**SVEUČILIŠTE U ZAGREBU**

**PRIRODOSLOVNO–MATEMATIČKI FAKULTET**

**MATEMATIČKI ODSJEK**

Luka Jovanović

# PHYSICS-INSPIRED DYNAMICAL SYSTEMS FOR OPTIMIZATION

Master thesis

Supervisors:
Izv. prof. dr. sc. Ilja Gogić,
Dr. Daniel Ebler

Zagreb, October 2023.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

*Posveta...*

# Contents

# Introduction

The quadratic unconstrained binary optimization (QUBO) task is the minimization task defined by

$$\min_{s_1,\ldots,s_n \in \{0,1\}} \sum_{i,j=1}^{n} J_{ij} s_i s_j + \sum_{i=1}^{n} h_i s_i + c$$

where the coefficients $J_{ij}, h_i, c \in \mathbb{R}$ are given. This simple combinatorial optimization task has many applications in a sense that it can be used to model various real-world problems. These real-world problems include combinatorial optimization problems which arise in finance [30], cluster analysis [36], economic analysis [24], computer-aided layout design [29], integrated chip design [14], physics [14], and many more [28]. Although being simple to state, this combinatorial optimization problems is in general very hard to solve if the number of variables is large. One obvious algorithm for solving such a problem would be to try every possible combination $s_i \in \{0, 1\}$, evaluate it, and then take the minimum result found. However, the problem with this algorithm is that the number of possible combinations which need to be evaluated is $2^n$ which grows exponentially with respect to the number of variables. For example, solving a problem with $n = 100$ variables on a computer which can perform $10^8$ multiply/add operations per second would require more than $2^{100} * 100^2/10^8 > 10^{26}$ seconds which is longer time than the predicted current age of the Universe (according to [11]).

The question which now emerges is whether there exists a significantly faster algorithm which would find the exact solution of QUBO? The short answer to this question is — *probably no*. In fact, QUBO is an archetype of a NP-hard combinatorial optimization problem [28][8], meaning that there is no polynomial-time algorithm which would solve it, unless P=NP which is suspected to be false [6]. Furthermore, many NP-complete and NP-hard problems, including all of Karp's 21 NP-complete problems, can be reduced to a QUBO problem efficiently [33].

Although being NP-hard, there is a vast number of algorithms for solving QUBO problem exactly. Each of these algorithms exploit some properties of QUBO and provide some smarter way to reach the exact solution than trying all combinations. Nevertheless, their complexity is still non-polynomial and thus they work efficiently for up to few hundreds

of variables at most. Many of these algorithms, developed before 2014., can be found in a survey [28].

On the other hand, there is a multitude of heuristic and metaheuristic algorithms for solving QUBO approximately but quickly. These algorithms usually do not have a guarantee of achieving certain accuracy, but their performance is rather based on empirical evidence. These algorithms include variations and adaptations of tabu search [20], simulated annealing [15][13], genetic algorithms [34], Hopfield neural networks [38], and many others [28]. There is also one notable metaheuristic algorithm called Breakout Local Search (BLS) [16] which provides high quality approximate solutions of MAX-CUT problem, which is equivalent to QUBO 1.2.5.

Along these classical algorithms designed for running on classical computers, significant effort was put into developing quantum algorithms designed for running on quantum computers. Part of these quantum algorithms rely on the adiabatic theorem of quantum mechanics [1][33], and are thus called quantum adiabatic optimization algorithms. The working principle is as follows. First, the combinatorial optimization problem is encoded into a Hamiltonian $H_P$ in such a way that the ground state of this Hamiltonian encodes the optimum solution of the combinatorial problem. In order to find the ground state of $H_P$, one first prepares the quantum state to be the ground state of some initial Hamiltonian $H_0$ whose ground state is easy to find and construct. Then, the system is *slowly* evolved in time from $H_0$ to $H_P$ according to

$$H(t) = (1 - \frac{t}{T})H_0 + \frac{t}{T}H_P$$

According to the adiabatic theorem of quantum mechanics, if the system is evolved slowly enough, the state will remain the ground state at each time instance. Thus, at the final time instance $T$, the ground state of the Hamiltonian $H(T) = H_P$ will be prepared. After measurement, it provides the solution of the original combinatorial optimization problem. There is, however, a debate whether or not would this technique be useful in practice because, in order to satisfy the assumptions of the adiabatic theorem of quantum mechanics, one often finds that the required time for evolution depends exponentially on the problem size [33]. Besides this, in order to run such a quantum algorithm, one would first need a quantum computer with a large enough number of qubits in order to outperform the existing classical computer architectures, which has (probably) not been built yet.

Recently, a new paradigm for heuristic approaches has been proposed in [40] and [22]. Instead of evolving a quantum Hamiltonian on a quantum device, one can define the corresponding classical dynamical system. This is done by approximating the expected value of annihilation operator $a$, present in quantum Hamiltonian $H$, by a complex number $x + iy$ where $x, y \in \mathbb{R}$. Thus, a dynamical system representing the equations of motion are provided for $x$ and $y$. This dynamical system can be simulated on a classical computer, providing new heuristic algorithms. By further modifying and simplifying these dynamical

systems, more efficient ones can be obtained in a sense that they can be simulated more efficiently while preserving or even improving the final solutions for combinatorial optimization problem. This is how Coherent Ising Machine (CIM) [32], Simulated Bifurcation (SB) [23], Simulated Coherent Ising Machine (SimCIM) [37], Ballistic Simulated Bifurcation (bSB) [19], Discrete Simulated Bifurcation (dSB) [19], and some other related algorithms were born. These algorithms are easily parallelized on GPUs or similar hardware devices which enables them to quickly provide high-quality solutions for large instances of QUBO and other combinatorial problems.

The goal of this master thesis is presenting and analyzing these physics-inspired dynamical systems from a mathematical perspective and explaining how these can be used for solving combinatorial optimization problems. To understand how a continuous dynamical system can provide a solution of the combinatorial optimization problem, consider a state vector which evolves in time $\mathbf{x} : [0, \infty\rangle \to \mathbb{R}^n$. Taking the sign of each component at some time instance $t$, $\sigma_i \coloneqq x_i(t)$ provides a candidate solution for the QUBO task $(\sigma_1, \ldots, \sigma_n)$. The evolution of a continuous dynamical system is determined by its vector field and initial conditions. The vector field is defined at each point in space and determines the velocity $\dot{\mathbf{x}}$ of the system if it passes through that point. The initial conditions $\mathbf{x}_0 \in \mathbb{R}^n$ determine the starting position of the dynamical system $\mathbf{x}(0) = \mathbf{x}_0$. The goal is to design the dynamical system in such a way that it *attracts* various trajectories towards such areas of space that provide high-quality approximate solutions of the QUBO problem, according to the mapping mentioned above $\sigma_i \coloneqq x_i(t)$.

In chapter 1, QUBO problem is studied. Several other combinatorial optimization problems are presented including MAX-CUT and Traveling Salesman Problem, and their embedding into QUBO problem is given.

In chapter 2, a mathematical theory for dynamical systems and certain other topics are briefly presented, with the main purpose of providing a good understanding of topics in chapter 3.

Chapter 3 is the core chapter as it presents and analyzes these aforementioned physics-inspired dynamical systems and their corresponding numerical simulations which are together used for approximately solving QUBO problem. These algorithms include Coherent Ising Machine (CIM), Simulated Bifurcation (SB), Simulated Coherent Ising Machine (SimCIM), and Ballistic Simulated Bifurcation (bSB). In order to provide a bigger picture and the mutual relation between these algorithms, they may be presented in slightly different form than in original articles.

All of these algorithms have efficient implementations which are run on classical computers or even parallelized architectures such as GPUs and FPGAs. These algorithms can thus be used for fast sampling of high-quality QUBO solutions. The results obtained by these algorithms are presented in the final chapter 4. These results cover performance of algorithms on a benchmark dataset GSet along with methods used for fine-tuning param-

eters. Finally, a *dropout* technique is proposed for improving the performance of these algorithms.

**Contributions:**

This thesis focuses on analyzing physics-inspired dynamical systems purely from a formal mathematical perspective. Thus, some arguments mentioned in the original articles are refined here and formalized as much as possible. In order to do that, it was first necessary to extract the relevant existing theorems from the theory of dynamical systems and other fields. Some arguments for explaining the mechanism of these algorithms are only applicable to autononomous dynamical systems. Since dynamical systems of interest are nonautonomous, the author proposes a theorem for connecting nonautonomous system whose vector field changes slowly in time to the corresponding autonomous systems with vector field frozen in time. This is the theorem 2.1.9.

As proposed in the original articles, the dynamical systems discussed in this thesis are solved approximately by simulating them numerically. In this thesis, the exact solutions of SimCIM and bSB algorithms are derived which is, as far as the author is aware, not yet reported in the literature, but could potentially be useful for further research.

