IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 24 June 2024, accepted 15 July 2024, date of publication 22 July 2024, date of current version 31 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3431540

RESEARCH ARTICLE

Stochastic Simulated Quantum Annealing for Fast Solution of Combinatorial Optimization Problems

NAOYA ONIZAWA[1], (Member, IEEE), RYOMA SASAKI[1], DUCKGYU SHIN[1], WARREN J. GROSS[2], AND TAKAHIRO HANYU[1], (Senior Member, IEEE)

[1]Research Institute of Electrical Communication, Tohoku University, Sendai 980-8577, Japan
[2]Department of Electrical and Computer Engineering, McGill University, Montreal, QC H3A 0G4, Canada

Corresponding author: Naoya Onizawa (naoya.onizawa.a7@tohoku.ac.jp)

This work was supported in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP21H03404; in part by the Japan Science and Technology (JST) Core Research for Evolutional Science and Technology (CREST) under Grant JPMJCR19K3; and in part by Tohoku Initiative for Fostering Global Researchers for Interdisciplinary Sciences (TI-FRIS) of Ministry of Education, Culture, Sports, Science and Technology (MEXT's) Strategic Professional Development Program for Young Researchers.

ABSTRACT Combinatorial optimization problems are frequently classified as NP-hard, which means that the time needed to find the optimal solution generally increases exponentially with the problem size. However, the existing research gap highlights the necessity to develop combinatorial optimization methods capable of scaling to large problems while maintaining low solution latency. In this paper, we introduce stochastic simulated quantum annealing (SSQA) for fast solution of combinatorial optimization problems. SSQA is designed based on stochastic computing, which can simulate quantum annealing (QA) by using multiple replicas of spins (probabilistic bits) in classical computing. The use of stochastic computing leads to an efficient parallel spin-state update algorithm, enabling quick search for a solution around the global minimum energy. Therefore, SSQA realizes quantum-like annealing for large-scale problems and can handle fully connected models in combinatorial optimization, unlike QA. The proposed method is evaluated in MATLAB on graph isomorphism problems, which are typical combinatorial optimization problems. It can handle a 100-times larger problem size compared to QA and a 25-times larger problem size compared to a traditional SA method, respectively, for similar convergence probabilities. The time to solution of SSQA is 11.5 times and 423 times smaller than that of the conventional stochastic simulated annealing method and traditional SA, respectively, when solving a 2,025-node combinatorial optimization problem.

INDEX TERMS Stochastic computing, Boltzmann machine, Ising model, Trotter-Suzuki decomposition.

I. INTRODUCTION

In diverse realms such as science, engineering, and economics, optimization problems play a pivotal role, modeling complex decision-making processes and requiring sophisticated solution strategies [1]. For example, in the scientific domain, optimization techniques are crucial in areas like bioinformatics for protein structure prediction, and in physics for parameter estimation in complex models [2]. These challenges are compounded by the vastness of the solution space and the intricate nature of the problems, necessitating the development of robust optimization methodologies to efficiently navigate the complex landscape and find optimal or near-optimal solutions in real-world applications, such as the unmanned aerial vehicles and the truss design [3], [4], [5]. Metaheuristics have come to the forefront in this context, offering a suite of high-level strategies capable of navigating through the solution landscape to uncover high-quality solutions [6] for single-objective and multi-objective design problems [7], [8]. Broadly categorized into population-based and single-solution approaches, each category of metaheuristics brings unique advantages to the table [9].

Population-based metaheuristics, exemplified by genetic algorithms (GA), particle swarm optimization (PSO),

The associate editor coordinating the review of this manuscript and approving it for publication was Brian Ng.

differential evolution (DE), and ant colony optimization (ACO), leverage a collective of solutions to explore the search space [10], [11]. These methods are adept at handling complex and large-scale optimization problems, offering the ability to explore multiple regions of the search space concurrently and enhancing the likelihood of locating global optima. In contrast, single-solution metaheuristics focus on iteratively refining a single solution, and they encompass a range of versatile algorithms. Algorithms, such as tabu search (TS), iterated local search (ILS), and variable neighborhood search (VNS) offer unique mechanisms for local and global search, demonstrating efficiency in terms of computational resources and showing particular strength in scenarios where solution evaluation is computationally demanding or memory resources are limited [12], [13].

In the realm of single-solution methods, simulated annealing (SA) is particularly noted for its effectiveness in addressing combinatorial optimization problems, where determining the optimal configuration of components is crucial [14], [15], [16], [17]. Compared with the aforementioned algorithms, SA stands out for its simplicity, facilitating its straightforward implementation in hardware solutions aimed at swiftly solving large-scale combinatorial optimization challenges. This study focuses on advancing SA to enable rapid processing of extensive combinatorial optimization problems, leveraging its inherent simplicity and hardware implementation potential to achieve significant performance improvements. Various hardware implementation of SA have been developed using an Ising model [18], [19]. SA-like parallel-update algorithms have also been reported in [20]. In addition to the classical approach, quantum annealing (QA) [21] is an alternative solution implemented using quantum devices, such as D-Wave quantum annealers [22]. QA is expected to solve large-scale combinatorial optimization problems faster than conventional SA methods; however, it currently faces limitations in handling small-size problems due to quantum device performance constraints [23].

Recently, a stochastic-computing-based SA (SSA) was proposed, which achieved orders of magnitude faster annealing compared to conventional SA and QA methods [24]. SSA employs a parallel form of simulated annealing by approximating probabilistic bits (p-bits) [25] using stochastic computing [26], [27], where p-bit based simulated annealing is called pSA [28]. The combination of p-bits and stochastic computing enables rapid convergence to the global minimum of the objective functions. In addition, SSA is scalable for solving large-scale problems as it can be designed using digital circuits in hardware implementation [29]. The primary objective of both SA and QA methods is the development of a rapid hardware accelerator designed for addressing large-scale combinatorial optimization challenges, as represented by fully connected Ising models. Within the SA framework, [18] demonstrates capability in managing problems up to 20,000 nodes, albeit with the limitation that connections are restricted to adjacent nodes.

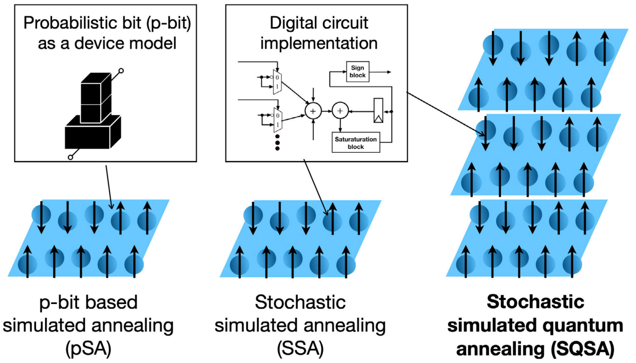

Probabilistic bit (p-bit) as a device model
Digital circuit implementation
0
1
0
1
Sign block
Saturaturation block
p-bit based simulated annealing (pSA)
Stochastic simulated annealing (SSA)
Stochastic simulated quantum annealing (SQSA)

FIGURE 1. **Overview of the proposed stochastic simulated quantum annealing (SSQA) with related existing works. Stochastic simulated annealing (SSA) was proposed by approximating probabilistic bits (p-bits) using stochastic computing [24], where p-bit based simulated annealing (pSA) realizes parallel updating of nodes [28]. The approximation leads to faster annealing than pSA and scalable digital hardware implementation. SSQA is the extended algorithm of SSA using multiple replicas of spins, which results in higher convergence than SSA while it can be implemented in digital circuits.**

Conversely, [19] and [29] extend this to handle a few hundred nodes in a fully connected fashion. Regarding QA, D-Wave systems are capable of processing up to 5,000 nodes, yet only with neighboring connections. This underscores a significant research gap: the critical need for advanced combinatorial optimization approaches that efficiently scale to large-scale, fully connected problems, while simultaneously ensuring low solution latency.

