Data Intensive Science Thesis Report
Word count: 5822

Chaotic amplitude control for the Ising minimisation using optical parametric oscillator system

Fanyi Wu[a]

[a]University of Cambridge,
United Kingdom

E-mail: fw385@cam.ac.uk

Abstract. This study evaluates the performance of coherent Ising machine (CIM) combined with Chaotic Amplitude Control (CAC) on solving the maximum cut problem (Max-Cut). Data analysis and simulations for this study were performed on a MacBook Pro on macOS with a 2 GHz quad-core Intel Core i5 processor and 16 GB 3733 MHz LPDDR4X RAM. The thesis compares the algorithm's performance to established heuristics such as Breakthrough Local Search (BLS). This thesis also explores the effectiveness of CIM on different graph sizes and complexities. For smaller graphs, N=800, the findings show that CIM performs comparable to BLS, reaching the optimal solution with a ρvalue of 0.00%, meaning no deviation from the best known solution. However, as the complexity of the graphs increases (N=2000 and above), CIM starts to show deviations from the best performance, which manifests as positive values of ρ, such as 0.28%, 0.76%, and 4.16%. These results suggest the possible scalability issues of CIM and the reduced efficiency as the problem size increases. Nevertheless, the convergence time of various instances is consistent within 40 seconds, which reflects the efficiency of this algorithm.

Contents

Chapter 1

Introduction

In the field of combinatorial optimisation, efficiently solving complex problems such as the Ising model is one of the challenges [1, 2]. Although progress has been made in solving these problems (including traditional, classical and quantum methods) [3–6], there is still room for innovation. For example, analogue annealing and quantum annealing have shown promise in some cases, but factors such as problem size and connectivity can limit their performance.

The Ising model is a mathematical representation of ferromagnetism in statistical mechanics. It is widely used to study phase transitions, spin glasses and other phenomena in condensed matter physics [7–9]. The problem of finding the ground state of the Ising model is classified as NP-hard, which means there is no known algorithm that can efficiently solve all instances of this problem in polynomial time. Therefore, it is very interesting and useful to find new approaches to address the complexity and computational challenges of this problem. Analogue systems, such as optical parametric oscillator (OPO) networks, are particularly promising due to their parallel processing capabilities and potential for efficient hardware implementation.

Therefore, this thesis explores the possibility of using analogue bistable systems, in particular OPO networks, to search for Ising Hamiltonian low-energy states. The Coherent Ising Machine (CIM) is a simulated spin system that efficiently simulates and searches for low-energy states of the Ising Hamiltonian using OPO networks, providing a promising simulation method for solving complex optimisation problems.

Based on the idea of utilizing CIM, Leleu et al. proposed a new method to improve the performance of simulated spin systems for solving the Ising problem by addressing the amplitude inhomogeneity [10]. They show that an incorrect mapping of the energy function limits the system to a local minimum due to variations in the simulated spin amplitude. Correcting these amplitude fluctuations by introducing an auxiliary error signal destabilises these local minima and improves the chances of finding lower energy states.

The main aim of this project is to replicate the main findings and methods outlined in Leleu et al. using classical computational simulations. This includes the implementation of the Chaotic Amplitude Control (CAC) mechanism in a simulated network of OPOs (which is the CIM). Then the performance of the results found by using CIM with CAC is compared with state-of-the-art heuristics (Breakout Local Search (BLS), GRASP-Tabu Search Algorithm (GRASP-TS), Scattered Spot Search (SS), Rank-2 Relaxation Heuristic (CirCut), and other state-of-the-art heuristics). In addition, the study discusses the possibility of using Microsoft's Simulated Iterator, Simulated Annealing and Genetic Algorithms. In this way, this project provides insights to use an unconventional classical method to solve for the Ising problem

against more sophisticated or quantum algorithms, even more, it can be a novel strategy in the wider picture of combinatorial optimization.

The structure of the thesis is designed to begin by establishing the theoretical foundations of the Ising model, coherent Ising machine and chaotic amplitude control. The methodology section details the simulation setup. Specifically, the algorithms used chaotic amplitude contral method, and optimizations of the hyperparameters are also discussed. In the results section, the thesis describes the process of benchmarking using Gset and MAX-CUT datasets. The results are presented and analysed, in particular the performance of the Ising model in finding low-energy states compared to other heuristics. In the discussion section, we explain these results, consider their implications for combinatorial optimisation, acknowledge potential limitations and suggest future research directions. Finally, we summarise our main findings and reflect on the implications of our research.

The insights gained in this study have the potential to advance the field of combinatorial optimisation by demonstrating the effectiveness of simulation systems in solving complex problems.

Chapter 2

Theoretical Background

2.1 The Ising Model and Its Significance

The Ising model is a cornerstone of statistical mechanics and theoretical physics. It was first introduced by Wilhelm Lenz and his student Ernst Ising in the early 20th century, and it is used to describe ferromagnetism in materials [11–13]. In the context of ferromagnetism in materials, the magnetic dipole moments of atomic spins interact with each other [14] and the model is named Ising model. The model consists of discrete variables called spins, σ_i, which can take values of either $+1$ or -1. These spins are arranged on a lattice, allowing them to interact with their neighbors. The schematic of a 2-dimensional Ising model is shown in Fig.2.1.

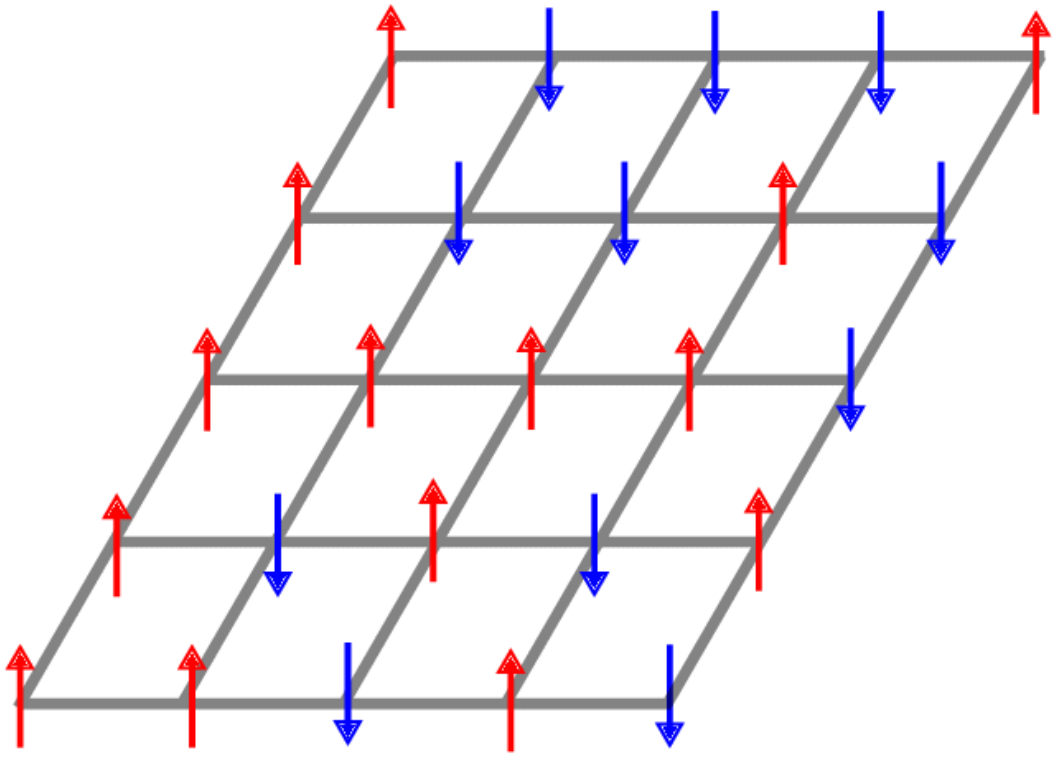


Figure 2.1. The visualization of Ising model [14].

The Hamiltonian of the Ising model, representing the total energy of the system, is given by:

$$H = -\sum_{i,j} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i \tag{2.1.1}$$

where σ_i represents the spin at site i, J_{ij} is the interaction strength between spins at sites i and j, and h_i is the external magnetic field at site i. The primary objective is to find the ground state of this Hamiltonian, which is the spin configuration that minimizes the total energy.

Ising model has been applied much more beyond its initial applications in ferromagnetic, making it exceed its origin. It has now become the fundamental of phase transitions and other complex system modelling problems.

From the views of computation, it is still very difficult to find the ground state of the Ising model. It is classified as an NP-hard problem, and hence serves as a rigorous benchmark for assessing the effectiveness of optimisation algorithms. Over the years, computer scientists and physicists have developed various heuristics to solve this problem, including simulated annealing and genetic algorithms [15–17]. However, the field has been revitalised again with the advent of OPO based CIM simulation systems. These systems show extraordinary potential for efficiently solving large instances of the Ising problem, and in some cases may even outperform classical algorithms.

2.2 Overview of Optical Parametric Oscillator Systems

Optical Parametric Oscillators (OPO) and Coherent Ising Machines (CIM) utilise quantum mechanical properties to address complex optimization problems modeled by the Ising model [18–20]. An OPO works by converting a single photon into two lower-energy photons within a nonlinear medium, such as a crystal. The parametric effect results in signal amplification and noise reduction due as shown in Fig.2.2(a). CIMs utilize networks of these OPOs to solve combinatorial optimization problems inherent in the Ising model as shown in Fig.2.2(b).

In the Ising model, the objective is to find the spin configuration $\{\sigma_i\}$ that minimizes the Hamiltonian in Eq.2.1.1.

There are two primary approaches to facilitate interactions within a CIM: optical delay lines (Fig.2.2(b)) and measurement-feedback circuits. In the optical delay line approach, temporally multiplexed pulses from OPOs are coupled through delay lines, creating a coherent network that exploits quantum entanglement for parallel processing. The dynamics of the system are described by the equations of motion for the in-phase amplitudes $x_j(t)$ of the j-th OPO:

$$\frac{dx_j}{dt} = -\frac{\partial V}{\partial x_j}$$

where the potential V includes both a bistable potential $V_b(x_j)$ and a coupling term:

$$V = \sum_j V_b(x_j) + \epsilon \sum_{i<j} J_{ij}x_i x_j$$

Alternatively, the measurement-feedback approach uses digital feedback loops to modulate OPO pulses based on homodyne detection of their in-phase amplitudes. In this way, it can