Although momentum has been introduced for SimCIM algorithm, it seems not to be used for CIM. It was observed that adding momentum to CIM improved solution quality for QUBO problem. During fine-tuning, the momentum had the option to be 0 (original CIM), or some other larger values (for example 0.8, 0.9, 1.0). It indeed turned out to always pick some value larger than 0.

Furthermore, it is shown in chapter 3 that CIM with momentum is in a certain sense a generalization of SB algorithm. Similarly, SimCIM with momentum is in some sense a generalization of bSB algorithm, which is even able to exactly reproduce the behaviour of bSB.

Most of these algorithms have already been benchmarked on GSet. However, the thesis provides another independent source of these results. This can be understood as comparing all of these algorithms in a consistent and unified way.

Finally, a new technique for adding *meaningful noise* was proposed in chapter 4, which we call *dropout*. This technique was tested on GSet instances and seems to improve the solution quality obtained by CIM, SimCIM, and bSB algorithms. How exactly does this technique enable these algorithms to find better solutions is yet to be researched.

# Chapter 1

# Quadratic Unconstrained Binary Optimization

Quadratic unconstrained binary optimization (QUBO) problem is a combinatorial optimization problem defined by

$$\min_{s_1,\dots,s_n\in\{0,1\}} \sum_{i,j=1}^{n} J_{ij}s_is_j + \sum_{i=1}^{n} h_is_i + c \tag{1.1}$$

where coefficients $J_{ij}, h_i, c \in \mathbb{R}$ are given. In other words, the task is to minimize a given quadratic polynomial in $n$ variables over the discrete domain $\{0,1\}^n$.

We are usually interested not only in finding a minimum but finding an argument which minimizes the function. Also, since finding such optimum is computationally very hard for large instances, we will be interested in finding as low value as possible and the corresponding argument. The domain $\{-1,+1\}^n$ is called the *search space* while the elements of this set are called *candidate solutions* or *feasible solutions* or simply *solutions*. An *optimum solution* is then the minimizer of the function over the set of feasible solutions - i.e. the best possible solution.

Since $h_is_i = h_is_i^2$ for $s_i \in \{0,1\}$, all terms of degree one can be transformed into quadratic terms. The constant $c$ does not play any role in minimization task. That being said, an equivalent formulation of QUBO task is minimizing a homogeneous quadratic polynomial in $n$ variables over the domain $\{0,1\}^n$,

$$\min_{s_1,\dots,s_n\in\{0,1\}} \sum_{i,j=1}^{n} J_{ij}s_is_j \tag{1.2}$$

Coefficients $J_{ij}$ can always be taken symmetrically i.e. such that $J_{ij} = J_{ji}$. Indeed, if they are not symmetric, taking coefficients $\frac{1}{2}(J_{ij} + J_{ji})$ in place of both $J_{ij}$ and $J_{ji}$ will make them symmetric.

As stated in the Introduction part, an obvious brute-force algorithm for solving QUBO problem is to evaluate the polynomial at all possible candidate solutions $s_i \in \{0, 1\}$ and take the minimum among them. However, the problem with this algorithm is that there are $2^n$ combinations in total, so the search space grows exponentially with respect to the number of variables. Depending on the computer's performance, this will work efficiently for number of variables of order $10^1$. However, already for $n = 100$ this becomes too many operations to perform on a computer within any reasonable amount of time.

The key point here is that, since QUBO is a NP-hard problem [28], there is no polynomial-time algorithm which would solve it exactly (unless P=NP which is suspected to be false [6]). Thus, there is probably no algorithm for solving QUBO exactly which would have significantly lower computational complexity than exponential one.

Nevertheless, there are many algorithms which solve QUBO exactly by leveraging various properties of the problem and using different techniques. Many of these methods are listed in [28]. However, all of these methods work effectively for number of vertices up to few hundreds at most.

For larger instances of QUBO, various heuristic and metaheuristic algorithms have been developed which provide some relatively high-quality approximate solution. A lot of such algorithms can be found in [28] as well. However, there is one special kind of heuristic algorithms recently proposed. These algorithms are based on physics-inspired dynamical systems. These algorithms are the core of this thesis and are presented in chapter 3.

It will be shown in 1.2 that QUBO problem is in fact equivalent to a specific graph problem called MAX-CUT. There are some approximation algorithms in the literature for solving MAX-CUT which are thus directly applicable to solving QUBO.

The question which emerges is whether there exists a polynomial-time algorithm which would provide arbitrarily good approximate solution. In fact, QUBO problem is APX-hard, which under the assumption P≠NP implies that there is no polynomial-time approximation scheme (PTAS) for it [7][3] [5]. In other words, no polynomial-time algorithm can guarantee to provide a solution which is as close to the optimal solution as we would require in advance (if P≠NP).

To address the question of how close to the optimal solution can some algorithm get with a guarantee, there is an article by Goemans and Williamson [21] proposing a randomized algorithm for solving MAX-CUT problem (see section 1.2) based on semidefinite programming which always provides solutions with expected value at least 0.87856 times the optimal solution. MAX-CUT is equivalent to QUBO 1.2.5 in such a way that there exists a mapping from candidate solutions of one problem to candidate solutions of the other problem, which preserves value obtained by these candidate solutions. Thus, algorithm which provides solutions with expected value at least 0.87856 times the optimal solution of the MAX-CUT problem could also be used to provide solutions of the QUBO problem with the same accuracy.

If the unique games conjecture [10] is true, this is the best possible approximation ratio which can be guaranteed for MAX-CUT [5] [10] 1.2 1.2.5, and thus for QUBO as well.

## Equivalent forms

Let $Q'(s_1, \ldots, s_n) = \sum_{i,j=1}^{n} J'_{ij} s_i s_j + \sum_{i=1}^{n} h'_i s_i + c'$ be a quadratic polynomial with the domain $\{0,1\}^n$ as in 1.1. By taking a linear change of variables given by $\sigma_i = 2s_i - 1$, a new quadratic polynomial $Q(\sigma_1, \ldots, \sigma_n)$ is given over the domain $\{-1,+1\}^n$, satisfying $Q'(s_1, \ldots, s_n) = Q(\sigma_1, \ldots, \sigma_n)$. Thus, the QUBO task 1.1 has an equivalent form

$$\min_{\sigma_1,\ldots,\sigma_n \in \{-1,+1\}} \sum_{i,j=1}^{n} J_{ij}\sigma_i\sigma_j + \sum_{i=1}^{n} h_i\sigma_i + c \tag{1.3}$$

for some new coefficients $J_{ij}, h_i, c$.

Although equivalent, each of these formulations has it's own benefit. As it will be seen in the following sections, some other combinatorial optimization problems can be embedded into QUBO task. Sometimes it will be easier to think of an embedding in 1.1 form and sometimes in 1.3 form. Form 1.3 might be more suitable for analysis because the domain $\{-1,+1\}^n$ consists of elements with equal norm. When solving QUBO task on a computer, the algorithm will often be implemented either as $Q$ over the domain $\{-1,+1\}^n$ or as $Q'$ over the domain $\{0,1\}^n$. It is thus useful to have an explicit relation between coefficients in $Q$ and $Q'$. That relation is given by

$$\begin{aligned}
&Q\colon \{-1,+1\}^n \to \mathbb{R} && Q'\colon \{0,1\}^n \to \mathbb{R} \\
&J_{ij} = \frac{1}{4} J'_{ij} && J'_{ij} = 4J_{ij} \\
&h_i = \frac{1}{4}\sum_{j=1}^{n}(J'_{ij} + J'_{ji}) + \frac{1}{2}h'_i && h'_i = -2\sum_{j=1}^{n}(J_{ij} + J_{ji}) + 2h_i \\
&c = \frac{1}{4}\sum_{i,j=1}^{n} J'_{ij} + \frac{1}{2}\sum_{i=1}^{n} h'_i + c' && c'_i = \sum_{i,j=1}^{n} J_{ij} - \sum_{i=1}^{n} h_i + c
\end{aligned} \tag{1.4}$$

Another equivalent form emerges if minimization task is replaced with maximization. Indeed,

$$\max_{\sigma_1,\ldots,\sigma_n \in \{-1,+1\}} Q(\sigma_1, \ldots, \sigma_n) = -\min_{\sigma_1,\ldots,\sigma_n \in \{-1,+1\}} -Q(\sigma_1, \ldots, \sigma_n) \tag{1.5}$$

and $-Q$ is still a quadratic polynomial.