In this paper, we present stochastic simulated quantum annealing (SSQA) for efficiently solving large-scale combinatorial optimization problems. Fig. 1 highlights the proposed SSQA algorithm along with the existing related works. SSQA is a simulated quantum annealing (SQA) approach, which emulates quantum behavior on classical computers by using multiple replicas of classical bits (spins) through the Trotter-Suzuki decomposition [30], [31]. SQA is expected to outperform conventional SA in large-scale combinatorial optimization problems [23], and unlike QA, it can handle fully connected Ising models for combinatorial optimization problems [32]. In the proposed SSQA, we employ an efficient spin-state update algorithm using stochastic computing, similar to SSA [24], enabling quick exploration of the solution space around the global minimum energy. The proposed algorithm is simulated using MATLAB on graph isomorphism (GI) problems with up to 2,500 spins, where GI is a typical combinatorial optimization problem represented by a fully connected Ising model. The simulation results demonstrate that the proposed method achieves an order of magnitude reduction in time-to-solution (TTS) compared to and SSA and a few orders of magnitude reduction compared to the traditional SA. Furthermore, when compared to experimental results of QA on the 504-qubit D-Wave Two machine [33], SSQA can handle problems

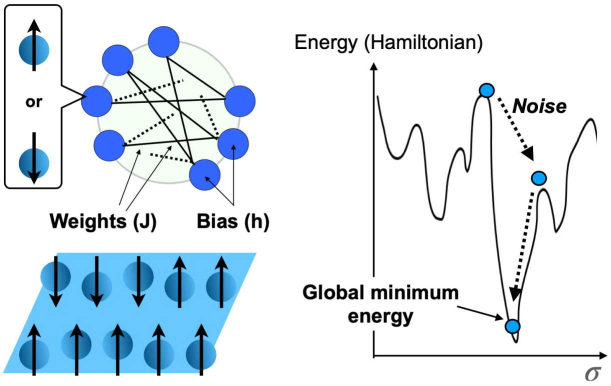


FIGURE 2. Simulated annealing (SA) based on a spin network that consists of spins, spin biases, and spin weights. Spin states can be flipped between '+1' and '-1' in an attempt to reach the global minimum energy of the Hamiltonian.

approximately two orders of magnitude larger and leads to a scalable digital hardware implementation, similar to SSA.

The contributions of the paper are:

1) Introducing the SSQA algorithm, which can be efficiently implemented in digital circuits for scalability,
2) Demonstrating the effectiveness of the proposed method in the GI problems,
3) Solving a problem 25 times larger than that handled by the conventional SSA method, with similar computation cost.

The rest of the paper is structured as follows. Section II provides a review of conventional SA methods and Ising models for combinatorial optimization problems. Section III introduces the proposed SSQA algorithm based on stochastic computing. Section IV compares the proposed algorithm with the traditional SA, SSA and QA methods. Finally, Section V concludes the paper.

II. PRELIMINARIES

A. SIMULATED ANNEALING (SA) AND ISING MODEL

Fig. 2 illustrates a spin network-based simulated annealing (SA) algorithm. The spin network is constructed using spins, spin biases (h), and spin weights (J) connecting the spins. The spin states (σ) can take two values: '-1' and '$+1$'. The spin network is based on the Ising model [32], which represents a Hamiltonian (energy function) as follows:

$$H(\sigma) = -\sum_{i} h_i \sigma_i - \sum_{i<j} J_{ij}\sigma_i\sigma_j. \tag{1}$$

where i and j ($1 \leq i, j \leq N$) are indices of spins, and N is the number of spins.

SA is commonly applied to solve various combinatorial optimization problems, including GI, traveling salesman, and maximum cut problems [33], [34], [35]. These problems can be represented by the coefficients h and J in Eq. (1). During the annealing process, spin states can be flipped between '$+1$' and '-1' in an attempt to reach the global minimum of the Hamiltonian. Several SA methods have been proposed, such as serial updating [17], parallel updating [18], and parallel tempering [19], to enhance the efficiency of the annealing process.

B. STOCHASTIC-COMPUTING-BASED SIMULATED ANNEALING (SSA)

Recently, a p-bit-based simulated annealing (pSA) approach was introduced in [28]. A p-bit is a probabilistic bit that can be in one of two spin states, '$+1$' and '-1'. It has been proposed for invertible logic, which is an unconventional computing technique [25], [36], [37]. pSA, implemented on an underlying Boltzmann machine [38], enables parallel updating of spins for faster simulated annealing. However, it suffers from slow convergence to the global minimum energy.

To address this issue, the SSA method was proposed in [24]. SSA utilizes p-bits that are approximated using integral stochastic computing [39]. It is worth noting that integral stochastic computing is an extended version of stochastic computing [26], [27] and offers area-efficient hardware implementation [40], [41], [42]. The approximation of p-bits in SSA leads to a faster simulated annealing process compared to pSA. In SSA, each spin state is updated as follows:

$$I_i(t+1) = h_i + \sum_{j} J_{ij} \cdot \sigma_j(t) + n_{rnd} \cdot r_i(t), \tag{2a}$$

$$Is_i(t+1) = \begin{cases} I_0(t) - \alpha, \text{ if } Is_i(t) + I_i(t+1) \geq I_0(t) \\ -I_0(t), \text{ else if } Is_i(t) + I_i(t+1) < -I_0(t) \\ Is_i(t) + I_i(t+1), \text{ otherwise} \end{cases} \tag{2b}$$

$$\sigma_i(t+1) = \begin{cases} 1, & \text{if } Is_i(t+1) \geq 0 \\ -1, & \text{otherwise,} \end{cases} \tag{2c}$$

where $\sigma_i(t) \in \{-1, 1\}$ and $\sigma_i(t+1) \in \{-1, 1\}$ represent the binary input and output spin states, respectively. I_0 is the pseudo inverse temperature, $I_i(t+1)$ and $Is_i(t+1)$ are real-valued internal signals, and n_{rnd} is the noise magnitude of a random signal $r_i(t) \in \{-1, 1\}$. α is the minimum resolution of data representation. If only integer values are used in Eq. (2), α can be 1 [24]. Equations (2b) and (2c) provide approximations of the tanh function for p-bits using integral stochastic computing. During the annealing process in SSA, I_0 gradually increases while updating all spin states. Since SSA is designed based on stochastic computing, it can be implemented in both software and hardware [29].

III. PROPOSED STOCHASTIC SIMULATED QUANTUM ANNEALING (SSQA) ALGORITHM

A. DEFINITION

Table 1 provides a summary of annealing methods, including SA, QA, and SSA, all utilized for solving combinatorial optimization problems. SA [14] is a conventional simulated annealing method that updates a randomly selected spin to reach the global minimum of a classical Hamiltonian

TABLE 1. Summary of annealing methods for solving combinatorial optimization problems.

	SA [14]	QA [21]	SSA [24]	This work (SSQA)
Hamiltonian	Classical	Quantum	Classical	Pseudo quantum
Spin update	Serial	Parallel	Parallel	Parallel
Computational model	Classical computing	Quantum computing	Classical computing (stochastic computing[a])	Classical computing (stochastic computing[a])
Problem limitation	No	Yes	No	No

[a] Stochastic computing can be implemented using typical digital circuits.

defined by Eq. (1). QA [21] achieves parallel spin updates using quantum devices to reach the global minimum of a quantum Hamiltonian; however, its problem-solving capabilities are limited due to current device performance constraints. SSA [24] achieves parallel spin updates based on stochastic computing (classical computing) using a classical Hamiltonian. In this paper, we propose stochastic simulated quantum annealing (SSQA) as an extension of SSA, which achieves parallel spin updates using a pseudo quantum Hamiltonian. The Hamiltonian and the spin-update algorithm will be described in the following subsections.

B. HAMILTONIAN

Let us explain the Hamiltonian of SSQA. SSQA solves a combinatorial optimization problem represented by a pseudo quantum Hamiltonian, which is derived from the quantum Hamiltonian used in QA. In contrast to SA, the quantum Hamiltonian $H_q(\sigma)$ is represented as follows:

$$H_q(\sigma) = -\sum_i h_i\sigma_i^z - \sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z - \Gamma_x\sum_i \sigma_i^x, \tag{3}$$

where $-\sum_i h_i\sigma_i^z - \sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z$ represents the problem Hamiltonian, and Γ_x is a scheduling parameter for annealing. Here, σ_i^z and σ_i^x are the Pauli matrices [21] that act on quantum devices used in quantum annealing machines, such as D-Wave [22].

