

Machine Learning-assisted High-speed Combinatorial Optimization with Ising Machines for Dynamically Changing Problems

Corresponding Author: Mr Yohei Hamakawa

This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.

Version 0:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

This paper describes a method to combine Ising machines using a machine learning parameter tuning method. The Ising machine is a state-of-the-art technique and the direction using plural machines and machine learning is promising, so the proposal of this paper is interesting for many readers. To clarify the achievement of this paper, some additional discussions are necessary. The details are as follows.

1. In Fig 2a, distribution of J-matrix is indicated, and the author mentions that "The reason for the limited variation is that ...". However, as the TSP, many problems in the real world do not have such distribution, I feel. The results in Fig. 2a are true but the author should carefully discuss the generalization of the problems.
2. In the text, there are a lot of down arrows, but I don't understand what they mean.
3. The use of FPGA-base Ising machine and CPU-base Ising machine may achieve the good effect as in Fig. 4f. However, I cannot understand whether the combination is truly effective or not. To clarify the performances of FPGA-base and CPU base Ising machine will support the understanding the effectiveness.
4. In Fig. 6, the reason why the parameter tuning considering some parameters at the same time can be understood, however, I cannot understand why machine learning is necessary. Only from Fig. 6, rule-base parameter tuning can be used. The author should clarify the effect of machine learning.
5. At the last part of the Discussion, the author mentioned that the proposed method can be used for various problems. However, only from the results, it cannot be verified. If the author claims this content, the author should carefully discuss the effect of the proposed method on the various problems.

Reviewer #2

(Remarks to the Author)

This paper proposes a combinatorial optimization framework based on embedded Ising machines, enhanced by machine learning-based parameter estimation. The authors aim to address the typical limitations of Ising machines, such as high system latency due to model transfer or cloud access and the need for runtime parameter tuning. While the work integrates simulated bifurcation-based Ising machines and learning-based configuration, several key issues significantly limit its technical contribution and practical viability.

Limited Problem Scope and Evaluation

Although the authors claim their framework can handle a wide variety of combinatorial optimization problems (COPs), the experimental validation is limited solely to time division multiple access (TDMA) scheduling. This narrow focus undermines the claimed generality.

Questionable Support for Dynamically Changing Problems

The authors emphasize the capability to handle dynamically changing problems. However, the encoding and processing flow remains sequential, with no clear demonstration of dynamic runtime adaptability. Furthermore, the justification for latency-critical performance is generic and applies to all COPs, not specifically to the demonstrated use case.

Lack of Novel Algorithmic Contribution

The low-latency computation relies on the ballistic simulated bifurcation algorithm, which has already been well studied. The manuscript does not present substantial innovation in this area. Additionally, the paper lacks a detailed analysis of the overhead and scalability of encoding and decoding the J-matrix. The compression benefits are problem-dependent and are not quantitatively substantiated.

Insufficient Details on Learning-Based Parameter Estimation

The proposed use of a machine learning model for parameter estimation is potentially interesting, but critical implementation aspects—such as model scale, latency, training overhead, and hardware cost—are not discussed. In practice, storing and accessing a large number of Ising parameters (especially multiple models) becomes infeasible for larger problems due to memory constraints, which the paper fails to address.

Lack of Experimental Clarity and End-to-End Demonstration

The experimental setup is unclear. If the approach is truly end-to-end, the system architecture and deployment platform (e.g., CPU+FPGA) must be explicitly described and justified. If the work is based primarily on simulation, the claims regarding embedded or edge deployment are misleading. Overall, there is a lack of convincing demonstration of the full system in operation.

Version 1:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

I have checked the author's response to Reviewer 1's comments, and it seems to have been well-revised. I would like to check only one point.

I understood the usefulness of machine learning from the added section. However, although "Moreover, performing a grid search (or other parameter search) like that in Fig.5 for every problem instance is impractical from a computational time perspective" is written on lines 484 and 485, the actual time taken to use the grid search and the machine learning is not shown. Could you please show the time? Or, if it's written somewhere, that would be fine. I'm sorry I couldn't find it.

(Remarks on code availability)

I have checked the author's response to Reviewer 1's comments, and it seems to have been well-revised. I would like to check only one point. The details are in the comments for the author.

Reviewer #2

(Remarks to the Author)

Thanks for the authors' efforts, but still, there are many points unclear

1. While the authors demonstrate significant speed advantages over NetworkX and OR-Tools, the choice of baselines lacks rigor. NetworkX is a general-purpose graph library not optimized for high-performance computing, and OR-Tools is an exact solver, which inherently suffers from scalability issues. The absence of comparisons against state-of-the-art heuristic MIS solvers (e.g., specialized GPU-based algorithms or advanced metaheuristics, and please, include high-quality references within 3 years) undermines the claim of superior performance.

2. The authors address parameter dependencies in the SB algorithm through a learned ratio-based approach. While empirically effective, this method lacks theoretical grounding or mechanistic explanations. The rationale behind why such parameterization mitigates dependency issues remains unclear, making it appear as an engineering workaround rather than a principled solution. A deeper analysis of the parameter interactions and their impact on algorithmic stability would strengthen this contribution.

3. The reported 5500× speedup is impressive but ambiguously attributed. The paper does not include ablation studies to disentangle the contributions of FPGA-based parallelization from those of the SB algorithm and the indexing architecture. It is plausible that the hardware acceleration dominates the performance gains, which would diminish the perceived novelty of the algorithmic components. Without such analysis, the claims of methodological advancement are weakly supported.

4. The discussion section lists potential applications in finance, automotive, and robotics without substantive evidence or domain-specific analysis. These fields often involve complex constraints, dynamic environments, and problem structures distinct from wireless network scheduling. The authors provide no justification for how their framework adapts to such diverse challenges, rendering the "one-size-fits-all" claim speculative and unsupported.

5. The motivation is still unclear, as I said, latency-critical performance is generic and applies to all COPs, the authors should prove clear definition of dynamically changing COPs. The authors should explicitly define: (1) the temporal

characteristics of problem changes, (2) the latency requirements for TDMA scheduling in concrete terms (e.g., 100ms for real-time control with solid reference), and (3) the performance gap between existing methods and requirements. Most importantly, the paper should quantify how much the proposed method closes this gap and how much does state-of-the-art method close this gap (with solid references).

(Remarks on code availability)

Reviewer #3

(Remarks to the Author)

The article provides various methods for solving time division multiple access (TDMS) problems using Ising machines, namely FPGA-based Ising machines. These methods include compressing the input data for FPGA-chips using a technique authors call indexing fast architecture, using XGBoost to predict model for the relevant parameters for the Ising machine and also using machine learning to select which Ising machine implementation would be fastest depending on the problem at hand.

The significance of the article is optimizing their approach to real-time solving of the TDMS problem using Ising machines. The article focuses on a single use case, but the approach can generalize to other contexts, if they fulfill the certain properties listed in the article. The approach compares favourably to quantum annealing research, but the comparisons to other FPGA-based approaches is lacking.

When comparing their work on other approaches, they only consider what sort of advantages the methods presented have compared to quantum annealing based solutions. It would be interesting to also hear how could the approaches of this article be used alongside quantum annealing, as none of the three methods seem incompatible with quantum annealing based solutions.

The key-plots of important results (For example 4-e, 5, 7. c and 7.d) are scattered across the text. It is difficult to visualise what are the advantages of the presented techniques over not using them. Also the article never compares their custom simulated bifurcation algorithm to baseline simulated bifurcation algorithm, but only to exact and simulated annealing solvers. Other than that, the methodology of the article is sound. The article provides enough details to replicate the results of the research.

