GPU-Accelerated Simulated Oscillator Ising/Potts Machine Solving Combinatorial Optimization Problems

Yilmaz Ege Gonul
Drexel University
Philadelphia, PA, USA
yeg26@drexel.edu

Ceyhun Efe Kayan
Drexel University
Philadelphia, USA
cek99@drexel.edu

Ilknur Mustafazade
Drexel University
Philadelphia, USA
im445@drexel.edu

Nagarajan Kandasamy
Drexel University
Philadelphia, PA, USA
nk78@drexel.edu

Baris Taskin
Drexel University
Philadelphia, PA, USA
bt62@drexel.edu

Abstract

Oscillator-based Ising machines (OIMs) and oscillator-based Potts machines (OPMs) have emerged as promising hardware accelerators for solving NP-hard combinatorial optimization problems by leveraging the phase dynamics of coupled oscillators. In this work, a GPU-accelerated simulated OIM/OPM digital computation framework capable of solving combinatorial optimization problems is presented. The proposed implementation harnesses the parallel processing capabilities of GPUs to simulate large-scale OIM/OPMs, leveraging the advantages of digital computing to offer high precision, programmability, and scalability. The performance of the proposed GPU framework is evaluated on the max-cut problems from the GSET benchmark dataset and graph coloring problems from the SATLIB benchmarks dataset, demonstrating competitive speed and accuracy in tackling large-scale problems. The results from simulations, reaching up to 11295× speed-up over CPUs with up to 99% accuracy, establish this framework as a scalable, massively parallelized, and high-fidelity digital realization of OIM/OPMs.

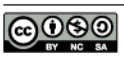
CCS Concepts

• **Hardware → Emerging architectures**.

Keywords

Ising Machine, GPU Acceleration, CUDA, Coupled Oscillators, Combinatorial Optimization, Kuramoto Model

ACM Reference Format:
Yilmaz Ege Gonul, Ceyhun Efe Kayan, Ilknur Mustafazade, Nagarajan Kandasamy, and Baris Taskin. 2025. GPU-Accelerated Simulated Oscillator Ising/Potts Machine Solving Combinatorial Optimization Problems. In *Great Lakes Symposium on VLSI 2025 (GLSVLSI '25), June 30–July 02, 2025, New Orleans, LA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3716368.3735247



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.
GLSVLSI '25, New Orleans, LA, USA
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1496-2/25/06
https://doi.org/10.1145/3716368.3735247

1 Introduction

Combinatorial optimization problems (COPs) [1] represent a critical class of computational challenges that have significant implications across numerous fields, including logistics, telecommunications, bioinformatics, and artificial intelligence. These problems, which involve finding an optimal object from a finite set of objects, are often NP-hard [1], for which the computational resources required to solve them optimally scale exponentially with problem size. Consequently, there is a persistent need for novel approaches that can provide high-quality solutions within reasonable computational constraints.

In recent years, growing interest in physics-inspired computing paradigms that leverage the natural dynamics of physical systems to solve computational problems has led to the emergence of Ising machines [2]. Ising machines are subject of research as promising specialized solvers for quadratic unconstrained binary optimization (QUBO) problems, which can encode many important combinatorial optimization problems (COPs). Potts machines [3] are the generalization of the Ising machines, allowing multivalued spins and natively mapping multivalued variable problems. Ising and Potts machines map optimization problems onto systems of coupled spins and utilize the system's tendency to settle into low-energy states to identify optimal or near-optimal solutions.

Various Ising and Potts machine implementations in the literature include quantum annealers [4], optical parametric oscillator networks [5], oscillator-based [6, 7], and CMOS implementations [8–12]. CMOS-based implementations offer significant advantages, including compatibility with existing semiconductor infrastructure, room-temperature operation, and relatively lower cost of fabrication, along with precision, programmability, and scalability advantages when implemented digitally. Oscillator-based approaches provide natural dynamical systems for optimization with inherent parallelism through phase dynamics that closely match the mathematical properties of Ising and Potts models.