The Trotter-Suzuki decomposition approximates the quantum Hamiltonian in equation (3) to generate a pseudo quantum Hamiltonian. This allows its representation using multiple replicas of spins on classical computers, thereby approximating and representing the time evolution of QA [30], [31]. The pseudo quantum Hamiltonian, denoted $H_c(\sigma)$, can be expressed as follows [43]:

$$H_c(\sigma) = \sum_{k=1}^{R}\Big(H_p(\sigma) - J_\perp\sum_i \sigma_{i,k}\sigma_{i,k+1}\Big), \tag{4}$$

$$H_p(\sigma) = -\sum_i h_i\sigma_{i,k} - \sum_{i<j} J_{ij}\sigma_{i,k}\sigma_{j,k}, \tag{5}$$

where $H_p(\sigma)$ is the problem Hamiltonian, $\sigma_{i,k}$ represents the spin state of the k-th replica of the spin network ($1 \leq k \leq R$), R is the number of replicas of spins used to represent the q-bit, and $J_\perp$ is a scheduling parameter corresponding to Γ_x. The problem Hamiltonian, $H_p(\sigma)$, is the same as the one in equation (1) used in SA.

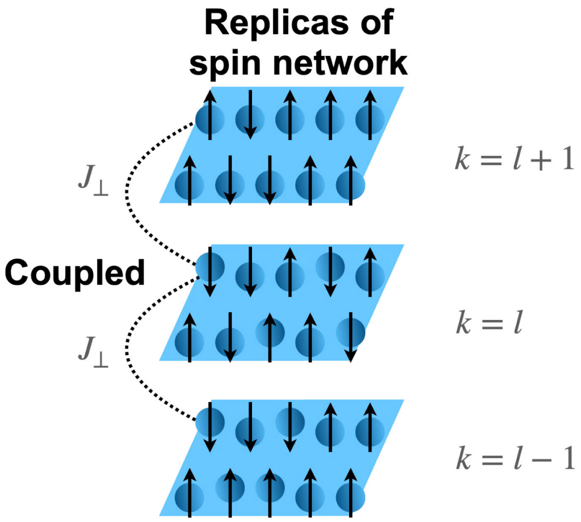


FIGURE 3. A pseudo quantum Hamiltonian based on the Trotter-Suzuki decomposition using spin network replicas for SSQA. Each spin network is coupled with the upper and the lower spin networks with $J_\perp$.

Moreover, the Trotter-Suzuki decomposition offers the advantage of allowing parallel implementation in hardware. By decomposing the time evolution operator into a series of simpler operations, each of which can be executed independently, the Trotter-Suzuki method facilitates parallel processing. This parallelism is especially beneficial in practical implementations, as it can significantly reduce computation time and enhance the efficiency of solving complex optimization problems. Fig. 3 illustrates the pseudo quantum Hamiltonian utilizing R replicas of the spin network.

Each spin network represents the problem Hamiltonian and is loosely coupled to the upper and lower spin networks using $J_\perp$. It is important to note that the top replica is coupled to the bottom one.

C. SPIN-UPDATE ALGORITHM

Fig. 4 illustrates the concept of the annealing process in SSQA that performs based on the pseudo quantum Hamiltonian with equations (4) and (5). There are R replicas of spins, which are coupled by $J_\perp$. During the annealing process, spin states are randomly flipped in order to reach the global minimum energy of the problem Hamiltonian. Each replica of the spin network independently searches for the global minimum energy with the effect of $J_\perp$. When $J_\perp$ is small, the spin network is loosely coupled with the upper and

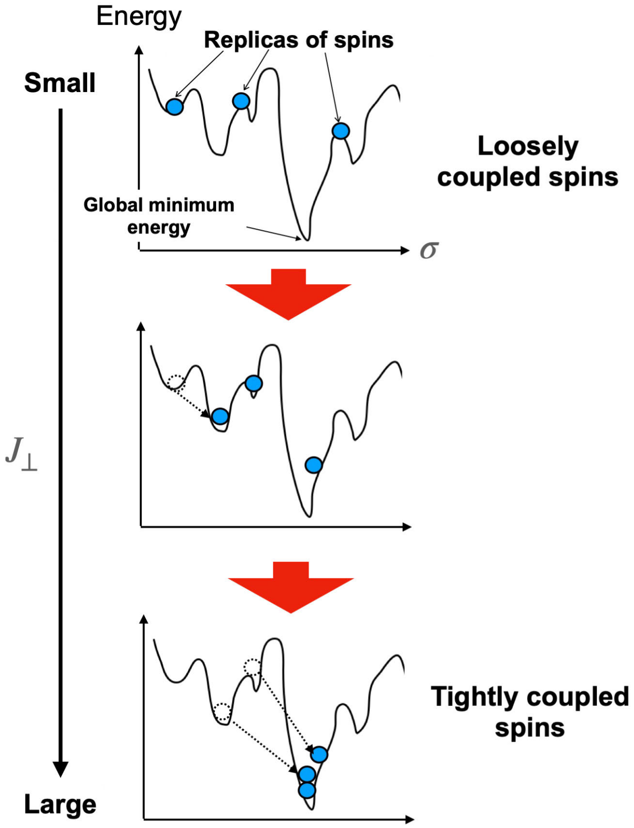


FIGURE 4. Concept of annealing process of SSQA. Each replica of the spin network searches for the global minimum energy with increasing $J_\perp$, which can reach the global minimum based on the coupled spins.

lower replicas, allowing each replica to search for the global minimum energy of the problem Hamiltonian independently with minimal influence from the neighboring replicas. On the other hand, when $J_\perp$ is large, the spin network becomes tightly coupled, enabling it to reach the global minimum energy by leveraging replicated spins with low energies.

SSQA is designed based on integral stochastic computing [39]. The spin-update algorithm of SSQA is designed based on Eq. (2) of SSA, as SSQA is an extension of SSA. The update rule for the i-th spin state in the k-th replica is as follows:

$$I_{i,k}(t+1) = h_i + \sum_j J_{ij} \cdot \sigma_{j,k}(t) + n_{rnd} \cdot r_i(t) + J_\perp(t) \cdot \sigma_{i,k+1}(t-d), \tag{6a}$$

$$Is_{i,k}(t+1) = \begin{cases} I_0 - \alpha, \text{ if } Is_{i,k}(t) + I_{i,k}(t+1) \geq I_0 \\ -I_0, \text{ else if } Is_{i,k}(t) + I_{i,k}(t+1) < -I_0 \\ Is_{i,k}(t) + I_{i,k}(t+1), \text{ otherwise} \end{cases} \tag{6b}$$

$$\sigma_{i,k}(t+1) = \begin{cases} 1, & \text{if } Is_{i,k}(t+1) \geq 0 \\ -1, & \text{otherwise.} \end{cases} \tag{6c}$$

where $\sigma_{i,k}(t) \in \{-1, 1\}$ and $\sigma_{i,k}(t+1) \in \{-1, 1\}$ represent the binary input and output spin states, respectively. Here, $\sigma_{i,k+1}(t-d)$ represents the i-th spin state in the $(k+1)$-th replica, and d is the delay cycle for the coupled effect. The coupled effect from the upper replica is represented by $J_\perp(t) \cdot \sigma_{i,k+1}(t-d)$. $I_{i,k}(t+1)$ and $Is_{i,k}(t+1)$ are real-valued internal signals. In stochastic computing, Eq. (6b) and Eq. (6c) approximates $tahn(I_0 \cdot I_{i,k})$ [27], [39], where $Is_{i,k}$ is the input of Eq. (6c). The mean of $\sigma_{i,k}$ is closer to the output of the tanh function.

In SSQA, all the $(N \cdot R)$ spin states are updated in parallel. Therefore, the computation cost of SSQA depends on the number of replicas R, which can influence the probability of convergence to the global minimum energy and the simulation time.