More detailed comments below:

1. The optimization technique referred to as indexing fast architecture, is a well known optimization technique called dictionary coding, and should not be considered a novel concept, though we do not know whether FPGA-models have used this encoding previously to improve data transmission times. The authors correctly observe that this way of encoding is very efficient in the use case, as the number of distinct values in the J is limited, which means that using this optimization method in this context might be novel, while the encoding algorithm itself is not.

2. Authors provide a list of relevant research at the end of the results section (Shouldn't this be earlier?) and criticizes that the quantum annealing approaches only use annealing time to benchmark their approaches and forego job queuing and offloading overhead completely. This is certainly true, but it would also be a bit misleading to compare computational overhead of integrated devices and experimental hardware that is currently only accessible to researchers through complicated cloud services. If quantum utility could be achieved here by using quantum annealers, some extra layers of classical overhead would probably be removed very quickly. Though quantum annealer specific overheads, like minor embedding overhead, might be the too large in this use case, as computing a new embedding map for every single problem can be quite expensive.

3. Furthermore the section seems quite defensive against quantum annealing, as the none of the techniques presented in this article are incompatible with quantum annealing approaches, but rather would be complementary. Authors are recommended to edit the text to reflect this.

4. The figure 4.c has a typo in the image: Iteretion → Iteration

5. The results section only shows how all the techniques operate with one another, but not how the distinct pieces perform on their own.

6. Currently the results that reflect the impact of the compression scheme and differences between CPU and FPGA-implementations are provided in the supplementary information, while being quite important for the results. This information should be brought into the results section properly.

7. Figure 3 only compares the custom SB algorithm to the NetworkX and attaining the exact solution. Should there also be a standard SB algorithm as a comparison point so the effect of the optimization techniques and hyperparameter model can be seen properly.

(Remarks on code availability)

Reviewer #4

(Remarks to the Author)

I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

(Remarks on code availability)

Version 2:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

I have reviewed the authors' responses to Reviewer 1's comments and find that they have been addressed satisfactorily.

(Remarks on code availability)

Reviewer #2

(Remarks to the Author)

The authors addressed my concerns.

(Remarks on code availability)

Reviewer #3

(Remarks to the Author)

I have checked the authors' edits to the manuscript and their answers to the reviewer comments. They have adequately addressed the article's key issues.

The only issue I would like to have corrected is:

1. The paragraph for TDMA scheduling on rows 395-409 should be structured better, as the added sentence about quantum annealing overheads is a bit abrupt and does not fit the logical flow of the text. Content itself is fine.

(Remarks on code availability)

Reviewer #4

(Remarks to the Author)

I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

(Remarks on code availability)

Code is not publicly available.

Open Access This Peer Review File is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

In cases where reviewers are anonymous, credit should be given to 'Anonymous Referee' and the source.

The images or other third party material in this Peer Review File are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this license, visit <https://creativecommons.org/licenses/by/4.0/>

Response to Reviewers:

We appreciate the time and effort that the Reviewers dedicated to reviewing the manuscript. We received many fruitful suggestions and comments from our two reviewers, based on their deep understanding, which led us to add substantial discussion and additional work. This has significantly improved the manuscript, making it much stronger, more readable, and accessible to the broad readership of *Nature Communications*.

Summary of Main Revisions:

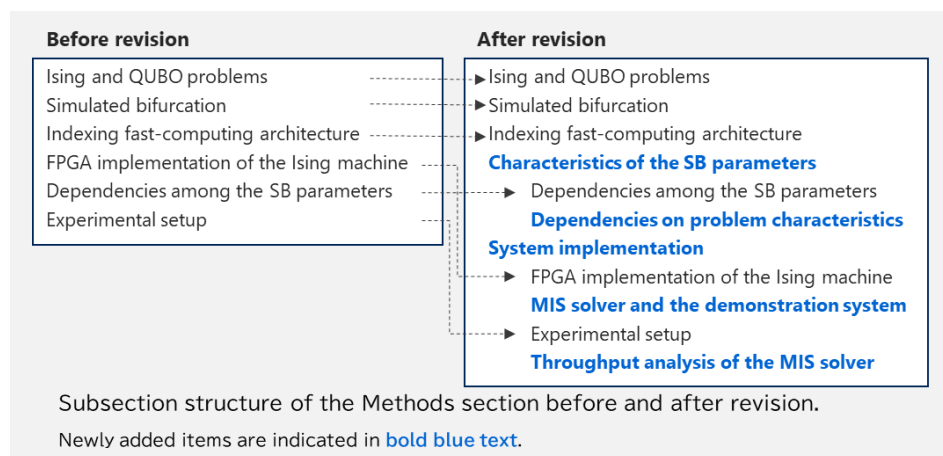
We sincerely appreciate the reviewers' constructive comments. In response, we have made the following major revisions to improve the clarity, technical depth, and overall structure of the manuscript:

- 1) **Introduction and Discussion** sections have been revised to more clearly articulate the problem setting, research objective, and the proposed method.
- 2) The scope and significance of the indexing fast-computational architecture have been reinforced. Additional descriptions were added to the corresponding section, and Table 1 now includes examples of Ising model formulations where our acceleration method is particularly effective.
- 3) At the end of the **Demonstration** section, we added examples of existing dynamic changing problem applications and discussed how our proposed method can contribute to them, thereby strengthening our claim that the method is applicable to a wide range of problems.
- 4) The **Methods** section has been restructured to improve clarity and logical flow. As illustrated in the figure below, the previous subsection structure has been reorganized and expanded. Specifically, we introduced three new subsections:

- *Dependencies on Problem Characteristics*
- *MIS Solver and the Demonstration System*
- *Throughput Analysis of the MIS Solver*

These are grouped under newly added higher-level subsections:

- *Characteristics of the SB Parameters*
- *System Implementation*



- 5) In the new subsection *Dependencies on Problem Characteristics*, we discuss the necessity of machine learning in the proposed framework.
- 6) The new subsection *MIS Solver and the Demonstration System* describes the hardware/software implementation and configuration of the demonstration system and MIS solver.
- 7) The new subsection *Throughput Analysis of the MIS Solver* presents a detailed breakdown of processing time, including system-wide latency, prediction time by the ML model, encoding time, Ising machine execution time, and offloading time. Performance comparisons between CPU-based and FPGA-based implementations are also provided.
- 8) While the qualitative discussion remains unchanged, experimental results have been updated (mainly Fig. 3d, Fig. 4e, Fig. 4f, and the ML-based machine selector graphs in Fig. 1 and Fig. 3c). These updates reflect improvements made during the detailed throughput analysis (especially in response to Reviewer #1 - Comment #3 and Reviewer #2 - Comment #3), including parallelization using multi-core processors and acceleration of post-processing, which led to enhanced solver performance.
- 9) To support the above-mentioned points, new Tables, Figures, and Algorithms were added: Table 1 and Table 2, as well as Figure 5b and Figure 7.

Point-by-point responses to Reviewers' comments:

Responses to Reviewer #1: (Bold: Reviewers' comments)

This paper describes a method to combine Ising machines using a machine learning parameter tuning method. The Ising machine is a state-of-the-art technique and the direction using plural machines and machine learning is promising, so the proposal of this paper is interesting for many readers. To clarify the achievement of this paper, some additional discussions are necessary. The details are as follows.

