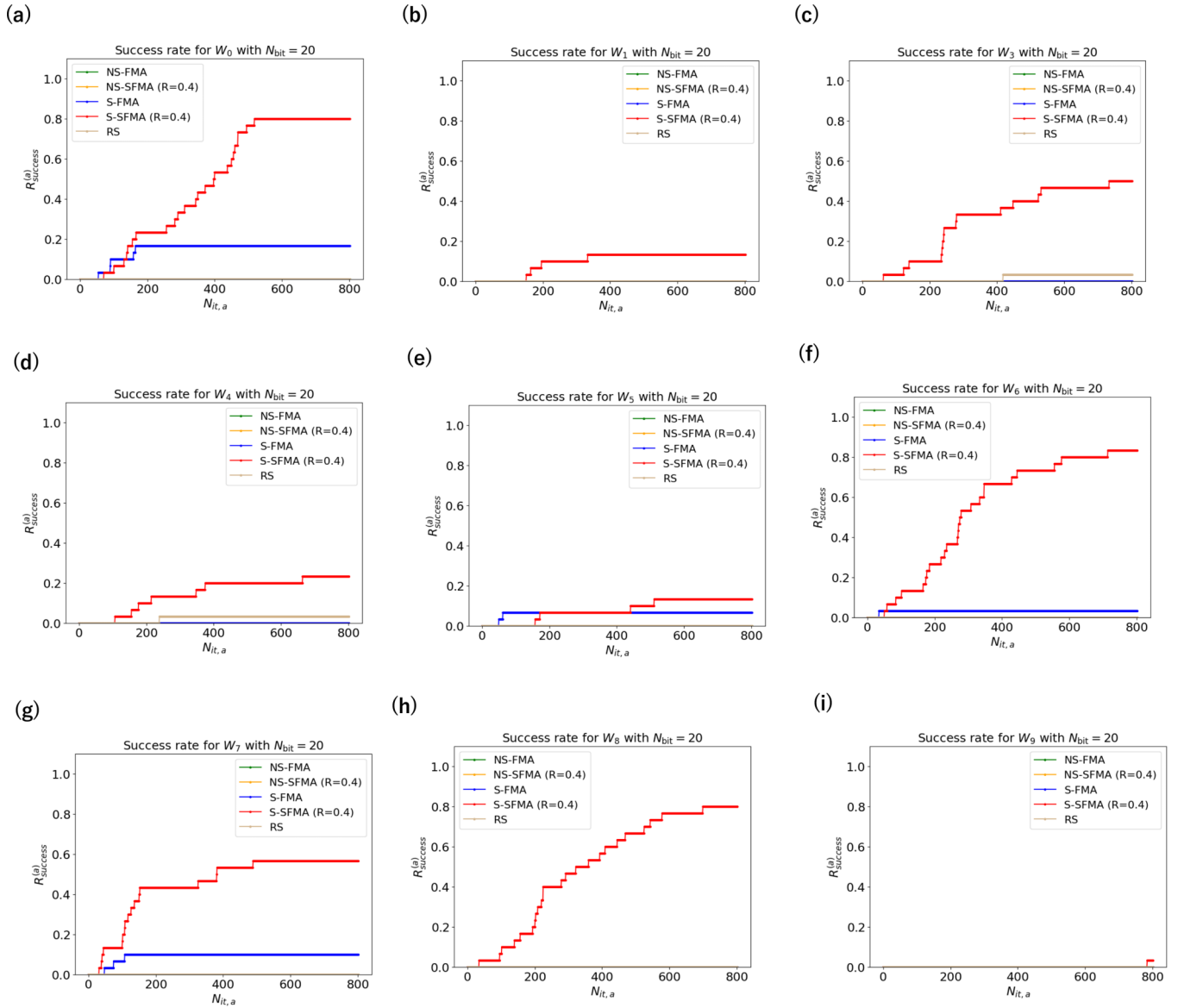


FIG. 13. Results for $R_{\min, \text{succ}}^{(a)}$ with $N_{\text{bit}} = 20$ and $N_{\text{ite}} = 2N_{\text{bit}}^2 + 1$. Plots in panels (a)–(i) are the ones for W_n with $n = 0, 1, 3, 4, 5, 6, 7, 8$ and 9, respectively.

APPENDIX E: IMPROVED SFMA FOR OTHER W MATRICES

In this Appendix, we discuss the improvements of SFMA for other W matrices and present the results for $\bar{y}^{(a)}$ and $R_{\min, \text{success}}^{(a)}$ in Figs. 14–17. Figures 14 and 15 display the results for $\bar{y}^{(a)}$ and $R_{\min, \text{success}}^{(a)}$ with $N_{\text{bit}} = 12$ and 16, respectively. Meanwhile, Figs. 16 and 17 present those for $\bar{y}^{(a)}$ and $R_{\min, \text{success}}^{(a)}$ with $N_{\text{bit}} = 20$, respectively. Note that, as explained in Sec. IV C, the four algorithms (S-SFMA, S-FMA, NS-SFMA, and NS-FMA) and RS are run such that the datasets with the sizes equal to $2N_{\text{bit}}^2 + 1$ are incorporated with additional data points generated by running the BBO loops $2N_{\text{bit}}^2$ ($4N_{\text{bit}}^2$) times for $N_{\text{bit}} = 12$ and 16 ($N_{\text{bit}} = 20$).