IV. EVALUATION

A. HAMILTONIAN DESIGN FOR GI

To evaluate the proposed SSQA algorithm, SSQA is simulated on the graph isomorphic (GI) problem, which is a typical combinatorial optimization problem [33]. A GI problem determines whether two graphs are isomorphic. When solving a combinatorial optimization problem, it is first presented using a quadratic unconstrained binary optimization (QUBO) model. The QUBO model is defined as follows:

$$H(x) = \sum_{i,j} Q_{ij} x_i x_j \tag{7}$$

where Q_{ij} is an upper triangular matrix and $x_i \in \{0, 1\}$ are binary variables.

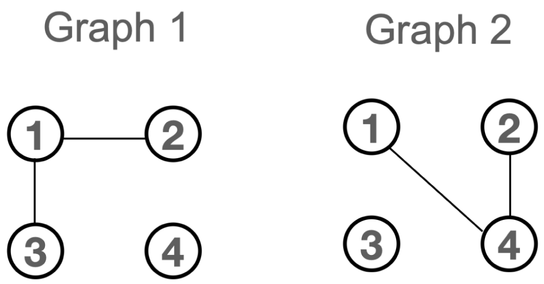
Fig. 5 (a) shows an example of the four-node GI problem. In this example, Graph 1 and Graph 2 are isomorphic. The QUBO model is obtained based on a vertex mapping penalty (C_1) and an edge inconsistency penalty (C_2) [32] as follows:

$$\begin{aligned} H(x) = {} & C_1 \sum_u (1 - \sum_i x_{u,i})^2 + C_1 \sum_i (1 - \sum_u x_{u,i})^2 \\ & + C_2 \sum_{i,j \notin E_1, i \neq j} \sum_{u,v \in E_2} x_{u,i} x_{v,j} \\ & + C_2 \sum_{i,j \in E_1} \sum_{u,v \notin E_2, u \neq v} x_{u,i} x_{v,j}, \end{aligned} \tag{8}$$

where $x_{u,i} \in \{0, 1\}$ for every possible mapping of a vertex u in Graph 2 to a vertex i in Graph 1. Q is an $N \times N$ matrix, where N is the square of the number of nodes in the GI problems. Note that N corresponds to the number of spins in the spin network. The Q coefficients of this example are shown in Fig. 5 (b).

B. SIMULATION SETUP

In this simulation, firstly, a GI problem for each graph size is generated according to [33]. Let us explain the process for a five-node graph. A graph with five nodes is randomly generated, where a probability of connecting between two nodes is 50%. Hence, five edges exist on average as there are 10 possible connections for the five-node graph. The generated graph ('Graph 1') is copied to another graph ('Graph 2'), which then creates a QUBO model from



(a)

$$Q = \left[\begin{array}{cccccccccccccccc}
x_{11} & x_{12} & x_{13} & x_{14} & x_{21} & x_{22} & x_{23} & x_{24} & x_{31} & x_{32} & x_{33} & x_{34} & x_{41} & x_{42} & x_{43} & x_{44} \\
-1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & -1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & -1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{array}\right]$$

(b)

FIGURE 5. Example of a four-node graph isomorphism (GI) problem. (a) Two graphs are isomorphic and (b) *Q* coefficients of the QUBO model corresponding to the two four-node graphs.

TABLE 2. Annealing parameters for SSQA.

Parameter	Value
$J_{\perp min}$	0
$J_{\perp max}$	0.5
β	3
d	1
I_0	2
n_{rnd}	1
τ	50 to 100
R	2 to 50
EC	10,000 to 40,000
SC	EC/R
Number of cycles per iteration	$\tau \cdot (\beta + 1)$
Number of iterations	$\mathrm{SC}/(\tau \cdot (\beta + 1))$

equation (8). Secondly, the QUBO model is converted into Hamiltonian coefficients h and J as described in equations (4) and (5), with $\sigma_i = 2x_i - 1$, $h_i = -\frac{1}{2}Q_{ii} - \frac{1}{4}\sum_{\substack{j\in\partial_i \\ i\neq j}} Q_{ij}$, and $J_{ji} = J_{ij} = -\frac{1}{4}Q_{ij}$ (see details in [24]).

With the established h and J, simulated annealing begins, searching for the global minimum energy. In SSQA, the spin states are updated in accordance with equation (6) during the annealing process. The parameters for SSQA are summarized in Table 2 and are described as follows. During each iteration, the scheduling parameter $J_\perp$ is incrementally increased from a minimum value, $J_{\perp min} = 0$, to a maximum, $J_{\perp max} = 0.5$. Within the iteration, $J_\perp$ is updated as $J_\perp(t+1) = J_\perp(t) + (J_{\perp max} - J_{\perp min})/\beta$ at every τ cycle. For instance, each iteration involves 400 cycles with $\beta = 3$ and $\tau = 100$. When a new iteration begins, $J_\perp$ is reset to $J_{\perp min}$. Parameters specific to SSQA, such as $d = 1$, $I_0 = 2$ and $n_{rnd} = 1$, are determined based on a grid search. Other parameters, like τ and R, are varied to evaluate the performance of SSQA. All simulations are conducted using MATLAB R2020b on an AMD Ryzen-9 5950X at 3.4 GHz with 32 GB of memory. The source codes are available on [44].

As the number of replicas R increases, the computation cost of SSQA based on equation (6) also increases. To objectively assess the dependency of performance on R, we define the term 'Equivalent Cycles (EC)' as follows:

$$\text{Equivalent Cycles (EC)} = R \times \text{Simulation Cycles (SC)}. \tag{9}$$

For instance, with EC set to 20,000, SC amount to 2,000 for $R = 10$ and drop to 500 for $R = 40$. Given the same EC value, the computational cost (i.e., simulation time) can be nearly identical for any arbitrary value of R.

C. SIMULATION ANALYSIS OF SSQA

Fig. 6 displays energy versus cycles for the 1st, 10th, and 20th replicas in SSQA applied to a GI problem of $N = 2,500$ with $R = 25$. The simulation encompasses four iterations, each of which consists of 400 cycles with $\tau = 100$. Each replica strives to minimize its energy as $J_\perp$ increases. A replica achieving a lower energy level can influence others in their quest to reach the global minimum energy. When $J_\perp$ reaches 0.5, each replica is significantly affected by its neighboring replicas. At the 1st, 3rd, and 4th iterations, these replicas reach the global minimum energy at different timings.

Fig. 7 (a) displays the probabilities of global minimum energy ($P_s(T)$) versus the number of replicas R in SSQA for EC set at 20,000 and 100 trials with varying problem sizes of N. As N is the square of the number of nodes in the GI problems, $N = 900$ implies that the isomorphism of two 30-node graphs is being checked. Fig. 7 (b) shows the simulation time T in SSQA for EC of 20,000. The simulation time remains almost equivalent for all R as assumed in Eq. (9). Considering problem size, $P_s(t)$ decreases and the simulation time increases as N increases.

To evaluate the performance of SSQA, time-to-solution (TTS) is selected as a performance metric [45]. TTS is the approximated time to obtain the global minimum energy of the problem Hamiltonian and is defined as follows:

$$\mathrm{TTS} = t\frac{\ln(1 - P_T)}{\ln(1 - P_s(t))}, \tag{10}$$

where P_T is the probability of finding the global minimum energy at least one time in T trials within the simulation (execution) time t. Using the simulation results of $P_s(t)$ and t, TTS is calculated with a target specification of P_T. When $P_s(t)$ is 0% or 100%, TTS cannot be calculated. For

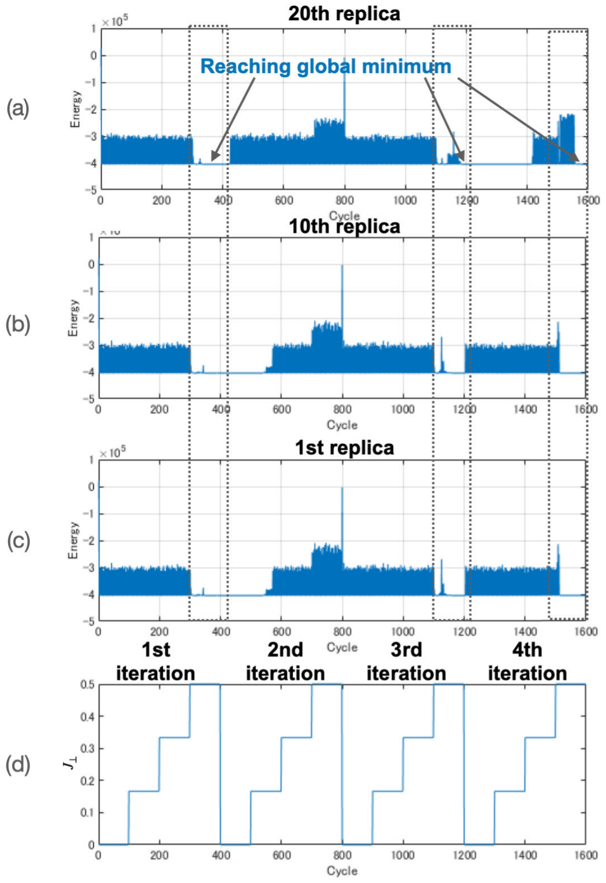


FIGURE 6. Energy versus cycles in SSQA for a GI problem of $N = 2,500$ with $R = 25$: (a) 20th replica, (b) 10th replica, (c) 1st replica, (d) J_{perp}. In each iteration, $J_\perp$ is increased with $\tau = 100$. At the 1st, 3rd, and 4th iterations, these replicas reach the global minimum energy with different timing.