Author response:

We sincerely thank Reviewer #1 for the insightful and constructive feedback. Your comments helped us clarify the generalizability of our approach (Comment #1), improve the notation and readability (Comment #2), and deepen the analysis of hardware performance and machine learning necessity (Comments #3 and #4). We also revised our claims regarding application scope to better reflect the supporting evidence (Comment #5). We have carefully addressed each point and made corresponding revisions to the manuscript, figures, and tables. Below, we provide detailed point-by-point responses.

Reviewer#1-Comment#1:

1. In. Fig 2a, distribution of J-matrix is indicated, and the author mentions that "The reason for the limited variation is that ...". However, as the TSP, many problems in the real world do not have such distribution, I feel. The results in Fig. 2a are true but the author should carefully discuss the generalization of the problems.

Author response:

Thank you for your valuable comment. As you pointed out, the description in Fig. 2a may have given the impression

that a limited variation in the distribution is common in general. To address this, we have revised the explanation in Fig. 2a (Lines 130–137). In the revised text, we acknowledge that general Ising models can represent a wide range of interactions. Nevertheless, when combinatorial optimization problems are formulated as Ising models, the numerical patterns that appear in the quadratic (off-diagonal) terms tend to be limited. This characteristic supports the significance of developing specialized acceleration methods, such as the one proposed in this study. Additionally, to supplement this point, we have added examples of problems with a small number of patterns in the newly introduced Table 1.

Reviewer#1-Comment#2:

2. In the text, there are a lot of down arrows, but I don't understand what they mean.

Author response:

Thank you for pointing this out. The downward arrows in the text were originally intended to indicate that a lower value is preferable for the corresponding metric. However, as this notation is not commonly used, we have revised the text to explicitly state “lower is better” or “larger is better” wherever appropriate.

Reviewer#1-Comments #3:

3. The use of FPGA-base Ising machine and CPU-base Ising machine may achieve the good effect as in Fig. 4f. However, I cannot understand whether the combination is truly effective or not. To clarify the performances of FPGA-base and CPU base Ising machine will support the understanding the effectiveness.

Author response:

Thank you for your insightful comment and suggestion. In response, we have added a new subsection titled “System-wide latency analysis of the MIS solver” in the Methods section (Lines 558–586) with a new Table (Table 2) and a new Figure (Figure 7), where we present a performance comparison between FPGA-based and CPU-based Ising machines for solving the MIS problem. The results are shown in Fig. 7d and Fig. 7e.

Our analysis indicates that while computation time increases with the number of steps and the size of $|V|$, the rate of increase is smaller for the FPGA-based implementation, allowing it to solve larger problems more efficiently. On the other hand, the FPGA-based solver incurs a constant offloading overhead (approximately 200 microsecond, as shown in Table 2), which makes the CPU-based solver faster for problems with smaller step counts or $|V|$ sizes.

These observations support the idea that using one of FPGA-based and CPU-based Ising machines depending on the characteristics of the problem can be an effective strategy. Please see the new subsection for further details.

Reviewer#1-Comment#4:

4. In Fig. 6, the reason why the parameter tuning considering some parameters at the same time can be understood, however, I cannot understand why machine learning is necessary. Only from Fig. 6, rule-base parameter tuning can be used. The author should clarify the effect of machine learning.

Author response:

Thank you for your comment. In response, we have expanded Fig. 5 (formerly Fig. 6) by adding a new panel, Fig. 5b, and introduced a new subsection in the Methods section titled “Dependencies on problem characteristics” (Lines 474–490) to explain the necessity of machine learning.

In this subsection, we show that even when the problem size ($|V| = 200$) and certain parameters (e.g., step = 200) are fixed, the optimal parameter ranges can vary significantly depending on the characteristics of the input graph, such as edge density and connectivity. At present, we do not have a strict rule-based formulation that can fully capture these dependencies. Machine learning offers the advantage of modeling such complex relationships using large datasets, which is why we have adopted this approach in our study. Please see the new subsection for further details.

Reviewer#1-Comment#5:

5. At the last part of the Discussion, the author mentioned that the proposed method can be used for various problems. However, only from the results, it cannot be verified. If the author claims this content, the author should carefully discuss the effect of the proposed method on the various problems.

Author response:

Thank you for your comment. To avoid potential overstatement, we have revised the wording from “various applications” to “a wide range of applications” and added supporting discussion at the end of Section 4 (Lines 318–354).

In the revised Section 4, we present several examples of existing applications involving dynamic changing problems (TDMA Scheduling, Wireless Interface Network Optimization, Resource Allocation for Small Star Networks, In-Vehicle Multi-Object Tracking, and Other Wireless Communication Problems). For each case, we identify aspects that were previously difficult to address (or not discussed in prior work), and explain how the proposed method could potentially overcome these limitations. This additional discussion strengthens our claim that the proposed approach can be applied to a wide range of problems. Please see the newly added descriptions (Lines 318–354) for further details.

=====

Responses to Reviewer #2: (Bold: Reviewers’ comments)

This paper proposes a combinatorial optimization framework based on embedded Ising machines, enhanced by machine learning-based parameter estimation. The authors aim to address the typical limitations of Ising machines, such as high system latency due to model transfer or cloud access and the need for runtime parameter tuning. While the work integrates simulated bifurcation-based Ising machines and learning-based configuration, several key issues significantly limit its technical contribution and practical viability.

Author response:

We sincerely thank Reviewer #2 for the detailed and constructive feedback. Your comments helped us clarify the scope and generality of our framework (Comment #1), refine our definition and demonstration of dynamic adaptability

(Comment #2), and provide a more thorough analysis of encoding overhead and hardware performance (Comment #3). We also expanded our explanation of the machine learning components (Comment #4) and clarified the system architecture and deployment platform to support our end-to-end claims (Comment #5). We have carefully addressed each point and revised the manuscript accordingly. Below, we provide detailed point-by-point responses.

Reviewer#2-Comment#1:

Limited Problem Scope and Evaluation

Although the authors claim their framework can handle a wide variety of combinatorial optimization problems (COPs), the experimental validation is limited solely to time division multiple access (TDMA) scheduling. This narrow focus undermines the claimed generality.

Author response:

Thank you for your comment. We have addressed this point by adding further discussion at the end of Section 4 (Lines 318–354).

In this revised section, we present several examples of existing applications involving dynamic changing problems (TDMA Scheduling, Wireless Interface Network Optimization, Resource Allocation for Small Star Networks, In-Vehicle Multi-Object Tracking, and Other Wireless Communication Problems). For each case, we identify aspects that were previously difficult to address (or not discussed in prior work), and explain how the proposed method could potentially overcome these limitations. This additional discussion reinforces our claim that the proposed framework can be applied to many combinatorial optimization problems beyond TDMA scheduling. Please see the newly added descriptions (Lines 318–354) for further details.

Additionally, we added a new Table (Table 1) to list fundamental combinatorial optimization problems to which our proposed method can be applied. Since these problems are very fundamental, the Table 1 implies the potential generality (wide applicability) of the proposed methodology.

Reviewer#2-Comment#2:

Questionable Support for Dynamically Changing Problems

The authors emphasize the capability to handle dynamically changing problems. However, the encoding and processing flow remains sequential, with no clear demonstration of dynamic runtime adaptability. Furthermore, the justification for latency-critical performance is generic and applies to all COPs, not specifically to the demonstrated use case.