In this paper, a GPU-based implementation of a simulated coupled oscillator Potts machine based on the Kuramoto model of coupled oscillators is presented, combining the strengths of CMOS and oscillator-based Ising/Potts machines. The proposed approach leverages CUDA programming to efficiently parallelize the numerical simulation of oscillator dynamics. A modified Kuramoto model [6] is employed that incorporates noise-driven stochastic phase transitions and external stimulation for phase stabilization, enabling

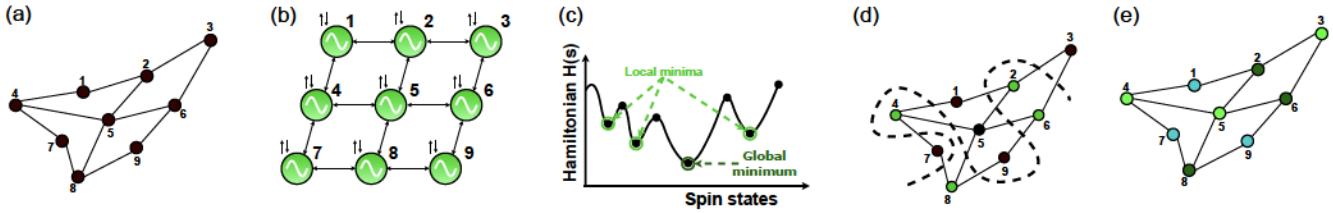


Figure 1: a) An example 9-node graph b) Problem mapped to a coupled oscillator array c) Energy minimization to ground states d) Max-cut of the graph e) 3-coloring of the graph

the system to escape local minima and explore the solution space more effectively. The developed GPU implementation leverages the advantages of digitally computing the oscillator dynamics to propose an Ising/Potts machine framework that achieves floating point precision in representing oscillator phases and coupling weights, scalability in computing on larger graph sizes, and the ability to map graphs of higher densities up to all-to-all connected graphs.

2 Technical Background

The Ising and Potts models are summarized in Section 2.1. Mathematical equations that describe the OIM/OPM behavior, utilized in the numerical computations, are shown in Section 2.2. Mapping of COP problems is presented in Section 2.3. The mechanism of SHIL is presented in Section 2.4 The modified Kuramoto model used in the numerical simulations is detailed in Section 2.5.

2.1 Ising/Potts Models and Machines

2.1.1 The Ising Model. The Ising model [2] consists of discrete variables (spins) that take binary values (±1), with interactions between pairs of spins described by coupling coefficients. Mathematically, the energy function (Hamiltonian), excluding the field term, is given by:

$$H_{\text{Ising}} = -\sum_{i<j} J_{ij} s_i s_j \tag{1}$$

where $s_i \in \{-1, +1\}$ represents the state of spin i, J_{ij} is the coupling coefficient between spins i and j.

2.1.2 The Potts Model. The Potts model [3] generalizes the Ising model by allowing each spin to take one of q possible states ($q \geq 2$), with the Ising model being the special case where $q = 2$. The Potts Hamiltonian, excluding the field term, can be written as:

$$H_{\text{Potts}} = -\sum_{i<j} J_{ij} \delta(s_i, s_j) \tag{2}$$

where $s_i \in \{1, 2, ..., q\}$ and δ is the Kronecker delta function.

Many important combinatorial optimization problems, including Max-Cut, Graph Coloring, and Boolean Satisfiability, can be reformulated as Ising or Potts problems [13]. The goal becomes finding spin configurations that minimize these Hamiltonians, corresponding to optimal solutions of the original problems.

2.2 OIM and OPM based on the Kuramoto Model of Coupled Oscillators

Prior study [6] mathematically demonstrates that the Kuramoto model of coupled oscillators can be used to represent an Ising/Potts machine through phase interactions of the oscillators. In the standard Kuramoto model, the phase evolution of each oscillator in a system of n coupled oscillators is governed by the following differential equation:

$$\frac{d\phi_i}{dt} = \omega_i + \sum_{j=1}^{n} K_{ij} \sin(\phi_i - \phi_j) \tag{3}$$

where ϕ_i is the phase, and ω_i is the natural frequency oscillator i, and K_{ij} is the coupling strength between oscillators i and j. The sine function creates a phase-dependent interaction where oscillators synchronize (when $K_{ij} > 0$) or desynchronize (when $K_{ij} < 0$) depending on the sign of the coupling coefficient.