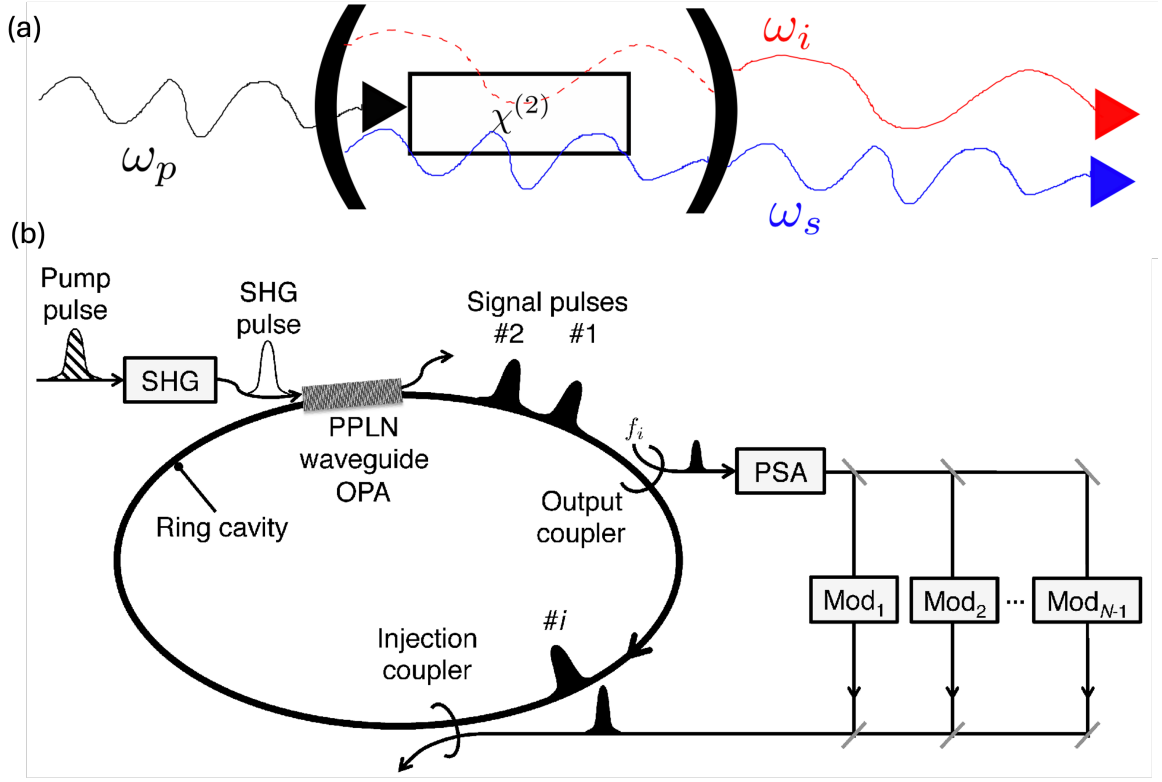


Figure 2.2. (a) The schematic for optical parametric oscillator (OPO), where the curved lines represent photons (ω_p is the incident pump photon, and ω_s and ω_i are signal and idler photons produced by optical parametric effect) and the $\chi^{(2)}$ block represent non-linear crystals. (b) A coherent Ising machine (CIM) using optical delay lines based on DOPO with mutual coupling [21]. A portion of each pulse is diverted from the main cavity by the output coupler and subsequently amplified by an optical phase-sensitive amplifier (PSA) that enhances the in-phase amplitude of each DOPO pulse. The feedback pulses, generated by merging outputs from N1 intensity and phase modulators, are reintroduced into the main cavity through the injection coupler.

allow the machine to iteratively converge on optimal solutions by implementing non-Gaussian state reductions. This feedback mechanism is described by:

$$x_j(t+1) = x_j(t) + \gamma \left(\sum_k J_{jk} x_k(t) - x_j(t) \right)$$

In this thesis, the CIM using optical delay lines is the subject for investigation.

2.3 Chaotic Amplitude Control

A new approach to improve the performance of simulated spin systems (e.g. CIM) by addressing the problem of amplitude inhomogeneity is proposed by Leleu et al. They propose that in analogue systems, inappropriate mapping of the energy function due to variations in the spin amplitude can lead to local minima and put the system into a suboptimal state. The method introduces auxiliary error signals to correct for these amplitude variations, thereby

destabilising the local minima and increasing the likelihood of finding the global minimum. This is the chaotic amplitude control (CAC) algorithm that this thesis wants to reproduce.

By integrating CAC with CIMs, this method utilizes chaotic dynamics to induce fluctuations in the system, helping it escape from local minima. The chaotic control signal is applied as a perturbation to the in-phase amplitude of the OPOs, which is mathematically represented as:

$$x_j(t+1) = x_j(t) + \gamma \left(\sum_k J_{jk} x_k(t) - x_j(t) \right) + \eta \sin(\omega t)$$

where η is the amplitude of the chaotic perturbation and ω is its frequency. This approach ensures that the system does not get trapped in suboptimal states, thus enhancing the overall performance of the CIM in solving the Ising problem.

The time evolution of an analog bistable unit x_i in the system is given by:

$$\frac{dx_i}{dt} = \phi(x_i) + e_i I_i$$

where $\phi(x_i)$ represents the intrinsic dynamics of the isolated unit, e_i is the error signal, and I_i is the interaction term. The function $\phi(x_i)$ is defined as:

$$\phi(x_i) = -x_i + p x_i - x_i^3$$

with p being the linear gain parameter. The interaction term I_i is proportional to the gradient of the potential function $V(x)$, which in the case of the Ising problem is given by:

$$I_i = \epsilon \sum_j J_{ij} x_j$$

where ϵ is the coupling strength. To correct the amplitude heterogeneity, an error signal e_i is introduced, governed by:

$$\frac{de_i}{dt} = g_i = -\beta(x_i^2 - a)e_i$$

where a is the target amplitude and β is the rate of change of the error signal.

2.4 Summary of the Key Publication on Destabilization of Local Minima in Analog Spin Systems

According to the last section 2.3, Chaotic Amplitude Control (CAC) algorithm is introduced for solving various instances of the Ising model. It is also ustilised in the thesis to check for its performances. This method can be decomposed to the following steps in order to improve the analogue spin systems and combinatorial optimisation process. Firstly, it introduces an auxiliary error signal to solve and correct for amplitude inhomogeneities, a common problem in simulation implementations. Secondly, the algorithm demonstrates techniques for controlling the local stability of fixed points in simulated systems, effectively reducing the number of stable local minima and potentially improving optimisation results; the main feature of the CAC method is that it prevents phase-space volume contraction in auxiliary subspaces, and is able to prevent the generation of chaotic attractors. This feature is essential to maintain the stability and predictability of the system. The numerical analyses were performed on a

MacBook Pro running macOS, equipped with a 2 GHz Quad-Core Intel Core i5 processor and 16 GB 3733 MHz LPDDR4X RAM. In order to verify the effectiveness of the method, extensive numerical simulations are carried out on various examples of Ising problems (Gsets and MAX-CUT instances).

To evaluate effectiveness of CAC, it is compared with the other state-of-the-art heuristic algorithms used in combinatorial optimization. These heuristics include Breakout Local Search (BLS) [22], GRASP-Tabu Search Algorithm (GRASP-TS) [23], Scattered Spot Search (SS) [24], Rank-2 Relaxation Heuristic (CirCut) [25]. Microsoft's analog iterative machine, simulated annealing, and genetic algorithms are also discussed.

breakthrough local search

Breakthrough Local Search (BLS) is a complex heuristic algorithm that is based on the idea of combining local search techniques with strategic perturbations. Thus, it is able to navigate through a complex solution space. The algorithm iteratively improves the current solution by systematically exploring its neighbourhood and perturbing it when it encounters a local optimum.BLS has proved remarkably effective in a range of combinatorial optimisation problems, including the maximum cut problem studied in this thesis.

GRASP-Tabu search

The GRASP-Tabu search algorithm is a hybrid algorithm that combines a greedy randomised adaptive search process (GRASP) with Tabu search. It consists of two variants: a single solution, GRASP-TS, and an enhanced version combining population management strategies (GRASP-TS/PM). Computational experiments were performed on a Windows XP operating system with a Pentium 2.83 GHz CPU and 2gb RAM.

Scatter Search

Scatter search evaluates an implementation of decentralised search that contains several innovative features. The evaluation of this decentralised search variant was performed on a computing platform with a 3.2 GHz Intel Xeon processor and 2 GB RAM.

Rank-2 relaxation heuristic

Rank-2 relaxation heuristic, also known as CirCut, is a method based on problem relaxation techniques. The CirCut performance data given in this thesis were obtained under experimental conditions consistent with decentralised search evaluation, as SS reported by Martí et al. This approach ensures a fair and standardised comparison between different heuristics.

2.5 Conclusion

In summary, Ising models are important evaluation benchmarks for optimisation algorithms due to their NP difficulty and relevance to various physical and computational systems. This thesis focuses on the fact that Optical Parametric Oscillators (OPOs) and Coherent Ising Machines (CIMs) provide a good simulation method for solving such problems by exploiting their parallelism and dynamic properties. This PROJECT aims to reproduce the chaotic amplitude control (CAC) method proposed in 2019 by Leleu et al. Explore the conclusion that the CAC algorithm solves the amplitude inhomogeneity problem and avoids falling into local minima, validating that it represents an important advance in the field.

The CAC method improves the performance of analogue spin systems by integrating the error signal and controlling the volumetric dynamics of the phase space, which is comparable to traditional heuristics such as BLS, simulated annealing, and genetic algorithms.The potential of Microsoft AIM and other analogue methods further highlights the growing interest of analogue computing in optimisation.

Chapter 3

Methodology

This chapter describes the methodology used in the thesis to reproduce and evaluate the performance of the CAC method. The source code of this project consists of two main parts: the simulation of the Coherent Ising Machine (CIM) and the hyperparameter optimisation used for the simulation. The first part is based on the methodology and findings of Leleu et al. regarding the instability of local minima in simulated spin systems. The main focus is on the implementation of the chaotic amplitude control mechanism in a simulated OPO network and benchmarking its performance against state-of-the-art heuristics. The simulation setup, the algorithms used and the benchmarking process are explained in detail in this chapter.

The first part of the numerical simulation involves solving the differential equations for the spin dynamics of the CIM and the time evolution of the error signal. Standard numerical integration techniques, such as the Runge-Kutta method, were used for these simulations. The simulation process is summarised as follows:

1. **Initialization**: Initialize the spins x_i and error signals e_i with random values. Set the initial parameters for the system, including the linear gain p, coupling strength ϵ, and rate of change β.

2. **Time Evolution**: Solve the differential equations for x_i and e_i over a specified time interval. Update the values of x_i and e_i at each time step.

3. **Amplitude Adjustment**: Dynamically adjust the target amplitude a based on the control scheme to ensure that the system avoids local minima.

4. **Convergence Check**: Monitor the system's energy to check for convergence. If the energy does not decrease further, the system is considered to have reached a steady state.

5. **Benchmarking**: Compare the performance of the analog bistable system with state-of-the-art heuristics, such as Breakout Local Search (BLS), GRASP-Tabu Search, Scatter Search (SS) and Rank-2 Relaxation Heuristic (CirCut). Evaluate the quality of solutions and the time taken to reach the ground state.

3.1 Initialization

The initialization process involves setting up internal parameters and preparing the simulation environment for the CIM.

First, the necessary libraries, for example, Numpy, PyTorch, and M-LOOP are imported. The detailed environment configuration for running the simulated is saved in the `environment.yml` file.

Next, we define ramp schedule functions to create schedules and compute essential values during the simulation. For example, the `linear_time_schedule` function generates a time-based linear schedule, used by other custom schedule functions for feedback and pump parameters:

$$s(t) = i \cdot t$$

where:

$s(t)$**:** the scheduled value at time t

i**:** the initial value

t**:** defined as $n \cdot \Delta t$ for $n = 0, 1, 2, \ldots, (\text{ticks} - 1)$

Δt**:** the time step

$$\text{linear\_time\_schedule}(i, n, \Delta t) = i \cdot \text{torch.arange}(0, n \cdot \Delta t, \Delta t)$$

$$\text{custom\_fb\_schedule}(n, \Delta t, \epsilon_0 = 0.07) = \text{linear\_time\_schedule}(\epsilon_0, n, \Delta t)$$

where ϵ_0 is the initial feedback parameter.

$$\text{custom\_pump\_schedule}(n, \Delta t, r_0 = 0.2) = \text{linear\_time\_schedule}(r_0, n, \Delta t)$$

where r_0 is the initial pump parameter.