## Homogenizing

Let us assume that a quadratic polynomial

$$Q(\sigma_1, \ldots, \sigma_n) = \sum_{i,j=1}^{n} J_{ij}\sigma_i\sigma_j + \sum_{i=1}^{n} h_i\sigma_i + c$$

is given over the domain $\{-1, +1\}^n$. For the minimization task, the constant term $c$ is irrelevant so here we assume without loss of generality that $c = 0$. By introducing one auxiliary variable $\sigma_{n+1} \in \{-1, +1\}$ we define a polynomial

$$P(\sigma_1, \ldots, \sigma_n, \sigma_{n+1}) \coloneqq \sum_{i,j=1}^{n} J_{ij}\sigma_i\sigma_j + \sum_{i=1}^{n} h_i\sigma_i\sigma_{n+1}$$

which is homogeneous of degree 2. For each configuration $(\sigma_1, \ldots, \sigma_n) \in \{-1, +1\}^n$, there are two corresponding configurations in $\{-1, +1\}^{n+1}$.
Those are $(\sigma_1, \ldots, \sigma_n, +1)$ and $(-\sigma_1, \ldots, -\sigma_n, -1)$. It follows that ($c = 0$ by assumption)

$$Q(\sigma_1, \ldots, \sigma_n) = P(\sigma_1, \ldots, \sigma_n, +1) = P(-\sigma_1, \ldots, -\sigma_n, -1) \tag{1.6}$$

Thus, finding a minimum of a (nonhomogeneous) quadratic polynomial with $n$ variables over the domain $\{-1, +1\}^n$ is equivalent to finding a minimum of the corresponding homogeneous quadratic polynomial with $n+1$ variables over the domain $\{-1, +1\}^{n+1}$. Because of 1.6, not only the minima are in correspondence, but all values are. Although increasing the number of variables by 1 does not play any important role in a sense of computational complexity, the performance of some algorithms might be affected because the *extra* variable $\sigma_{n+1}$ is (possibly) coupled to all other variables. If some algorithm leverages the coupling structure of the original problem (for example sparsity of $J$), the new coupling might lose this structure after introducing $\sigma_{n+1}$ (sparsity might be compromised because $\sigma_{n+1}$ could be coupled to all other variables), thus causing the algorithm to drop in performance.

Nevertheless, we will not be concerned with the effect of adding this *extra* variable in the rest of the thesis, but rather take this theoretical result as a justification for solving and analyzing mostly the homogeneous case

$$\min_{\sigma_1,\ldots,\sigma_n \in \{-1,+1\}} \sum_{i,j=1}^{n} J_{ij}\sigma_i\sigma_j \tag{1.7}$$

## Matrix-Vector notation

Given a quadratic polynomial with real coefficients

$$Q(\sigma_1, \ldots, \sigma_n) = \sum_{i,j=1}^{n} J_{ij}\sigma_i\sigma_j + \sum_{i=1}^{n} h_i\sigma_i + c$$

it's evaluation can be written in matrix-vector notation as follows. Put all coefficients $J_{ij}$ into the matrix $J \in M_n(\mathbb{R})$, coefficients $h_i$ into the vector $\mathbf{h} \in \mathbb{R}^n$ and arguments $\sigma_i$ into the vector $\boldsymbol{\sigma} \in \{-1, +1\}^n$. Then we write

$$\sum_{i,j=1}^{n} J_{ij}\sigma_i\sigma_j + \sum_{i=1}^{n} h_i\sigma_i + c = Q(\sigma_1, \ldots, \sigma_n) = Q(\boldsymbol{\sigma}) = \boldsymbol{\sigma}^T J\boldsymbol{\sigma} + \boldsymbol{\sigma}^T\mathbf{h} + c$$

and so the minimization task is rewritten as

$$\min_{\boldsymbol{\sigma}\in\{-1,+1\}^n} \boldsymbol{\sigma}^T J\boldsymbol{\sigma} + \boldsymbol{\sigma}^T\mathbf{h} + c \tag{1.8}$$

This transition between vectors, matrices and their coefficients, will be used in what follows without further noticing.

## 1.1 Analysis of the problem

### Bounds with coefficients

Define the following matrix and vector norms

$$\|J\|_1 := \sum_{i,j=1}^{n} \left|J_{ij}\right| \qquad \|\mathbf{h}\|_1 := \sum_{i=1}^{n} |h_i|$$

$$\|J\|_\infty := \max_{i,j=1,\ldots,n} \left|J_{ij}\right| \qquad \|\mathbf{h}\|_\infty := \max_{i=1,\ldots,n} |h_i|$$

Then we have the following bounds for the value in QUBO

$$\left|\sum_{i,j=1}^{n} J_{ij}\sigma_i\sigma_j + \sum_{i=1}^{n} h_i\sigma_i + c\right| \leq \|J\|_1 + \|h\|_1 + |c| \leq n^2\|J\|_\infty + n\|h\|_\infty + |c| \tag{1.9}$$

### Bounds with the largest and the smallest eigenvalue

Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of the symmetric coupling matrix $J$ sorted in descending ascending order, meaning that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Let $v_1, \ldots, v_n$ be the orthonormal basis which diagonalizes the coupling matrix $J$, corresponding to eigenvalues $\lambda_1, \ldots, \lambda_n$. Then we have the following:

$$\boldsymbol{\sigma} = \sum_i \mathbf{v}_i\mathbf{v}_i^T\boldsymbol{\sigma}$$

$$\boldsymbol{\sigma}^T J\boldsymbol{\sigma} = (\sum_i \mathbf{v}_i\mathbf{v}_i^T\boldsymbol{\sigma})^T J(\sum_i \mathbf{v}_i\mathbf{v}_i^T\boldsymbol{\sigma}) = \sum_{i,j} \boldsymbol{\sigma}^T\mathbf{v}_i\mathbf{v}_i^T J\mathbf{v}_j\mathbf{v}_j^T\boldsymbol{\sigma} = \sum_i \lambda_i\boldsymbol{\sigma}^T\mathbf{v}_i\mathbf{v}_i^T\boldsymbol{\sigma}$$

Thus,

$$\boldsymbol{\sigma}^T J\boldsymbol{\sigma} \leq \max\{\lambda_1, \ldots, \lambda_n\} \sum_i \boldsymbol{\sigma}^T \mathbf{v}_i \mathbf{v}_i^T \boldsymbol{\sigma} = \max\{\lambda_1, \ldots, \lambda_n\} \boldsymbol{\sigma}^T \boldsymbol{\sigma}$$

$$\boldsymbol{\sigma}^T J\boldsymbol{\sigma} \geq \min\{\lambda_1, \ldots, \lambda_n\} \sum_i \boldsymbol{\sigma}^T \mathbf{v}_i \mathbf{v}_i^T \boldsymbol{\sigma} = \min\{\lambda_1, \ldots, \lambda_n\} \boldsymbol{\sigma}^T \boldsymbol{\sigma}$$

Since for all vectors $\boldsymbol{\sigma} \in \{-1, +1\}^n$ the quantity $\boldsymbol{\sigma}^T \boldsymbol{\sigma}$ is equals $n$, we obtain the bounds

$$n \cdot \min\{\lambda_1, \ldots, \lambda_n\} \leq \boldsymbol{\sigma}^T J\boldsymbol{\sigma} \leq n \cdot \max\{\lambda_1, \ldots, \lambda_n\} \tag{1.10}$$

## Degree of synchronization

Let $Q(\mathbf{x}) = -\mathbf{x}^T J\mathbf{x}$ and consider the following minimization QUBO task

$$\min_{\boldsymbol{\sigma} \in \{-1,+1\}^n} Q(\boldsymbol{\sigma}) \tag{1.11}$$

We have that $\lambda_{\min}(-J) = -\lambda_{\max}(J)$ with eigenvectors corresponding to each other. According to the previous subsection, if the eigenvector $\mathbf{v}_{\max}$ corresponding to the largest eigenvalue $\lambda_{\max}(J)$ consists only of components with magnitude 1 i.e. $\mathbf{v}_{\max} \in \{-1, +1\}^n$, then the solution of the QUBO problem (1.11) is exactly the vector $\mathbf{v}_{\max}$. Intuitively this should also hold in case that $\mathbf{v}_{\max}$ is very close to some point from the feasible set $\{-1, +1\}^n$ (ignoring the scale and considering all vectors to be normalized). This is formally justified in the following result from [39].

**Definition 1.1.1.** *Let* $\mathbf{x}$ *be an arbitrary vector in* $R^n$*. The feasible solution for QUBO task corresponding to this vector is considered to be* $\boldsymbol{\sigma} = \text{sign}(\mathbf{x})$*. We define the* degree of synchronization *of* $\mathbf{x}$ *to be*

$$\alpha^2(\mathbf{x}) = \left( \frac{\mathbf{x}^T}{||\mathbf{x}||} \frac{\boldsymbol{\sigma}}{||\boldsymbol{\sigma}||} \right)^2 \tag{1.12}$$

*where* $\boldsymbol{\sigma} = \text{sign}(\mathbf{x})$*.*

**Theorem 1.1.2.** *Consider the minimization QUBO task* (1.11)*. Denote with* $Q_0$ *the optimum (minimum) solution for this minimization task. Denote with* $Q_1$ *the second smallest value if it exists, otherwise let* $Q_1 = Q_0$*. Denote with* $\Delta Q = Q_1 - Q_0$*. Let* $\lambda_{\max} = \lambda_1 \geq \cdots \geq \lambda_n = \lambda_{\min}$ *be all eigenvalues of* $J$ *sorted in descending order. Assume that the largest eigenvalue's multiplicity is 1 (i.e.* $\lambda_1 \neq \lambda_2$*). Denote with* $\mathbf{v}_{\max}$ *a normalized eigenvector corresponding to* $\lambda_{\max}$*.*

*If for the degree of synchronization of vector* $\mathbf{v}_{\max}$ *it holds*

$$\alpha^2(\mathbf{v}_{\max}) \geq 1 - \frac{\Delta Q}{n(\lambda_{\max} - \lambda_{\min})} \tag{1.13}$$

*then the corresponding solution* $\boldsymbol{\sigma} = \text{sign}(\mathbf{v}_{\max})$ *is the optimum solution of QUBO task* (1.11).

