A Dive into Cluster Monte-Carlo Algorithms

A Quantum-Inspired Approach

by Adrien Vandenbroucque

École Polytechnique Fédérale de Lausanne

Supervised by

Dr. Nicolas Macris[1]
Dr. Ewan Munro[2]

April 19, 2025

[1]Communication Theory Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland
[2]Chief Technology Officer, Entropica Labs Pte. Ltd., Singapore

Acknowledgements

I would like to first thank Dr. Nicolas Macris, my supervisor at EPFL for this Master's thesis, for helping in all kinds of ways with the more theoretical aspects of the project and for being present even though we were on opposite sides of the world. He pushed me to really dive into the technical details and fully understand the subject.
This research would not have been possible without Entropica Labs, the company who offered me this great project. In particular, I would like to thank my supervisor Dr. Ewan Munro for the constant support throughout the 6 months, and for all the great ideas I wouldn't have been able to come up with myself. A special thanks also to Dr. Tommaso Demarie, for very interesting discussions about many topics. Thank you to everyone at Entropica Labs for making this experience in Singapore so unique and fun.
Finally, I would like to thank my family and all the friends who supported me through the entire journey towards becoming an engineer.

Singapore, April 19, 2025

Abstract

Many discrete optimization problems can be mapped onto Ising spin-glass models, which gives a general framework to solve them. However, it is well-known that such systems can be hard to study, especially when frustration is present, with most results coming from numerical simulations. It has been shown that cluster Monte-Carlo algorithms are particularly efficient for simulating ferromagnetic Ising models, but results on more generic statistical mechanical models are not conclusive. Here we compare multiple of these algorithms and dive deeper into the analysis of the Houdayer algorithm, leading to an improved version which can yield better convergence to equilibrium. We also argue that this proposed method can be used to improve the sampling of optimal solutions when many of them exist, and this can be further tweaked to allow for better sampling of feasible solutions in constrained problems. We illustrate the various benefits of using our extended cluster move through experiments involving multiple graph topologies, in particular we look at a discrete optimization problem arising from industry.

Contents

List of Figures

List of Tables

1 Introduction

Discrete optimization problems arise across a large number of disciplines, many of which fit into the Ising model, or equivalently, into a quadratic unconstrained binary optimization (QUBO) problem. Such optimization formulations emerge in a number of fields such as machine learning [1], logistics [2], chemistry [3], and biology [4]. Thus, being able to solve these problems is of importance not only for research purposes, but it is also relevant in industrial applications.

With the increase in computing power during the last decades, the use of well-known Monte-Carlo methods like the Metropolis-Hastings algorithm [5], [6] to tackle such problems has proven to be effective, showing the ability to search efficiently even through large spaces of solutions. But even then, the exponential growth in computing power that took place until today is not expected to continue indefinitely [7], which calls for new techniques especially when one is interested to solve NP-hard combinatorial optimization problems.

With the advent of quantum computing, new optimization techniques have been introduced [8]–[11], some of which may give speedups over their classical counterparts in practice. For example, it has been proposed that quantum tunneling may give quantum computers the power to solve some non-convex optimization problems faster than classical computer can [12]. Nevertheless, it remains an open question how ubiquitous quantum speed-ups will turn out to be in real-world settings. At the same time, other algorithms referred to as quantum-inspired (or physics-inspired) have emerged in an attempt to simulate quantum effects in classical computations [13]. The Markov chain Monte-Carlo methods considered in this project mostly fall into this category, and they attempt to explore large solution spaces in novel ways.

In particular, a procedure introduced by Houdayer [14] has been used recently used to develop an algorithm using parallel tempering with isoenergetic cluster moves, which is regarded as one of the most efficient for finding low energy states of Ising spin-glass models [15]. Based on similar ideas with the Houdayer move, in this thesis a new Monte-Carlo move which extends it is developed to allow for even further exploration of the solution space when combined with heuristic algorithms like simulated annealing [16] or parallel tempering [17]. The advantage is that the proposed Monte-Carlo update can be used in combination with other algorithms, resulting for example in new heuristic schemes to improve the sampling of low-energy solutions in difficult optimization problems, or to generate feasible solutions in constrained problems.

In order to test the efficiency of the methods, benchmarking is a crucial tool that allows to easily compare and choose which scheme is most relevant [18]. Attributes such as running time, precision to which an optimal solution is reached, and number of optimal solutions attained can completely affect which methods are most adapted to the task. We benchmark algorithms like simulated annealing and parallel tempering and compare their behavior when the Houdayer cluster update or its extensions are used. The experiments performed also test how capable the new method for improved sampling is on canonical problems like Maximum Cut [19]. A scheduling-type problem that arises in a range of industrial settings is also studied in detail, employing the methods developed in this work. In particular we show through numerical simulations that many feasible solutions of this constrained problem can be found, permitting further optimization on this subset of solutions.

Part of the project is also dedicated to get a deeper understanding of the theoretical underpinnings for the convergence of the methods studied in the context of the Ising spin-glass model. This is related to the concept of mixing time [20], which has been studied for both local [21]–[23] and non-local Monte-Carlo updates [24], [25]. However for the Houdayer cluster move and its extended version, the reasons behind their efficiency remain yet unknown, and this is where some partial theoretical results can also lead to a more informed development. We study the eigenvalues of the transition matrices induced by the algorithms chosen since they partially determine how fast the stochastic processes reach their equilibrium distributions.

This project contributes in multiple ways to the improvement of existing state-of-the-art algorithms that are used in the simulation of Ising spin-glass models, as well as to the development of new schemes to improve the optimization of hard combinatorial problems. Chapter 2 gives a review of the fundamentals about sampling and optimization and how this relates to the Ising model. In chapter 3, the topic of Markov chain Monte-Carlo is introduced, showing various known algorithms to solve Ising systems and how they compare with each other. In chapter 4, we build on the knowledge of existing methods to design a new cluster Monte-Carlo move. In particular, we show how this novel scheme can be used in combination with some heuristics to improve the sampling of low-energy solutions. We study the performance of those in the context of an optimization problem arising from industry. Next, chapter 5 discusses the convergence of the algorithms studied in terms of mixing time in order to get a more theoretical understanding of their behavior. Finally, a summary is given in chapter 6 together with the general trends observed. Some possible future directions are highlighted, concluding the report.

1.1 About the Company

Entropica Labs is a Singapore-based company, spin-off of Singapore's Centre for Quantum Technologies (National University of Singapore), founded in 2018 by Tommaso Demmarie and

Ewan Munro. The company strives to create algorithms, software tools, methods and models to make quantum computers useful. The current focus of Entropica is quantum optimization and machine learning, supporting enterprise customers to understand and integrate quantum computing. In particular, solving hard optimization problems using both quantum and classical techniques is part of Entropica's expertise.

2 Preliminaries

2.1 The Ising Model

2.1.1 Definition and Link to Quadratic Unconstrained Binary Optimization

In this section, we present two models, the Ising and the Quadratic Unconstrained Binary Optimization (QUBO) models, used to represent a large number of optimization problems such as Maximum Cut, Number Partitioning, Graph Coloring and more [26]. We are interested in these in particular because only in this setting the techniques developed in this project can be applied. We define those models more formally below.

Definition 2.1.1 (Ising Model)**.** *Let $G = (V, E)$ be a graph. To each vertex $v \in V$ (also called "site"), we attach a "spin" σ_v which can take values in $\{-1, 1\}$. This defines a state space S which is made of the spin assignments $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{|V|}) \in \{-1, 1\}^{|V|}$, so that $S = \{-1, 1\}^{|V|}$. Moreover, one can define interactions between the spins through a function $\mathcal{H} : S \to \mathbb{R}$ called the Hamiltonian which represents the energy of a spin configuration. It takes the form $\mathcal{H}(\boldsymbol{\sigma}) = -\sum_{(v,w)\in E} J_{vw}\sigma_v\sigma_w - \sum_{v\in V} h_v\sigma_v$, where $J_{vw}, h_v \in \mathbb{R}$.*

The Ising model was initially presented as a mathematical model for ferromagnetism [27], where when neighboring spin values agree they have lower energy than when the values disagree, which happens when we have $J_{ij} > 0 \, \forall i, j$. Another simplification that is often done is to not consider the linear part of the Hamiltonian, which is supposed to represent an external field interacting with the system, i.e., $h_i = 0 \, \forall i$. In general, the *ferromagnetic model* refers to the model when the weights are such that $J_{ij} > 0 \, \forall i, j$ and $h_i = 0 \, \forall i$, while we use the term *spin-glass model* [28] when the weights are arbitrary, i.e., $J_{ij} \in \mathbb{R} \, \forall i, j$ and $h_i \in \mathbb{R} \, \forall i$. However here, we often restrict to the case $h_i = 0 \, \forall i$.

Such systems are not in isolation in the real-world, but rather in an environment which also interacts with the spins in the system. For example the state of a system depends on both the energy (given by $\mathcal{H}$) and the ambient temperature. How this precisely works is made formal by thermodynamics [29] and in particular, the third law of thermodynamics states that a system at zero temperature is in a *ground state*, which is defined as a state which has lowest energy. Thus we can define our goal to be to find the configurations of spins $\boldsymbol{\sigma}^*$ which leads to the lowest level of energy, i.e., we want to compute $\boldsymbol{\sigma}^* = \min_{\boldsymbol{\sigma}\in\{-1,1\}^n} \mathcal{H}(\boldsymbol{\sigma})$.

Example 2.1.1 (Ising Lattice)**.** *In an Ising lattice, spins are placed on a rectangular grid and connected to their nearest neighbors. An example is shown in figure 2.1 for a 2 by 3 system.*

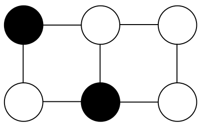


Figure 2.1 – A 2 by 3 lattice of spins, where spins with value -1 are depicted in black and those with value 1 in white.

Example 2.1.2 (Number Partitioning)**.** *Let $C = \{c_1, c_2, \ldots, c_n\}$ be a finite set of positive numbers. We wish to find a partition of C into two disjoint subsets A and $C \setminus A$ such that the sum of the elements in both sets is equal, i.e. $\sum_{a\in A} a = \sum_{b\in C\setminus A} b$.*
One can fit this into an Ising formulation by defining the following Hamiltonian:

$$\mathcal{H}(\boldsymbol{\sigma}) = \left(\sum_{i=1}^{n} c_i\sigma_i\right)^2 = \sum_{i,j=1}^{n} c_ic_j\sigma_i\sigma_j,$$

which is of the desired form $-\sum_{i,j} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i$ of an Ising model, with $J_{ij} = -c_ic_j$ and $h_i = 0 \quad \forall i, j \in \{1, 2, \ldots, n\}$.

Definition 2.1.2 (QUBO Model)**.** *Let $Q : \{0,1\}^n \to \mathbb{R}$ be a quadratic polynomial of n binary variables of the form $Q(\boldsymbol{x}) = \sum_{i,j=1}^{n} Q_{ij}x_ix_j$, where $Q_{ij} \in \mathbb{R}$.*

Again here, we are interested in finding the assignment $\boldsymbol{x}^*$ which minimizes the function Q, that is we want to compute $\boldsymbol{x}^* = \min_{\boldsymbol{x}\in\{0,1\}^n} Q(\boldsymbol{x})$.

Note that for any of the two models, adding a constant term does not alter the structure of the energy spectrum, and in particular, the lowest energy configurations remain unchanged.

A large number of problems fit into the structure of a quadratic polynomial [26], and in fact the Ising and QUBO models are equivalent in terms of the problems they can represent.

Lemma 2.1.1. *There is a bijection between QUBO problems and Ising models.*

Proof. Let $\mathcal{H}(\boldsymbol{\sigma}) = -\sum_{(v,w)\in E} J_{vw}\sigma_v\sigma_w - \sum_{v\in V} h_v\sigma_v$ be the Hamiltonian of some Ising model. In order to map it to a QUBO formulation, we need to change the range of the variables from

$\{-1,1\}$ to $\{0,1\}$, which is achieved by applying the mapping $\sigma \mapsto 2x-1$. Denoting by $Q(\boldsymbol{x})$ the resulting function, we get

$$\begin{aligned}
Q(\boldsymbol{x}) &= -\sum_{(v,w)\in E} J_{vw}(2x_v-1)(2x_w-1) - \sum_{v\in V} h_v(2x_v-1) \\
&= -\sum_{(v,w)\in E} (4J_{vw}x_vx_w - 2J_{vw}x_v - 2J_{vw}x_w + J_{vw}) - \sum_{v\in V}(h_v 2x_v - h_v) \\
&= -\sum_{(v,w)\in E} (4J_{vw}x_vx_w - 2J_{vw}x_vx_v - 2J_{vw}x_wx_w + J_{vw}) - \sum_{v\in V}(h_v 2x_vx_v - h_v) \\
&= \sum_{(v,w)\in E} (-4J_{vw}x_vx_w + 2J_{vw}x_vx_v + 2J_{vw}x_wx_w) + \sum_{v\in V} 2h_vx_vx_v \\
&\qquad - \left(\sum_{(v,w)\in E} J_{vw} + \sum_{v\in V} h_v\right) \\
&= \sum_{v,w=1}^{n} Q_{vw}x_vx_w + C,
\end{aligned}$$

where

$$Q_{vw} = \begin{cases} -4J_{vw}, & \text{if } (v,w)\in E \\ \sum_{(u,w)\in E} 2J_{uw} + \sum_{(v,u)} 2J_{vu} + 2h_v, & \text{if } v = w \\ 0, & \text{otherwise.} \end{cases}$$

and

$$C = -\left(\sum_{(v,w)\in E} J_{vw} + \sum_{v\in V} h_v\right).$$

Note that this is precisely in the QUBO format defined in definition 2.1.2.

Similarly, we can start from a QUBO problem and apply the mapping $x \mapsto \frac{\sigma+1}{2}$ to retrieve the corresponding Hamiltonian, from which we can conclude that the two models are equivalent. $\square$

For the remaining of the report, the type of problems that are solved are always expressed in the form of a quadratic polynomial, and depending on the range of our variables ($\{-1,1\}$ or

$\{0,1\}$), we choose either the Ising formulation or the QUBO one.

2.1.2 A Probabilistic View

Another way to look at the Ising model is to ask, what probability distribution can be assigned to a configuration $\boldsymbol{\sigma}$ with energy $\mathcal{H}(\boldsymbol{\sigma})$? In previous section, we mentioned that the temperature of the environment should be taken into account, which we do through the use of a parameter T representing the temperature at which the system is, or more commonly the inverse temperature $\beta = 1/(k_B T)$, where k_B is the Boltzmann constant.

To choose an adequate probability distribution, one should follow the principle of maximum entropy which states that if we do not have any constraint about a particular quantity, we should remain maximally flexible in this aspect when choosing a model while remaining consistent with quantities that are constrained. This is known as Jaynes' maximum entropy principle [30], and it yields a probability measure called the Boltzmann distribution for a certain spin configuration at inverse temperature β, defined as

$$\mu_\beta(\boldsymbol{\sigma}) = \frac{1}{Z_\beta} \exp\left(-\beta \mathcal{H}(\boldsymbol{\sigma})\right),$$

where $\beta > 0$ and $Z_\beta = \sum_{\boldsymbol{\sigma} \in S} \exp\left(-\beta \mathcal{H}(\boldsymbol{\sigma})\right)$ is the normalization constant.

One can then answer various questions about the system by computing average quantities such as the total energy of the system $\mathcal{H} = \sum_{\boldsymbol{\sigma} \in S} \mathcal{H}(\boldsymbol{\sigma}) \mu_\beta(\boldsymbol{\sigma}) = \mathbb{E}_{\mu_\beta}[\mathcal{H}(\boldsymbol{\sigma})]$ or the magnetization $m = (1/|V|) \sum_{\boldsymbol{\sigma} \in S} \mu_\beta(\boldsymbol{\sigma}) \left(\sum_{v \in V} \sigma_v\right) = (1/|V|)\mathbb{E}_{\mu_\beta}[\sum_{v \in V} \sigma_v]$. More generally, given a function $f(\boldsymbol{\sigma})$ of the spins, we are interested to compute

$$\mathbb{E}_{\mu_\beta}[f(\boldsymbol{\sigma})] = \sum_{\boldsymbol{\sigma} \in S} f(\boldsymbol{\sigma}) \mu_\beta(\boldsymbol{\sigma}),$$

the expected value of the function with respect to the probability density μ_β.

2.2 Sampling and Optimization

In this section, we review two central topics of this project, sampling and optimization, and how they relate. In particular we explain how sampling can be used to solve optimization problems.

2.2.1 Sampling

Let S be a sample space and $\mu : S \to [0, 1]$ a probability measure over this space. The goal of sampling is to produce elements from S according to μ. By that, we mean that if the number of samples produced is infinitely large, then the frequency of each element $\omega \in S$ in the samples would correspond to $\mu(\omega)$. That is, denoting our samples by $s_1, s_2, \dots$, we want $\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{s_i = \omega\} = \mu(\omega)$ for all $\omega \in S$. Similarly, we can formulate the problem as the following question: how can we pick an element $\omega \in S$ so that $\mathbb{P}(\omega) = \mu(\omega)$?

Example 2.2.1 (Fair Die)**.** *Suppose we are throwing a fair six-sided die. Then we have* $S = \{1, 2, \dots, 6\}$ *and* $\mu(\omega) = \frac{1}{6}$ *for* $\omega \in S$.

There exists various sampling techniques including uniform sampling, rejection sampling, importance sampling, each with their own advantages and disadvantages [31]. One common issue is that for some of them, one must know $\mu(i)$ for all $i \in S$, which may be non-trivial to compute. We present in chapter 3 methods to alleviate some of these issues.

Example 2.2.2 (Ising Model)**.** *Recall from section 2.1.2 that for the Ising model, we are interested in computing various average quantities of the form*

$$\mathbb{E}_{\mu_\beta}[f(\boldsymbol{\sigma})] = \sum_{\boldsymbol{\sigma}\in S} f(\boldsymbol{\sigma})\mu_\beta(\boldsymbol{\sigma}),$$

where $f : S \to \mathbb{R}$ *is some function of the spins. One way to do so is by sampling independently a certain number of spin configurations* $\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}, \dots, \boldsymbol{\sigma}^{(M)}$ *and to compute* $\frac{1}{M}\sum_{i=1}^{M} f(\boldsymbol{\sigma}^{(i)})$ *instead.*

By the law of large numbers we have that

$$\frac{1}{M}\sum_{i=1}^{M} f(\boldsymbol{\sigma}^{(i)}) \overset{M\to\infty}{\longrightarrow} \mathbb{E}_{\mu_\beta}[f(\boldsymbol{\sigma})] \text{ almost surely,}$$

so we can hope to find the values of interests by sampling repeatedly. Note however, the distribution for the Ising model is equal to $\mu_\beta(\boldsymbol{\sigma}) = \frac{\exp(-\beta\mathcal{H}(\boldsymbol{\sigma}))}{Z_\beta}$*, and the normalization constant* $Z_\beta = \sum_{\boldsymbol{\sigma}\in S} \exp\left(-\beta\mathcal{H}(\boldsymbol{\sigma})\right)$ *can become impossible to compute if the number of spins is large since the state space has size* $|S| = 2^n$*. In this case, the methods presented in the next chapter describe how to perform the sampling without the need to evaluate* Z_β*.*

2.2.2 Optimization of a Function Using Sampling

Let S be a finite set of elements and $f : S \to \mathbb{R}$ a function to be minimized. To find the minima of the function, one can solve it by sampling from the distribution

$$\pi(i) = \frac{\mathbb{1}\{i \text{ is a global minimum of } f\}}{Z}, \tag{2.1}$$

where $Z = \sum_{i \in S} \mathbb{1}\{i \text{ is a global minimum of } f\}$ is a normalization constant, which is equal to the number of global minima of f. Essentially, π is the uniform distribution over the global minima of f, so that if the sampling is performed correctly, it is ensured we will find one of the global minima at the end of the process. However, this is simply too difficult under this form, first because we would need to be able to tell whether some i is a global minimum to infer $\mathbb{1}\{i \text{ is a global minimum of } f\}$. This could be possible to determine in a neighborhood around i, but not globally so instead we sample from the Boltzmann distribution

$$\pi_\beta(i) = \frac{e^{-\beta f(i)}}{Z_\beta},$$

where $\beta > 0$ is a parameter (called the inverse temperature) and $Z_\beta = \sum_{i \in S} e^{-\beta f(i)}$ is the normalization constant. With such a distribution, one can observe that is easy to evaluate the different probabilities as they only depend on the values of the function f. We explain briefly why this distribution is of particular importance in this context by looking at its behavior as β goes from 0 to $+\infty$.

When $\beta \to 0$, we essentially get

$$\lim_{\beta \to 0} \frac{e^{-\beta f(i)}}{\sum_{j \in S} e^{-\beta f(j)}} = \frac{e^{-0 \cdot f(i)}}{\sum_{j \in S} e^{-0 \cdot f(j)}} = \frac{1}{\sum_{j \in S} 1} = \frac{1}{|S|},$$

the uniform distribution over the entire space. So at high temperature (low β), there is as much randomness as one can get. For other values of β, it is easier to rewrite the Boltzmann distribution as

$$\pi_\beta(i) = \frac{e^{-\beta f(i)}}{\sum_{j \in S} e^{-\beta f(j)}} = \frac{e^{-\beta f(i)}}{e^{-\beta f(i) +} \sum_{j \in S : j \neq i} e^{-\beta f(j)}} = \frac{1}{1 + \sum_{j \in S : j \neq i} e^{-\beta (f(j) - f(i))}}.$$

We see in particular that when $\beta \to +\infty$, we have to check all the terms in the sum in the denominator individually. There are two cases to consider:

1. If one of the terms is such that $f(j) - f(i) < 0$ (so i is not a global minimum of f), we get $\lim_{\beta \to +\infty} \pi_\beta(i) = \frac{1}{1+\infty} = 0$.
2. If all terms are such that $f(j) - f(i) > 0$ (that is i is the global minimum of f), then they all vanish to zero, giving $\lim_{\beta \to +\infty} \pi_\beta(i) = \frac{1}{1+0} = 1$. However, in the case where there are multiple global minima, some of the terms are such that $f(j) - f(i) = 0$ which give them a weight of $e^{-\beta \cdot 0} = 1$. So overall, if all terms are such that $f(j) - f(i) \geq 0$ (so i is a global minimum of f), we have $\lim_{\beta \to +\infty} \pi_\beta(i) = \frac{1}{1+\sum_{j \neq i : f(j) = f(i)} 1} = \frac{1}{\sum_{j \in S} \mathbb{1}\{j \text{ is a global minimum of } f\}}$.

We can combine the two cases to yield

$$\lim_{\beta \to +\infty} \pi_\beta(i) = \frac{\mathbb{1}\{i \text{ is a global minimum of } f\}}{\sum_{j \in S} \mathbb{1}\{j \text{ is a global minimum of } f\}}$$

In fact, we can observe that as β increases, the distribution concentrates around the global minima, eventually reaching the distribution π from equation 2.1. This is depicted in figure 2.2 in the case of a simple Ising model using the Hamiltonian as function to minimize, which we recall from section 2.1.1 what we aim to achieve.

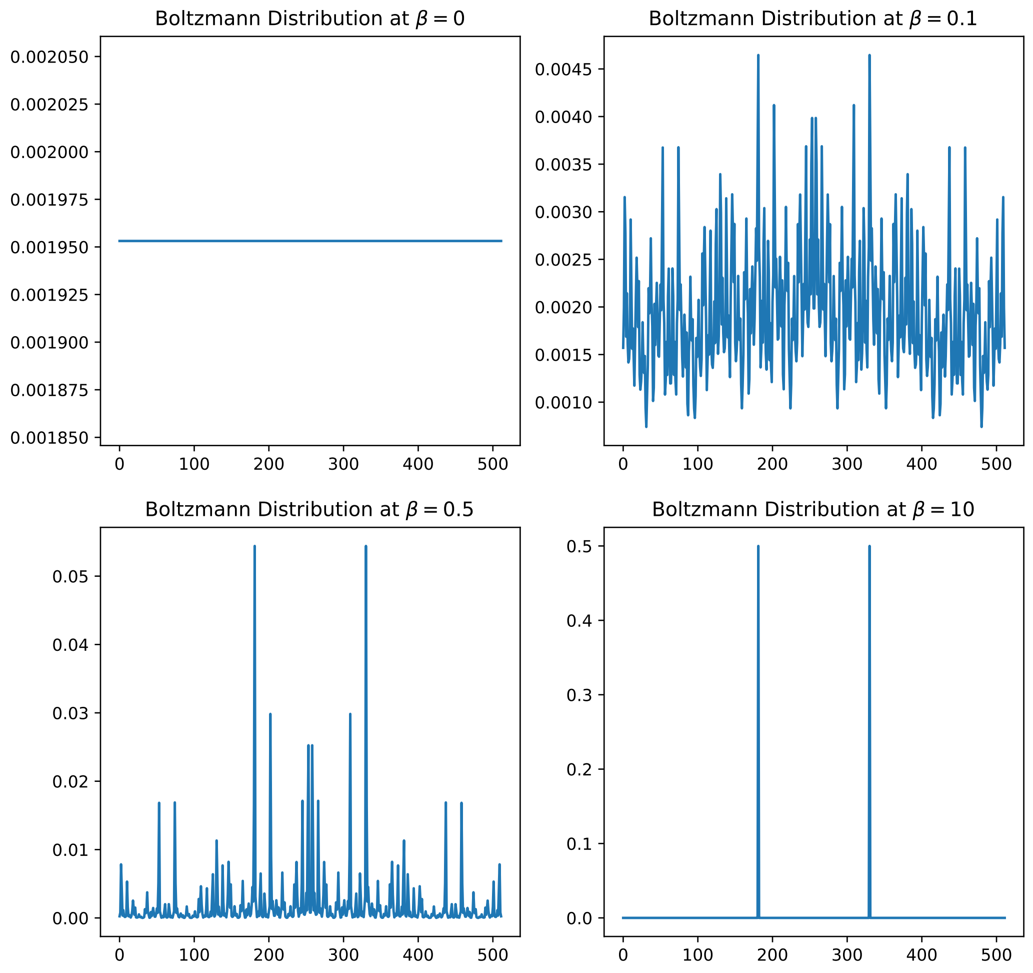


Figure 2.2 – Evolution of the Boltzmann distribution of an Ising model with Hamiltonian $\mathcal{H}$ consisting of a 3 by 3 lattice of spins with bonds being normally distributed around 0 with variance 1. As β increases, the distribution concentrates on the global minima of $\mathcal{H}$.

3 Markov Chain Monte-Carlo Methods

This chapter focuses on Monte-Carlo methods, which are numerical methods used to solve various problems which have an underlying probabilistic interpretation. In the case of an Ising model, we highlighted some of the interests as being the evaluation of various expected values.

These methods rely on sampling repeatedly to arrive at a satisfactory answer. Even though a plethora of such methods exist, we focus in this project on Markov Chain Monte-Carlo (MCMC) methods and dive more deeply into particular algorithms called "cluster algorithms".

3.1 Markov Chain Monte-Carlo Methods

As explained in previous chapter, sampling can be hard for multifarious reasons. MCMC methods give us a way to sample from a distribution by constructing a Markov chains which has as limiting distribution the desired distribution. For a full treatment of Markov chains and main results in the field, we invite the reader to refer to [20].

Let us first restate the ergodic theorem, which is of central importance in the theory of Markov chains.

Theorem 3.1.1 (Ergodic Theorem [20]). *Let $(X_n, n \geq 0)$ be an ergodic (aperiodic and irreducible) finite Markov chain with state space S and transition matrix P (i.e., $P_{ij} = \mathbb{P}(X_n = j | X_{n-1} = i)$). Then it admits a unique limiting and stationary distribution π, i.e., $\forall \pi^{(0)}, \lim_{n\to\infty} \pi^{(0)} P^n = \pi$ and $\pi = \pi P$.*

Moreover, given a function $f : S \to \mathbb{R}$ and starting from any distribution $\pi^{(0)}$, then

$$\frac{1}{M}\sum_{i=1}^{M} f(X_i) \overset{M\to+\infty}{\longrightarrow} \mathbb{E}_\pi[f] \text{ almost surely,}$$

,

where $\mathbb{E}_\pi[f] = \sum_{i \in S} f(i)\mu(i)$ *is the expected value of* f *under* π*.*

The ergodic theorem stated in this form informs of two things. First, ergodic Markov chains always have a limiting distribution which also happens to be the equilibrium distribution. Second, performing a random walk to get $X_1, X_2, ...$ is such that when averaging the function values $f(X_1), f(X_2), ...$ obtained, the result corresponds approximately to the expected value under the stationary distribution. The more sampling is done, the more precise the approximation is, which is similar to the Law of Large Numbers but for Markov chains. Note that this is exactly what we desire to achieve in the context of the Ising model (see example 2.2.2).

We can now ask the question, given a distribution π from which samples are needed, how to construct an ergodic Markov chain which has π as its limiting and stationary distribution?

3.1.1 Metropolis-Hastings Algorithm

Let π be some distribution we want to sample from, defined on a finite set S. The Metropolis-Hastings algorithm [6] is a procedure to construct a Markov chain on S which has π as limiting distribution. It mainly uses the fact that finding a solution to the equation $\pi = \pi P$ is greatly simplified by imposing detailed balance on the system, i.e., $\pi_i P_{ij} = \pi_j P_{ji}$ [32]. In fact, a chain represented by transition matrix P converges to the desired distribution if it is both ergodic and satisfies detailed balance.

The algorithm is as follows:

1. Select an arbitrary ergodic chain on S, represented by the transition matrix ψ. This chain proposes moves to explore S.
2. Design acceptance probabilities $a_{ij} = \mathbb{P}(\text{transition } i \to j \text{ is accepted}) = \min(1, \frac{\pi_j \psi_{ji}}{\pi_i \psi_{ij}})$.
3. Construct matrix P as: $P_{ij} = \begin{cases} \psi_{ij} a_{ij}, & \text{if } i \neq j \\ 1 - \sum_{k \neq i} \psi_{ik} a_{ik}, & \text{otherwise.} \end{cases}$

The resulting transition matrix P represents a chain which has the desired limiting distribution π. By performing a random walk on this chain, we are guaranteed by the ergodic theorem to get a sample distributed according to π if we do an infinite number of steps. Note also that when the chain ψ is symmetric, the acceptance probabilities simplify to $a_{ij} = \min(1, \frac{\pi_j}{\pi_i})$.

Alternatively, we can describe the Metropolis procedure in a more "online" fashion as follows:

1. Start from some state $i \in S$.
2. Choose a move $j \in S$ according to the proposal move given by ψ.

3. With probability a_{ij} (as defined previously) accept that move and go to state j, otherwise stay in state i.

4. Go back to 2. and repeat.

Again just as before, if we repeat this process long enough, we can use the state we are in as a sample since we expect it to be distributed close to distribution π.

Example 3.1.1 (Heat-Bath Dynamics)**.** *The simplest dynamics for sampling configurations of an Ising system with Hamiltonian $\mathcal{H}$ at temperature β can be done as follows using the Metropolis algorithm.*

1. *Start at some random spin assignment $\boldsymbol{\sigma} \in S$.*

2. *Choose a vertex $v \in V$ uniformly at random and consider now the assignment $\boldsymbol{\sigma'}$, where $\sigma'_v = -\sigma_v$, i.e., we flip the value of spin at vertex v.*

   That is, we use as base chain ψ such that $\psi_{\boldsymbol{\sigma},\boldsymbol{\sigma'}} = \begin{cases} 1/|V|, & \textit{if } \boldsymbol{\sigma} \textit{ and } \boldsymbol{\sigma'} \textit{ differ by one spin} \\ 0, & \textit{otherwise.} \end{cases}$

   for all $\boldsymbol{\sigma}, \boldsymbol{\sigma'} \in S$.