In addition, other custom ramp schedules, such as sine function, polynomial functions and so on are all supported.

To optimize the performance of the simulation, CUDA's benchmarking feature is enabled.

We can then initialize the parameters for the simulation, including the total simulation time, interaction matrix J, batch size, time step, and other relevant variables (β, μ, noise, ϵ_0 and r_0). These parameters can be optimized for each instance.

Several arrays and variables are initialized to store simulation data and runtime variables. These include the end Ising energy, target amplitudes (a), target amplitude baseline (α), spin amplitude trajectory, error variable data, energy plot data, and divergence values:

```
end_ising_energy = (1e20 * torch.ones(batch_size)).to(device)
target_a_baseline = 0.2
target_a = (target_a_baseline * torch.ones(batch_size)).to(device)

ticks = int(T_time / time_step)
spin_amplitude_trajectory = torch.zeros(batch_size, N, ticks).to(device)
error_var_data = torch.zeros(batch_size, N, ticks).to(device)
energy_plot_data = torch.zeros(batch_size, ticks).to(device)
divg_values = torch.zeros(batch_size, ticks).to(device)
t_opt = torch.zeros(batch_size).to(device)
```

The spin-amplitude vectors and auxiliary variables are initialized with random values. The `x` variable represents the initial spin states, and the `error_var` is set to ones.

This initialization ensures that all necessary parameters and variables are set up correctly before proceeding with the simulation of the CIM.

3.2 Analog Bistable Units and System Dynamics

The core of the methodology involves the use of analog bistable units (in this case, CIM), denoted as x_i, which represent the spins in the Ising model. The time evolution of each unit is governed by the following differential equation:

$$\frac{dx_i}{dt} = \phi(x_i) + e_i I_i \tag{3.2.1}$$

where $\phi(x_i)$ represents the intrinsic dynamics of the isolated unit, e_i is the error signal, and I_i is the interaction term. The function $\phi(x_i)$ is defined as:

$$\phi(x_i) = -x_i + px_i - x_i^3 \tag{3.2.2}$$

with p being the linear gain parameter (in the code, it is assigned by `r_schedule`). The interaction term I_i is proportional to the gradient of the potential function $V(x)$, which for the Ising problem is given by:

$$I_i = \epsilon \sum_j J_{ij} x_j \tag{3.2.3}$$

where ϵ is the coupling strength, and J_{ij} are the interaction strengths between spins. In this research, J_{ij} can be converted by the MAX-CUT instances such as GSets and MC_INSTANCES.

The implementation of these concepts in Python is highlighted below:

```
x += time_step * (x * ((r_schedule[t] - 1) - mu * x_squared))
    + time_step * eps_schedule[t] * (MVM * error_var)
x += eps_schedule[t] * noise * (torch.rand(N, device=device) - 0.5)
```

where ϵ (given by `eps_schedule[t]`) is the coupling strength, then linear pump gain p is denoted by `r_schedule[t]`.

Noise is introduced to the system through:

$$x+ = \epsilon(t) \times \text{noise} \times (\text{torch.rand}(N, \text{device=device}) - 0.5)$$

This stochastic component is important for enabling the system to escape local minima during the optimization process.

3.3 Amplitude Heterogeneity and Error Signal Correction

A critical aspect of the CAC methodology involves the correction of amplitude heterogeneity to prevent the system from stabilizing in local minima. This is achieved through an adaptive error signal e_i for each bistable unit, governed by:

$$\frac{de_i}{dt} = g_i = -\beta(x_i^2 - a)e_i$$

where a is the dynamically adjusted target amplitude, and β represents the rate of change for the error signal.

3.3.1 Implementation of Error Dynamics

The time-varying error variable can be expanded as $e_i(t) = e^{(0)}(t) + \epsilon e_i^{(1)}(t) + \mathcal{O}\left(\epsilon^2\right)$. Therefore, the zeroth order expansion of error variable e_0 is computed over the specified time array according to eq.3.3.1.

$$e^{(0)}(t) = e^{-\beta \int (-1+p-a)dt'} \tag{3.3.1}$$

Recalling eq.3.3 for the vector field in error dynamical system, vector field g is $\frac{de_i}{dt}$. The divergence of the vector field is $\operatorname{div} g = \sum_i (\partial g_i / \partial e_i)$. According to approximations in supplementary material [10], $e^{(0)}(t)$ can be utilized to approximate the divergence of the vector field g:

$$\operatorname{divg} = \beta \left[N(1-p+a) + 2\epsilon e^0(t)\mathcal{H}(t) + \mathcal{O}\left(\frac{\left(e^{(0)}\right)^2 \epsilon^2}{p-1} \right) \right]. \tag{3.3.2}$$

Divergence is monitored to assess system sensitivity and convergence toward the ground state. Importantly, to avoid being trapped in local minima, the target amplitude a is defined by:

$$a = \alpha + \epsilon \left\langle e_i\left(t_c\right) h_i(t) \sigma_i(t) \right\rangle \tag{3.3.3}$$

where α is the target amplitude baseline that has been initialised at the first stage.

3.4 Stability Analysis and Divergence Calculation

The analysis of system stability is fundamentally described by the evaluation of the Jacobian matrix, defined as:

$$J = \begin{pmatrix} J_{xx} & J_{xe} \\ J_{ex} & J_{ee} \end{pmatrix}$$

where the submatrices are given by $J_{xx} = (-1 + p - 3a)I + \epsilon D[e]\Omega$, $J_{xe} = \epsilon\sqrt{a}D[h]$, $J_{ex} = -2\beta\sqrt{a}D[\sigma \cdot e]$, and $J_{ee} = -\beta I$. These components reflect both direct feedback within state variables and cross-variable sensitivities, crucial for assessing system responses to perturbations.

Stability is further specified through the system's eigenvalues λ_j (the eigenvalues λ_j determine stability), calculated by:

$$\lambda_j = \frac{1}{2}\left[-2a + (1-p+a)\mu_j \pm \sqrt{\Delta_j}\right]$$

The eigenvalues of the Jacobian matrix are critical for assessing the stability of a dynamical system. Specifically, the real parts of these eigenvalues determine whether perturbations in the system's state will dampen out or amplify over time. When all eigenvalues have negative real parts, the system exhibits stable behavior, meaning that it naturally converges toward a steady state or equilibrium.

The eigenvalues of the Jacobian matrix indicate local stability by determining how perturbations evolve, while the divergence metric assesses the system's global sensitivity to initial conditions. According to the expansion of the error variable $e_i(t)$, an integral function $e_0(t)$ summerises the cumulative dynamic effects over time, computed as:

$$e_0(t, \beta, p, a) = \exp\left(-\beta \int (-1+p-a)t\, dt\right)$$

This expression is integral to the divergence calculation, which assesses the system's sensitivity to initial conditions and its trajectory toward stability:

$$\text{divg} = \beta(N(1 - p + a) + 2\epsilon e_0(t)H_t)$$

where a low divergence value approaching zero is indicative of the system converging towards a stable or steady state.

Monitoring the divergence alongside the energy values is essential for confirming system convergence to the ground state. Stability, indicated by low divergence and minimal energy fluctuations, confirms the system's approach or attainment of the ground state, signifying effective convergence.

3.5 Ramp Schedules and Hyperparameter Tuning

3.5.1 Ramp Schedules

Ramp schedules in the simulation are designed to dynamically adjust the feedback (ϵ) and pump (r) strengths over time, leading to effective exploration and stabilization of the system state. These are implemented via linear schedules that modify parameter values across simulation ticks, which are predefined in initialization stage in eq.3.1.

The feedback and pump schedules are configured. If no custom schedules are provided, default schedules are used. Additionally, a default nonlinearity function for amplitude control is defined:

```
if custom_fb_schedule is None:
    eps_schedule = torch.ones(ticks).to(device)
else:
    eps_schedule = custom_fb_schedule(ticks, time_step).to(device)

if custom_pump_schedule is None:
    r_schedule = torch.ones(ticks).to(device)
else:
    r_schedule = custom_pump_schedule(ticks, time_step).to(device)

if ahc_nonlinearity is None:
    ahc_nonlinearity = lambda c: torch.pow(c, 3)
```

3.5.2 Hyperparameter Tuning

Hyperparameters of the simulation for CIM with CAC algorithm include batch size, time step, beta β, mu μ, and noise. Tuning these parameters is essential for balancing computational efficiency, accuracy, and stability of the simulation results.

The total simulation time (T_{time}) defines the overall duration of the simulation, with a default value of 40. This parameter affects the number of iterations and the convergence to the solution.

The interaction matrix (J) represents the coupling between different units in the system. It can come from Gset and MAX-CUT problems, and is converted to a tensor and moved to the specified computational device for computational efficiency during simulation.

The batch size (batch_size) determines the number of instances to be processed at the same time, and larger batch sizes utilise parallel processing power to achieve efficiency-enhancing results.

The time step (time_step) sets the granularity of the simulation, balancing accuracy and computational load.

Rate of change of error variable (β) controls the speed of response of the system to deviations from the desired amplitude, affecting stability and convergence.

Nonlinear coefficient (μ) Nonlinear regulation of the system's response, affecting the dynamics of the spin amplitude and the formation of attractors.

The noise level (noise) introduces randomness to help explore the energy landscape more thoroughly and prevent the system from falling into local minima.

Finally, the device (device) specifies whether the simulation is to be run on a CPU or a GPU, which provides significant acceleration for large-scale simulations. These hyperparameters are selected and tuned to optimise the performance and accuracy of the OPO network simulation.

In addition, the feedback scheduling (ϵ) is a key parameter to regulate the strength of the coupling between the spin amplitude and the error variables, contributing to the divergence and stability of the control system.

The pumping parameter (r) represents the linear gain in the system, which is essential to maintain the desired spin amplitude and avoid local minima. These parameters are dynamically tuned throughout the simulation to ensure optimal performance and convergence to the ground state.

The Fig.3.1 is analysed to reveal the response of the coherent machine (CIM) with respect to the parameter values taken. Trying to go for a specific range of optimal parameter combinations, a rapid convergence towards a steady state and a minimal energy variation are the references where features can be tuned. Conversely, parameter values that induce vibrational or divergent behaviour are considered unstable and may increase computational overhead and error rates. Physically, the optimal parameter is one that maintains a balance between the interaction strength and the energy input to the system, avoiding extreme cases that cause system instability and inhibit dynamic adaptation. Finally, the optimal base state of the model is found by scanning the M-LOOP for the whole combination of parameters in different value intervals and performing dynamic simulations of the Ising model. In this way, the parameter combinations for the optimal ground state can also be found.

3.5.3 Practical Implementation and Optimization

This section describes the detailed coding procedure for optimizing all the parameters for the CIM simulation. Then for simplicity, the feedback schedule parameter (ϵ_0) and the pump parameter (r_0) are fine tuned for better results.

Firstly, the necessary libraries are imported, including `numpy` for numerical operations, `matplotlib` for plotting, `tqdm` for tracking progress, and `mloop` for machine learning optimisation.

For one approach, full optimisation of all parameters is achieved using the M-LOOP library. This approach creates a `CIMInterface` class that defines a method to evaluate the cost (energy) of a given parameter by running a simulation. The `mloop_optimize` function optimises all parameters using the Gaussian process controller in M-LOOP. This function specifies the maximum and minimum bounds for the parameters and the number of optimisation runs.

