DOI: 10.15514/ISPRAS-2026-38(4)-23

Quantizing Ising model matrices for combinatorial optimization on RISC-V with Lichee Pi 4A

A.M. Bratenkov, ORCID: 0009-0003-6203-3430 <bratenkov.am@edu.spbstu.ru>
N.O. Stepina, ORCID: 0009-0001-4740-637X <gubenko_no@spbstu.ru>
I.V. Nikiforov, ORCID: 0000-0003-0198-1886 <nikiforov_iv@spbstu.ru>
O.A. Yusupova, ORCID: 0000-0002-1757-1657 <yusupova_oa@spbstu.ru>

Peter the Great St. Petersburg Polytechnic University,
29, Polytechnicheskaya st., St. Petersburg, 195251, Russia.

Abstract. This study addresses problem of low efficiency of software implementation of combinatorial optimization algorithms on RISC-V processors. The main obstacle is the quadratic growth of the Ising model interaction matrix. Using double-precision floating-point representation leads to high cache miss rates and degrades performance. The aim of the study is to improve efficiency of solving combinatorial problems on the RISC-V platform by reducing the bit width of the interaction matrix using quantization methods. The proposed approach implements three classes of rounding methods: simple rounding, stochastic rounding, and eight diffusion rounding schemes. A software prototype is developed for the Lichee Pi 4A board, integrating matrix quantization with simulated annealing adapted for integer data types. Experiments are conducted on three NP-hard problems: the traveling salesman problem, the knapsack problem, and the max-cut problem, with bit widths from 2 to 16 bits. The novelty lies in a systematic comparison of these methods on three classical NP-hard problems and in the analysis of their effectiveness depending on the problem structure and available bit width. The results show that quantization effectiveness depends on the problem structure. For the max-cut problem, diffusion rounding improves solution quality by 10% compared to the unrounded matrix. For the knapsack problem, diffusion methods offer no advantage over simple rounding, while stochastic rounding requires at least 8 bits. For the traveling salesman problem, diffusion rounding is beneficial only at 2 bits. Performance evaluation on Lichee Pi 4A shows that switching from double-precision to 8-bit integer matrices reduces execution time by a factor of 3.6–4.7 and decreases cache miss rates from 2.64% to 0.13–0.25%. The best performance is achieved with int8 matrix and int32 accumulator. The findings enable solving larger combinatorial problems on resource-constrained RISC-V platforms.

Keywords: RISC-V; Lichee Pi 4A; combinatorial optimization; Ising model; matrix quantization; error diffusion rounding; stochastic rounding; simulated annealing.

For citation: Bratenkov A.M., Stepina N.O., Nikiforov I.V., Yusupova O.A. Quantizing Ising model matrices for combinatorial optimization on RISC-V with Lichee Pi 4A. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 4, part 2, 2026, pp. 143-160. DOI: 10.15514/ISPRAS-2026-38(4)-23.

Квантование матриц модели Изинга для комбинаторной оптимизации на RISC-V с использованием Lichee Pi 4A

А. М. Братенков, ORCID: 0009-0003-6203-3430 <bratenkov.am@edu.spbstu.ru>
Н. О. Степина, ORCID: 0009-0001-4740-637X <gubenko_no@spbstu.ru>
И. В. Никифоров, ORCID: 0000-0003-0198-1886 <nikiforov_iv@spbstu.ru>
О. А. Юсупова, ORCID: 0000-0002-1757-1657 <yusupova_oa@spbstu.ru>

Санкт-Петербургский политехнический университет Петра Великого,
195251, Россия, Санкт-Петербург, Политехническая улица, д. 29.

Аннотация. В работе рассматривается проблема низкой эффективности программной реализации алгоритмов комбинаторной оптимизации на RISC-V процессорах. Основное препятствие – квадратичный рост матрицы взаимодействия модели Изинга, который при использовании чисел с плавающей точкой двойной точности приводит к высокому уровню кэш-промахов и падению производительности. Цель исследования – повышение эффективности решения комбинаторных задач на RISC-V платформе за счёт уменьшения разрядности матрицы с использованием методов квантования. Предложенный подход реализует три класса методов округления: простое, стохастическое и восемь схем диффузионного округления. Новизна работы заключается в систематическом сравнении этих методов на трёх классических NP-трудных задачах, а также в анализе их эффективности в зависимости от структуры задачи и доступной разрядности данных. Разработан программный прототип для платы Lichee Pi 4A, интегрирующий квантование матрицы с алгоритмом имитации отжига, который адаптирован для целочисленных типов данных. Эксперименты проведены на трёх NP-трудных задачах: коммивояжёра, о рюкзаке и максимального разреза графа, с разрядностью от 2 до 16 бит. Результаты показывают, что эффективность квантования зависит от структуры задачи. Для задачи максимального разреза диффузионное округление улучшает качество решения на 10% по сравнению с неквантованной матрицей. Для задачи о рюкзаке диффузионные методы не дают преимущества перед простым округлением, а стохастическое требует не менее 8 бит. Для задачи коммивояжёра диффузионное округление эффективно только на 2 битах. Оценка производительности на Lichee Pi 4A показывает, что переход с матрицы двойной точности на 8-битную целочисленную сокращает время выполнения в 3,6–4,7 раза и снижает долю кэш-промахов с 2,64% до 0,13–0,25%. Наилучшая производительность достигается с матрицей int8 и аккумулятором int32. Результаты позволяют решать задачи комбинаторной оптимизации большего размера на RISC-V системах с ограниченными ресурсами.

Ключевые слова: RISC-V; Lichee Pi 4A; комбинаторная оптимизация; модель Изинга; квантование матрицы; диффузионное округление; стохастическое округление; имитация отжига.

Для цитирования: Братенков А.М., Степина Н.О., Никифоров И.В., Юсупова О.А. Квантование матриц модели Изинга для комбинаторной оптимизации на RISC-V с использованием Lichee Pi 4A. Труды ИСП РАН, том 38, вып. 4, часть 2, 2026 г., стр. 143–160 (на английском языке). DOI: 10.15514/ISPRAS–2026–38(4)–23.

1. Introduction

The RISC-V architecture is an open and free instruction set architecture [1] based on the principles of reduced instruction set computing. Unlike proprietary architecture such as x86 (Intel/AMD) and ARM, which are controlled by commercial organizations, RISC-V is developed under the oversight of the non-profit organization RISC-V International and is available for use without payment. Due to its modularity, extensibility, and openness, RISC-V is being actively adopted across various domains – from embedded systems to high-performance computing [2-4].

High-performance computing (HPC) plays a key role in solving complex scientific and engineering problems, including physical process modeling, weather forecasting, materials development, as well as tasks in artificial intelligence and machine learning [5].

Among the many application areas of HPC, combinatorial optimization problems are particularly important. Such problems occur in logistics (optimal vehicle routing), planning (scheduling),

telecommunications (optimal network topology), as well as in machine learning (optimal configuration of neural networks). A key feature of combinatorial problems is that they are NP-hard: the solution space grows exponentially with problem size, making exhaustive search practically impossible even for relatively small dimensions [6].

We use metaheuristics as approximate methods to solve NP-hard combinatorial optimization problems efficiently. These do not guarantee finding the global optimum but allow obtaining a solution close to the optimal one. Among such methods, notable examples include genetic algorithms, ant colony optimization algorithms, tabu search, and particularly simulated annealing, which has been successfully applied to a wide range of optimization problems [7].

