scientific reports

OPEN

Lossy compression of matrices by black box optimisation of mixed integer nonlinear programming

Tadashi Kadowaki[1,2]✉ & Mitsuru Ambai[3]

In edge computing, suppressing data size is a challenge for machine learning models that perform complex tasks such as autonomous driving, in which computational resources (speed, memory size and power) are limited. Efficient lossy compression of matrix data has been introduced by decomposing it into the product of an integer and real matrices. However, its optimisation is difficult as it requires simultaneous optimisation of an integer and real variables. In this paper, we improve this optimisation by utilising recently developed black-box optimisation (BBO) algorithms with an Ising solver for binary variables. In addition, the algorithm can be used to solve mixed-integer programming problems that are linear and non-linear in terms of real and integer variables, respectively. The differences between the choice of Ising solvers (simulated annealing, quantum annealing and simulated quenching) and the strategies of the BBO algorithms (BOCS, FMQA and their variations) are discussed for further development of the BBO techniques.

In the last decade, there have grown a number of applications of neural networks and machine learning. Based on the success of the neural network and machine learning applications in the real-world setting, there has also been an increase in the complexity level of the mathematical models. These models are represented by parameters in large matrices, such as weight matrices in the neural network. Suppressing the memory size of the model is a rising issue for implementing those applications in the real-world, especially in edge computing.

One of the authors proposed a lossy decomposition scheme of such matrices into a product of an integer matrix and a real matrix (hereinafter referred to as integer decomposition)[1]. The concept of this matrix decomposition is that the integer matrix has a smaller memory footprint compared with the real matrix in floating-point representation. The integer variable uses only one or two bits for binary or ternary representation, whereas the floating-point variable uses 32 or 64 bits. Thus, the compression rate is approximated by the ratio of the sizes of the original matrix and the real matrix. For a specific machine learning task, the memory footprint is reduced to 1/3, and the performance is 36.9 times faster than the original one with 1.5% increase in loss of accuracy.

This process of matrix compression optimises the integer and the real matrices simultaneously. The problem is a mixed-integer non-linear programming (MINLP) problem, an NP-hard problem. In addition, the original proposal includes a strategy in a greedy manner to reduce optimisation complexity; however, the greedy method does not reach the best solution in general.

In this paper, we improve the procedure of the integer decomposition through a recently developed black-box optimisation (BBO) technique for binary variables[2–4]. These methods use a data set of binary input and real-value output to produce a surrogate model, i.e. a pseudo-Boolean function, and optimise the model using Ising solvers. The tentative optimal solution from the Ising solver can be evaluated by the black-box function, e.g. conducting experiments, simulations or other methods depending on the problem. In the matrix decomposition, we transform our MINLP problem into a non-linear integer programming (NLIP) problem. Although we still have an explicit form of the cost function with integer variables, we can consider this function as a black-box function. Although this paper demonstrates a binary variable problem, it is straightforward to handle an integer variable with binary variables.

Data acquisition is not a hard task thanks to the explicit form of the function. Thus, the number of iterations and evaluations of the cost function is not a big deal in this case. We will conduct $O(n^2)$ iterations for BBO, which is a unique feature of this study from a typical case with an expensive-to-evaluate function.

[1]DENSO CORPORATION, Kounan, Minato-ku, Tokyo 108-0075, Japan. [2]Research Center for Emerging Computing Technologies (RCECT), National Institute of Advanced Industrial Science and Technology (AIST), Umezono, Tsukuba, Ibaraki 305-8568, Japan. [3]DENSO IT Laboratory, Shibuya, Shibuya-ku, Tokyo 150-0002, Japan. ✉email: tadashi.kadowaki.j3m@jp.denso.com

Several tasks on lossy data compression have been commonly used in image and audio data processing, such as JPEG and MP3. Their algorithms are based on human limitations of frequency and time domains. General-purpose algorithms, such as low-rank approximation, non-negative matrix factorisation and non-negative/binary matrix factorisation algorithms, have been proposed[5–7]. These methods are not applicable for the integer decomposition because they target the decomposition by real or non-negative matrix.

Recently, a general matrix decomposition method has been proposed[8]. The proposed method optimises integer matrix and real matrix separately, while our algorithm does simultaneously. In addition, Thompson sampling of the MINLP surrogate model is studied with commercial mixed-integer programming (MIP) solvers[9]. Our algorithm provides solutions to MIP problems using Ising solvers, whereas such commercial solvers do not disclose details of algorithms.

A number of studies on the applications of the BBO of binary variables with Ising solvers are carried out, including aero-structural problem[2], nanomaterials[4], spin glass[10], commonality optimisation[11] and chemical structure optimisation[12,13]. This paper not only introduces another application but also proposes a novel algorithm for solving MINLP problems by the BBO using the Ising solver.

This paper is organised as follows. The "Integer decomposition" section defines the integer decomposition and derives a black-box formulation of the problem. Variations of BBO algorithms and Ising solvers tested in this study are described in the "Black-box optimisation" section. The "Results" section dedicates to the results of these algorithms and solvers. The final section summarises and discusses the results.

Integer decomposition

Original formulation of the integer decomposition.

The integer decomposition approximates a target matrix $\boldsymbol{W}$ by a product of binary (or ternary) matrix $\boldsymbol{M}$ and real matrix $\boldsymbol{C}$,

$$\boldsymbol{W} \sim \boldsymbol{V} = \boldsymbol{MC}. \tag{1}$$

The matrix sizes of $\boldsymbol{W}$, $\boldsymbol{M}$ and $\boldsymbol{C}$ are $N \times D$, $N \times K$ and $K \times D$, respectively. This decomposition is parameterised by K, which controls the approximation accuracy.

If $K = N$ (no compression), $\boldsymbol{M}$ and $\boldsymbol{C}$ can be $\boldsymbol{I}$ and $\boldsymbol{W}$,

$$\boldsymbol{V} = \boldsymbol{MC} = \boldsymbol{IW} = \boldsymbol{W}. \tag{2}$$

In the matrix compression, K is smaller than N. Thus, we cannot reproduce $\boldsymbol{W}$ by $\boldsymbol{M}$ and $\boldsymbol{C}$. Then, the optimal binary and real matrices can be obtained through minimisation of the cost function expressed by L_2 matrix norm,

$$\underset{\substack{\boldsymbol{M} \in \{-1,1\}^{N\times K} \\ \boldsymbol{C} \in \mathbb{R}^{K\times D}}}{\arg\min} \quad \|\boldsymbol{W} - \boldsymbol{MC}\|_2^2. \tag{3}$$

This is an MINLP problem; therefore, there is no general algorithm to efficiently find the exact solution.