Let us begin from explaining the results in Fig. 14. Here, we have made the improvements for instances W_5 and W_7 by taking $R = 0.1$, $N_{\text{ite}} = 4N_{\text{bit}}^2 + 1$. The results demonstrate that the fastest convergence or the lowest value of N_{conv} is achieved by ISFMA (S-SFMA with $R = 0.4$) for W_5 (W_7). Similarly, ISFMA (S-SFMA with $R = 0.4$) exhibits the highest accuracy for W_5 (W_7). Next, let us explain the results for $N_{\text{bit}} = 16$ displayed in Fig. 15. We have investigated two instances W_2 and W_5 , and for both cases, ISFMA are created by setting $R = 0.1$, $N_{\text{ite}} = 4N_{\text{bit}}^2 + 1$. As a result, ISFMA only yields N_{conv} and demonstrates the highest $R_{\text{success}}^{\text{final}}$ in these

two instances. Finally, we discuss the results for $N_{\text{bit}} = 20$. We analyze five cases, W_n with $n = 1, 3, 4, 5$, and 7, and take N_{ite} to be $N_{\text{ite}} = 6N_{\text{bit}}^2 + 1$. For W_n with $n = 3, 5$, and 7, we improve SFMA by taking $R = 0.1$, while for the rest we refine SFMA using two different values of R as we have done for W_9 : see Sec. IV C. Namely, for W_1 (W_4) in the first part we run SFMA with $R = 0.1$, $N_{\text{ite}} = 5N_{\text{bit}}^2$ ($R = 0.1$, $N_{\text{ite}} = 4N_{\text{bit}}^2$), whereas in the second part we run SFMA by tuning the value of R to be $R = 0.01$. As we have done in Sec. IV C, we call the improved algorithms given solely by $R = 0.1$ ISFMA₁ while we call those created by both $R = 0.1, 0.01$ ISFMA₂. Note that ISFMA₁ is performed by augmenting the dataset obtained in the first part of the BBO loops in ISFMA₂ with the datasets generated by the second parts of the BBO loops. We benchmark the performance of the five (six) algorithms and RS for W_3 , W_5 , and W_7 (W_1 and W_4). As a result, ISFMA or ISFMA_i ($i = 1, 2$) do not yield the lowest value of N_{conv} in all the instances; however, they enable us to achieve the highest values of $R_{\text{success}}^{\text{final}}$ in all the instances. Consequently, we consider that the advantage of the improved SFMA (ISFMA and ISFMA_i) in accuracy becomes more prominent when the system size N_{bit} gets larger. Moreover, ISFMA as well as ISFMA_i outperform the standardized SFMA with $R = 0.4$ in all the instances for $N_{\text{bit}} = 16, 20$.

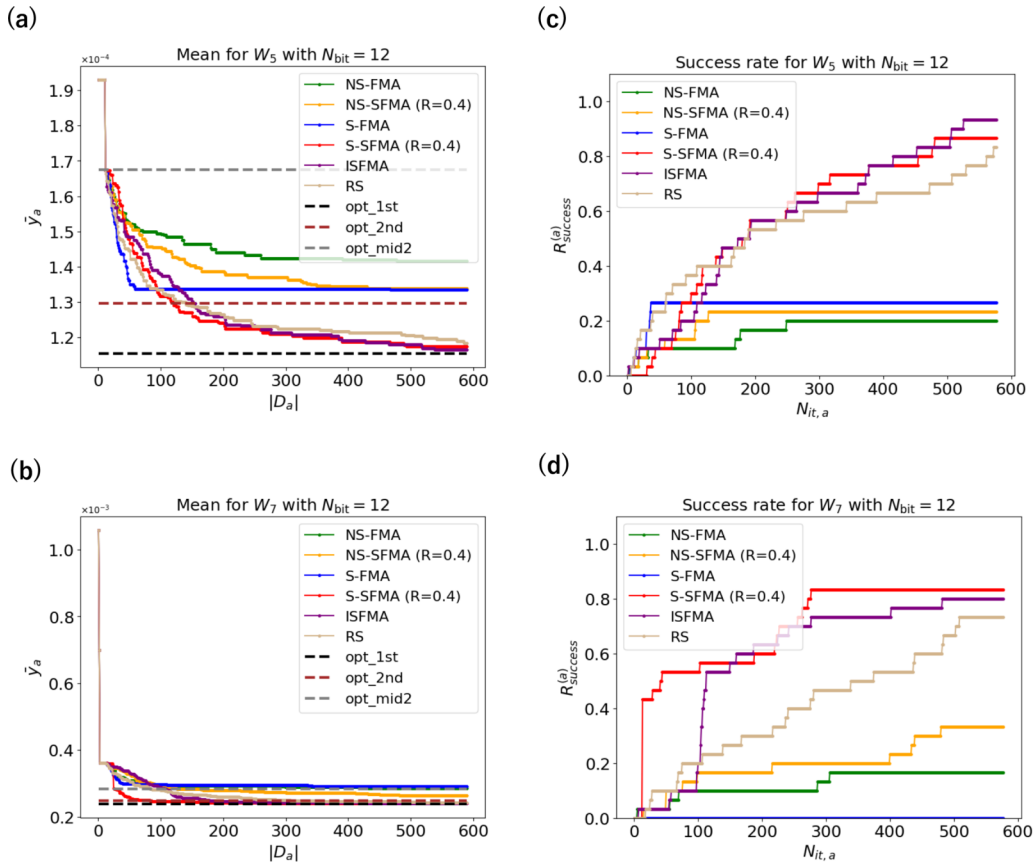


FIG. 14. Results for $\bar{y}^{(a)}$ and $R_{\min, \text{success}}^{(a)}$ with $N_{\text{bit}} = 12$ and $N_{\text{ite}} = 4N_{\text{bit}}^2 + 1$ for W_5 [panels (a) and (c)] and W_7 [(b) and (d)], respectively. The plots in panels (a) and (b) [panels (c) and (d)] represent the results for $\bar{y}^{(a)}$ ($R_{\min, \text{success}}^{(a)}$).