*Proof.* If $\Delta Q = 0$ we have that all feasible solutions are minima so there is nothing to prove. Thus, let us assume that $\Delta Q > 0$.

$\alpha^2(\mathbf{v}_{\max}) = 1$ is equivalent to the fact that $\boldsymbol{\sigma}$ is proportional to $\mathbf{v}_{\max}$. In this case, $\boldsymbol{\sigma}$ reaches the bound value of (1.10) so $\boldsymbol{\sigma}$ must be the optimum solution and the proof is done.

Consider the opposite case where $\boldsymbol{\sigma}$ is not proportional to $\mathbf{v}_{\max}$. Since $\lambda_i < \lambda_{\max}, \forall i > 1$ we have that

$$Q_0 > \min_{\|\mathbf{x}\|^2=n} Q(\mathbf{x}) = -n\lambda_{\max} \tag{1.14}$$

which implies

$$Q_1 > -n\lambda_{\max} + \Delta Q \tag{1.15}$$

Using the fact that $J = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i^T$ we have that

$$\begin{aligned} Q(\boldsymbol{\sigma}) &= -\boldsymbol{\sigma}^T J \boldsymbol{\sigma} = -\boldsymbol{\sigma}^T \Big(\sum_i (\lambda_i - \lambda_{\min}) \mathbf{v}_i \mathbf{v}_i^T\Big)\boldsymbol{\sigma} - \sum_i \lambda_{\min} \boldsymbol{\sigma}^T \mathbf{v}_i \mathbf{v}_i^T \boldsymbol{\sigma} \\ &= -\sum_i (\lambda_i - \lambda_{\min})(\mathbf{v}_i^T \boldsymbol{\sigma})^2 - n\lambda_{\min} \\ &\leq -(\lambda_{\max} - \lambda_{\min})(\mathbf{v}_{\max}^T \boldsymbol{\sigma})^2 - n\lambda_{\min} \end{aligned} \tag{1.16}$$

where for obtaining the last inequality we have dropped all but one summation terms. Since $n\alpha^2(\mathbf{v}_{\max}) = (\mathbf{v}_{\max}^T \boldsymbol{\sigma})^2$, by assumption of the theorem we have that

$$(\mathbf{v}_{\max}^T \boldsymbol{\sigma})^2 \geq n - \frac{\Delta Q}{\lambda_{\max} - \lambda_{\min}} \tag{1.17}$$

Combining these we get

$$Q(\boldsymbol{\sigma}) \leq \Delta Q - n(\lambda_{\max} - \lambda_{\min}) - n\lambda_{\min} = \Delta Q - n\lambda_{\max} < Q_1 \tag{1.18}$$

Thus, $Q(\boldsymbol{\sigma}) = Q_0$ i.e. $\boldsymbol{\sigma}$ is the optimum solution of the QUBO task. □

## 1-opt solution

Let $Q(\sigma_1, \ldots, \sigma_n) = \sum_{i,j=1}^n J_{ij}\sigma_i\sigma_j + \sum_{i=1}^n h_i\sigma_i + C$.

**Definition 1.1.3.** *For the maximization QUBO task*

$$\max_{\boldsymbol{\sigma} \in \{-1,+1\}^n} Q(\boldsymbol{\sigma}), \tag{1.19}$$

*vector* $\boldsymbol{\sigma} \in \{-1, +1\}^n$ *is called a* 1-opt solution *if for every* $i = 1, \ldots, n$ *we have that*

$$\Delta_i Q(\sigma_1, \ldots, \sigma_n) \leq 0 \tag{1.20}$$

*where*

$$\Delta_i Q(\sigma_1, \ldots, \sigma_n) \coloneqq Q(\sigma_1, \ldots, \sigma_{i-1}, -\sigma_i, \sigma_{i+1}, \ldots, \sigma_n) - Q(\sigma_1, \ldots, \sigma_{i-1}, \sigma_i, \sigma_{i+1}, \ldots, \sigma_n) \tag{1.21}$$

*That is, for a 1-opt solution, switching the value of a single variable cannot produce a better solution.*

*By replacing* $\leq$ *with* $\geq$ *in equation* (1.21) *we get the definition of a* 1-opt solution *for the minimization QUBO task.*

*In a similar way we can define a* $k$-opt solution *for integer* $k > 1$.

Writing out further the formula for $\Delta_i Q$ and using the fact that $J_{ij} = J_{ji}$, $J_{ii} = 0$, we get

$$\Delta_i Q(\sigma_1, \ldots, \sigma_n) = -2 \sum_{j=1}^{n} (J_{ij} + J_{ji}) \sigma_i \sigma_j - 2 h_i \sigma_i = -4 \sum_{j=1}^{n} J_{ij} \sigma_i \sigma_j - 2 h_i \sigma_i \tag{1.22}$$

## 1.2 Applications

Many combinatorial optimization problems can be embedded into a QUBO problem. Those include all of Karp's 21 NP-complete problems. [33] This embedding refers to the fact that the original combinatorial optimization problem can be reformulated into a QUBO task in such a way that the solution of QUBO task encodes the solution of the original optimization problem. In the following subsections, some examples of these embeddings are provided. Namely, those are MAX-CUT problem, Number partitioning problem, and Traveling salesman problem. Many more examples can be found in [33].

There is one notable difference among these embeddings of different combinatorial optimization problems. As an example, let us consider the Traveling salesman problem (TSP). In order to embed this problem into the QUBO formulation, we need to introduce some *ancillary* variables. So, for a TSP problem with $n$ variables (cities), we would need to solve the QUBO task with $n^2$ variables. Nevertheless, the number of variables required for embedding all of Karp's 21 NP-complete problems is at most cubic with respect to the number of variables in the original problem [33]. Additionally to ancillary variables, we will need to impose some constraints on those variables i.e. we will have to solve a quadratic binary optimization task which is *not unconstrained*. Fortunately, these constraints can often be included into the QUBO task as well, in a sense that every feasible solution of QUBO which does not satisfy required constraints cannot be a minimum of the

QUBO task. Although theoretically perfectly valid, these constraints can however impose problems in practice because algorithms for solving QUBO often do not necessarily provide *the minimum*, but rather an *approximate* solution. It is very problem-specific and also algorithm-specific to determine how efficient those embedded constraints are, and whether there are some other ways to embed them which would be more efficient.

## MAX-CUT

This section is based on [5].

MAX-CUT is a combinatorial optimization problem on graphs which can be directly modeled with QUBO task. Consider an undirected weighted graph $G = (V, E)$ with no self-loops and no multiple edges between vertices. $V$ denotes the set of vertices, $E$ the set of edges. Each edge $e \in E$ is a triplet of the form $e = (a, b, J_{ab})$ where vertices $a, b \in V$ are connected with an edge with weight $J_{ab}$. If the number of vertices is $n = |V|$ then all weights can be stored in a symmetric adjacency matrix $J \in M_n(\mathbb{R})$. The fact that vertices $a, b \in V$ are connected with an edge having weight $w$ is represented in adjacency matrix by $J_{ab} = J_{ba} = w$. Throughout this section, only graphs of this type will be considered.

In order to describe the MAX-CUT problem we need a precise definition of what the *graph cut* means.

**Definition 1.2.1.** *Given a graph $G = (V, E)$ with adjacency matrix $J$, a* (graph) cut *is a bipartition $\mathcal{P} = \{A, B\}$ of the vertices (meaning that $A, B \subset V; A \cap B = \emptyset; A \cup B = V$).*

*The* value *of a certain graph cut $\mathcal{P} = \{A, B\}$ is a number*

$$\sum_{a \in A, b \in B} J_{ab} \tag{1.23}$$

*A cut is* maximum *if no other cut produces greater value.*

The MAX-CUT problem is then straightforward

**Problem 1.2.2** (MAX-CUT problem)**.** *Given a graph G, determine its maximum cut.*

MAX-CUT problem is a NP-hard problem.
There is a similar version with *yes-no* solution

**Problem 1.2.3** (MAX-CUT problem, binary)**.** *Given a graph G and a value k, determine whether or not there exists a cut with value at least k in G.*

This version is a NP-complete problem and is on the Karp's list of 21 NP-complete problems.