Author response:

Thank you for your comment. We define the requirements for dynamic runtime adaptability as follows:

- Selecting parameters based on problem characteristics without incurring overhead
- Achieving high system-wide computational efficiency

This definition has been explicitly stated in the Introduction (Lines 98–102). To support our claim that the proposed method satisfies these requirements, we have added a detailed analysis of computation time (see newly added Fig. 7c

and Table 2), demonstrating that the encoding time does not become a bottleneck in the overall system latency. These additions reinforce the argument that our method possesses runtime adaptability for dynamic problems and meets latency-critical performance requirements in a meaningful and problem-specific manner. Please see the new subsection “MIS solver and the demonstration system” (Lines 506–537) for further details on time-component analysis.

Reviewer#2-Comment#3:

Lack of Novel Algorithmic Contribution

The low-latency computation relies on the ballistic simulated bifurcation algorithm, which has already been well studied. The manuscript does not present substantial innovation in this area. Additionally, the paper lacks a detailed analysis of the overhead and scalability of encoding and decoding the J -matrix. The compression benefits are problem-dependent and are not quantitatively substantiated.

Author response:

Thank you for your comment. While the ballistic simulated bifurcation (bSB) algorithm itself has been well studied (see Subsection “Simulated bifurcation”, Lines 422-439), the low-latency computation enabled by our proposed indexing fast-computation architecture (see the Main text, Lines 73-80, and the Subsection “Indexing fast-computing architecture, Lines 440-447, with Algorithms 1, 2, and 3) is new and is not limited to bSB. Although there are constraints on the dynamic range and precision of the variable $\mathbf{x}$, the technique can be applicable to general MAC operations commonly used across various domains. To clarify this point, we have added an explanation in Section 2.1 (Lines 174–177).

Regarding the overhead and scalability of encoding the J -matrix, we conducted a detailed analysis using the case without the indexing fast-computation architecture as a baseline. A new subsection titled “System-wide latency analysis of the MIS solver” (Lines 558-586) has been added, and the results are presented in Fig. 7c (along with 7a and 7b) and Table 2. For example, Table 2 shows that the encoding time increases with $|V|$, reaching 514 microsecond for $|V| = 1024$ and 1913 microsecond for $|V| = 2048$, indicating a quadratic growth. However, even in such cases, the proportion of encoding time within the total system latency remains relatively small.

Moreover, we demonstrate that although the encoding time slightly increases, the resulting gain from the significant reduction in Ising machine execution time—approximately one-third in the given example—far outweighs this overhead, leading to a substantial decrease in overall system latency when applying the indexing fast-computation architecture. Note that the decoding time is included in the Ising machine execution time and is implemented via table lookup within MAC operations, making it negligible (on the order of nanoseconds).

Finally, to address the lack of explanation regarding the algorithmic novelty, we have added a comparative discussion in Section 4 (Lines 367–374), highlighting the limitations of prior approaches and the contributions of our proposed method.

Reviewer#2-Comment#4:

Insufficient Details on Learning-Based Parameter Estimation

The proposed use of a machine learning model for parameter estimation is potentially interesting, but critical implementation aspects—such as model scale, latency, training overhead, and hardware cost—are not discussed. In practice, storing and accessing a large number of Ising parameters (especially multiple models) becomes infeasible for larger problems due to memory constraints, which the paper fails to address.

Author response:

Thank you for your comment. Regarding the scale of the machine learning models, the parameter estimator, as shown in Fig. 3b, is a lightweight regression model designed to estimate the Ising machine parameters (c , dt , $step$). The model is compact and does not require large computational resources. The machine selector is even smaller in scale. To clarify this, we have added an explanation in the subsection “MIS solver and the demonstration system” (Lines 524–528). As written in the subsection, these models are implemented with compact sizes of 160 KB and 916 bytes, respectively.

As for latency, we have addressed this in response to Comment #3, showing that the overhead introduced by the estimator is minimal. It is important to note that the focus of this study is on runtime acceleration; therefore, the preparation of training data and the training process itself, which are performed offline (Lines 83, Lines 485-490), are outside the scope of this work and not discussed in the manuscript.

Concerning memory constraints, our approach does not rely on storing multiple Ising models (multiple J -matrixes and h -vectors). Instead, it uses a lightweight model to estimate parameters (c , dt , $step$) based on problem features, which keeps the memory footprint low even for large-scale problems.

Reviewer#2-Comment#5:

Lack of Experimental Clarity and End-to-End Demonstration

The experimental setup is unclear. If the approach is truly end-to-end, the system architecture and deployment platform (e.g., CPU+FPGA) must be explicitly described and justified. If the work is based primarily on simulation, the claims regarding embedded or edge deployment are misleading. Overall, there is a lack of convincing demonstration of the full system in operation.

Author response:

Thank you for your comment. The system architecture used in our experiments is thoroughly illustrated in new Figures (Fig. 7a and Fig. 7b). As shown in these figures, both the TDMA scheduling demonstration and the MIS solver are implemented as end-to-end systems, covering the full processing flow from input to output. These systems are deployed not only on the host CPU but also on actual hardware (FPGA), demonstrating a fully integrated and operational setup.

While the input graphs for the tree topology and the corresponding interference graph are generated via simulation, the TDMA scheduler and MIS solver are implemented as software modules that can be deployed on either CPU or FPGA platforms, depending on the target environment. This modular design allows for straightforward integration into embedded or edge systems. We have added clarifying details in the Methods section to better explain this aspect (Lines 529-534).

=====

Yours sincerely,

Yohei Hamakawa, Tomoya Kashimata, Masaya Yamasaki, Kosuke Tatsumura

Yohei Hamakawa

Corporate Research and Development Center, Toshiba Corporation

1 Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan

E-mail: yohei.hamakawa@toshiba.co.jp Tel: +81-50-3148-5260

Response to Reviewers:

We sincerely appreciate the time and effort that the reviewers dedicated to the evaluation of our manuscript. We are grateful for the many constructive and insightful comments, which have helped us to substantially improve the manuscript in terms of clarity, technical grounding, organization, and scope. In response to the reviewers' suggestions, we have revised the manuscript extensively, including clarifying the definition and motivation of dynamically changing combinatorial optimization problems, strengthening the experimental evaluation, adding more explicit analyses of the proposed framework, and refining the discussion of applicability and positioning with respect to related approaches. We believe that these revisions have significantly strengthened the manuscript and made its contributions clearer and more accessible to the broad readership of *Nature Communications*.

Summary of Main Revisions:

We sincerely appreciate the reviewers' constructive comments. In response, we have made the following major revisions to improve the clarity, technical depth, and overall structure of the manuscript:

- 1) We revised the Introduction to more clearly define dynamically changing combinatorial optimization problems and to explain their temporal characteristics across application domains. In particular, we clarified that the target problem class is characterized by repeatedly changing and sequentially solved optimization instances, rather than by low-latency requirements alone.
- 2) We added a new subsection, "Evaluation on the MIS problem" (Section 2.3), in the Results section. This subsection reorganizes and expands the evaluation of the proposed framework, including system-wide latency breakdowns, comparisons among Baseline / Proposed (Generic) / Proposed (Optimized), CPU-versus-FPGA comparisons, and solver-level comparisons against SB (baseline), NetworkX, OR-Tools, and ReduMIS.
- 3) We strengthened the analysis of the proposed framework by clarifying the individual contributions of its main components. In particular, the revised manuscript now more clearly separates the effects of indexing-based acceleration, FPGA/CPU implementation differences, problem-specific optimization, and parameter-estimation-based adaptation.
- 4) We revised the Methods section to provide a more principled explanation of the dependencies among the SB parameters. By referring to the ballistic SB update equation, we now explain the qualitative relationships between c and dt , and between $step$ and dt , and clarify the rationale for the ratio-based learning strategy.
- 5) We added quantitative information on the computational cost of grid search used for training-data preparation. The revised manuscript now states the offline search settings and their runtime per instance, and also explains why per-instance online grid search is impractical compared with the trained parameter estimator.
- 6) We strengthened the description and positioning of the indexing fast-computation architecture. In particular, we now clarify that the indexing scheme is conceptually related to dictionary-based encoding, and we explain more explicitly that the methodological contribution lies in its selective application to the Ising-model structure and in the encoded-state acceleration of MAC operations.
- 7) We revised the Demonstration and Discussion sections to avoid overgeneralized claims of applicability. The revised manuscript now more clearly states that the proposed framework is intended for repeatedly solved QUBO/Ising-formulated subproblems under changing conditions, and explicitly delimits problems that are

outside the direct scope of the present work.