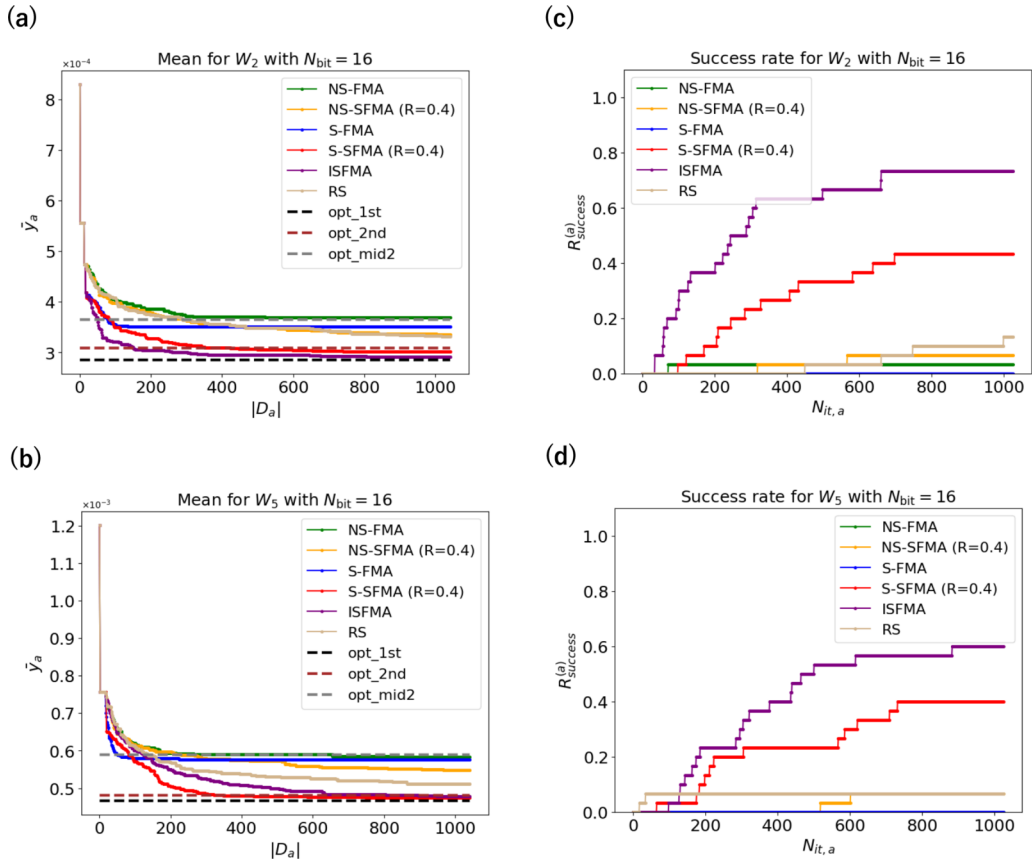


FIG. 15. Results for $\bar{y}^{(a)}$ and $R_{\text{min},\text{success}}^{(a)}$ with $N_{\text{bit}} = 16$ and $N_{\text{ite}} = 4N_{\text{bit}}^2 + 1$ for W_2 [panels (a) and (c)] and W_5 [panels (b) and (d)], respectively. The plots in panels (a) and (b) [panels (c) and (d)] represent the results for $\bar{y}^{(a)}$ ($R_{\text{min},\text{success}}^{(a)}$).