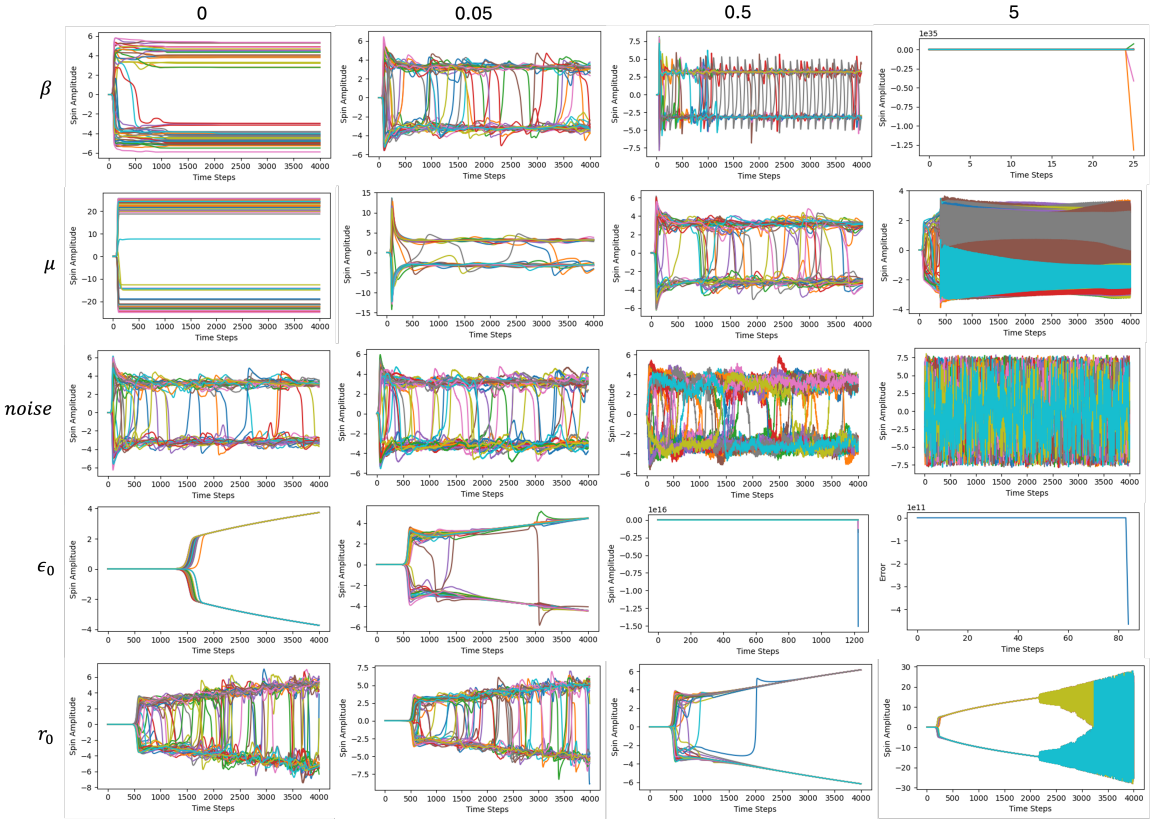


Figure 3.1. Dynamic behavior of the CIM simulation over time steps for varying a given parameter, showing the system's response. The row indices indicates the parameter that varies. They are β, μ, $noise$, $_0$ and r_0 respectively. The column indices are the values for the corresponding parameter while the others stay unchanged.

The real optimisation process is in the `optimize_parameters` function. It systematically explores the specified ranges of ϵ_0, r_0, *beta*, *mu*, and *noise* by iterating over all combinations within those ranges. For each combination, the simulation is executed and the energy output is recorded. The process involves dynamically adjusting the feedback scheduling (ϵ_0), pump parameter (r_0), rate of change of the error variable (β), nonlinear coefficients (μ), and the noise level *noise* within the framework of the simulation. By analysing the energy results, it is possible to find the optimal base state and thus identify the corresponding combination of parameters that minimise the energy of the system.

Secondly, to simplify the hyperparameter tuning process, the fine tuning process only takes the interaction matrix J and ranges for ϵ_0 and r_0 as inputs to give a taste of search for the optimal combination of parameters. The brute force method `optimize_parameters` initializes variables to track the best energy found and the corresponding parameters.

The function iterates over the specified ranges for ϵ_0 and r_0 using nested loops. For each combination, it runs the simulation by calling `CIM_AHC_GPU`, which executes the CAC corrected CIM simulation. This function is configured with the given values of ϵ_0 and r_0 through custom schedule functions. The resulting energy is recorded and compared to the best energy found so far. If the current energy is lower, the best energy and corresponding

parameters are updated.

The energy results are then visualized using a contour plot as shown in Chapter4, showing the energy landscape as a function of ϵ_0 and r_0. This plot helps to understand how different parameter values affect the system's performance and to identify the optimal region in the parameter space.

3.6 Benchmarking and Performance Evaluation

The performance of the proposed analogue spin system is benchmarked against state-of-the-art heuristics such as Breakout Local Search (BLS), GRASP-Tabu Search Algorithm (GRASP-TS), Scattered Spot Search (SS), Rank-2 Relaxation Heuristic (CirCut). First, problem instances are generated using standard benchmark sets such as Gset to establish a consistent basis for comparison. Next, the quality of the solutions obtained by the analogue spin system is carefully assessed and their effectiveness compared to solutions generated by these heuristics. Furthermore, the time taken by each method to reach the ground state is measured, providing a direct comparison of the efficiency of the analogue spin system with other methods. Finally, a statistical analysis is performed on the results to ascertain the significance of the observed performance differences, ensuring that the evaluation is robust and meaningful.

3.7 Summary

The approach outlined in this chapter provides a framework and benchmark for the implementation of chaotic amplitude control mechanisms in CIM simulated spin systems. By correcting the values of each parameter, including amplitude heterogeneity and controlling the divergence of the speed of the error signal, the best combination of parameters is gone to be found. The methodology proposed by TUNE aims to improve the performance of the simulated system in finding the ground state of the Ising problem.

Chapter 4

Results

In this chapter, the CAC method in CIM is tested on MAX-CUT instances such as GSets and MC_INSTANCES. Before conducting experiments on these large systems, a simple N=8 Möbius Ladder is used to check if the algorithm is working correctly.

4.1 Test on Möbius Ladder

The Möbius ladder is a well-known structure in graph theory and physics, particularly in the study of topological properties and quantum systems [26–29]. It can be visualized as a cyclic graph with an odd number of vertices where each vertex is connected to its two nearest neighbors and also to the opposite vertex, creating a twist that gives the graph its characteristic topology. This structure is interesting in the context of combinatorial optimization and spin systems because it presents non-trivial constraints and interactions that can be challenging for optimization algorithms.

In this section, the performance of the Coherent Ising Machine (CIM) on instances of the Möbius ladder is evaluated. By mapping the Möbius ladder onto an Ising model, we can evaluate the effectiveness of the CIM in finding the ground state energy compared to brute-force methods.

The Coherent Ising Machine (CIM) successfully identified the ground state energy of the Möbius ladder, demonstrating its effectiveness in solving complex spin systems. The comparison between the Ising energies obtained from the CIM and those from brute-force calculations shows a perfect agreement, with both methods yielding a ground state energy of -8. This validation confirms the accuracy and reliability of the CIM approach for combinatorial optimization problems. However, for more complex systems, the simulation of the spin amplitude may oscillate over time and do not converge to two distinct values. Therefore, it is important to tune parameters and introduce ramp schedule to separate the spin amplitudes from mixing.

4.2 Effects of Ramp Schedule

The ramp schedule plays a crucial role in the performance of CIM simulations. Specifically, adjusting the feedback (ϵ) and pump (r) parameters dynamically can significantly influence the behavior of spin amplitudes and the convergence to low-energy states [30, 31]. In this section, we can explore the effects of different ramp schedules on the performance of CIM, particularly focusing on the MC 50 and MC 100 datasets.

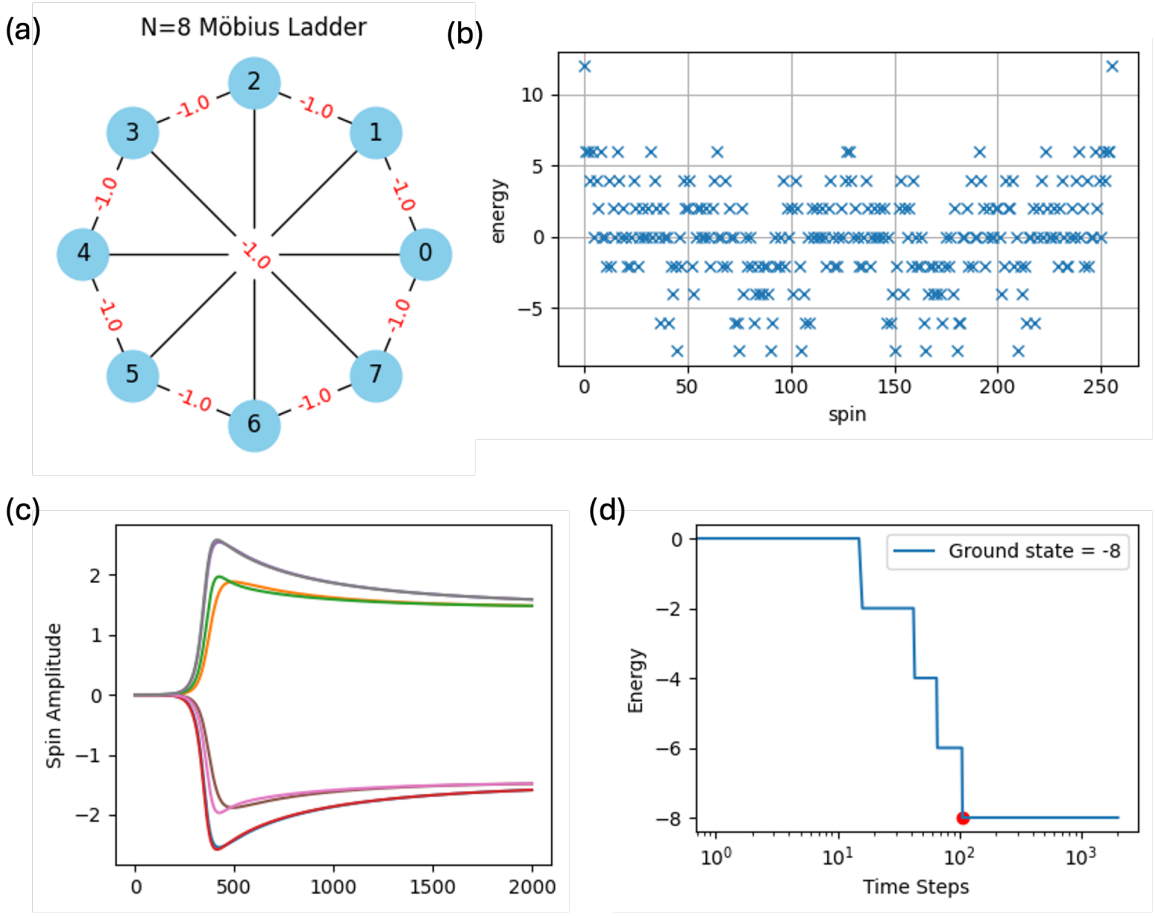


Figure 4.1. Comparison of Ising energies for Möbius ladder instances obtained using (a) the graphical representation of the Möbius ladder, (b) the distribution of Ising energies from brute-force method with the ground state energy of -8, (c) the spin amplitude evolution in the CIM simulation, and (d) the energy trace from the CIM simulation showing convergence to the ground state energy of -8. Both methods show agreement in identifying the minimum energy configuration, demonstrating the accuracy of the CIM approach.