- 8) We also revised the text to clarify the relationship between the proposed framework and quantum annealing.

The manuscript now explicitly states that the proposed approach is not intended to replace quantum annealing methods, but rather to complement them within a system-aware and low-latency optimization framework.

Point-by-point responses to Reviewers' comments: (Bold: Reviewers' comments)

Reviewer #1 (Remarks to the Author):

I have checked the author's response to Reviewer 1's comments, and it seems to have been well-revised. I would like to check only one point.

I understood the usefulness of machine learning from the added section. However, although "Moreover, performing a grid search (or other parameter search) like that in Fig.5 for every problem instance is impractical from a computational time perspective" is written on lines 484 and 485, the actual time taken to use the grid search and the machine learning is not shown. Could you please show the time? Or, if it's written somewhere, that would be fine. I'm sorry I couldn't find it.

Author response:

Thank you very much for your careful reading and for pointing out this important issue. We agree that the previous version did not explicitly quantify the actual time required for grid search, although the usefulness of the machine-learning-based parameter estimator was discussed

To address this point, we have added a quantitative explanation in the Methods section, "Dependencies on problem characteristic" (Lines 588–594). Specifically, we now state that the offline grid search used for training-data preparation sampled 11 values of c , 10 values of dt , and 5 values of $step$ (550 parameter combinations in total), with 100 shots executed for each combination, and that this required approximately 500 s per graph instance under the experimental environment used in this study. We further added that even a much coarser online search, for example using 3 values for each parameter (27 combinations in total) and 4 shots per combination, would still require on the order of 1 s per instance.

In addition, the online overhead of the trained machine-learning-based parameter estimation is already reflected in the "Evaluation on the MIS problem" section (Line 291). There, the combined overhead of feature extraction and estimation is reported to be approximately 2 ms for $|V| = 1024$ and 2.8 ms for $|V| = 2048$, indicating that the estimator itself is not a performance bottleneck.

We believe that these additions clarify both (i) the computational cost of direct parameter search and (ii) the practical advantage of replacing per-instance online grid search with offline training and runtime parameter estimation.

=====

Reviewer #2 (Remarks to the Author):

Thanks for the authors' efforts, but still, there are many points unclear

Author response:

Thank you very much for your careful review and for your continued comments. We appreciate that several points remained unclear in the previous revision. In response, we have further revised the manuscript to improve clarity, strengthen the experimental validation, and provide more explicit explanations of the proposed framework and its evaluation. The specific changes are detailed in our responses to the comments below.

Reviewer#2-Comment#1:

1. While the authors demonstrate significant speed advantages over NetworkX and OR-Tools, the choice of baselines lacks rigor. NetworkX is a general-purpose graph library not optimized for high-performance computing, and OR-Tools is an exact solver, which inherently suffers from scalability issues. The absence of comparisons against state-of-the-art heuristic MIS solvers (e.g., specialized GPU-based algorithms or advanced metaheuristics, and please, include high-quality references within 3 years) undermines the claim of superior performance.

Author response:

Thank you very much for this important comment. We agree that the comparison in the previous version was not sufficient to support a strong performance claim, because NetworkX is a general-purpose graph library and OR-Tools is an exact solver with limited scalability for large instances.

To address this concern, we added a new subsection, “Evaluation on the MIS problem” (Section 2.3), and substantially expanded the baseline comparison (Lines 300–335). In particular, we newly included (i) ReduMIS, a strong classical MIS heuristic based on reduction (kernelization) and evolutionary search, and (ii) an SB-based hardware baseline with fixed parameters, denoted as SB (baseline). We then evaluated computation time and solution quality (#IS) across different graph sizes and edge densities, and summarized the results in Fig. 4d. In the revised manuscript, we now explain that ReduMIS consistently provides very strong solution quality, whereas the proposed solver achieves a markedly better trade-off between latency and solution quality for dynamically changing problems. We also clarify the role of SB (baseline) to show the benefit of the proposed parameter-estimation framework over a fixed-parameter SB implementation (Lines 300–335).

In addition, we clarified the implementation details of ReduMIS in the Experimental setup section by specifying that it was evaluated using the `redumis` solver in the KaMIS framework (KaMIS v3.0) (Lines 647–649).

We believe that these revisions strengthen the rigor of the baseline comparison and better position the proposed method relative to both mature software heuristics and an SB-based baseline.

Reviewer#2-Comment#2:

2. The authors address parameter dependencies in the SB algorithm through a learned ratio-based approach. While empirically effective, this method lacks theoretical grounding or mechanistic explanations. The rationale behind why such parameterization mitigates dependency issues remains unclear, making it appear as an engineering workaround rather than a principled solution. A deeper analysis of the parameter interactions and their impact on algorithmic stability would strengthen this contribution.

Author response:

Thank you very much for this important comment. We agree that the previous version did not sufficiently explain why the ratio-based parameterization provides a meaningful way to address the dependencies among the SB parameters. We also appreciate the reviewer's point that, without such an explanation, the approach could appear to be an engineering workaround rather than a principled design.

To address this concern, we added a mechanistic explanation in the Methods section, "Dependencies among the SB parameters" (Lines 557–564). In the revised manuscript, we now explicitly refer to the update equation of ballistic SB (Eq. (18)) and explain that, under the setting used in this work, where $c_0 = \eta = c$ and $\Delta_t = dt$, the magnitude of the interaction-induced update per iteration is proportional to $c \cdot dt$. This provides a qualitative explanation for why different combinations of c and dt with similar products can yield similar dynamical behavior and solution quality. We further explain that the total simulated physical time is given by $step \cdot dt$, which naturally induces a coupling between $step$ and dt .

We also clarify that these relationships are not exact invariants, because nonlinear effects such as the inelastic wall at $x = \pm 1$ affect the dynamics. However, we now state that they provide a useful dynamics-based intuition for the observed parameter correlations and therefore justify the ratio-based learning strategy as a principled approximation.

We hope that this added explanation makes clearer that the ratio-based parameterization is not introduced merely as an ad hoc engineering choice, but is motivated by the structure of the SB dynamics itself.

Reviewer#2-Comment#3:

3. The reported 5500× speedup is impressive but ambiguously attributed. The paper does not include ablation studies to disentangle the contributions of FPGA-based parallelization from those of the SB algorithm and the indexing architecture. It is plausible that the hardware acceleration dominates the performance gains, which would diminish the perceived novelty of the algorithmic components. Without such analysis, the claims of methodological advancement are weakly supported.

Author response:

Thank you very much for this important comment. We agree that, to support the methodological contribution more clearly, it is necessary to disentangle the effects of hardware acceleration from those of the proposed architectural and algorithmic components. Although a latency-breakdown figure and related discussion were already included in the previous version, we recognize that the presentation was not sufficiently organized to make this point clear. In response, we restructured the explanation and introduced a dedicated subsection, "Evaluation on the MIS problem" (Section 2.3), so that the contributions of the different components can be understood more explicitly in the Results section.