The original algorithm finds the decomposition as a series of products,

$$\boldsymbol{V} = \sum_i^K \boldsymbol{m}_i \boldsymbol{c}_i^\top, \tag{4}$$

where $\boldsymbol{m}_i$ and $\boldsymbol{c}_i$ are N and D dimensional vectors and calculated step by step from $i = 1$ to $i = K$. The i-th vectors $\boldsymbol{m}_i$ and $\boldsymbol{c}_i$ are optimised by using pre-optimised vectors $\boldsymbol{m}_j$ and $\boldsymbol{c}_j$ ($j = 1, \ldots, i-1$),

$$\underset{\substack{\boldsymbol{m}_i \in \{-1,1\}^{N} \\ \boldsymbol{c}_i \in \mathbb{R}^{D}}}{\arg\min} \quad \left\| \left(\boldsymbol{W} - \sum_j^{i-1} \boldsymbol{m}_j \boldsymbol{c}_j^\top \right) - \boldsymbol{m}_i \boldsymbol{c}_i^\top \right\|_2^2. \tag{5}$$

The search space is drastically reduced from NK- to N-dimension, although convergence to the exact solution is not guaranteed. In each step, this algorithm finds the best rank-one approximation of the residual from the previous approximation. Therefore, updating the variables that are fixed previously is not possible, i.e. it cannot escape from local minima.

As shown in Eq. (4), the matrix decomposition has two types of arbitrariness. The first one is the order of columns (or rows) in the binary matrix $\boldsymbol{M}$ (the real matrix $\boldsymbol{C}$) indexed by i, that is, the order of sum in Eq. (4). The second one is the sign of each column in $\boldsymbol{M}$ (i.e. $\boldsymbol{m}_i, \boldsymbol{c}_i \to -\boldsymbol{m}_i, -\boldsymbol{c}_i$). Consequently, the total number of equivalent matrices is $K! \times 2^K$. In the case of $K = 3$, there are 48 exact solutions of Eq. (3). Figure 5 in the appendix shows examples of the 48 solutions for a specific instance.

Black-box formulation of the integer decomposition.

As shown in the previous subsection, the integer decomposition is an MINLP problem. To solve this problem by BBO algorithms, we convert it to an NLIP problem. If given $\boldsymbol{M}$ has linearly independent columns, we can calculate $\boldsymbol{C}$ using the least-squares method in matrix form,

$$\boldsymbol{C} = \left(\boldsymbol{M}^\top \boldsymbol{M}\right)^{-1} \boldsymbol{M}^\top \boldsymbol{W}, \tag{6}$$

where $\left(\boldsymbol{M}^\top \boldsymbol{M}\right)^{-1} \boldsymbol{M}^\top$ refers pseudoinverse, and thus the approximated matrix $\boldsymbol{V}$ is a function of $\boldsymbol{M}$

$$\boldsymbol{V}(\boldsymbol{M},\boldsymbol{C}) = \boldsymbol{M}\boldsymbol{C} = \boldsymbol{M}\left(\boldsymbol{M}^\top\boldsymbol{M}\right)^{-1}\boldsymbol{M}^\top\boldsymbol{W} = \boldsymbol{V}(\boldsymbol{M}). \tag{7}$$

Substituting this equation to Eq. (3), we have

$$\underset{\boldsymbol{M}\in\{-1,1\}^{NK}}{\arg\min} \ \left\|f(\boldsymbol{M})\right\|_2^2, \tag{8}$$

where

$$f(\boldsymbol{M}) = \boldsymbol{W} - \boldsymbol{M}\boldsymbol{C} = \boldsymbol{W} - \boldsymbol{M}\left(\boldsymbol{M}^\top\boldsymbol{M}\right)^{-1}\boldsymbol{M}^\top\boldsymbol{W}. \tag{9}$$

Now, we remove real-value parameters and have an NLIP problem. Note that the Taylor series of this cost function has infinite terms, and thus Ising solvers, including quantum annealer, cannot solve this optimisation problem directly. On the other hand, BBO can solve this problem if we deal with the input-output relationship of the cost function $\left\|f(\boldsymbol{M})\right\|_2^2$ as a black-box function. The optimisation algorithm employs not the explicit form of the function but a specific data set of the input-output relationship calculated from the function.

Black-box optimisation

BBO algorithms. Ising solvers find a solution of a quadratic function so we can approximate the data set $(\boldsymbol{x}, y)$ by the quadratic function $\hat{y}(\boldsymbol{x}) = \boldsymbol{x}^\top\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^\top\boldsymbol{x} + c$, where $\boldsymbol{x} \in \{-1,1\}^{NK}$ and $\boldsymbol{A}, \boldsymbol{b}$ and c are model parameters. Note that we use $\boldsymbol{x}$ in this surrogate model instead of $\boldsymbol{M}$, y as the cost associated with a given $\boldsymbol{x}$ and $\hat{y}$ as the approximated cost by the surrogate model. This function can be expressed in a simplified quadratic form $\boldsymbol{x}^\top\boldsymbol{A}\boldsymbol{x}$ if we include an additional dimension in the vector $\boldsymbol{x}$ as $(x_1, \ldots, x_{NK}, 1)$. The different algorithm handles the surrogate model differently.

Bayesian optimisation of combinatorial structures (BOCS)[2] treats the parameter $\boldsymbol{A}$ in Bayesian linear regression. The authors proposed to use the horseshoe prior[14],

$$\begin{aligned} \alpha_k|\beta_k^2, \tau^2, \sigma^2 &\sim N(0, \beta_k^2\tau^2\sigma^2) \\ \tau, \beta_k &\sim C^+(0,1) \\ P(\sigma^2) &= \sigma^{-2}, \end{aligned} \tag{10}$$

where α_k stands for the coefficient of the k-th variable in the linear regression and $C^+(0,1)$ is a half-Cauchy distribution. Note that as the surrogate model is linear, second-order terms x_ix_j are treated as independent explanatory variables, i.e. $(x_1, \ldots, x_n, x_1x_2, x_1x_3, \ldots, x_{n-1}x_n)$, where $n = NK$. Thus, the index k runs from 1 to $n + n(n-1)/2$. As the parameter $\boldsymbol{A}$ of the surrogate model is a distribution in BOCS, a specific value of $\boldsymbol{A}$ is chosen based on the distribution inspired by Thompson sampling[15]. In addition to the horseshoe prior, normal prior $\alpha_k \sim N(0, \sigma^2)$ and normal-gamma prior $\alpha_k, \sigma^{-2} \sim \mathrm{NormalGamma}(0, 1, 1, \beta)$ are tested.