2.3 Mapping COPs to Ising/Potts Models

To solve COPs using coupled oscillators, a mapping between the Ising model and the Kuramoto model is proposed in [6]. The key insight is that the phases of oscillators can encode the states of the Ising spins, and the coupling coefficients in the Kuramoto model can represent the interaction strengths in the Ising Hamiltonian.

2.3.1 Max-Cut Problem. The Max-Cut problem, as shown in Figure 1(d), divides graph vertices into two sets maximizing edge weights on the cut. This maps to the Ising model with $s_i \in \{-1, +1\}$ and $J_{ij} = -1$ for each edge. In oscillator implementations, stable phases $\phi_i \in \{0, \pi\}$ correspond to spin states, with anti-phase synchronization representing the optimal cut.

2.3.2 Graph Coloring Problem. Graph Coloring problem, illustrated in Figure 1(e), assigns colors to vertices where no adjacent vertices share colors. The Potts model allows direct mapping [13] of spin states to colors with $J_{ij} < 0$ for adjacent vertices, making the mapping more efficient than the Ising model, which would require complex penalties and ancillary variables. In oscillator networks, this appears as stable configurations where adjacent oscillators settle into different phases from $\{\frac{2\pi k}{N} | k = 0, 1, ..., N-1\}$, each representing a distinct color.

2.4 Sub-Harmonic Injection Locking (SHIL)

Sub-Harmonic Injection Locking (SHIL) is a technique that modifies the standard Kuramoto model by adding a nonlinear drive term to

encourage oscillators to settle into specific discrete phase states [6]. In the SHIL-enhanced Kuramoto equation:

$$\frac{d\theta_i}{dt} = K \sum_{j=1}^{n} J_{ij} \sin(2\pi(\theta_i - \theta_j)) + K_s \sin(2\pi N \theta_i) \quad (4)$$

The additional term $K_s \sin(2\pi N\theta_i)$ creates stable fixed points at evenly spaced phase values. Here, $\theta_i \in [0, 1)$ is the normalized phase of oscillator i, K is the coupling strength between oscillators, K_s is the strength of the injection locking term (which can be varied over time), and N determines the number of stable phase states.

This modification enables combinatorial optimization by discretizing the phase space, allowing oscillators to represent discrete variables in both Ising and Potts models. For binary-variable Ising problems ($N = 2$), oscillators settle at phases that map to Ising spin states +1 and −1. For multi-valued Potts problems ($N \geq 3$), oscillators converge to one of N equally spaced phases. As the system evolves according to the coupling matrix J_{ij}, the network converges toward minimum-energy configurations as demonstrated in Figure 1(c) that correspond to optimal or near-optimal solutions to the encoded problem.

2.5 Modified Kuramoto Model for Optimization

The proposed GPU-based framework employs a modified version of the Kuramoto model that is specifically tailored [6] for solving combinatorial optimization problems with increased accuracy. The dynamics of each oscillator is governed by the following differential equation:

$$\frac{d\phi_i}{dt} = K \sum_{j=1}^{n} J_{ij} \sin(2\pi(\phi_i - \phi_j)) + K_s \sin(2\pi N \phi_i) + \eta_i(t) \quad (5)$$

where:

- $\phi_i \in [0, 1)$ is the normalized phase of oscillator i
- K is the global coupling strength, scaling the strength of every interaction J_{ij}
- J_{ij} is the coupling coefficient between oscillators i and j
- K_s is the strength of the influence of the SHIL term
- N is the number of stable phase states (e.g., $N = 2$ for Ising problems)
- $\eta_i(t)$ is Gaussian white noise with standard deviation $\sigma = \sqrt{K_n \cdot dt}$, where K_n is the noise strength

This formulation includes several important modifications to the standard Kuramoto model. Phases are normalized to the range $[0, 1)$ rather than $[0, 2\pi)$ for computational convenience. The SHIL term $K_s \sin(2\pi N\phi_i)$ introduces N stable fixed points at phases $\{\frac{2\pi k}{N} | k = 0, 1, ..., N-1\}$, corresponding to the N possible states of each oscillator phase. Additionally, the addition of Gaussian noise allows the system to escape local minima and explore the solution space more effectively, similar to thermal fluctuations in physical annealing processes.