**Remark 1.2.4.** *A dual definition of "MIN-CUT" of a graph could be provided and so the problem of finding a MIN-CUT would be equivalent to the problem of finding a MAX-CUT by just taking the opposite sign of each edge weight. However, in the literature a MIN-CUT is usually considered only for weighted graphs whose weights are strictly positive and with the requirement that the cut is not trivial i.e.* $A, B \neq \emptyset$ *(otherwise a trivial cut* $A = V, B =$ *would always be the solution). Under these restrictions, a MIN-CUT problem becomes P instead of NP-hard. It is in fact a dual problem to the max-flow problem i.e. finding the maximum flow from source to sink (source and sink are newly added vertices) which is as well solvable in polynomial time. Here, only graphs with arbitrary real weights are considered, so in order to stay aligned with nomenclature in the literature, only MAX-CUT problem will be considered, as defined in 1.2.1.*

Let us formulate the MAX-CUT problem as a QUBO task. Suppose a graph $G = (V, E)$ is given, with adjacency matrix $J$ and the set of vertices being $V = \{1, \dots, n\}$ . Assume that the graph is weighted, has no self-loops, and is undirected. For a given cut $\mathcal{P} = \{A, B\}$, define a vector $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_n) \in \{-1, +1\}^n$ such that $\sigma_i = -1$ if $i \in A$, and $\sigma_i = +1$ if $i \in B$. It is clear that this mapping is a one-to-one correspondence between all possible cuts and all vectors $\{-1, +1\}^n$. Furthermore, define

$$Q(\boldsymbol{\sigma}) \coloneqq \frac{1}{4} \sum_{i,j=1}^{n} J_{ij}(1 - \sigma_i \sigma_j) \tag{1.24}$$

which is a quadratic polynomial over the domain $\{-1, +1\}^n$. For a particular cut $\mathcal{P} = \{A, B\}$, plugging in $\boldsymbol{\sigma}$ corresponding to $\mathcal{P}$, the value of $Q(\boldsymbol{\sigma})$ becomes precisely the value of that cut. To see this, note that $(1 - \sigma_i \sigma_j)$ is equals to 2 in case that $i$ and $j$ belong to different parts of the partition (i.e. $i \in A$, $j \in B$ or $i \in B$, $j \in A$) while it is 0 otherwise.

The conclusion is that finding a MAX-CUT of the graph can be embedded into solving a QUBO task given by

$$\max_{\boldsymbol{\sigma} \in \{-1,+1\}^n} Q(\boldsymbol{\sigma})$$

with $Q$ defined by 1.24.

Note that up to a constant term (which is irrelevant for maximization task), $Q$ is a homogeneous polynomial. The number of variables that we required to embed a MAX-CUT problem into a QUBO task is $O(n)$, where $n$ is the number of vertices in the graph.

On the other hand, consider an arbitrary QUBO problem given by

$$\max_{\boldsymbol{\sigma} \in \{-1,+1\}^n} \boldsymbol{\sigma}^T J \boldsymbol{\sigma} + \boldsymbol{\sigma}^T \mathbf{h} + c \tag{1.25}$$

As discussed in subsection 1, this is equivalent to solving the QUBO task with a corresponding homogeneous polynomial with $n + 1$ variables. In this correspondence, all candidate solution values are preserved. Thus, we may assume that $\mathbf{h} = \mathbf{0}$ and $c = 0$ without loss

of generality. Let us define an undirected graph $G = (V, E)$ with adjacency matrix given by $-4J$. Up to a constant term, all the cuts of this graph and their values correspond to the evaluation of the given quadratic polynomial, as seen in 1.24. Thus, solving a QUBO problem can be embedded into a MAX-CUT problem.

These two statements are summarized in the following remark.

**Remark 1.2.5.** *QUBO problem and MAX-CUT problem are equivalent in a sense that given an arbitrary instance of one of these problems with n variables, one can formulate an instance of the other problem with at most $n + 1 = O(n)$ variables in such a way that there is a mutual correspondence between candidate solutions as well as between values produced by these candidate solutions.*

Thus, statements and techniques for solving MAX-CUT can almost always be applied to QUBO, and vice versa.

## Number Partitioning

Number partitioning is the following combinatorial problem.

**Problem 1.2.6** (Number Partitioning)**.** *A sequence of n real numbers $a_1, \dots, a_n$ is given. Determine whether or not there exists a bipartition $\mathcal{P} = \{A, B\}$ of those numbers ($A, B \subset \{1, \dots, n\}; A \cup B = \{1, \dots, n\}; A \cap B = \emptyset$) such that $\sum_{i \in A} a_i = \sum_{i \in B} a_i$.*
*If such a bipartition exists, determine it.*

For example, can we divide a set of assets with values $a_1, \dots, a_n$ fairly between two people?
This problem is known to be NP-complete [33].

To formulate this problem in terms of a QUBO task, let us define the following quadratic polynomial with $n$ variables over the domain $\{-1, +1\}^n$.

$$Q(\sigma_1, \dots, \sigma_n) \coloneqq (\sum_{i=1}^{n} a_i \sigma_i)^2 \tag{1.26}$$

There is a one-to-one correspondence between each bipartition $\mathcal{P} = \{A, B\}$ and the domain $\{-1, +1\}^n$ given by the rule: $\sigma_i = -1$ if $i \in A$, $\sigma_i = +1$ if $i \in B$. It is clear that a solution of the number partitioning problem exists if and only if there exists a configuration $(\sigma_1, \dots, \sigma_n) \in \{-1, +1\}^n$ that evaluates to $Q(\sigma_1, \dots, \sigma_n) = 0$. Additionally, those solutions are in correspondence with minimizers of $Q$ by the described rule. Evaluation $Q(\sigma_1, \dots, \sigma_n) = 0$ means that the minimum of $Q$ has been found and it is equals to 0. Thus, minimizing $Q$ is equivalent to solving the Number partitioning problem.

If there does not exist a solution to the Number partitioning problem, then one might want to find a bipartition which is the *closest* possible to the fair partition. Minimizing $Q$ again solves this problem.

The polynomial $Q$ is generally not homogeneous. The number of variables required for encoding the number partitioning problem into a QUBO task is linear with respect to the problem size, i.e. $O(n)$, where $n$ is the number of assets that are being partitioned.

## Traveling Salesman Problem

There are many variants of the traveling salesman problem. We will present the following version and it's QUBO formulation. Many other variants can be formulated as QUBO task in a similar fashion.

**Problem 1.2.7** (Traveling Salesman Problem (TSP))**.** *A weighted directed complete graph $G = (V, E)$ is given. Edges of the form $(a, b, W_{ab}) \in E$ represent that there is an edge from $a$ to $b$ with weight $W_{ab}$. All weights are positive. The fact that $G$ is directed here refers to the possibility that $W_{ab} \neq W_{ba}$. Let us assume that vertices are numbered $V = (1, \ldots, n)$. A* tour *$v_1, \ldots, v_n$ is a permutation of vertices $V$ representing the order for visiting each vertex, that is $v_1, v_2, \ldots, v_n, v_1$ because one wants to return to the starting vertex.*
*The task is to find a tour which minimizes the sum of traversed edges i.e. the sum*

$$\sum_{i=1}^{n-1} W_{v_i, v_{i+i}} + W_{v_n, v_1}$$

*which will often be refered to as* tour length*.*

One example of the traveling salesman problem is: how to plan the route for a delivery vehicle given that it needs to visit each of the specified locations once, while minimizing the total path length and thus minimizing the transport costs.

Let us present a QUBO formulation of the TSP. Since the route length does not depend on the starting vertex, we may choose vertex $a = 1$ as the starting one. Define $(n-1)^2$ variables $s_{i,u} \in \{0, 1\}$ where $i \in \{2, \ldots, n\}$ represents the step and $u \in V \setminus \{a\}$ represents the vertex to be visited in $i$-th step. $s_{i,u} = 1$ means that we want to visit vertex $u$ at step $i$, while $s_{i,u} = 0$ otherwise. Since we want a permutation of vertices, we want exactly one $s_{i,u} = 1$ per turn $i$, and all others to be $s_{i,u} = 0$. The same should hold per each vertex $u \in V \setminus \{a\}$. Let us thus define the following quadratic polynomial

$$Q_A(\mathbf{s}) \coloneqq \sum_{i=2}^{n} (1 - \sum_{u \in V \setminus \{a\}} s_{i,u})^2 + \sum_{u \in V \setminus \{a\}} (1 - \sum_{i=2}^{n} s_{i,u})^2 \tag{1.27}$$

We have that $Q_A(\mathbf{s}) = 0$ if and only if $\mathbf{s}$ represents a permutation of vertices as described above, and $Q_A(\mathbf{s}) \geq 1$ otherwise.