TABLE 3. Five cases of different EC and τ with $R = 25$ used for evaluation in Fig. 9.

Parameter	Case 1	Case 2	Case 3	Case 4	Case 5
EC	10,000	10,000	20,000	20,000	40,000
τ	50	100	50	100	100
# iterations	2	1	4	2	4

example, when $P_s(t)$ is equal to 100%, TTS approaches 0 as the denominator of TTS is negative infinity.

Fig. 8 displays TTS versus the number of replicas R in SSQA for the GI problems, with $P_T = 0.99$. When EC is set to 20,000, TTS is minimized at an R value of approximately 20. When EC is set to 40,000, some TTS values cannot be calculated because $P_s(t)$ equals 1 in the case of $N = 900$. This implies that an EC of 40,000 is too large for smaller problem sizes. An R value of approximately 30 is ideal when EC is set to 40,000. Based on these findings, $R = 25$ is selected for comparing SSQA with conventional methods, which will be described in the subsequent subsections.

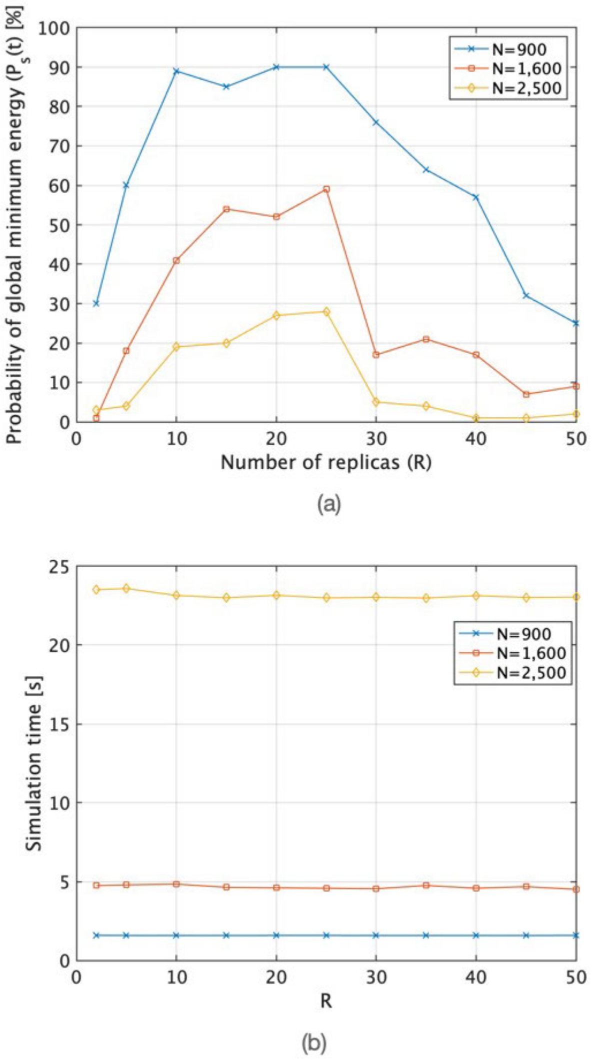
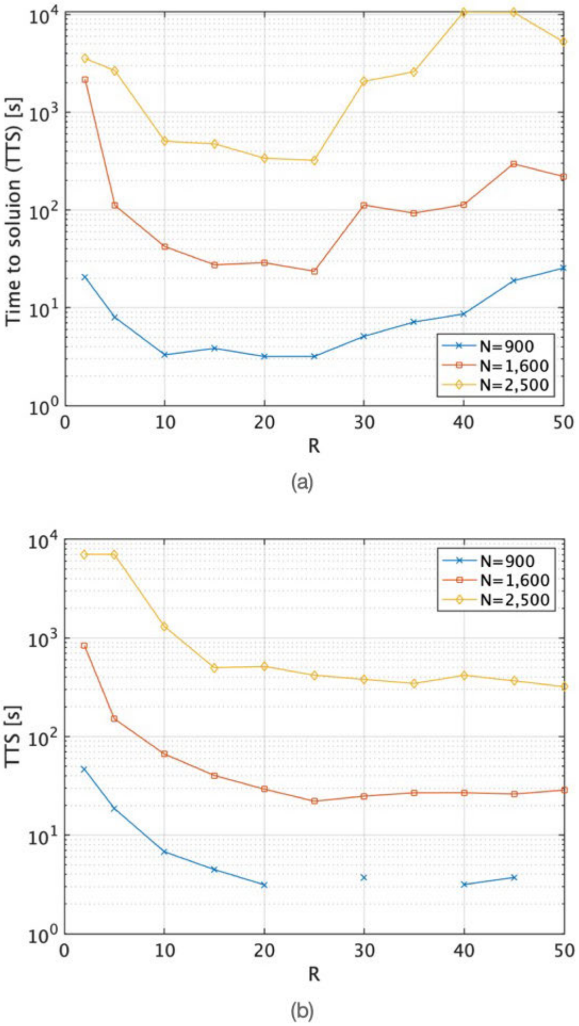


FIGURE 7. Performance of SSQA in GI problems for EC = 20,000 and 100 trials: (a) probability of global minimum energy ($P_a(t)$) versus Number of replicas (R) and (b) simulation time vs. R. $R = 20$ can be a good parameter for this simulation condition.

Fig. 9 illustrates the probability of achieving the global minimum energy $P_s(t)$ and the corresponding TTS across different problem sizes N. These simulation results are based on 100 trials for five different cases. The parameters for these five cases, all with $R = 25$, are summarized in Table 3. The number of iterations is calculated using $(\mathrm{EC}/R/(\tau \cdot (\beta + 1)))$ summarized in Table 2. From the simulation results, it appears that case 2 rarely achieves the global minimum energy. This is likely due to it having only a single iteration, while all other cases have multiple iterations. When comparing case 1 and 3, case 3 attains a higher $P_s(t)$ due to its larger EC. In terms of TTS, case 1 demonstrates a smaller TTS for $N = 900$ and $N = 1,600$, suggesting that an EC of 20,000 may be too large for smaller problem sizes. When comparing case 3 and 4, case 4 achieves a better $P_s(t)$, even though both cases use the same EC, indicating that a larger τ might lead to



(a)

(b)

FIGURE 8. Time to solution (TTS) [s] versus R of SSQA in the GI problems with $P_T = 0.99$: (a) EC $= 20,000$ and (b) EC $= 40,000$. Based on the results, $R = 25$ can be a good parameter for SSQA. When EC is set to 40,000, some TTS values cannot be calculated and are missing in the figure because $P_s(t)$ equals 1 in the case of $N = 900$.

better convergence to the global minimum energy. Comparing case 4 and 5, case 4 achieves smaller TTS for $N = 1,600$ and $N = 2,500$, whereas case 5 shows a higher $P_s(t)$. Based on these simulation results, the optimal EC and τ can vary depending on the problem size N.