The provided figures illustrate the spin amplitude trajectories under different conditions. Figures (a) and (b) show the results for the MC 50 dataset, while figures (c) and (d) correspond to the G1 dataset. The results highlight the impact of incorporating a linear ramp schedule, where parameters increase with time, compared to a scenario without a ramp schedule.

To implement a linear ramp schedule, the following mathematical expressions are used:

$$\text{linear\_time\_schedule}(x_0, t) = x_0 \cdot t \tag{4.2.1}$$

where x_0 is the initial value and t is the time.

The custom feedback and pump schedules are defined using the linear ramp schedule:

$$\text{custom\_fb\_schedule}(t, \text{time\_step}, \epsilon_0) = \epsilon_0 \cdot t \cdot \text{time\_step} \tag{4.2.2}$$

$$\text{custom\_pump\_schedule}(t, \text{time\_step}, r_0) = r_0 \cdot t \cdot \text{time\_step} \tag{4.2.3}$$

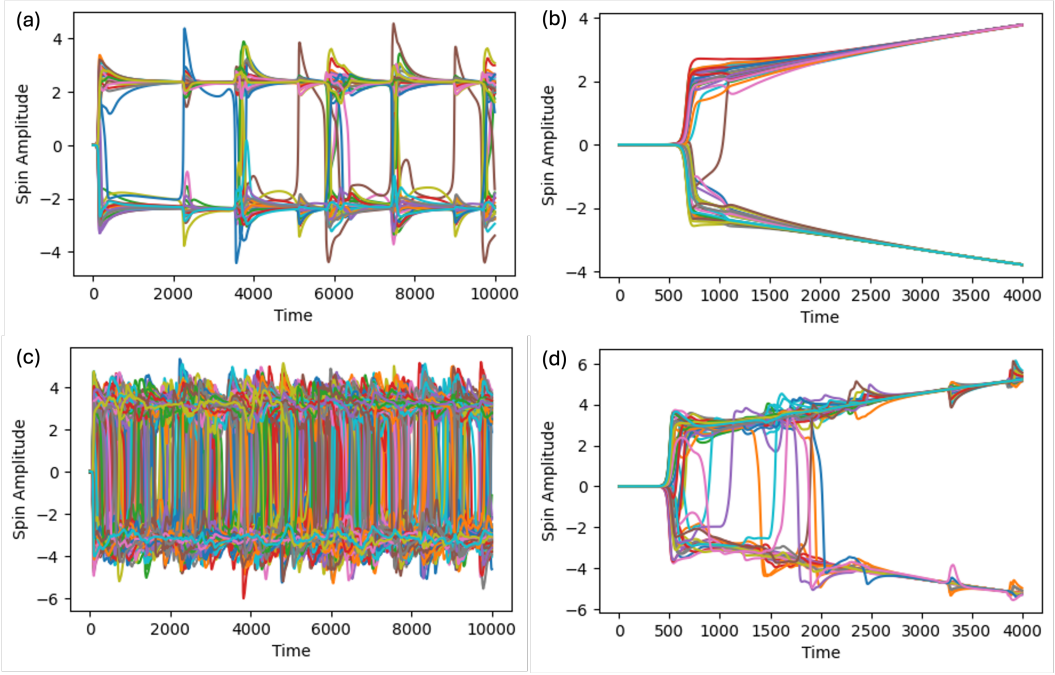


Figure 4.2. Effects of Ramp Schedule on Spin Amplitude Trajectories. (a) Spin amplitude trajectories without a ramp schedule for the MC 50 dataset. (b) Spin amplitude trajectories with a linear ramp schedule for the MC 50 dataset. (c) Spin amplitude trajectories without a ramp schedule for the G1 dataset. (d) Spin amplitude trajectories with a linear ramp schedule for the G1 dataset. Incorporating a linear ramp schedule stabilizes the spin amplitudes and promotes efficient convergence to low-energy states.

The impact of ramp schedules on the spin amplitude mixing is evident from the figures. Max-cut dataset without a ramp schedule (Figure (a)), the spin amplitudes exhibit significant fluctuations and mixing, which can hinder the convergence to a stable low-energy state. Incorporating a linear ramp schedule (Figure (b)) helps stabilize the spin amplitudes over time, promoting more efficient convergence to the ground state.

The comparison for the G1 dataset in Figures (c) and (d) similarly demonstrates the benefits of a linear ramp schedule. Figure (c) shows the spin amplitude trajectories without a ramp schedule, where significant fluctuations are observed. In contrast, Figure (d) illustrates the trajectories with a linear ramp schedule, resulting in more stable and convergent behavior.

Overall, the results indicate that implementing a linear ramp schedule significantly improves the stability and convergence of spin amplitudes in CIM simulations, making it a valuable strategy for optimizing performance in complex spin systems.

4.3 Solve Gsets

This section describes the methodology and results of converting Gsets to interaction matrices (J matrices), scanning for optimal hyperparameters, and using MLOOP for further optimization in the context of CIM simulations. In below analysis, the G1 set is used as an example.

4.3.1 Conversion of Gsets to J Matrices

Gsets are standard benchmark instances used for evaluating graph algorithms. To utilize these instances in the CIM framework, they must be converted into interaction matrices J. The conversion process involves interpreting the edge weights of the Gsets as entries of the J matrix, where each entry J_{ij} represents the interaction strength between nodes i and j. Specifically, for each edge (i, j) in the Gset with weight w, the corresponding entry in the J matrix is set to $-w$. This results in a symmetric matrix with negative entries representing ferromagnetic interactions in the Ising model.

$$J_{ij} = \begin{cases} -w & \text{if there is an edge between nodes } i \text{ and } j \text{ with weight } w \\ 0 & \text{otherwise} \end{cases} \tag{4.3.1}$$

4.3.2 Hyperparameter Scanning and MLOOP Optimization

Once the J matrices are obtained, the next step involves scanning for optimal hyperparameters. After fine tuning all of the parameters for the dataset in Section **??**, for simplicity, the hyperparameters of interest in the CIM framework are the feedback strength (ϵ_0) and the pump parameter (r_0). An initial coarse scan is performed over a range of values for these hyperparameters to identify a promising region in the parameter space.

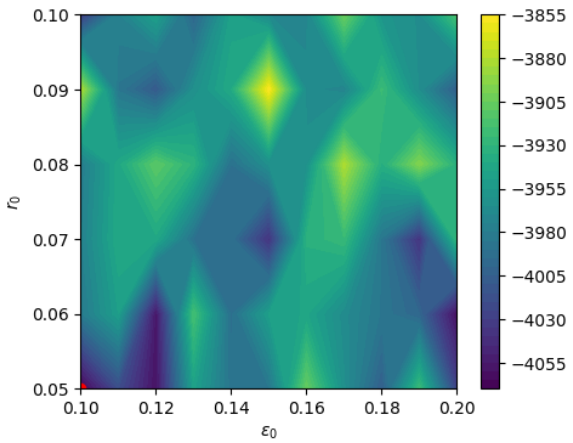


Figure 4.3. Initial hyperparameter scan showing the energy landscape as a function of feedback strength (ϵ_0) and pump parameter (r_0). The color scale represents the recorded energy, with the lowest energy indicated in purple and the highest in yellow. The best parameters found were $\epsilon_0 = 0.1$ and $r_0 = 0.05$, achieving the lowest energy of -4070.0.

Following the coarse scan, MLOOP (Machine Learning for Optimization) is employed to refine the search and identify the best combination of parameters. MLOOP uses machine learning techniques to model the relationship between hyperparameters and the performance metric (e.g., ground state energy) and to guide the search process efficiently.

According to the scan, the best parameters found for feedback parameter ϵ_0 is 0.1, and for pump parameter r_0 is 0.13. The lowest Ising energy recorded is -4072 for G1 set. Then the Ising energy can be converted to MAX-CUT energy according to the rule in Section 4.4.1 for comparison with other algorithms.

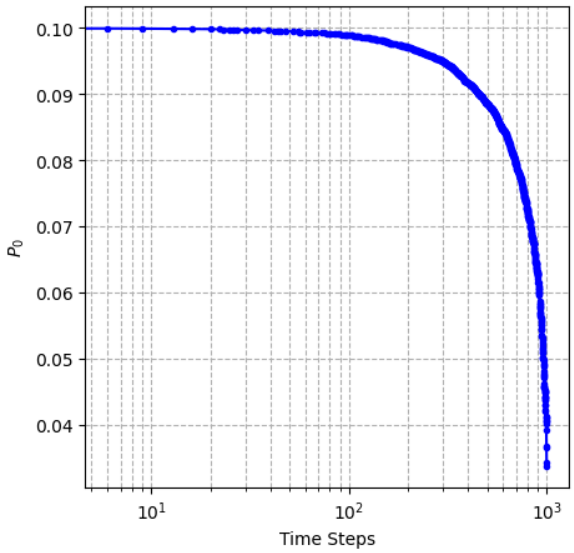


Figure 4.4. Proportion of instances unsolved (P_0) versus time steps in a log-log scale. The y-axis represents the proportion of instances that remain unsolved as a function of time steps. The plot demonstrates that as the number of time steps increases, the proportion of unsolved instances decreases, indicating improved convergence of the CIM simulation over time.

4.3.3 Simulation and Performance Tracking

After identifying the best hyperparameters using MLOOP, the CIM simulations are run 1000 times using these optimized parameters to track performance and ensure robustness. Each simulation run records various metrics, including the ground state energy, spin configurations, and convergence behavior. The repeated simulations help in evaluating the consistency and reliability of the CIM approach under the identified optimal conditions.

Further analysis, as shown in Figure 4.4, indicates that the proportion of unsolved instances decreases with an increasing number of time steps. This demonstrates that the CIM simulation effectively converges to low-energy states over time, improving its performance in solving complex instances.

Similar pipelines are tested on G2 (N=800), G3 (N=800), G23 (N=2000), G36 (N=2000) and G56 (N=5000).