The Hamiltonian corresponding to this system can be expressed as:

$$H = \sum_{i<j} J_{ij} \cos(2\pi(\phi_i - \phi_j)) \quad (6)$$

when all oscillators are at stable fixed points (i.e., after thresholding). This Hamiltonian reaches the minimum value when the oscillator phases are configured to minimize the overall energy of the system, which corresponds to the optimal solution of the mapped optimization problem.

Algorithm 1 Simulated Coupled Oscillator Ising/Potts Machine

```
// Initialize variables
φ ← Random values in [0, 1); J ← Coupling matrix; t ← 0
// Define simulation parameters
K, K_s_max, K_n // Coupling, self-interaction and noise strengths
h, t_stop // Time step for integration and total simulation time
// Generate triangular waveform for K_s
K_s ← generateTriangularWaveform(t_stop, K_s_max)
// Begin simulation loop
while t < t_stop do
    // For each oscillator, compute interaction terms
    for i = 0 to n − 1 do
        ∂_i ← 0 // Initialize partial derivative
        for j = 0 to n − 1 do
            ∂_i ← Σ_{j≠i} J_ij sin(2π(φ_i − φ_j))
        end for
    end for
    // Update all oscillator phases using Forward Euler method
    for i = 0 to n − 1 do
        derivative ← K · ∂_i + K_s[t] · sin(2πNφ_i)
        noise ← K_n · N(0, 1) · √h // Gaussian noise term
        φ_i ← φ_i + h · derivative + noise
        // Threshold phases to stable solutions if needed
        φ_i ← NormalizeToRange(φ_i, 0, 1) // Keep phases in [0,1)
    end for
    t ← t + h // Update time
end while
```

3 GPU Implementation Details

The algorithm employed to simulate the coupled oscillator dynamics, CUDA implementation and optimization methods, and the annealing technique used to improve accuracy are detailed in this section.

3.1 Numerical Simulation Algorithm

The proposed GPU-accelerated framework implements OIM/OPM by efficiently solving the governing differential equations that describe the coupled oscillator dynamics. The implementation utilizes a direct numerical integration approach with the Forward Euler method, inspired by the work in [14] and [7], which offers a balance between computational simplicity and numerical stability. The pseudocode of the algorithm is shown in Algorithm 1. The proposed GPU framework can simultaneously update all oscillator phases in each time step, enabling the simulation of large-scale networks in parallel.

The proposed GPU-accelerated algorithm offers several computational advantages. The parallel computation of phase derivatives across all oscillators eliminates the $O(n^2)$ sequential bottleneck inherent in CPU implementations. Memory hierarchy optimizations significantly reduce global memory accesses during Hamiltonian

calculations. Additionally, the digital implementation enables precise control over the triangular modulation of K_s and exact phase discretization, which are challenging to achieve in analog. The modular structure of the algorithm allows for easy experimentation with different coupling topologies and noise parameters without hardware modifications, enabling rapid exploration of solution spaces for various graphs.

3.2 CUDA Implementation

Memory efficiency is achieved by transforming the coupling matrix from 2D to 1D flattened arrays for improved access patterns, storing oscillator phases in separate device memory arrays to prevent race conditions, and utilizing the CURAND [15] library with per-thread generator states for efficient parallel noise generation. Two primary kernels drive the computation: the Kuramoto integration kernel handles the computationally demanding numerical integration of oscillator dynamics by assigning individual oscillators to dedicated GPU threads for computing derivatives, adding noise, and updating phases based on the modified Kuramoto model; while the phase threshold kernel independently processes the phase of each oscillator to find the nearest stable solution. This dual-kernel approach ensures increased parallelism throughout the computation pipeline, enabling efficient simulation of large-scale OIM/OPMs.