For practical implementation on computing platforms, including the RISC-V architecture, the application of simulated annealing within the Ising model framework becomes especially important. The Ising model is a mathematical model from statistical physics that describes a system of interacting spins. Interest in the Ising model in the context of optimization stems [8] from its universality: any problem from the NP class can be reduced to an Ising energy minimization problem. This enables a unified approach to solving various combinatorial problems [9].

However, practical implementation of methods based on the Ising model is limited by quadratic data growth. This creates substantial limitations:

- for software implementation on general-purpose processors, since large matrices do not fit in cache memory, leading to a multiple increase in the number of accesses to RAM and a decrease in performance [10];
- for specialized hardware solutions (annealing processors, FPGA implementations), physical constraints on chip area and power consumption impose strict requirements on the bit-width for coefficient representation (typically 4-16 bits) [11].

A promising way to overcome these limitations is quantization of the interaction matrix – reducing the bit-width of coefficient representation to integer types such as int8 or int16. However, simple rounding of real-valued coefficients distorts the energy landscape of the problem: the global minimum of the original problem may no longer be a minimum for the quantized version, and spurious local minima may appear [12].

To minimize the loss of solution quality during quantization, advanced rounding methods can be used:

- diffusion rounding, where the quantization error of each element is distributed to neighboring elements, preserving the integral information of the matrix;
- stochastic rounding, where rounding is performed probabilistically, which preserves the statistical properties of the data.

This work aims to improve the efficiency of solving combinatorial optimization problems on the RISC-V architecture through the use of advanced quantization methods for the Ising model interaction matrix.

To achieve this goal, the study addresses the following tasks:

1) implement quantization methods for the Ising model interaction matrix, including simple rounding, stochastic rounding, and eight diffusion rounding schemes (Floyd-Steinberg, Jarvis-Judice-Ninke, Stucki, Atkinson, Burkes, Sierra-3, Sierra-2, Sierra-Lite);
2) develop a software prototype on the RISC-V architecture that integrates matrix quantization with simulated annealing adapted for quantized integer data types;
3) experimentally evaluate the impact of different rounding methods and bit widths (2–16 bits) on solution quality for three NP-hard problems: the traveling salesman problem, the knapsack problem, and the max-cut problem;
4) measure the performance of the quantized simulated annealing algorithm on RISC-V hardware (Lichee Pi 4A) across different data type configurations, analyzing execution time and cache miss rates.

This requires a detailed analysis of the theoretical foundations, including the formulation of combinatorial problems in terms of the Ising model and the simulated annealing algorithm. These aspects are detailed in the next section. The novelty of this work lies in a systematic comparison of these rounding methods on three classical NP-hard problems (knapsack, max-cut, traveling salesman) and in the analysis of their effectiveness depending on the problem structure and available bit width.

2. Related Work on Ising Model Quantization

This section presents an analysis of modern research on solving NP-hard combinatorial optimization problems using the Ising model, as well as data quantization methods and their application to the RISC-V architecture. This section summarizes key works and their main contributions and limitations.

In [13], the authors propose methods to reduce the required bit depth of coefficients in the Ising model by adding auxiliary spins, which reduces the coefficient bit-width without distorting the objective function. However, this approach increases the problem dimensionality, which amplifies the quadratic growth of the interaction matrix and leads to higher memory and computational requirements, limiting its applicability to large-scale problems.

In the context of annealers and Ising machines, the work most closely related to this paper is [14], where diffusion rounding of the interaction matrix was first proposed for a CMOS annealer. The authors demonstrate improved performance on a 512-spin knapsack problem: using a matrix quantized to 7–9 bits and processed with the Floyd–Steinberg method, they achieve constraint satisfaction where simple rounding fails to produce a single feasible solution. However, this study is limited to a single problem (0–1 knapsack) and does not investigate the impact of different error diffusion schemes on solution quality.

In [15], the DSAPS (Dual Scalable Annealing Processing System) is proposed, which enables simultaneous scaling of both the number of spins and the interaction bit-width. The authors show that for the knapsack problem, increasing the bit-width from 10 to 37 bits reduces the weight constraint violation from 99% to 0.73%, confirming the importance of high bit-width for accurate constraint representation. However, this work uses only simple rounding and does not explore other methods.

In the context of solving NP-hard combinatorial optimization problems, the combination of metaheuristic methods and the Ising model formalism occupies a special place. As shown in [16], metaheuristics, including simulated annealing, allow finding near-optimal solutions without requiring complete knowledge of the search space structure, which makes them indispensable for intractable problems. On the other hand, as established in [17], any problem from the NP class can be reduced to an Ising energy minimization problem. This means that classical NP-hard problems such as the traveling salesman problem, the knapsack problem, and the maximum cut problem can be formulated within a unified formalism. However, practical implementation of this unified approach faces two major obstacles: first, the quadratic growth of the interaction matrix leads to memory and cache inefficiency on general-purpose processors; second, hardware implementations (annealing processors) impose strict limitations on the bit width of coefficients, typically 4-16 bits. These limitations necessitate the development of quantization methods that preserve solution quality while reducing bit width – a gap that existing studies have not fully addressed, particularly for RISC-V platforms.

3. Problem Statement and Theoretical Background

To solve the research tasks outlined in this paper, three key aspects must be considered: the formulation of combinatorial optimization problems, the Ising model as a unified mathematical

formalism for representing such problems, and the simulated annealing algorithm adapted to operate with quantized data. This section provides a sequential exposition of these issues.

3.1 Combinatorial optimization

Combinatorial optimization is a class of mathematical problems in which the goal is to find an optimal solution from a finite discrete set [18]. Formally, a combinatorial optimization problem can be defined as a triple (*X, F, f*), where:

- X is a finite set of feasible solutions (a discrete search space);
- $F \subseteq X$ is a subset of feasible solutions that satisfy given constraints;
- $f: F \to \mathbb{R}$ is an objective function (a quality criterion).

The problem consists of finding an element $x^* \in F$ such that $f(x^*) \leq f(x)$ for a minimization problem (or $f(x^*) \geq f(x)$ for a maximization problem) for all $x \in F$.

A main characteristic of combinatorial problems is the discreteness of the solution space. Unlike continuous optimization problems, where variables can take any value within an interval, in combinatorial optimization variables are typically integer or Boolean. The solution is often a combinatorial object: a permutation (traveling salesman problem), a subset (knapsack problem), or a graph (max-cut problem).

The cardinality of the feasible solution set $|F|$ grows exponentially or factorially as the problem dimension increases [19]. For example, for the traveling salesman problem with n cities, the number of possible tours is $(n-1)!/2$, which makes exhaustive search practically infeasible even for $n > 20$. This fundamental property places a broad class of combinatorial problems among NP-hard problems, for which no polynomial-time algorithm exists.

3.2 The Ising model

The Ising model is a mathematical model from statistical physics that describes a system of interacting spins located at the nodes of a crystal lattice [20]. In the classical Ising model, the system consists of N discrete variables σ_i (spins, each of which can take one of two values $\{-1, +1\}$, where $+1$ corresponds to a spin "up" and -1 to a spin "down". The configuration of the system is fully described by the vector $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_N)$.