Now define the following quadratic polynomial

$$Q_B(\mathbf{s}) \coloneqq \sum_{u \in V \setminus \{a\}} W_{a,u} s_{2,u} + \sum_{i=3}^{n} \sum_{u \in V \setminus \{a\}} \sum_{v \in V \setminus \{a\}} W_{u,v} s_{i-1,u} s_{i,v} + \sum_{u \in V \setminus \{a\}} W_{u,a} s_{n,u} \tag{1.28}$$

We have that $Q_B \geq 0$. Considering only those configurations $\mathbf{s}$ that already represent a valid permutation as described above, the value $Q_B(\mathbf{s})$ is precisely the length of the tour corresponding to the given permutation. Now define

$$Q(\mathbf{s}) \coloneqq C \cdot Q_A(\mathbf{s}) + Q_B(\mathbf{s}) \tag{1.29}$$

for properly chosen constant $C > 0$. It should be chosen in such a way that any configuration $\mathbf{s}$ which violates the above requirements for producing a valid permutation, evaluates to suboptimal $Q(\mathbf{s})$. For example, for any $\mathbf{s}$ corresponding to a valid permutation of vertices, $Q_B(\mathbf{s}) \leq n \cdot \max_{u,v \in V} W_{u,v}$. Choosing any $C > n \cdot \max_{u,v \in V} W_{u,v}$ as a constant will be sufficient. Indeed, for any $\mathbf{s}$ which does not correspond to a valid permutation, the value $Q(\mathbf{s})$ will be higher than every possible tour length, so any valid route encoding $\mathbf{s}'$ will provide $Q(\mathbf{s}') < Q(\mathbf{s})$. Minimizing $Q$ thus simplifies to minimizing $Q_B$ over the set of all valid permutations. This is exactly minimizing the tour length from the traveling salesman problem.

Since Q is a quadratic polynomial over the domain $\{0,1\}^{(n-1)^2}$, we have showed that solving a traveling salesman problem can be embedded into solving a minimization QUBO task. The number of variables required for embedding a traveling salesman problem into a QUBO task is $O(n^2)$ where $n$ is the number of vertices in the graph.

# Chapter 2

# Preliminary theory

This chapter is a brief recapitulation of certain well-known topics in mathematics. Its first purpose is to give a mathematical background for dynamical systems, along with a few examples which will be essential for understanding the behaviour of dynamical systems in chapter 3. Its second purpose is to serve as a reference for some arguments used in chapter 3.

In terms of notation, a time derivative is denoted by a dot, $\dot{x} = \frac{dx}{dt}$. Column vectors in $\mathbb{R}^n$ are denoted by bold letters, while their components are denoted with regular letters and a subscript denoting their index of component $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = (x_1, \ldots, x_n)^T$. The superscript $T$ denotes a transpose. For a vector function $\mathbf{f} : \Omega \subset \mathbb{R}^n \to \mathbb{R}^n$, its *matrix derivative* or *differential* is considered to be a matrix of its partial derivatives arranged as

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

So, for a scalar function $E : \mathbb{R}^n \to \mathbb{R}$, $\frac{\partial R}{\partial \mathbf{x}}$ is a row vector. Its *second differential* is considered to be a matrix

$$\frac{\partial^2 E}{\partial \mathbf{x}^2} = \begin{bmatrix} \frac{\partial^2 E}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 E}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 E}{\partial x_n \partial x_n} \end{bmatrix}$$

Throughout the thesis, the second differential will always be taken of $C^2$ functions, so the matrix $\frac{\partial^2 E}{\partial \mathbf{x}^2}$ will always be symmetric due to the Schwarz's theorem.

Matrix derivations of higher order and shape could be defined as well, but will not be used in the thesis.

## 2.1 Dynamical systems

In this section, the results and concepts from the theory of dynamical systems are briefly given. All systems discussed are continuous dynamical systems meaning that the time is continuous and they are governed by ordinary differential equations.

Let $I \subset \mathbb{R}, U \subset \mathbb{R}^n$ be open sets, and $\mathbf{f} \in C^1(I \times U; \mathbb{R}^n)$. A *(continuous) dynamical system* is an ordinary differential equation of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}) \tag{2.1}$$

where $\mathbf{f}$ is a given vector field. When *initial condition* $\mathbf{x}(0) = \mathbf{x}_0$ is specified, we call such system

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}) \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \tag{2.2}$$

an *initial value problem*.

The following two theorems are fundamental for dynamical systems in a sense that given a vector field, and the initial condition, the trajectory of the dynamical system exists and is uniquely determined. Moreover, this solution can be extended in time until it either reaches the domain boundary or blows up to infinity.

**Theorem 2.1.1** (Existence-Uniqueness)**.** *[18] Let $t_0 \in I \subset \mathbb{R}, \mathbf{x}_0 \in U \subset \mathbb{R}^n$ be open sets, and $\mathbf{f} \in C^1(I \times U; \mathbb{R}^n)$. Then there exist open subsets $t_0 \in I_0 \subset I, \mathbf{x}_0 \in U_0 \subset U$ and a function $\phi : I_0 \times i_0 \times U_0 \to \mathbb{R}^n$ such that for each $(t_1, \mathbf{x}_1) \in I_0 \times U_0$, the function $t \mapsto \phi(t, t_1, \mathbf{x}_1)$ is the unique solution defined on $I_0$ of the initial value problem* (2.2).

**Theorem 2.1.2** (Extension)**.** *[18] Let $t_0 \in I \subset \mathbb{R}, \mathbf{x}_0 \in U \subset \mathbb{R}^n$ be open sets, and $\mathbf{f} \in C^1(I \times U; \mathbb{R}^n)$. Let $\langle \alpha, \beta \rangle$ be the maximal open interval where the solution $\mathbf{x}$ of dynamical system* (2.2) *exists, with $-\infty \leq \alpha \leq \beta \leq \infty$. Then either $|\mathbf{x}(t)|$ approaches $\infty$ or $\mathbf{x}(t)$ approaches the boundary of $U$ as $t \to \beta$.*

When $\mathbf{f}$ does not depend on the time variable $t$, we call such dynamical system an *autonomous dynamical system*. Otherwise, we call it a *nonautonomous dynamical system*.

When $\mathbf{f}(t, \mathbf{x}) = A\mathbf{x}$ for some constant matrix $A \in M_n(\mathbb{R})$, we say that the dynamical system is *linear*. Otherwise we call it a *nonlinear dynamical system*.

**Remark 2.1.3.** *Nonautonomous dynamical systems are often referred to as a generalization of autonomous ones, so if we do not care if $\mathbf{f}$ depends on $t$ or not, we may just say a* nonautonomous dynamical system.

*Analogously, nonlinear dynamical systems are a generalization of linear ones.*

Consider a nonautonomous dynamical system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}) \tag{2.3}$$

**Definition 2.1.4.** *An* equilibrium point *of a nonautomous dynamical system is a point* $\mathbf{x}_0 \in \Omega$ *which is also a trajectory of the system, meaning that* $\mathbf{x}(t) \coloneqq \mathbf{x}_0$ *is the solution of* (2.3).

The definition of equilibrium point is equivalent to the fact that $\mathbf{f}(t, \mathbf{x}_0) = \mathbf{0}, \forall t$. When studying a dynamical system, it is often useful to look for its equilibrium points. The character of these points can often tell us about the qualitative behavior of the dynamical system in the neighborhood of these points. The main result about this is the Hartman-Grobman theorem for nonlinear autonomous dynamical systems.

## The Hartman-Grobman theorem

Consider a nonlinear autonomous dynamical system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}) \tag{2.4}$$

where $\mathbf{f} \in C^1(U; \mathbb{R}^n), U \subset \mathbb{R}^n$.

**Definition 2.1.5.** *For each* $\mathbf{x}_0 \in U$ *let* $\mathbf{x}$ *be the solution of 2.4 with initial condition* $\mathbf{x}(0) = \mathbf{x}_0$. *Let us define a one-parameter family of functions* $\phi_t$ *called the* flow *of dynamical system 2.4 as*

$$\phi_t(\mathbf{x}_0) = \mathbf{x}(t)$$

*for every t in which* $\mathbf{x}$ *is defined.*

**Proposition 2.1.6.** *The flow* $\phi_t$ *of the dynamical system 2.4 has the following properties*

*a)* $\phi_{t+s} = \phi_t \circ \phi_s$

*b)* $\phi_0(\mathbf{x}) = \mathbf{x}$

*whenever both sides of the equation are defined.*

Let us assume that the dynamical system 2.4 has an equilibrium point at $\mathbf{x}_0$.

The behavior of a nonlinear autonomous dynamical system near an equilibrium point $\mathbf{x}_0$ is qualitatively the same as the dynamics of the corresponding *linearization* i.e.

$$\dot{\mathbf{x}}(t) = A\mathbf{x} \tag{2.5}$$

where $A = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_0)$. That is formally stated in the Hartman-Grobman theorem.