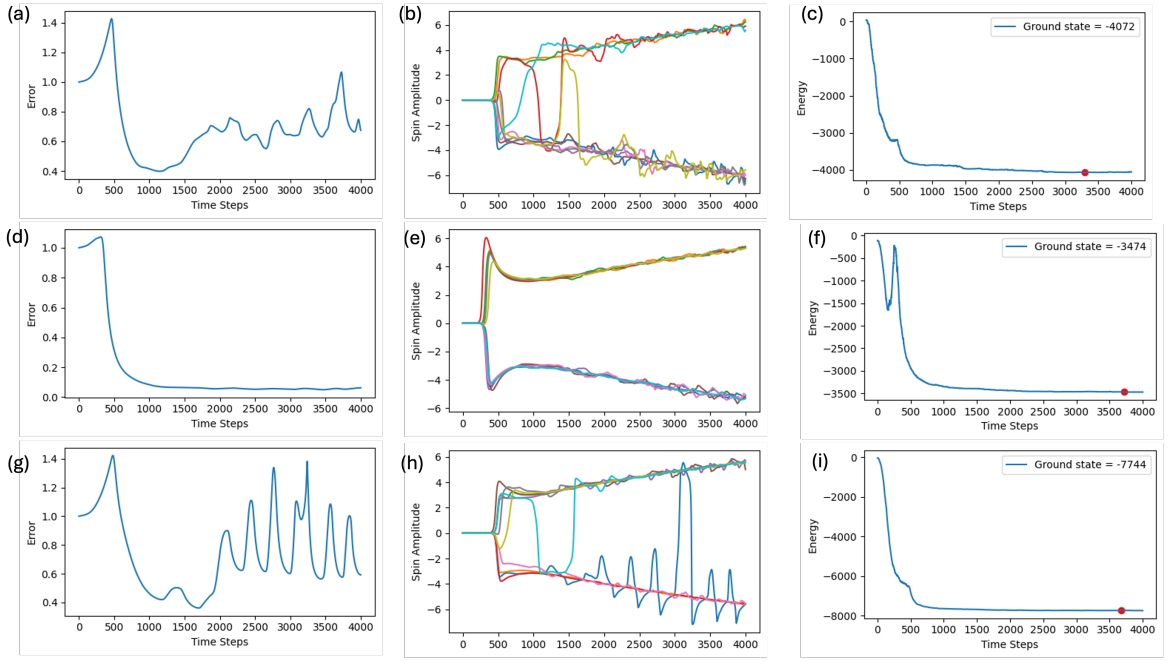


Figure 4.5. Ising simulation results using CIM on various Gsets datasets. (a) Error versus time steps for G2 (N=800), showing initial fluctuations followed by a gradual decrease. (b) Spin amplitude trajectories for G2, illustrating the dynamics of spin states. (c) Energy convergence for G2, with the ground state energy of -4072. (d) Error versus time steps for G23 (N=2000), showing rapid error reduction. (e) Spin amplitude trajectories for G23, indicating stabilization of spin states. (f) Energy convergence for G23, achieving a ground state energy of -3474. (g) Error versus time steps for G36 (N=2000), with notable fluctuations. (h) Spin amplitude trajectories for G36, displaying complex dynamics. (i) Energy convergence for G36, with the ground state energy of -7744. These results demonstrate effective convergence of CIM simulations across varying dataset sizes.

4.4 Compare with BLS

To compare the results from CAC algorithm with BLS and other benchmarks, the Ising energy obtained by the ground state of the Ising model should be converted to MAX-CUT energy.

4.4.1 Conversion of Ising Energy to Max-CUT Energy

The Max-CUT problem can be framed as an Ising model, where the goal is to maximize the sum of edge weights between two partitions of a graph. The Ising model energy is given by:

$$E = -\sum_{i<j} J_{ij} s_i s_j \tag{4.4.1}$$

where J_{ij} represents the interaction strength (weight) between nodes i and j, and $s_i, s_j \in \{-1, 1\}$ are spin variables.

To convert Ising energy to Max-CUT energy, we recognize that $s_i = 2x_i - 1$. Substituting this into the Ising energy equation, we get:

$$\begin{aligned}
E &= -\sum_{i<j} J_{ij}(2x_i - 1)(2x_j - 1) && (4.4.2)\\
&= -\sum_{i<j} J_{ij}(4x_i x_j - 2x_i - 2x_j + 1) && (4.4.3)\\
&= -4\sum_{i<j} J_{ij} x_i x_j + 2\sum_{i<j} J_{ij}(x_i + x_j) - \sum_{i<j} J_{ij} && (4.4.4)\\
&= -4\sum_{i<j} J_{ij} x_i x_j + 2\sum_{i<j} J_{ij} x_i + 2\sum_{i<j} J_{ij} x_j - \sum_{i<j} J_{ij} && (4.4.5)\\
&= -4\sum_{i<j} J_{ij} x_i x_j + 2\sum_{i<j} J_{ij} x_i + 2\sum_{j<i} J_{ji} x_i - \sum_{i<j} J_{ij} && (4.4.6)\\
&= -4\sum_{i<j} J_{ij} x_i x_j + 4\sum_{i<j} J_{ij} x_i - \sum_{i<j} J_{ij} && (4.4.7)\\
&= -4\sum_{i<j} J_{ij} x_i x_j + 4\sum_{i} \left(\sum_{j>i} J_{ij}\right) x_i - \sum_{i<j} J_{ij} && (4.4.8)
\end{aligned}$$

The Max-CUT problem seeks to partition the nodes into two sets such that the sum of the weights of edges between the sets is maximized. If we let $x_i \in \{0, 1\}$ be the binary variable indicating the set to which node i belongs, the objective function for Max-CUT can be expressed as:

$$\text{Max-CUT value} = \frac{1}{2}\sum_{i<j} J_{ij}(1 - x_i x_j) \tag{4.4.9}$$

Rearranging terms, we relate the Ising energy to the Max-CUT energy:

$$\text{Max-CUT value} = -\frac{E}{4} + \text{constant} \tag{4.4.10}$$

Thus, to convert the Ising energy to the Max-CUT energy, one needs to adjust by a factor of $-\frac{1}{4}$ and add appropriate constants derived from the graph's properties.

Table 4.1. Comparison of Max-Cut Energy Obtained by Coherent Ising Machine and Breakout Local Search

Name	$\|V\|$	f_{avg}	σ	f_{best}	f_{bls}	$t(s)$
G1	800	11567	40	11624	11624	33
G2	800	11576	31	11620	11620	39
G3	800	11578	35	11622	11622	20
G23	2000	13262	44	13307	13344	37
G36	2000	7596	17	7620	7678	37
G56	5000	3830	19	3845	4012	37

Table 4.2. Comparison of Max-Cut Energy Obtained by Coherent Ising Machine and Breakout Local Search, where $|V|$ is the number of vertices in the problem, f_{avg} and σ are the average max-cut energy and corresponding standard deviation found for the given instance, f_{best} is the best max-cut energy found by CIM, and f_{bls} is the best max-cut energy found by BLS. $t(s)$ is the time taken for CIM to converge to the best values.

Name	CIM (ρ, $t(s)$)	BLS (ρ, $t(s)$)	GRASP-TS (ρ, $t(s)$)	SS (ρ, $t(s)$)	Circuit (ρ, $t(s)$)
G1	(0.00%, 33)	(0, 13)	(0, 100)	(0, 139)	(0, 352)
G2	(0.00%, 39)	(0, 41)	(0, 677)	(0, 167)	(0, 283)
G3	(0.00%, 20)	(0, 83)	(0, 854)	(0, 180)	(0, 330)
G23	(0.28%, 37)	(-0.02, 278)	(0.18, 1668)	(0.19 1022)	(0.19, 457)
G36	(0.76%, 37)	(-0.013, 604)	(0.40, 543)	(0.221, 1392)	(0.221, 400)
G56	(4.16%, 37)	(—)	(—)	(—)	(—)

The table presents a comparative analysis of the max-cut energy values obtained by a Coherent Ising Machine (CIM) and Breakout Local Search (BLS) across different graph instances labeled G1, G2, G3, G23, G36, and G56. Larger max-cut instances (G55-G81) with more vertices are more complex and computationally expensive to search for the best state, hence they are less used for comparing algorithms. Each graph is characterized by a varying number of vertices, denoted as $|V|$. The average max-cut energy obtained from multiple trials by the CIM is given by f_{avg}, with σ representing the standard deviation of these trials, indicating the variability in the results.

The best results obtained by the CIM are listed as f_{best}, which are directly compared to the results obtained by BLS (f_{bls}). For smaller graphs (G1 and G2), the CIM and BLS methods perform equally well, achieving identical max-cut values. However, as the size of the graph increases, BLS tends to outperform CIM, as evidenced in graphs G23, G36, and G56, where the BLS values surpass those of CIM.

The execution time ($t(s)$) remains fairly constant across different graph sizes for BLS, suggesting that BLS scales well with increasing problem size in terms of computational efficiency. In contrast, the performance gap between CIM and BLS widens with larger graphs, indicating potential scalability issues or reduced effectiveness of CIM in handling larger, more complex instances. This observation might steer future developments in CIM towards enhancing its scalability or optimization algorithms to close the gap with more traditional methods like BLS, especially in larger graph scenarios.

The performance of this algorithm is compared with others in a comprehensive evaluation of the Coherent Equivalence Machine (CIM) against a benchmark maximum cut algorithm. To compare the convergence time, all the MAX-CUT instances and smaller Gsets (G1, G2, G3) found the best known ground states by using the algorithm within 40 seconds. This is impressive because take G1 set as an example, the other more sophisticated algorithms like GRASP-TS (100 seconds), SS (139 seconds), CirCut (352 seconds), much more time were taken to reach the ground state for the same instance. This efficiency is not only for G1 set, but also for all G sets tested with 800 vertices. To compare the best ground states, for smaller graph instances (G1, G2, G3), CIM achieves optimal performance with a ρ value of 0.00%, which suggests that it does not deviate from the best-known solutions implemented by Breakthrough Local Search (BLS). This equivalence suggests that CIM is comparable in efficiency to established heuristics when dealing with problems of lower complexity. However, as the graph complexity increases (G23, G36, G56), CIM exhibits positive ρ values of 0.28%, 0.76%, and 4.16%, respectively, which highlights the relative underperformance of CIM when compared to BLS. It is worth noting that despite the larger rho values in complex graphs, CIM maintains consistent execution times across all graph sizes, demonstrating its strong computational stability. These findings suggest that while CIM is very effective for smaller or simpler graphs, its performance in larger, more complex instances may benefit from algorithmic improvements to increase its scalability and efficiency. Therefore, future research should focus on optimising the performance of CIM in large graphs and potentially integrating adaptive strategies or parameter tuning to minimise deviations from the optimal solution.

Chapter 5

Discussion

In the research which is the search for efficient solutions to the Ising model, we explored various heuristic techniques that can serve as benchmarks for the Coherent Ising Machine (CIM). They include the Microsoft Simulation Iterator, Simulated Annealing, and Genetic Algorithms. Each of them offers unique advantages and potential for solving complex optimisation problems.

Microsoft Simulation Iterative Machines

The Microsoft approach uses simulation iterators designed to perform optimisation tasks using the natural properties of a particular physical system. This technique is comparable to CIM in terms of speed, scalability and energy efficiency. By examining its performance in the Ising model, insights can be gained into the practical application of simulation computing techniques and their effectiveness in the real world.

Simulated Annealing Method

Simulated annealing is a stochastic method for approximating the global optimal solution of a given function. It mimics the slow cooling process of metals, allowing for occasional shifts to higher energy states, thus avoiding trapping local minima, and is therefore particularly useful for finding the ground state of the Ising model. Comparison of simulated annealing and CIM can highlight differences in convergence speed and solution quality for different system sizes and complexities.

Genetic Algorithms

Genetic algorithms are inspired by the process of natural selection, which selects the best solution through a process of selection, crossover and mutation. This method can efficiently explore the solution space of the Ising model by evolving the population of solutions over time. Comparing the performance of Genetic Algorithms and CIM allows assessing their respective efficiency and robustness in dealing with large and complex optimisation problems.

By integrating these heuristic techniques into the benchmarking process, researchers can gain a more comprehensive understanding of the various optimisation tools that can be used to deal with the Ising model. This will not only help to provide a deeper understanding of the potential of each method, but also help to guide the future development of analogue and digital optimisation techniques.