Specifically, Fig. 4a and Table 1 (Lines 273–291) now provide a system-wide latency breakdown for three MIS solver configurations: Baseline, Proposed (Generic), and Proposed (Optimized). This comparison is intended to serve as an ablation-style analysis of the proposed framework. The Baseline configuration uses a conventional SB implementation, Proposed (Generic) incorporates the proposed framework with a generic indexing-related path, and

Proposed (Optimized) further introduces problem-specific optimization. By presenting these three configurations side-by-side in a dedicated Results subsection, we clarified the individual contributions that were less apparent in the previous organization.

In the revised manuscript, we explicitly explain that the reduction in the Ising-machine execution time (H) is observed already when moving from Baseline to Proposed (Generic), and that this speedup is enabled by the indexed J-specialized MAC units, which allow a denser parallel implementation. We also explain that the reduction in offloading overhead (G) comes from compression of the J-matrix, while the further improvement from Proposed (Generic) to Proposed (Optimized) is due to the elimination of the separate index-encoding overhead by unifying Ising model conversion and index encoding (D/E). In this way, Fig. 4a and Table 1 now more clearly separate the contributions of (i) conventional SB hardware execution, (ii) indexing-based acceleration, and (iii) problem-specific architectural optimization.

Furthermore, Fig. 4b and Fig. 4c (Lines 293–299) compare CPU-based and FPGA-based implementations and clarify the hardware contribution itself: FPGA is advantageous for larger step and larger problem size, whereas CPU is faster for smaller instances. We therefore hope that the revised organization makes clearer that the manuscript now includes an explicit analysis disentangling the major factors behind the reported speedup, and that the contribution is not due to hardware acceleration alone.

We would also like to emphasize that Fig. 4d provides a solver-level comparison showing how much a conventional FPGA-based SB solver already outperforms software-based MIS solvers. In the revised manuscript, SB (baseline) is explicitly included together with NetworkX, OR-Tools, and ReduMIS, so that the reader can directly compare a conventional FPGA-implemented SB solver, without the proposed framework, against mature software-based baselines. This comparison clarifies that FPGA-based SB itself already provides a substantial speed advantage for medium-to-large problems, whereas the proposed framework further improves practicality by enhancing robustness, improving solution quality in most cases, and avoiding the small-problem inefficiency of the fixed-parameter FPGA-based SB through adaptive parameter estimation and CPU/FPGA switching.

We hope that this reorganization and the expanded explanation make clearer that the revised manuscript now includes an ablation-oriented analysis of the major factors contributing to the reported speedup, and that the proposed contribution goes beyond hardware acceleration alone.

Reviewer#2-Comment#4:

4. The discussion section lists potential applications in finance, automotive, and robotics without substantive evidence or domain-specific analysis. These fields often involve complex constraints, dynamic environments, and problem structures distinct from wireless network scheduling. The authors provide no justification for how their framework adapts to such diverse challenges, rendering the "one-size-fits-all" claim speculative and unsupported.

Author response:

Thank you very much for this important comment. We agree that the previous version could be interpreted as overgeneralizing the applicability of the proposed framework. To address this concern, we revised both the Demonstration and Discussion sections to clarify the intended scope of the method.

Specifically, in the Demonstration section (Lines 391–439), we replaced the previous broad statement with a more qualified one, now stating that the framework is applicable only to a broader class of dynamically changing problems provided that they can be formulated as QUBO/Ising models and require repeated low-latency solving under changing problem instances. We also added an explicit statement that the proposed framework does not change the underlying problem formulation, but rather improves system-level efficiency for repeatedly solved Ising-formulated subproblems.

In addition, in the Discussion section (Lines 469–473), we now state more explicitly that the intended scope is limited to real-time systems involving repeatedly solved QUBO/Ising-formulated combinatorial subproblems under changing conditions, and that continuous-time control problems and tasks that cannot be naturally decomposed into such subproblems are outside the direct scope of this work.

We hope these revisions make clear that the manuscript no longer makes a “one-size-fits-all” claim, but instead describes a more clearly delimited and practically motivated scope of applicability.

Reviewer#2-Comment#5:

5. The motivation is still unclear, as I said, latency-critical performance is generic and applies to all COPs, the authors should prove clear definition of dynamically changing COPs. The authors should explicitly define: (1) the temporal characteristics of problem changes, (2) the latency requirements for TDMA scheduling in concrete terms (e.g., 100ms for real-time control with solid reference), and (3) the performance gap between existing methods and requirements. Most importantly, the paper should quantify how much the proposed method closes this gap and how much does state-of-the-art method close this gap (with solid references).

Author response:

Thank you very much for this important comment. We agree that the previous version did not sufficiently clarify the motivation and definition of “dynamically changing” combinatorial optimization problems, and that latency-critical performance alone is too generic to serve as a clear defining feature. In response, we revised the manuscript to (i) more explicitly describe the temporal characteristics of such problems, (ii) provide a more concrete latency context for TDMA scheduling, and (iii) better delimit the intended scope of the proposed methodology.

First, in the Introduction (Lines 29–42), we added an explicit discussion of the temporal characteristics of dynamically changing problems across different application domains. In the revised manuscript, we now explain that these problems are characterized not simply by low-latency requirements, but by the repeated sequential solving of changing combinatorial optimization instances, where the next problem instance becomes available only after the current one has been solved. We also distinguish domain-dependent temporal patterns, including cumulative latency over many sequential iterations in TDMA scheduling, event- or market-driven repeated optimization in finance, and periodic real-time cycles in automotive applications.

Second, in the Demonstration section under the TDMA scheduling application example (Lines 395–409), we added a contextual explanation of latency requirements in TDMA/TSCH-based communication settings. In particular, by citing Petreska *et al.*, we now explain that end-to-end communication and schedule-update latencies on the order of hundreds of milliseconds are generally practical in process automation, whereas stricter sub-10-ms requirements are

typically associated with factory automation. Based on this context, we clarify that TDMA schedule generation/update time constitutes a non-negligible component of the application-specific latency budget, which motivates evaluating system-wide wall-clock latency rather than only accelerator execution time.

As an additional clarification, we also revised the Discussion section (Lines 469–473) so that the intended scope of the proposed methodology is more explicitly delimited. The manuscript now states that the method is intended for real-time systems in which combinatorial subproblems formulated as QUBO/Ising models must be solved repeatedly under changing conditions, and explicitly notes that continuous-time control problems and optimization tasks that cannot be naturally decomposed into repeatedly solvable QUBO/Ising subproblems are outside the direct scope of this work.

We hope these revisions clarify that “dynamically changing COPs” are defined in the manuscript not merely as latency-critical COPs, but as a more specific class of repeatedly changing and sequentially solved optimization problems, and that the motivation for TDMA scheduling is now grounded in a more concrete application and latency context.

=====

Reviewer #3 (Remarks to the Author):

The article provides various methods for solving time division multiple access (TDMS) problems using Ising machines, namely FPGA-based Ising machines. These methods include compressing the input data for FPGA-chips using a technique authors call indexing fast architecture, using XGBoost to predict model for the relevant parameters for the Ising machine and also using machine learning to select which Ising machine implementation would be fastest depending on the problem at hand.

The significance of the article is optimizing their approach to real-time solving of the TDMS problem using Ising machines. The article focuses on a single use case, but the approach can generalize to other contexts, if they fulfill the certain properties listed in the article. The approach compares favourably to quantum annealing research, but the comparisons to other FPGA-based approaches is lacking.