The Hamiltonian (energy function) of the system is given by:

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

where:

- $J_{ij} \in \mathbb{R}$ is the interaction coefficient between spins i and j;
- $h_i \in \mathbb{R}$ is the interaction coefficient of the external magnetic field with spin i;
- $H(\sigma) \in \mathbb{R}$ is the energy of configuration σ.

The Ising model formalism has an important property: many classical NP-hard problems can be formulated within the Ising model, and the problem of finding an optimal solution reduces to energy minimization. This enables a unified approach to solving such problems using methods developed for spin systems, such as simulated annealing and specialized hardware accelerators known as annealing processors [21].

3.3 Simulated Annealing

Simulated annealing is a probabilistic metaheuristic algorithm for global optimization [22]. The algorithm is based on an analogy with the physical annealing process in metallurgy – a thermal

treatment of materials in which a substance is first heated to a high temperature and then gradually cooled, allowing it to reach a state with minimal free energy and a perfect crystal lattice.

In physical annealing, the atoms of a material are highly mobile at high temperatures and can transition between different energy states, including states with higher energy. As the system slowly cools, it tends toward a state of thermodynamic equilibrium at each temperature. When cooling is sufficiently slow, the system is highly likely to reach the global energy minimum [23].

Simulated annealing transfers this concept to the field of optimization by replacing:

- the spin configuration σ with the current solution x;
- the energy $H(\sigma)$ with the objective function of the optimization problem $f(x)$;
- the temperature T with a control parameter that determines the probability of accepting non-improving solutions.

At the core of simulated annealing lies the Metropolis algorithm, originally developed for simulating physical systems [24]. This algorithm generates a sequence of states distributed according to the Boltzmann distribution at a given temperature.

For an optimization problem, the Metropolis algorithm works as follows: given a current solution x a new solution x' is generated (typically by applying a small perturbation to the current solution). The change in the objective function $\Delta f = f(x') - f(x)$ is computed. If $\Delta f \leq 0$ (the solution improves), the transition is accepted unconditionally. If $\Delta f > 0$ (the solution worsens), the transition is accepted with probability:

$$\mathrm{P} = \exp\left(-\frac{\Delta f}{T}\right) \qquad (2)$$

where T is the current temperature. This probabilistic strategy allows the algorithm to escape local minima by accepting solutions that worsen the objective function in the early stages (at high T) while gradually reducing the probability of such transitions as the temperature decreases.

In the context of the Ising model, simulated annealing acquires a direct physical meaning: the algorithm simulates the thermal dynamics of a spin system. The temperature T in the algorithm corresponds to the actual physical temperature, and the objective function $f(\sigma) = H(\sigma)$ corresponds to the energy of the spin configuration.

Due to the local interaction structure of the Ising model, the energy change resulting from flipping a single spin can be computed locally, without recalculating the full Hamiltonian, which significantly improves the efficiency of the algorithm [25].

Consider a spin flip $\sigma_i \to -\sigma_i$. The energy change ΔE_i is computed based on the interactions of spin i with its neighbors and the external field:

$$\Delta E_i = E(-\sigma_i) - E(\sigma_i) = 2\sigma_i\left(\sum_{j\neq i} J_{ij}\,\sigma_j + h_i\right) \qquad (3)$$

The implementation of the simulated annealing algorithm used in this work is shown in Listing 1.

4. Quantization Methods for the Interaction Matrix

The practical implementation of combinatorial optimization methods based on the Ising model faces the problem of quadratic data growth as the number of spins N increases. The interaction matrix J contains N^2 elements, and the external field vector h contains N elements. Using the standard double-precision floating-point representation (64 bits), the required memory is $N^2 \cdot 8\ bytes + N \cdot 8\ bytes \approx 8N^2 bytes$.

Listing 1. Simulated annealing implementation.

```
template <typename SpinType, typename MatrixType, typename AccType>
std::vector<SpinType> simulated_annealing (
  const IsingModel<MatrixType>& im,
  double T_start,
  int LoopNum,
  double Tc
) {
  std::uniform_real_distribution<double> real_dist(0.0, 1.0);
  std::uniform_int_distribution<int> int_dist(0, 1);
  auto& gen = rng_sa_base;
  const auto& J = im.J; const auto& h = im.h;
  size_t N = im.h.size();
  std::vector<SpinType> spins(N);
  for (size_t i = 0; i < N; ++i) {
    spins[i] = (int_dist(gen) ? +1 : -1);
  }
  std::vector<AccType> LF(N);
  for (size_t i = 0; i < N; ++i) {
    AccType dEi = h[i];
    for (size_t j = 0; j < N; ++j) {
      dEi += J[i][j] * spins[j];
    }
    LF[i] = dEi;
  }
  double current_T = T_start;
  for (size_t loop = 0; loop < LoopNum; ++loop) {
    for (size_t i = 0; i < N; ++i) {
      AccType deltaEi = LF[i] * spins[i];
      if (deltaEi < 0 || real_dist(gen) < std::exp(-std::abs(deltaEi) / current_T)) {
        spins[i] *= -1;
        for (size_t j = 0; j < N; ++j) {
          LF[j] += 2 * J[i][j] * spins[i];
        }
      }
    }
    current_T *= Tc;
  }
  return spins;
}
```

For different problem dimensions, this imposes the following constraints:

- at $N = 10^3$: ≈ 8 MB – fits in L2/L3 cache of most modern processors;
- at $N = 10^4$: ≈ 800 MB – does not fit in cache, requires main memory access;
- at $N = 10^5$: ≈ 80 GB – exceeds the main memory capacity of typical single-board computers.

As the problem dimension increases, two interrelated problems arise. For software implementation on general-purpose processors, when the dimension $N \geq 10^4$, the interaction matrix no longer fits in the processor cache. Since the simulated annealing algorithm involves random access to matrix elements (each spin flip requires access to row i of matrix J), this leads to a significant increase in cache misses. Each main memory access takes two to three orders of magnitude longer than a cache access, making algorithm performance limited not by processor clock speed but by memory subsystem bandwidth. For specialized hardware solutions, physical constraints on chip area and power consumption impose strict requirements on the bit width used to represent interaction coefficients. Typically, coefficients are represented as integer types with bit widths ranging from 4 to 16 bits. This makes quantization (bit-width reduction) not merely desirable but a necessary condition for implementation [26-27].

However, directly rounding real-valued coefficients to low-bit-width integers distorts the energy landscape of the problem. The global minimum of the original problem may no longer be a minimum for the quantized version, and spurious local minima that did not exist in the original problem may appear. This can significantly degrade the quality of solutions obtained using the quantized model.

To minimize the loss of solution quality during quantization, advanced rounding methods can be used, which reduce the distortion of the problem's energy landscape structure.

In contrast to previous works that are limited to a single error diffusion scheme or a single problem class, this study proposes and implements a comprehensive methodology for quantizing Ising model interaction matrices. The authors develop software implementations of eight error diffusion rounding schemes, as well as stochastic rounding and simple rounding. All methods are adapted for integer data types and integrated into a simulated annealing algorithm on the RISC-V platform. To the best of the authors' knowledge, this work provides the first systematic comparison of the impact of various rounding methods on solution quality for three NP-hard problems under different bit-width constraints.