The sampling from the horseshoe distribution is performed using the Monte Carlo sampling[14], which requires a longer execution time compared to the normal and normal-gamma distributions. Samplings of the variables from these distribution functions are accelerated using fast Gaussian samplers[16,17].

Factorisation machine with quantum annealing (FMQA)[4] utilises the factorisation machine (FM)[18] as the surrogate model. The surrogate model of degree $d = 2$ is defined as

$$\hat{y}(\boldsymbol{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{i=i+1}^{n} \langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle x_i x_j, \tag{11}$$

where $\langle \cdot, \cdot \rangle$ represents the dot product of two vectors of size k_{FM},

$$\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle := \sum_{l=1}^{k_{\mathrm{FM}}} v_{i,l} v_{j,l}. \tag{12}$$

The horseshoe prior and FM introduce sparsity in the surrogate model, whereas normal and normal-gamma priors do not. The results section shows the effects of this sparsity on the performances of algorithms. Each algorithm has its hyperparameter(s) to be fixed before conducting the BBO, i.e. the variance σ^2 in the normal prior, the shape parameter $\alpha (= 1)$ and the inverse scale parameter β in the normal-gamma prior and the size parameter k_{FM} in the FM, while the horseshoe prior has no hyperparameters. Hyperparameters σ^2 and β are optimised for a specific instance. Then, the optimal values are applied to other instances. We do not optimise the size parameter k_{FM} but choose eight from FMQA's proposal as well as 12 to have enough degree of freedom to represent α_k.

We refer to the vanilla BOCS as vBOCS, the normal prior BOCS as nBOCS and the normal-gamma BOCS as gBOCS. The differences in k_{FM} are identified by FMQA08 and FMQA12. In addition, a random search algorithm is referred to as RS, in which each vector $\boldsymbol{x}$ is randomly sampled without utilising the pre-obtained data set.

Ising solvers. BOCS and FMQA use an Ising solver to find the solution of the surrogate model represented in a quadratic form. We evaluate three Ising solvers: simulated annealing, QA and simulated quenching. Simulated annealing (SA) introduces a thermal fluctuation in the exploring process of the cost function[19]. It is implemented on the Monte Carlo simulation. The temperature in the simulation is a parameter that controls the probability of variable reversal such that the cost function increases. Initially, the temperature is high to help global

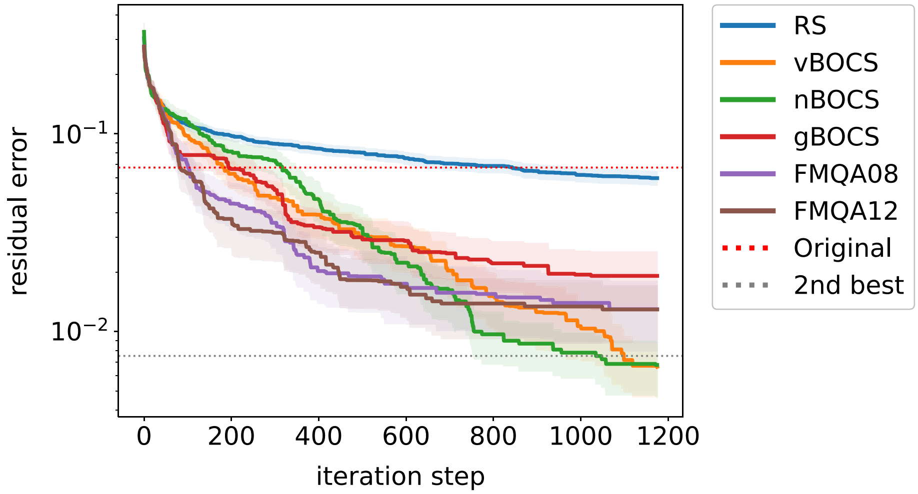

Figure 1. The residual error of lossy compression of a matrix (instance) as a function of iteration step among various algorithms with 95% confidence intervals. The red- and grey-dotted lines indicate the residual error of the original algorithm and the second-best solution by brute-force search.

search in the solution space and find the candidate basin harbouring the global minimum. Later, the temperature becomes low to find the lowest solution in the basin. If the cooling schedule is slow enough, i.e. $\propto 1/\log(t)$, SA finds the global minimum[20].

Quantum annealing (QA) takes place in the thermal fluctuation in SA by quantum fluctuation through a scheduled transverse field. The quantum system starts from a trivial state (superposition of all solutions) and finds the ground state of the cost function at the end of the annealing[21]. If the transverse field is scheduled as $\propto t^{-1/(2N-1)}$, the system converges to the ground state at the end[22].

We refer to simulated quenching (SQ) as a variation of SA with extremely rapid quenching of the temperature from high to zero. Although this algorithm simplifies and accelerates the Monte Carlo calculation, it eliminates the ability to search solution space globally at the early stage of annealing. Thus, this algorithm tends to be trapped in local minima more frequently than SA and QA.

We utilise SA and QA solvers in D-Wave Ocean SDK with default parameters. The default initial and final temperatures for SA are determined from approximately estimated maximum and minimum effective fields with scaling factors 2.9 and 0.4, respectively. The default annealing time for QA is 20 microseconds. In SQ, the temperature is not annealed but kept constant at 0.1. We optimise the surrogate model 10 times in each iteration step for all solvers to find a better solution.