3. *The Metropolis algorithm is used to get the acceptance probabilities given that π_β is the Boltzmann distribution, yielding* $a_{\boldsymbol{\sigma},\boldsymbol{\sigma'}} = \min(1, \exp(-\beta\,(\mathcal{H}(\boldsymbol{\sigma'}) - \mathcal{H}(\boldsymbol{\sigma}))))$

4. *Go back to 2. and repeat.*

It is often called the heat-bath dynamics or also single-flip dynamics.

For simple Ising models such as the ferromagnetic one, using the Heat-Bath dynamics described in example 3.1.1 works well at high enough temperature. However, once the system starts getting more complex (with both positive and negative weights for the interactions) or the temperature is very low, this renders the single-flip strategy inefficient. In particular, when searching in the vicinity of a state, differences in energy may be large, resulting in the acceptance probabilities to vanish exponentially fast. So for such systems, especially at low temperatures, techniques like Heat-Bath fail to find the ground states and get stuck in regions where the energy is locally minimal.

In general, this is one of the main hurdles when performing optimization. There is a state space we are searching in order to find a state with lowest value, but the function to optimize might be complex and contain multiple local minima, making it difficult for local search methods like Heat-Bath to find the best solution. In general, there is a trade-off of exploration versus exploitation, that is how much are we willing to explore at the cost of being precise enough, versus how much we want to locally exploit at the cost of missing the best value. This idea

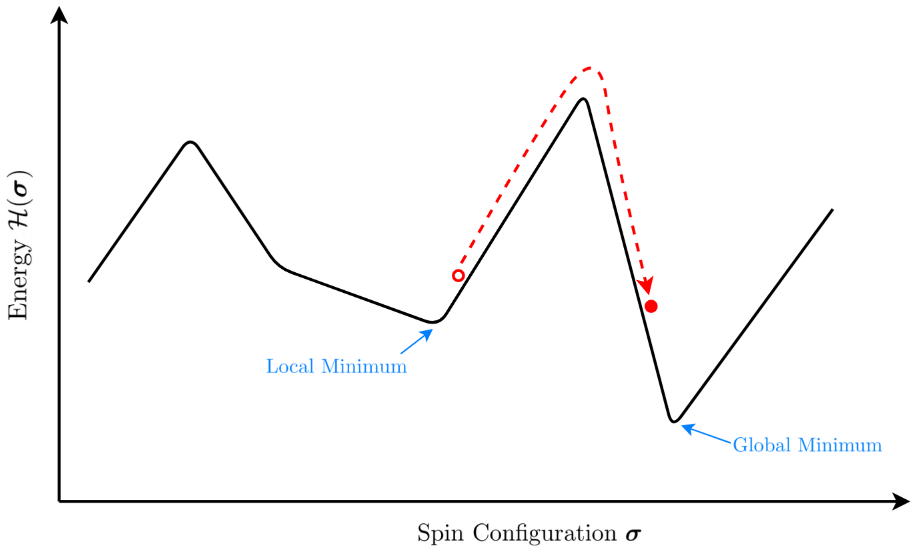

Energy $\mathcal{H}(\boldsymbol{\sigma})$
Local Minimum
Global Minimum
Spin Configuration $\boldsymbol{\sigma}$

Figure 3.1 – A representation of the energy landscape of some Ising system. One notices a local minimum and the global one, separated by an energy barrier. Algorithms like Heat-Bath, when close to a local minimum, will tend to get stuck there (red point on the left). So one would like to develop techniques to overcome such barriers (red arrow) to improve the exploration and get to the global optimum.

is pictured in figure 3.1, showing that one would like to overcome energy barriers whenever possible to reach the global minimum.

In the next sections, we present techniques to improve the minimization procedure for the Ising model, and introduce concepts like parallel tempering and cluster algorithms.

3.2 Parallel Tempering

We argued in previous section that one of the downsides of the single-flip strategy is that close to a local minimum, the Metropolis acceptance probability can become exponentially small due to low temperature or high difference in energy in the neighborhood, resulting in "getting stuck" at a particular configuration. One way to prevent this from happening and to escape from such minima is to use the Exchange Monte-Carlo method or Parallel Tempering [17].

In this method, multiple spin configurations called replicas are simulated simultaneously at different temperatures $\beta_1 < \beta_2 < \cdots < \beta_k$. Each replica is simulated independently through some usual algorithm like the heat-bath one, but after each iteration, replicas between neighboring temperatures can be exchanged, allowing them to move up and down the temperature space. More precisely, the probability that two replicas $\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}$ at temperatures β_n and β_m

are exchanged is given by

$$\mathbb{P}\left(\boldsymbol{\sigma}^{(1)} \leftrightarrow \boldsymbol{\sigma}^{(2)}\right) = \min\left(1, \exp\left(-(\beta_m - \beta_n)\left(\mathcal{H}(\boldsymbol{\sigma}^{(2)}) - \mathcal{H}(\boldsymbol{\sigma}^{(1)})\right)\right)\right) \tag{3.1}$$

when employing the Metropolis method. For convenience, the replica-exchange is usually restricted to neighboring temperatures (so $m = i + 1$ and $n = i$) because the corresponding probability of exchange decreases exponentially fast with respect to the difference in temperature $\beta_m - \beta_n$. By employing this technique, the convergence to the stationary distribution is accelerated because a configuration now can move up in temperature space, where Metropolis moves have a higher chance of being accepted, allowing to escape local minima as shown in figure 3.1.

In practice, there can be more than just one replica simulated at each temperature. In that case, after each iteration of the Metropolis algorithm for each replica, those between neighboring temperatures are first randomly paired, and then exchanged with probability as described in equation 3.1. As presented here, parallel tempering, although improving the exploration of the state space, still remains a local update algorithm when using heat-bath to update the replicas independently. As we will see later on, parallel tempering is often combined with other types of spin updates, like cluster moves instead of the single-flip which allow for further accelerated dynamics.

3.3 Cluster Monte-Carlo Algorithms

Earlier, we introduced how to use Metropolis-Hastings algorithm in the context of the Ising model, yielding the heat-bath (or single-flip) dynamics. However, it is not able to solve complex instances of Ising spin-glasses, in part due to the fact that it is a local update algorithm. Here, we hope to explore the state space in a better way by allowing multiple spins to change at the same time, effectively "teleporting" to somewhere different which is hopefully closer to one of the global minima. There are then two ways to design new algorithms which use more global updates. Either by staying in the framework of the Metropolis algorithm, in which case we aim to propose global moves through the chain ψ (see step 1 of Metropolis in section 3.1.1), or we can come up with different algorithms. We first present two similar algorithms which are different from the usual heat-bath dynamics, and then go in more depth into the Houdayer cluster move.

3.3.1 Swendsen-Wang Algorithm

The Swendsen-Wang algorithm [33] is a cluster Monte-Carlo algorithm as it updates a collection of spins at every iteration. It works as follows:

1. Start with a random state $\boldsymbol{\sigma} \in S$.

2. For each pair of spins that are connected in the graph, $(v, w) \in E$, create a bond between the two spins with probability $1 - e^{-2\beta J_{vw}}$ only if the spins are the same, i.e., $\sigma_v = \sigma_w$.

3. The bonds created in previous step also created clusters, where a cluster is a set of spins such that for any two spins in that set, there exist a connected path of bonds joining them.

   After identifying all clusters, one assigns for each cluster the value 1 or -1 uniformly at random to all spins inside that cluster. Equivalently, one flips each cluster with probability 1/2.

4. Go back to step 2.

It was mainly designed to simulate ferromagnetic models, and thus has difficulties when dealing with models that are frustrated, like spin-glass models [34].

3.3.2 Wolff Algorithm

Wolff algorithm [33] is another cluster Monte-Carlo algorithm, which is a variation of the previous one. Similar to the previous algorithm, it is although slightly different in the sense that it flips only one cluster with probability 1, while the Swendsen-Wang method flips multiple clusters, each with probability 1/2. Its description is given below.

1. Start with a random state $\boldsymbol{\sigma} \in S$.

2. Pick a vertex $v \in V$ uniformly at random. From there, grow a cluster by considering all neighbors w of v according to the graph G and include them in the cluster with probability $1 - e^{-2\beta J_{vw}}$ only if $\sigma_v = \sigma_w$. Then repeat this process recursively for each w that was added to the cluster. It will eventually terminate, and we are left with a cluster.

3. Flip the value of all spins inside the cluster with probability 1.

4. Go back to step 2.

Note that this is a "rejection-free" algorithm, in the sense that the cluster chosen in step 2 is always flipped. Wolff's algorithm gives good result for the ferromagnetic model, but fails for more general instances just like the Swendsen-Wang algorithm. Part of the reason why they become inefficient is that at low temperatures, the cluster created encompasses almost all spins, making the move a trivial one [35].

3.4 Houdayer Algorithm

3.4.1 Description

The Houdayer procedure [14] is slightly different from other cluster Monte-Carlo algorithms as it requires to start with a pair of spin configurations at the same temperature instead of just a single one. As such, in this setup, a state is a pair of independent spin configurations $(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}) \in S \times S$ and we want to sample from this new state space according to the joint distribution which is

$$\begin{aligned}\mu_\beta\left((\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})\right) &= \mu_\beta\left(\boldsymbol{\sigma}^{(1)}\right)\mu_\beta\left(\boldsymbol{\sigma}^{(2)}\right)\\ &= \frac{1}{Z}\exp\left(-\beta\left(\mathcal{H}(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}(\boldsymbol{\sigma}^{(2)})\right)\right),\end{aligned}$$

where $\beta > 0$ is the inverse temperature and $Z = \sum_{\tau^{(1)},\tau^{(2)} \in S\times S} \exp\left(-\beta\left(\mathcal{H}(\boldsymbol{\tau}^{(1)}) + \mathcal{H}(\boldsymbol{\tau}^{(2)})\right)\right)$ is the normalization constant.

The move works as follows:

1. Given a pair of independent spin configurations $(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})$, compute the local overlap at every site. For site i, it is defined as $q_i = \sigma_i^{(1)}\sigma_i^{(2)}$. Note that this defines two domains, -1 and 1, and one can alternatively write these as $q_i = \begin{cases} 1 & \text{if } \sigma_i^{(1)} = \sigma_i^{(2)}, \\ -1 & \text{if } \sigma_i^{(1)} \neq \sigma_i^{(2)}. \end{cases}$
2. The overlap computed in previous step defines clusters, which are the connected parts having the same overlap value (we say that two sites are connected if there is an edge between them, i.e., $J_{ij} \neq 0$). Select at random a site for which $q_i = -1$ and flip the cluster to which it belongs in both configurations.

In words, to perform a Houdayer move, one computes the site overlap and then makes a random choice to select one of the clusters of spins for which their value disagree in the pair of configurations. Then the corresponding spin values are swapped between the two configurations. A depiction of the move is displayed in figure 3.2. For the remainder of this report, we refer to the clusters induced by the local overlap as *Houdayer clusters*.

3.4.2 Properties

An interesting feature of the Houdayer move is that the sum of energies of the configurations in the pair stays constant.

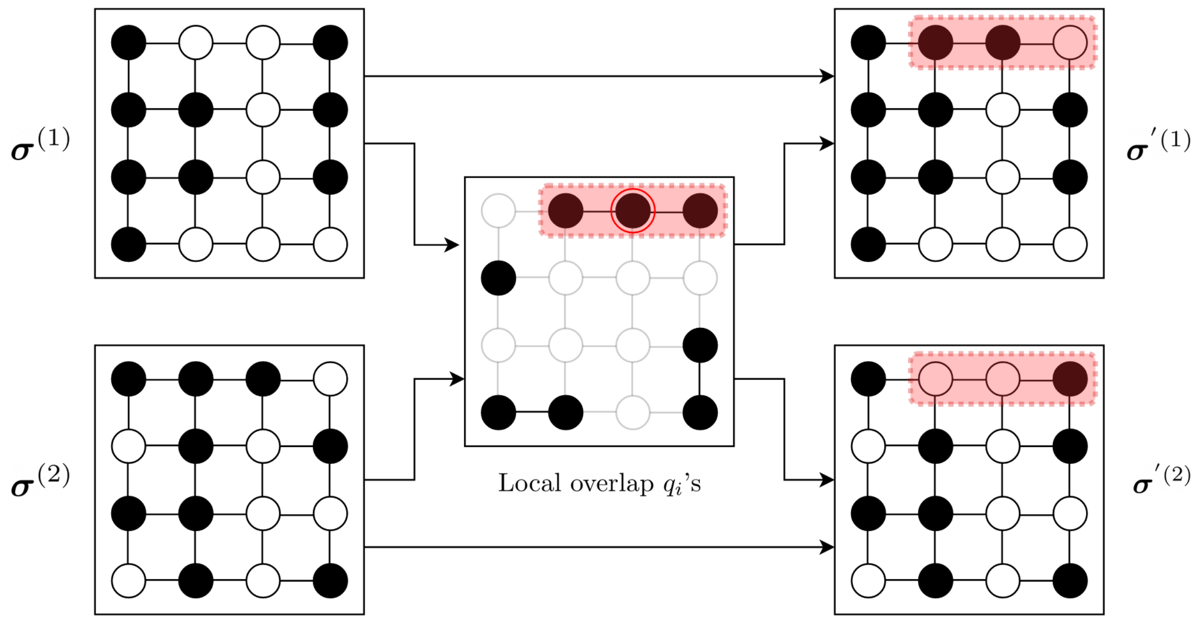

$\boldsymbol{\sigma}^{(1)}$
$\boldsymbol{\sigma}^{(2)}$
Local overlap q_i's
$\boldsymbol{\sigma}'^{(1)}$
$\boldsymbol{\sigma}'^{(2)}$

Figure 3.2 – A visual representation of the Houdayer move. It starts with a pair of independent spin configurations $(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})$ and first compute their local overlap, revealing where spins differ between the two configurations. One of the sites with overlap value -1 is then chosen uniformly at random (circled in red) and the corresponding cluster is grown from there (highlighted in red). After that, the spins in the cluster are swapped between the two replicas, or equivalently they are flipped, yielding the new pair $(\boldsymbol{\sigma}'^{(1)}, \boldsymbol{\sigma}'^{(2)})$.

Lemma 3.4.1. *Let $\mathcal{H}(\boldsymbol{\sigma}) = -\sum_{(v,w)\in E} J_{vw}\sigma_v\sigma_w - \sum_{v\in V} h_v\sigma_v$ be the Hamiltonian of an Ising model represented by the graph $G = (V, E)$, where the state space is defined as $S = \{-1, 1\}^{|V|}$. Let $(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})$ be a pair of spin configurations and suppose that a Houdayer move results in the new pair $(\boldsymbol{\sigma}'^{(1)}, \boldsymbol{\sigma}'^{(2)})$. Then we have $\mathcal{H}(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}(\boldsymbol{\sigma}^{(2)}) = \mathcal{H}(\boldsymbol{\sigma}'^{(1)}) + \mathcal{H}(\boldsymbol{\sigma}'^{(2)})$.*

Proof. We first split the Hamiltonian into linear and quadratic terms as

$$\mathcal{H}_l(\boldsymbol{\sigma}) = -\sum_{v\in V} h_v\sigma_v$$

and

$$\mathcal{H}_q(\boldsymbol{\sigma}) = -\sum_{(v,w)\in E} J_{vw}\sigma_v\sigma_w,$$

so that $\mathcal{H}(\boldsymbol{\sigma}) = \mathcal{H}_q(\boldsymbol{\sigma}) + \mathcal{H}_l(\boldsymbol{\sigma})$.

Since the move done is a Houdayer move, we suppose that after computing the site overlap and selecting a site with overlap value -1, the corresponding chosen cluster to be flipped is the set $\mathcal{C} = \{i_1, i_2, \dots, i_{|\mathcal{C}|}\}$ (where each element designates a spin). Writing $(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})$ and $(\boldsymbol{\sigma}'^{(1)}, \boldsymbol{\sigma}'^{(2)})$ as the starting and resulting pairs, we prove the lemma by showing that for both the quadratic and linear part of the Hamiltonian, the property that the sum of the function values is preserved holds. We mainly use the fact that the Houdayer move swaps the spin values in $\mathcal{C}$ between the two configurations, i.e., $\sigma_v'^{(1)} = \sigma_v^{(2)} \quad \forall v \in \mathcal{C}$ and $\sigma_v'^{(2)} = \sigma_v^{(1)} \quad \forall v \in \mathcal{C}$ while other spin values remain unchanged.

1. Linear part

   We want to show that $\mathcal{H}_l(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}_l(\boldsymbol{\sigma}^{(2)}) = \mathcal{H}_l(\boldsymbol{\sigma}'^{(1)}) + \mathcal{H}_l(\boldsymbol{\sigma}'^{(2)})$.

   We have

$$\begin{aligned}\mathcal{H}_l(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}_l(\boldsymbol{\sigma}^{(2)}) - \mathcal{H}_l(\boldsymbol{\sigma}'^{(1)}) - \mathcal{H}_l(\boldsymbol{\sigma}'^{(2)}) &= -\sum_{v\in V} h_v(\sigma_v^{(1)} + \sigma_v^{(2)} - \sigma_v'^{(1)} - \sigma_v'^{(2)}) \\ &= -\sum_{v\in \mathcal{C}} h_v(\sigma_v^{(1)} + \sigma_v^{(2)} - \sigma_v'^{(1)} - \sigma_v'^{(2)}) \\ &= -\sum_{v\in \mathcal{C}} h_v(\sigma_v^{(1)} + \sigma_v^{(2)} - \sigma_v^{(2)} - \sigma_v^{(1)}) \\ &= 0,\end{aligned}$$

   where in the second equality we used the fact that for all spins not in the cluster, their value is unchanged resulting in the corresponding term in the sum to be zero.

2. Quadratic part

   In the same way, we want to show $\mathcal{H}_q(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}_q(\boldsymbol{\sigma}^{(2)}) = \mathcal{H}_q(\boldsymbol{\sigma}'^{(1)}) + \mathcal{H}_q(\boldsymbol{\sigma}'^{(2)})$.

   We have

$$\begin{aligned}&\mathcal{H}_q(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}_q(\boldsymbol{\sigma}^{(2)}) - \mathcal{H}_q(\boldsymbol{\sigma}'^{(1)}) - \mathcal{H}_q(\boldsymbol{\sigma}'^{(2)}) \\ &\qquad = -\sum_{(v,w)\in E} J_{vw}\left(\sigma_v^{(1)}\sigma_w^{(1)} + \sigma_v^{(2)}\sigma_w^{(2)} - \sigma_v'^{(1)}\sigma_w'^{(1)} - \sigma_v'^{(2)}\sigma_w'^{(2)}\right) \\ &\qquad = -\sum_{\substack{(v,w)\in E:\\ v\in\mathcal{C} \text{ or } w\in\mathcal{C}}} J_{vw}\left(\sigma_v^{(1)}\sigma_w^{(1)} + \sigma_v^{(2)}\sigma_w^{(2)} - \sigma_v'^{(1)}\sigma_w'^{(1)} - \sigma_v'^{(2)}\sigma_w'^{(2)}\right),\end{aligned}$$

   where again we used that when we consider an interaction involving spins which are unchanged, the term in the sum is zero. Let us now look at the other terms.

   In the case where $v, w \in \mathcal{C}$ so that both spin values are swapped, then we get

$$\begin{aligned}
J_{vw}(\sigma_v^{(1)}\sigma_w^{(1)} + \sigma_v^{(2)}\sigma_w^{(2)} - \sigma_v'^{(1)}\sigma_w'^{(1)} - \sigma_v'^{(2)}\sigma_w'^{(2)})& \\
&= J_{vw}\left(\sigma_v^{(1)}\sigma_w^{(1)} + \sigma_v^{(2)}\sigma_w^{(2)} - \sigma_v^{(2)}\sigma_w^{(2)} - \sigma_v^{(1)}\sigma_w^{(1)}\right) \\
&= 0.
\end{aligned}$$

In the case where only one of the spins (but not both) is in the cluster (w.l.o.g assume $v \notin \mathcal{C}$ and $w \in \mathcal{C}$), we find that

$$\begin{aligned}
J_{vw}(\sigma_v^{(1)}\sigma_w^{(1)} + \sigma_v^{(2)}\sigma_w^{(2)} - \sigma_v'^{(1)}\sigma_w'^{(1)} - \sigma_v'^{(2)}\sigma_w'^{(2)})& \\
&= J_{vw}\left(\sigma_v^{(1)}\sigma_w^{(1)} + \sigma_v^{(2)}\sigma_w^{(2)} - \sigma_v^{(1)}\sigma_w^{(2)} - \sigma_v^{(2)}\sigma_w^{(1)}\right) \\
&= J_{vw}\left(\sigma_v^{(1)} - \sigma_v^{(2)}\right)\left(\sigma_w^{(1)} - \sigma_w^{(2)}\right) \\
&= 0,
\end{aligned}$$

where in the last equality, we used the fact that $\sigma_v^{(1)} = \sigma_v^{(2)}$ by design of the Houdayer move. Indeed, since $v \notin \mathcal{C}$, it can only be that their site overlap was equal to 1, i.e., $\sigma_v^{(1)} = \sigma_v^{(2)}$. The case where the site overlap is -1 is impossible as otherwise v would have been added to $\mathcal{C}$ since v is connected to w.

Putting it all together we get

$$\begin{aligned}
\mathcal{H}(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}(\boldsymbol{\sigma}^{(2)}) &= \mathcal{H}_q(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}_l(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}_q(\boldsymbol{\sigma}^{(2)}) + \mathcal{H}_l(\boldsymbol{\sigma}^{(2)}) \\
&= \mathcal{H}_q(\boldsymbol{\sigma}'^{(1)}) + \mathcal{H}_l(\boldsymbol{\sigma}'^{(1)}) + \mathcal{H}_q(\boldsymbol{\sigma}'^{(2)}) + \mathcal{H}_l(\boldsymbol{\sigma}'^{(2)}) \\
&= \mathcal{H}(\boldsymbol{\sigma}'^{(1)}) + \mathcal{H}(\boldsymbol{\sigma}'^{(2)}),
\end{aligned}$$

which concludes the proof. □

From this and the fact that the two configurations are at the same temperature, we conclude that the move chosen is always accepted with probability 1. Also, by noting that after the move the local overlap remains the same, the detailed balance equation is satisfied (see lemma 3.4.2). To be a valid method though, we should still enforce ergodicity, which is not yet the case. So the move has to be combined with some other method, the most simple being to add another simple move on top of it, the usual one being the single-flip procedure.

Lemma 3.4.2. *The Houdayer move satisfies the detailed balance equation.*

Proof. Let $\left(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}\right)$ and $\left(\boldsymbol{\sigma'}^{(1)}, \boldsymbol{\sigma'}^{(2)}\right)$ be two pairs of configuration. We want to show that

$$\begin{aligned}\mu_\beta\Big((\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})\Big)\mathbb{P}\Big((\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}) \to (\boldsymbol{\sigma'}^{(1)}, \boldsymbol{\sigma'}^{(2)})\Big)& \\ = \mathbb{P}\Big((\boldsymbol{\sigma'}^{(1)}, \boldsymbol{\sigma'}^{(2)}) \to (\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})\Big)&\mu_\beta\Big((\boldsymbol{\sigma'}^{(1)}, \boldsymbol{\sigma'}^{(2)})\Big).\end{aligned}$$

If the pairs are not reachable by the move, then the transition probabilities will be both equal to 0, satisfying the equation. Otherwise both pairs have the same total energy, and we have that $\mu_\beta\left((\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})\right) = \mu_\beta\left((\boldsymbol{\sigma'}^{(1)}, \boldsymbol{\sigma'}^{(2)})\right)$. Moreover, noting that when going from one pair to the other through the move the local overlap remains unchanged, this means that the probability of going from a specific pair p to another pair q accessible by flipping one of the Houdayer clusters is the same as the probability from q to p. This is in particular true for the the two pairs in consideration $\left(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}\right)$ and $\left(\boldsymbol{\sigma'}^{(1)}, \boldsymbol{\sigma'}^{(2)}\right)$, so that the detailed balance equation is satisfied. □

Below, we describe two simple properties of the Houdayer move.

Lemma 3.4.3. *Consider a pair of spin configurations $\left(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}\right)$ such that their local overlap induces the set of Houdayer clusters $\mathcal{C}$. Then, when performing a Houdayer move, we have that:*

1. *The move is able to reach $|\mathcal{C}|$ unique pairs.*
2. *If $|\mathcal{C}| = 1$, performing the move leads to the pair $\left(\boldsymbol{\sigma}^{(2)}, \boldsymbol{\sigma}^{(1)}\right)$.*

Proof. Recall that for the Houdayer move, we are allowed to select exactly one of the clusters in $\mathcal{C}$. Moreover since the clusters are spins where the pair of configurations differ, a unique pair will be reached for each choice of cluster.

Since there are $|\mathcal{C}|$ clusters to choose from, the number of unique pairs reachable is simply $|\mathcal{C}|$.

For the second part of the proof, note that if there is only one cluster to choose, then we swap exactly all spins which are different between the two configurations in the pair, which is equivalent to just switching the elements of the pair. □

3.4.3 In Practice

As mentioned earlier, the Houdayer move has to be combined with another move like the single-flip to ensure that the system is ergodic. Moreover, in practice one frequently combines Houdayer moves with parallel tempering in the following way [15]: one starts with multiple replicas at various temperatures. At each iteration, for each temperature, replicas are randomly paired and a Houdayer move is performed, followed by a single-flip one. After that, replicas at neighboring temperatures are randomly paired and exchanged according to equation 3.1. This is described more precisely in algorithm 1.

Algorithm 1: Parallel Tempering with Houdayer Move

```
Input: Number of iterations $N$, Number of replicas $R$ at each temperature, Temperatures
       $\beta_1, \beta_2, \dots, \beta_k$
for $i = 1$ to $k$ do
    Set $R$ random replicas at temperature $\beta_i$.
end
for $n = 1$ to $N$ do
    for $i = 1$ to $k$ do
        Randomly pair the replicas at temperature $\beta_i$ and perform a Houdayer move on
          each of these pairs.
    end
    for $i = 1$ to $k$ do
        for $r = 1$ to $R$ do
            Perform one iteration of the Metropolis algorithm using a single-flip on replica $r$
              at temperature $\beta_i$.
        end
    end
    Pair neighboring temperatures in the variable $t$.
    for $(\beta_n, \beta_m)$ in $t$ do
        Randomly pair replicas at the different temperatures and exchange them with
          probability as described in equation 3.1.
    end
end
```

Notice here the use of the most general setting: many replicas are simulated at multiple different temperatures, and are all updated independently (through the single-flip procedure) and collectively (through the Houdayer move). On one hand, having many replicas allows for faster mixing, while the use of both single-flip and Houdayer moves allows to explore the space of configuration reasonably. On the other hand, having a range of temperatures where replicas can move up and down gives the possibility to further escape the local minima. This way, parallel tempering with Houdayer moves explores both horizontally and vertically.

3.4.4 Limitations

There are multiple limitations and questions that can be raised concerning the Houdayer move. First, one should observe that whenever the local overlap only induces one Houdayer cluster, we know from lemma 3.4.3 that the move will just yield the same pair but reversed, essentially doing nothing (see figure 3.3). This already raises two questions: how often does this happen, and is it possible to avoid it? There are also other details of interest, which we list below.

- **The Houdayer move can fail**

A partial answer to the question of when failures occur is that the more connected the graph of the Ising model G is, the more likely it is that there is only one Houdayer cluster. Indeed in the limit where we consider a complete graph, whatever the local overlap is, all sites with value -1 are necessary be connected and thus form a single Houdayer cluster. The question is related to percolation theory, which describes the formation of long-range connectivity in random graphs, but for topologies like a lattice the Houdayer move works well [14].

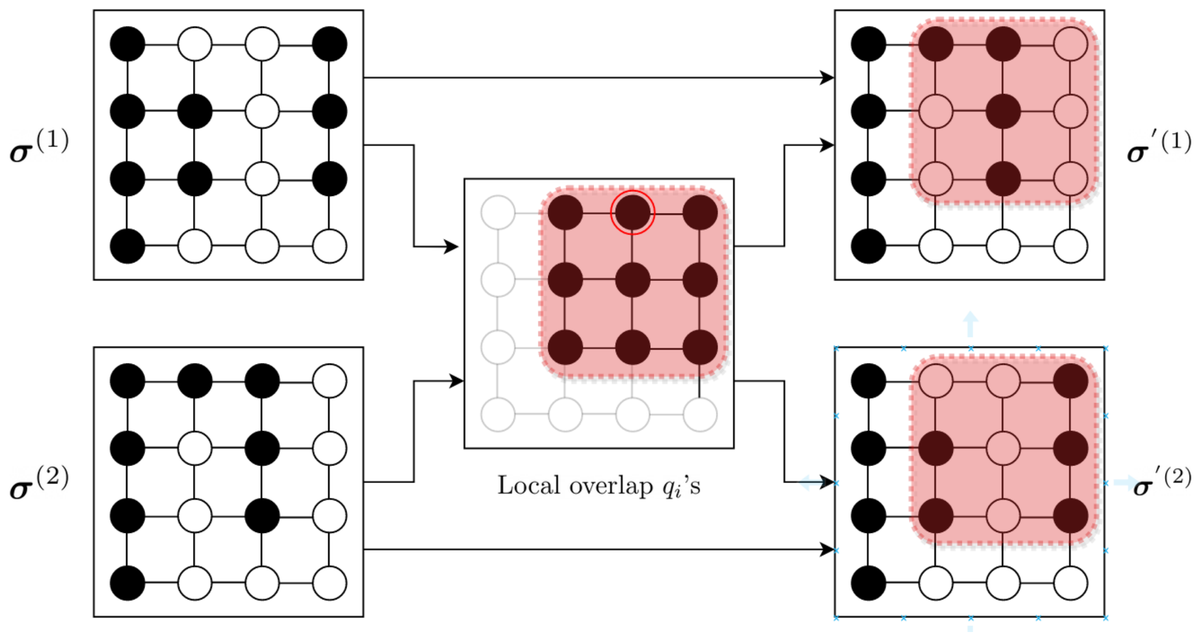

$\boldsymbol{\sigma}^{(1)}$
$\boldsymbol{\sigma}^{(2)}$
Local overlap q_i's
$\boldsymbol{\sigma}'^{(1)}$
$\boldsymbol{\sigma}'^{(2)}$

Figure 3.3 – A visual representation of a failed Houdayer move. It starts with a pair of independent spin configurations $(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})$ and first compute their local overlap, revealing where spins differ between the two configurations. In this case there is only one cluster to choose from, which has the effect of exchanging the two configurations, i.e., $(\boldsymbol{\sigma}'^{(1)}, \boldsymbol{\sigma}'^{(2)}) = (\boldsymbol{\sigma}^{(2)}, \boldsymbol{\sigma}^{(1)})$
.

It is however not clear completely clear how one can avoid the formation of spanning Houdayer clusters. Some research in this line of work has been done to solve the issue in the case of the Ising model with no external magnetic field [15], but it still remains unanswered in the more general case.

- **The Houdayer move is biased**

  Unlike cluster algorithms like Wolff or Swendsen-Wang, the choice of cluster here does not depend on the weights of the bonds, but just on the topology of the graph underlying the Ising system at hand. So one can question the way we decide which cluster to pick. Indeed, since during the cluster selection process we first choose a site with local overlap value $q_i = -1$ and then grow the corresponding cluster, the bigger a cluster is the higher the chance of selecting it. In fact, the probability of selecting a particular cluster is directly proportional to its size. More precisely, if we have Houdayer clusters

$\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ (which form a partition of sites with local overlap $q_i = -1$), then we have that $\mathbb{P}(C_i) = |C_i| / \sum_{j=1}^{n} |C_j|$. But it is not clear why it should be the case that larger changes should be made in order to improve the sampling. In the next chapter, we solve some of these issues by extending the Houdayer move and allowing for example the selection of clusters uniformly at random, and also the possibility to select multiple clusters at the same time.