4.1 Error Diffusion Rounding

Error diffusion rounding is a quantization method in which the rounding error of each element is not discarded but distributed to neighboring, as-yet-unprocessed elements. This method was first proposed in the field of image processing for halftoning problems, where the goal was to convert an image to a bitwise representation with minimal loss of visual quality.

The main idea of the method is to preserve the integral information of the matrix: the sum of the quantized elements, accounting for propagated error, approximates the sum of the original values.

The most well-known error diffusion algorithm is the Floyd–Steinberg algorithm [28], originally developed for converting images to halftone format. In this algorithm, the quantization error of the current pixel is distributed to neighboring pixels with weighting coefficients, as shown in Fig. 1.

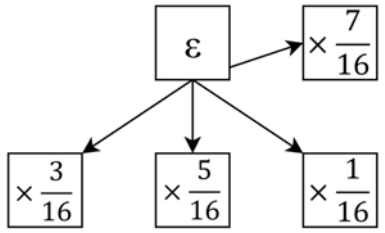

Fig. 1. Floyd-Steinberg weighting coefficients.

The error diffusion rounding process is carried out by sequentially traversing the matrix elements in a specific order. Each element is processed as follows:

1) the element is rounded to the nearest integer $\hat{J}_{ij} = round(J_{ij})$;
2) the quantization error $\varepsilon = J_{ij} - \hat{J}_{ij}$ is computed;
3) the error is propagated to neighboring, as-yet-unprocessed elements with specified weighting coefficients $J_{kl} = J_{kl} + \varepsilon \cdot w_{kl}$, where w_{kl} are the weighting coefficients.

When quantizing Ising model interaction matrices, the symmetry of the matrix must be taken into account, since $J_{ij} = J_{ji}$. In this work, all elements are traversed sequentially, after which the upper triangle of the matrix is mirrored to the lower triangle. The order in which the elements are traversed also affects the error distribution. This work uses the standard traversal scheme (Fig. 2) – row-wise, left to right, top to bottom.

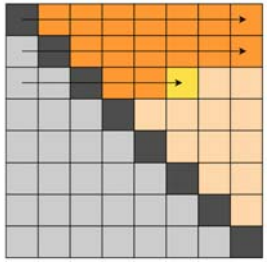

Fig. 2. Matrix traversal scheme (orange – processed, yellow – currently being processed, pink – not yet processed, gray – elements of the lower triangle, black – diagonal elements equal to zero).

In this work, eight variants of error diffusion rounding are implemented, differing in the error propagation scheme and coefficients (Fig. 3): Floyd-Steinberg (FS), Jarvis-Judice-Ninke (JJN), Stucki (St), Atkinson (A), Burkes (B), Sierra-3 (S3), Sierra-2 (S2), Sierra-Lite (SL).

$\frac{1}{16}\begin{bmatrix} - & \times & 7 \\ 3 & 5 & 1 \end{bmatrix}$ Floyd-Steinberg	$\frac{1}{48}\begin{bmatrix} - & - & \times & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{bmatrix}$ Jarvis-Judice-Ninke	$\frac{1}{42}\begin{bmatrix} - & - & \times & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$ Stucki
$\frac{1}{8}\begin{bmatrix} - & \times & 1 & 1 \\ 1 & 1 & 1 & - \\ - & 1 & - & - \end{bmatrix}$ Atkinson		$\frac{1}{32}\begin{bmatrix} - & - & \times & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \end{bmatrix}$ Burkes
$\frac{1}{32}\begin{bmatrix} - & - & \times & 5 & 3 \\ 2 & 4 & 5 & 4 & 2 \\ - & 2 & 3 & 2 & - \end{bmatrix}$ Sierra-3	$\frac{1}{16}\begin{bmatrix} - & - & \times & 4 & 3 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$ Sierra-2	$\frac{1}{4}\begin{bmatrix} - & \times & 2 \\ 1 & 1 & - \end{bmatrix}$ Sierra-Lite

Fig. 3. Error diffusion kernels.

4.2 Stochastic Rounding

Stochastic rounding is a probabilistic quantization method in which rounding is performed not deterministically but randomly, with a probability proportional to the fractional part of the number [29]. This approach makes the quantization process unbiased, meaning the expected value of the quantized number equals the original value:

$$E[\hat{a}] = a \quad (4)$$

The stochastic rounding algorithm for an element a is as follows:

1) compute the integer part: $a_{floor} = \lfloor a \rfloor$;
2) compute the fractional part: $d = a - a_{floor}$;
3) with probability $P = d$, round up: $\hat{a} = a_{floor} + 1$;
4) with probability $1 - P$, round down: $\hat{a} = a_{floor}$.

5. Software Implementation and Evaluation on RISC-V

When implementing a system for solving combinatorial optimization problems based on the Ising model using the RISC-V architecture, three different approaches are possible, each characterized by a specific trade-off between performance, flexibility, and development complexity.

Approach 1: purely software implementation on a standard RISC-V processor. The original combinatorial optimization problem is reduced to the Ising model by constructing the interaction matrix J and the external field vector h. Then, a software implementation of the simulated annealing algorithm is executed on a standard RISC-V core without any hardware extensions. All computation stages – spin selection, energy change calculation, decision making, and configuration updates – are implemented in software.

Approach 2: implementation on a RISC-V processor with an extended instruction set. Thanks to the open and modular RISC-V architecture, which allows for custom instruction set extensions, specialized instructions are introduced into the processor core. These instructions accelerate key operations of the simulated annealing algorithm for the Ising model. This reduces the number of cycles required for critical operations while maintaining processor generality [30].

Approach 3: hardware implementation with a RISC-V host processor. A system-on-chip is created in which a standard RISC-V core acts as a host controller, responsible for high-level management, data loading, coordination, and communication with the external environment. The simulated annealing algorithm itself is implemented in specialized hardware blocks – annealing processors – which enables maximum performance and energy efficiency.

In this work, the first approach was chosen. It is important to emphasize that the proposed diffusion and stochastic rounding methods for the interaction matrix remain relevant for all three architectural approaches considered. For Approach 1, these methods reduce memory requirements and increase computational speed by using more compact data types. For Approach 2, advanced rounding enables efficient use of the limited bit width of specialized registers. For Approach 3, these methods are a key factor determining solution quality, since hardware annealing processors fundamentally operate on data with limited bit width.

The developed software implementation covers the complete cycle of solving combinatorial optimization problems: from constructing the interaction matrix to executing the simulated annealing algorithm with quantized data. The overall architecture of the software implementation is shown in Fig. 4.

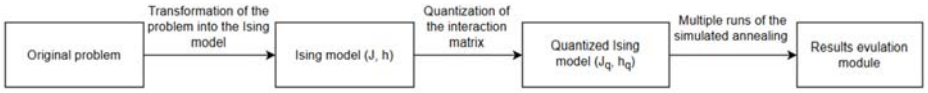

Fig. 4. Structural diagram of the implemented software solution for combinatorial optimization problems.

The main components are:

1. Interaction matrix generator – a module that reduces a specific combinatorial problem to the Ising model. For each of the three problems (TSP, Knapsack, Max-Cut), a separate function constructs the matrix J и and vector h based on input parameters.
2. Quantization module – implements three classes of rounding methods: simple, stochastic, and diffusion (with eight different error propagation schemes). Quantization is performed to a specified bit width (2–16 bits) using fixed-width integer types (int8_t, int16_t).