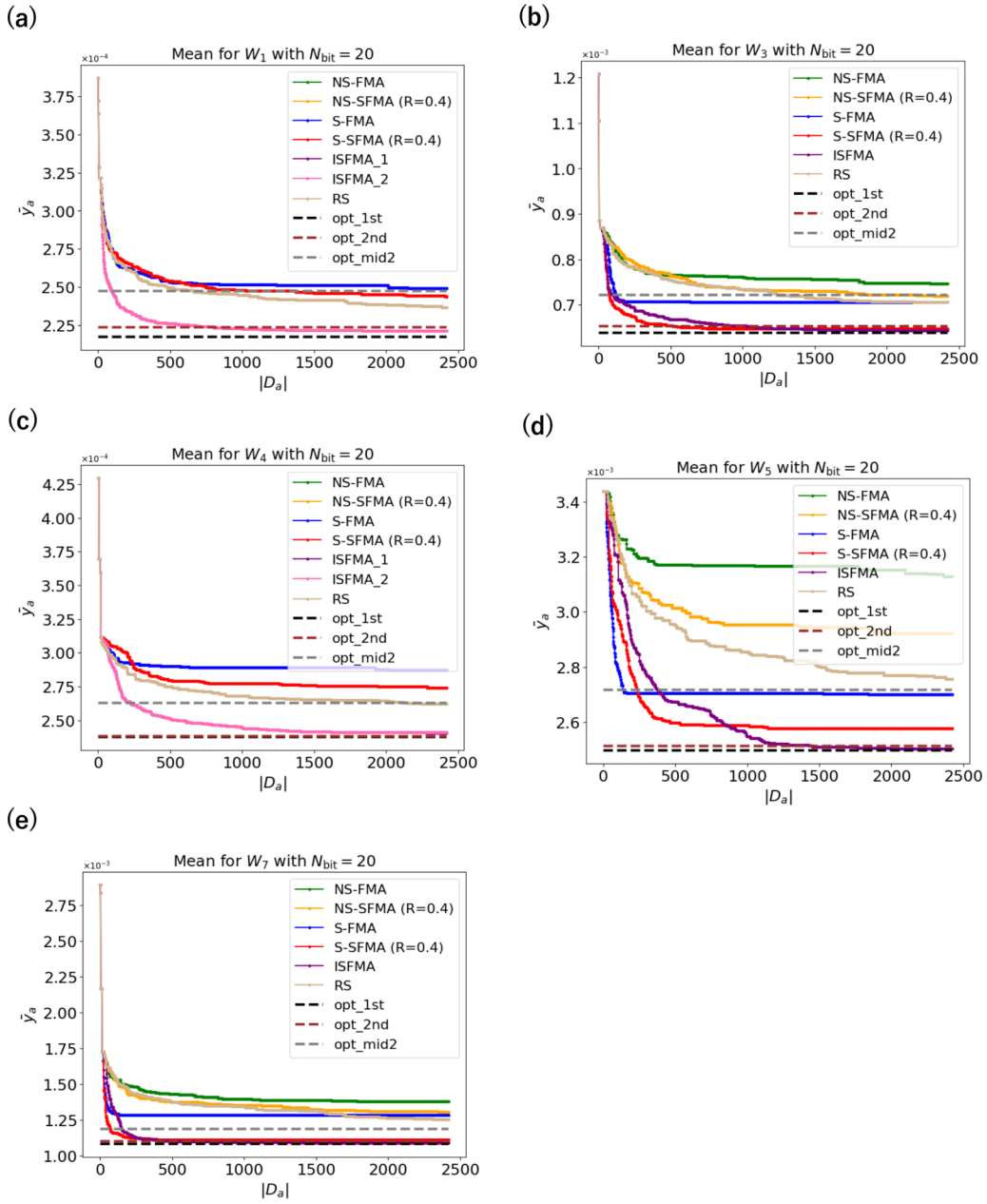


FIG. 16. Results for $\bar{y}_a^{(a)}$ with $N_{\text{bit}} = 20$ and $N_{\text{ite}} = 6N_{\text{bit}}^2 + 1$ for W_1 [panel (a)], W_3 [panel (b)], W_4 [panel (c)], W_5 [panel (d)], and W_7 [panel (e)], respectively.