When comparing their work on other approaches, they only consider what sort of advantages the methods presented have compared to quantum annealing based solutions. It would be interesting to also hear how could the approaches of this article be used alongside quantum annealing, as none of the three methods seem incompatible with quantum annealing based solutions.

The key-plots of important results (For example 4-e, 5, 7. c and 7.d) are scattered across the text. It is difficult to visualise what are the advantages of the presented techniques over not using them. Also the article never compares their custom simulated bifurcation algorithm to baseline simulated bifurcation algorithm, but only to exact and simulated annealing solvers. Other than that, the methodology of the article is sound. The article provides enough details to replicate the results of the research.

Author response:

Thank you very much for your careful and constructive review. We sincerely appreciate your positive assessment that the methodology is sound and that the manuscript provides enough detail to replicate the reported results. We also appreciate your broader remarks regarding the clarity of the presentation, the comparison with baseline simulated bifurcation and other relevant approaches, the relationship to quantum annealing, and the organization of the key results in the manuscript. In response, we have revised the manuscript to reorganize the presentation of the main results, strengthen the comparative evaluation (including baseline SB and additional MIS baselines), clarify the positioning of the proposed framework with respect to quantum annealing, and better delimit the intended scope of applicability. The detailed changes are described in our point-by-point responses below.

Reviewer#3-Comment#1:

1. The optimization technique referred to as indexing fast architecture, is a well known optimization technique called dictionary coding, and should not be considered a novel concept, though we do not know whether FPGA-models have used this encoding previously to improve data transmission times. The authors correctly observe that this way of encoding is very efficient in the use case, as the number of distinct values in the J is limited, which means that using this optimization method in this context might be novel, while the encoding algorithm itself is not.

Author response:

Thank you very much for this important comment. We agree with the reviewer that the indexing scheme itself is conceptually related to well-known dictionary-based encoding, and we do not intend to claim novelty for the general encoding idea in isolation. We appreciate this clarification and have revised the manuscript to make our contribution more precisely stated.

In the revised manuscript, we now explicitly note in Section 2.1 that the proposed indexing scheme is conceptually related to dictionary-based encoding (Lines 158–163). At the same time, we clarify that our contribution is not merely to encode the data, but to exploit the structure of the Ising formulation so that the coupling matrix J , which often has limited variation, is selectively encoded while the bias vector $\mathbf{h}$ is kept in its original form. This selective treatment is specific to the structure of the Ising model used in our framework.

Furthermore, we strengthened the explanation that the main methodological novelty lies not only in the compact representation itself, but also in the associated computation scheme. In particular, we propose the indexed J -specialized MAC, which allows the dominant accumulation process to be carried out largely in the encoded state, without fully decoding J into floating-point values (Lines 185–187). This point distinguishes our approach from a simple application of dictionary coding for transmission or storage alone.

We hope that these revisions make clearer that our claim is not that dictionary-style encoding itself is new, but that the combination of (i) selective indexing tailored to the Ising model structure and (ii) encoded-state acceleration of MAC operations constitutes the methodological contribution of this part of the work.

Reviewer#3-Comment#2:

2. Authors provide a list of relevant research at the end of the results section (Shouldn't this be earlier?) and criticizes

that the quantum annealing approaches only use annealing time to benchmark their approaches and forego job queuing and offloading overhead completely. This is certainly true, but it would also be a bit misleading to compare computational overhead of integrated devices and experimental hardware that is currently only accessible to researchers through complicated cloud services. If quantum utility could be achieved here by using quantum annealers, some extra layers of classical overhead would probably be removed very quickly. Though quantum annealer specific overheads, like minor embedding overhead, might be too large in this use case, as computing a new embedding map for every single problem can be quite expensive.

Author response:

Thank you very much for this thoughtful comment. We agree that it could be misleading to criticize current quantum-annealing-based studies solely on the basis of deployment-related overheads, because such overheads are influenced not only by the optimization principle itself but also by the present hardware access model and execution environment. Our intention was not to suggest that these overheads are intrinsic limitations of quantum annealing, but rather to highlight the importance of evaluating system-wide latency under practical deployment conditions for dynamically changing problems.

To clarify this point, we revised the manuscript in two ways. First, in the Introduction (Lines 111–114), we added an explicit sentence stating that the Demonstration section concludes with a qualitative evaluation of representative application domains and discusses how the proposed framework relates to previously reported application-specific studies. This was intended to make clearer why related work is discussed there and to position that part of the manuscript as contextual and qualitative rather than as the main quantitative benchmark section.

Second, in the TDMA scheduling application discussion in the Demonstration section (Lines 398–400), we added a clarifying sentence immediately after the discussion of deployment-related overheads: “It is worth noting that such deployment-related overheads are not intrinsic to quantum annealing itself and may be mitigated as hardware and execution environments mature.” This revision was made precisely to address the concern raised by the reviewer and to avoid giving the impression that we regard current cloud-access overheads as a fundamental weakness of quantum annealing.

As a related clarification, the Discussion section also now states that the proposed approach is not intended to replace quantum annealing methods, but rather to complement them with a system-aware and low-latency optimization framework that may be used independently or together with emerging quantum hardware (Lines 459–462).

We hope these revisions make clearer both why the related-application discussion is placed in the Demonstration section and that our comparison is intended as a practical system-level perspective under current deployment conditions, rather than as a claim that such overheads are fundamental to quantum annealing itself.

Reviewer#3-Comment#3:

3. Furthermore the section seems quite defensive against quantum annealing, as the none of the techniques presented in this article are incompatible with quantum annealing approaches, but rather would be complementary. Authors are recommended to edit the text to reflect this.

Author response:

Thank you very much for this important comment. We agree that some parts of the previous version could be read as overly defensive toward quantum annealing, even though the techniques proposed in this work are not inherently incompatible with quantum-annealing-based approaches. Our intention is not to argue that the proposed framework should replace quantum annealing, but rather to emphasize system-level considerations for low-latency optimization under dynamically changing problem instances.

To make this point explicit, we revised the Discussion section and added the following clarifying statement at the end of the second paragraph (Lines 459–462): “Importantly, the proposed approach is not intended to replace quantum annealing methods, but rather to complement them by providing a system-aware and low-latency optimization framework that can be used independently or in combination with emerging quantum hardware.”

We hope this revision makes clear that the manuscript now presents the proposed methodology as complementary to quantum annealing, rather than as an alternative positioned in opposition to it.

Reviewer#3-Comment#4:

4. The figure 4.c has a typo in the image: Iteretion → Iteration

Author response:

Thank you very much for pointing out this typo. We have corrected the label from “Iteretion” to “Iteration” in the revised figure. In the current revised manuscript, the corresponding figure is Fig. 5c (previously referred to as Fig. 4c in the earlier version).

Reviewer#3-Comment#5:

5. The results section only shows how all the techniques operate with one another, but not how the distinct pieces perform on their own.

Reviewer#3-Comment#6:

6. Currently the results that reflect the impact of the compression scheme and differences between CPU and FPGA-implementations are provided in the supplementary information, while being quite important for the results. This information should be brought into the results section properly.

Author response (to #5 and #6):

Thank you very much for these important comments. We agree that, in the previous version, the manuscript did not present the effects of the individual components of the proposed framework clearly enough in the main Results section. We also agree that the differences between CPU- and FPGA-based implementations, as well as the impact of the compression/indexing scheme, are central to the contribution and therefore should be explained more explicitly in the main text rather than being left difficult to identify.