3. Simulated annealing algorithm – an implementation adapted to work with quantized matrices.
4. Quality evaluation module – computes solution quality metrics.

5.1 Experiments hardware equipment

All experiments were conducted on the Lichee Pi 4A single-board computer, a mainstream RISC-V platform. The board is built around the T-Head TH1520 system-on-chip, which integrates a quad-core RISC-V 64GCV C910 processor. Each core operates at a nominal frequency of 1.85 GHz and includes 64 KB of L1 instruction cache and 64 KB of L1 data cache, while all four cores share 1 MB of L2 cache. The board is equipped with 8 GB of LPDDR4X RAM and 32 GB of eMMC storage.

5.2 Influence of rounding methods on solution quality

This subsection presents the results of an experimental study on the influence of different interaction matrix quantization methods on solution quality for three classical NP-hard problems: the traveling salesman problem (TSP), the knapsack problem, and the max-cut problem. Solution quality is measured by these metrics: the average value of the objective function for solutions satisfying the constraints (for all three problems), and the ratio of the number of feasible solutions to the total number of solutions (for the knapsack and max-cut problems). The graphs show averaged results over 100 runs of the annealing algorithm for unrounded matrices (None), simple rounding (SR), diffusion rounding (FS, JJN, St, A, B, S3, S2, SL), and stochastic rounding (Stoh).

For the knapsack problem, the results are shown in Fig. 5 and Fig. 6. A 512-spin problem is considered with the following conditions: 500 items, item weights and values in the range [1, 30], weight constraint set to 3000 (requiring 12 auxiliary spins).

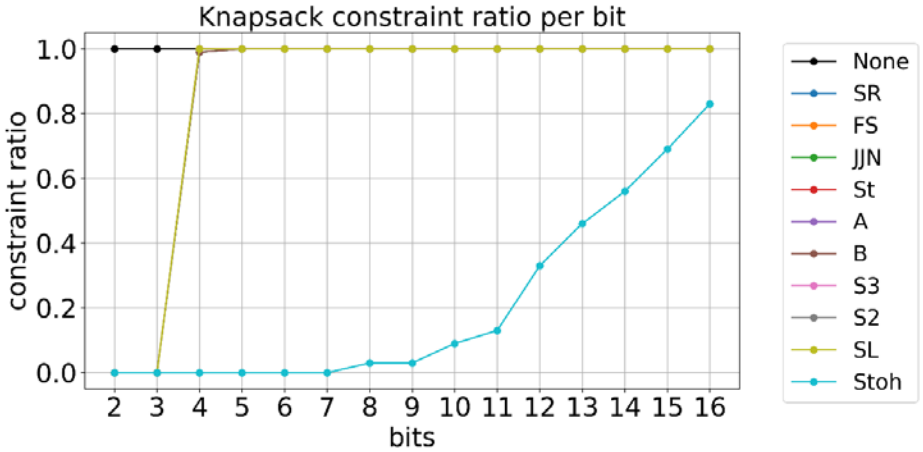

Fig. 5. Knapsack constraint ratio per bit.

Fig. 5 shows that at 2-3 bits, all rounded matrices produce invalid solutions. Starting from 4 bits, all rounding methods except stochastic begin to satisfy the constraints. Stochastic rounding only starts to converge at 8 bits, and as the bit width increases, its convergence improves monotonically. The average value of the objective function for all rounded matrices except stochastic behaves identically regardless of bit width. At 4 bits, it is anomalously close to that of the unrounded matrix, then decreases by 9 bits, and begins to rise again from 11 bits. Meanwhile, from 8 to 16 bits, the stochastic matrix shows an average objective function value comparable to that of the unrounded matrix. Thus, the diffusion rounding methods did not exhibit behavior different from simple rounding. Stochastic

rounding showed completely different behavior – zero convergence at bit widths below 8 and poor convergence at higher bit widths, but with the average objective function value matching that of the exact matrix.

For the max-cut problem, the results are shown in Fig. 7. A 64-spin problem is considered with the following conditions: 64 vertices, edge weights in the range [1, 100] (complete graph).

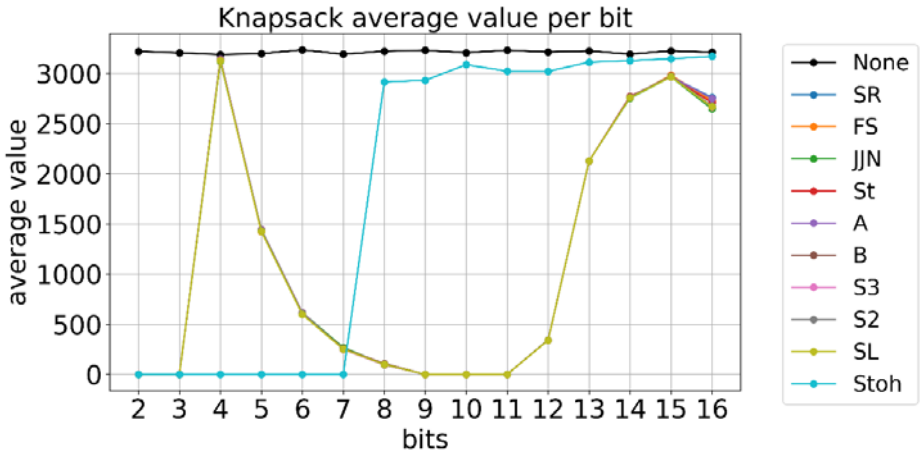

Fig. 6. Knapsack average value per bit.

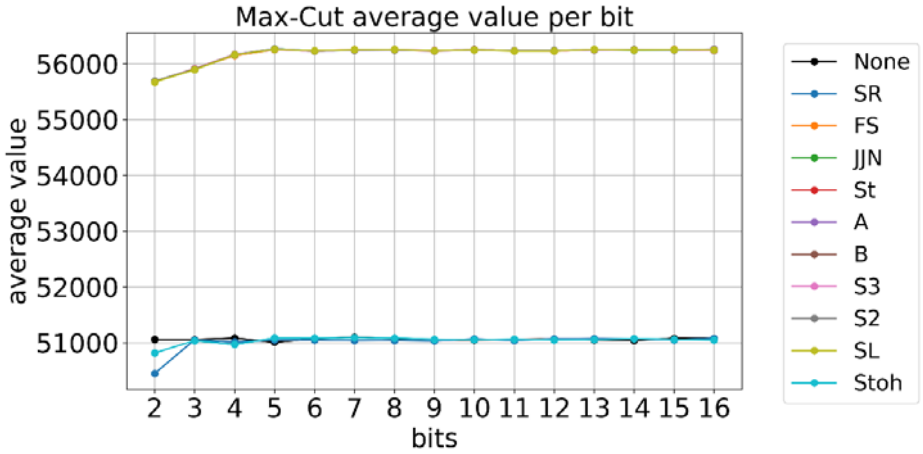

Fig. 7. Max-Cut average value per bit.