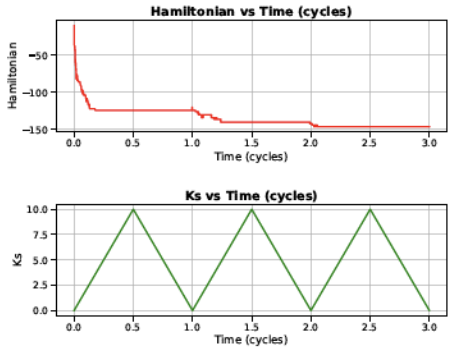


Figure 2: Ks parameter ramped up and down in a triangular waveform and the Ising hamiltonian pushed out of local minima to converge into lower energy states

3.3 CUDA Optimization Techniques

The following steps are taken to ensure proper utilization of the memory hierarchy within the GPU: The kernel is tested with float32 instead of double values while maintaining precision, which results in a ~2.46× improvement in speed compared to the naive version of the implementation. The largest improvement is observed to come from batch processing of oscillators, rather than a GPU thread per oscillator, resulting in a ~4.6× improvement. Other optimizations include using on-chip shared memory, using cache directives to prioritize L1 usage, unrolling loops to increase register usage, fusing multiplication and addition with CUDA intrinsic instructions, and host-side buffer pinning, which all contribute to another ~2.85× speedup.

The implementation employs a hierarchical decomposition approach where threads from the same thread blocks cooperatively process batches of oscillator interactions. The batch size can be user-selected and depends on both the GPU and problem size. The oscillator space is partitioned across multiple blocks. The block-level batching of the problem allows users to fine-tune their application by selecting the distribution of the batches between thread blocks.

3.4 Annealing Schedules

The varying Ks parameter in the modified Kuramoto model functions as a critical annealing mechanism that enhances solution accuracy [6]. When the Ks value is modulated through a triangular waveform pattern (ramping up and down periodically), the modulation enables the system to thoroughly navigate the solution space rather than becoming prematurely trapped in suboptimal configurations as shown in Figure 2. In this work, the annealing schedule is implemented as a triangular waveform for Ks, periodically ramping up to a maximum value and then decreasing to zero.

4 Experimental Results

The experimental evaluation of the proposed GPU framework is conducted with simulations on max-cut problems from the G-set benchmark suite [16], and flat 3-coloring problems from the SATLIB benchmark suite [17] and four larger custom-generated 3-colorable graphs. All simulations are executed on a Linux server with an NVIDIA A100 SXM4 GPU with 80GB of memory, leveraging its 6912 CUDA cores to accelerate the parallel computation of oscillator dynamics across large networks. For baseline comparisons and validation, the same algorithm is also implemented in C to run on CPU, executed on the same server with an AMD EPYC 7513 processor featuring 32 cores at 2.6 GHz. The goal of these simulations is to demonstrate the speed of the proposed framework as a digital Ising/Potts solver, while reaching competitive quality of solutions.

It is shown in [6] that a coupled oscillator system exhibits an intrinsic convergence time (i.e the number of oscillation cycles to minimize the Hamiltonian) that scales sub-linearly with the oscillator count (i.e. mapped problem size), independent of hardware implementation. In the benchmark results, the stop time of the simulation is adjusted based on the convergence time for the largest problem instance, meaning smaller node graphs could be solved even more efficiently with appropriately calibrated simulation durations.

4.1 Performance Evaluation

Table 1 presents the performance metrics of the GPU-accelerated implementation across 32 max-cut problems (i.e. a 2-spin Ising problem) from the G-set benchmarks, ranging from 800 to 20,000 nodes. The proposed framework consistently achieves over 94% accuracy when compared to the best-known solutions for these benchmark instances, where the worst and best accuracies are 94.08% and 99.27% not based on the problem size but based on the problem difficulty. This consistent performance across graph sizes demonstrates the robustness of the framework approach for solving these NP-hard problems, even as the computational complexity increases with larger graphs.

The run-time performance shows near-linear scaling properties with problem size as demonstrated in Figure 3(c). For graphs with