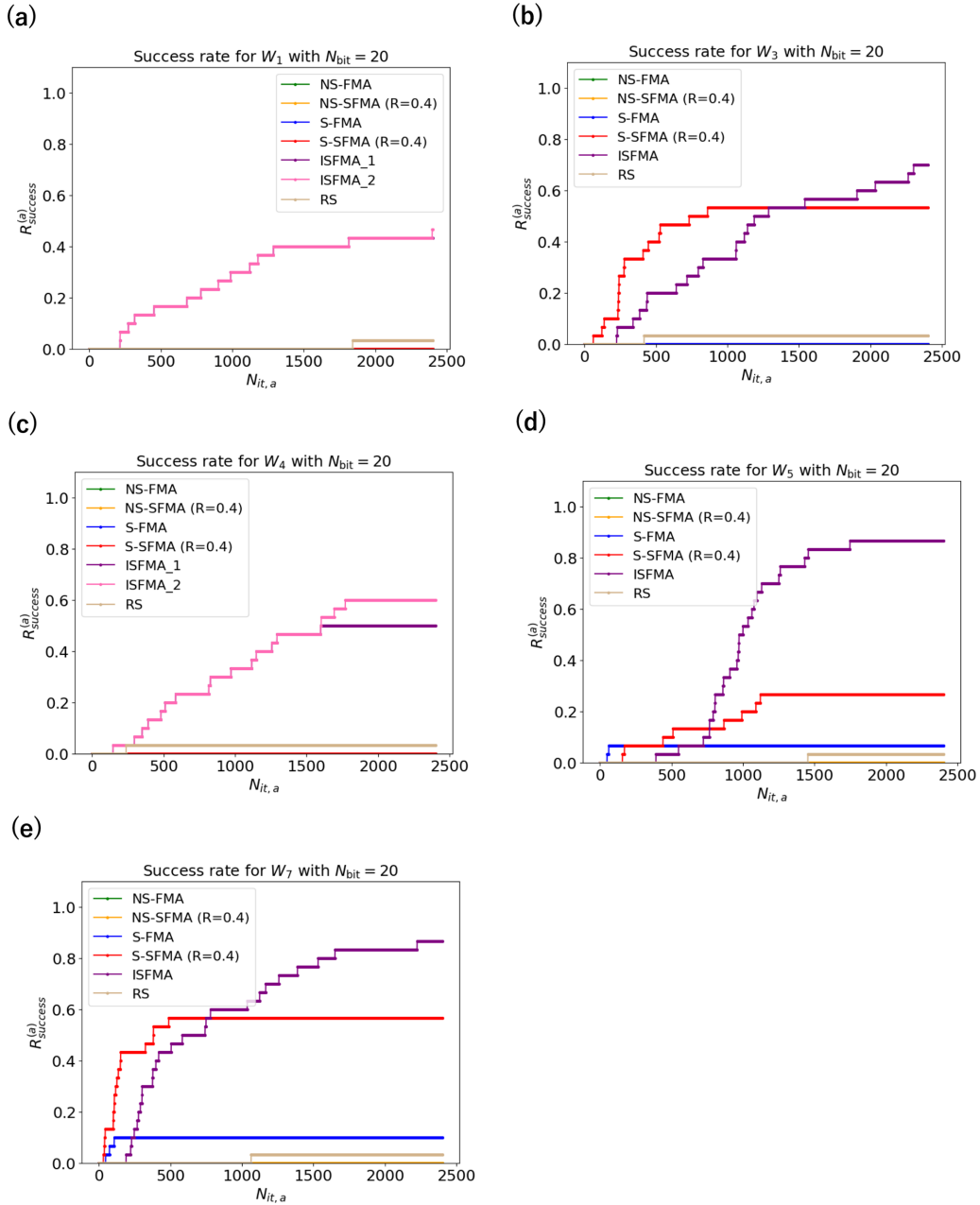


FIG. 17. Results for $R_{\min, \text{success}}^{(a)}$ with $N_{\text{bit}} = 20$ and $N_{\text{ite}} = 6N_{\text{bit}}^2 + 1$ for W_1 [panel (a)], W_3 [panel (b)], W_4 [panel (c)], W_5 [panel (d)], and W_7 [panel (e)], respectively.