To address these concerns, we introduced a new subsection in the Results section, “Evaluation on the MIS problem”

(Section 2.3), and reorganized the presentation so that the individual contributions of the proposed techniques can be understood more clearly. In particular, Fig. 4a and Table 1 (Lines 273–291) now provide a system-wide latency breakdown for three configurations—Baseline, Proposed (Generic), and Proposed (Optimized)—which serves as an ablation-style analysis of the proposed framework. This breakdown clarifies how the distinct pieces contribute to the overall speedup, including the reduction in Ising-machine execution time through the indexed $\mathcal{J}$ -specialized MAC units, the reduction in offloading overhead through $\mathcal{J}$ -matrix compression, and the additional improvement obtained by the optimized implementation path.

In addition, Fig. 4b and Fig. 4c (Lines 293–299) are now presented in the Results section to explicitly compare CPU-based and FPGA-based implementations as functions of *step* and $|V|$. These plots make the hardware-dependent behavior much clearer by showing that FPGA is advantageous for larger step counts and larger problem sizes, whereas CPU is faster for smaller instances. This organization makes it easier to understand both the contribution of the compression/indexing scheme and the role of CPU/FPGA implementation differences in the overall framework. We therefore revised the manuscript so that these previously less clearly presented component-level results are now properly incorporated and explained in the main Results section. We hope this reorganization addresses both concerns by making the impact of the distinct pieces of the method more explicit and easier to interpret.

Reviewer#3-Comment#7:

7. Figure 3 only compares the custom SB algorithm to the NetworkX and attaining the exact solution. Should there also be a standard SB algorithm as a comparison point so the effect of the optimization techniques and hyperparameter model can be seen properly.

Author response:

Thank you very much for this important comment. We agree that including a standard/conventional SB solver as a comparison point is necessary in order to make the effects of the proposed optimization techniques and the parameter-estimation framework more clearly visible.

In response, we revised the manuscript so that the comparison is no longer limited to NetworkX and an exact solver. Specifically, in the new subsection “Evaluation on the MIS problem” (Section 2.3), Fig. 4d now includes a conventional SB-based hardware baseline with fixed parameters, denoted as SB (baseline), in addition to the proposed solver, NetworkX, OR-Tools, and ReduMIS (Lines 300–335). This comparison was added precisely to clarify how much improvement comes from the proposed adaptive framework relative to a conventional FPGA-implemented SB solver without the proposed enhancements.

In the revised text, we also explain that SB (baseline) uses a fixed parameter set ($c = 0.2$, $dt = 0.2$, $step = 1000$), whereas the proposed solver uses problem-dependent parameter estimation and machine selection. As a result, Fig. 4d now makes it possible to assess not only the difference between SB and software-based MIS solvers, but also the benefit of the proposed optimization techniques over a conventional SB implementation (Lines 300–335).

We hope this revision addresses the reviewer’s concern by making the role of the optimization techniques and the parameter-estimation model much clearer in the main comparison.

=====

Reviewer #4 (Remarks to the Author):

I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

Author response:

Thank you very much for your time and for contributing to the review of our manuscript.

=====

Yours sincerely,

Yohei Hamakawa, Tomoya Kashimata, Masaya Yamasaki, Kosuke Tatsumura

Yohei Hamakawa

Corporate Laboratory Toshiba Corporation

1 Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan

E-mail: yohei.hamakawa.r13@mail.toshiba Tel: +81-50-3148-5260

Response to Reviewers:

We sincerely thank the reviewers and the editor for their careful evaluation of our manuscript and for their constructive feedback throughout the review process. We are grateful that the major scientific issues were considered to have been satisfactorily addressed. In this final revision, we have made targeted changes to address the remaining reviewer comment and to complete the editorial requirements for final submission. In particular, we improved the logical flow of the TDMA scheduling discussion, updated the Data Availability and Code Availability statements, added Supplementary Data 1 with an accompanying README file, and finalized the organization and description of the supporting information files. We believe that these final revisions further improve the clarity, transparency, and completeness of the manuscript.

Summary of Main Revisions:

In this final revision, we made the following updates:

- 1) We revised the paragraph on TDMA scheduling to improve its logical flow, as suggested by Reviewer #3. In particular, we reorganized the paragraph so that the application context and the practical importance of system-wide latency are introduced more clearly, and the discussion of quantum annealing-related deployment overheads is incorporated more smoothly.
- 2) We added Supplementary Data 1 containing the benchmark input data used in the experiments. This dataset includes the random graph instances used for the MIS benchmarking in Fig. 4d and the precomputed TDMA scheduling instances used in Fig. 5e, together with a README file describing the file formats, naming conventions, directory structure, loading methods, and generation procedures.
- 3) We revised the Data Availability statement to reflect the addition of Supplementary Data 1 and to clarify the availability of the benchmark input data used in this study.
- 4) We revised the Code Availability statement to clarify that the custom code is not publicly available because it includes proprietary and hardware-dependent implementation components developed for the target embedded system, while the algorithmic details are described in the main text.
- 5) We finalized the organization of the supporting information files and their descriptions for submission, including the Supplementary Information PDF, Supplementary Data 1, Supplementary Movie 1, Supplementary Movie 2, and the Inventory of Supporting Information document.
- 6) We also completed the remaining editorial and submission-related updates requested in the checklist.

Point-by-point responses to Reviewers' comments: (Bold: Reviewers' comments)

Reviewer #1 (Remarks to the Author):

I have reviewed the authors' responses to Reviewer 1's comments and find that they have been addressed satisfactorily.

Author response:

We thank Reviewer #1 for the careful evaluation of our manuscript and for confirming that the responses have satisfactorily addressed the comments.

=====
Reviewer #2 (Remarks to the Author):

The authors addressed my concerns.

Author response:

We thank Reviewer #2 for the careful evaluation of our manuscript and for confirming that the responses have satisfactorily addressed the comments.

=====
Reviewer #3 (Remarks to the Author):

I have checked the authors' edits to the manuscript and their answers to the reviewer comments. They have adequately addressed the article's key issues.

The only issue I would like to have corrected is:

1. The paragraph for TDMA scheduling on rows 395-409 should be structured better, as the added sentence about quantum annealing overheads is a bit abrupt and does not fit the logical flow of the text. Content itself is fine.

Author response:

We thank the reviewer for this helpful suggestion. We have revised the paragraph on TDMA scheduling to improve its logical flow. In particular, we first clarify the application context and the practical importance of system-wide latency, and then discuss the evaluation setting in Ref. [1], including the note that deployment-related overheads are not intrinsic to quantum annealing itself. This reorganization preserves the original content while making the argument more coherent.

=====
Reviewer #4 (Remarks to the Author):

I co-reviewed this manuscript with one of the reviewers who provided the listed reports. This is part of the Nature Communications initiative to facilitate training in peer review and to provide appropriate recognition for Early Career Researchers who co-review manuscripts.

Reviewer #4 (Remarks on code availability):

Code is not publicly available.

Author response:

We thank the reviewer for this comment. As stated in the Code Availability section, the custom code used in this study is not publicly available because it includes proprietary and hardware-dependent implementation components developed for the target embedded system. The algorithmic details are described in the main text using pseudocode, mathematical equations, and implementation descriptions.

=====

Yours sincerely,

Yohei Hamakawa, Tomoya Kashimata, Masaya Yamasaki, Kosuke Tatsumura

Yohei Hamakawa

Corporate Laboratory Toshiba Corporation

1 Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan

E-mail: yohei.hamakawa.r13@mail.toshiba Tel: +81-50-3148-5260