3.5 Efficient Optimization

So far, we presented the Metropolis algorithm as a way to sample from a desired distribution, enabling the minimization of the Hamiltonian $\mathcal{H}$ of some spin-glass system when performed at low temperature. By combining it with procedures like parallel tempering and new proposal moves like cluster moves, this can potentially improve the sampling process. We present two additional heuristic methods that are important in this project and are widely used in the field of optimization.

3.5.1 Simulated Annealing

Simulated Annealing (SA) [16] is a method used to solve all kinds of optimization problems, many of which are of combinatorial nature like the Maximum Cut problem [36]. It is a slight adaptation of the Metropolis-Hastings to improve the optimization process. Indeed, running the Metropolis algorithm directly at low temperature where we can theoretically sample the global minima often does not work well because we would have to wait too long reach the equilibrium distribution. Instead, simulated annealing uses an idea coming from metallurgy when working with some materials. Often one is interested in increasing the size of the material's crystals and reduce their defect, which is achieved by heating and cooling the material in a controlled way. In the context of optimization, we simulate this annealing process as follows: starting at high temperature, multiple iterations (also called "sweeps") of the Metropolis algorithm are executed until approximate equilibrium is reached. Typically for Ising models, the number of sweeps is often chosen to be equal to the number of spins in the system. Once this happens, the temperature is lowered and the process starts again, as described in algorithm 2. The way the temperature is updated can follow various schemes (a linear decrease or a geometric one for example) and we often refer to it as the "temperature schedule". When the temperature becomes sufficiently low, the corresponding equilibrium distribution is an approximation of the system's ground state distribution. Sampling then gives access to approximately optimal energy configurations.

It can be observed that the algorithm is in a way quite similar in spirit to the idea of parallel tempering, in the sense that it tries to improve sampling by allowing spin configurations to move in the temperature space. It has the advantage of being simple and thus quite fast in practice compared to parallel tempering, however it is good to keep in mind that in its original

Algorithm 2: Simulated Annealing

```
Input: Number of iterations $N$, Number of sweeps $N_s$, Temperature Schedule
       $\beta_1 < \beta_2 < \cdots < \beta_N$
Initialize a random spin configuration $\boldsymbol{\sigma}$.
for $i = 1$ to $N$ do
    Perform $N_s$ iterations of the Metropolis algorithm on $\boldsymbol{\sigma}$ at temperature $\beta_i$.
end
```

form, simulated annealing can only decrease the temperature of the system. Modified versions exist where the temperature can go both up and down which can sometimes make the annealing process more efficient [37].

3.5.2 Genetic Algorithms

Genetic algorithms form a large family of optimizers for both constrained and unconstrained problems. They are based on the process of natural selection and aim to imitate biological evolution by performing a randomized search of the state space [38]. They are presented in more details in section 4.2.3, where we show how the Houdayer move can be used in this setting.

3.6 Comparison of the Methods

A comparison of the various methods introduced above is presented here, with the aim of determining what the limits are depending on the chosen Ising system. We first compare the moves presented so far, from the single-flip strategy to the more complicated Houdayer move all used within the simulated annealing procedure. To do so, simulations are run on multiple Ising systems with different bond weights and topologies. We then proceed to look at how performant simulated annealing and parallel tempering compare on identical problems, but also in terms of running time.

3.6.1 Ferromagnetic Model

For the first set of experiments, the simple ferromagnetic Ising model is considered, i.e., the system can be represented by a graph $G = (V, E)$ with Hamiltonian of the form $\mathcal{H}(\boldsymbol{\sigma}) = -\sum_{(i,j)\in E} J_{ij}\sigma_i\sigma_j$ with $J_{ij} > 0$. The optimum is then known, which is when all spins are aligned, so we have $\mathcal{H}^* = -\sum_{(i,j)\in E} J_{ij}$. The weights are chosen randomly and are distributed uniformly on the interval $[0, 1]$. Notice that in the simple model, there is no external magnetic field, so $h_i = 0$. Also, we always assume that G is a connected graph, since otherwise we can separate the spins in more than one component and solve independently.

Simulations are run using simulated annealing for the following moves: single-flip, Houdayer, Wolff and Swendsen-Wang. Note that the Houdayer move is always combined with a single-flip, so that at each iteration both moves are performed. Each time, the process is run with 25 replicas for 300 iterations so that we can then display the average behavior and also visualize the standard deviation of the mean. The temperature schedule starts with an inverse temperature of $\beta_1 = 0.01$ and follows a geometric increase with factor 1.04. This choice of temperature enables to not perform too much random sampling at first (since the initial inverse temperature is not too low) and allows to reach inverse temperatures high enough so that we are closer to the distribution we want to sample from.

Here, we consider two types of graph, all containing 1024 nodes.

Square Lattice

The first topology, which is of central importance in physics, is the 2-dimensional lattice with results presented in figure 3.4.

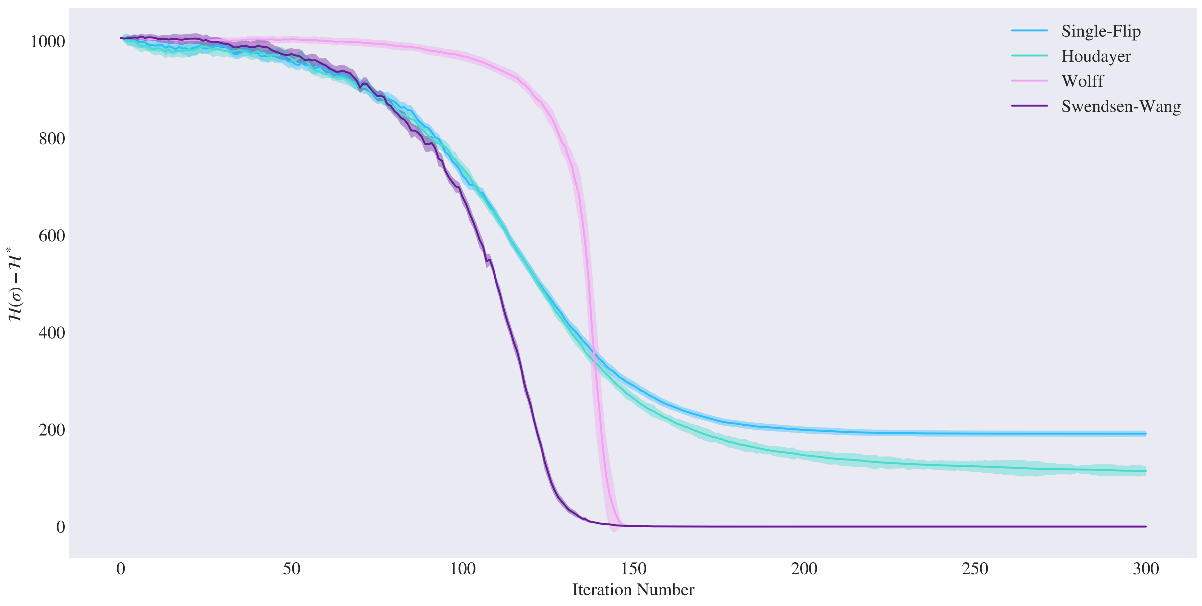


Figure 3.4 – Result of SA for various moves on a 32×32 square lattice graph for the simple ferromagnetic model.

.

For this model and choice of topology, it is well-known that cluster Monte-Carlo algorithms like Wolff and Swendsen-Wang perform very well as observed. An interesting thing to notice is how Wolff algorithm takes a while before really starting to reduce the value of the energy, but then follows the same trajectory as the Swendsen-Wang algorithm. On the other hand, the single-flip method is the worse of all and seems to get stuck in a local minima. Enhancing it with the addition of Houdayer does help a quite a bit, but this is not as good as the other cluster algorithms. Indeed, even with the Houdayer move the optimization process seems to require many additional iterations to reach the ground state.

Erdös-Rényi Graph

For the rest of the experiments, we decide to not consider a specific topology and instead use the Erdös-Rényi model [39] to generate random graphs with a desired number of nodes n.

Definition 3.6.1 (Erdös-Rényi Graph)**.** *An Erdös-Rényi (ER) graph on vertex set V and with parameter p is a random graph which connects independently every possible pair of vertices (v, w) with probability p. It is also denoted as $ER(n, p)$, where $n = |V|$.*

We run simulations for three values of p (0.007, 0.01 and 0.05), resulting in more and more connected graphs since the expected number of edges is given by $\binom{n}{2}p$. Results are presented in figures 3.5, 3.6 and 3.7.

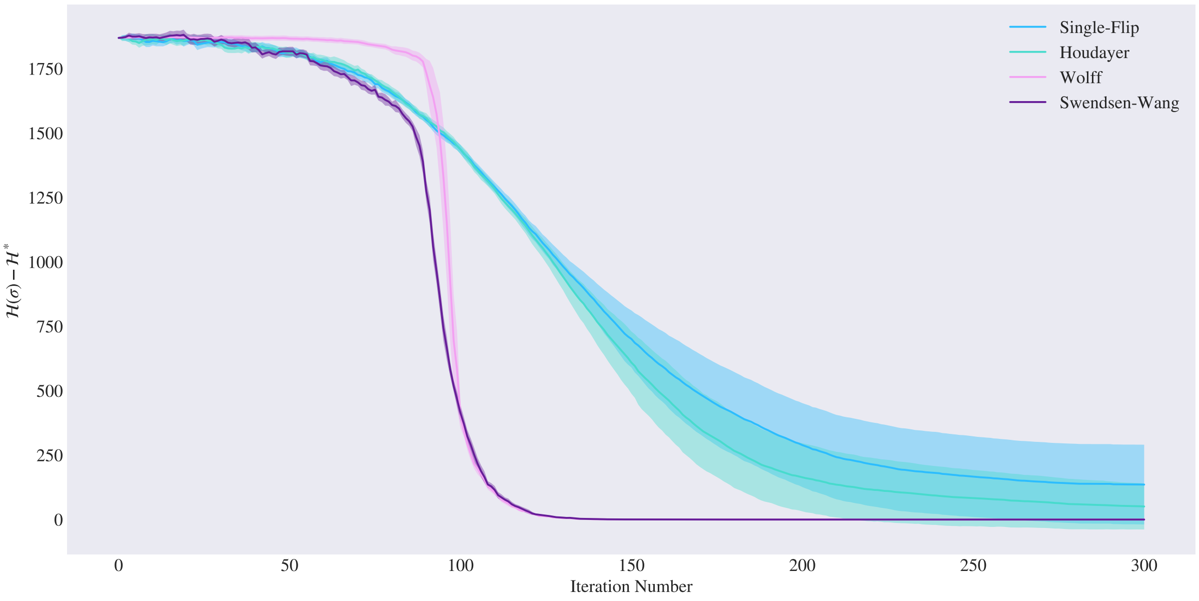


Figure 3.5 – Result of SA for various moves on a $ER(1024, 0.007)$ graph for the simple ferromagnetic model.

.

The first thing to notice is that once again methods like Wolff and Swendsen-Wang are able to reach equilibrium in a low number of iterations for the simple ferromagnetic model. So let us shift the focus to the single-flip strategy and the Houdayer one, and see if results are as expected. When $p = 0.007$ the graph is least dense and this can be seen first through the fact that there is a lot of variability in the trajectories taken since there are less interactions between spins to take into account. Both methods seem to get at least close to optimality but not exactly, with the use of the Houdayer move helping in this sense. When we increase the value of p, we see two main things. First, both methods achieve optimality, with less and less variability as p increases. Secondly, as the graph gets more and more connected, using the Houdayer move becomes less effective since there is more chance of a failed move in this case.

In terms of running time, the picture seems to be reversed as shown in table 3.1. The fastest

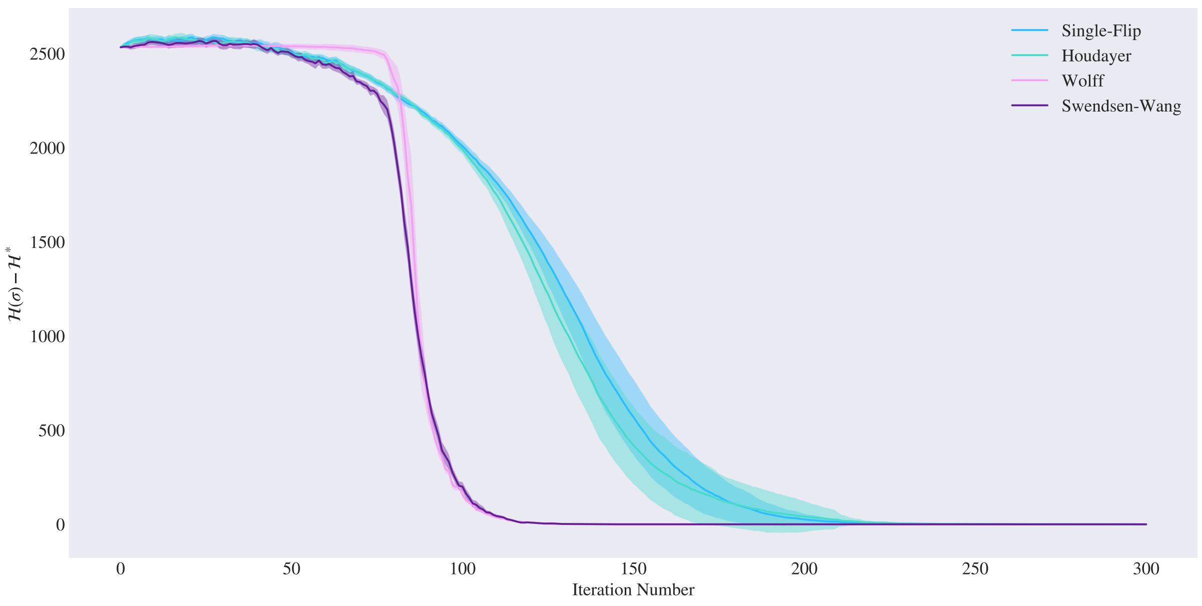


Figure 3.6 – Result of SA for various moves on a $ER(1024, 0.01)$ graph for the simple ferromagnetic model.

.

method is the single-flip since it is an easy update and there exist various ways of making it fast by taking into account the fact that the move is local. As we move toward cluster moves, the running time increases slightly with the Houdayer move. Wolff's algorithm in particular takes more time as it has to grow a cluster and consider many edges in the process. The worst is Swendsen-Wang, as it does the same as Wolff but grows multiple clusters to be flipped. In general for any method, the running time grows proportionally to the number of edges in the graph.

	Single-Flip	Houdayer	Wolff	Swendsen-Wang
2D Lattice ($n = 32 \times 32$)	1.17	1.60	4.5	7.42
ER($n = 1024, p = 0.007$)	1.86	3.63	7.95	13.32
ER($n = 1024, p = 0.01$)	2.38	4.48	10.36	18.11
ER($n = 1024, p = 0.05$)	10.02	16.26	43.82	85.12

Table 3.1 – Summary of average running time (in seconds) over 25 runs of the simulated annealing for different moves and different topologies on simple the ferromagnetic Ising model.

Overall for the ferromagnetic model, it seems clear that Wolff and Swendsen-Wang should be used in most cases to arrive at the stationary distribution. If time is a big constraint, then using the combination of single-flips with Houdayer move does bring some improvement when the graph is not too dense.

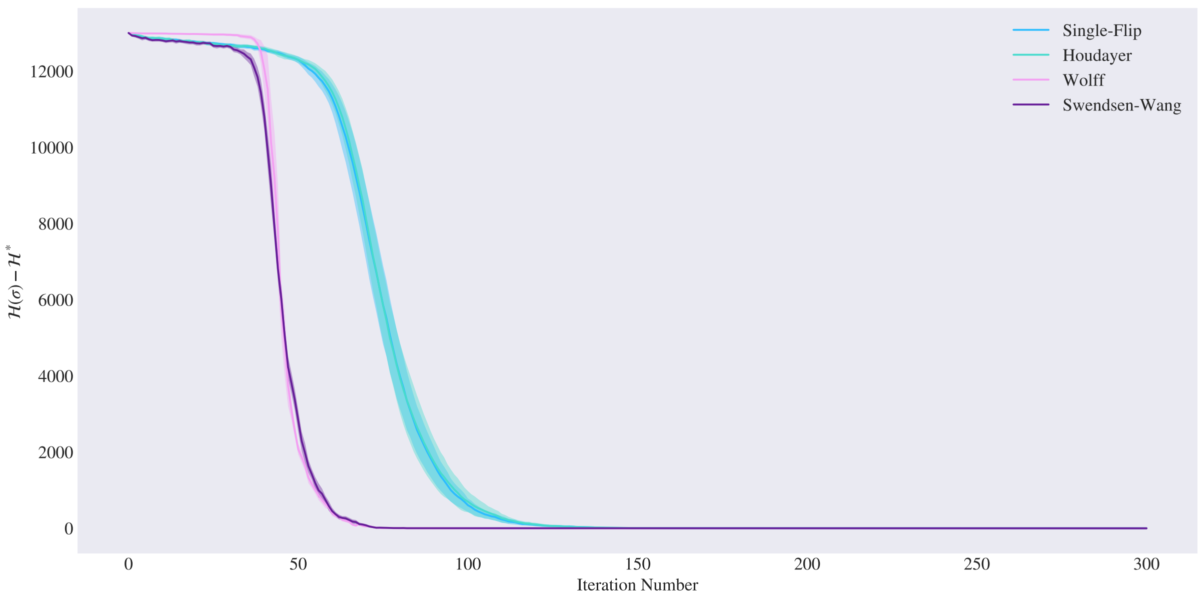


Figure 3.7 – Result of SA for various moves on a $ER(1024, 0.05)$ graph for the simple ferromagnetic model.

.

3.6.2 Spin-glass Model

For the more general spin-glass model, we repeat the same experiments with the same choice of topologies. Recall that the general spin-glass Ising model with underlying graph $G = (V, E)$ and no external magnetic field has Hamiltonian of the form $\mathcal{H}(\boldsymbol{\sigma}) = -\sum_{(i,j)\in E} J_{ij}\sigma_i\sigma_j$ with $J_{ij} \in \mathbb{R}$. The weights are chosen randomly and are distributed following a Gaussian distribution with mean 0 and variance 1. Note also that here the optimum cannot easily be found here so in the plots we display the evolution of the energy of the system.

Simulations are run using simulated annealing with the same parameters as in previous section and results can be found in the appendix A.2.

Square Lattice

The first observation here is that the previously best methods, Wolff's algorithm and Swendsen-Wang, are not able to reach an optimum value. They diverge and stabilize around the same local minimum, showing the limits of the two techniques and the need for new dynamics when simulating general spin-glass systems. With single-flip dynamics, a significantly better minimum is reached, and this is further improved by the use of the Houdayer move as expected [14].

Erdös-Rényi Graph

Similar results are obtained for the generated random graphs, in particular the divergence behavior is observed just as in the lattice case. The local update method attains a lower energy value and is slightly enhanced by the use of the Houdayer move, but again as the graph gets more connected its efficiency decreases as expected.

Overall for the general spin-glass model, it is clear that both Wolff and Swendsen-Wang become inappropriate and this is where the use of the Houdayer move can be interesting. Even though the improvement is not huge, it is still noticeable. In next section, we dig deeper in this direction and see whether the use of parallel tempering can further improve optimization.

3.6.3 Simulated Annealing vs Parallel Tempering

It was identified in section 3.5.1 that even though similar in spirit, simulated annealing and parallel tempering are still different in the sense that in one case replicas can only go down in temperature while in the other, they are able to freely move up and down. Moreover for parallel tempering, one can have multiple replicas simulated at each temperature. A comparison of the two methods is presented, making it clear that parallel tempering is more powerful in finding better minima. Running times are also compared in order to determine which method would make more sense to be used in practice.

The benchmark procedure is similar as before: each method is ran for 1000 iterations, 25 times each in order to capture the average behavior when minimizing the energy of a spin-glass system with random bond weights distributed according to a Gaussian distribution with mean 0 and variance 1. For simulated annealing, the temperature schedule starts at $\beta_1 = 0.01$ and increases geometrically with a factor of 1.04. The temperatures chosen for parallel tempering are such that the first and last temperatures match for both schedules, however only 40 inverse temperatures are considered for parallel tempering and they also follow a geometric increase. Moreover at each temperature, two replicas are simulated. The results reported are the energies of the replicas at the lowest temperature.

Square Lattice

In the case of the lattice (see figure 3.8), one notices that the addition of parallel tempering offers the possibility to lower the energy just slightly more when we restrict to the single-flip move. In that case, using the simple simulated annealing would make sense as it can be ten times faster. However, once the Houdayer move is added, the difference is very clear in the sense that parallel tempering converges much faster to equilibrium.

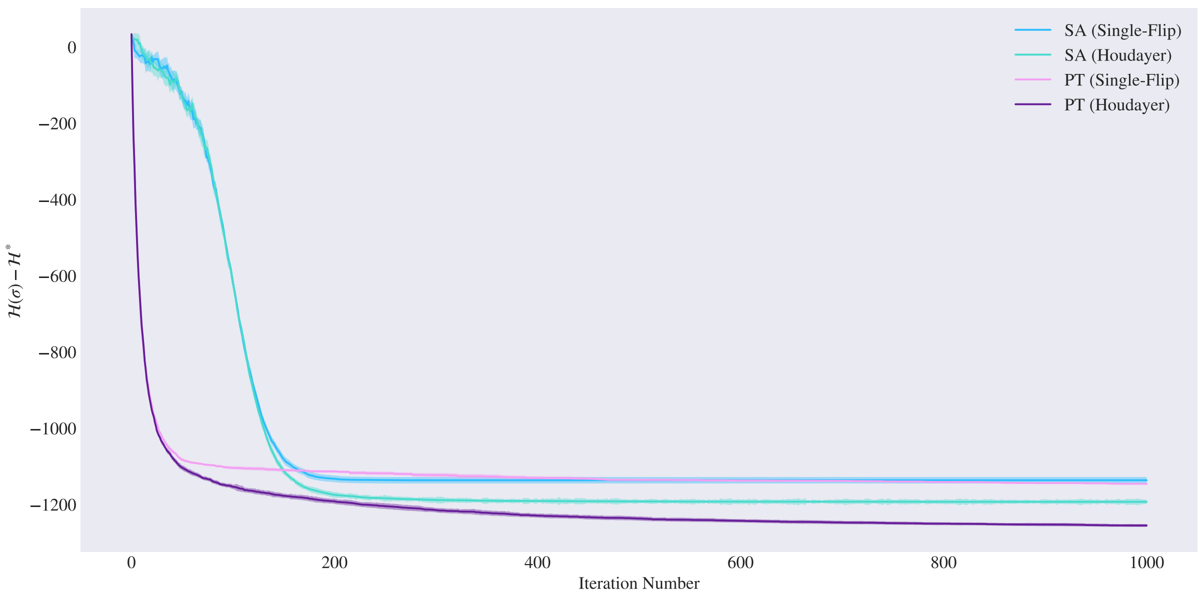


Figure 3.8 – Result of SA vs PT for various moves on a 32 × 32 square lattice graph for the spin-glass model.

.

Erdös-Rényi Graph

For random graphs, the same behavior is observed by generating multiple Erdös-Rényi graphs. We show in figure 3.9 an example when we pick $p = 0.01$ and the conclusions are the same as for the lattice, that is parallel tempering accelerates the convergence to equilibrium.

In terms of running time, we already commented for the simulated annealing in previous sections so we focus on parallel tempering. Results are displayed in table 3.2. We simply notice the running time is significantly higher on average, which is simply explained. Since at each of the 40 temperatures, there are multiple replicas simulated the time increases proportionally to the number of these. By increasing the number of temperatures and the number of replicas simulated at each temperature, we can reach much better minima but at the cost of having a significantly longer running time.

	SA(Single-Flip)	SA(Houdayer)	PT(Single-Flip)	PT(Houdayer)
2D Lattice ($n = 32 \times 32$)	2.01	3.11	20.26	24.32
ER($n = 1024, p = 0.01$)	7.52	17.61	100.14	114.95

Table 3.2 – Summary of average running time (in seconds) over 25 runs of the simulated annealing and parallel tempering for different moves and different topologies for the spin-glass model.

Another consideration with parallel tempering is what number of temperatures should be considered in order to be efficient. It turns out that about 30 to 40 of them is usually sufficient, as shown in figure 3.10 when optimizing an Ising lattice. The best energy level found first

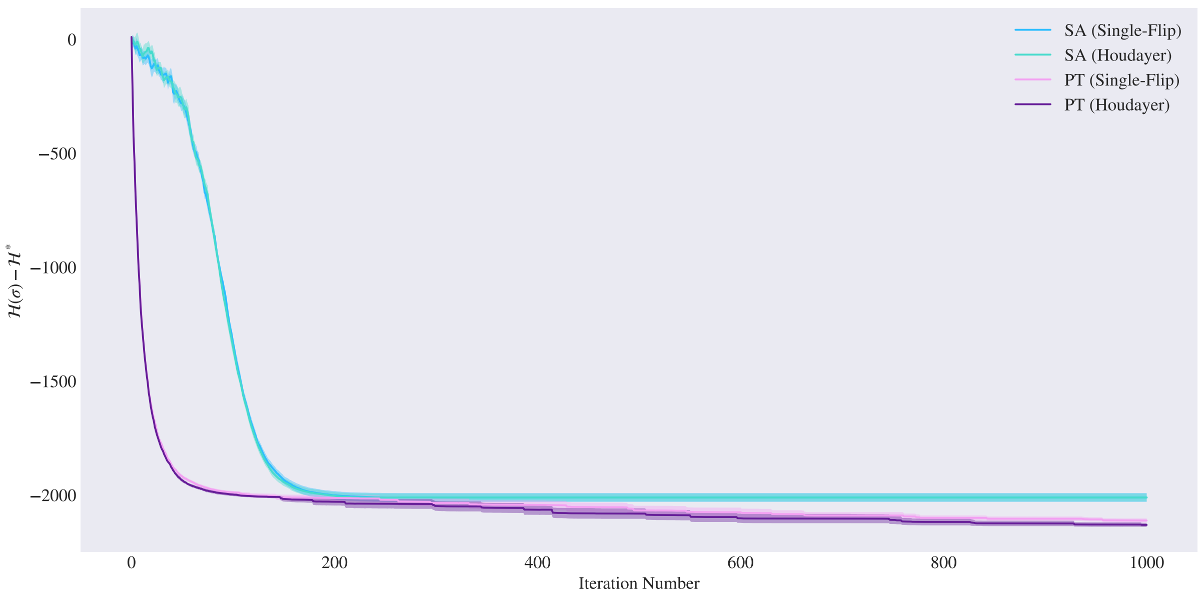


Figure 3.9 – Result of SA vs PT for various moves on a $ER(1024, 0.01)$ graph for the spin-glass model.

.

decreases when we allow more temperatures at which replicas are simulated, but after a certain number the improvement becomes negligible. To summarize this experiment, parallel tempering can yield quite some improvement over the simple simulated annealing procedure. By allowing replicas to move more freely in temperature space, exploration becomes easier and one can supplement it with Houdayer moves to reach lower energy levels.

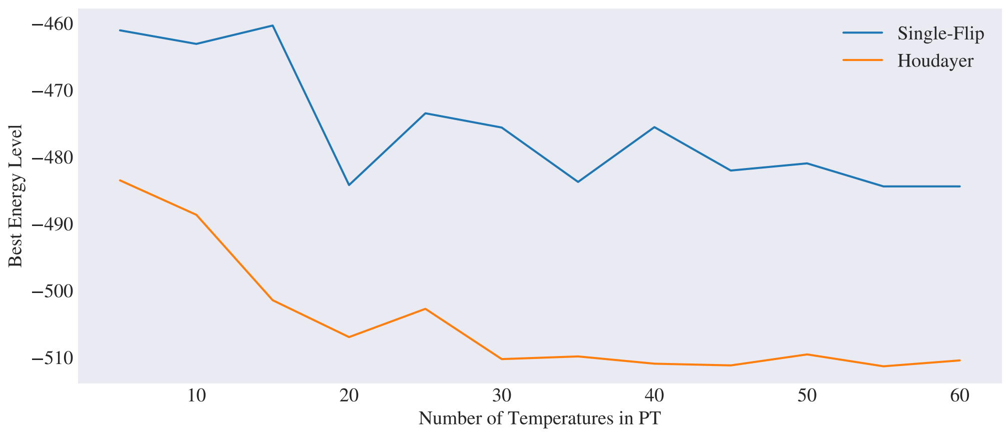


Figure 3.10 – Comparison of the best energy value found when varying the number of temperatures in parallel tempering. The problem solved is an Ising lattice with normally distributed weights, and results are shown for both single-flip and Houdayer moves.

4 Applications to Concrete Problems

This chapter explores further uses of the Houdayer move, by first proposing an extended version allowing for various improvements. A description of an alternate way to use the Houdayer move on its own and not through the Metropolis algorithm is presented in order to refine the sampling of low-energy configurations. Again here, the simple extension gives the possibility to further ameliorate the process, and a genetic algorithm based on these ideas is developed.

The remaining part of the chapter is dedicated to a concrete optimization problem coming from industry, going into details about how the previous developed tools can be used to solve it. In particular, we perform a benchmark to get some empirical results and assess the efficiency of the algorithms.

4.1 Extending the Houdayer Move

The main feature of the Houdayer move is its ability to reach new pairs while preserving the total energy. In section 3.4.4, we raised a few limitations of the original approach, and highlighted the fact that it would give more freedom if after computing the local overlap, any Houdayer cluster could be chosen uniformly at random. By also noting that choosing multiple clusters to be flipped does not impair the move's main property (see lemma 4.1.1), we propose an extension of the original Houdayer move by allowing to attain an exponentially larger amount of pairs at the same total energy level.

4.1.1 Description

Recall that the Houdayer move starts by computing the local overlap, and in so defines a set of Houdayer clusters. The next step is to then choose one of these clusters and perform the swap of the corresponding spin values between the two configurations. The variant we propose, which we refer to as the *extended Houdayer move*, works as follows:

1. Compute the local overlap $q_i = \sigma_i^{(1)}\sigma_i^{(2)}$ at every site just as in the original Houdayer move.
2. The overlap computed in previous step defines a set of Houdayer clusters $\mathcal{C}$. Select any combination $T \subseteq \mathcal{C}$ of Houdayer clusters according to a desired probability distribution and flip the corresponding spins in both configurations. Since there is no specific reason to take a special distribution over all possible combinations, we argue that selecting uniformly at random out of all possible combinations is a reasonable choice.

To summarize, the extended move considers not only one Houdayer cluster to flip, but any combination (could be none or all clusters as well) of them. It might not seem clear yet why we would allow to flip no clusters or all of them as this essentially corresponds to a failed Houdayer move, but it will be treated in chapter 5. A depiction of the move can be found in figure 4.1. Note that by enabling the choice of any distribution on the combination of clusters to flip, this can allow to control for example "how far" we jump in terms of Hamming distance in the configuration space, which could potentially be beneficial in certain applications.