**Theorem 2.1.7** (The Hartman-Grobman Theorem [18])**.** *Let* $\mathbf{f} \in C^1(\mathbb{R}^n; \mathbb{R}^n)$ *be a smooth vector field, and let* $\phi_t$ *be the flow of the nonlinear system* (2.4). *Suppose that* $\mathbf{x}_0$ *is a hyperbolic equilibrium point, meaning that* $\mathbf{f}(\mathbf{x}_0) = \mathbf{0}$ *and none of the eigenvalues of* $A = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_0)$ *have zero real part. Let* $\psi_t$ *be the flow of the corresponding linearization* (2.5).
*Then there exist open sets* $\mathbf{x}_0 \in U \subset \mathbb{R}^n, \mathbf{0} \in V \subset \mathbb{R}^n$ *and a homeomorphism* $H : U \to V$ *such that* $H(\phi_t(\mathbf{x})) = \psi_t(H(\mathbf{x}))$ *whenever* $\mathbf{x} \in U$ *and both sides of the equation are defined.*

## A connection between autonomous and nonautonomous systems

In chapter 3, we will be dealing with nonautonomous dynamical systems. However, the vector field $\mathbf{f}$ will *slowly* vary in time, meaning that $\frac{\partial \mathbf{f}}{\partial t}$ has relatively small norm. In this case, if we want to analyze the local (in time) behavior of the dynamical system, it might be tempting to analyze the behaviour of the dynamical system which has its vector field *frozen in time*, i.e. the dynamical system

$$\dot{\mathbf{x}} = \mathbf{f}(t_0, \mathbf{x}) \tag{2.6}$$

where $t_0$ is some time instance at which the vector field is frozen.

For analyzing the global (in time) behavior of the system, this strategy is inadequate and even misleading, as seen from the following example.

**Example 2.1.8.** *Consider a one-dimensional nonautonomous dynamical system*

$$\dot{x}(t) = t - x(t) \tag{2.7}$$

*If we were to analyze this system by observing the system which has its vector field frozen in time, that is*

$$\dot{x}(t) = t_0 - x(t) \tag{2.8}$$

*for some $t_0 \in \mathbb{R}$, we would see that it has equilibrium point at $x = t_0$. The linearization of such system is*

$$\dot{x}(t) = -x(t) \tag{2.9}$$

*which by Hartman-Grobman theorem (or by solving this directly) would imply that it is an attractive equilibrium. It might be tempting to conclude that the system follows the trajectory given by instantaneous equilibrium points, that is the trajectory $x_{ISP}(t) = t$.*

*However, the exact solutions of this equation are $x(t) = Ce^{-t} + t - 1$ for some $C \in \mathbb{R}$. It is clear that all of these solutions converge to the solution $x_{DHT}(t) = t - 1$ as $t \to \infty$.*

*First of all, $x_{ISP}(t)$ is not even a solution of the above differential equation. Second, and even more important, the system globally behaves in such a way that it converges towards the trajectory $x_{DHT}(t)$, and not $x_{ISP}(t)$.*

In [26] they develop the concept of a *distinguished hyperbolic trajectory* which in the above example is precisely $x_{\mathrm{DHT}}(t)$, contrary to the *instantaneous stagnation points* which are $x_{\mathrm{ISP}}(t)$ in this example.

However, the global (in time) behavior of the dynamical system will not be of great interest in chapter 3, but rather a local one. In this case, analyzing the system as if its vector field was frozen in time makes sense. This concept is formalized in the following.

Let $\mathbf{f} \in C^1(\mathbb{R} \times \mathbb{R}^n; \mathbb{R}^n)$ be a given vector field, $\mathbf{x}_0 \in \mathbb{R}^n$, $t_0 < t_1$.
Consider the following dynamical system

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t)) \\ \mathbf{x}(t_0) = \mathbf{x}_0 \end{cases} \tag{2.10}$$

and the corresponding dynamical system with vector field frozen at time instance $t_0$, with the same initial conditions.

$$\begin{cases} \dot{\mathbf{y}}(t) &= \mathbf{f}(t_0, \mathbf{y}(t)) \\ \mathbf{y}(t_0) &= \mathbf{x}_0 \end{cases} \tag{2.11}$$

For vector $\mathbf{v} \in \mathbb{R}^n$ we will denote the norm $\|\mathbf{v}\|_\infty = \max_{i=1,\dots,n} |v_i|$, and for a vector function $\mathbf{g} : \Omega \subset \mathbb{R}^m \to \mathbb{R}^n$ we will denote the norm

$$\|\mathbf{g}\|_{L^\infty(\Omega)} \coloneqq \max_{i=1,\dots,n} \|g_i\|_{L^\infty(\Omega)} \tag{2.12}$$

Let us formally define the successive approximations as

$$\begin{aligned} \mathbf{x}_0(t) &\coloneqq \mathbf{x}_0 \\ \mathbf{x}_{k+1}(t) &\coloneqq \mathbf{x}_0 + \int_{t_0}^{t} f(\tau, \mathbf{x}_k(\tau)) d\tau \\ \mathbf{y}_0(t) &\coloneqq \mathbf{x}_0 \\ \mathbf{y}_{k+1}(t) &\coloneqq \mathbf{x}_0 + \int_{t_0}^{t} f(t_0, \mathbf{y}_k(\tau)) d\tau \end{aligned} \tag{2.13}$$

**Theorem 2.1.9** (Connection between autonomous and nonautonomous dynamical systems)**.** *Let* $\mathbf{f} \in C^1(\mathbb{R} \times \mathbb{R}^n; \mathbb{R}^n), \mathbf{x}_0 \in \mathbb{R}^n, t_0 < t_1$. *Assume that all successive approximations* (2.13) *for dynamical systems* (2.10) *and* (2.11) *are well defined and continuous on* $[t_0, t_1]$ *and that they uniformly converge to solutions*

$$\mathbf{x}_k \xrightarrow{L^\infty([t_0,t_1])} \mathbf{x} \tag{2.14}$$

$$\mathbf{y}_k \xrightarrow{L^\infty([t_0,t_1])} \mathbf{y} \tag{2.15}$$

*Assume that* $D \subset \mathbb{R}^n$ *is a convex compact set which contains both solutions and all successive approximations* $\mathbf{x}([t_0, t_1]), \mathbf{x}_k([t_0, t_1]), \mathbf{y}([t_0, t_1]), \mathbf{y}_k([t_0, t_1]) \subset D, \forall k$. *Let* $L > 0$ *be a Lipschitz constant in second variable for* $\mathbf{f}$ *in a sense that*

$$\|\mathbf{f}(t, \mathbf{x}) - \mathbf{f}(t, \mathbf{y})\|_\infty \leq L \|\mathbf{x} - \mathbf{y}\|_\infty , \quad \forall t \in [t_0, t_1], \forall \mathbf{x}, \mathbf{y} \in D \tag{2.16}$$

*Let*

$$M \coloneqq \left\| \frac{\partial \mathbf{f}}{\partial t} \right\|_{L^\infty([t_0,t_1] \times D)} \tag{2.17}$$

*Then,*

$$\|\mathbf{x}-\mathbf{y}\|_{L^\infty([t_0,t])} \le \frac{M}{L^2}e^{L(t-t_0)} - \frac{M}{L}(t-t_0) - \frac{M}{L^2}, \quad \forall t \in [t_0,t_1] \tag{2.18}$$

*Proof.* First, let's show that $\mathbf{f}$ is Lipschitz continuous in second variable with some constant $L > 0$ in a sense of (2.16). By following the proof of proposition 2.3.4 and applying it to each component of $\mathbf{f}$, we get that $f_1, \dots, f_n$ are Lipschitz continuous in second variable. From here, (2.16) follows easily.

By inductive argument we prove a bound for $\left\|\mathbf{x}_k(t) - \mathbf{y}_k(t)\right\|_\infty$. For $k = 0$ we have $\left\|\mathbf{x}_0(t) - \mathbf{y}_0(t)\right\|_\infty = 0$. Assume that for some $k$ it holds

$$\left\|\mathbf{x}_k(t) - \mathbf{y}_k(t)\right\|_\infty \le M\sum_{i=2}^{k+1}\frac{L^{i-2}}{i!}(t-t_0)^i, \quad \forall t \in [t_0,t_1] \tag{2.19}$$

Then,

$$\begin{aligned}
\left\|\mathbf{x}_{k+1}(t) - \mathbf{y}_{k+1}(t)\right\|_\infty &\le \int_{t_0}^{t}\left\|\mathbf{f}(\tau,\mathbf{x}_k(\tau)) - \mathbf{f}(t_0,\mathbf{y}_k(\tau))\right\|_\infty d\tau && (2.20)\\
&= \int_{t_0}^{t}\left\|\mathbf{f}(\tau,\mathbf{x}_k(\tau)) - \mathbf{f}(\tau,\mathbf{y}_k(\tau)) + \mathbf{f}(\tau,\mathbf{y}_k(\tau)) - \mathbf{f}(t_0,\mathbf{y}_k(\tau))\right\|_\infty d\tau && (2.21)\\
&\le \int_{t_0}^{t} L\left\|\mathbf{x}_k(\tau) - \mathbf{y}_k(\tau)\right\|_\infty + M(\tau-t_0)d\tau && (2.22)\\
&\le \int_{t_0}^{t} LM\sum_{i=2}^{k+1}\frac{L^{i-2}}{i!}(\tau-t_0)^i + M(\tau-t_0)d\tau && (2.23)\\
&= LM\sum_{i=2}^{k+1}\frac{L^{i-2}}{i!}\int_{t_0}^{t}(\tau-t_0)^i d\tau + \int_{t_0}^{t}M(\tau-t_0)d\tau && (2.24)\\
&= M\sum_{i=2}^{k+1}\frac{L^{i-1}}{(i+1)!}(t-t_0)^{i+1} + \frac{M}{2}(t-t_0)^2 && (2.25)\\
&= M\sum_{i=2}^{k+2}\frac{L^{i-2}}{i!}(t-t_0)^i && (2.26)
\end{aligned}$$

for arbitrary $t \in [t_0,t_1]$. In order to get (2.22), we used triangle inequality, mean value theorem, Lipschitz condition (2.16) and bound for the partial derivative of $\mathbf{f}$ by $t$ (2.17).