- [1] M. Mezard and A. Montanari, *Information, Physics, and Computation* (Oxford University Press, Oxford, 2009).
- [2] M. J. Kochenderfer, *Algorithms for Optimization* (The MIT Press, Cambridge, MA, 2019).
- [3] A. Lucas, Ising formulations of many np problems, *Front. Phys.* **2**, 05 (2014).
- [4] Edited by C. Suh, *Convex Optimization for Machine Learning* (Emerald Publishing Limited, 2022).
- [5] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, Quantum annealing for industry applications: Introduction and review, *Rep. Prog. Phys.* **85**, 104001 (2022).
- [6] F. Phillipson, Quantum computing in logistics and supply chain management an overview, [arXiv:2402.17520](https://arxiv.org/abs/2402.17520).
- [7] A. Das and B. K. Chakrabarti, Colloquium: Quantum annealing and analog quantum computation, *Rev. Mod. Phys.* **80**, 1061 (2008).
- [8] S. Tanaka, R. Tamura, and B. K. Chakrabarti, *Quantum Spin glasses, Annealing and Computation* (Cambridge University Press, Cambridge, MA, 2017).
- [9] T. Albash and D. A. Lidar, Adiabatic quantum computation, *Rev. Mod. Phys.* **90**, 015002 (2018).

- [10] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, Perspectives of quantum annealing: Methods and implementations, *Rep. Prog. Phys.* **83**, 054401 (2020).
- [11] A. Rajak, S. Suzuki, A. Dutta, and B. K. Chakrabarti, Quantum annealing: An overview, *Phil. Trans. R. Soc. A* **381**, 20210417 (2023).
- [12] Göran Wendin, Quantum information processing with superconducting circuits: A review, *Rep. Prog. Phys.* **80**, 106001 (2017).
- [13] S. Kirkpatrick, C. D. Daniel Gelatt, Jr., and M. P. Vecchi, Optimization by simulated annealing, *Science* **220**, 671 (1983).
- [14] D. Bertsimas and J. Tsitsiklis, Simulated annealing, *Stat. Sci.* **8**, 10 (1993).
- [15] T. Kadowaki and H. Nishimori, Quantum annealing in the transverse Ising model, *Phys. Rev. E* **58**, 5355 (1998).
- [16] J. Brooke, D. Bitko, T. F. Rosenbaum, and G. Aeppli, Quantum annealing of a disordered magnet, *Science* **284**, 779 (1999).
- [17] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem, *Science* **292**, 472 (2001).
- [18] D. Aharonov, W. V. Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, Adiabatic quantum computation is equivalent to standard quantum computation, *SIAM Rev.* **50**, 755 (2008).
- [19] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, *et al.*, Quantum annealing with manufactured spins, *Nature (London)* **473**, 194 (2011).
- [20] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, Evidence for quantum annealing with more than one hundred qubits, *Nat. Phys.* **10**, 218 (2014).
- [21] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, Defining and detecting quantum speedup, *Science* **345**, 420 (2014).
- [22] S. Boixo, V. N. Smelyanskiy, A. Shabani, S. V. Isakov, M. Dykman, V. S. Denchev, M. H. Amin, A. Y. Smirnov, M. Mohseni, and H. Neven, Computational multiqubit in programmable quantum annealers, *Nat. Commun.* **7**, 10327 (2016).
- [23] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, What is the computational value of finite-range tunneling? *Phys. Rev. X* **6**, 031015 (2016).
- [24] T. Albash and D. A. Lidar, Demonstration of a scaling advantage for a quantum annealer over simulated annealing, *Phys. Rev. X* **8**, 031016 (2018).
- [25] S. Tsukamoto, M. Takatsu, S. Matsubara, and H. Tamura, An accelerator architecture for combinatorial optimization problems, *Fujitsu Sci. Tech. J.* **53**, 8 (2017).
- [26] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, Physics-inspired optimization for quadratic unconstrained problems using a digital annealer, *Front. Phys.* **7**, 48 (2019).
- [27] R. Au-Yeung, B. Camino, O. Rathore, and V. Kendon, Quantum algorithms for scientific computing, *Rep. Prog. Phys.* **87**, 116001 (2024).
- [28] A. D. King, A. Nocera, M. M. Rams, J. Dziarmaga, R. Wiersema, W. Bernoudy, J. Raymond, N. Kaushal, N. Heinsdorf, R. Harris, *et al.*, Beyond-classical computation in quantum simulation, *Science* **388**, 199 (2025).
- [29] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, [arXiv:1411.4028](https://arxiv.org/abs/1411.4028).
- [30] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, *et al.*, Quantum optimization using variational algorithms on near-term quantum devices, *Quantum Sci. Technol.* **3**, 030503 (2018).
- [31] S. Hadfield, Z. Wang, Bryan O’Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, From the quantum approximate optimization algorithm to a quantum alternating operator ansatz, *Algorithms* **12**, 34 (2019).
- [32] N. Sachdeva, G. S. Hartnett, S. Maity, S. Marsh, Y. Wang, A. Winick, R. Dougherty, D. Canuto, Y. Q. Chong, M. Hush, *et al.*, Quantum optimization using a 127-qubit gate-model IBM quantum computer can outperform quantum annealers for nontrivial binary optimization problems, [arXiv:2406.01743](https://arxiv.org/abs/2406.01743).
- [33] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, A review on quantum approximate optimization algorithm and its variants, *Phys. Rep.* **1068**, 1 (2024).
- [34] A. Abbas, A. Ambainis, B. Augustino, A. Bäertschi, H. Buhrman, C. Coffrin, G. Cortiana, V. Dunjko, D. J. Egger, B. G. Elmegreen, *et al.*, Challenges and opportunities in quantum optimization, *Nat. Rev. Phys.* **6**, 718 (2024).
- [35] B. Camino, J. Buckeridge, P. A. Warburton, V. Kendon, and S. M. Woodley, Quantum computing and materials science: A practical guide to applying quantum annealing to the configurational analysis of materials, *J. Appl. Phys.* **133**, 221102 (2023).
- [36] K. Hatakeyama-Sato, H. Adachi, M. Umeki, T. Kashikawa, K. Kimura, and K. Oyaizu, Automated design of Li⁺-conducting polymer by quantum-inspired annealing, *Macromol. Rapid Commun.* **43**, 2200385 (2022).
- [37] K. Hatakeyama-Sato, Y. Uchima, T. Kashikawa, K. Kimura, and K. Oyaizu, Extracting higher-conductivity designs for solid polymer electrolytes by quantum-inspired annealing, *RSC Adv.* **13**, 14651 (2023).
- [38] Q. Gao, G. O. Jones, T. Kobayashi, M. Sugawara, H. Yamashita, H. Kawaguchi, S. Tanaka, and N. Yamamoto, Quantum-classical computational molecular design of deuterated high-efficiency oled emitters, *Intell. Comput.* **2**, 0037, (2023).
- [39] H. Sampei, K. Saegusa, K. Chishima, T. Higo, S. Tanaka, Y. Yayama, M. Nakamura, K. Kimura, and Y. Sekine, Quantum annealing boosts prediction of multimolecular adsorption on solid surfaces avoiding combinatorial explosion, *JACS Au* **3**, 991 (2023).
- [40] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, Quantum computational chemistry, *Rev. Mod. Phys.* **92**, 015003 (2020).
- [41] B. Bauer, S. Bravyi, M. Motta, and Garnet Kin-Lic Chan, Quantum algorithms for quantum chemistry and quantum materials science, *Chem. Rev.* **120**, 12685 (2020).
- [42] S. Lee, J. Lee, H. Zhai, Y. Tong, A. M. Dalzell, A. Kumar, P. Helms, J. Gray, Z.-H. Cui, W. Liu, *et al.*, Evaluating the evidence for exponential quantum advantage in