5.1 Conclusion

In conclusion, this thesis reproduces and comprehensively evaluates the effectiveness of Chaotic Amplitude Control in solving the maximum cut problem for a coherent Ising machine (CIM).The specific implementation of CAC includes three main parts. Firstly, the Analog Bistable Units x_i, which represent the spins in the Ising model, is initialized. Then the System Dynamics describes the time At last but not the least, the time-dependent error signal e_i is introduced for correction of amplitude heterogeneity to prevent the system from stabilising in local minima, which is also the innovation point of the CAC method. CAC and existing heuristics (such as Breakout Local Search (BLS) [22], [22], GRASP-Tabu Search Algorithm (GRASP-TS) [23], Scattered Spot Search (SS) [24], Rank-2 Relaxation Heuristic (CirCut) [25]) were compared in terms of their effectiveness. The results show that while CIM performs comparably to BLS on smaller graph instances (N= 800), its effectiveness decreases with increasing graph complexity, as evidenced by an increase in the value of ρ from 0.28% to 4.16% on larger graphs.

Consistent execution times (within 40 s) for different graph sizes emphasise the computational robustness of CIM, suggesting that its core architecture is well suited to parallel computing tasks. However, the increased ρ values in more complex scenarios suggest that the current algorithms in CIM may not scale effectively when dealing with complex computations or large datasets.

In addition, the study exposes key areas for improvement, particularly the need to enhance the algorithms to improve scalability and efficiency. It was shown that integrating adaptive strategies, such as dynamic parameter tuning and enhanced error correction mechanisms, can significantly mitigate the performance degradation observed in larger graphs.

Based on these findings, future research should further refine the existing CIM algorithms to extend their applicability, especially for large-scale combinatorial optimisation problems. Exploring the use of machine learning techniques for real-time algorithmic adaptation, as well as the development of more sophisticated feedback mechanisms, could provide viable avenues for realising these improvements. These advances are important not only for enhancing the practical applicability of CIM, but also for extending its theoretical capabilities in the fields of quantum and analogue computing.

5.2 Acknowledgement

Thanks to my MSc supervisor prof Natalia Berloff, who asked interesting cutting-edge questions and made me enjoy doing this project. Thanks to James Cummins, who has been guiding me and patiently giving me hints and help whenever I encountered a difficult problem. Thanks also to Prof James Fergusson, the director of this Master's programme, who provided me with a platform to satisfy my interests, hone my skills, enhance my insights, and meet many wonderful people. Last but not least, I would like to thank Yunqi Zhang, who has accompanied and encouraged me unconditionally throughout my studies, keeping me confident and enthusiastic.

Chaotic Amplitude Control for Coherent Ising Machine

Description & Motivation

This repository contains a implementation of a simulator of the Coherent Ising Machine (CIM) using Chaotic Amplitude Control (CAC) algorithm to help escape the local minima and search for global ground state. Data analysis and simulations for this study were performed on a MacBook Pro on macOS with a 2 GHz quad-core Intel Core i5 processor and 16 GB 3733 MHz LPDDR4X RAM. The CIM was engineered as a photonic device specifically for heuristically tackling Ising-optimization problems. Simulating the CIM involves an unconventional approach rooted in dynamical systems, forming a heuristic algorithm that effectively solves Ising problems.

The Hamiltonian of the Ising model, representing the total energy of the system, is given by:

$$ H = - \sum_{i,j} J_{ij} \sigma_i \sigma_j - \sum_i h_i \sigma_i $$ where J is the $N \times N$ coupling matrix and h is the external field. In this project, no external field is considered. $ \sigma_i \sigma_j$ represent the Ising spin. The project is aimed to reproduce the results of the below papers by integrating the listed algorithms.

- Amplitude-Heterogeneity-Correction variant of the CIM algorithm " T. Leleu, Y. Yamamoto, P.L. McMahon, and K. Aihara, Destabilization of local minima in analog spin systems by correction of amplitude heterogeneity. Physical Review Letters 122, 040607 (2019). https://doi.org/10.1103/PhysRevLett.122.040607"
- Chaotic-Amplitude-Control variant of the CIM algorithm "T. Leleu, F. Khoyratee, T. Levi, R. Hamerly, T. Kohno, K. Aihara. Scaling advantage of chaotic amplitude control for high-performance combinatorial optimization. Commun Phys 4, 266 (2021). https://doi.org/10.1038/s42005-021-00768-0"

Contents

Installation

The project is uploaded in gitLab. To install the application, clone the repository:

```
git clone https://gitlab.developers.cam.ac.uk/phy/data-intensive-science-mphil/projects/fw385.git
```

Navigate to the project directory and install the dependencies.

Dependencies

This project is written in Python (The project was developed using python=3.9.18). To run the project, **Python 3** or later need to be installed. Necessary packages used in the project are listes in environment.yml. They are:

- numpy - A fundamental package for scientific computing in Python.
- PyTorch - An open source machine learning library used for applications such as computer vision and natural language processing.
- Jupyter Notebook - An open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text.
- MLOOP - A machine learning optimization library for optimizing experimental and computational processes.

for generating offline documentation:

- doxygen - A tool for generating documentation from annotated code.

For full details on all packages, please refer to the `environment.yml` file.

Usage

All the modules, functions and classes for the CIM algorithm is saved in the `src` folder. The testing intances are the Gsets and MAX-CUT instances, which are saved in `G_sets` and `MC_Instances` folders respectively. The `Notebooks` folder contains the **introductions** to the project and the algorithm. They cover topics from solving an Ising model to hyperparameter optimisation.

To solve for a intance, run the `Solve_Ising.py` file in `src` folder a single command line with the path to the example MAX-CUt files in `MC_Instances` folder as an argument:

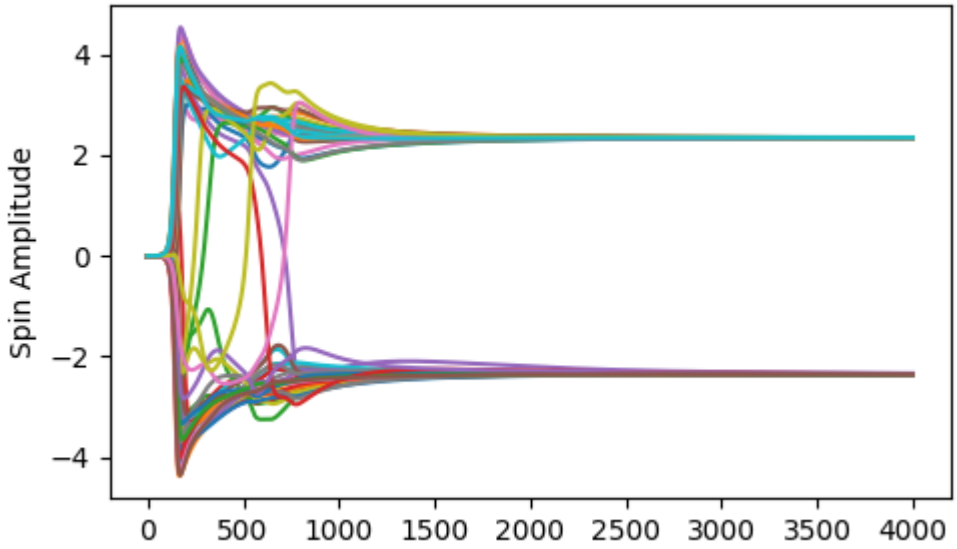
```
$ python src/Solve_Ising.py MC_Instances/MC50_N=50_1.npz
```

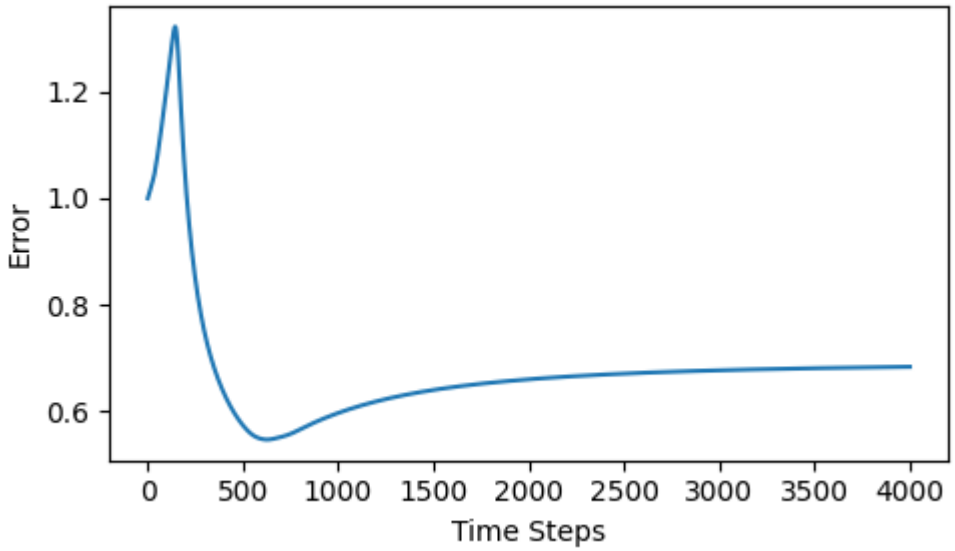
Make sure you are in the project directory when running the command. The solver `Solve_Ising.py` is inside the `src` folder, and the `MC50_N=50_1.npz` file should in the `MC_Instances` folder. If you want to solve for G sets, you need to convert the graph problem to a J coupling matrix. The detailed procedures for solving Gsets are shown in `Gset Evaluation` folder in `Notebooks`.

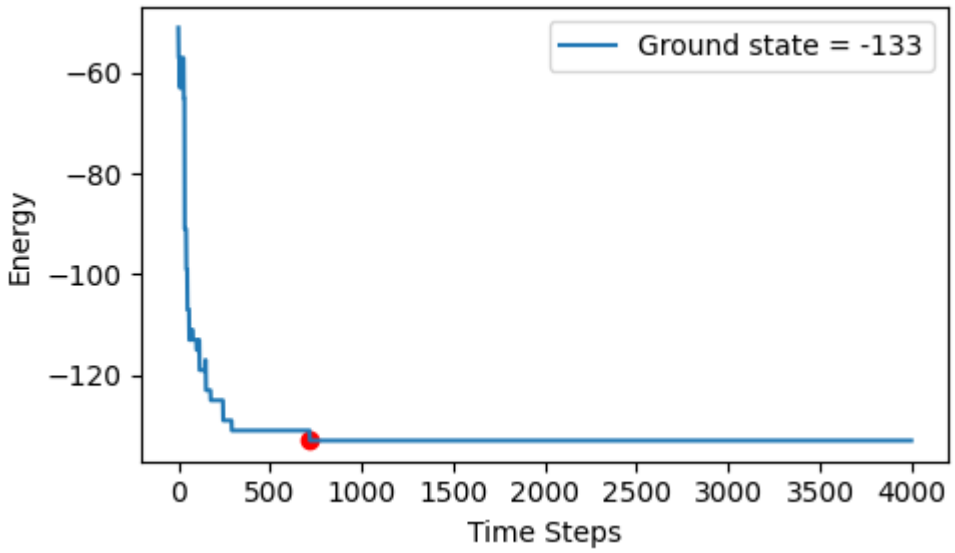
The simple usage of the code is:

```
results = CIM_AHC_GPU(T_time=20,
                      J=J,
                      batch_size=1,
                      time_step=0.01,
                      custom_fb_schedule=None,
                      custom_pump_schedule=None)
spin_config, x_trajectory, t, energy_plot_data, error_var_data, divg,
kappa = results
```

The output terminal will display the ground state found for Ising model, and the corresponding MAX-CUT energy, as well as the time for reaching the ground state. Also, the figures for the simulation will be generated.