Since any solution is valid for this problem, the constraint ratio is not considered. The graphs show that the rounding methods can be divided into two groups: the first includes all diffusion rounding methods, and the second includes all the others. For this problem, the bit width has little effect on the objective function value. However, the first group of rounding methods consistently achieves a better average objective function value, which is 10% higher. Notably, all diffusion-rounded matrices produced results that are consistently better than those obtained with the exact matrix. This suggests that diffusion rounding introduces a distortion in the objective function that does not degrade the algorithm's ability to find good solutions but rather improves it.

For the traveling salesman problem, the results are shown in Fig. 8 and Fig. 9. A 144-spin problem is considered with the following conditions: 12 cities, intercity distances in the range [0.5, 2] (cities are uniformly placed on a circle of unit radius).

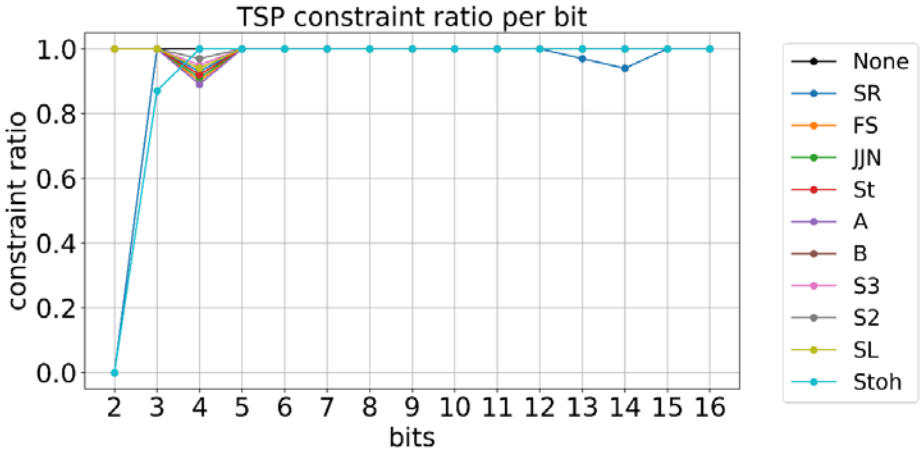

Fig. 8. TSP constraint ratio per bit.

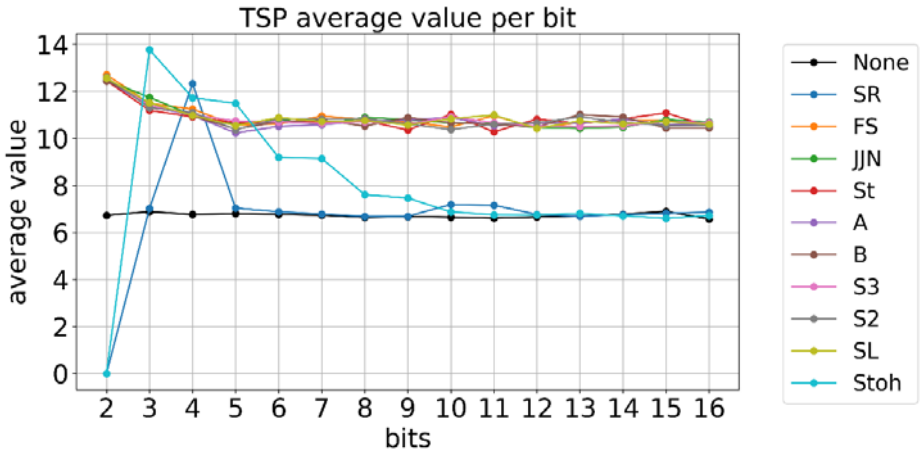

Fig. 9. TSP average value per bit.

At 2 bits, all diffusion rounding methods satisfy the constraints, while simple and stochastic rounding produce only invalid solutions. Above 3 bits, all rounding methods show excellent convergence. At 4 bits, all advanced rounding methods yield a better average objective function value than simple rounding; however, at other bit widths, their performance is noticeably worse than simple rounding. Moreover, from 10 bits onward, stochastic rounding produces results similar to simple rounding, while diffusion methods remain on a solution plateau that is consistently worse than simple rounding.

The experimental study shows that the effectiveness of interaction matrix quantization methods depends significantly on the structure of the problem being solved. For the Max-Cut problem, diffusion rounding consistently improves solution quality by 10% compared to the unrounded matrix, which can be explained by the "smoothing" of the energy landscape. For the knapsack

problem, diffusion methods offer no advantage over simple rounding, while stochastic rounding requires a bit width of at least 8 bits. For the traveling salesman problem, diffusion rounding is effective only at low bit widths (2 bits), whereas as the bit width increases, simple rounding becomes preferable. These results confirm that the choice of quantization method should be determined by the characteristics of the problem being solved and the available data representation bit width. A comparison of the applied rounding methods is summarized in Table 1. Simple rounding (SR) is used as the baseline for comparison.

Table 1. Comparison of rounding methods for different problem.

Problem	Simple Rounding	Diffusion Rounding	Stochastic Rounding
Knapsack	baseline	same as SR	worse than SR (CR = 0, bit < 8) better AV (+6-90%), but lower CR (<0.8) (bit ≥ 8)
Max-Cut	baseline	better than SR (+10% AV)	same as SR
TSP	baseline	better than SR (CR = 1, bit = 2) worse than SR (+60% AV, bit > 2)	worse than SR (up to +50% AV, bit < 10) same as SR (bit ≥ 10)

5.3 Performance analysis on Lichee Pi 4A

This subsection presents the results of an experimental study on the impact of data types on the execution speed of the simulated annealing algorithm on the Lichee Pi 4A board. For this purpose, the Max-Cut problem was used under the same conditions as in the previous section, but with 10,000 vertices, which corresponds to a 10,000-spin problem in the Ising model. The simulated annealing implementation shown in Listing 1 was used with different template parameters. For each set of template parameters, the results were averaged over 100 runs. The perf profiler was used to obtain numerical performance characteristics.

Table 2 compares execution time and cache usage characteristics for different combinations of data types. The template parameters SpinType, MatrixType, and AccType define the data types for storing spins, interaction matrix elements, and the local field accumulator, respectively.

Table 2. Performance comparison for different data type combinations.

Parameters (SpinType, MatrixType, AccType)	Execution time, sec	L1 cache misses	L1 cache loads	L2 cache misses	L2 cache loads
int8, double, double	3.35	2.64%	2,272,847,184	27.09%	492,746,409
int8, int8, int16	0.78	0.13%	659,345,424	31.06%	18,832,115
int8, int8, int32	0.72	0.25%	658,410,668	21.78%	27,229,163
int8, int16, int32	0.94	0.16%	661,943,864	32.10%	19,380,860
int8, int16, int64	0.93	0.17%	661,423,278	23.33%	27,145,687

The longest execution time is observed for the configuration with a double-precision matrix. This is because a 10000×10000 matrix in double format occupies 800 MB, which far exceeds the L2 cache size (1 MB) of the processor. When using quantized int8 matrices (matrix size 100 MB), execution time drops to 0.72–0.94 s, corresponding to a speedup of 3.6–4.7 times. The best result is achieved using int8 for the matrix and int32 for the accumulator. The configuration with int16 for the accumulator shows a slightly worse time (0.78 s), which may be due to additional type conversion overhead.