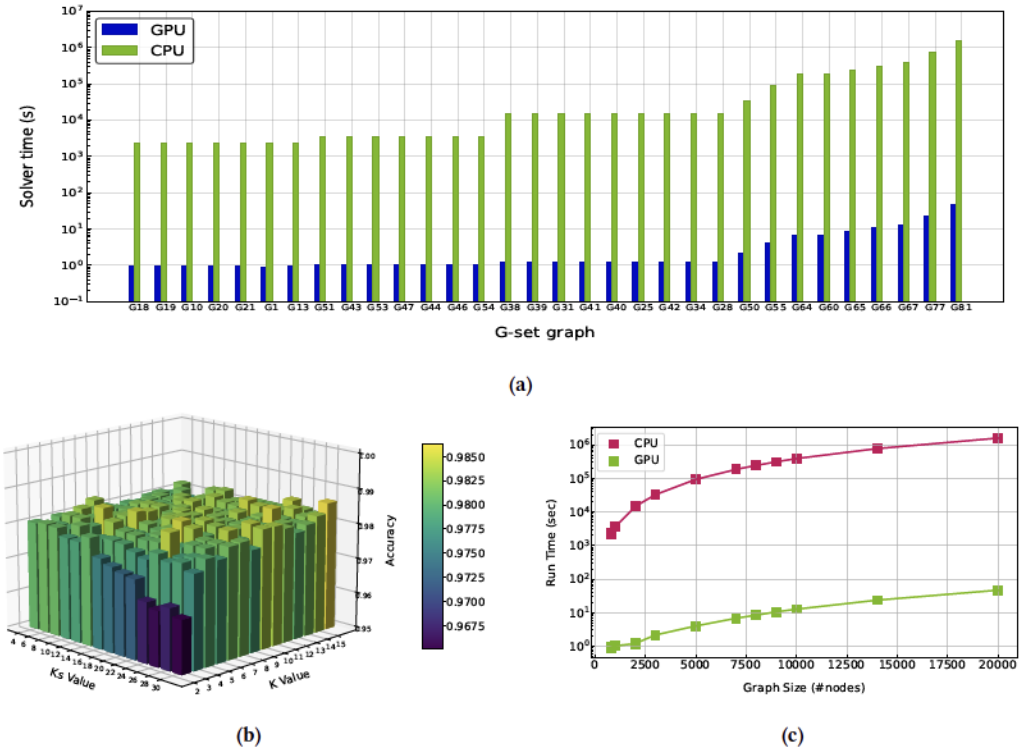


Figure 3: a) Speed comparison of CPU vs GPU against the G-set benchmark problems b) Parameter sweep analysis on K and Ks_{max}, and accuracy with each combination c) CPU vs GPU run-time scaling in log scale

800 nodes, the GPU framework completes in under 0.5 seconds, while the largest graphs with 20,000 nodes require approximately 23 seconds. This represents speed-ups ranging from 795× for smaller instances to 11,295× for the largest, as demonstrated in Figure 3(a), showing that the GPU parallelization strategy effectively distributes the computational load across thousands of cores. It is important to note that the accuracy of the GPU and CPU implementations are the same as the same algorithm runs on both.

Table 2 presents the performance of the GPU framework on the SATLIB 3-coloring benchmark instances and four custom large 3-colorable (i.e. a 3-spin Potts problem [12]) graphs. Despite the inherently higher complexity of graph-coloring compared to max-cut, with a search space of 3^n versus 2^n for n nodes where the GPU approach maintains similar accuracy levels with graph-coloring problems. For the standard benchmark instances (flat50-flat200), the GPU achieves speed-ups that increase with problem size, from 1.5× for the smallest instance to 50× for the 200-node graph. On the custom larger instances (1000-8000 nodes), the speed-ups grow from 1091× to 8550×, demonstrating the scaling properties of the GPU implementation for problems of practical size. Execution times remain practical even for the largest 8000-node instance at just 4.72 seconds. The accuracy remains competitive for the entire test set, with most instances achieving over 98% accuracy.

4.2 Parameter Effects on Accuracy and Speed

Parameter optimization is critical in the numerical simulations. Coupling strength K determines convergence characteristics, where higher values accelerate oscillator information exchange and promote clearer phase separation, improving the solution accuracy while risking numerical instability at excessive levels. SHIL amplitude Ks, especially when it is triangle-modulated, functions as an annealing schedule, with time-varying Ks outperforming constant values. The K/Ks ratio must balance the exploration and stability states of the oscillators as shown in Figure 3(b).

Noise amplitude Kn enables escape from local minima when moderate, but degrades solutions when extreme. Time parameters also impact performance, where smaller step sizes h yield more accurate results by reducing truncation errors but increase computational cost, while simulation time t_{stop} must be sufficient for phase synchronization without wasting resources. In the simulations, these parameters are optimized to balance run-time efficiency and solution quality, maximizing K to achieve convergence within a smaller t_{stop}.