Features

- **Insing Model Solving**: Solves Ising model effectively. Finds the optimal ground state and corresponding spin configurations.
- **File I/O**: Reads instances from the either `G_sets` or `MC_Instances` folders and writes solutions to the terminal.
- **Efficiency Evaluation**: Measures and displays the time taken for reaching the solution in the solving process.
- **CLI Interface**: Offers a simple command-line interface for ease of use.

Frameworks

- **Language**: Python 3.9.19
- **Containerisation**: Docker was used for reproducing the project. Key configurations include continuumio/miniconda3 as the base image for a Miniconda-based Python environment. It sets /usr/IsingSolver as the working directory, and copies all project files into the container. The environment is set up by executing conda env update using the `environment.yml` file, and the Docker shell is configured to use the sudoku-solver-env. It is runnable by using the single command:

```
docker run ising-solver
```

Credits

Author: Fanyi Wu

Generation Tools

- VSCode - Python IDE for development.
- Git - Version control system for managing code versions and collaboration.
- Docker - Containerization platform for ensuring consistent environments.
- Doxygen - Documentation generator tool used for writing software documentation from annotated source code.
- Copilot - Autofill codes and debug for the scripts.

ChatGPT (https://www.chat.openai.com/)

AI generation tool used to help with developing and embellishing the project.

Ising Model Contributions

During the project implementation involving the Ising model, various computational techniques and insights from the ChatGPT were utilized. Here are detailed descriptions of these interactions, how the insights were integrated, and the modifications made to adapt the suggestions to the project's specific requirements.

Model Setup and Configuration

Inquiry: Investigated the optimal setup for simulating the Ising model across different temperature regimes.

- **ChatGPT Output**: Recommendations on lattice configurations, boundary conditions, and temperature scaling.

- **Application**: Configured the simulation environment with suggested lattice structures and temperature scales, adapted to study phase transitions in ferromagnetic and antiferromagnetic systems.

Simulation Optimization

Inquiry: What are the efficient algorithms for simulating the Ising model dynamics?

- **ChatGPT Output**: Insights into Monte Carlo methods like the Metropolis-Hastings algorithm and Wolff algorithm for spin flip simulations.
- **Application**: Implemented these algorithms, optimizing them for increased computational efficiency and accuracy in capturing critical phenomena.

Gset to J Matrix Conversion

Inquiry: How to convert graph-based Gset data into a J matrix suitable for Ising model simulations?

- **ChatGPT Output**: Methodologies for transforming graph adjacency matrices into Ising model J matrices.
- **Application**: Developed a conversion algorithm that translates Gset graph data into J matrices, ensuring accurate representation of interaction strengths for Ising simulations.

Chaotic Amplitude Control

Inquiry: Techniques for controlling amplitude in chaotic systems within the context of statistical mechanics simulations.

- **ChatGPT Output**: Overview of control methods for chaotic systems and their application to physical simulations.
- **Application**: Implemented amplitude control techniques in simulation scenarios to stabilize system behavior under chaotic conditions, enhancing the predictability and reliability of simulation outcomes.

Ising Energy to Max-Cut Conversion

Inquiry: How to translate Ising model energy configurations into Max-Cut problems?

- **ChatGPT Output**: Detailed steps for mapping Ising model energies to Max-Cut problem formulations.
- **Application**: Applied the conversion technique to analyze Ising model simulations in terms of graph theory, facilitating a broader application of the results to optimization problems.

Using MLOOP

Inquiry: How to implement and utilize MLOOP for optimizing simulation parameters in complex systems?

- **ChatGPT Output**: Instructions and best practices for integrating MLOOP into scientific simulations.
- **Application**: Integrated MLOOP to automate the tuning of simulation parameters, significantly enhancing model performance and reducing manual tuning efforts.

Visualization and Analysis

Inquiry: Requested methods for visualizing magnetization and correlation functions over time.

- **ChatGPT Output**: Techniques for plotting state changes and correlation data.
- **Application**: Developed visualization tools that dynamically display the evolution of the system state, aiding in the analysis of critical behavior and phase transitions.

Theoretical Background and Literature Review

Inquiry: Sought a concise theoretical foundation on statistical mechanics related to the Ising model and relevant academic papers.

- **ChatGPT Output**: A summary of fundamental principles in statistical mechanics with references to seminal papers on the Ising model.
- **Application**: The theoretical overview informed the introduction and discussion sections of the project documentation, incorporating suggested readings to substantiate the methodology and results discussion.

Debugging and Code Refinement

Inquiry: Consulted on best practices for debugging complex simulation code and improving code readability.

- **ChatGPT Output**: Strategies for debugging scientific code and enhancing code structure for readability and maintainability.
- **Application**: Applied these strategies to refine the codebase, making it more modular and easier to troubleshoot, enhancing overall project maintainability.

Bibliography

[1] R. A. Quintero, L. F. Zuluaga, Qubo formulations of combinatorial optimization problems for quantum computing devices, in: Encyclopedia of Optimization, Springer, 2022, pp. 1–13.

[2] A. Shukla, M. Erementchouk, P. Mazumder, Scalable almost-linear dynamical ising machines, Natural Computing (2024) 1–13.

[3] S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi, Optimization by simulated annealing, science 220 (4598) (1983) 671–680.

[4] T. Kadowaki, H. Nishimori, Quantum annealing in the transverse ising model, Physical Review E 58 (5) (1998) 5355.

[5] W. Wang, J. Machta, H. G. Katzgraber, Comparing monte carlo methods for finding ground states of ising spin glasses: Population annealing, simulated annealing, and parallel tempering, Physical Review E 92 (1) (2015) 013303.

[6] P. L. McMahon, A. Marandi, Y. Haribara, R. Hamerly, C. Langrock, S. Tamate, T. Inagaki, H. Takesue, S. Utsunomiya, K. Aihara, et al., A fully programmable 100-spin coherent ising machine with all-to-all connections, Science 354 (6312) (2016) 614–617.

[7] R. L. Dobrushin, Existence of a phase transition in two-dimensional and three-dimensional ising models, Theory of Probability & Its Applications 10 (2) (1965) 193–213.

[8] W. Wolff, J. Zittartz, Reentrant behaviour in spin glasses, Zeitschrift für Physik B Condensed Matter 60 (2) (1985) 185–188.

[9] J. Machta, Daniel stein and charles newman: spin glasses and complexity (2013).

[10] T. Leleu, Y. Yamamoto, P. L. McMahon, K. Aihara, Destabilization of local minima in analog spin systems by correction of amplitude heterogeneity, Phys. Rev. Lett. 122 (2019) 040607. doi:10.1103/PhysRevLett.122.040607.
URL https://link.aps.org/doi/10.1103/PhysRevLett.122.040607

[11] W. Lenz, Beitrag zum verständnis der magnetischen erscheinungen in festen körpern, Z. Phys. 21 (1920) 613–615.

[12] V. Fock, Über den begriff der geschwindigkeit in der diracschen theorie des elektrons, Zeitschrift für Physik 55 (2) (1929) 127–140.

[13] J. V. Selinger, J. V. Selinger, Ising model for ferromagnetism, Introduction to the Theory of Soft Matter: From Ideal Gases to Liquid Crystals (2016) 7–24.

[14] S. Wald, Thermalisation and relaxation of quantum systems, Ph.D. thesis (09 2017). doi:10.13140/RG.2.2.25169.63842.

[15] K. F. Pál, The ground state energy of the edwards-anderson ising spin glass with a hybrid genetic algorithm, Physica A: Statistical Mechanics and its Applications 223 (3-4) (1996) 283–292.

[16] S. V. Isakov, I. N. Zintchenko, T. F. Rønnow, M. Troyer, Optimised simulated annealing for ising spin glasses, Computer Physics Communications 192 (2015) 265–271.

[17] A. Venugopalan, R. Karishmah, R. Pillai, D. Sreedev, V. Balachandran, P. Bhattacharjee, Study of quantum annealing and the type of related applications, in: 2021 IEEE Pune Section International Conference (PuneCon), IEEE, 2021, pp. 1–6.

[18] Z. Wang, A. Marandi, K. Wen, R. L. Byer, Y. Yamamoto, Coherent ising machine based on degenerate optical parametric oscillators, Physical Review A 88 (6) (2013) 063853.

[19] H. Takesue, T. Inagaki, K. Inaba, T. Honjo, Solving large-scale optimization problems with coherent ising machine, in: Conference on Lasers and Electro-Optics/Pacific Rim, Optica Publishing Group, 2017, p. s1561.

[20] M. Calvanese Strinati, D. Pierangeli, C. Conti, All-optical scalable spatial coherent ising machine, Physical Review Applied 16 (5) (2021) 054022.

[21] Y. Haribara, S. Utsunomiya, Y. Yamamoto, Computational principle and performance evaluation of coherent ising machine based on degenerate optical parametric oscillator network, Entropy 18 (4) (2016). doi:10.3390/e18040151.
URL https://www.mdpi.com/1099-4300/18/4/151

[22] U. Benlic, J.-K. Hao, Breakout local search for the max-cutproblem, Engineering Applications of Artificial Intelligence 26 (3) (2013) 1162–1173.

[23] Y. Wang, Z. Lü, F. Glover, J.-K. Hao, Probabilistic grasp-tabu search algorithms for the ubqp problem, Computers & Operations Research 40 (12) (2013) 3100–3107.

[24] R. Martí, A. Duarte, M. Laguna, Advanced scatter search for the max-cut problem, INFORMS Journal on Computing 21 (1) (2009) 26–38.

[25] S. Burer, R. D. Monteiro, Y. Zhang, Rank-two relaxation heuristics for max-cut and other binary quadratic programs, SIAM Journal on Optimization 12 (2) (2002) 503–521.

[26] J. H. Liu, H. T. Wang, Quantum fidelity, string order parameter, and topological quantum phase transition in a spin-1/2 dimerized and frustrated heisenberg chain, The European Physical Journal B 88 (2015) 1–8.

[27] P. Gao, Averaging principle for the higher order nonlinear schrödinger equation with a random fast oscillation, Journal of Statistical Physics 171 (5) (2018) 897–926.

[28] D. Xie, W. Gou, T. Xiao, B. Gadway, B. Yan, Topological characterizations of an extended su–schrieffer–heeger model, npj Quantum Information 5 (1) (2019) 55.

[29] Z.-Y. Sun, M. Li, H.-X. Wen, H.-G. Cheng, J. Xu, Y.-S. Chen, Multipartite quantum nonlocality and topological quantum phase transitions in a spin-1/2 two-leg kitaev ladder, The European Physical Journal B 94 (2021) 1–9.

[30] F. Böhm, T. Inagaki, K. Inaba, T. Honjo, K. Enbutsu, T. Umeki, R. Kasahara, H. Takesue, Understanding dynamics of coherent ising machines through simulation of large-scale 2d ising models, Nature communications 9 (1) (2018) 5020.

[31] T. Honjo, T. Sonobe, K. Inaba, T. Inagaki, T. Ikuta, Y. Yamada, T. Kazama, K. Enbutsu, T. Umeki, R. Kasahara, et al., 100,000-spin coherent ising machine, Science advances 7 (40) (2021) eabh0952.