When using a double matrix, the L1 miss rate is 2.64%, which is 10–20 times higher than for quantized matrices (0.13–0.25%). This is because the quantized matrix fits better in the cache, reducing the number of accesses to slower memory levels. The absolute number of L1 loads decreases from 2.27 billion to approximately 660 million (a factor of 3.4). The L2 miss rate ranges from 21.78% to 32.10% across different configurations. The lowest value (21.78%) is achieved for the int8, int8, int32 configuration. For the double matrix configuration, the L2 miss rate (27.09%) falls within the same range as for the quantized matrices; however, the absolute number of L2 loads for double (492 million) is significantly higher than for int8 (18–27 million). This is because when working with a double matrix, each memory access is more likely to cause a new cache line to be loaded.

Using int32 as the accumulator type yields slightly better execution time compared to int16 and int64. This is because 32-bit integer operations are native to the RISC-V architecture and do not incur additional overhead from bit-width extension (as with int64) or overflow checks (as with int16)

6. Conclusion

This study addressed the problem of efficient software implementation of combinatorial optimization algorithms on the RISC-V architecture. The key obstacle is the quadratic growth of the Ising model interaction matrix with the number of spins. Using double-precision floating-point representation (64 bits) leads to cache inefficiency on RISC-V processors, a sharp drop in performance, and makes it impossible to solve large-scale problems on devices with limited memory, such as single-board computers.

The proposed approach consists of reducing the bit width of the interaction matrix coefficients using advanced rounding methods – diffusion rounding (with eight error propagation schemes) and stochastic rounding – as opposed to the previously studied simple rounding. The novelty lies in a systematic comparison of these methods on three classical NP-hard problems (knapsack, max-cut, traveling salesman) and in the analysis of their effectiveness depending on the problem structure and available bit width.

The experimental results on the Lichee Pi 4A board show that switching from a double-precision matrix (800 MB for a 10,000-spin problem) to an 8-bit integer matrix (100 MB) reduces execution time by a factor of 3.6–4.7 (from 3.35 s to 0.72–0.94 s). The L1 cache miss rate drops from 2.64% to 0.13–0.25%, and the absolute number of L2 cache loads decreases from 492 million to 18–27 million. The best performance (0.72 s) is achieved with int8 for the matrix and int32 for the accumulator.

Regarding solution quality, the effectiveness of quantization methods depends strongly on the problem structure. For the max-cut problem, diffusion rounding consistently improves solution quality by 10% compared to the unrounded matrix. For the knapsack problem, diffusion methods offer no advantage over simple rounding, while stochastic rounding requires at least 8 bits to satisfy constraints. For the traveling salesman problem, diffusion rounding is beneficial only at very low bit widths (2 bits); as bit width increases, simple rounding becomes preferable.

The potential impact of the obtained results is twofold. From a practical perspective, the proposed methodology allows solving larger combinatorial problems directly on RISC-V systems with limited memory and computing resources, including single-board computers (e.g., Lichee Pi 4A) and embedded devices. From a research perspective, the established dependence of rounding method

effectiveness on the problem structure opens a new direction – the development of adaptive quantization strategies that select the optimal rounding scheme based on the properties of a specific optimization task. Although the results are also relevant to annealing processors, the primary contribution is to efficient software execution on general-purpose RISC-V cores.

Large language model (LLM) tools were used for language editing and stylistic refinement of the manuscript. All scientific content, experimental design and conclusions were developed and verified by the authors. The source code of the software prototype is available in the public repository at [31].

References

[1]. Cui E., Li T., Wei Q. RISC-V instruction set architecture extensions: A survey. IEEE Access, 2023, vol. 11, pp. 24696-24711. DOI: 10.1109/ACCESS.2023.3246491.

[2]. Cherepanov N.I., Stepina N.O., Nikiforov I.V. Improving image analysis and processing performance on the RISC-V platform with Lichee Pi 4A. Trudy ISP RAN/Proc. ISP RAS, 2025, vol. 37, issue 5, pp. 157-172. DOI: 10.15514/ISPRAS-2025-37(5)-12.

[3]. Volokitin V.D., Vasiliev E.P., Kozinov E.A., Kustikova V.D., Liniov A.V., Rodimkov Y.A., Sysoyev A.V., Meyerov I.B. Vectorization of gradient boosting of decision trees prediction in the CatBoost library for RISC-V processors. Lobachevskii Journal of Mathematics, 2024, vol. 45, no. 1, pp. 130-142. DOI: 10.1134/S1995080224010530.

[4]. Pirova A., Vodeneeva A., Kovalev K., Ustinov A., Kozinov E., Liniov A., Volokitin V., Meyerov I. Performance optimization of BLAS algorithms with band matrices for RISC-V processors. Future Generation Computer Systems, vol. 174, 2026, 107936, 12 p. DOI: 10.1016/j.future.2025.107936.

[5]. Arianyan E., Gholipour N., Maleki D., Ghorbani N., Sepahvand A., Goudarzi P. A Systematic Review and Classification of HPC-Related Emerging Computing Technologies. Electronics, 2025, vol. 14, no. 12, 2476. DOI: 10.3390/electronics14122476.

[6]. Bourgeois N., Escoffier B., Paschos V. Th. An Introduction to Exponential Time Exact Algorithms for Solving NP-Hard Problems. In Mahjoub R. A. (ed.) Progress in Combinatorial Optimization. London: ISTE; Hoboken, NJ: John Wiley & Sons, 2012, pp. 443-468. Zbl 1242.90188.

[7]. Yang X.-S. Nature-Inspired Metaheuristic Algorithms. 2nd ed. Luniver Press, 2010. 160 p.

[8]. Gao Y., Chen G., Qi L., et al. Photonic Ising machines for combinatorial optimization problems. Applied Physics Reviews, 2024, vol. 11. DOI: 10.1063/5.0216656.

[9]. Mohseni N., McMahon P.L., Byrnes T. Ising machines as hardware solvers of combinatorial optimization problems. Nature Reviews Physics, 2022, vol. 4, pp. 363-379. DOI: 10.1038/s42254-022-00440-8.

[10]. Yamamoto K., Kawamura K., Ando K., et al. STATICA: A 512-spin 0.25M-weight annealing processor with an all-spin-updates-at-once architecture for combinatorial optimization with complete spin-spin interactions. IEEE Journal of Solid-State Circuits, 2021, vol. 56, no. 1, pp. 165-178. DOI: 10.1109/JSSC.2020.3027702.

[11]. Chen J., Yu Z., Zhu X. A 28 nm Ising machine with adaptive majority voter and reduction algorithms for high-performance combinatorial optimization. Microelectronics Journal, 2025, vol. 157. DOI: 10.1016/j.mejo.2025.106589.

[12]. Sato Y., Konoshima M., Tamura H., Ohkubo J. Characterization of Locality in Spin States and Forced Moves for Optimizations. Journal of the Physical Society of Japan, 2024, vol. 93, 044802. DOI: 10.7566/JPSJ.93.044802.

[13]. Oku D., Tawada M., Tanaka S., Togawa N. How to reduce the bitwidth of an Ising model by adding auxiliary spins. IEEE Transactions on Computers, 2022, vol. 71, no. 1, pp. 223-234. DOI: 10.1109/TC.2020.3045112.

[14]. Cui D., Kawahara T. Proposal and validation of annealing-processor-calculation method using error-diffusion rounded interaction matrix. 2024 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Taipei, Taiwan, 2024, pp. 50-54. DOI: 10.1109/APCCAS62557.2024.00017.