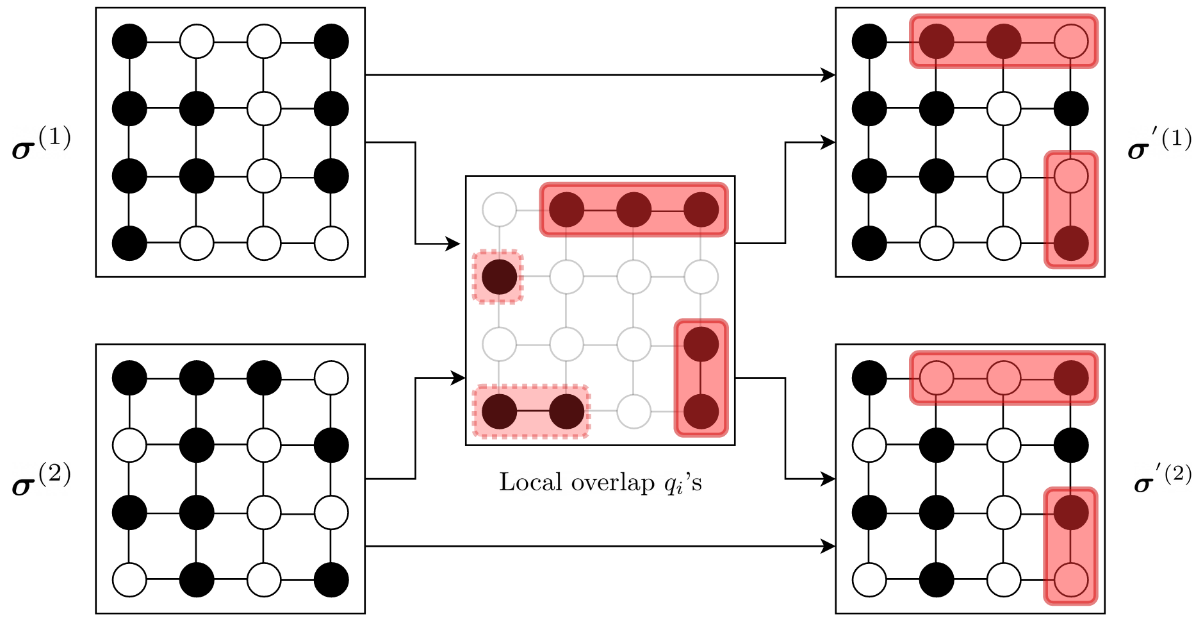

$\boldsymbol{\sigma}^{(1)}$
$\boldsymbol{\sigma}^{(2)}$
Local overlap q_i's
$\boldsymbol{\sigma}'^{(1)}$
$\boldsymbol{\sigma}'^{(2)}$

Figure 4.1 – A visual representation of the extended Houdayer move. It starts with a pair of independent spin configurations $(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})$ and first computes their local overlap, revealing where spins differ between the two configurations. This defines a set of Houdayer clusters (highlighted in red, both dashed and solid lines), from which a combination of these is chosen uniformly at random (circled in red with solid lines). After that, the spins in the selected clusters are swapped between the two replicas, or equivalently they are flipped, yielding the new pair $(\boldsymbol{\sigma}'^{(1)}, \boldsymbol{\sigma}'^{(2)})$.

From this general framework, we can find the original Houdayer move simply by choosing the distribution over the possible combinations to be equal to 0 whenever the combination

$T \subseteq \mathcal{C}$ does not have exactly one cluster (that is $|T| \neq 1$), otherwise we weight it just like in the original move as explained in section 3.4.4. Moreover, this means that whatever pairs are accessible through the original Houdayer move, they will also be accessible for the extended move.

4.1.2 Properties

One can easily follow the same proof technique used for the original move to show that the total energy of the pair is preserved through the move.

Lemma 4.1.1. *Consider a pair of spin configurations $(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})$ and suppose that an extended Houdayer move results in the new pair $(\boldsymbol{\sigma}'^{(1)}, \boldsymbol{\sigma}'^{(2)})$. Then we have $\mathcal{H}(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}(\boldsymbol{\sigma}^{(2)}) = \mathcal{H}(\boldsymbol{\sigma}'^{(1)}) + \mathcal{H}(\boldsymbol{\sigma}'^{(2)})$.*

Proof. We follow the same proof idea as in lemma 3.4.1 and prove that both the linear part and quadratic part of the sum of the Hamiltonian values remains unchanged.

Suppose that the local overlap induces the set of Houdayer clusters $\mathcal{C}$ and that we choose the combination of clusters $T \subseteq \mathcal{C}$.

1. Linear part

   We have

$$\begin{aligned}
&\mathcal{H}_l(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}_l(\boldsymbol{\sigma}^{(2)}) - \mathcal{H}_l(\boldsymbol{\sigma}'^{(1)}) - \mathcal{H}_l(\boldsymbol{\sigma}'^{(2)}) \\
&\qquad = -\sum_{v \in V} h_v(\sigma_v^{(1)} + \sigma_v^{(2)} - \sigma_v'^{(1)} - \sigma_v'^{(2)}) \\
&\qquad = -\sum_{C \in T} \sum_{v \in C} h_v(\sigma_v^{(1)} + \sigma_v^{(2)} - \sigma_v'^{(1)} - \sigma_v'^{(2)}) \\
&\qquad = -\sum_{C \in T} \sum_{v \in C} h_v(\sigma_v^{(1)} + \sigma_v^{(2)} - \sigma_v^{(2)} - \sigma_v^{(1)}) \\
&\qquad = 0,
\end{aligned}$$

   where in the second equality we used the fact that for all spins not in the combination of clusters, their value is unchanged resulting in the corresponding term in the sum to be zero.

2. Quadratic part

$$
\begin{aligned}
\mathcal{H}_q(\boldsymbol{\sigma}^{(1)})+&\mathcal{H}_q(\boldsymbol{\sigma}^{(2)}) - \mathcal{H}_q(\boldsymbol{\sigma}'^{(1)}) - \mathcal{H}_q(\boldsymbol{\sigma}'^{(2)}) \\
&= - \sum_{(v,w)\in E} J_{vw} \left(\sigma_v^{(1)}\sigma_w^{(1)} + \sigma_v^{(2)}\sigma_w^{(2)} - \sigma_v'^{(1)}\sigma_w'^{(1)} - \sigma_v'^{(2)}\sigma_w'^{(2)}\right) \\
&= - \sum_{C\in T} \sum_{\substack{(v,w)\in E: \\ v\in C \text{ or } w\in C}} J_{vw} \left(\sigma_v^{(1)}\sigma_w^{(1)} + \sigma_v^{(2)}\sigma_w^{(2)} - \sigma_v'^{(1)}\sigma_w'^{(1)} - \sigma_v'^{(2)}\sigma_w'^{(2)}\right),
\end{aligned}
$$

where again we used that when we consider an interaction involving spins which are unchanged, the term in the sum is zero. For the other terms involving the combination of clusters, we continue to follow the proof of lemma 3.4.1 since the analysis also applies in this case, and get that

$$
\mathcal{H}_q(\boldsymbol{\sigma}^{(1)}) + \mathcal{H}_q(\boldsymbol{\sigma}^{(2)}) - \mathcal{H}_q(\boldsymbol{\sigma}'^{(1)}) - \mathcal{H}_q(\boldsymbol{\sigma}'^{(2)}) = 0,
$$

as expected

$\square$

The improved properties of the extended move are presented below, and can be compared to those presented in lemma 3.4.3 for the original move.

Lemma 4.1.2. *Consider a pair of spin configurations $\left(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}\right)$ such that their local overlap induces the set of Houdayer clusters $\mathcal{C}$. Then, when performing an extended Houdayer move, we have that:*

1. *The move is able to reach $2^{|\mathcal{C}|}$ unique pairs of configurations.*
2. *For any pair $\left(\boldsymbol{\sigma}'^{(1)}, \boldsymbol{\sigma}'^{(2)}\right)$ that is reachable, the pair $\left(\boldsymbol{\sigma}'^{(2)}, \boldsymbol{\sigma}'^{(1)}\right)$ also is.*
3. *The number of unique configurations found through the move is equal to $2^{|\mathcal{C}|}$.*

Proof. Recall that for the extended move, we are allowed to select any combination of clusters in $\mathcal{C}$. Moreover since the clusters are spins where the pair of configurations differ, a unique pair will be reached for each combination of clusters chosen.

Since there are $|\mathcal{C}|$ clusters to choose from, and we can decide any number of clusters, the number of total combinations (or unique pairs) is equal to

$$\sum_{i=0}^{|\mathcal{C}|} \binom{|\mathcal{C}|}{i} = 2^{|\mathcal{C}|},$$

as required.

For the second part of the proof, we show how to choose clusters that yield two pairs where one of them is the "inverted" version of the other. To this end, suppose one chooses to flip the combination of clusters $T \subseteq \mathcal{C}$. Since in the pair of configurations spins are different on $\mathcal{C}$, let us focus on these spins precisely.

Looking at $\boldsymbol{\sigma}^{(1)}$, we get that the configuration $\boldsymbol{\sigma}'^{(1)}$ after swapping is essentially the same as before except now that for spins in T, they have value of those in $\boldsymbol{\sigma}^{(2)}$. Notice that spins in $\mathcal{C} \setminus T$ still have the values of those in $\boldsymbol{\sigma}^{(1)}$.

On the other hand, we have that $\boldsymbol{\sigma}'^{(2)}$ is unchanged except on T where it has now the values of $\boldsymbol{\sigma}^{(1)}$. Notice that spins in $\mathcal{C} \setminus T$ still have the values of those in $\boldsymbol{\sigma}^{(2)}$.

Now instead of doing the move using T for the clusters to flip, suppose we use the clusters $\mathcal{C} \setminus T$. Looking at $\boldsymbol{\sigma}^{(1)}$, we get that the configuration $\boldsymbol{\sigma}'^{(1)}$ after swapping is the same as before except now that for spins in $\mathcal{C} \setminus T$, they have value of those in $\boldsymbol{\sigma}^{(2)}$. Notice that spins in T still have the values of those in $\boldsymbol{\sigma}^{(1)}$. But this is exactly equal to $\boldsymbol{\sigma}'^{(2)}$. In the same way, getting the second element of the pair actually yields $\boldsymbol{\sigma}'^{(1)}$.

We can thus conclude that from any pair of configurations $\left(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}\right)$, performing the extended Houdayer move with clusters in T gives us a pair $\left(\boldsymbol{\sigma}'^{(1)}, \boldsymbol{\sigma}'^{(2)}\right)$, while using the clusters $\mathcal{C} \setminus T$ yields the pair $\left(\boldsymbol{\sigma}'^{(2)}, \boldsymbol{\sigma}'^{(1)}\right)$.

The last part is proven by combining the first two results. From second point, we know that we should only consider half of the pairs, and from first point, we know every pair is unique. Since there are two configurations in a pair, the number of unique configurations is $2 \cdot 2^{|\mathcal{C}|-1} = 2^{|\mathcal{C}|}$ as claimed. $\square$

Point 2 gives an interesting take on what is considered a failure in the original Houdayer move. Essentially if one chooses no cluster to flip (that is we choose the combination $T = \emptyset$), we get back the exact same pair as we started with, but if we pick the complement $\mathcal{C} \setminus T = \mathcal{C}$, so all clusters, we know that the pair is inverted. So in the framework of the more extended Houdayer move, a "failure" happens whenever all clusters are selected.

A detail which is interesting to highlight as it can make a difference in practice, is another consequence of point 2. The fact that for any accessible pair, the pair with same elements but inverted is also accessible tells us that when not considering the ordering of a pair, half of them are in fact identical. As such, if we are interested in computing all unique pairs but without considering the ordering, then only half of them needs to be computed. In fact, the configurations that form these pairs corresponds to the $2^{|\mathcal{C}|}$ unique configurations mentioned in point 3.

4.1.3 In Practice

In practice, the selection of one of the combination uniformly at random is done in two steps. The following scheme is proposed:

1. Choose a number i of Houdayer clusters to be picked according to $\mathbb{P}(\text{select } i \text{ clusters}) = \binom{|\mathcal{C}|}{i}/2^{|\mathcal{C}|}$. So i is distributed according to a Binomial random variable $\text{Bin}(n, p)$ with $n = |\mathcal{C}|$ and $p = 1/2$.
2. Sample uniformly at random i Houdayer clusters from the $|C|$ available ones.

The explanation comes from the fact that in total there are $2^{|\mathcal{C}|}$ combinations to choose from (see proof of lemma 4.1.2), so at the end of the selection process we want to get any of them with the same probability $1/2^{|\mathcal{C}|}$. Let us see that it is indeed the case, so consider a combination $T \subseteq \mathcal{C}$ with $|T| = j$ Houdayer clusters. The probability to pick T is equal to

$$\begin{aligned}
&\mathbb{P}(\text{select } j \text{ clusters and select the combination } T)\\
&\qquad\qquad = \mathbb{P}(\text{select } j \text{ clusters}) \cdot \mathbb{P}(\text{select the combination } T)\\
&\qquad\qquad = \frac{\binom{|\mathcal{C}|}{j}}{2^{|\mathcal{C}|}} \cdot \frac{1}{\binom{|\mathcal{C}|}{j}}\\
&\qquad\qquad = \frac{1}{2^{|\mathcal{C}|}}
\end{aligned}$$

as desired.

4.1.4 Greedy Version of the Move

The version of the algorithm presented in section 4.1.1 has the advantage of being "as random as it gets". Indeed, we argued that the original Houdayer move had the downside of being biased into picking certain new pairs out of all possible ones. With the extended Houdayer move with uniform choice of combination, any pair is accessed with the same probability.

One can of course question this and instead follow a more greedy approach when getting to a new pair of configurations. A simple idea is to not choose just one pair at random, but first generate all the $2^{|\mathcal{C}|}$ that are accessible (see lemma 4.1.2). In particular, we also know that there are $2^{|\mathcal{C}|}$ unique configurations in total. From all these, we can make a greedy choice and pick the two with lowest energy level as the new pair. This strategy is shown in figure 4.2.

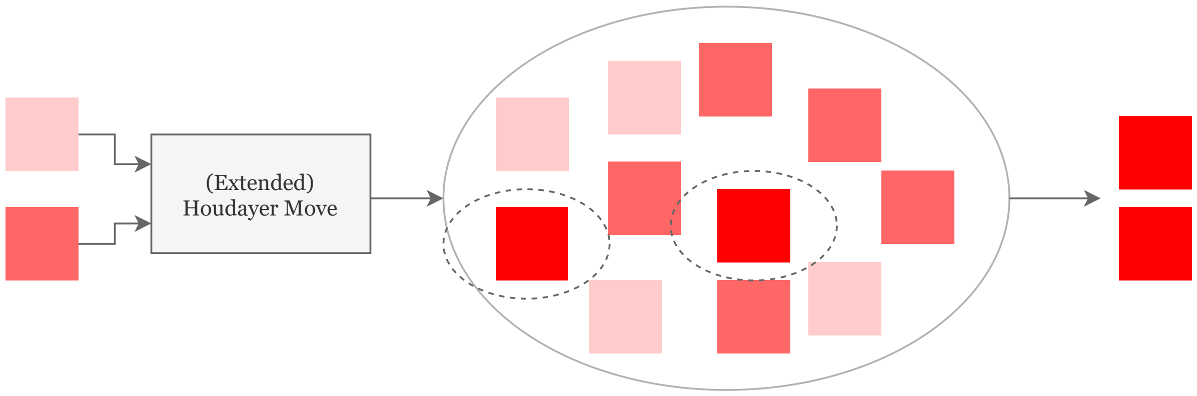


Figure 4.2 – A visual representation of the greedy Houdayer move, where configurations have a darker red color when their energy is lower. One starts by computing the local overlap, determining all possible combinations as in the extended Houdayer move, and thus all accessible pairs at the same energy level. The greedy choice happens when the new pair is chosen to be the two configurations with lowest energy values out of all generated (which visually corresponds to the two with the darker red).

Unfortunately, since the number of accessible pairs is exponential in the number of Houdayer clusters in the local overlap, this strategy can become incredibly costly. In order to reduce this complexity, we can instead pick a random sample of p pairs out of all possible pairs that are accessible through the extended move and keep the two best configurations among these just as before. This way, the time spent finding new pairs can be adjusted by changing the value of p. Note that when we take only $p = 1$ sample pair, this scheme reduces to the extended Houdayer move with uniform choice over all combinations of Houdayer clusters.

4.1.5 Numerical Results

We present multiple experiments to compare the extended Houdayer move with the original one and the greedy version of it. Since the greedy version can be more or less greedy depending on the number of pairs sampled when computing accessible pairs, we wish to understand how much it affects the optimization process. Experiments are performed for the same topologies as experiments in previous chapter and with the same schedules and parameters. Also, multiple runs are still done and the average behavior is displayed together with the standard deviation.

Square Lattice

First, we compare the behavior on the 32×32 square lattice, using the Houdayer move, its extended version, and the greedy version where at each step we sample 50 pairs from all possible ones. The resulting plot is shown in figure 4.3, where we notice that using the extended version of the move slightly improves the convergence. There is however a more apparent improvement when employing the greedy version. Indeed, the same energy level as the other methods is reached using only a few iterations. In an attempt to make the difference between the techniques more visible, a similar experiment is done on a larger lattice.

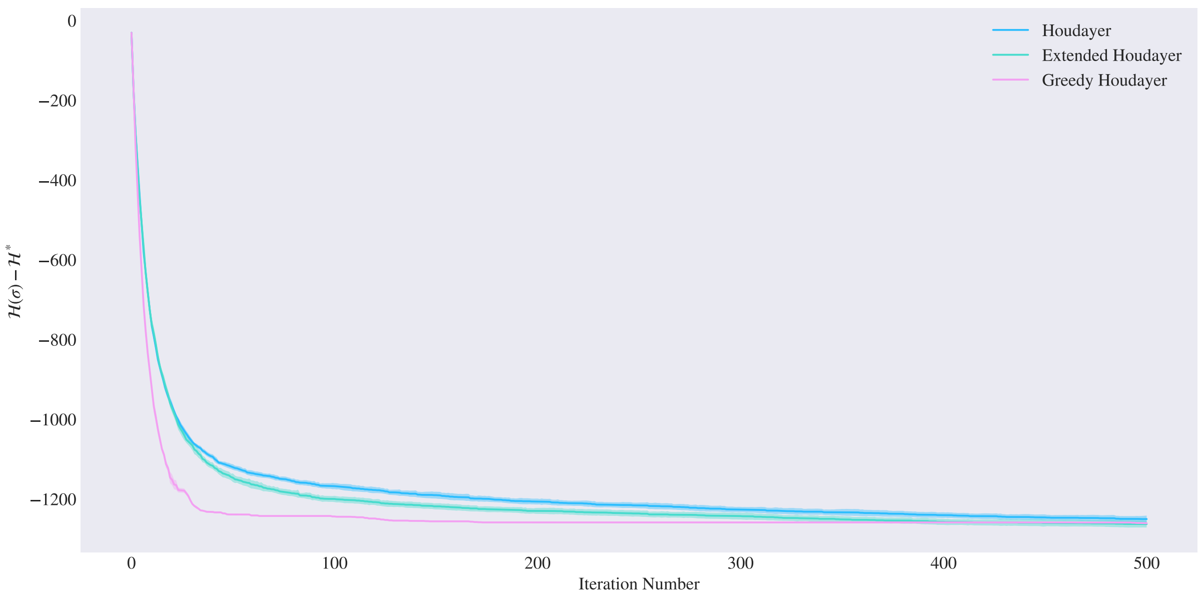


Figure 4.3 – Result of PT for different variants of the Houdayer moves on a 32×32 square lattice graph for the spin-glass model.

The two plots 4.4 and 4.5 are representative of what can happen in a large square lattice when using the different forms of the Houdayer move. In the top plot, one can notice first that the extended Houdayer move performs slightly better than the original one. The interesting outcome is for the greedy strategy, which is able to reach lower levels of energy much faster than the other two. One should also keep in mind that in order to do so, a lot more processing has to be done since many possible pairs are being generated and then compared to keep the two best configurations. Additionally, even though it can go down the energy landscape faster, we note that it get stuck in some local minimum and at iteration 900, the extended Houdayer move is able to reach better values.

Figure 4.5 shows another run of the same model (but with different random weights), but this time the greedy procedure is able to perform much better than the others. To sum up, the greedy procedure can be very powerful but from its random nature when sampling pairs, it can happen to get stuck in a local minimum.

Another type of experiment one can do is to look at the behavior of the greedy method depending on the number of pairs taken. We thus repeat previous experiment but this time for a spin-glass model with bonds following the $\mathcal{N}(0,1)$ distribution, and comparing greedy methods when sampling 50 and 250 pairs. Results are displayed in figure 4.6 showing that eventually, all methods converge to a common energy, with the greedy methods arriving in

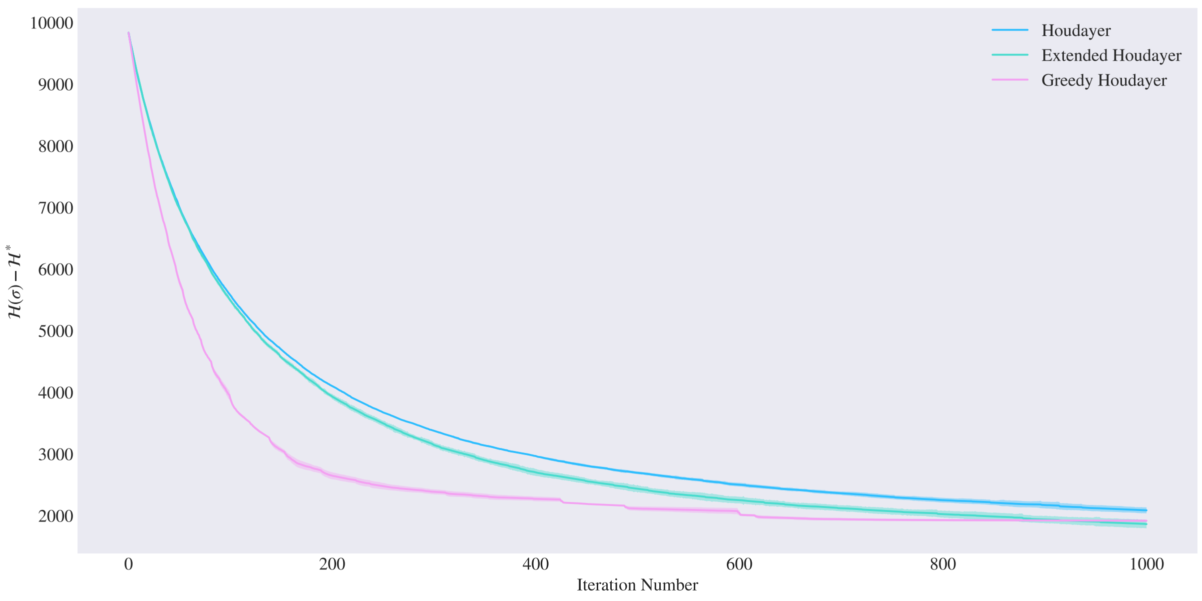


Figure 4.4 – Result of PT for different variants of the Houdayer moves on a 100×100 square lattice graph for the ferromagnetic model. Here, the greedy Houdayer move gets stuck in a local minimum which is worse than the other methods.

approximately half as many iterations as the extended Houdayer method. Comparing the two variants of the greedy approach, we see that choosing 250 pairs instead of 50 pairs results in a slightly faster convergence rate per iteration. More results about running time can be found in table 4.1.

Erdös-Rényi

For Erdös-Rényi random graphs, a comparison of the different Houdayer moves on 1024-nodes graph is performed, with results in figure 4.7.

Results are similar in the sense that greedy methods tend to reach a minimum faster than the others. However we see here that there is not much difference between the original Houdayer move and the extended one. Indeed, since the graph is now random and does not have a particular structure like a lattice (with low degree at each vertex), it could be that there are fewer Houdayer clusters in the local overlap, making both the original and extended move essentially perform similar moves. We thus re-iterate the experiment but on a random graph with 10'000 nodes (figure A.6). There is still not much difference with the use of the extended move here as well, suggesting that the original Houdayer move is already pretty good in terms of "horizontal" exploration of the landscape, even though at each step it can only access less pairs than the extended move.

In terms of running time, the trend is clear: the greedier the method, that is the more pairs are computed, the more processing time is required. Table 4.1 summarizes the values obtained for

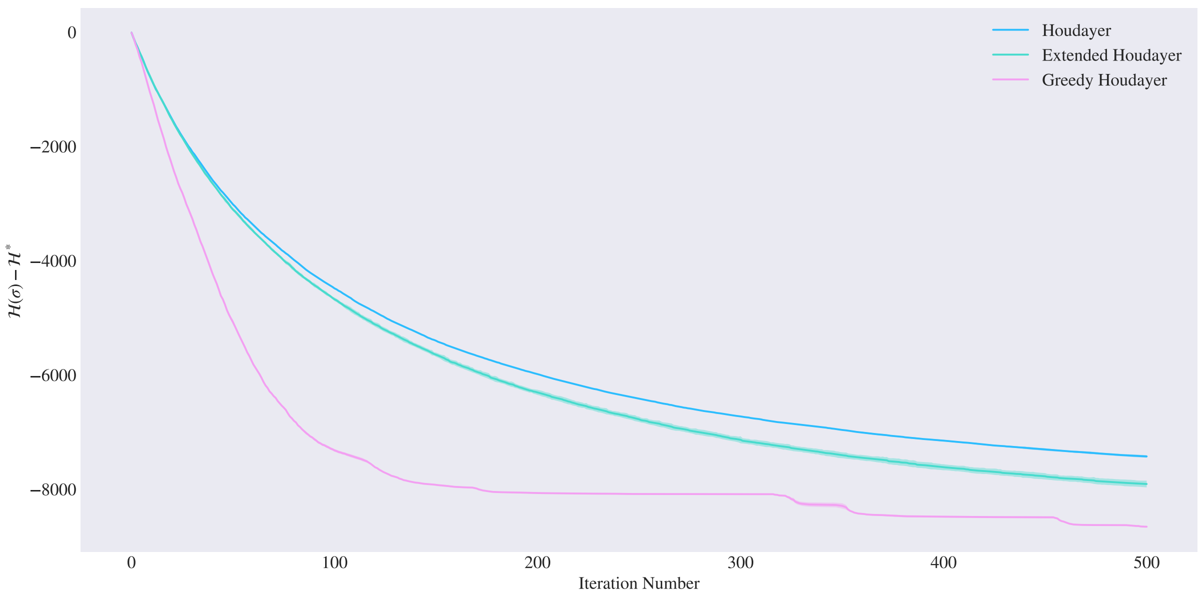


Figure 4.5 – Result of PT for different variants of the Houdayer moves on a 100×100 square lattice graph for the ferromagnetic model. Here, the greedy Houdayer move is able to reach lower energy levels than other methods.

the graphs with 10'000 nodes.

	SA(Extended)	SA(Greedy 50 pairs)	SA(Greedy 250 pairs)
2D Lattice ($n = 100 \times 100$)	15.86	19.29	79.39
ER($n = 10000, p = 0.007$)	24.22	31.3	125.23

Table 4.1 – Summary of average running time (in seconds) over 25 runs of the simulated annealing for different Houdayer variants and different topologies for the spin-glass model.

4.2 Improved Sampling of Low Energy Configurations

Existing classical optimization techniques can solve computationally hard problems exactly but their complexity is often worse than polynomial in the size of the input. For example, calculating exact ground states of Ising models on general graph topologies is NP-hard [40]. Here, we are interested in finding many of the ground states of degenerate systems while still requiring a reasonable computing time, that is polynomial in the size of the input. In previous chapter, we mentioned the use of heuristic methods in order to solve these hard problems approximately in a more adequate amount of time. One issue that arise in this context is that these techniques tend to be biased and produce correlated solutions [41]. Depending on how well they balance exploration and exploitation, some parts of the state space will be visited more than others, and in the worst case some may never be visited at all.

Moreover as we explore in section 4.3, for some industrial applications we do not necessarily

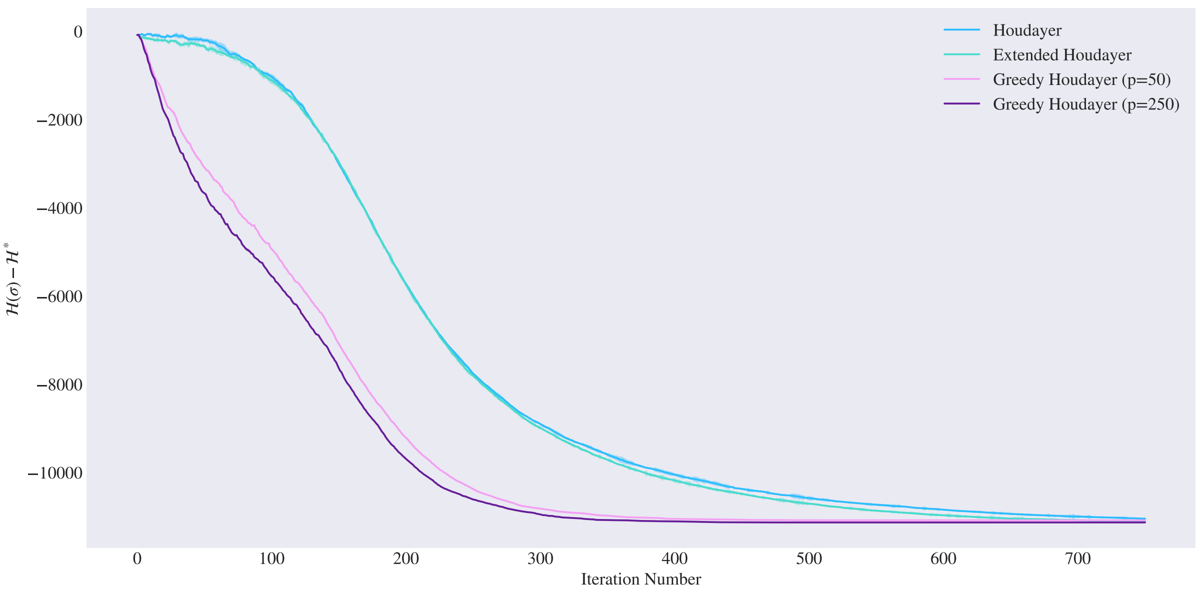
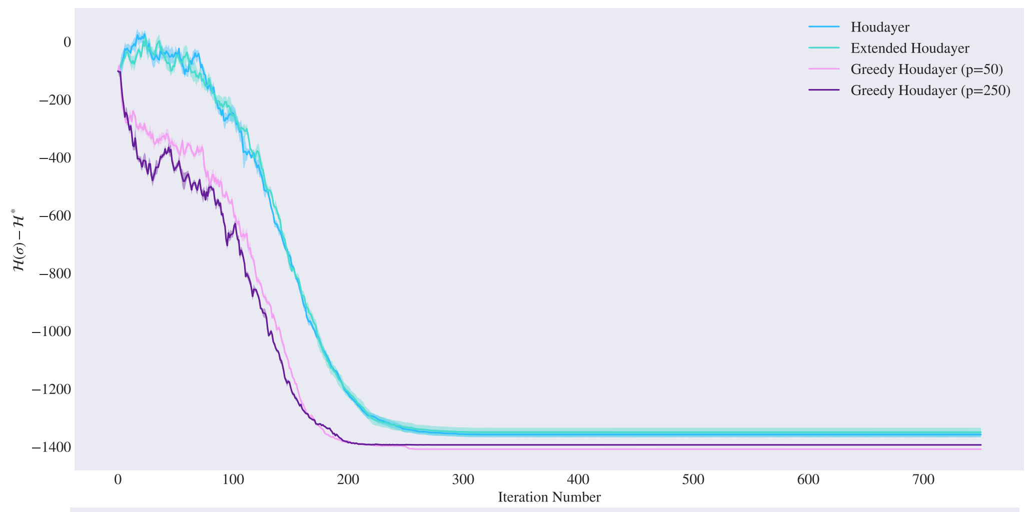


Figure 4.6 – Result of SA for different variants of the Houdayer moves on a 100×100 square lattice graph for the spin-glass model.

care about the minimum cost solution but often desire to have many varied low-energy solutions. Here, we demonstrate that cluster updates combined with some simple heuristics can yield algorithms which can sample more fairly the low-energy regions of the state space. A similar approach has been explored in [42], where a scheme to generate unexplored ground states of an Ising problem is proposed.