Results

Integer decomposition was conducted for $(N \times D =)$ 8×100 matrix $\boldsymbol{W}$, which is constructed by shrinking from the final fully connected layer of VGG16 convolutional neural network[23]. We choose the decomposition parameter $K = 3$, and thus the matrix $\boldsymbol{W}$ is decomposed into a 8×3 binary matrix $\boldsymbol{M}$ and a 3×100 real matrix $\boldsymbol{C}$. As discussed in the "Original formulation of the integer decomposition" subsection, the size n of the problem is determined by the size of the binary matrix. In this case, $n = 8 \times 3 = 24$. The dimension of the model parameter α_k is $O(n^2)$ so that we start from the initial data set of size n and then add $2n^2$ data points iteratively to reach enough size for estimating the model parameter. In the current analysis ($n = 24$), we generate initial 24 data points followed by 1152 iterations (1176 in total). This number is remarkably small compared to the solution space $2^{24} \sim 1.7 \times 10^7$. All the algorithms we test in this paper are randomised algorithms and/or starting from a randomly selected data set, and thus we conduct 25 runs (or 100 runs for RS) to estimate the average performance. The performance also depends on the matrix $\boldsymbol{W}$. Therefore, we generate ten individual problem matrices (instances) to evaluate the performance over instances.

Figure 1 shows the results from different algorithms for the first instance. In the BBO process, SA is used as an Ising solver. The x and y axes represent the iteration step, the size of acquired data (including initial data) in linear scale and the residual error from the exact solution in log scale, respectively. Due to the limitation of the problem size n we tested, the compression quality is insufficient for typical applications. In this situation, the absolute error is not an appropriate measure to compare algorithms. Thus, we employ the residual error $\left(\|f(\boldsymbol{M})\|_2 - \|f(\boldsymbol{M}^*)\|_2\right)/\|W\|_2$, where $\boldsymbol{M}^*$ is the exact solution of Eq. (8). With this measure, we can evaluate how the solutions are close to the exact solution. The exact and the second-best solutions were separately obtained from brute-force search. The absolute error of the exact solution $\|f(\boldsymbol{M}^*)\|_2/\|W\|_2$, subtracted in the plot, is 0.461. All algorithms outperform the original approximated solution in the red-dotted line. The FMQA algorithm improves its solution faster than other algorithms at the early stage. However, the improvement does not continue at the later stage. vBOCS and nBOCS improve slowly, but the improvement continues during the process. They go under the line of the second-best solution (in grey-dotted line), suggesting that some individual runs find the exact solution.

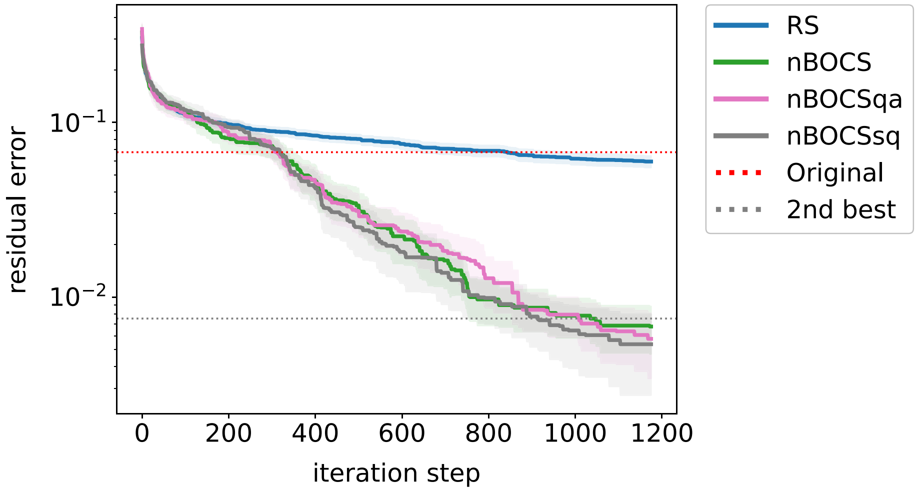

Figure 2. The residual error of lossy compression of a matrix as a function of iteration step among the Ising solvers (SA, QA and SQ).

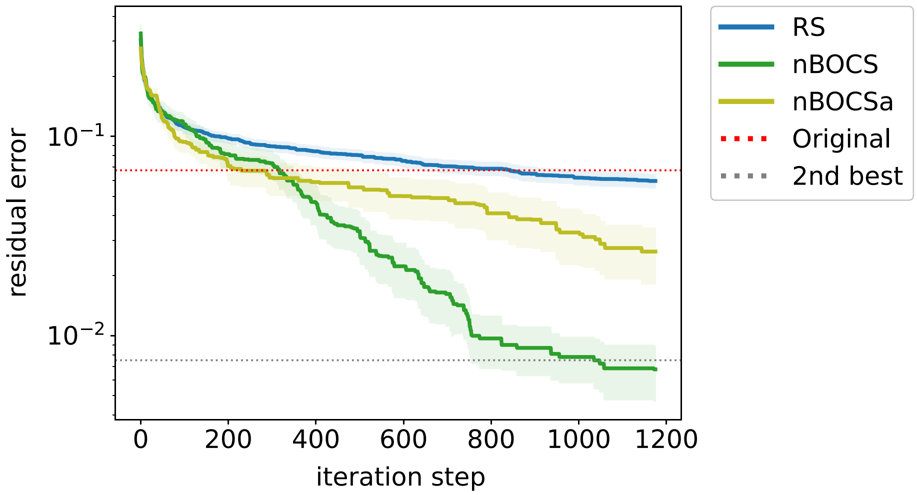

Figure 3. The residual error of lossy compression of a matrix for RS and nBOCS variants with and without data augmentation (nBOCSa and nBOCS).

Figure 2 shows the differences between Ising solvers (SA, QA and QS) applied to nBOCS, labelled as nBOCS, nBOCSqa and nBOCSsq, respectively. Although SQ generally has a poor performance in finding a global minimum in a complex landscape of the cost function, there are no clear differences among Ising solvers. This finding suggests that the landscape of the surrogate model is simple. Thus, such a simple Ising solver is enough for the current BBO task.

As discussed in "Original formulation of the integer decomposition" subsection, there are $K! \times 2^K$ exact solutions in the optimisation problem. In other words, any $\boldsymbol{x}$ has $K! \times 2^K - 1$ equivalent vectors, which give the same value of the cost function. These equivalent vectors can be added to the data set and may accelerate the BBO process. Figure 3 shows the results of this data augmentation. The data augmentation (nBOCSa) does not require the additional calculation of the cost function; thus, we do not change the scale of the x-axis. The results clearly show that the data augmentation negatively affects the performance in the later stage, while there is a little advantage in the early stage.