4.3 Comparison with Related Work

Table 3 presents a comparison between this work and a previous GPU-based Ising solver leveraging a simulated annealing algorithm [18] across four G-set benchmark instances. The proposed framework consistently demonstrates superior solution quality

Table 1: Accuracy, runtime and speed-up (over CPU) of the GPU framework solving G-set max-cut benchmark problems

Graph	#Nodes	Best Known	GPU-Cut	Acc(%)	t_{GPU} (s)	SpeedUp
G81	20000	<NA>	13254	<NA>	23.37	11295x
G77	14000	<NA>	9390	<NA>	11.79	10908x
G67	10000	6950	6546	94.19	6.34	10234x
G66	9000	6364	6018	94.56	5.29	9828x
G65	8000	5562	5366	96.47	4.20	9640x
G60	7000	14188	13828	97.46	3.42	9062x
G64	7000	8751	8313	94.99	3.40	9051x
G55	5000	10299	10049	97.57	2.00	7756x
G50	3000	5880	5588	95.03	1.08	5039x
G28	2000	3298	3149	95.48	0.58	4127x
G34	2000	1384	1302	94.08	0.59	4125x
G42	2000	2481	2402	96.83	0.59	4088x
G25	2000	13340	13173	98.75	0.59	4054x
G40	2000	2400	2296	95.66	0.60	4053x
G41	2000	2405	2300	95.63	0.60	4019x
G31	2000	3310	3125	94.41	0.60	4013x
G39	2000	2408	2348	97.51	0.60	4011x
G38	2000	7688	7570	98.47	0.61	4003x
G54	1000	3852	3793	98.47	0.51	1163x
G46	1000	6649	6531	98.23	0.51	1158x
G44	1000	6650	6554	98.56	0.51	1158x
G47	1000	6657	6566	98.63	0.52	1156x
G53	1000	3850	3807	98.88	0.52	1144x
G43	1000	6660	6604	99.16	0.52	1144x
G51	1000	3848	3789	98.47	0.53	1127x
G13	800	582	566	97.25	0.47	831x
G1	800	11624	11539	99.27	0.45	828x
G21	800	931	880	94.52	0.47	822x
G20	800	941	900	95.64	0.47	813x
G10	800	2000	1926	96.30	0.47	806x
G19	800	906	867	95.70	0.47	805x
G18	800	992	942	94.96	0.47	795x

Table 2: Accuracy, runtime, and speed-up of the SATLIB 3-coloring benchmarks and custom large 3-colorable graphs

Graph	#Nodes	#Edges	$t_{GPU}(s)$	SpeedUp	Acc (%)
flat50_115_1	60	115	0.01	1.5x	99.1
flat100_239_1	100	239	0.03	14x	98.7
flat150_360_1	150	360	0.04	26x	98.2
flat200_479_1	200	479	0.05	50x	98.8
custom_1000	1000	3453	0.53	1091x	96.7
custom_2000	2000	7600	0.81	3000x	95.2
custom_5000	5000	21292	2.26	6857x	99.2
custom_8000	8000	35948	4.72	8550x	99.5

Table 3: Comparison with related work over benchmark instances

	G13		G34		G39		G42	
	Acc (%)	Time (s)	Acc (%)	Time (s)	Acc (%)	Time (s)	Acc (%)	Time (s)
This work	97.25	0.47	94.08	0.59	97.51	0.60	96.83	0.59
[18]	90.00	0.11	86.8	0.11	94.7	0.20	94.20	0.26

compared to [18], achieving accuracy improvements of 2.63-7.28 percentage points across all instances, notably due to the capability of the OIM framework to escape from suboptimal solutions. This accuracy advantage comes with a modest increase in computation time (0.47-0.60s versus 0.11-0.26s), that could be higher, considering the use of different GPUs. Nonetheless, the trade-off is favorable for practical applications, as the proposed framework maintains sub-second run times with solutions closer to the global optimum.

5 Conclusion

In this paper, a GPU-based framework for oscillator Ising/Potts machines is presented, solving max-cut and 3-coloring problems with substantial speed-ups over CPU implementations. The proposed GPU-based implementation is a framework for solving combinatorial optimization problems with direct problem mapping, floating point precision coupling, and scalable accommodation of graph size and density.