D. COMPARISONS

The proposed SSQA method is compared with traditional SA [14], QA [33], and SSA [24] for solving GI problems. To ensure a fair comparison, the SA and SSA methods are simulated on the same computer that is used for SSQA. In SA, the temperature parameter $T_0(t)$ is gradually decreased by Δ_{IT} according to the schedule $T_0(t+1) = 1/(1/T_0(t)+\Delta_{IT})$ at each cycle. During each of these cycles, the algorithm randomly flips a spin state and accepts the new state if the new energy (E_{new}) is lower than the current energy (E_{cur}), or with a probability of $\exp(-(E_{new} - E_{cur})/T_0(t))$ if it is higher. The SA process starts with an initial temperature of 1,000 and ends with a final temperature of 0.1. On the other hand, in SSA, a pseudoinverse temperature parameter $I_0(t)$ is gradually increased for each iteration. I_0 follows the update rule $I_0(t + 1) = (1/\beta) \cdot I_0(t)$ every τ cycles, ranging from $I_{0min} = 1$ to $I_{0max} = 16$ with parameters $\tau = 10$ and $\beta = 0.5$. For SSA, this means that each iteration consists of 50 cycles. Another parameter, specific to SSA, is $n_{rnd} = 1$, as suggested by [24]. For QA, we refer to and compare with the experimental results presented in [33].

Table 4 summarizes the performance of different methods in solving GI problems. In the case of QA, a 504-qubit D-Wave Two machine was used, and an execution time t of 1 second was allocated. Both SA and SSA are simulated for a total of $40,000$ cycles. SSQA, in contrast, is configured to run for SC $= 1,600$ and $R = 25$, which effectively corresponds to EC $= 40,000$, as explained in Eq. (9). For SA, SSA, and SSQA, $P_s(t)$ values are obtained through an average of 100 trials. ''-'' indicates instances where the TTS values could not be computed due to the probability of success, $P_s(t)$, being zero. ''N/A'' denotes the absence of corresponding data in the referenced work [33], where evaluations were conducted up to a maximum of 400 nodes.

When comparing SSA and SSQA, the traditional SA method requires less computational cost as it employs a serial spin-update process. $P_s(t)$ significantly decreases as the problem size N increases, leading to an increase in TTS. The selected 40,000 simulation cycles appear to be insufficient for the SA method to converge for large-scale problems. In QA, the D-Wave machine can process approximately 500 spins (bits) for combinatorial optimization problems, but it only supports neighborhood connections between spins, as reported in [33]. Given that the GI problems in this study are modeled using fully connected spins, adjustments to the model are necessary for it to be compatible with the QA machine. As such, the QA method was only applied to GI problems with 25 spins. Interestingly, a new D-Wave machine, known as D-Wave 5000, has been released and is capable of handling 5,000 spins [22]. However, as of now, there are no available reports on the performance of the new machine concerning GI problems, and the issue with fully connected spins might still persist due to hardware limitations.

In contrast, SSA is able to solve considerably larger problems than SA and QA, and it also exhibits a smaller TTS compared to SSQA for small-scale problems. However, as the problem size N increases, SSQA outperforms SSA by achieving higher $P_s(t)$ values and smaller TTS. For instance, with $N = 2,025$, SSQA reduces the TTS by a remarkable 91.4% compared to SSA. The difference in TTS between SSA and SSQA continues to widen as N grows. Consequently, the proposed SSQA algorithm demonstrates to be significantly more effective than SSA when tackling large-scale combinatorial optimization problems. Compared with SA, SSQA can solve 25-times larger nodes with similar $P_s(t)$. Additionally, SSQA can handle problems with two

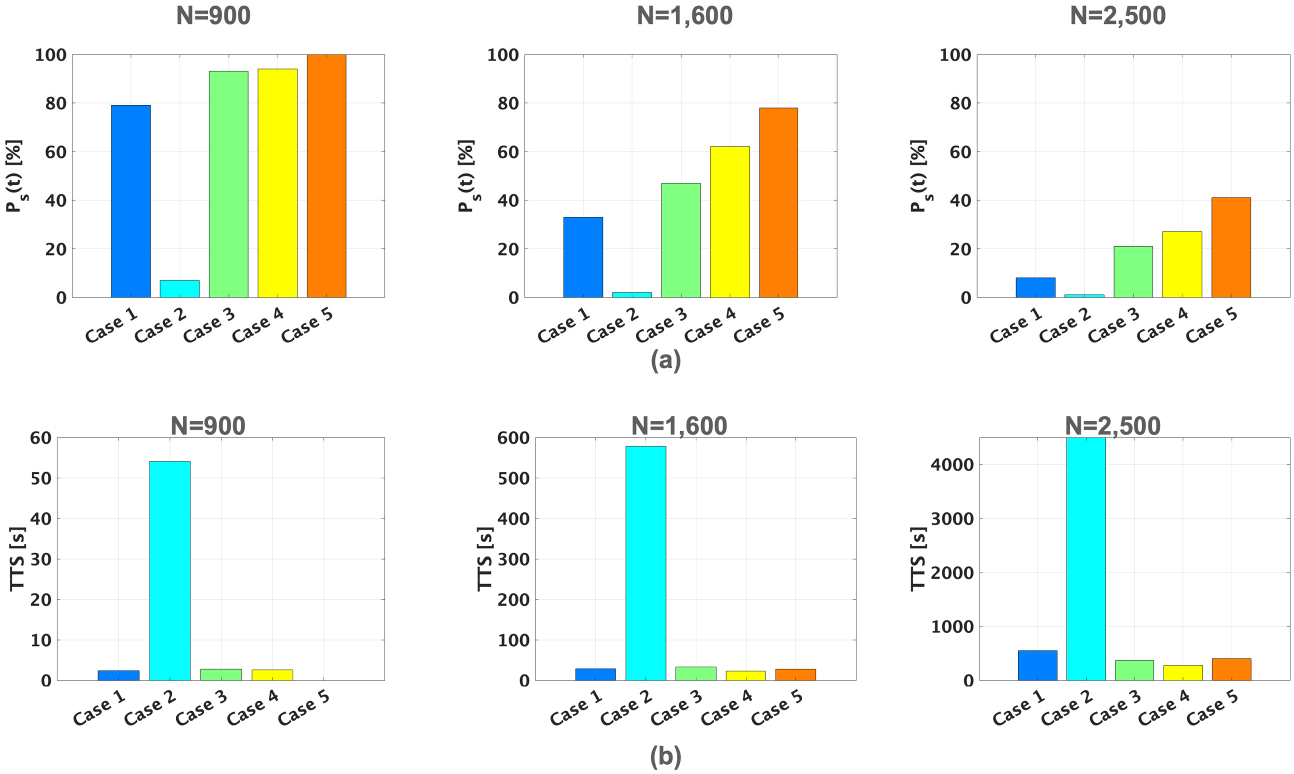


FIGURE 9. Probability of achieving the global minimum energy $P_S(t)$ and the corresponding TTS using SSQA with $R = 25$ for $N = 900$, $N = 1,600$, and $N = 2,500$. Note that case 2, which uses only one iteration, rarely reaches the global minimum energy. Additionally, the optimal EC and τ can vary depending on the problem size N.

TABLE 4. Performance comparisons in GI problems with $P_T = 0.99$.

N	SA [14]			QA [33]			SSA [24]			SSQA (proposed)		
	$P_s(t)$ [%]	t [s]	TTS [s]	$P_s(t)$ [%]	t [s]	TTS [s]	$P_s(t)$ [%]	t [s]	TTS [s]	$P_s(t)$ [%]	t [s]	TTS [s]
25	100	0.0876	≈ 0	98	1	1.17	100	0.0593	≈ 0	100	0.131	$\sim$0
100	43	0.123	1.01	0	1	-	100	0.121	≈ 0	100	0.198	≈ 0
400	2	0.649	148	0	1	-	100	0.717	≈ 0	100	0.836	≈ 0
625	3	1.35	203	N/A	N/A	N/A	92	1.48	2.71	100	1.61	≈ 0
1225	0	4.99	-	N/A	N/A	N/A	49	5.30	36.2	95	5.42	8.33
2025	0	19.4	-	N/A	N/A	N/A	6	22.7	1690	51	22.7	146
2500	0	36.1	-	N/A	N/A	N/A	0	45.6	-	41	46.4	405

orders of magnitude larger number of spins compared to QA, making it an extremely powerful tool for solving large and complex GI problems.