4.2.1 Improved Sampling of Ground States

We first start with an idea similar to that in [42]: starting from a pool of a few optimal configurations, the algorithm generates potentially new solutions using the Houdayer move. The concept is the following: the main property of the Houdayer operation is its ability to "teleport" from a pair of configurations to another one with same total energy, and this property happens to be interesting when we are at optimality. Indeed, suppose we are given two ground states $\left(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}\right)$ of a certain Ising problem, that is $\mathcal{H}\left(\boldsymbol{\sigma}^{(1)}\right) + \mathcal{H}\left(\boldsymbol{\sigma}^{(2)}\right) = 2\mathcal{H}^*$. Since the move preserves this energy, what one sees typically is that the energy of one of the configuration decreases, while the other increases by a similar amount. However in the case of two ground states, decreasing is impossible so it must be that the two configurations in the new pair are also ground states. Note that we still have the issue that when there is only one Houdayer cluster induced by the local overlap, this will result in the same pair but with configurations exchanged which is of no use here. Thus, this method is mainly applicable in cases where the topology is not too dense and the Houdayer clusters do not percolate.



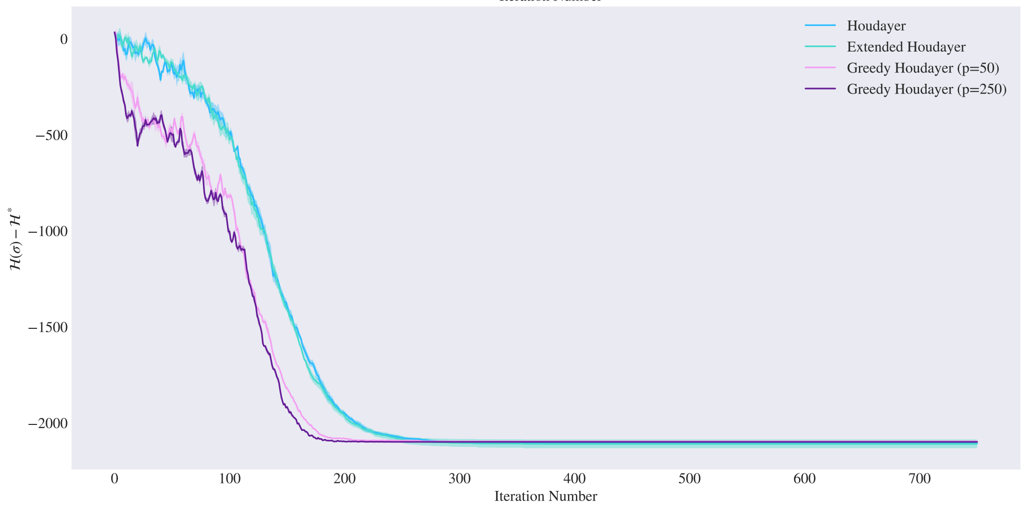


Figure 4.7 – Result of SA for different variants of the Houdayer moves on a random 1024-nodes graph ($ER(n = 1024, p = 0.005)$ on the top plot, and $ER(n = 1024, p = 0.01)$ on the bottom one) for the spin-glass model.

Algorithm Description

The procedure, also described in algorithm 3, is as follows:

1. Initialize a pool P of configurations.
2. Construct (random) pairs of configurations from P.
3. For each pair, compute the local overlap and generate all accessible unique configurations (see considerations from lemma 4.1.2) from that pair. Add the new configurations to the pool P if they are not there already.
4. Go back to step 2, or stop when desired or if no new configuration can be generated.

Algorithm 3: Ground States Generator

```
Input : Pool of configurations $P$, Maximum number of iterations $n_{max}$
i ← 0
while  $i < n_{max}$ do
    Construct a set of (random) pairs of configurations $Q$ from $P$.
    for $q$ in $Q$ do
        Compute the local overlap and generate the set of all accessible unique
          configurations $C$ (see considerations from lemma 4.1.2) from pair $q$.
        P ← $P \cup C$
        if $|C| = 0$ then
            break
        end
        i ← $i + 1$
    end
end
return $P$
```

We can make the following observations:

- In step 1 we actually allow any configuration to be part of the pool (not only optimal ones), simply because the algorithm can also find ground states when it is given low-energy configurations. This property is leveraged to develop a genetic algorithm in section 4.2.3.
- In step 2, we are essentially performing an extended Houdayer move, but actually returning all accessible pairs. To reduce the running time, one can just return a sample of the accessible pairs. If we were to only use a Houdayer move to generate a new pair, the process would still work but would be potentially much worse since many pairs would not be considered.
- Regarding the running time, the main dependence is on how many iterations are performed and how pairs are chosen in step 2, so that the overall running time is polynomial in the input size. Indeed, if one considers all possible pairs in the pool of configurations this gives more opportunities for a successful Houdayer move to happen but then one needs to spend $O(|P|^2)$ time processing the pairs at each iteration. Denoting the number of spins in a configuration by n, the processing of a pair mainly requires to find the Houdayer clusters, which can be done in $O(n^2)$ time algorithm like Depth-First Search [43]. The running time at each iteration is then $O(|P|^2 n^2)$.
- Moreover, as the algorithm keeps going the pool gets larger and larger so it could scale very badly if the pool has many configurations. Instead, one can choose to simply pair randomly the configurations in the pool for example, or also sample a certain number of pairs out of the $O(|P|^2)$ available ones.

It is also important to note that this process does not ensure that all ground states are going to be found, which is simply due to the fact that the (extended) Houdayer move is not an ergodic operation. This is due to the fact that starting from a certain pair at a certain total energy level, there is no guarantee that the computation of local overlap and the choice of clusters to flip allow to access all possible pairs at this energy level in the state space.

Experiments

The purpose of these experiments is two-fold as they both assess the efficiency of the algorithm 3 and also show the improvement gained by using the extended Houdayer move. Given a few ground states of degenerate Ising problems with N^{GS} ground states, we use the algorithm 3 in various ways in order to generate more ground states. In order to compare, we give different numbers of ground states N^{GS}_{in} to start with as input, and after the algorithm is done we compare with the number of ground states at the output N^{GS}_{out}. Note that this kind of experiment is also performed in [42]. In the worst case, the algorithm does not manage to generate anything so that $N^{GS}_{out} = N^{GS}_{in}$, while in the best case, the algorithm finds all the ground states, i.e., $N^{GS}_{out} = N^{GS}$.

Computations are done on simple Ising models that in fact are instances of the Maximum Cut problem on graphs of increasing complexity with unit edge weights. Three different instances of triangular lattices are chosen (see figure 4.8) as they have many ground states and are not too densely connected, giving a chance for our procedure based on the Houdayer move to be useful. To get all ground states in each case, a brute-force procedure is applied as the size of the graphs is not too big. We recall that a Maximum Cut instance for a graph $G = (V, E)$ with unit edge weight can be defined as the minimization of the Hamiltonian $\mathcal{H}(\boldsymbol{\sigma}) = \sum_{(v,w)\in E} \sigma_v \sigma_w$ [19].

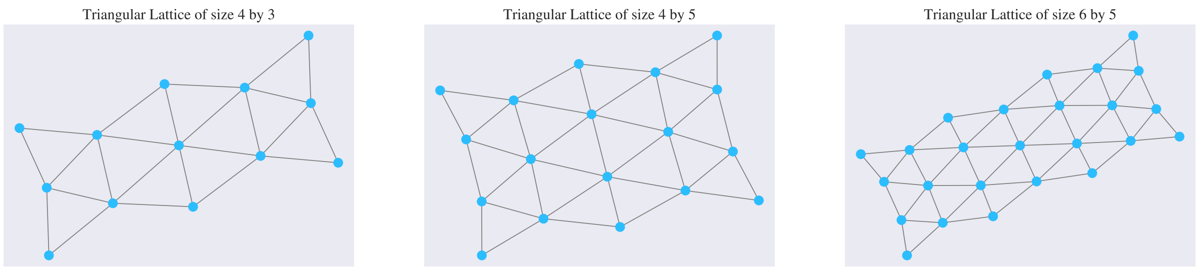

Triangular Lattice of size 4 by 3
Triangular Lattice of size 4 by 5
Triangular Lattice of size 6 by 5

Figure 4.8 – Triangular lattices of sizes 4×3, 4×5 and 6×5, for which the maximum cut is evaluated. They have $N^{GS} = 64, 150$ and 225 ground states, respectively.

For every graph, we first run the algorithm only for one iteration, comparing how using the extended Houdayer move over the original one can be beneficial in finding more ground states. We also include the results still comparing the two strategies but this time performing the full algorithm. That is, as we keep finding new ground states, we add them to the pool and start again until nothing more can be generated.

As briefly mentioned earlier, step 2 of the scheme can take time proportional to the total number of possible pairs from the pool. To speed it up, a simple tweak is to take only a sample of a couple pairs from all possible ones. In our experiments, we test this by repeating the algorithm, sampling different number of pairs and comparing the performance. Results are shown in figures 4.9, 4.10 and 4.11.

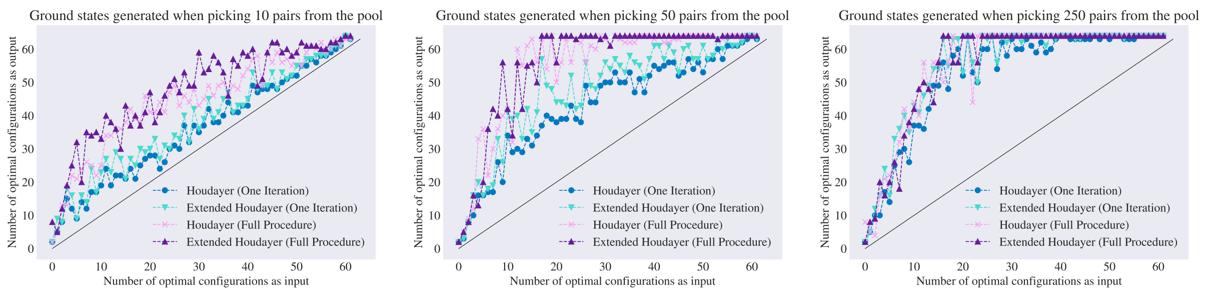


Figure 4.9 – Comparison between the number of generated ground states compared to the number of given ones for the MaxCut problem on the 4 by 3 triangular lattice (with $N^{GS} = 64$). The different plots show the difference when a different amount of pairs is sampled from the pool.

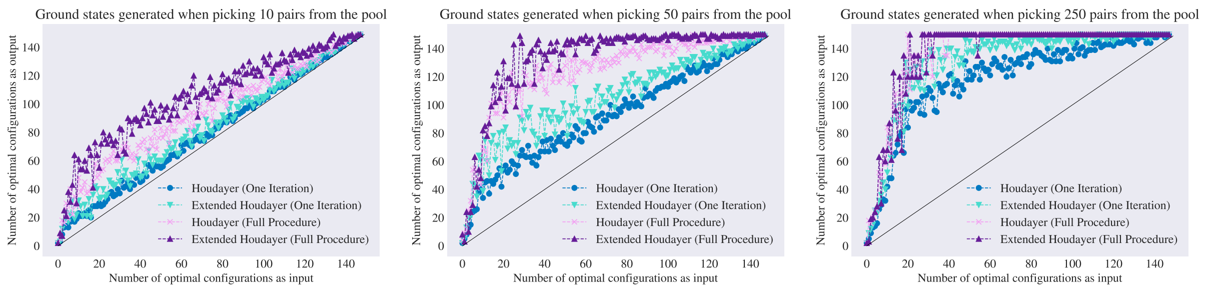


Figure 4.10 – Comparison between the number of generated ground states compared to the number of given ones for the MaxCut problem on the 4 by 5 triangular lattice (with $N^{GS} = 150$). The different plots show the difference when a different amount of pairs is sampled from the pool.

Having various graph complexities gives us a way to see how methods compare not only for toy problems, but also for systems with many spins. In every case, the following behavior is observed: running the algorithm for only one iteration already gives access to more ground states when N_{in}^{GS} is sufficiently large, in particular when using the strategy derived from the extended Houdayer move, which can only perform better since it always access the same pairs as the original Houdayer move, plus all additional ones. Things get even more interesting once we perform many iterations and keep adding new optimal configurations to the pool, allowing for many new ground states to be found. Indeed, especially when the number of pairs sampled gets higher, it is often possible to find all ground states even from a couple given ones. For example in figure 4.9 in the middle plot, with fewer than 20 ground states, we are able to find the full set of 64 ground states that give the maximum cut value. Maybe impressively, this behavior is observed even more sharply for higher-dimensional problems where for the largest triangular lattice we see from figure 4.11 that even when taking only 50 pairs from a pool of only 20 optimal configurations, we can find all 225 ground states. We also notice that the more

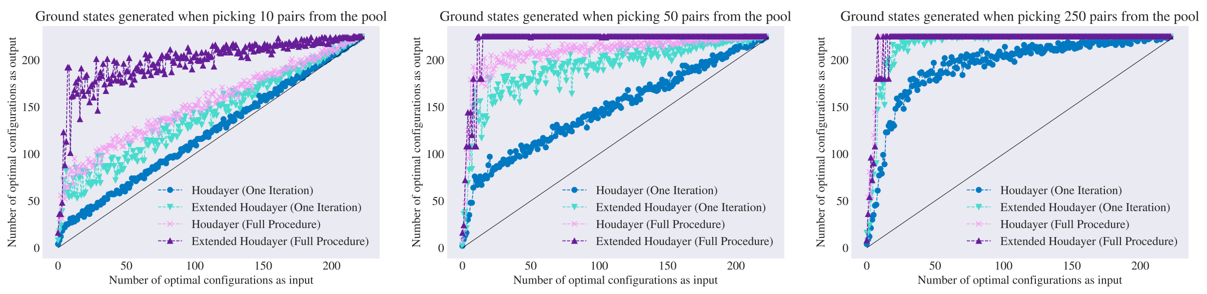


Figure 4.11 – Comparison between the number of generated ground states compared to the number of given ones for the MaxCut problem on the 6 by 5 triangular lattice (with $N^{GS} = 225$). The different plots show the difference when a different amount of pairs is sampled from the pool.

sample pairs we take, the less difference between the different methods since we give more opportunities to have successful Houdayer moves and thus reach new pairs of ground states.

4.2.2 Improved Sampling of Feasible Solutions in Constrained Problems

Consider the case of a constrained minimization problem, which defines a state space with feasible and non-feasible regions. The feasible region is a subspace of the entire state space whose elements respect all constraints of the problem, while the non-feasible region is comprised of elements which violate at least one of the constraints. For such problems, even getting in the feasible region can be a non-trivial task, so one is often interested in finding multiple configurations that are feasible, and then choose the best among them. Ultimately, one would like to be able to navigate the space of feasible configurations in order to carry out the minimization directly in this space without having to worry about the constraints anymore.

Constrained problems represented by Hamiltonian

In this section, we still restrict to the case of the Ising model, so somehow there should be a way to separate through the Hamiltonian whether a configuration is feasible or not. A simple way to do so is to think of the Hamiltonian as a penalty function, assigning high values to infeasible configurations in order to easily determine feasibility from the value of the Hamiltonian. In theory, it would be convenient to have a Hamiltonian of the form

$$\mathcal{H}(\boldsymbol{\sigma}) \begin{cases} < +\infty, & \text{if } \boldsymbol{\sigma} \text{ respects all constraints} \\ = +\infty, & \text{otherwise,} \end{cases} \tag{4.1}$$

effectively encoding whether a configuration is feasible or not. This of course cannot be done for any type of constraints since the Hamiltonian is restricted to be a function which is a quadratic polynomial. For now, we omit those details and explain later on which additional assumptions are needed.

With this setup, algorithm 3 can be readily used by giving it a pool of *feasible* solutions. Note that we are not concerned with how those solutions are actually previously obtained. Whatever pair is chosen in the pool, its total energy is finite since all configurations are feasible, so that after generating using the extended Houdayer move we only reach other pairs with finite energy, that is none of them can have their Hamiltonian equal to $+\infty$. This process thus help in generating new feasible configurations from a set of given initial ones.

However in practice, we can never really have $+\infty$ so this should be taken care of. One can do so by setting the energy value to a very high positive number λ, so that any configuration with energy $\geq \lambda$ is considered infeasible. But then, does the algorithm still works? As long as during the (extended) Houdayer move, the energy of one the pair does not exceed the threshold λ, then we are ensured that the move works.

Example 4.2.1. *Suppose we have a Hamiltonian with $\mathcal{H}^* = 0$ and we are able to encode feasibility by setting the penalty threshold at $\lambda = 10$. Ths way, every configuration with energy higher than λ is considered infeasible.*

Now, consider a pair of feasible configurations $\left(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}\right)$ with $\mathcal{H}\left(\boldsymbol{\sigma}^{(1)}\right) = 8$ and $\mathcal{H}\left(\boldsymbol{\sigma}^{(2)}\right) = 7$, from which we want to get more feasible configurations. After a successful (extended) Houdayer move, the new pair has the same total energy of 15, but it could well be that one is $12 > \lambda$ while the other is 3, not yielding two feasible solutions. In fact, if the configurations in the starting pair have energy already close to the threshold λ, the procedure can fail.

In fact, there is a simple way to make the scheme work as we argue below.

Claim 4.2.1. *Let $\mathcal{H}$ be the Hamiltonian corresponding to a constrained minimization problem, with minimum $\mathcal{H}^*$, and such that feasible solutions have their energy lower than the threshold value λ. Then the scheme presented before is ensured to work only for pairs $\left(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}\right)$ with $\mathcal{H}\left(\boldsymbol{\sigma}^{(i)}\right) < \frac{\lambda+\mathcal{H}^*}{2}$ for $i = 1, 2$. Geometrically, this imposes that the energy values of both configurations have to be closer to the minimum $\mathcal{H}^*$ than the threshold λ.*

Proof. Let $\left(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}\right)$ be a pair with $\mathcal{H}\left(\boldsymbol{\sigma}^{(i)}\right) < \frac{\lambda+\mathcal{H}^*}{2}$ for $i = 1, 2$. Note that after the Houdayer move, in the most extreme case, one of the configuration will see its value decrease to the minimum $\mathcal{H}^*$, while the other's energy will increase by the same amount. Any change in energy greater than that would be impossible as it would contradict the fact that $\mathcal{H}^*$ is a minimum. We thus check that the new value for the configuration which gets its energy increased does not exceed λ.

W.l.o.g. suppose that the first configuration's energy level decreases to the minimum $\mathcal{H}^*$, inducing an energy difference of $\Delta E := \mathcal{H}\left(\boldsymbol{\sigma}^{(1)}\right) - \mathcal{H}^*$. From the assumptions, we have

$\Delta E < \frac{\lambda+\mathcal{H}^*}{2} - \mathcal{H}^* = \frac{\lambda-\mathcal{H}^*}{2}$. Thus the second configuration's new energy level becomes $\mathcal{H}\left(\boldsymbol{\sigma}^{(2)}\right) + \Delta E < \frac{\lambda+\mathcal{H}^*}{2} + \frac{\lambda-\mathcal{H}^*}{2} = \lambda$, still ensuring feasibility as desired. □

Essentially, the higher the value of λ, the wider the range of possible pairs we can take into account, so it makes sense to pick the largest finite number representable for λ. In the limit case where $\lambda \to +\infty$ we recover a Hamiltonian of the form of equation 4.1.

Overall, still using the same algorithm but with low-energy solutions in the starting pool, this gives the ability to generate many more of them, which can be particularly useful in the context of constrained problems where finding feasible solutions can be hard. With our procedure, we can do so in a very fast and easy way and only require a handful of solutions to start with. The process could be taken even further: now that we are able to explore the feasible space more, can we continue the optimization process at the same time? We discuss it in the next section.

4.2.3 Further Optimization: A Genetic Algorithm

Consider a hard (constrained) minimization problem for which it is difficult to sample low-energy solutions. Maybe through various optimization schemes one can get a couple of good solutions, but this is often a very long process depending on the computing requirements. Here, we construct a genetic algorithm which starting from an initial pool of solutions is potentially able to continue the minimization process further. In the next section, it is tested on an actual optimization problem coming from industry.

Structure of a Genetic Algorithm

There exist many ways to formulate genetic algorithms, but the core idea is that it is a procedure which maintains a population of configurations (called individuals) that usually are set to be initially random, and that evolve in a way which is supposed to imitate evolution, hopefully getting us closer to lower-energy configurations [38]. The three main steps that the algorithm iterates over are:

1. **Crossover**, in which the individuals in the population are combined in order to produce offspring which have attributes coming from the parent individuals.
2. **Mutation**, in which individuals go through random transformations which produce new individuals.
3. **Selection**, in which individuals that are believed to be the fittest (according to some criteria) are kept in the population while others are removed in order to regulate the size of the population.

Our algorithm follows this framework, making it simple to use and implement. Note that even though a genetic algorithm is a heuristic method, some convergence results exist [44] and they are simple to implement which makes them still an attractive choice when solving optimization problems.

Algorithm Description

In our case, the different steps of a genetic algorithm translate to the following:

1. **Crossover**

   New offspring are generated by randomly pairing individuals in the population, and then performing an (extended) Houdayer move to potentially reach new pairs.

2. **Mutation**

   There is no specific mutation done in our case since we want to ensure that we stay at a low-energy level. In a sense, the random part of the algorithm also comes from the way the random pairing is done in previous step.

3. **Selection**

   Individual which have a low energy are considered to be of better quality, thus the selection process simply keeps the k individuals which have the lowest value for the Hamiltonian. The parameter k is to be determined and fixes a limit on the size of the population.

A more precise description is given in algorithm 4.

Algorithm 4: Genetic Algorithm for Better Optimization

```
Input: Number of iterations $N$, Initial population $P$, Threshold $k$ for the population size
       limit
for $n = 1$ to $N$ do
    Construct a set of (random) pairs of configurations $Q$ from $P$.
    for $q$ in $Q$ do
        Compute the local overlap and generate the set of all accessible unique
          configurations $C$ (see considerations from lemma 4.1.2) from pair $q$.
        $P \leftarrow P \cup C$
    end
    Retain $k$ configurations from $P$ with lowest energy values.
end
return $P$
```

There are in fact many degrees of freedom here where different variants of the algorithm can be used depending on also how much computing power one wants to spend generating new configurations.

- **Random pairing of individuals in the population**

Just as in the algorithm from section 4.2.1, the pairing of individuals can be done in various ways. Assuming high computing power, considering all pairs of individuals maximizes the chance of successful creation of new pairs during the crossover phase. Since the population size is limited by k, the complexity of this step is of order $O(k^2)$. In practice, this is unfortunately a time-consuming operations given that for each pair, all possible new pairs have to be computed.

A solution to this issue is to once again consider only a small sample of p pairs out of all possible ones, which can drastically reduce the running time. Indeed, the number of pairs is simply always constant with respect to the population size, yielding a complexity of $O(1)$. However with this approach, it can often happen that not all individuals in the population are used. Indeed, if there are 100 individuals, but we only consider 10 random pairs, many of the individuals are not used at all in the process.

A second solution is to simply randomly pair the elements of the population. With a population size of k individuals, $\lfloor k/2 \rfloor$ pairs are created, giving now a linear complexity $O(k)$.

- **Move to execute when generating offsprings**

  During the crossover phase of the algorithm, the idea of the extended Houdayer move is used in order to generate all possible pairs at the same energy level. However since the number of accessible pairs can be large, it can take too long to compute all clusters in the overlap and then find all combinations. Hence, at this step it can make sense to trade the computation of all pairs as done in the extended Houdayer move to a simple Houdayer move which only produces one new pair. This of course depends on how successful the creation of new pair is, which itself depends on whether there is only one Houdayer cluster in the local overlap. Note that one could also sample some of the reachable pairs in the same spirit as the greedy version of the extended Hoduayer move (section 4.1.4).

- **Customize the selection process**

  During the selection process, the best k individuals are chosen in terms of their energy which is a greedy strategy. Unfortunately what can happen is that given a certain population, new offsprings created have a higher or same energy level, preventing them from being retained for the next iteration. At this point, the algorithm is stuck since it cannot generate any new individuals with low enough energy.

  One remedy to this problem when creating the new population is to force a certain fraction of the individuals to come from the offsprings, and another fraction to be from previous population. This way, even though we may keep individuals with higher energy coming from the offsprings, it ensures more variety in the population. To even allow for a balance, one can even change the fraction at each iteration, reducing slowly the number of considered offsprings to allow convergence to happen.

4.3 An Industry Problem: Sequential Scheduling

This section introduces the main optimization problem of this thesis. A description is given and a mathematical formulation is derived, with the goal of designing an Ising system such that minimizing its energy is equivalent to getting the optimal solution of our problem. We demonstrate how to adapt it in order to be able to use the techniques presented previously and test their effectiveness by performing several numerical experiments.

We refer to the problem of interest as the "sequential scheduling" problem, and describe its key features below. For large enough instances, finding a feasible solution to this problem can be challenging, due to the non-trivial structure of the constraints. The goal is to come up with an alternate methodology inspired from physics-based algorithm (like the Houdayer move) in order to better solve the problem and attain the feasible region. In particular, we ask two questions. Is it possible to generate many (uncorrelated) feasible solutions of the problem and also reduce the processing time to do so? And if so, can the optimization go further and yield higher-quality solutions?

4.3.1 Description

The problem studied is concerned with sequential scheduling, where one is interested in scheduling the processing of certain items in order to minimize the time spent processing them. At the same time, some constraints are expected to be met in terms of number of items processed and in terms of space constraints.

In its simplest form, the problem can be described in the following terms. There are two entities, called A and B. Every day, entity A runs for a certain number of hours and is able to produce a certain number of items, and entity B is able to process a certain number of items that come from entity A. In between the two, there is a cache which is able to hold some items, as it may happen that entity B processes less elements than entity A produces. The setup is depicted in figure 4.12. As an example, entities could be computing units producing and processing jobs. Part of a job is processed by entity A while the other part is done by entity B. In between, the cache would represent some kind of queue in that case.

The goal is to process a certain amount N of items from entity B at the end of a specified time period, which for simplicity here we will take to be n days. Additionally, we wish to minimise the total operating time of the entities, while ensuring that the cache never overflows. Here, we will measure the operating time of the entities in hours. In more complex formulations of the problem, the number of items processed is required to fall in a range $[(1-\delta)N, (1+\delta)N]$, where $\delta \geq 0$, instead of being exactly equal to N. For every day $k \in \{1, 2, \dots, n\}$, we need to choose the number of hours A runs from a set of numbers A_k, and the number of hours B runs from a set of numbers B_k. In this simple version of the problem, we assume that one hour of running time amounts to one item produced or processed. The cache in-between can hold a

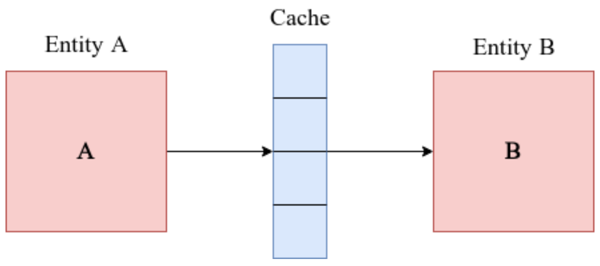


Figure 4.12 – A simple depiction of the setup for the sequential scheduling problem. Entity A produces items which are then fed to entity B for further processing. In between, there is a cache which can hold items in case entity B is not able to process as much as entity A is outputting.

maximum of Z items at any point in time.

4.3.2 Mathematical Formulation

A formulation is described using binary variables, since the goal is to apply the Houdayer move to generate new feasible solutions from given ones. To find feasible solutions, it is much simpler to use a formulation which uses discrete variables and not binary ones. One can then use for example a simulated annealing procedure to solve that problem. However in order to use our improved sampling schemes, we are constrained to work in the Ising (or equivalently QUBO) model since it is based on the Houdayer move. On the good side, given that the Houdayer move only depends on the graph topology and not the strengths of the interactions between variables, we need not care about the weights J_{ij} of the Ising problem.

Since we need to decide for each day on the number of hours for each entity, we define decision variables precisely for that. In particular, let $x_a^{A_k}$ be the variables which decide whether on day k the number of hours $a \in A_k$ is chosen, and similarly let the variables $x_b^{B_k}$ decide whether $b \in B_k$ is chosen. To have a simpler view of the objectives and constraints, let us denote the cache value on day k as $C(k)$ and the amount produced by entity A and B on day k by $A(k)$ and $B(k)$ respectively. With this simplified notation we can rewrite our requirements as:

- The total number of hours for entities A and B should be minimized, i.e., we should minimize $\sum_{k=1}^{n} A(k) + B(k)$.
- For every day and every entity, only one possible number of hours can be chosen among the available ones.
- The cache value must never exceed the limit Z. Note that we have the following equation which relates the cache value from one day to the next one: $C(k) = C(k-1) + A(k) - B(k)$.
- At the end of the n days, the number of items produced and processed $\sum_{k=1}^{n} B(k)$ should be equal to N. Since the values $B(k)$ depend on the cache throughout the days, we can

rewrite this in terms of entity A as how much it produced, minus the cache value on the last day (since it is not processed by B in time). So we want $(\sum_{k=1}^{n} A(k)) - C(n) = N$.

Using the binary variables $x_a^{A_k}$ and $x_b^{B_k}$ defined before, note that we have $A(k) = \sum_{a \in A_k} a x_a^{A_k}$ and $B(k) = \sum_{b \in B_k} b x_b^{B_k}$. The requirements translate to the following constrained minimization problem:

$$\begin{aligned}
\text{Minimize } & \sum_{k=1}^{n} \left(\sum_{a \in A_k} a x_a^{A_k} + \sum_{b \in B_k} b x_b^{B_k} \right) \qquad (4.2)\\
\text{Subject To: } & \sum_{a \in A_k} x_a^{A_k} = 1 \quad \forall k \in \{1, 2, \ldots, n\},\\
& \sum_{b \in B_k} x_b^{B_k} = 1 \quad \forall k \in \{1, 2, \ldots, n\},\\
& \sum_{k=1}^{m} \left(\sum_{a \in A_k} a x_a^{A_k} - \sum_{b \in B_k} b x_b^{B_k} \right) \leq Z \quad \forall m \in \{1, 2, \ldots, n\},\\
& \sum_{k=1}^{n} \sum_{a \in A_k} a x_a^{A_k} - \left(\sum_{k=1}^{n} \left(\sum_{a \in A_k} a x_a^{A_k} - \sum_{b \in B_k} b x_b^{B_k} \right) \right) = N.
\end{aligned}$$

This is thus a binary constrained linear program formulation of the problem, but recall that all we used so far is applicable to Ising type of problems (or equivalently QUBO problems as mentioned in section 2.1.1 since we are dealing with binary decision variables). In next section, we describe how one can convert the various constraints into quadratic polynomials in order to have a proper QUBO formulation.

4.3.3 Converting the Problem to QUBO Formulation

Essentially, the entire formulation given previously should be turned into a quadratic polynomial which is able to represent at the same time the minimization objective together with the various constraints to be respected. The way we do so is similar to the method employed in [45].

A simple way to deal with equality constraints is to introduce quadratic penalties (see chapter 5 in [46]). Consider the simple minimization problem

$$\begin{aligned}\text{Minimize } & \sum_{i=1}^{n} c_i x_i \\ \text{Subject To: } & \sum_{i=1}^{n} b_i x_i = B.\end{aligned}$$

One can turn the constraint $\sum_{i=1}^{n} b_i x_i = B$ into the penalty term $\left(\sum_{i=1}^{n} b_i x_i - B\right)^2$ giving rise to the new minimization problem

$$\text{Minimize } \sum_{i=1}^{n} c_i x_i + \lambda \left(\sum_{i=1}^{n} b_i x_i - B\right)^2,$$

where $\lambda > 0$ is a parameter indicating how stringent the penalty is.