References

[1] I. L. Markov. 2014. Limits on fundamental limits to computation. *Nature* 512, 7513 (Aug. 2014), 147–154.

[2] E. Ising. 1925. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik* 31, 1 (Feb. 1925), 253–258. doi:10.1007/bf02980577

[3] F. Y. Wu. 1982. The Potts model. *Rev. Mod. Phys.* 54 (Jan 1982), 235–268. Issue 1. doi:10.1103/RevModPhys.54.235

[4] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. 2011. Quantum annealing with manufactured spins. *Nature* 473, 7346 (May 2011), 194–198. doi:10.1038/nature10012

[5] K. Inoue, K. Yoshida, and S. Kitahara. 2023. Coherent Potts machine based on an optical loop with a multilevel phase-sensitive amplifier. *Optics Communications* 528 (2023), 129022. doi:10.1016/j.optcom.2022.129022

[6] T. Wang and J. Roychowdhury. 2019. OIM: Oscillator-Based Ising Machines for Solving Combinatorial Optimisation Problems. In *Unconventional Computation and Natural Computation*, Ian McQuillan and Shinnosuke Seki (Eds.). Springer International Publishing, Cham, 232–256.

[7] J. Roychowdhury and S. Seal. 2022. *Oscillator-Based Potts Machine (OPM) for the Implementation of the Vector Potts Model.* Technical Report No. UCB/EECS-2022-198. EECS University of California, Berkeley, http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-198.html.

[8] T. Takemoto, M. Hayashi, C. Yoshimura, and M. Yamaoka. 2019. 2.6 A 2 ×30k-Spin Multichip Scalable Annealing Processor Based on a Processing-In-Memory Approach for Solving Large-Scale Combinatorial Optimization Problems. In *Proceedings of IEEE International Solid-State Circuits Conference - (ISSCC).* 52–54. doi:10.1109/ISSCC.2019.8662517

[9] I. Ahmed, P. Chiu, W. Moy, and C. H. Kim. 2021. A Probabilistic Compute Fabric Based on Coupled Ring Oscillators for Solving Combinatorial Optimization Problems. *IEEE Journal of Solid-State Circuits* 56, 9 (2021), 2870–2880. doi:10.1109/JSSC.2021.3062821

[10] N. Sica, R. Kuttappa, V. Honkote, and B. Taskin. 2024. High-Speed Phase-Based Computing. In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS).* 1–5. doi:10.1109/ISCAS58744.2024.10558674

[11] Y. E. Gonul and B. Taskin. 2025. A Multi-Stage Potts Machine based on Coupled CMOS Ring Oscillators. In *Proceedings of the Design, Automation & Test in Europe (DATE).*

[12] Y. E. Gonul and B. Taskin. 2025. Multi-phase Coupled CMOS Ring Oscillator based Potts Machine. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design* (Newark Liberty International Airport Marriott, New York, NY, USA) *(ICCAD '24).* Association for Computing Machinery, New York, NY, USA, Article 190, 9 pages. doi:10.1145/3676536.3676720

[13] A. Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014). doi:10.3389/fphy.2014.00005

[14] S. Sreedhara, J. Roychowdhury, J. Wabnig, and P. Srinath. 2023. Digital Emulation of Oscillator Ising Machines. In *Design, Automation & Test in Europe Conference & Exhibition (DATE).* 1–2. doi:10.23919/DATE56975.2023.10137084

[15] NVIDIA Corporation. 2024. CURAND Library. https://docs.nvidia.com/cuda/curand/index.html.

[16] [n. d.]. G-set benchmarks, https://web.stanford.edu/~yyye/yyye/Gset/, Accessed: 2025-03-01.

[17] H. Hoos and T. Stützle. 2000. *SATLIB: An online resource for research on SAT.* 283–292.

[18] C. Cook, H. Zhao, T. Sato, M. Hiromoto, and S. X. D. Tan. 2019. GPU Based Parallel Ising Computing for Combinatorial Optimization Problems in VLSI Physical Design. arXiv:1807.10750 [physics.comp-ph] https://arxiv.org/abs/1807.10750