To compare with SA in terms of TTS in larger nodes, SA is simulated for a total of 800,000 cycles, which is 20 times longer than that in Table 4. The longer cycles lead to finding the global minimum energy and calculating TTS for larger nodes. Fig. 10 shows TTS versus the number of nodes in GI problems for 100 trials with $P_T = 0.99$ in SA, SSA and SSQA. The TTS values of SSA and SSQA are equivalent to that in Table 4. The TTS difference between SA and SSQA increases as the number of nodes increases. For $N = 2,025$, the TTS value of SA is 62,022 seconds, which is 423 times larger than that of SSQA.

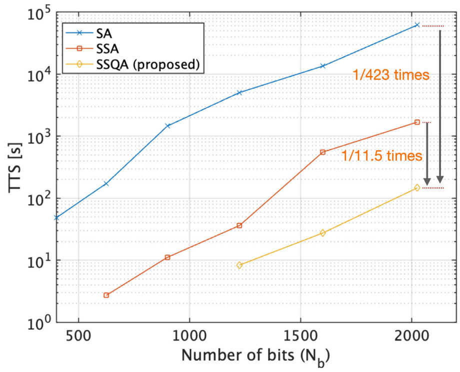


FIGURE 10. TTS versus number of nodes in GI problems in GI problems or 100 trials with $P_T = 0.99$ in SA, SSA and SSQA.

As outlined in Section I, while there are numerous optimization algorithms beyond SA, our proposed SSQA algorithm falls within the SA category. Although alternative algorithms may offer broader applicability and potentially greater power in solving a wide array of optimization problems – not just combinatorial ones – they often require the implementation of complex equations. Such complexity poses significant challenges for hardware integration. In contrast, this study concentrates on advancing a rapid SA-based approach, specifically through SSQA, tailored for large-scale combinatorial optimization challenges. The simplicity and efficiency of SSQA not only align with our objective of facilitating swift SA hardware acceleration but also underscore its potential as a practical tool in contexts demanding high computational efficiency and scalability.

V. CONCLUSION

In this paper, we present SSQA, a new method devised for tackling large-scale combinatorial optimization problems. SSQA employs an innovative spin-state update algorithm rooted in integral stochastic computing, which uses randomization and approximation, making it adept at solving complex optimization problems. We have experimentally evaluated the impact of varying the number of replicas on the performance through simulations. The findings suggest that using 25 replicas is particularly effective for GI problems, a common class of combinatorial optimization problems. Additionally, SSQA is compared to conventional SSA, QA using a 504-qubit D-Wave Two machine, and traditional SA. SSQA exhibits one to two orders of magnitude smaller TTS compared to SSA and SA, respectively. It can also solve problems with roughly 100 times more nodes than QA and 25 times more nodes than SA.

Looking ahead, the development of a large-scale hardware implementation of SSQA could be promising, as it holds the potential to be a fast and efficient solver for real-world combinatorial optimization challenges. Implementing SSQA on an FPGA is a straightforward method for accelerating the speed, as the previous SSA has been efficiently realized using FPGA technology [29]. Additionally, GPUs could offer an alternative means for rapid processing, since SSQA operates in a parallel manner.

REFERENCES

[1] J. Handl, D. B. Kell, and J. Knowles, "Multiobjective optimization in bioinformatics and computational biology," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 4, no. 2, pp. 279–292, Apr. 2007.

[2] A. E. Ezugwu, A. K. Shukla, R. Nath, A. A. Akinyelu, J. O. Agushaka, H. Chiroma, and P. K. Muhuri, "Metaheuristics: A comprehensive overview and classification along with bibliometric analysis," *Artif. Intell. Rev.*, vol. 54, no. 6, pp. 4237–4316, Aug. 2021, doi: 10.1007/s10462-020-09952-0.

[3] A. Nonut, Y. Kanokmedhakul, S. Bureerat, S. Kumar, G. G. Tejani, P. Artrit, A. R. Yildiz, and N. Pholdee, "A small fixed-wing UAV system identification using metaheuristics," *Cogent Eng.*, vol. 9, no. 1, Dec. 2022, Art. no. 2114196, doi: 10.1080/23311916.2022.2114196.

[4] P. Singh, R. Kottath, and G. G. Tejani, "Ameliorated follow the leader: Algorithm and application to truss design problem," *Structures*, vol. 42, pp. 181–204, Aug. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352012422004672

[5] S. Kumar, G. G. Tejani, N. Pholdee, and S. Bureerat, "Performance enhancement of meta-heuristics through random mutation and simulated annealing-based selection for concurrent topology and sizing optimization of truss structures," *Soft Comput.*, vol. 26, no. 12, pp. 5661–5683, Jun. 2022, doi: 10.1007/s00500-022-06930-2.

[6] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003, doi: 10.1145/937503.937505.

[7] S. Kumar, P. Jangir, G. G. Tejani, M. Premkumar, and H. H. Alhelou, "MOPGO: A new physics-based multi-objective plasma generation optimizer for solving structural optimization problems," *IEEE Access*, vol. 9, pp. 84982–85016, 2021.

[8] S. Kumar, P. Jangir, G. G. Tejani, and M. Premkumar, "MOTEO: A novel physics-based multiobjective thermal exchange optimization algorithm to design truss structures," *Knowl.-Based Syst.*, vol. 242, Apr. 2022, Art. no. 108422. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095070512200171X

[9] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: A comprehensive survey," *Artif. Intell. Rev.*, vol. 52, no. 4, pp. 2191–2233, Dec. 2019, doi: 10.1007/s10462-017-9605-z.

[10] A. Slowik and H. Kwasnicka, "Evolutionary algorithms and their applications to engineering problems," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12363–12379, Aug. 2020, doi: 10.1007/s00521-020-04832-8.

[11] A. G. Gad, "Particle swarm optimization algorithm and its applications: A systematic review," *Arch. Comput. Methods Eng.*, vol. 29, no. 5, pp. 2531–2561, Aug. 2022, doi: 10.1007/s11831-021-09694-4.

[12] H. R. Lourenčo, O. C. Martin, and T. Stützle, *Iterated Local Search*. Boston, MA, USA: Springer, 2003, pp. 320–353, doi: 10.1007/0-306-48056-5_11.

[13] P. Hansen and N. Mladenović, *Variable Neighborhood Search*. Boston, MA, USA: Springer, 2003, pp. 145–184, doi: 10.1007/0-306-48056-5_6.

[14] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[15] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: An experimental evaluation; Part II, graph coloring and number partitioning," *Oper. Res.*, vol. 39, no. 3, pp. 378–406, Jun. 1991.

[16] P. Serafini, "Simulated annealing for multi objective optimization problems," in *Multiple Criteria Decision Making*, G. H. Tzeng, H. F. Wang, U. P. Wen, and P. L. Yu, Eds., New York, NY, USA: Springer, 1994, pp. 283–292.

[17] T. G. J. Myklebust, "Solving maximum cut problems by simulated annealing," 2015, *arXiv:1505.03068*.

[18] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, "A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 303–309, Jan. 2016.

[19] H. Gyoten, M. Hiromoto, and T. Sato, "Enhancing the solution quality of hardware Ising-model solver via parallel tempering," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*. New York, NY, USA: Association for Computing Machinery, Nov. 2018, pp. 1–8, doi: 10.1145/3240765.3240806.

[20] H. Goto, K. Tatsumura, and A. R. Dixon, "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems," *Sci. Adv.*, vol. 5, no. 4, Apr. 2019, Art. no. eaav2372, doi: 10.1126/sci-adv.aav2372.

[21] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, "Evidence for quantum annealing with more than one hundred qubits," *Nature Phys.*, vol. 10, no. 3, pp. 218–224, Mar. 2014, doi: 10.1038/nphys2900.

[22] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, "Evidence for quantum annealing with more than one hundred qubits," *Nature Phys.*, vol. 10, pp. 218–224, 2014.