With this, the set of optimal feasible solutions (if they exist) is identical to the one of the constrained version of the problem, since any feasible solutions incurs a penalty of 0 in the second version. Moreover, since this is a quadratic penalty term only involving linear combinations of the decision variables, the overall function to minimize is still a quadratic polynomial.

Converting all equality constraints to penalty terms using the same penalty $\lambda > 0$, we can turn 4.2 to

$$\begin{aligned}\text{Minimize } & \sum_{k=1}^{n} \left[\left(\sum_{a \in A_k} a x_a^{A_k} + \sum_{b \in B_k} b x_b^{B_k} \right) + \lambda \left(\sum_{a \in A_k} x_a^{A_k} - 1 \right)^2 + \lambda \left(\sum_{b \in B_k} x_b^{B_k} - 1 \right)^2 \right] \\ & + \lambda \left(\sum_{k=1}^{n} \sum_{a \in A_k} a x_a^{A_k} - \left(\sum_{k=1}^{n} \left(\sum_{a \in A_k} a x_a^{A_k} - \sum_{b \in B_k} b x_b^{B_k} \right) \right) - N \right)^2 \\ \text{Subject To: } & \sum_{k=1}^{m} \left(\sum_{a \in A_k} a x_a^{A_k} - \sum_{b \in B_k} b x_b^{B_k} \right) \leq Z \quad \forall m \in \{1, 2, \ldots, n\}.\end{aligned} \tag{4.3}$$

Already at this intermediary step, there is an issue that makes the use of Houdayer moves completely useless, which is due to the last penalty term involving the volume. Indeed, if one expands the what is inside the squared term, there is $\left(\sum_{k=1}^{n} \left(\sum_{a \in A_k} a x_a^{A_k} - \sum_{b \in B_k} b x_b^{B_k}\right)\right)^2$ which appears, essentially pairing all the binary variables. This means that the induced topology is that of a complete graph, making it impossible for the Houdayer move to be successful (see 3.4.4). Hence, we should find a way to reduce the graph connectivity, or the interactions between variables, without changing the problem itself.

After using multiple tricks which we cannot reveal here because of confidentiality issues, we can find a way to break the connectivity of the graph and formulate a new version of the problem which induces a much less dense graph connectivity, allowing the use of Houdayer moves. Note also that in the previous formulation, the minimization problem is still a constrained one because of the inequality constraints for the cache level on each day. We thus need to turn these inequalities into quadratic penalties, and to do so we just need to transform them into equality constraints with the help of "slack" variables (see chapter 4.1 in [46]).

After this conversion process, even though the new formulation might be complicated, it is in the form of a QUBO formulation, and if one expands everything, the minimization is done over a function of the type $Q(\boldsymbol{x}) = \sum_{i,j} Q_{ij} x_i x_j + \sum_i x_i$ as needed, where $\boldsymbol{x} \in \{0,1\}^d$ is a vector collecting all d binary variables in the formulation. From there, it is easy to retrieve the interactions (which correspond to edges in the underlying graph) and use the Houdayer move. As last technical detail, the Houdayer procedure only works in the Ising model (so variables in $\{-1,1\}$) which is not the case here so we should ensure that the right topology is being used. In fact, we know from proof of lemma 2.1.1 that one can apply a linear map to go from Ising to QUBO and vice-versa. Even though this means the weights of the interactions change, the interactions themselves in the two models remain between the same variables, so that the graph is identical when not considering edge weights.

4.4 A Modified Topology

From previous sections, the sequential scheduling problem can be seen as minimizing the Hamiltonian of an Ising system, thus inducing a certain graph topology. After converting the graph topology which was first a complete one into a more disconnected one, we ask the following question: with such a connectivity, how good can we hope the Houdayer to perform? In order to test this, we complete the following small experiment: taking the same instance but making the number of days vary, we each time pick 100 random pairs of configurations and see whether the Houdayer move is successful in this case. This is repeated 100 times for each number of days, so that the results can be visualized in the form of boxplots. With this, we aim know more at least in the random case about how often the move fails, and if the number of nodes plays a role in it.

Figure 4.13 shows at least promising results, considering that the ratio of successful Houdayer move can be strictly greater than 0 for every instance. An interesting detail to distinguish is that as the instances get more complex (larger number of days), it induces graphs with a larger number of nodes, seemingly giving higher and higher odds of success for the Houdayer move. Notably, for instances with 40 days or more, the success ratio can be larger than 0.1. In the next section, experiments are conducted this time using actual feasible solutions in order to

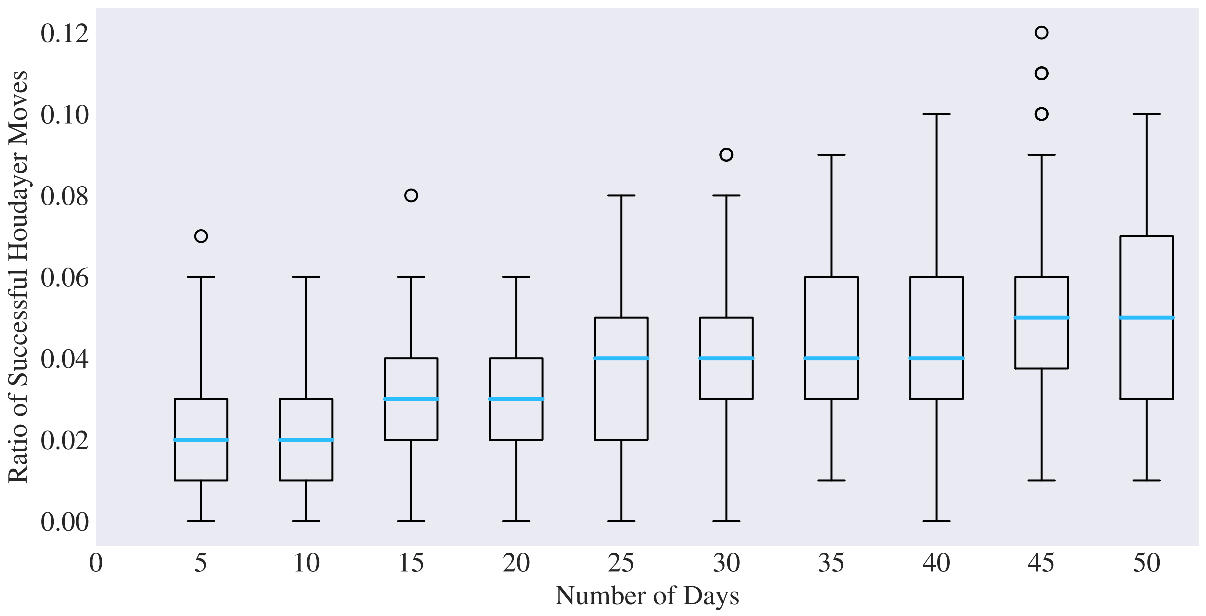


Figure 4.13 – Multiple boxplots showing the ratio of successful Houdayer moves when taking random pairs on various instances of the sequential scheduling problem. The number of days is slowly increased from 5 to 50 days.

create more of them using the techniques from section 4.2.

4.5 Numerical Results

We present in this section how the algorithms developed in this chapter can be applied in the context of the sequential scheduling problem. Experiments are performed in order to understand how such techniques can help when solving a hard optimization problem.

For all experiments, the instances chosen have the following specificities:

- The number of days n varies from one instance to the other.
- The number of items output is equal to $N = 8n$, but need not be precisely equal to that, and can fall in a range $[(1-\delta)N, (1+\delta)N]$, as mentioned in section 4.3.1.

4.5.1 Generating Feasible Solutions

A first step is to use the idea that with low energy solutions, Houdayer-like moves give the possibility to generate more of such solutions. Indeed, we know from section 4.2.2 that in theory this can work well, but what about the topology induced by the sequential scheduling problem? In order to explore this, we perform one iteration of the algorithm presented in section 4.2.1 when starting with a pool of 50 feasible solutions of the problem. Since there are 50 solutions, there are $\binom{50}{2} = 1225$ unique pairs to be used to generate new ones using a Houdayer-like move. We consider instances from 10 to 90 days, by steps of 10, and for every instance the value for relaxing the range of number of items produced takes values such that $n\delta \in \{5, 10, 15, 20\}$.

Figure 4.14 depicts various results about how often and how many new solutions are generated, and whether they are feasible. For each day, we compute these values for all for instances (varying the δ value), and plotting results as boxplots. The leftmost plot highlights the most important to us: it is possible to generate new solutions for the sequential scheduling problem. This also appear to once again show that an increased number of days (and thus of nodes) yields better and better generation results as observed in section 4.4. But what about the feasibility of the solution found after a successful move?

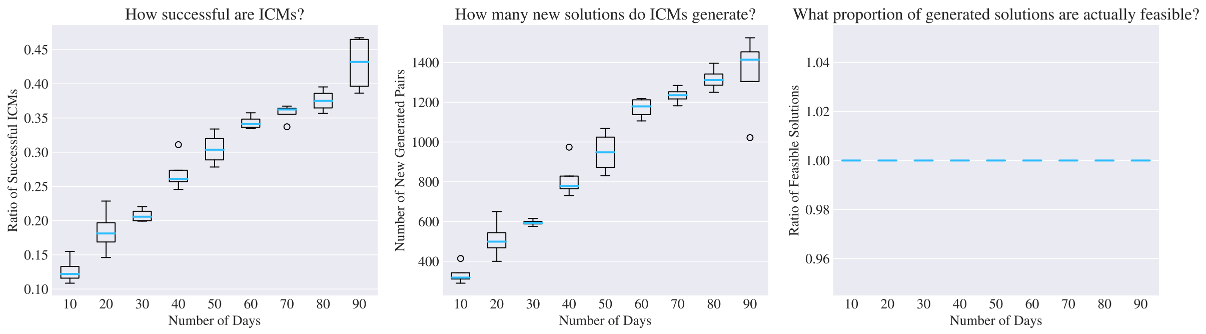
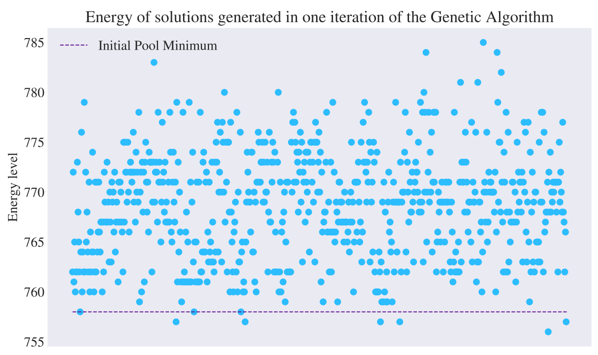


Figure 4.14 – Various results about how good Houdayer moves can be in the context of the sequential scheduling problem. From left to right, one can visualize the ratio of successful Houdayer moves, the number of new solutions produced, and finally the ratio of solutions that are actually feasible.

Recall from our QUBO encoding of the problem that if we take a very large coefficient λ for the penalty (essentially infinite), then we expect that whenever a Houdayer-like move is successful on a pair of feasible solutions, the resulting pairs are all feasible too. This is actually what can be seen on the rightmost plot, which shows that whenever new pairs are found, they are always feasible.

In figure 4.15, the costs of generated solutions are shown for two instances with different number of days, namely 50 and 80 days. There are two interesting things happening here: first, the algorithm is able to generate new feasible solutions which is already a big step. Indeed, for a very specific set of pairs which is given as initial pool, and not just some random configurations, it can lead to many successful Houdayer moves. Not only that, one can notice a variety of different energy levels in the configurations that are generated, and most interestingly it can be seen that some of the new solutions have lower energy than any of those in the initial pool. This in particular motivates the use of some kind of heuristic (which in our case turns out to be a genetic algorithm) in order to generate solutions with increasingly lower energy values.

Comparing the two plots in figure 4.15, it seems that the more complex the sequential scheduling instance is, the more successful the Houdayer moves can be since more feasible solutions are



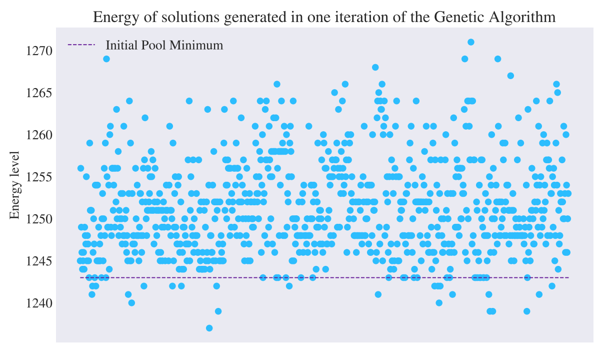


Figure 4.15 – Visualization of the feasible solutions generated and their costs when using the algorithm from section 4.2.1 for one iteration. Each point corresponds to a specific solution, and a dashed purple line indicates the lowest level of energy present in the initial pool. Left plot shows results on an instance on 50 days and with $\delta = 0.4$ while right plot is for an instance on 80 days and with $\delta = 0.4$.

created. Also, more of them end up having a lower energy value than the smallest one in the initial set of solutions. This can be also seen through the histograms in figure 4.16, where we can essentially visualize the distribution of the costs of the generated solutions when using different strategies for generation.

As expected, using the extended Houdayer move yields more solutions, but in general they tend to have an energy value close to those from the initial pool. Specifically, the new costs tend to more or less follow the same distribution as the one of the solutions in the initial pool. This maybe suggests that the variety of solutions produced by the algorithm depends on those given as input.

4.5.2 Improving Optimization with the Genetic Algorithm

The results in the previous section show that even for a constrained problem where getting feasible solutions is non-trivial, we are able to "navigate" within a subspace of the feasible space, reachable by using the techniques developed. The next goal is then to not only explore this region, but to continue the optimization process in order to lower the energy of solutions, which is exactly the purpose of the genetic algorithm presented in section 4.2.3.

This time we consider an even bigger instance of the problem, one on 365 days and with $\delta = 0.4$. The genetic algorithm is run for 10 iterations, and fixes a limit of $k = 1000$ on the population size. At each iteration, it picks at most 1000 random pairs from all possible ones induced by the current population and computes all accessible pairs. Once all is generated, it keeps the $0.8k = 800$ solutions with lowest energy out of those newly generated, $0.1k = 100$ are chosen randomly from that same set (different from the ones selected in previous step), and $0.1k = 100$ are randomly chosen from the previous population. This choice comes from the fact that one

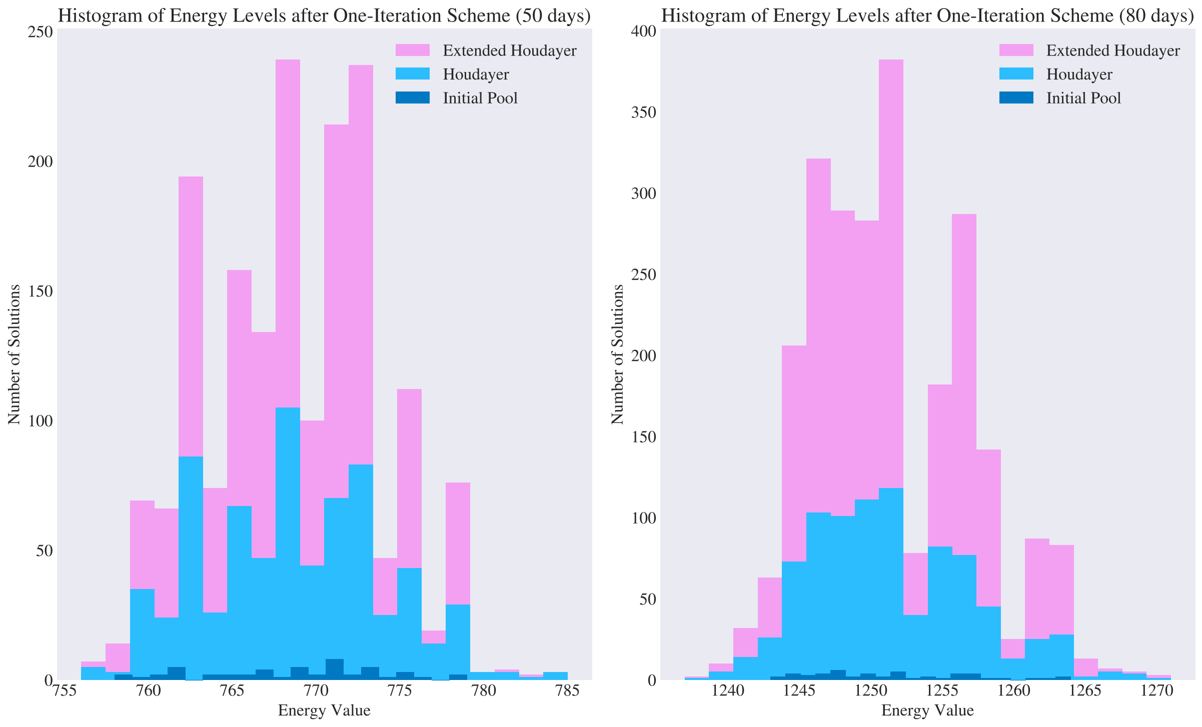


Figure 4.16 – Histograms of the costs of solutions on an 50-day instance (left plot) and an 80-day instance (right plot). More precisely, it shows superimposed histograms of the costs for the initial pool of solutions (dark blue), for the solutions created when using just the Houdayer move (light blue), and for the solutions created when using the extended Houdayer move (pink).

also wants to keep solutions from the past which are very good, also allow for new good ones to come, but also keep a fair share of randomness to not get stuck in local minima.

Looking at the various cost values of the generated solutions in figure 4.17, they are now clearly tilted towards lower energy values. In total, starting from a pool of 25 feasible solutions with minimum cost of 5595, more than 250’000 new unique feasible solutions are produced, many of which are lower than the minimum initial cost.

In terms of the variety of solutions generated, more can be said when as doing follows: after generating solutions, interpret them as vectors. For example, in a 2-day instance where entity A works 5 hours on day 1 and 8 hours on day 2, while entity B works 7 hours on day 1 and 4 hours on day 2, we collect this as vector $[5, 8, 7, 4]^\top$. From there, one can compare how close solutions are by computing some similarity measure between vectors such as the cosine similarity [47], defined as $\mathrm{SIM}(\boldsymbol{u}, \boldsymbol{v}) := \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|}$. A glance at figure 4.18 reveals that although constrained to a cosine similarity in the range $[0.9, 1]$, the solutions are still varied since there is a concentration around the similarity value 0.91.

Finally, we also look at how the optimization process evolves in terms of energy. In the left

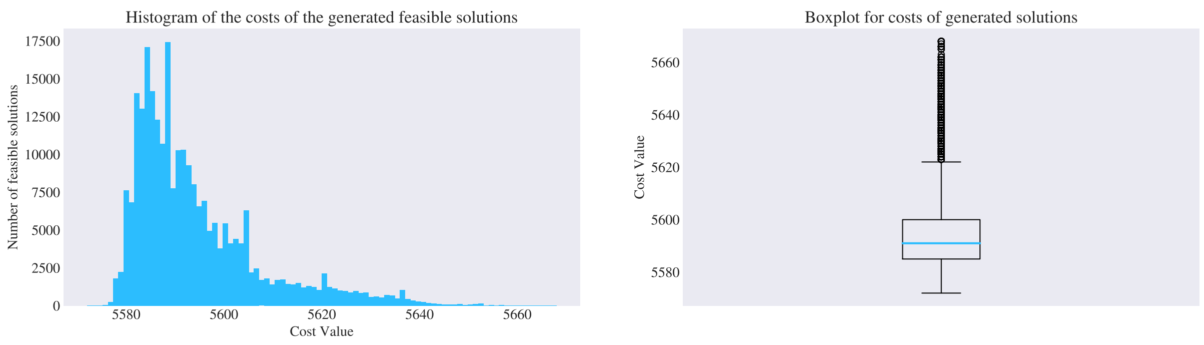


Figure 4.17 – Histogram (left) and boxplot (right) to visualize the distribution of the costs of the solutions generated when using the genetic algorithm.

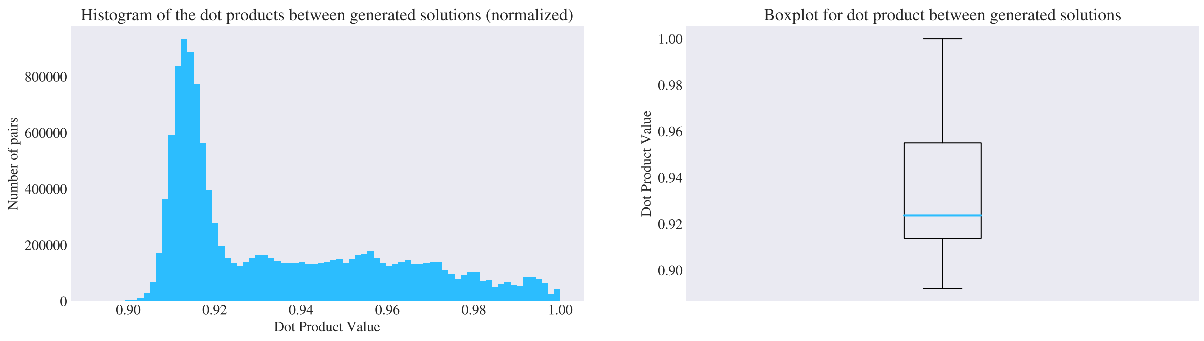


Figure 4.18 – Histogram (left) and boxplot (right) to visualize the distribution of the cosine similarity between the solutions generated when using the genetic algorithm.

plot of figure 4.19, for each iteration, the current best energy level is shown. A lower bound on the cost is also depicted and it is computed as follows: since for n days the number items to be produced is $8n$, but we can be off by a factor δ, the lowest target number is $8n(1-\delta)$. To get to that number, both entities would have to operate at least this number of hours (since in the end for each day only a couple of possible working hours are proposed). As such, the lower bound on the cost is given by $2 \cdot 8n(1-\delta) = 16n(1-\delta)$. On the right plot, one can see at each iteration of the genetic algorithm how many new solutions are generated. It is informative of something which is very typical that happens in such an algorithm: the first iterations are able to generate an increasing amount of feasible solutions up to a peak, after which the number goes down to zero. Note that depending on the tuning of the parameters of the algorithm, one can make it so that the generation is more or less uniform through the iterations.

4.5.3 Summary

Overall, these experiments highlight the following points:

- One can use Houdayer-like moves in order to generate low-energy solutions, which in our case correspond to feasible solutions. Within these solutions, the energy spans a range

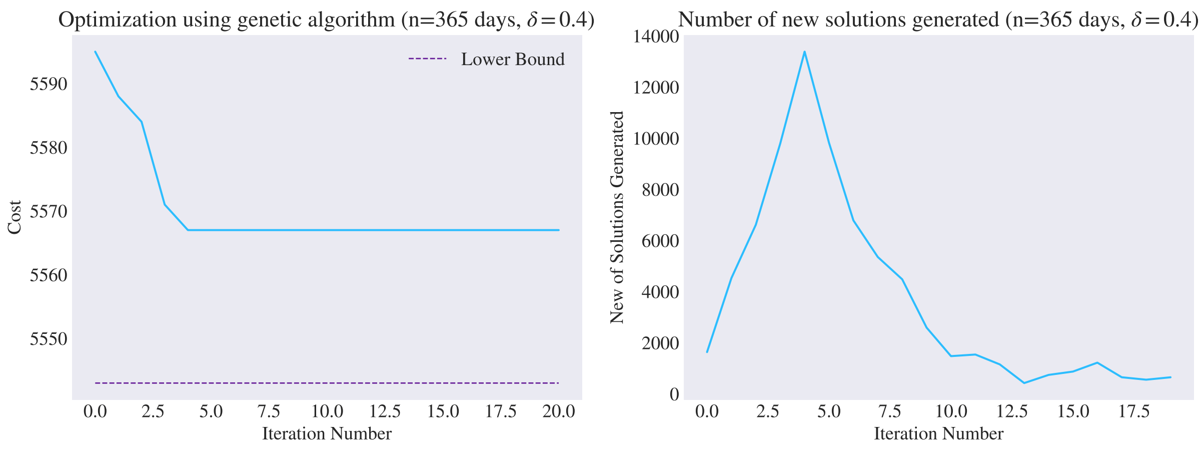


Figure 4.19

of values which at the same time shows how the method can attain solutions which are different from the starting ones, but the energy values nevertheless fall into a range which is not too far from those of the starting pool. Thus it is believed that the starting set is important as it will partially determine the variety of solutions generated.

- Once one knows how to navigate part of the feasible region through Houdayer moves, attempting to continue the optimization process is made possible by using the genetic algorithm. Here, the encouraging results suggest that lower and lower solutions can be found up to a certain point. Indeed it is always good to keep in mind that since the (extended) Houdayer is not ergodic, it is not possible to explore the entire feasible region with this algorithm. Nonetheless, it gives access to a variety of solutions at neighboring levels of energy which can help escape local minima.

- For the simple version of the sequential scheduling problem considered here, we were in fact able to reach a solution whose energy corresponds to the lower bound by using simulated annealing. However, this may no longer be true in more elaborate versions of the problem, for example with multiple caches and more complex layouts. In such a case, the simulated annealing procedure might reach the feasible region but fail to reach the ground states, which is where the genetic algorithm developed would then be used.

5 Experimental and Theoretical Discussion of Mixing Time

So far, we developed various schemes stemming from the idea of the Houdayer move and tested their efficiency through multiple experiments. From these, it seems relatively clear that using the (extended) Houdayer move improves convergence to equilibrium compared to a simple single-flip strategy. In this chapter, we attempt to study these behaviors more precisely and explore the theoretical aspect of convergence using tools from Markov chains theory.

5.1 Markov Chains and Mixing Time

Maybe one of the most useful quantities one can get about Markov chain is about how fast the process converges to the stationary distribution. This is captured by what is called the mixing time, and it has been extensively researched for multiple types of Markov chains related to the Ising model [20]. In particular, the mixing time of slightly different versions of the heat bath dynamics have been studied, where the Metropolis acceptance scheme is replaced by the Glauber acceptance one [21]–[23]. However in these, only the ferromagnetic case is considered, and the behavior in the general spin-glass model is yet unknown. For cluster Monte-Carlo moves, some advances have been made in studying the Swendsen-Wang algorithm that we described in section 3.3.1, in particular comparing it to the heat-bath dynamics [24], [25]. Here, we keep our focus on the single-flip strategy with Metropolis acceptance, and consider what can happen once we combine it with the (extended) Houdayer move. As we will see, this endeavor entails inspecting the spectral structure of the transition matrix of the various Markov chains. To begin, we define some basic notions such as the distance between probability distributions.

Definition 5.1.1 (Total Variation Distance)**.** *The total variation distance between two probability distributions μ and ν on a discrete set S is defined as*

$$\|\mu - \nu\|_{TV} = \frac{1}{2} \sum_{x \in S} |\mu(x) - \nu(x)|.$$

Definition 5.1.2 (Mixing Time)**.** *Let P be the transition matrix of an ergodic Markov chain with limiting and stationary distribution π. Denoting by δ_x the distribution which starts on state $x \in S$ (i.e., $\delta_x(y) = \mathbb{1}\{x = y\} \, \forall y \in S$), define*

$$c(n) = \max_{x \in S} \|\delta_x P^n - \pi\|_{TV}, \tag{5.1}$$

the maximum total variation distance between the distribution after n time steps and the stationary distribution, starting from any state $x \in S$. With this, one can define the mixing time as

$$t(\epsilon) = \inf\{n | c(n) \leq \epsilon\}.$$

This is the minimum number of time steps required so that the distance $c(n)$ is lower than some specified ϵ.

There is then a very interesting upper bound on the total variation distance in equation 5.1 as the number of steps n grows. It can be found in the proof of theorem 12.4 in [20], and we first define some important quantities.

Definition 5.1.3. *Let P be the transition matrix of a reversible and ergodic Markov chain. Then, letting $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{|S|}$ be the eigenvalues of P, we can define two quantities. The first one is denoted by $\lambda_* := \max\{|\lambda| : \lambda \text{ is an eigenvalue of } P, \, |\lambda| \neq 1\}$. The second one, defined by $\gamma := 1 - \lambda_* \in [0, 1]$, is called the spectral gap.*

Theorem 5.1.1. *Let P be the transition matrix of a reversible and ergodic Markov chain, and let $\pi_{min} = \min_{x \in S} \pi_x$. Then, letting $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{|S|}$ be the eigenvalues of P, we have that $\lambda_1 = 1$ and $|\lambda_i| < 1 \, \forall i \in \{2, 3, \ldots, |S|\}$. This means we can rewrite $\lambda_* = \max\{\lambda_2, -\lambda_{|S|}\}$. Moreover, the following bound on total variation holds:*

$$\max_{x \in S} \|\delta_x P^n - \pi\|_{TV} \leq \frac{\lambda_*^n}{2\sqrt{\pi_{min}}} = \frac{(1-\gamma)^n}{2\sqrt{\pi_{min}}} \leq \frac{e^{-\gamma n}}{2\sqrt{\pi_{min}}}. \tag{5.2}$$

In fact, it can be slightly strengthened when considering all eigenvalues and not just λ_* (see lemma 12.2 in [20]).

This theorem essentially says something crucial about the convergence of Markov chains to equilibrium: it has a strong dependency on the eigenvalues of P, which motivates the ideas in this chapter. Since this is only an upper bound, it could be that the convergence is not entirely ruled by the spectral gap, but it is still interesting to study the behavior of Metropolis chains when we add Houdayer-like moves.

5.1.1 Computing Transition Matrices of Houdayer-like Moves

Here we consider the simplest version of the Houdayer move, that is we start with a pair of configurations and then perform a single-flip on each of the configurations independently, followed by a Houdayer move, in which case the result depends on both configurations.

Denoting by P_{SF} the matrix representing the transition matrix of the Metropolis scheme with the single-flip strategy, and by P_H the transition matrix for a Houdayer move, we wish to study the dynamics defined by doing a Houdayer move followed by a single flip. When studying the extended Houdayer move, we denote the corresponding transition matrix by P_{H_+}. Since the Metropolis scheme is applied on both configurations independently, the corresponding transition matrix when working on pairs is given by the Kronecker product $P_{SF} \otimes P_{SF}$. The overall procedure when adding the Houdayer move can be written by $P_H(P_{SF} \otimes P_{SF})$ (or $(P_{SF} \otimes P_{SF})P_H$ if the single-flip is done before the Houdayer move).

The first step before being able to analyze the Houdayer move is to find a way to compute the corresponding transition matrices. Note that for an Ising model with n spins, the corresponding space of configurations is given by $S = \{-1, 1\}^n$ so that there are 2^n configurations. Since the transition matrix captures all transitions from one element of the space to the other, it has 2^n rows and 2^n columns, inducing a total size of $2^n \cdot 2^n = 2^{2n}$. That is, considering 10 spins, the resulting matrix has about one million entries. When using the Houdayer move, we employ pairs of configurations, and thus work in the space $S \times S = \{-1, 1\}^n \times \{-1, 1\}^n$, which has total size 2^{2n}. The resulting matrix then has 2^{4n} entries, so that for a very simple system with 5 spins, the number of entries in the transition matrix is already about one million. This restricts how much can be done through numerical simulations since the memory required to store transition matrices grows extremely fast with respect to the number of spins n, but we can nonetheless still determine what those matrices look like for very simple instances.

To compute the transition matrices, we do the following simple scheme: we go through every possible transition and compute the corresponding probability numerically. For example, to compute P_{SF} we go through all possible transitions from a configuration $\boldsymbol{\sigma}$ to configuration $\boldsymbol{\sigma}'$, and compute the probability according to the Metropolis procedure from section 3.1.1. To then compute the behavior of the single-flip strategy on independent pairs, we just need to compute $P_{SF} \otimes P_{SF}$ as mentioned earlier. For the Houdayer transition matrix P_H, we have to compute all transitions from a pair $(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})$ to a pair $(\boldsymbol{\sigma}'^{(1)}, \boldsymbol{\sigma}'^{(2)})$, which is done by actually computing the local overlap as explained in section 4.1.1 and retrieving the corresponding probabilities to reach each attainable pair depending on the Houdayer clusters arising from $(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)})$.