Table 1 shows the total count of finding exact solutions among all ten instances for all algorithms and their variations. Algorithms of nBOCS with SA, QA and SQ perform better than others. RS and nBOCSa cannot find the exact solution in most instances.

To investigate the behaviour of the algorithms, we perform a cluster analysis of the candidate solution for each step. If an algorithm gradually focuses on subspace harbouring a specific exact solution, sampling of candidate solutions will also be biased gradually. We divide the solution space into four domains based on the hierarchical clustering of the exact solutions. Other solutions in the solution space are assigned to the closest exact solution measured using the Hamming distance and then grouped into the four domains. The population of the four domains reflects the sampling bias. Figure 4 shows the population of the candidate solutions among four domains

Instance No.	RS	vBOCS	nBOCS	gBOCS	FMQA08	FMQA12	nBOCSqa	nBOCSsq	nBOCSa
1	0	7	7	2	1	3	9	**11**	0
2	1	6	11	5	6	9	12	**15**	1
3	0	0	**13**	5	6	9	10	2	4
4	1	16	**20**	10	10	13	14	17	0
5	0	0	1	0	3	**4**	2	1	0
6	0	14	12	4	6	4	18	**20**	0
7	0	1	4	6	7	**8**	7	6	0
8	1	**4**	2	2	**4**	1	**4**	1	0
9	1	5	1	3	3	4	**6**	4	0
10	5	21	20	15	20	21	22	**25**	6
Total	9	74	91	52	66	76	**104**	102	11

Table 1. Counts of finding exact soulution per 25 runs. The highest values are in bold.

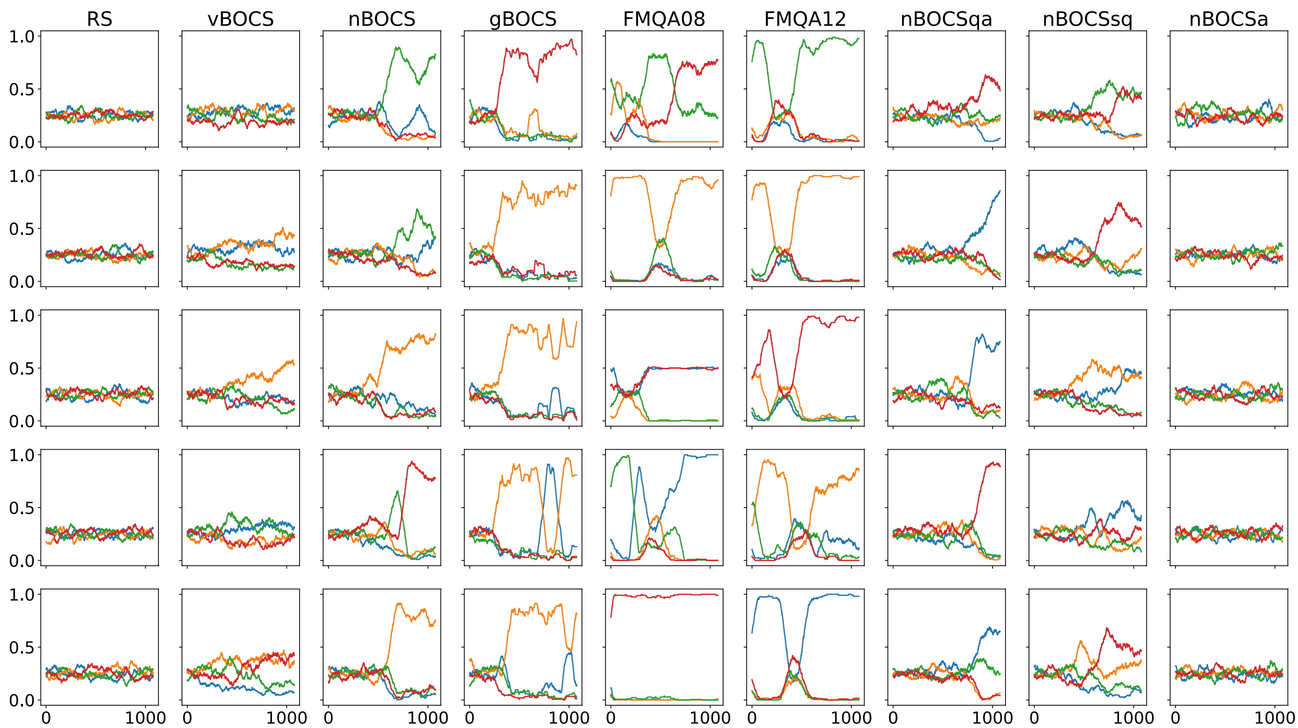

Figure 4. The population of the four domains for five individual runs for various algorithms. The x-axis represents the iteration step.

	RS	vBOCS	nBOCS	gBOCS	FMQA08	FMQA12	nBOCSqa	nBOCSsq	nBOCSa
CPU	0.72	7165.06	55.39	112.39	3711.31	3625.92	241.46	55.94	319.98
QPU	–	–	–	–	–	–	11.60	–	–

Table 2. Average execution time (s) per run.

for individual five runs. The plots are convolved for smoothing (window size, 100). RS and nBOCSa show no trend, while FMQAs start focusing on a domain from an early stage. Other BOCS variants select a domain in the middle of the analysis but continue to explore other domains.

Execution times for all algorithms are shown in Table 2. nBOCSqa takes a longer time than nBOCS due to the overhead to prepare the data matrix to be uploaded to the quantum annealer. The communication time between CPU and QPU is not taken into account. nBOCS is 129 and 67 times faster than vBOCS and FMQA08. The original algorithm and the brute-force search execution times are 0.00096 and 5553.51 s, respectively.

Discussion

We conducted BBO of the integer decomposition, a lossy compression of a matrix. Here, we demonstrated that this MINLP problem is transformed into an NLIP problem. The cost function is pseudo-Boolean because the exploratory variables are binary. We employed BBO algorithms such as BOCS and FMQA to the cost function. These algorithms optimise a surrogate model constructed from the input-output data of the cost function to obtain the next candidate for data acquisition and update the model. This transformation can be generalised to solve MIP problems if the cost function is linear in terms of the real variables.