[15]. Cui D., Megumi T., Endo A., Kawahara T. Dual scalable annealing processing system that scales number of spins and interaction bit width simultaneously. IEEE Access, 2025, vol. 13, pp. 52592-52606. DOI: 10.1109/ACCESS.2025.3553542.

[16]. Щербина О.А. Метаэвристические алгоритмы для задач комбинаторной оптимизации (обзор). Таврический вестник информатики и математики, 2014, № 1 (24), стр. 2-17. / Shcherbina O. A. Metaheuristic algorithms for combinatorial optimization problems (review). Tavricheskiy Vestnik Informatiki i Matematiki, 2014, no. 1 (24), pp. 2-17 (in Russian).

[17]. Lucas A. Ising formulations of many NP problems. Frontiers in Physics, 2014, vol. 2, article 5, 27 p. DOI: 10.3389/fphy.2014.00005.
[18]. Pardalos P.M., Du D.-Z., Graham R.L. (eds.) Handbook of Combinatorial Optimization. 2nd ed. New York: Springer, 2013. 3409 p.
[19]. Korte B., Vygen J. Combinatorial Optimization: Theory and Algorithms. 6th ed. Berlin: Springer, 2018. (Algorithms and Combinatorics, vol. 21). DOI: 10.1007/978-3-662-56039-6.
[20]. Pathria R.K., Beale P.D. Statistical Mechanics. 4th ed. Academic Press, 2022. 944 p.
[21]. Mohseni N., McMahon P.L., Byrnes T. Ising machines as hardware solvers of combinatorial optimization problems. Nature Reviews Physics, 2022, vol. 4, pp. 363-379. DOI: 10.1038/s42254-022-00440-8.
[22]. Kirkpatrick S., Gelatt C.D., Vecchi M.P. Optimization by simulated annealing. Science, 1983, vol. 220, no. 4598, pp. 671-680. DOI: 10.1126/science.220.4598.671.
[23]. Hajek B. Cooling schedules for optimal annealing. Mathematics of Operations Research, 1988, vol. 13, no. 2, pp. 311-329. DOI: 10.1287/moor.13.2.311.
[24]. Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H., Teller E. Equation of state calculations by fast computing machines. The Journal of Chemical Physics, 1953, vol. 21, no. 6, pp. 1087-1092. DOI: 10.1063/1.1699114.
[25]. Ferreira A.L.C., Toral R. Projected single-spin-flip dynamics in the Ising model. Physical Review E, 2007, vol. 76, no. 1, pp. 1-10. DOI: 10.1103/PhysRevE.76.011117.
[26]. Garofalo A., et al. A 3 TOPS/W RISC-V parallel cluster for inference of fine-grain mixed-precision quantized neural networks. 2023 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2023. DOI: 10.1109/ISVLSI59297.2023.00045.
[27]. Martínez H., Castelló A., Igual F. D., Quintana-Ortí E.S. The Cambrian explosion of mixed-precision matrix multiplication for quantized deep learning inference. arXiv preprint, 2025. DOI: 10.48550/arXiv.2506.11728.
[28]. Floyd R. W., Steinberg L. An adaptive algorithm for spatial grey scale. Journal of the Society for Information Display, 1976, vol. 17, pp. 75-77.
[29]. Gupta S., Agrawal A., Gopalakrishnan K., Narayanan P. Deep learning with limited numerical precision. Proceedings of the 32nd International Conference on Machine Learning (ICML), 2015, pp. 1737-1746.
[30]. Armeniakos G., Maras A., Xydis S., Soudris D. MaRVIn: A cross-layer mixed-precision RISC-V framework for DNN inference, from ISA extension to hardware acceleration. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2025. DOI: 10.48550/arXiv.2509.15187.
[31]. Quantizing Ising model matrices for combinatorial optimization on RISC-V with Lichee Pi 4A. GitLab repository. Available at: https://gitlab.com/risc-v-spbstu/the_interaction_matrix_with_edm/, accessed 01.06.2026.

Информация об авторах / Information about authors

Александр Михайлович БРАТЕНКОВ – бакалавр высшей школы программной инженерии Санкт-Петербургского политехнического университета Петра Великого по специальности 09.04.03 – «Технология разработки и сопровождения качественного программного продукта». Область научных интересов – архитектура RISC-V, высокопроизводительные вычисления, низкоуровневое программирование.

Alexander Mikhailovich BRATENKOV is a bachelor's student at the Higher School of Software Engineering at the St. Petersburg State Polytechnic University, specializing in «Technology for developing and maintaining a high-quality software product». His research interests include RISC-V architecture, high-performance computing, low-level programming.

Надежда Олеговна СТЕПИНА – ассистент высшей школы программной инженерии Санкт-Петербургского политехнического университета Петра Великого. В 2023 году окончила Санкт-Петербургский государственный политехнический университет по специальности «Программная инженерия». В 2024 году стала аспирантом по специальности 2.3.5 – «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей». Область научных интересов – разработка программного обеспечения, машинное обучение, высокопроизводительные вычисления, IoT и embedded-системы.

Nadezhda Olegovna STEPINA is an Assistant Professor at the Higher School of Software Engineering at Peter the Great St. Petersburg Polytechnic University. In 2023, she graduated from the St. Petersburg State Polytechnic University with a degree in Software Engineering. In 2024, she became a postgraduate student in the field of Mathematical and Software Support for Computing Machines, Complexes, and Computer Networks. Her research interests include software development, machine learning, high-performance computing, IoT, and embedded systems.

Игорь Валерьевич НИКИФОРОВ – кандидат технических наук по специальности «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей», доцент высшей школы программной инженерии Санкт-Петербургского политехнического университета Петра Великого. Автор 100 научных публикаций. Область научных интересов – разработка программного обеспечения, имитационное моделирование, аналитика больших данных, распределенные вычисления.

Igor Valerievich NIKIFOROV – Cand. Sci. (Tech.) in Mathematical and software support for computers, complexes and computer networks (2014), graduated from St. Petersburg State Polytechnic University with a degree in «Computer Science and Automated Systems Software» (2011). He is an Associate Professor at the Higher School of Software Engineering at Peter the Great St. Petersburg Polytechnic University. He is the author of more than 100 scientific publications. Research interests – software engineering, simulation modeling, big data analytics, distributed computing.

Ольга Андреевна ЮСУПОВА – старший преподаватель высшей школы программной инженерии Санкт-Петербургского политехнического университета Петра Великого. В 2009 году окончила Санкт-Петербургский государственный политехнический университет по направлению «Информатика и вычислительная техника». Имеет второе высшее образование в области экономики по специальности – «Прикладная информатика в экономике». Является автором учебных пособий, научных статей. Сфера профессиональных интересов охватывает математическое моделирование, системный анализ, тестирование и верификацию программного обеспечения.

Olga Andreevna YUSUPOVA is a Senior Lecturer at the Higher School of Software Engineering, Peter the Great St. Petersburg Polytechnic University. In 2009, she graduated from St. Petersburg State Polytechnic University with two degrees – in Informatics and Computer Engineering and in Applied Informatics in Economics. She is the author of numerous textbooks and scientific papers. Her professional interests include mathematical modeling, systems analysis, software testing and verification.