Once these are determined, we can use tools to numerically compute the values of interest like the eigenvalues and the spectral gap.

5.2 Spectral Gaps and Eigenvalues

A detail to notice first is that when studying the spectrum of composition of two matrices, the order in which they are applied does not matter. In our case, this means that applying the Houdayer move before or after the single-flip does not change the eigenvalues. This comes from the following:

Lemma 5.2.1. *Let $A, B \in \mathbb{R}^{n\times n}$ be two matrices. Then AB and BA have the same eigenvalues.*

Proof. Let v be an eigenvector of AB with corresponding non-zero eigenvalue $\lambda > 0$. Then we have

$$\lambda Bv = B(AB)v = (BA)Bv,$$

which means that Bv is an eigenvector of BA with the same eigenvalue. For the case where 0 is an eigenvalue of AB, we then have that

$$0 = \det(AB) = \det(A)\det(B) = \det(BA),$$

implying that zero is also an eigenvalue of BA. □

5.2.1 Results for the Houdayer Move

One first interesting question to ask is, since the addition of Houdayer moves seems to improve convergence, is this reflected in the spectral gap? As a first experiment, we create many instances of Ising spin-glass models (with bonds distributed as $J_{ij} \sim \mathcal{N}(0,1)$) and compute both the transition matrix when doing a single-flip procedure on pairs (that is $P_{SF} \otimes P_{SF}$) and the one when adding the Houdayer move (that is $P_H(P_{SF} \otimes P_{SF})$), and compare their spectral gap. That is, denoting the spectral gap of a transition matrix P as $\gamma(P)$, we compute $\gamma(P_H(P_{SF} \otimes P_{SF})) - \gamma(P_{SF} \otimes P_{SF})$.

As graph topologies, we consider simple examples on 4 spins with square lattice and Erdös-Rényi graphs with parameter $p = 0.4, 0.5$ and 0.6. For each, 500 instances with random bonds are generated and the transition matrices are computed at inverse temperature $\beta = 0.1$. The

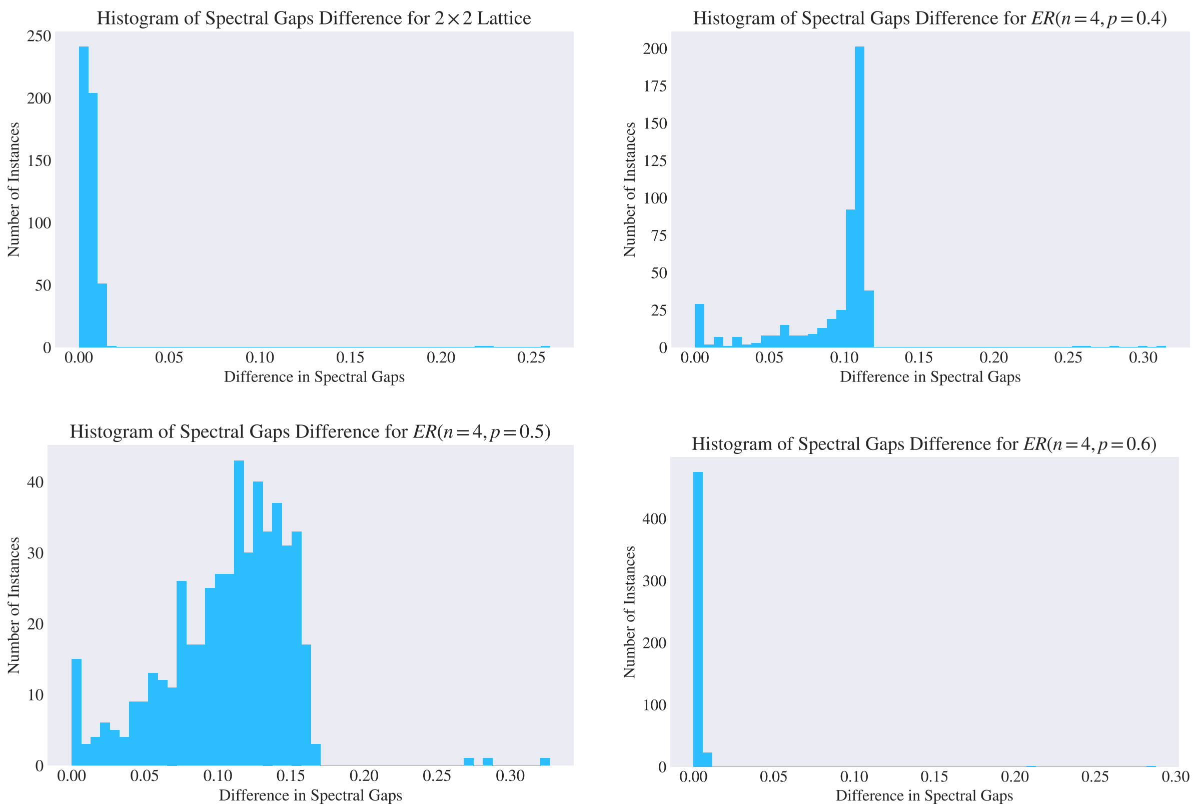


Figure 5.1 – Histograms showing spectral gap differences, when using the Houdayer move, and when just employing the single-flip dynamics.

spectral gaps are then determined and the difference is computed, yielding results in figure 5.1.

The histograms seem to highlight a pretty clear observation: the spectral gap of the transition matrix when using a Houdayer move is always greater or equal than the transition matrix of just single-flips since all the differences are positive. It is then interesting to note the different patterns of histograms depending on the topology chosen. For example for the lattice, the differences are concentrated in the range [0, 0.025], essentially saying that spectral gaps are quite close for all instances. But this is not the case when considering random graph topologies, where we see that there is much more spread in the spectral gaps differences.

We push the experimentation further to see whether the inverse temperature has a role to play in it. For that, we instantiate some Ising spin-glass systems with number of spins up to 8 and compute spectral gaps for many different inverse temperatures in order to discover whether the addition of the Houdayer move gives a higher spectral gap at all temperatures. The plots in figure 5.2 depict the resulting spectral gaps, from which we can comment the following:

- The spectral gap when using the Houdayer move is always larger than when it is not used.
- After a certain temperature (which in the figure seems to be around $\beta = 0.2$ in our

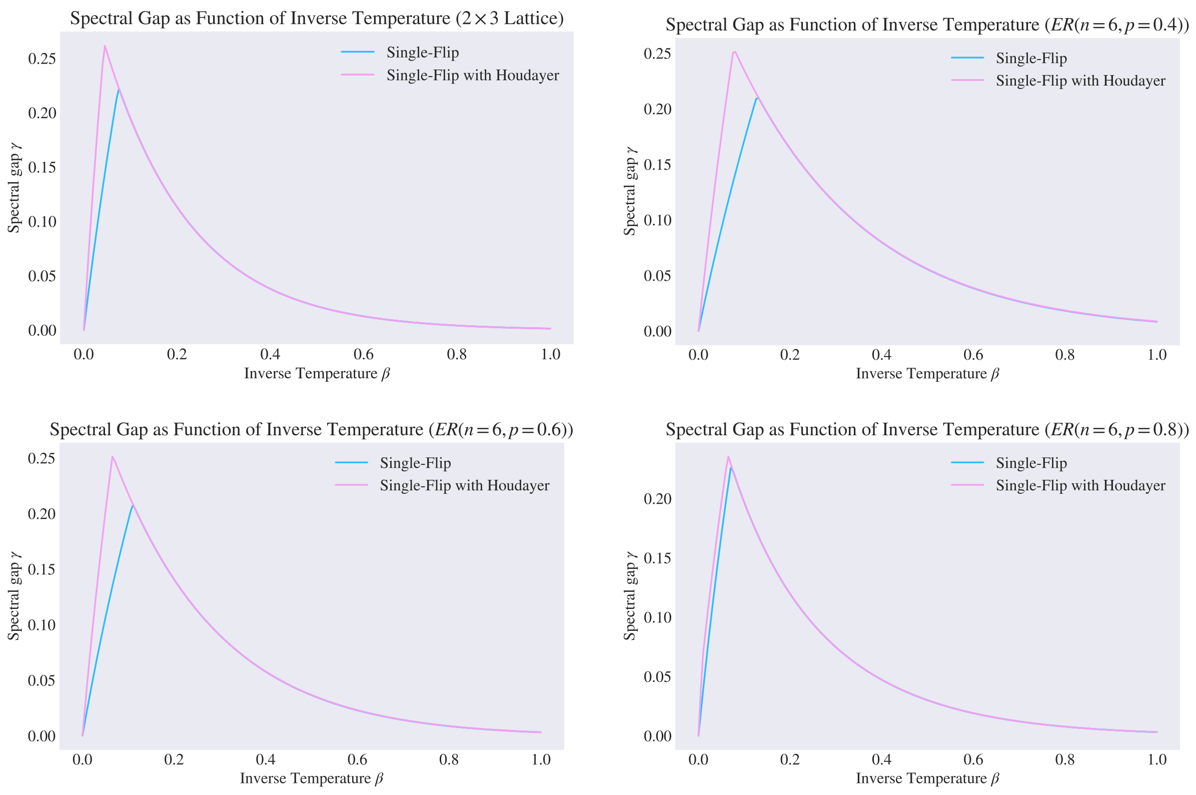


Figure 5.2 – Plots of spectral gap for both single-flip strategy and with the addition of the Houdayer move as a function of β. Each plot corresponds to a different graph topology.

examples), both spectral gaps seem to be equal.

- There is a certain inverse temperature β at which each procedure reaches a maximum spectral gap.

From even more extensive numerical computations which seem to always confirm the results seen ealier, we formulate a conjecture relating the spectral gap of the dynamics with the Houdayer move, and the dynamics without it.

Conjecture 5.2.1.1. *Let $P_{SF} \otimes P_{SF} \in \mathbb{R}^{S\times S}$ be the transition matrix of the Metropolis procedure with single-flip on pairs of spin configurations, and $P_H(P_{SF} \otimes P_{SF}) \in \mathbb{R}^{S\times S}$ the transition matrix of the same procedure, but augmented with the Houdayer move. Then, we have*

$$\gamma(P_H(P_{SF} \otimes P_{SF})) \geq C \cdot \gamma(P_{SF} \otimes P_{SF}), \tag{5.3}$$

for some constant $C \geq 1$.

5.2.2 Some Comments About the Single-Flip Dynamics

For every experiment performed where the spectral gap is visualized as a function of the inverse temperature β, there is always one thing one can observe: the spectral gap increases in value up until it reaches a maximum, after which it decreases. Moreover around the maximum, it seems like the value of the spectral gap changes abruptly. This is in fact due to the fact that $\lambda_* = \max\{\lambda_2, \lambda_{|S|}\}$ (see theorem 5.1.1), or equivalently $\gamma = 1 - \lambda_* = \min\{1 - \lambda_2, 1 + \lambda_{|S|}\}$. Through more extensive experiments one can actually see that for values of β after the maximum spectral gap is reached, it is always the case that $\lambda_* = |\lambda_2|$, while for values of β lower than when the maximum spectral gap is attained, we have $\lambda_* = |\lambda_{|S|}|$, which explains the behavior observed.

Moreover, these observations lead us to believe that both λ_2 and $\lambda_{|S|}$ are increasing as the value of β increases. This has already been proved for particular setups of the Ising model [48]. Indeed, we have the two cases:

1. When $\lambda_* = \lambda_2$ is chosen, the spectral gap $\gamma = 1 - \lambda_2$ in this case decreases as β increases, which means λ_2 increases.
2. When $\lambda_* = -\lambda_{|S|}$ is chosen, the spectral gap $\gamma = 1 + \lambda_{|S|}$ in this case increases as β increases, which means $\lambda_{|S|}$ increases.

Thus we formulate the following conjecture:

Conjecture 5.2.1.2. *Consider any Ising spin-glass system, and suppose that one uses the Metropolis dynamics with single-flip to simulate it. Let λ_i, $1 \leq i \leq |S|$ denote the ordered eigenvalues of the transition matrix of the induced Markov chain. Then, looking at these values as a function of the inverse temperature β, we have that*

$$\frac{d\lambda_2}{d\beta} \geq 0 \quad \textit{and} \quad \frac{d\lambda_{|S|}}{d\beta} \geq 0.$$

5.2.3 Results for the Extended Houdayer Move

We now shift our focus to the particular case of the extended Houdayer move, when we make the uniform choice when selecting combinations of Houdayer clusters (see section 4.1.1). We do so since in this case the transition matrix for the move has a particular form which can make it easier to analyze the whole dynamics when combined with the single-flip procedure.

First, the experiment where we compute the spectral gap as a function of the temperature for different topologies is repeated, but this time including the results for the extended Houdayer move. One can observe in figure 5.3 three types of outcome for different instances of the

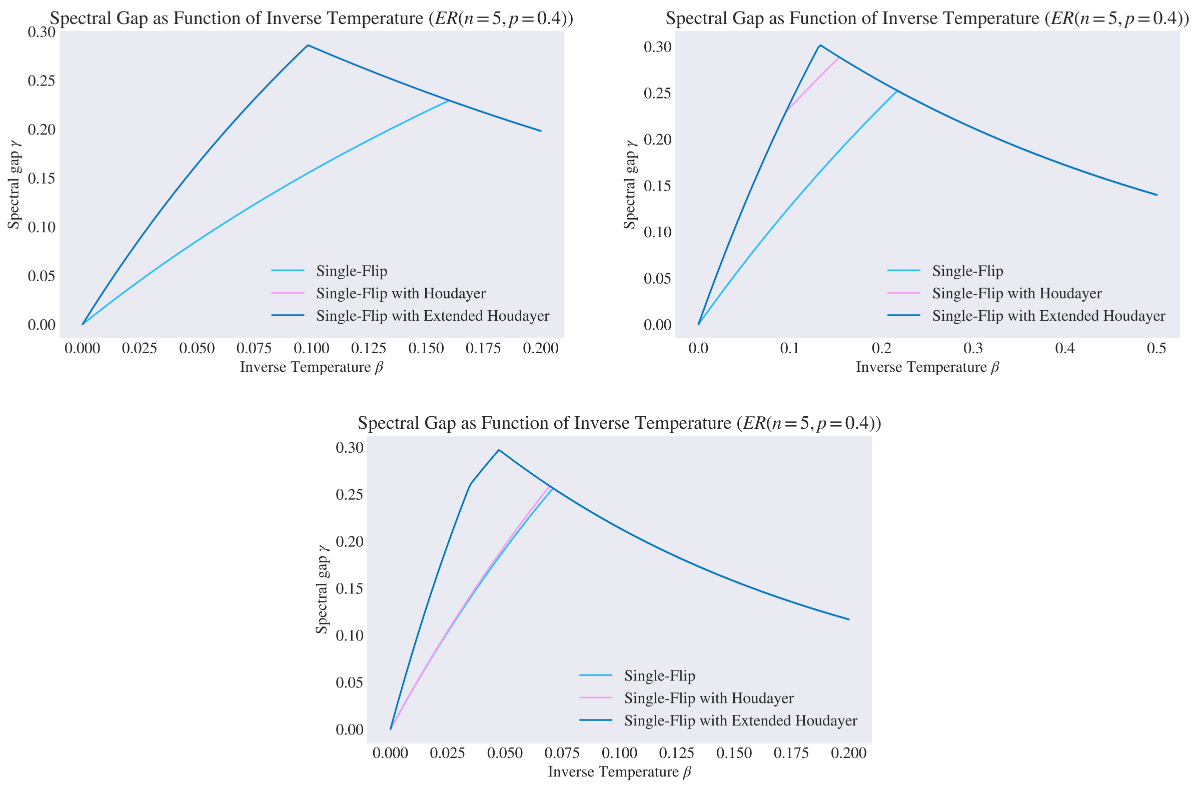


Figure 5.3 – Plots of spectral gap for both single-flip strategy and with the addition of the Houdayer move and its extended version as a function of β. Each plot corresponds to a different random graph.

$ER(n = 5, p = 0.4)$ random graph. Either both the Houdayer move and the extended one yield the same spectral gap, either there is a slight difference between the two where for a certain range of temperature, the extended version can reach a higher spectral gap. The third possibility is that the extended version yields a spectral gap which is quite larger from that of the original Houdayer, but only for low values of β. Indeed, in any case, we note that for β large enough, all spectral gaps seems to be identical.

While running the experiments, we also noted the following: the more connected the graph is, the bigger the difference between the spectral gap of the Houdayer move versus its extended version. However, once full connectivity is reached, we know that all methods are equivalent so that their spectral gaps coincide. In general, these results seem to suggest that the spectral gap for the extended Houdayer move is larger than the other methods. However, looking more closely at it, figure 5.4 reveals that actually the original Houdayer move can have a very slightly larger spectral gap than the extended version once β gets large enough. However a similar conjecture as the one in 5.2.1.1 can be stated but using P_{H_+} instead of P_H.

Next, we prove some results about the structure of the transition matrix P_{H_+} of the extended Houdayer move.

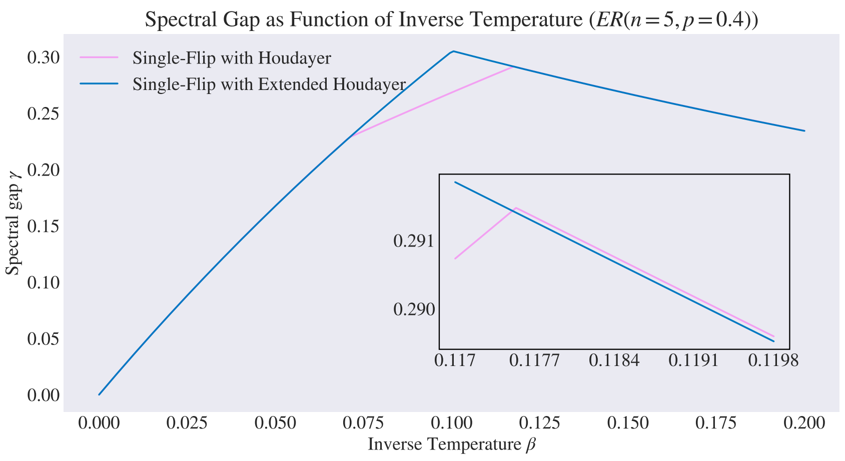


Figure 5.4 – Plots of spectral gap for both single-flip strategy and with the addition of the Houdayer move as a function of β. The smaller plot is a zoomed in version where β is in the range $[0.117, 0.2]$.

Lemma 5.2.2. *Suppose we perform the extended Houdayer move on the state space $S \times S$, and let $\mathcal{C}$ denote the set of clusters which partitions the space into pairs that are accessible from one another (that is, every $C \in \mathcal{C}$ is a set of pairs). Let $P_{H_+} \in \mathbb{R}^{S \times S}$ be the matrix defined by the extended Houdayer move, where the choice of combinations of Houdayer clusters is done uniformly at random. Then P_{H_+} is a symmetric projection matrix with rank $|\mathcal{C}|$.*

Proof. We show that $P_{H_+}^2 = P_{H_+}$. First note that this version of the extended Houdayer move partitions the state space $S \times S$ into a set of clusters $\mathcal{C}$. Note that these clusters have nothing to do with Houdayer clusters. In this case, every cluster $C \in \mathcal{C}$ is a set of pairs of configurations which are all accessible from one another through the cluster move. Moreover, when given a certain pair $p \in C$, the probability to reach pair $q \in C$ is uniform, i.e. $\mathbb{P}(p \to q | p, q \in C) = \frac{1}{|C|}$. In general, for $p, q \in S \times S$, we can write using the law of total probability that

$$\begin{aligned}
(P_{H_+})_{p,q} &= \sum_{C \in \mathcal{C}} \mathbb{P}(p \to q | p, q \in C)\, \mathbb{P}\{p \in C, q \in C\} \\
&= \sum_{C \in \mathcal{C}} \mathbb{P}(p \to q | p, q \in C) \mathbb{1}\{p \in C, q \in C\} \\
&= \sum_{C \in \mathcal{C}} \mathbb{P}(p \to q | p, q \in C) \mathbb{1}\{p \in C\} \mathbb{1}\{q \in C\} \\
&= \sum_{C \in \mathcal{C}} \frac{\mathbb{1}\{p \in C\} \mathbb{1}\{q \in C\}}{|C|},
\end{aligned} \tag{5.4}$$

where in the second equality we use the fact that since p and q are fixed, the probability they belong to a certain cluster can be written using an indicator function. We can see from here that P_{H_+} is symmetric since $\left(P_{H_+}\right)_{p,q} = \left(P_{H_+}\right)_{q,p}$.

Moreover, we can compute

$$\begin{aligned}
\left(P_{H_+}^2\right)_{p,q} &= \sum_{r\in S\times S} \left(P_{H_+}\right)_{p,r} \left(P_{H_+}\right)_{r,q} \\
&= \sum_{r\in S\times S} \sum_{C_1,C_2\in\mathcal{C}} \frac{\mathbb{1}\{p\in C_1\}\mathbb{1}\{r\in C_1\}\mathbb{1}\{r\in C_2\}\mathbb{1}\{q\in C_2\}}{|C_1||C_2|} \\
&= \sum_{C_1,C_2\in\mathcal{C}} \frac{\mathbb{1}\{p\in C_1\}\mathbb{1}\{q\in C_2\}}{|C_1||C_2|} \sum_{r\in S\times S} \mathbb{1}\{r\in C_1\}\mathbb{1}\{r\in C_2\}. \qquad (5.5)
\end{aligned}$$

Focusing on $\sum_{r\in S\times S} \mathbb{1}\{r\in C_1\}\mathbb{1}\{r\in C_2\}$, we note that since $\mathcal{C}$ partitions the state space $S\times S$, a pair r has to belong to exactly one cluster. Thus we have

$$\begin{aligned}
\sum_{r\in S\times S} \mathbb{1}\{r\in C_1\}\mathbb{1}\{r\in C_2\} &= \mathbb{1}\{C_1 = C_2\} \sum_{r\in S\times S} \mathbb{1}\{r\in C_2\} \\
&= \mathbb{1}\{C_1 = C_2\}|C_2|.
\end{aligned}$$

Plugging this back into 5.5 yields

$$\begin{aligned}
\left(P_{H_+}^2\right)_{p,q} &= \sum_{C_1,C_2\in\mathcal{C}} \frac{\mathbb{1}\{p\in C_1\}\mathbb{1}\{q\in C_2\}}{|C_1|}\mathbb{1}\{C_1 = C_2\} \\
&= \sum_{C\in\mathcal{C}} \frac{\mathbb{1}\{p\in C\}\mathbb{1}\{q\in C\}}{|C|} \\
&= \left(P_{H_+}\right)_{p,q},
\end{aligned}$$

which concludes the proof that P_{H_+} is a projection matrix.

For the computation of the rank, we show that the dimension of the space spanned by the columns of P_{H_+} is equal to $|\mathcal{C}|$. From the transition matrix entries computed in equation 5.4, we deduce that for each $C\in\mathcal{C}$, all pairs $p\in C$ have identical column entries in the matrix. Indeed, comparing for example columns for two pairs p and $q\in C$, we have that for any $r\in S\times S$

$$
\begin{aligned}
(P_{H_+})_{r,p} &= \sum_{C' \in \mathcal{C}} \frac{\mathbb{1}\{r \in C'\}\mathbb{1}\{p \in C'\}}{|C'|} \\
&= \frac{1}{|C|} \\
&= \sum_{C' \in \mathcal{C}} \frac{\mathbb{1}\{r \in C'\}\mathbb{1}\{q \in C'\}}{|C'|} \\
&= (P_{H_+})_{r,q} \, .
\end{aligned}
$$

Moreover, one can easily check that column vectors corresponding to the clusters are orthogonal since for every entry, a non-zero value in one of the vector corresponds to a zero value in the other and vice-versa. Since the columns form a set of $|\mathcal{C}|$ orthogonal vectors, the rank of P_{H_+} is $|\mathcal{C}|$. $\square$

In fact, one can see the operation of the extended Houdayer move as follows: starting from a certain pair, we teleport (or get projected) equally among all accessible pairs.

Lemma 5.2.3. *Suppose we perform the extended Houdayer move on the state space $\mathcal{S} \times \mathcal{S}$, and let $\mathcal{C}$ denote the set of clusters which partitions the space into pairs that are accessible from one another. Then the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{|\mathcal{S}\times\mathcal{S}|}$ of P_{H_+} are in $\{0, 1\}$, such that $\lambda_1 = \lambda_2 = \cdots = \lambda_{|\mathcal{C}|} = 1$ and $\lambda_{|\mathcal{C}|+1} = \lambda_{|\mathcal{C}|+1} = \cdots = \lambda_{|S\times S|} = 0$. In fact, for each $C \in \mathcal{C}$, there is a corresponding eigenvector $\boldsymbol{v}_C$ with associated eigenvalue 1 which is given by $(\boldsymbol{v}_C)_p = \mathbb{1}\{p \in C\}$.*

Proof. Recall that a projection matrix only has two possible eigenvalues, 0 or 1. Since the rank of P_{H_+} is $|\mathcal{C}|$, it has $|\mathcal{C}|$ eigenvalues equal to 1. By the rank-nullity theorem, we deduce that there are $|S \times S| - |\mathcal{C}|$ eigenvalues with value 0.

For the eigenvectors, one can check that for all $C \in \mathcal{C}$ and all $p \in S \times S$, we have

$$\begin{aligned}
\left(P_{H_+}\mathbf{v}_C\right)_p &= \sum_{q\in S\times S} \left(P_{H_+}\right)_{p,q} \left(\mathbf{v}_C\right)_q \\
&= \sum_{q\in S\times S} \sum_{C'\in\mathcal{C}} \frac{\mathbb{1}\{p\in C'\}\mathbb{1}\{q\in C'\}}{|C'|} \mathbb{1}\{q\in C\} \\
&= \sum_{q\in S\times S} \frac{\mathbb{1}\{p\in C\}\mathbb{1}\{q\in C\}}{|C|} \\
&= \mathbb{1}\{p\in C\} \sum_{q\in S\times S} \frac{\mathbb{1}\{q\in C\}}{|C|} \\
&= \mathbb{1}\{p\in C\}.
\end{aligned}$$

Moreover, these eigenvectors form an orthogonal basis since for any two different $C_1, C_2 \in \mathcal{C}$, their dot-product is equal to

$$\begin{aligned}
\mathbf{v}_{C_1}^\top \mathbf{v}_{C_2} &= \sum_{p\in S\times S} \left(\mathbf{v}_{C_1}\right)_p \left(\mathbf{v}_{C_2}\right)_p \\
&= \sum_{p\in S\times S} \mathbb{1}\{p\in C_1\}\mathbb{1}\{p\in C_2\} \qquad (5.6) \\
&= 0.
\end{aligned}$$

Here, we used the fact that since a pair can only belong to one cluster, every element in the sum in equation 5.6 is 0. □

These are preliminary results about the structure of the transition matrices, but they could be used to prove further results that we mention in next section.

5.3 Possible Future Directions

From previous sections, it seems clear that there is some relationship between the resulting Markov chains when one uses the Houdayer move or not. In particular, it would be already a step in better understanding the inner workings of the Houdayer if one can prove why the spectral gaps are identical for large enough values of β when comparing the single-flip strategy versus the one with the addition of Houdayer-like moves. In this section, we provide some possible directions to follow in order to deepen the understanding of the spectral structure of the transition matrices studied.

- Concerning the single-flip dynamics itself, we highlighted the behavior of the spectral gap were it reaches a maximum which corresponds to when $|\lambda_2| = |\lambda_{|S|}|$. Here, there are multiple results one could go after. The first one is to try and prove that both these eigenvalues are increasing as a function of the inverse temperature β. That is, $\frac{d\lambda_2}{d\beta} \geq 0$ and $\frac{d\lambda_{|S|}}{d\beta} \geq 0$. The second interesting result to obtain would be to find at which temperature β the spectral reaches its maximum, which is equivalent to finding the inverse temperature at which $|\lambda_2| = |\lambda_{|S|}|$.

- For the Houdayer move, getting an understanding of why the spectral gap is identical to that when using only the single-flips for large enough β would be a first step. From there, it would maybe possible to even understand the behavior for low values of beta, which is when the spectral gaps start differing.
- In comparing the Houdayer move with its extended version, understanding why it is the case that the extended version can sometimes reach larger value of spectral gap could help highlight their differences (see figure 5.3).
- Finally, a deeper study of the general extended Houdayer move, which is when we allow to use any distribution for the combination of Houdayer clusters to choose after computing the local overlap (see section 4.1.1). Indeed, this could then motivate the use of particular distributions over others, showing that using the most uniform choice is not necessarily the best one.

6 Conclusion

6.1 Summary

In this work, we have studied in detail the Houdayer move which is a particular cluster Monte-Carlo update applied in the context of optimization of Ising models. This research has given the possibility to better simulate some Ising spin-glass systems, and to solve hard optimization problems that come from both academia and industry.

In chapter 2, the general Ising model is presented together with its relationship to the equivalent QUBO formulation. The concept of sampling is introduced, and we explain how it can be applied to solve optimization problems. Chapter 3 then builds on this knowledge and present the general Markov chain Monte-Carlo sampling method, simulated annealing, parallel tempering and cluster Monte-Carlo moves. Specifically, the Houdayer cluster update is reviewed in more depth and some of its limitations are highlighted. Through a series of numerical simulations, these various schemes are compared on ferromagnetic and spin-glass Ising systems.

After establishing the necessary concepts and techniques, an improved version of the Houdayer move is introduced in chapter 4, giving a more general way to perform cluster moves. We show how this adaptation ends up giving access to many new isoenergetic locations in the energy landscape compared to the original move, thus enabling to escape local minima even more easily.

Moreover, we employ this technique to improve the sampling of the ground state manifold of degenerate problems, and more generally the sampling of low-energy solutions. This shows to be particularly efficient in the experiments conducted, where starting from a handful of ground states, all configurations at the ground level can be found in most cases. For the specific case of constrained optimization problems, we present an adaptation of the scheme in order to efficiently sample feasible solutions of the problem.

Applied in the context of an industrial scheduling-type optimization problem, the results exhibit that a variety of feasible solutions can be efficiently generated even though this may depend on the set of solutions given as input. Because the generation is particularly successful, this paves the way for the creation of a genetic algorithm to carry out further optimization directly in the feasible space. Benchmarking it against simpler methods like simulated annealing turns out to indicate that for this simple version of the production planning problem, no improved sampling is actually required to find optimal solutions. This may however not be the case for more complex versions where standard heuristics might already have difficulties reaching the feasible region. In such a case, employing the genetic algorithm can drastically improve the optimization process even starting with only a couple of feasible solutions.

In chapter 5, we discuss the theoretical aspects behind the Houdayer cluster move, focusing primarily on the study of spectral gaps and eigenvalues of the corresponding transition matrices. We attempt to get a better understanding of why using the Houdayer move yields higher spectral gaps and thus a potentially faster mixing time. This also leads to exploring the structure of transition matrices for the extended Houdayer move, giving some directions for future research.

In summary, this project focuses on the problem of solving discrete optimization problems that are formulated as Ising spin-glass systems. In order to simulate such systems, multiple algorithms are reviewed and developed in order to improve the state-of-the-art methods. Although many encouraging results are obtained, there is room to both further refine and create new algorithms, but also to study the theory behind these complex dynamics.

6.2 Future Work

There are multiple directions one can take to push the work that has been done in this project further. There is indeed a myriad of experiments that can be performed to further understand how useful the extended Houdayer move presented in chapter 4 is. As only few theoretical results are known in the field, it would of course be crucial to dig deeper in that way. In particular, having a theoretical understanding of why the (extended) Houdayer procedure can be faster than other techniques could lead to design an "optimal" Houdayer move. Indeed, the framework presented in section 4.1.1 being as general as possible, it could be that the choice of distribution when choosing Houdayer clusters in the local overlap might influence the entire process.