Among the variations of BOCS and FMQA algorithms, BOCS with normal prior showed the best performance, with an execution time that is one or two digits faster than FMQA or BOCS with horseshoe prior. FMQA improves the solution rapidly at the initial stage, while BOCS slowly but continuously improves and obtains a superior solution. The differences in finding the solution space for these algorithms were analysed through clustering. FMQA tends to focus on a subspace earlier than BOCS. However, once the algorithm is trapped in a local minimum, the algorithm cannot escape from the local minimum because FMQA is deterministic. As BOCS is a randomised algorithm, it takes steps but explores larger space and finds a better solution. In the integer decomposition, the cost function is not an expensive-to-evaluate function; thus, we can conduct enough iterations. The strength of BOCS in the late stage of the iteration is favourable compared to FMQA. A randomised version of FMQA[24] should be studied in the future.

In the current formulation, the solution space is divided into equivalent $K! \times 2^K$ subspaces due to the nature of the problem, where each subspace has the exact solution. We confirmed that the data augmentation for the subspaces did not improve the performance. BOCS and FMQA approximate the cost function in the quadratic function, which means the surrogate model fits well locally, and not globally. Therefore, these algorithms try to focus their sampling on a subspace that harbours one of the exact solutions to the problem. However, data augmentation deals with all the subspaces equally. Thus, it is impossible to sample from biased solution space to improve the model locally around the exact solution. Bias is introduced randomly from the selection of initial data and the following biased candidate selections.

With regard to the choice of Ising solvers as a back-end of a black-box optimiser, there is no significant difference between SA, QA and SQ. This finding is non-trivial because SQ has a poor performance in general. SQ does not explore the solution space globally because the algorithm only accepts the next solution that lowers the cost. This fact leads to the following hypothesis. In the early stage of exploration, the low-quality solutions from the Ising solver are enough to construct a more accurate surrogate model. Then, in the late stage, the surrogate model approximates the cost function around one of the exact solutions as the sampling is biased. Optimisation of the surrogate model by the Ising solver might be relatively easy compared to the explicit form of the original cost function. Therefore, the advantage of QA in BBO is an open question. Further studies are needed to reveal how the approximated landscape of the cost function learned from the acquired data set changes according to the increase of the data.

While the original algorithm cannot find the exact solution, the proposed BBO-based algorithm can find it with a certain probability in the tested problem size ($n = 24$). However, the execution time of the proposed algorithm takes five digits longer than the original. (Brute-force search needs additional two digits.) As the surrogate model is quadratic, BBO needs $O(n^2)$ iterations to estimate the model parameters. If we employ nBOCSsq (normal prior BOCS with SQ), the most expensive calculation in each iteration is matrix inversion $O(n^3)$ for building the surrogate model. Therefore, the algorithm takes $O(n^5)$ calculation time. This scaling may worsen if we choose other Ising solvers (SA and QA). The proposed algorithm has an advantage in solution quality against the original algorithm and calculation time against the brute-force search. However, with the current scaling, the typical use of matrix compression, such as weight matrix in a machine learning task, is not applicable. Figure 4 of the reference[4] shows that FMQA finds meaningful solutions for $n = 12 \sim 50$. Although they fixed the iteration at 2000, more iterations seem needed for large problems. Further investigation is required to accelerate the calculation for handling the typical size of matrices.

In this paper, the algorithm's limitation is that the model is approximated by quadratic form. The COMBO algorithm considers higher-order terms by diffusion kernel of graph representation[25]. Kernel-based algorithms relaxing the binary variables to the continuous ones are also proposed[26,27]. They may work better for problems where the kernel removes essential difficulties in combinatorial optimisation. Once the gradient can be calculated, construction and optimisation of the surrogate model can be accelerated[28]. Comparison with these algorithms will be a future task.

Methods

Shrunk VGG matrix. To test BBO for lossy matrix compression, we prepare matrices by shrinking the VGG matrix, the weight matrix in the convolutional neural network for image recognition. We choose the matrix of the final fully connected layer (4096×1000 matrix). As the matrix is too large to conduct BBO, we reduce the size by keeping the structural information of the matrix as follows: the weight matrix $\boldsymbol{w}_0$ is decomposed by singular value decomposition,

$$\boldsymbol{W}_0 = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \tag{13}$$

where $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{V}$ are $4,096 \times 4,096$, 4096×1000 and 1000×1000 matrices, respectively. Then, we choose eight and a hundred rows/columns from $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively, and eight singular values from $\boldsymbol{\Sigma}$ to construct the shrunk 8×100 matrix.

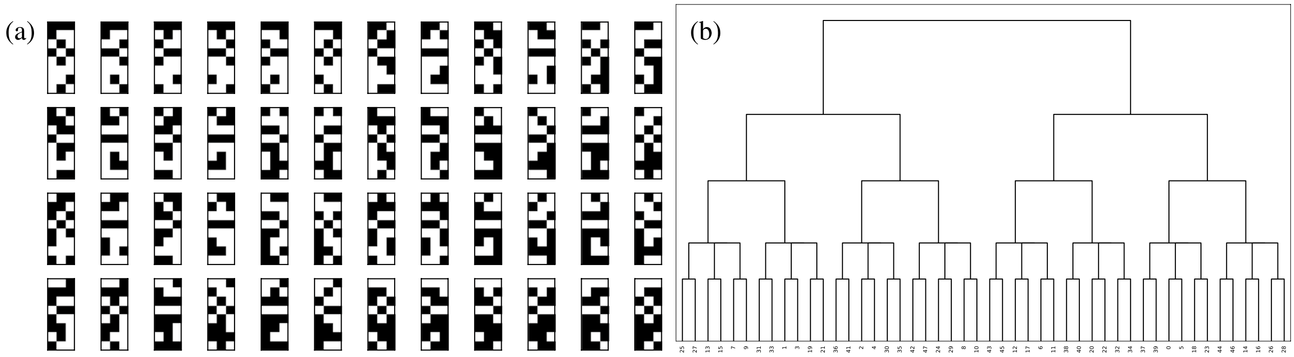

Figure 5. (**a**) 48 exact solutions for the first instance. Each box represents a 3×8 matrix $\boldsymbol{M}$. Black and white pixels represent 1 and -1, respectively. (**b**) clustering results of the 48 exact solutions by the Ward method. Each solution is labelled from 0 to 47 and assigned left-to-right and top-to-bottom in (**a**).