- ground-state quantum chemistry, *Nat. Commun.* **14**, 1952 (2023).
- [43] D. Venturelli and A. Kondratyev, Reverse quantum annealing approach to portfolio optimization problems, *Quantum Mach. Intell.* **1**, 17 (2019).
- [44] J. Lang, S. Zielinski, and S. Feld, Strategic portfolio optimization using simulated, digital, and quantum annealing, *Appl. Sci.* **12**, 12288 (2022).
- [45] G. Buonaiuto, F. Gargiulo, G. D. Pietro, M. Esposito, and M. Pota, Best practices for portfolio optimization by quantum computing, experimented on real quantum devices, *Sci. Rep.* **13**, 19434 (2023).
- [46] E. Aguilera, J. d. Jong, F. Phillipson, S. Taamallah, and M. Vos, Multi-objective portfolio optimization using a quantum annealer, *Mathematics* **12**, 1291 (2024).
- [47] A. Perdomo, C. Truncik, I. Tubert-Brohman, G. Rose, and A. Aspuru-Guzik, Construction of model Hamiltonians for adiabatic quantum computation and its application to finding low-energy conformations of lattice protein models, *Phys. Rev. A* **78**, 012320 (2008).
- [48] A. Perdomo-Ortiz, N. Dickson, M. Drew-Brook, G. Rose, and A. Aspuru-Guzik, Finding low-energy conformations of lattice protein models by quantum annealing, *Sci. Rep.* **2**, 571 (2012).
- [49] R. Babbush, A. Perdomo-Ortiz, B. O’Gorman, W. Macready, and A. Aspuru-Guzik, Construction of energy functions for lattice heteropolymer models: Efficient encodings for constraint satisfaction programming and quantum annealing, in *Advances in Chemical Physics*, edited by S. A. Rice and A. R. Dinner (Wiley, 2014), Vol. 155, Chap. 5, p. 201.
- [50] A. Irbäck, L. Knuthson, S. Mohanty, and C. Peterson, Folding lattice proteins with quantum annealing, *Phys. Rev. Res.* **4**, 043013 (2022).
- [51] A. Irbäck, L. Knuthson, S. Mohanty, and C. Peterson, Using quantum annealing to design lattice proteins, *Phys. Rev. Res.* **6**, 013162 (2024).
- [52] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning* (Springer, New York, NY, 2006).
- [53] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT Press, Cambridge, MA, 2017), Vol. 1.
- [54] P. Mehta, M. Bukov, Ching-Hao Wang, Alexandre G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, A high-bias, low-variance introduction to machine learning for physicists, *Phys. Rep.* **810**, 1 (2019).
- [55] K. P. Murphy, *Probabilistic Machine Learning: An Introduction* (MIT Press, Cambridge, MA, 2022).
- [56] Y. Tian, Y. Zhang, and H. Zhang, Recent advances in stochastic gradient descent in deep learning, *Mathematics* **11**, 682 (2023).
- [57] C. M. Bishop and H. Bishop, *Deep Learning: Foundations and Concepts* (Springer Nature, Switzerland AG, 2023).
- [58] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proc. Natl. Acad. Sci. USA* **79**, 2554 (1982).
- [59] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, A learning algorithm for Boltzmann machines, *Cognit. Sci.* **9**, 147 (1985).
- [60] G. E. Hinton, Training products of experts by minimizing contrastive divergence, *Neural Comput.* **14**, 1771 (2002).
- [61] Mohammad H. Amin, E. Andriyash, J. Rolfe, B. Kulchysky, and R. Melko, Quantum Boltzmann machine, *Phys. Rev. X* **8**, 021050 (2018).
- [62] P. Date, D. Arthur, and L. Pusey-Nazzaro, Qubo formulations for training machine learning models, *Sci. Rep.* **11**, 10029 (2021).
- [63] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, *Nature (London)* **549**, 195 (2017).
- [64] V. Dunjko and H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: A review of recent progress, *Rep. Prog. Phys.* **81**, 074001 (2018).
- [65] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, *Quantum Sci. Technol.* **4**, 043001 (2019).
- [66] M. Schuld and F. Petruccione, *Machine Learning with Quantum Computers* (Springer, Cham, 2021), Vol. 676.
- [67] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles, Challenges and opportunities in quantum machine learning, *Nat. Comput. Sci.* **2**, 567 (2022).
- [68] M. Sajjan, J. Li, R. Selvarajan, S. H. Sureshbabu, S. S. Kale, R. Gupta, V. Singh, and S. Kais, Quantum machine learning for chemistry and physics, *Chem. Soc. Rev.* **51**, 6475 (2022).
- [69] A. Zeguendry, Z. Jarir, and M. Quafafou, Quantum machine learning: A review and case studies, *Entropy* **25**, 287 (2023).
- [70] Y. Wang and J. Liu, A comprehensive review of quantum machine learning: From NISQ to fault tolerance, *Rep. Prog. Phys.* **87**, 116402 (2024).
- [71] A. Agrawal and A. Choudhary, Perspective: Materials informatics and big data: Realization of the fourth paradigm of science in materials science, *APL Mater.* **4**, 053208 (2016).
- [72] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and C. Kim, Machine learning in materials informatics: Recent applications and prospects, *npj Comput. Mater.* **3**, 54 (2017).
- [73] K. Alberi, M. B. Nardelli, A. Zakutayev, L. Mitas, S. Curtarolo, A. Jain, M. Fornari, N. Marzari, I. Takeuchi, M. L. Green, *et al.*, The 2019 materials by design roadmap, *J. Phys. D* **52**, 013001 (2018).
- [74] A. Agrawal and A. Choudhary, Deep materials informatics: Applications of deep learning in materials science, *MRS Commun.* **9**, 779 (2019).
- [75] S. Hong, C. H. Liow, J. M. Yuk, H. R. Byon, Y. Yang, E. Cho, J. Yeom, G. Park, H. Kang, S. Kim, *et al.*, Reducing time to discovery: Materials and molecular modeling, imaging, informatics, and integration, *ACS Nano* **15**, 3971 (2021).
- [76] Y. Iwasaki, R. Sawada, E. Saitoh, and M. Ishida, Machine learning autonomous identification of magnetic alloys beyond the Slater-Pauling limit, *Commun. Mater.* **2**, 31 (2021).
- [77] A. Merchant, S. Batzner, S. S. Schoenholz, M. Aykol, G. Cheon, and E. Dogus Cubuk, Scaling deep learning for materials discovery, *Nature (London)* **624**, 80 (2023).
- [78] C. Li and K. Zheng, Methods, progresses, and opportunities of materials informatics, *InfoMat* **5**, e12425 (2023).
- [79] C. Chen, D. T. Nguyen, S. J. Lee, N. A. Baker, A. S. Karakoti, L. Lauw, C. Owen, K. T. Mueller, B. A. Bilodeau, V. Murugesan, *et al.*, Accelerating computational materials discovery with machine learning and cloud high-performance