[23] H. Neven. (Jan. 2016). *When Can Quantum Annealing Win?* [Online]. Available: https://ai.googleblog.com/2015/12/when-can-quantum-annealing-win.html

[24] N. Onizawa, K. Katsuki, D. Shin, W. J. Gross, and T. Hanyu, "Fast-converging simulated annealing for Ising models based on integral stochastic computing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 10999–11005, Dec. 2023.

[25] K. Camsari, R. Faria, B. Sutton, and S. Datta, "Stochastic *p*-bits for invertible logic," *Phys. Rev. X*, vol. 7, Jul. 2017, Art. no. 031014.

[26] B. R. Gaines, "Stochastic computing systems," *Adv. Inf. Syst.*, vol. 2, no. 2, pp. 37–172, 1969.

[27] B. D. Brown and H. C. Card, "Stochastic neural computation. I. Computational elements," *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 891–905, Sep. 2001.

[28] K. Y. Camsari, B. M. Sutton, and S. Datta, "P-bits for probabilistic spin logic," *Appl. Phys. Rev.*, vol. 6, no. 1, Mar. 2019, Art. no. 011305.

[29] D. Shin, N. Onizawa, W. J. Gross, and T. Hanyu, "Memory-efficient FPGA implementation of stochastic simulated annealing," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 13, no. 1, pp. 108–118, Mar. 2023.

[30] M. Suzuki, "Relationship between d-dimensional quantal spin systems and (d+1)-dimensional Ising systems: Equivalence, critical exponents and systematic approximants of the partition function and spin correlations," *Prog. Theor. Phys.*, vol. 56, no. 5, pp. 1454–1469, Nov. 1976, doi: 10.1143/ptp.56.1454.

[31] G. E. Santoro, R. Martoňák, E. Tosatti, and R. Car, "Theory of quantum annealing of an Ising spin glass," *Science*, vol. 295, no. 5564, pp. 2427–2430, Mar. 2002, doi: 10.1126/science.1068774.

[32] A. Lucas, "Ising formulations of many NP problems," *Frontiers Phys.*, vol. 2, p. 5, Feb. 2014, doi: 10.3389/fphy.2014.00005.

[33] K. M. Zick, O. Shehab, and M. French, "Experimental quantum annealing: Case study involving the graph isomorphism problem," *Sci. Rep.*, vol. 5, no. 1, p. 11168, Jun. 2015, doi: 10.1038/srep11168.

[34] S. Burer, R. D. C. Monteiro, and Y. Zhang, "Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs," *SIAM J. Optim.*, vol. 12, no. 2, pp. 503–521, Jan. 2002.

[35] G. Reinelt, "TSPLIB—A traveling salesman problem library," *ORSA J. Comput.*, vol. 3, no. 4, pp. 376–384, Nov. 1991.

[36] S. C. Smithson, N. Onizawa, B. H. Meyer, W. J. Gross, and T. Hanyu, "Efficient CMOS invertible logic using stochastic computing," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 6, pp. 2263–2274, Jun. 2019.

[37] N. Onizawa, S. C. Smithson, B. H. Meyer, W. J. Gross, and T. Hanyu, "In-hardware training chip based on CMOS invertible logic for machine learning," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 5, pp. 1541–1550, May 2020.

[38] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, "Boltzmann machines: Constraint satisfaction networks that learn," Dept. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-84-119, 1984.

[39] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, "VLSI implementation of deep neural network using integral stochastic computing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2688–2699, Oct. 2017.

[40] V. C. Gaudet and A. C. Rapley, "Iterative decoding using stochastic computation," *Electron. Lett.*, vol. 39, no. 3, pp. 299–301, 2003.

[41] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 3, pp. 449–462, Mar. 2014.

[42] Y. Liu and K. K. Parhi, "Architectures for recursive digital filters using stochastic computing," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3705–3718, Jul. 2016.

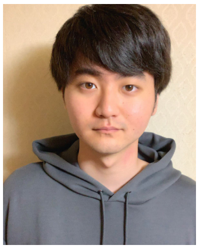
[43] K. Y. Camsari, S. Chowdhury, and S. Datta, "Scalable emulation of sign-problem–free Hamiltonians with room-temperature p-bits," *Phys. Rev. Appl.*, vol. 12, no. 3, p. 5, Sep. 2019, doi: 10.1103/physrevapplied.12.034061.

[44] N. Onizawa. (2023). *SSQA-for-GI-MATLAB*. [Online]. Available: https://github.com/nonizawa/SSQA-for-GI-MATLAB

[45] T. Albash and D. A. Lidar, "Demonstration of a scaling advantage for a quantum annealer over simulated annealing," *Phys. Rev. X*, vol. 8, no. 3, Jul. 2018, Art. no. 031016, doi: 10.1103/physrevx.8.031016.

NAOYA ONIZAWA (Member, IEEE) received the D.E. degree in electrical and communication engineering from Tohoku University, Japan, in 2009.

He was a Postdoctoral Fellow with the University of Waterloo, Canada, in 2011, and McGill University, Canada, from 2011 to 2013. He was a Visiting Associate Professor with the University of Southern Brittany and IMT Atlantique, France, in 2015 and 2022, respectively. He is currently an Associate Professor and a Distinguished Researcher with the Research Institute of Electrical Communication, Tohoku University. His main research interests and activities include energy-efficient VLSI design based on asynchronous circuits and probabilistic computation, and their applications, such as brain-like computers. He received the Best Paper Award from the 2010 IEEE ISVLSI, the Best Paper Finalist from 2014 IEEE ASYNC, the Kenneth C. Smith Early Career Award for Microelectronics Research from 2016 IEEE ISMVL, the MEXT Young Scientists' Prize, Japan, in 2020, and the 16th Aoba Engineering Promotion Society Award, in 2022.



RYOMA SASAKI received the B.E. degree in electrical and communication engineering from Tohoku University, Japan, in 2022, where he is currently pursuing the M.E. degree with the Research Institute of Electrical Communication. His main research interests and activities include stochastic simulated annealing and related applications.

DUCKGYU SHIN received the B.E. and M.E. degrees in electrical and communication engineering from Tohoku University, Japan, in 2019 and 2021, respectively, where he is currently pursuing the D.E. degree with the Research Institute of Electrical Communication. His main research interests and activities include high-speed VLSI design based on stochastic simulated annealing and related applications, such as the implementation of a fast Ising processor.

WARREN J. GROSS received the B.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1996, and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada, in 1999 and 2003, respectively.

Currently, he is a James McGill Professor and Chair of the Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada. His research interests include the design and implementation of signal-processing systems and custom computer architectures. He served as the Chair of the IEEE Signal Processing Society Technical Committee on Design and Implementation of Signal Processing Systems. He has served as the General Co-Chair for IEEE GlobalSIP 2017 and IEEE SiPS 2017 and the Technical Program Co-Chair for SiPS 2012. He has also served as an Organizer for the Workshop on Polar Coding in Wireless Communications at WCNC 2017, the Symposium on Data Flow Algorithms and Architecture for Signal Processing Systems (GlobalSIP 2014), and the IEEE ICC 2012 Workshop on Emerging Data Storage Technologies. He served as an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING and a Senior Area Editor. He is a licensed Professional Engineer in the Province of Ontario.

TAKAHIRO HANYU (Senior Member, IEEE) received the B.E., M.E., and D.E. degrees in electronic engineering from Tohoku University, Sendai, Japan, in 1984, 1986 and 1989, respectively.

He is currently a Professor with the Research Institute of Electrical Communication, Tohoku University. His general research interests include nonvolatile logic circuits and their applications to ultra-low-power and/or highly dependable VLSI processors, and post-binary computing and its application to brain-inspired VLSI systems. He received the Sakai Memorial Award from the Information Processing Society of Japan, in 2000; the Judge's Special Award at the 9th LSI Design of the Year from the Semiconductor Industry News of Japan, in 2002; the Special Feature Award at the University LSI Design Contest from ASP-DAC, in 2007; the APEX Paper Award of Japan Society of Applied Physics, in 2009; the Excellent Paper Award of IEICE, Japan, in 2010; the Ichimura Academic Award, in 2010; the Best Paper Award of IEEE ISVLSI 2010; the Paper Award of SSDM 2012; the Best Paper Finalist of IEEE ASYNC 2014; and the Commendation for Science and Technology by MEXT, Japan, in 2015.