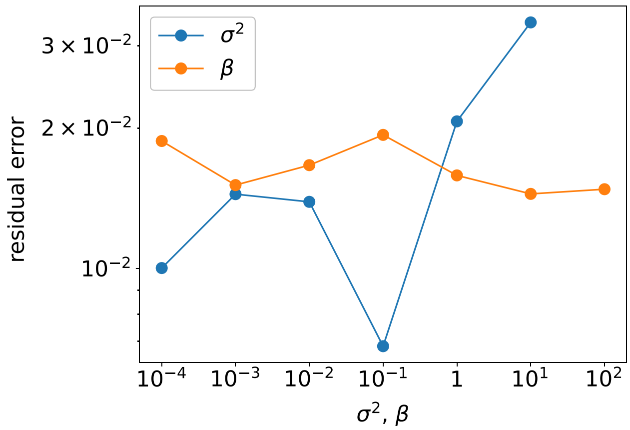

Figure 6. Hyperparameter dependence on the cost.

Exact solutions. The exact solutions in Fig. 5a are obtained by solving Eq. (8). For the current problem size, we can perform a brute-force search for all candidates in the solution space 2^{24}. Each solution is presented in a box of 8×3 pixels. Black and white pixels represent 1 and -1, respectively. Figure 5b shows clustering results of the 48 exact solutions by the Ward method. Similar solutions are grouped, e.g. the second and fourth boxes (labelled 1 and 3 in the cluster) are one of the closest pairs. We can make four groups by choosing an appropriate cut-off value (the height of the tree). These four groups are used for colouring in Fig. 4.

Hyperparameter optimisation. Hyperparameters σ^2 (variance) for nBOCS and β (inverse scale) for gBOCS are optimised for the first instance. For each hyperparameter, grid search 0.0001, 0.001, 0.01, 0.1, 1, 10 and 0.0001, 0.001, 0.01, 0.1, 1, 10, 100 are conducted (Fig. 6). In nBOCS, we select $\sigma^2 = 0.1$, which gives the lowest cost. On the other hand, in gBOCS, we select not $\beta = 10$ but $\beta = 0.001$, as broader prior distribution is preferable for exploring an accurate model, and there is less cost variation among different hyperparameter values.

Results of the other nine instances. Figure 7 shows the results of the other 9 instances used in Table 1.

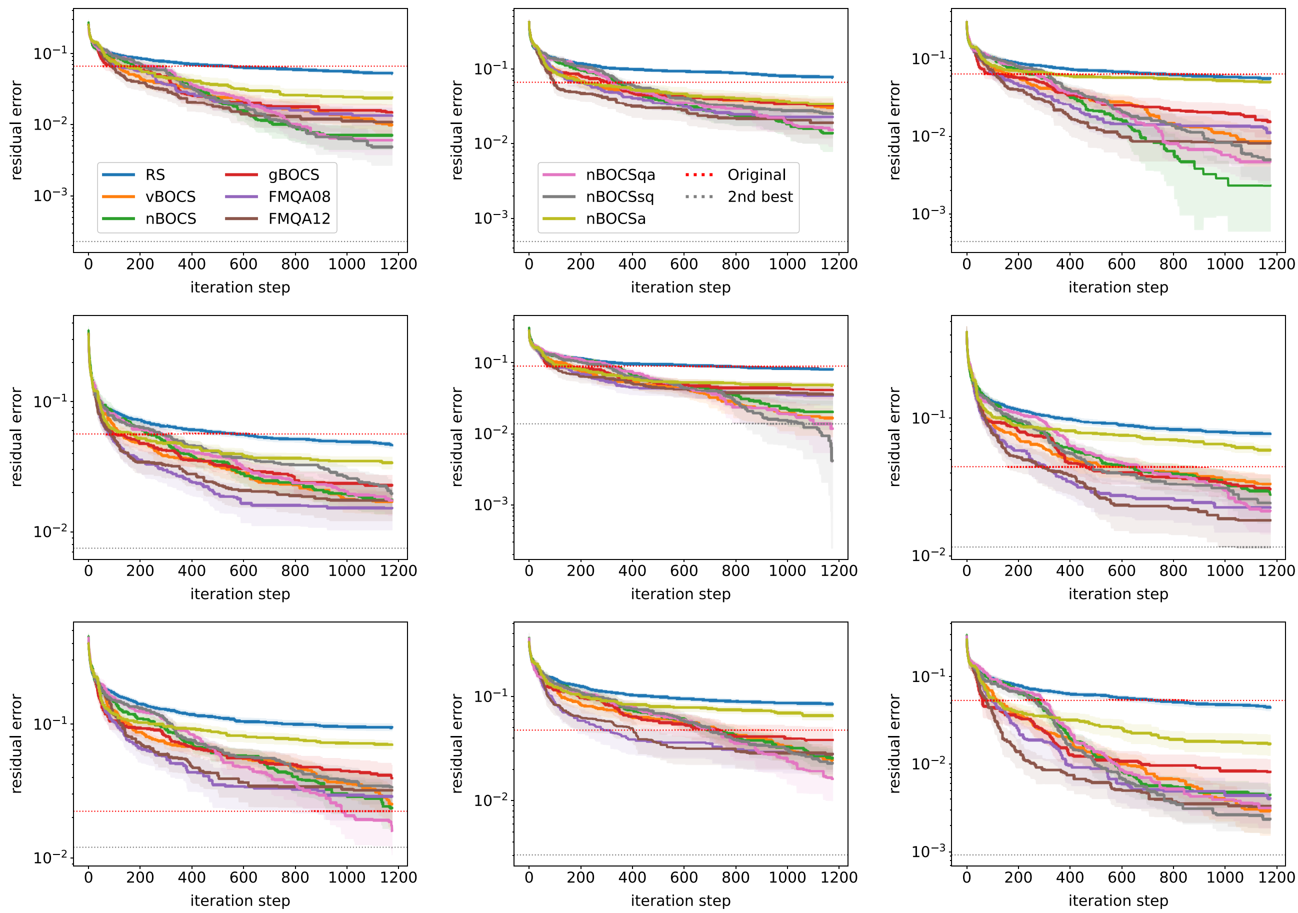

Figure 7. The residual error of lossy compression of the other nine matrices (instances) as a function of the iteration step among various algorithms with 95% confidence intervals. The errors of the nine exact solutions (baseline of the plots) are 0.535, 0.388, 0.509, 0.487, 0.488, 0.379, 0.367, 0.422, and 0.520, respectively.