- computing: From large-scale screening to experimental validation, *J. Am. Chem. Soc.* **146**, 20009 (2024).
- [80] D. Sivan, K. S. Kumar, A. Abdullah, V. Raj, I. I. Misnon, S. Ramakrishna, and R. Jose, Advances in materials informatics: A review, *J. Mater. Sci.* **59**, 2602 (2024).
- [81] K. Kitai, J. Guo, S. Ju, S. Tanaka, K. Tsuda, J. Shiomi, and R. Tamura, Designing metamaterials with quantum annealing and factorization machines, *Phys. Rev. Res.* **2**, 013319 (2020).
- [82] Y. Seki, R. Tamura, and S. Tanaka, Black-box optimization for integer-variable problems using Ising machines and factorization machines, [arXiv:2209.01016](https://arxiv.org/abs/2209.01016).
- [83] B. A. Wilson, Z. A. Kudyshev, A. V. Kildishev, S. Kais, V. M. Shalaev, and A. Boltasseva, Machine learning framework for quantum sampling of highly constrained, continuous optimization problems, *Appl. Phys. Rev.* **8**, 041418 (2021).
- [84] T. Inoue, Y. Seki, S. Tanaka, N. Togawa, K. Ishizaki, and S. Noda, Towards optimization of photonic-crystal surface-emitting lasers via quantum annealing, *Opt. Express* **30**, 43503 (2022).
- [85] T. Matsumori, M. Taki, and T. Kadowaki, Application of QUBO solver using black-box optimization to structural design for resonance avoidance, *Sci. Rep.* **12**, 12143 (2022).
- [86] T. Kadowaki and M. Ambai, Lossy compression of matrices by black box optimisation of mixed integer nonlinear programming, *Sci. Rep.* **12**, 15482 (2022).
- [87] Z. Mao, Y. Matsuda, R. Tamura, and K. Tsuda, Chemical design with GPU-based Ising machines, *Digital Discovery* **2**, 1098 (2023).
- [88] A. Tucs, F. Berenger, A. Yumoto, R. Tamura, T. Uzawa, and K. Tsuda, Quantum annealing designs nonhemolytic antimicrobial peptides in a discrete latent space, *ACS Med. Chem. Lett.* **14**, 577 (2023).
- [89] K. Nawa, T. Suzuki, K. Masuda, S. Tanaka, and Y. Miura, Quantum annealing optimization method for the design of barrier materials in magnetic tunnel junctions, *Phys. Rev. Appl.* **20**, 024044 (2023).
- [90] C.-Y. Liu, H.-Y. Wang, P.-Y. Liao, C.-J. Lai, and M.-H. Hsieh, Implementation of trained factorization machine recommendation system on quantum annealer, in *Proceedings of the 2024 International Joint Conference on Neural Networks (IJCNN)* (IEEE, Piscataway, NJ, 2024), pp. 1–8.
- [91] Y. Couzinié, Y. Seki, Y. Nishiya, H. Nishi, T. Kosugi, S. Tanaka, and Y.-i. Matsushita, Machine learning supported annealing for prediction of grand canonical crystal structures, *J. Phys. Soc. Jpn.* **94**, 044802 (2025).
- [92] J. Lin, T. Tada, A. Koizumi, M. Sumita, K. Tsuda, and R. Tamura, Determination of stable proton configurations by black-box optimization using an Ising machine, *J. Phys. Chem. C* **129**, 2332 (2025).
- [93] C. Liang, D. Das, J. Guo, R. Tamura, Z. Mao, and K. Tsuda, Predicting symmetric structures of large crystals with GPU-based Ising machines, *Commun. Phys.* **8**, 477 (2025).
- [94] M. Nakano, Y. Seki, S. Kikuchi, and S. Tanaka, SWIFT-FMQA: Enhancing factorization machine with quadratic-optimization annealing via sliding window, *IEEE Access* **14**, 10977 (2026).
- [95] Y. Minamoto and Y. Sakamoto, A black-box optimization method with polynomial-based kernels and quadratic-optimization annealing, [arXiv:2501.04225](https://arxiv.org/abs/2501.04225).
- [96] T. Kashimata, Y. Hamakawa, M. Yamasaki, and K. Tatsumura, Real-time black-box optimization for dynamic discrete environments using embedded Ising machines, [arXiv:2506.16924](https://arxiv.org/abs/2506.16924).
- [97] R. Baptista and M. Poloczek, Bayesian optimization of combinatorial structures, in *Proceedings of the 35th International Conference on Machine Learning* (PMLR, 2018), pp. 462–471.
- [98] Ami S. Koshikawa, M. Ohzeki, T. Kadowaki, and K. Tanaka, Benchmark test of black-box optimization using D-wave quantum annealer, *J. Phys. Soc. Jpn.* **90**, 064001 (2021).
- [99] K. Morita, Y. Nishikawa, and M. Ohzeki, Random postprocessing for combinatorial bayesian optimization, *J. Phys. Soc. Jpn.* **92**, 123801 (2023).
- [100] J. E. Kim and Y. Wang, Quantum approximate bayesian optimization algorithms with two mixers and uncertainty quantification, *IEEE Trans. Quantum Eng.* **4**, 3102817 (2023).
- [101] M. Ambai and I. Sato, SPADE: Scalar product accelerator by integer decomposition for object detection, in *Proceedings of the 13th European Conference on Computer Vision (ECCV 2014)* (Springer, Cham, 2014), Part V 13, pp. 267–281.
- [102] B. Yoon, Nga T. T. Nguyen, Chia Cheng Chang, and E. Rrapaj, Lossy compression of statistical data using quantum annealer, *Sci. Rep.* **12**, 3814 (2022).
- [103] S. Rendle, Factorization machines, in *Proceedings of the 2010 IEEE International Conference on Data Mining* (IEEE, Piscataway, NJ, 2010), pp. 995–1000.
- [104] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. Tang, On large-batch training for deep learning: Generalization gap and sharp minima, [arXiv:1609.04836](https://arxiv.org/abs/1609.04836).
- [105] O. Räisä, J. Jälkö, and A. Honkela, Subsampling is not magic: Why large batch sizes work for differentially private stochastic optimisation, in *Proceedings of the 41st International Conference on Machine Learning (ICML'24)* (JMLR.org, Vienna, Austria, 2024).
- [106] D. D. Lee and H. S. Seung, Learning the parts of objects by non-negative matrix factorization, *Nature (London)* **401**, 788 (1999).
- [107] C. J. Long, D. Bunker, X. Li, V. L. Karen, and I. Takeuchi, Rapid identification of structural phases in combinatorial thin-film libraries using x-ray diffraction and non-negative matrix factorization, *Rev. Sci. Instrum.* **80**, 103902 (2009).
- [108] H. Asaoka and K. Kudo, Image analysis based on nonnegative/binary matrix factorization, *J. Phys. Soc. Jpn.* **89**, 085001 (2020).
- [109] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, *International Conference on Learning Representations (ICLR)*, 2015.
- [110] N. Pirnay, V. Ulitzsch, F. Wilde, J. Eisert, and J.-P. Seifert, An in-principle super-polynomial quantum advantage for approximating combinatorial optimization problems via computational learning theory, *Sci. Adv.* **10**, eadj5170 (2024).
- [111] <https://doi.org/10.5281/zenodo.17960406>.
- [112] D-Wave Systems, Inc., *neal*: Simulated annealing sampler (2025), <https://github.com/dwavesystems/neal>.
- [113] D-Wave Systems, Inc., *d-wave quantum computing platform* (2025), <https://www.dwavesys.com>.