We list below a few possible future directions.

- The theory behind cluster Mont-Carlo moves, especially the Houdayer move, is not yet well understood. In particular when taking the perspective of Markov chains theory, no clear convergence results are known and it is not known how the Houdayer move affects the convergence of the simple single-flip strategy once it is combined with it. More generally, understanding the transition matrix corresponding to the most generic Houdayer move (as described in section 4.1.1) is another important problem and can only lead to a better comprehension of the corresponding dynamics.

- Section 4.2 introduced how one could use Houdayer moves to generate more uncorrelated solutions given a couple of them, and we saw it could indeed work for a concrete problem from the industry. There exist many problems of such nature, that is they are highly degenerate and contain multiple optimal solutions but it is difficult to find all of them. An example is the k-SAT problem and its variants such as ALL-SAT or MAX-SAT (see chapter 1 in [49]). However such a problem has more than just pairwise interactions between variables, at most all variables could interact together. Supposing that the largest interaction sees M variables, the function to minimize becomes a multivariate polynomial of degree equal to M, and in such a case, the image of a graph as in the quadratic case becomes a hypergraph where an edge can now join up to M vertices. Could one generalize the Houdayer move to such a structure? If yes, this means one could teleport at the same energy level even for complex problems involving interactions between more than just two variables.

- Another way to see how the Houdayer move can be generalized is to not think in terms of the interactions between the variables, but focus more on what values the variables can take. So far, it was applied to the case of spins or binary variables but this heavily restricts the type of problems to represent. Moreover, the conversion of a generic problem to Ising or QUBO form is often cumbersome, and typically requires many auxiliary variables. A first step in generalizing the Houdayer move in this sense would be to consider a problem where variables can take a finite number of discrete values. A second step could be to further allow for general discrete variables. With such a procedure, many discrete optimization problems could potentially benefit from this kind of cluster Monte-Carlo move.

- In general, do there exist other Monte-Carlo moves in the same spirit as the Houdayer one which would allow to better explore the space of solutions? Indeed, there could be further research in that area since no specific efficient method exists to solve general spin-glass models. The various experiments demonstrated how the addition of the cluster algorithm improves the convergence, but it is likely that other similar approaches may yield even better results. A suggestion of similar move as the Houdayer one could be to use more than just two configurations when doing a move. Recall that in the basic move, clusters of spins are being swapped between configurations, but one could imagine a version with three or more configurations, where spins are exchanged in a certain way between configurations such that the overall energy remains constant. This would allow for a potentially higher number of new configurations to be generated when doing improved sampling as presented in section 4.2.

- Concerning the industry problem presented, many more complicated versions of it can be created. As an example, one could consider more complex chains with more entities that can share a same cache, forming a structure more complex than just a linear chain.

A Appendix

A.1 A Python Package for Optimizing General Ising Models

As part of this Master's project, a programming tool has been developed using the Python programming language [50] in order to test the algorithms presented in this report. After many iterations and improvements, the package thought of two smaller sub-packages:

A.1.1 Improving Sampling and Visualizing the Houdayer Move

The first sub-package was first developed to visualize and better understand the Houdayer move. It can for example accept any topology given as a graph, and one can visualize how the Houdayer move is performed. This is useful when working with particular problems and when one wants to understand whether Houdayer moves can be successful for the corresponding graph topology (see figure A.1).

Later on, all the algorithms for fair sampling were added, giving it the power to generate many new solutions given a few. In particular, a generic version of the genetic algorithm presented in section 4.2.3 is implemented. It is highly customizable and contains multiple parameters in order to give the user the freedom to adapt the algorithm to the problem at hand.

A.1.2 Solving General Ising Spin-Glass Systems

The second sub-package's goal is to solve general Ising spin-glass systems by giving the user plenty of freedom when choosing which algorithms and parameters to choose.

First, the user can easily define an instance of an Ising model, either by inputting the different interactions with corresponding weights manually, or even give a weighted graph as input. There is also the possibility to generate random instances, or well-known ones like the square lattice. Since many optimization problems are formulated in QUBO format, one can easily

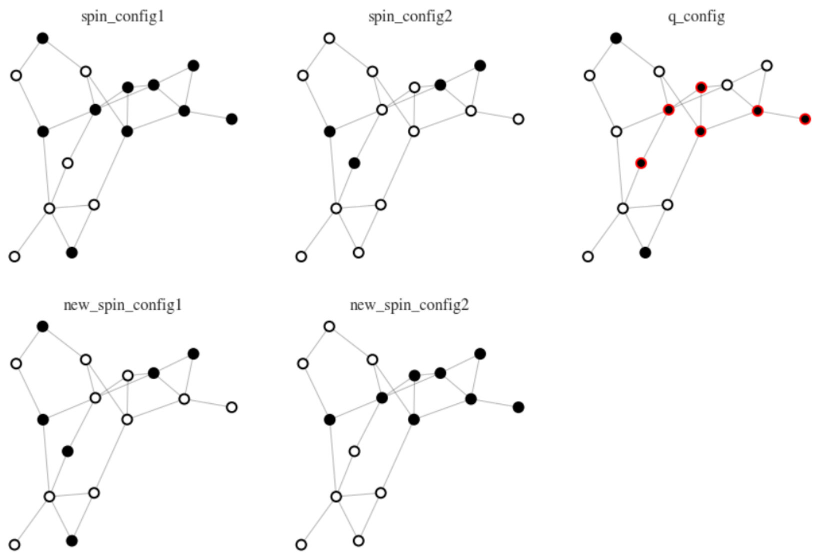


Figure A.1 – An example of visualization if how the Houdayer move works on a certain graph topology using the Python package.

convert from Ising to QUBO and vice-versa in order to easily optimize all kinds of problems.

As a solver, it uses the Metropolis algorithm at its core, and allows the user to specify the desired Monte-Carlo move, which can be the single-flip move for example. There is of course the (extended) Houdayer move available, which can be easily combined with any other move. Other details can be chosen such as the acceptance scheme (Metropolis or Glauber), and even which main optimization method. Currently, both parallel tempering and simulated annealing are implemented, but one could also extend to include algorithms like population annealing [51].

The end goal is to make this code open-source at some point, in which case it will be accessible in a GIT repository at *https : //github.com/Adirlou/*.

A.2 MCMC Methods Comparison

Some of the plots from experiments run in chapter 3 and 4 are presented here.

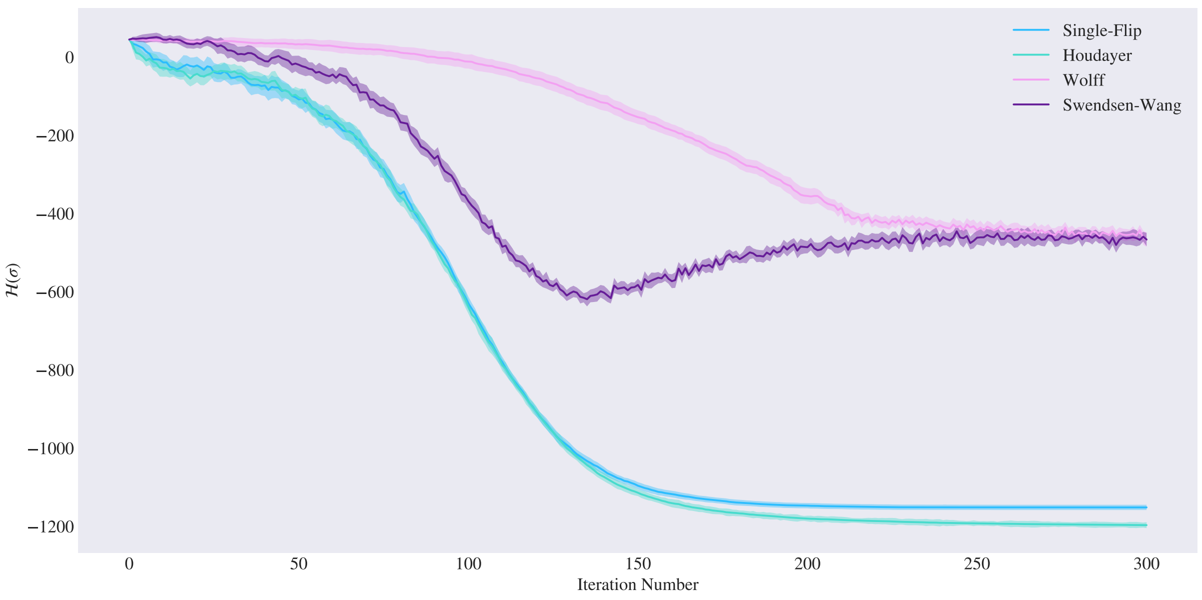


Figure A.2 – Result of SA for various moves on a 32×32 square lattice graph for the spin-glass model.

.

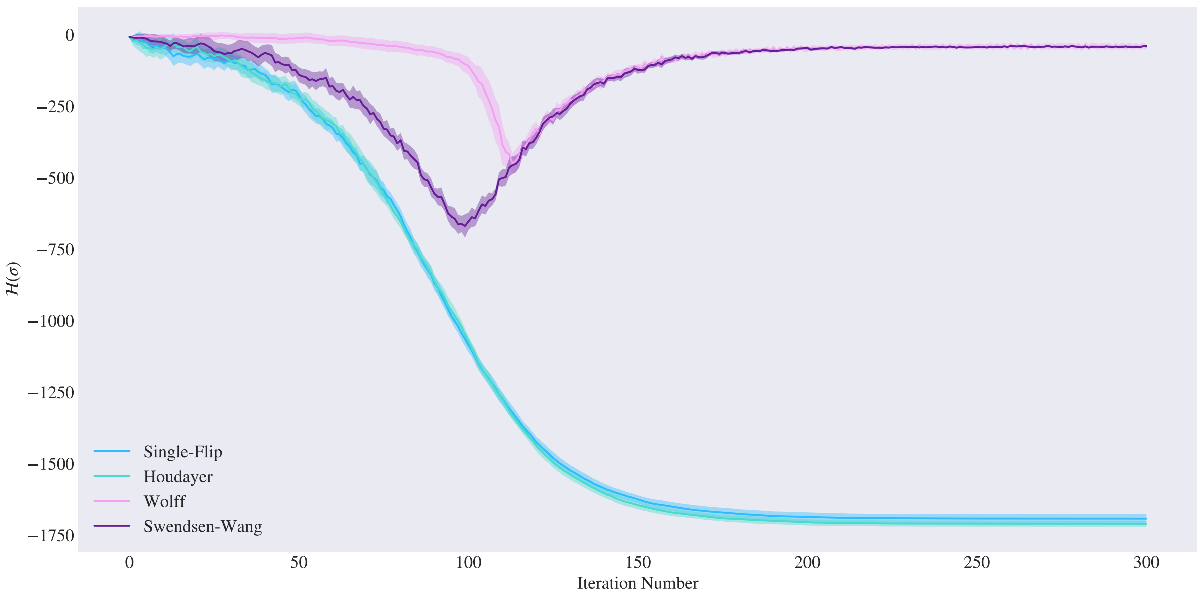


Figure A.3 – Result of SA for various moves on a $ER(1024, 0.007)$ graph for the spin-glass model.

.

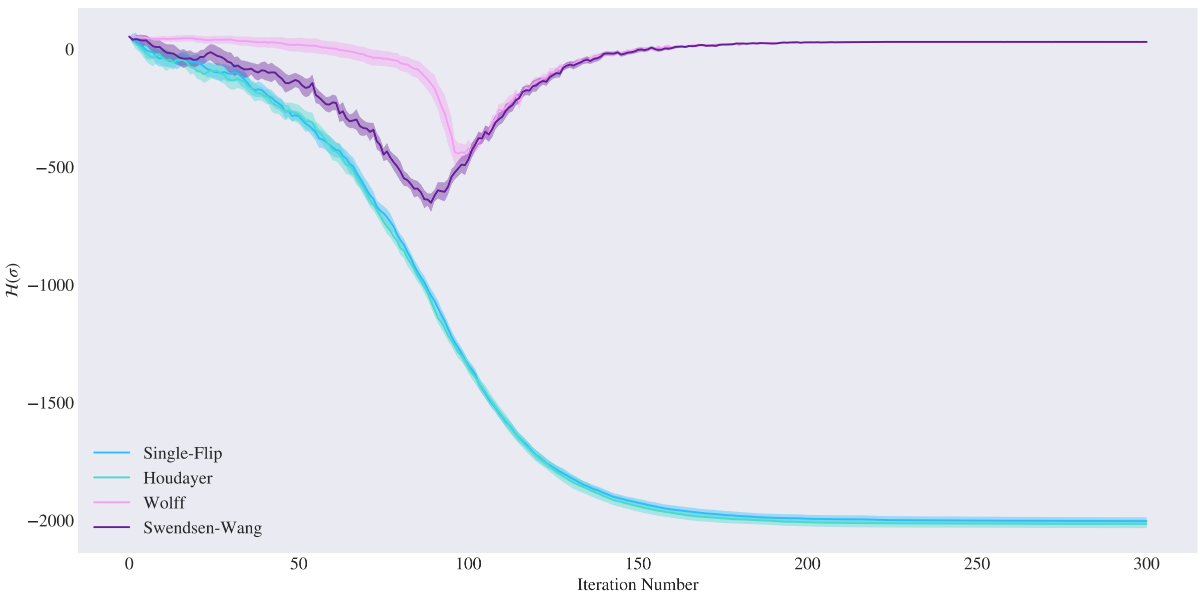


Figure A.4 – Result of SA for various moves on a $ER(1024, 0.01)$ graph for the spin-glass model.
.

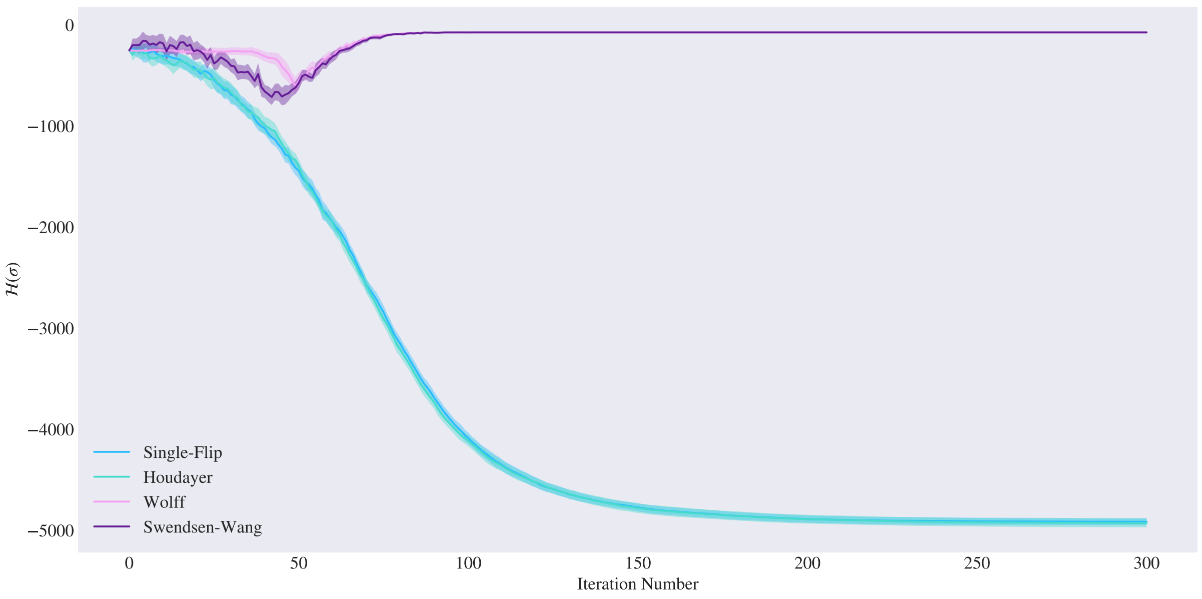


Figure A.5 – Result of SA for various moves on a $ER(1024, 0.05)$ graph for the spin-glass model.
.

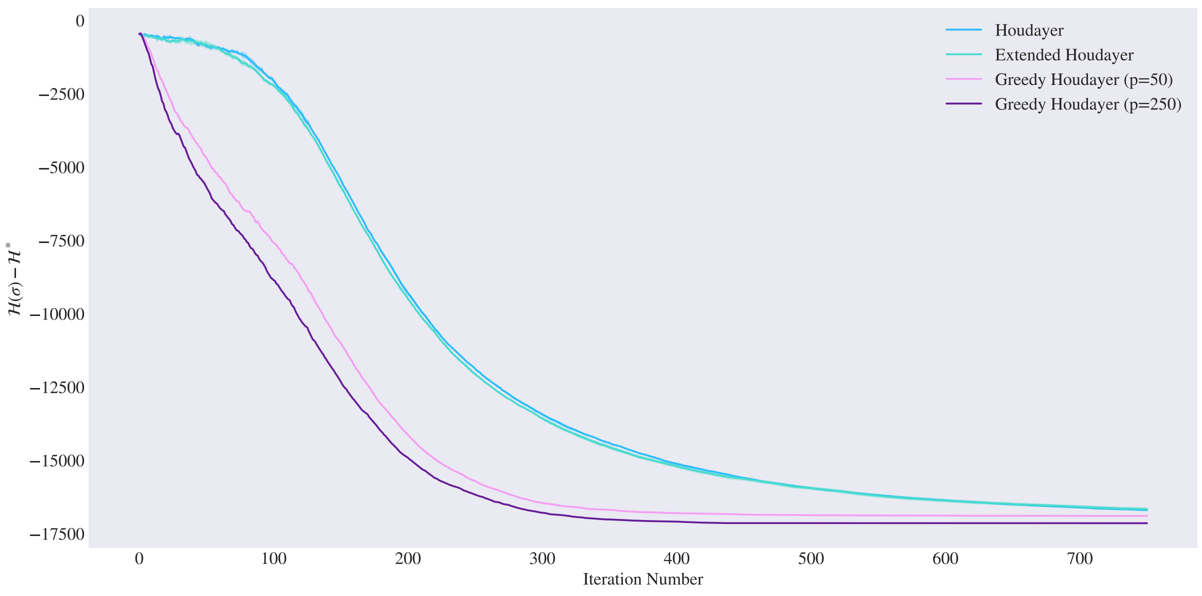


Figure A.6 – Result of SA for different variants of the Houdayer moves on a random graph with $ER(n = 10000, p = 0.0008)$ for the spin-glass model.

Bibliography

[1] P. Date, D. Arthur, and L. Pusey-Nazzaro, *Qubo formulations for training machine learning models*, 2020. arXiv: 2008.02369 `[cs.LG]`.

[2] Y. Ding, X. Chen, L. Lamata, E. Solano, and M. Sanz, "Implementation of a hybrid classical-quantum annealing algorithm for logistic network design," *SN Computer Science*, vol. 2, no. 2, Feb. 2021, ISSN: 2661-8907. DOI: 10.1007/s42979-021-00466-2. [Online]. Available: http://dx.doi.org/10.1007/s42979-021-00466-2.

[3] D. Snelling, G. Shahane, W. Shipman, A. Balaeff, M. Pearce, and S. Keinan, "A quantum-inspired approach to de-novo drug design," *ChemRxiv*, 2020. DOI: 10.26434/chemrxiv.12229232.v1.

[4] N. M. D. Oliveira, R. M. D. A. Silva, and W. R. D. Oliveira, "Qubo formulation for the contact map overlap problem," *International Journal of Quantum Information*, vol. 16, no. 08, p. 1 840 007, 2018. DOI: 10.1142/S0219749918400075. eprint: https://doi.org/10.1142/S0219749918400075. [Online]. Available: https://doi.org/10.1142/S0219749918400075.

[5] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953. DOI: 10.1063/1.1699114. [Online]. Available: http://link.aip.org/link/?JCP/21/1087/1.

[6] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970, ISSN: 0006-3444. DOI: 10.1093/biomet/57.1.97. eprint: https://academic.oup.com/biomet/article-pdf/57/1/97/23940249/57-1-97.pdf. [Online]. Available: https://doi.org/10.1093/biomet/57.1.97.

[7] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, Apr. 1965.

[8] E. Farhi, J. Goldstone, and S. Gutmann, *A quantum approximate optimization algorithm*, 2014. arXiv: 1411.4028 `[quant-ph]`.

[9] F. G. S. L. Brandao and K. Svore, *Quantum speed-ups for semidefinite programming*, 2017. arXiv: 1609.05537 `[quant-ph]`.

[10] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information*, vol. 2, no. 1, Jan. 2016, ISSN: 2056-6387. DOI: 10.1038/npjqi.2015.23. [Online]. Available: http://dx.doi.org/10.1038/npjqi.2015.23.

[11] A. Gilliam, S. Woerner, and C. Gonciulea, "Grover Adaptive Search for Constrained Polynomial Binary Optimization," *Quantum*, vol. 5, p. 428, Apr. 2021, ISSN: 2521-327X. DOI: 10.22331/q-2021-04-08-428. [Online]. Available: https://doi.org/10.22331/q-2021-04-08-428.

[12] B. Apolloni, C. Carvalho, and D. de Falco, "Quantum stochastic optimization," *Stochastic Processes and their Applications*, vol. 33, no. 2, pp. 233–244, 1989, ISSN: 0304-4149. DOI: https://doi.org/10.1016/0304-4149(89)90040-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304414989900409.

[13] S. Mandrà, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber, "Strengths and weaknesses of weak-strong cluster problems: a detailed overview of state-of-the-art classical heuristics versus quantum approaches," *Physical Review A*, vol. 94, no. 2, Aug. 2016, ISSN: 2469-9934. DOI: 10.1103/physreva.94.022337. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.94.022337.

[14] J. Houdayer, "A cluster monte carlo algorithm for 2-dimensional spin glasses," *The European Physical Journal B*, vol. 22, no. 4, pp. 479–484, Aug. 2001, ISSN: 1434-6028. DOI: 10.1007/pl00011151. [Online]. Available: http://dx.doi.org/10.1007/PL00011151.

[15] Z. Zhu, A. Ochoa, and H. Katzgraber, "Efficient cluster algorithm for spin glasses in any space dimension," *Physical review letters*, vol. 115, Jan. 2015. DOI: 10.1103/PhysRevLett.115.077201.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, New Series, vol. 220, no. 4598, pp. 671–680, 1983, ISSN: 00368075. DOI: 10.1126/science.220.4598.671. [Online]. Available: http://www.jstor.org/stable/1690046.

[17] K. Hukushima and K. Nemoto, "Exchange monte carlo method and application to spin glass simulations," *Journal of the Physical Society of Japan*, vol. 65, no. 6, pp. 1604–1608, Jun. 1996, ISSN: 1347-4073. DOI: 10.1143/jpsj.65.1604. [Online]. Available: http://dx.doi.org/10.1143/JPSJ.65.1604.

[18] T. Bartz-Beielstein, C. Doerr, D. van den Berg, J. Bossek, S. Chandrasekaran, T. Eftimov, A. Fischbach, P. Kerschke, W. L. Cava, M. Lopez-Ibanez, K. M. Malan, J. H. Moore, B. Naujoks, P. Orzechowski, V. Volz, M. Wagner, and T. Weise, *Benchmarking in optimization: best practice and open issues*, 2020. arXiv: 2007.03488 `[cs.NE]`.

[19] E. Boros and P. Hammer, "The max-cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds," *Annals of Operations Research*, vol. 33, pp. 151–180, 1991.

[20] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov chains and mixing times.* American Mathematical Society, 2006. [Online]. Available: http://scholar.google.com/scholar.bib?q=info:3wf9IU94tyMJ:scholar.google.com/&output=citation&hl=en&as_sdt=2000&ct=citation&cd=0.

[21] J. Ding, E. Lubetzky, and Y. Peres, "The mixing time evolution of glauber dynamics for the mean-field ising model," *Communications in Mathematical Physics*, vol. 289, no. 2, pp. 725–764, Apr. 2009, ISSN: 1432-0916. DOI: 10.1007/s00220-009-0781-9. [Online]. Available: http://dx.doi.org/10.1007/s00220-009-0781-9.

[22] J. Ding and Y. Peres, *Mixing time for the ising model: a uniform lower bound for all graphs*, 2013. arXiv: 0909.5162 `[math.PR]`.

[23] M. Dyer, C. Greenhill, and M. Ullrich, "Structure and eigenvalues of heat-bath markov chains," *Linear Algebra and its Applications*, vol. 454, pp. 57–71, 2014, ISSN: 0024-3795. DOI: https://doi.org/10.1016/j.laa.2014.04.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0024379514002328.

[24] M. Ullrich, "Comparison of swendsen-wang and heat-bath dynamics," *Random Structures Algorithms*, vol. 42, no. 4, pp. 520–535, May 2012, ISSN: 1042-9832. DOI: 10.1002/rsa.20431. [Online]. Available: http://dx.doi.org/10.1002/rsa.20431.

[25] M. Raginsky, "Strong data processing inequalities and <inline-formula> <tex-math notation="latex">Φ </tex-math> </inline-formula>-sobolev inequalities for discrete channels," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3355–3389, Jun. 2016, ISSN: 1557-9654. DOI: 10.1109/tit.2016.2549542. [Online]. Available: http://dx.doi.org/10.1109/TIT.2016.2549542.

[26] A. Lucas, "Ising formulations of many np problems," *Frontiers in Physics*, vol. 2, p. 5, 2014, ISSN: 2296-424X. DOI: 10.3389/fphy.2014.00005. [Online]. Available: https://www.frontiersin.org/article/10.3389/fphy.2014.00005.

[27] E. Ising, "Contribution to the Theory of Ferromagnetism," *Z. Phys.*, vol. 31, pp. 253–258, 1925. DOI: 10.1007/BF02980577.

[28] D. Chowdhury, *Spin Glasses and Other Frustrated Systems*. CO-PUBLISHED WITH PRINCETON UNIVERSITY PRESS., 1986. DOI: 10.1142/0223. eprint: https://www.worldscientific.com/doi/pdf/10.1142/0223. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/0223.

[29] M. S. Shell, *Thermodynamics and Statistical Mechanics: An Integrated Approach*, ser. Cambridge Series in Chemical Engineering. Cambridge University Press, 2015. DOI: 10.1017/CBO9781139028875.

[30] E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, vol. 106, no. 4, pp. 620–630, May 1957. DOI: 10.1103/PhysRev.106.620. [Online]. Available: http://prola.aps.org/abstract/PR/v106/i4/p620_1.

[31] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press, 2003.

[32] E. Luijten, “Introduction to cluster monte carlo algorithms,” in *Computer Simulations in Condensed Matter Systems: From Materials to Chemical Biology Volume 1*, M. Ferrario, G. Ciccotti, and K. Binder, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 13–38, ISBN: 978-3-540-35273-0. DOI: 10.1007/3-540-35273-2_1. [Online]. Available: https://doi.org/10.1007/3-540-35273-2_1.

[33] J.-S. Wang and R. H. Swendsen, “Cluster monte carlo algorithms,” *Physica A: Statistical Mechanics and its Applications*, vol. 167, no. 3, pp. 565–579, 1990, ISSN: 0378-4371. DOI: https://doi.org/10.1016/0378-4371(90)90275-W. [Online]. Available: http://www.sciencedirect.com/science/article/pii/037843719090275W.

[34] V. Cataudella, G. Franzese, M. Nicodemi, A. Scala, and A. Coniglio, “Critical clusters and efficient dynamics for frustrated spin models,” *Phys. Rev. Lett.*, vol. 72, pp. 1541–1544, 10 Mar. 1994. DOI: 10.1103/PhysRevLett.72.1541. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.72.1541.

[35] P. D. Coddington and L. Han, “Generalized cluster algorithms for frustrated spin models,” *Physical Review B*, vol. 50, no. 5, pp. 3058–3067, Aug. 1994, ISSN: 1095-3795. DOI: 10.1103/physrevb.50.3058. [Online]. Available: http://dx.doi.org/10.1103/PhysRevB.50.3058.

[36] T. G. J. Myklebust, *Solving maximum cut problems by simulated annealing*, 2015. arXiv: 1505.03068 `[math.OC]`.

[37] J. Lam and J.-M. Delosme, “An efficient simulated annealing schedule: implementation and evaluation,” *Tech Rep 8817*, Jan. 1988.

[38] J. King, M. Mohseni, W. Bernoudy, A. Fréchette, H. Sadeghi, S. V. Isakov, H. Neven, and M. H. Amin, *Quantum-assisted genetic algorithm*, 2019. arXiv: 1907.00707 `[quant-ph]`.

[39] P. Erdos and A. Renyi, “On the evolution of random graphs,” *Publ. Math. Inst. Hungary. Acad. Sci.*, vol. 5, pp. 17–61, 1960.

[40] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity.* Jan. 1982, vol. 32, ISBN: 0-13-152462-3. DOI: 10.1109/TASSP.1984.1164450.

[41] U. Wolff, “Critical slowing down,” *Nuclear Physics B - Proceedings Supplements*, vol. 17, pp. 93–102, 1990, ISSN: 0920-5632. DOI: https://doi.org/10.1016/0920-5632(90)90224-I. [Online]. Available: https://www.sciencedirect.com/science/article/pii/092056329090224I.

[42] A. J. Ochoa, D. C. Jacob, S. Mandrà, and H. G. Katzgraber, “Feeding the multitude: a polynomial-time algorithm to improve sampling,” *Physical Review E*, vol. 99, no. 4, Apr. 2019, ISSN: 2470-0053. DOI: 10.1103/physreve.99.043306. [Online]. Available: http://dx.doi.org/10.1103/PhysRevE.99.043306.

[43] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd. The MIT Press, 2001, ISBN: 0262032937. [Online]. Available: http://www.amazon.com/Introduction-Algorithms-Thomas-H-Cormen/dp/0262032937%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0262032937.

[44] J. He and L. Kang, “On the convergence rates of genetic algorithms,” *Theor. Comput. Sci.*, vol. 229, no. 1, pp. 23–39, 1999. DOI: 10.1016/S0304-3975(99)00091-2. [Online]. Available: https://doi.org/10.1016/S0304-3975(99)00091-2.

[45] A. Mandal, A. Roy, S. Upadhyay, and H. Ushijima-Mwesigwa, *Compressed quadratization of higher order binary optimization problems*, 2020. arXiv: 2001.00658 `[quant-ph]`.

[46] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, Mar. 2004, ISBN: 0521833787.

[47] J. Han, M. Kamber, and J. Pei, “2 - getting to know your data,” in *Data Mining (Third Edition)*, ser. The Morgan Kaufmann Series in Data Management Systems, J. Han, M. Kamber, and J. Pei, Eds., Third Edition, Boston: Morgan Kaufmann, 2012, pp. 39–82, ISBN: 978-0-12-381479-1. DOI: https://doi.org/10.1016/B978-0-12-381479-1.00002-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123814791000022.

[48] V. Kargin, *Relaxation time is monotone in temperature in the mean-field ising model*, 2011. arXiv: 1103.0327 `[math.PR]`.

[49] U. Schöning and J. Torán, *Das Erfüllbarkeitsproblem SAT - Algorithmen und Analysen*, ser. Mathematik für Anwendungen. Lehmann, 2012, vol. 1, ISBN: 978-3-86541-473-1. [Online]. Available: http://www.lehmanns.de/shop/mathematik-informatik/22631438-9783865414731-das-erfuellbarkeitsproblem-sat.

[50] G. Van Rossum and F. L. Drake Jr, *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[51] W. Wang, J. Machta, and H. G. Katzgraber, “Population annealing: theory and application in spin glasses,” *Physical Review E*, vol. 92, no. 6, Dec. 2015, ISSN: 1550-2376. DOI: 10.1103/physreve.92.063307. [Online]. Available: http://dx.doi.org/10.1103/PhysRevE.92.063307.