Data availability

The program code to reproduce the analyses is in Supplementary Information.

Received: 26 April 2022; Accepted: 5 September 2022
Published online: 15 September 2022

References

1. Ambai, M. & Sato, I. SPADE: Scalar product accelerator by integer decomposition for object detection. In *Lecture Notes in Computer Science* Vol. 8693 LNCS. 267–281 (Springer, 2014). https://doi.org/10.1007/978-3-319-10602-1_18.
2. Baptista, R. & Poloczek, M. Bayesian optimization of combinatorial structures. In *35th International Conference on Machine Learning, ICML 2018* **2**, 782–796 (2018).
3. Leprêtre, F., Verel, S., Fonlupt, C. & Marion, V. Walsh functions as surrogate model for pseudo-boolean optimization problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 303–311. https://doi.org/10.1145/3321707.3321800 (ACM, 2019).
4. Kitai, K. *et al.* Designing metamaterials with quantum annealing and factorization machines. *Phys. Rev. Res.* **2**, 013319. https://doi.org/10.1103/PhysRevResearch.2.013319 (2020).
5. Eckart, C. & Young, G. The approximation of one matrix by another of lower rank. *Psychometrika* **1**, 211–218. https://doi.org/10.1007/BF02288367 (1936).
6. Lee, D. D. & Seung, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature* **401**, 788–791. https://doi.org/10.1038/44565 (1999).
7. O'Malley, D., Vesselinov, V. V., Alexandrov, B. S. & Alexandrov, L. B. Nonnegative/binary matrix factorization with a D-wave quantum annealer. *PLoS One* **13**, e0206653. https://doi.org/10.1371/journal.pone.0206653 (2018).
8. Yoon, B., Nguyen, N. T. T., Chang, C. C. & Rrapaj, E. Lossy compression of statistical data using quantum annealer. *Sci. Rep.* **12**, 3814. https://doi.org/10.1038/s41598-022-07539-z (2022).
9. Daxberger, E., Makarova, A., Turchetta, M. & Krause, A. Mixed-variable Bayesian optimization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, vol. 3, 2633–2639, https://doi.org/10.24963/ijcai.2020/365 (International Joint Conferences on Artificial Intelligence Organization, 2020).
10. Koshikawa, A. S., Ohzeki, M., Kadowaki, T. & Tanaka, K. Benchmark test of black-box optimization using D-wave quantum annealer. *J. Phys. Soc. Jpn.* **90**, 064001. https://doi.org/10.7566/JPSJ.90.064001 (2021).
11. Koshikawa, A. S. *et al.* Combinatorial black-box optimization for vehicle design problem.
12. Hatakeyama-Sato, K., Kashikawa, T., Kimura, K. & Oyaizu, K. Tackling the challenge of a huge materials science search space with quantum inspired annealing. *Adv. Intell. Syst.* **3**, 2000209. https://doi.org/10.1002/aisy.202000209 (2021).

13. Gao, Q. *et al.* Quantum-classical computational molecular design of deuterated high-efficiency OLED emitters.
14. Carvalho, C. M., Polson, N. G. & Scott, J. G. The horseshoe estimator for sparse signals. *Biometrika* **97**, 465–480. https://doi.org/10.1093/biomet/asq017 (2010).
15. Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* **25**, 285. https://doi.org/10.2307/2332286 (1933).
16. Rue, H. Fast sampling of Gaussian Markov random fields. *J. R. Stat. Soc. Ser. B (Stat. Methodol.)* **63**, 325–338. https://doi.org/10.1111/1467-9868.00288 (2001).
17. Bhattacharya, A., Chakraborty, A. & Mallick, B. K. Fast sampling with Gaussian scale mixture priors in high-dimensional regression. *Biometrika* **103**, 985–991. https://doi.org/10.1093/BIOMET/ASW042 (2016).
18. Rendle, S. Factorization machines. In *2010 IEEE International Conference on Data Mining*, 995–1000. https://doi.org/10.1109/ICDM.2010.127 (IEEE, 2010).
19. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680. https://doi.org/10.1126/science.220.4598.671 (1983).
20. Geman, S. & Geman, D. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 721–741. https://doi.org/10.1109/TPAMI.1984.4767596 (1984).
21. Kadowaki, T. & Nishimori, H. Quantum annealing in the transverse Ising model. *Phys. Rev. E* **58**, 5355–5363. https://doi.org/10.1103/PhysRevE.58.5355 (1998).
22. Morita, S. & Nishimori, H. Convergence of quantum annealing with real-time Schrödinger dynamics. *J. Phys. Soc. Jpn.* **76**, 064002. https://doi.org/10.1143/JPSJ.76.064002 (2007).
23. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015-Conference Track Proceedings* (2015).
24. Matsumori, T., Taki, M. & Kadowaki, T. Application of QUBO solver using black-box optimization to structural design for resonance avoidance. *Sci. Rep.* **12**, 12143. https://doi.org/10.1038/s41598-022-16149-8 (2022).
25. Oh, C., Tomczak, J. M., Gavves, E. & Welling, M. Combinatorial Bayesian optimization using the graph cartesian product. *Adv. Neural Inf. Process. Syst.* **32**, 25. https://doi.org/10.1109/TPAMI.1984.4767596 (2019).
26. Buathong, P., Ginsbourger, D. & Krityakierne, T. Kernels over sets of finite sets using RKHS embeddings, with application to Bayesian (combinatorial) optimization. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics* **108**, 2731–2741 (2020).
27. Deshwal, A. & Doppa, J. R. Combining latent space and structured kernels for Bayesian optimization over combinatorial spaces. *Adv. Neural. Inf. Process. Syst.* **34**, 8185–8200 (2021).
28. Wu, T. C., Flam-Shepherd, D. & Aspuru-Guzik, A. Bayesian variational optimization for combinatorial spaces.

Acknowledgements

T.K. thanks Ikuro Sato, Kentaro Matsuura and Takashi Imoto for useful discussions.

Author contributions

T.K. and M.A. conceived the concept. T.K. conducted the experiments and analysed the results. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-022-19763-8.

Correspondence and requests for materials should be addressed to T.K.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.