Now, by induction we have that (2.19) holds for each $k$. Thus, for all $k$ we have

$$\left\|\mathbf{x}_k - \mathbf{y}_k\right\|_{L^\infty([t_0,t])} \leq M \sum_{i=2}^{k+1} \frac{L^{i-2}}{i!}(t-t_0)^i \tag{2.27}$$

$$\leq M \sum_{i=2}^{\infty} \frac{L^{i-2}}{i!}(t-t_0)^i \tag{2.28}$$

$$= \frac{M}{L^2} \sum_{i=0}^{\infty} \frac{L^i}{i!}(t-t_0)^i - \frac{M}{L}(t-t_0) - \frac{M}{L^2} \tag{2.29}$$

$$= \frac{M}{L^2} e^{L(t-t_0)} - \frac{M}{L}(t-t_0) - \frac{M}{L^2}, \quad \forall t \in [t_0, t_1] \tag{2.30}$$

Since

$$\mathbf{x}_k \xrightarrow{L^\infty([t_0,t_1])} \mathbf{x}, \quad k \to \infty \tag{2.31}$$

$$\mathbf{y}_k \xrightarrow{L^\infty([t_0,t_1])} \mathbf{y}, \quad k \to \infty \tag{2.32}$$

we get the bound

$$\|\mathbf{x} - \mathbf{y}\|_{L^\infty([t_0,t])} \leq \frac{M}{L^2} e^{L(t-t_0)} - \frac{M}{L}(t-t_0) - \frac{M}{L^2}, \quad \forall t \in [t_0, t_1] \tag{2.33}$$

□

## Gradient dynamical systems

A special type of dynamical systems are *gradient dynamical systems*. The name is derived by the fact that the vector field is negative gradient of some landscape function. More formally, let $\Omega \subset \mathbb{R}^n, I \subset \mathbb{R}$ be open sets and $E \in C^2(I \times \Omega; \mathbb{R})$. The dynamical system

$$\dot{\mathbf{x}}(t) = -\frac{\partial E}{\partial \mathbf{x}}^T (t, \mathbf{x}) \tag{2.34}$$

is a *gradient dynamical system* with corresponding *landscape* function $E$.

Generally, the function $E$ can vary over time. However, let us analyze an example where $E$ is time-independent, i.e. $E = E(\mathbf{x})$.

**Example 2.1.10.** *Let $A \in M_n(\mathbb{R})$ be a real symmetric matrix and $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x}$. Consider a gradient dynamical system with landscape function $E$.*

$$\begin{aligned} \dot{\mathbf{x}}(t) &= -\frac{\partial E}{\partial \mathbf{x}}^T (\mathbf{x}(t)) \\ &= -A\mathbf{x} \end{aligned} \tag{2.35}$$

*Since A is symmetric, it is orthogonally diagonalizable, so let* $U^T AU = \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ *be its orthogonal diagonalization.*

*Define a change of variables* $\bar{\mathbf{x}} \coloneqq U^T \mathbf{x}$*. The equation* (2.35) *becomes*

$$\begin{aligned} \dot{\bar{\mathbf{x}}} &= -U^T AU\bar{\mathbf{x}} \\ &= -\Lambda\bar{\mathbf{x}} \end{aligned} \tag{2.36}$$

*which is a system of decoupled ordinary differential equations in each component. Written out component-wise, the system is*

$$\dot{\bar{x}}_i = -\lambda_i \bar{x}_i; \quad i = 1, \ldots, n \tag{2.37}$$

*The solution for each component is thus*

$$\bar{x}_i(t) = \bar{x}(0)e^{-\lambda_i t}; \quad i = 1, \ldots, n \tag{2.38}$$

*where* $\bar{x}(0)$ *is the initial condition. The solution in the original reference frame is then* $\mathbf{x}(t) = U\bar{\mathbf{x}}(t)$*.*

*This change of reference provides us not only with the exact solution, but also with a qualitative understanding of solution's behavior. For each component we have different behavior depending on the sign of the corresponding eigenvalue. For* $\lambda_i > 0$ *the solution component rapidly collapses towards* 0*. For* $\lambda_i = 0$*, the solution component is stationary. For* $\lambda_i < 0$ *the solution component rapidly expands to* $\pm\infty$*.*

*This gives us the qualitative understanding of the behavior of* $\mathbf{x}(t)$*. When viewed in reference frame which diagonalizes A, the solution's components either collapse, stay stationary, or expand, all with respect to the sign of the corresponding eigenvalue.*

## Hamiltonian dynamical systems

Let $I \subset \mathbb{R}, \Omega \subset \mathbb{R}^{2n}$ be open sets and $H \in C^2(I \times \Omega)$ where $H = H(t, \mathbf{x}, \mathbf{y})$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. A system of the form

$$\begin{aligned} \dot{\mathbf{x}} &= \frac{\partial H}{\partial \mathbf{y}}^T (t, \mathbf{x}, \mathbf{y}) \\ \dot{\mathbf{y}} &= -\frac{\partial H}{\partial \mathbf{x}}^T (t, \mathbf{x}, \mathbf{y}) \end{aligned} \tag{2.39}$$

is called a Hamiltonian dynamical system.

Function $H$ is called a *Hamiltonian function* or *total energy* of the system.

Hamiltonian $H$ can take various forms.

The motion of the particles in space can be modeled by taking $H$ to be of the following form

$$H(t, \mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \frac{m_i}{2} y_i^2 + E(t, \mathbf{x}) \tag{2.40}$$

Each particle has certain number of degrees of freedom (for example 1D, 2D or 3D motion). Each degree of freedom has its corresponding *position* $x_i$ and *momentum* $y_i$. The quantity $\sum_{i=1}^{n} \frac{m_i}{2} y_i^2$ is called the *kinetic energy* of the system, while $E(t, \mathbf{x})$ is a function called *potential energy* of the system. Kinetic energy depends only on constants $m_i$ which represent the *mass* of each particle, and momenta $y_i$. On the other hand, the potential energy depends only on the position vector $\mathbf{x}$ and the time instance in case that the potential energy changes over time.

**Example 2.1.11.** *Let $A \in M_n(\mathbb{R})$ be a real symmetric matrix and $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x}$. Consider an autonomous Hamiltonian dynamical system corresponding to the motion of particles, each with mass m, in time-independent potential $E(\mathbf{x})$*

$$\begin{aligned} H(\mathbf{x}, \mathbf{y}) &= \sum_{i=1}^{n} \frac{m}{2} y_i^2 + \frac{1}{2}\mathbf{x}^T A\mathbf{x} \\ \dot{\mathbf{x}} &= \frac{\partial H^T}{\partial \mathbf{y}} \\ &= m\mathbf{y} \\ \dot{\mathbf{y}} &= -\frac{\partial H^T}{\partial \mathbf{x}} \\ &= -A\mathbf{x} \end{aligned} \tag{2.41}$$

*Since A is symmetric, it is orthogonally diagonalizable, so let $U^T AU = \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ be its orthogonal diagonalization.*

*Define a change of variables $\bar{\mathbf{x}} := U^T\mathbf{x}, \bar{\mathbf{y}} := U^T\mathbf{y}$. The dynamical system* (2.41) *becomes*

$$\begin{aligned} \dot{\bar{\mathbf{x}}} &= m\bar{\mathbf{y}} \\ \dot{\bar{\mathbf{y}}} &= -U^T AU\bar{\mathbf{x}} = -\Lambda\bar{\mathbf{x}} \end{aligned} \tag{2.42}$$

*or written out component-wise*

$$\begin{aligned} \dot{\bar{x}}_i &= m\bar{y}_i; \quad i = 1, \ldots, n \\ \dot{\bar{y}}_i &= -\lambda_i \bar{x}_i; \quad i = 1, \ldots, n \end{aligned} \tag{2.43}$$

*By taking the derivative of $\dot{\bar{x}}_i$ we obtain an equivalent equation*

$$\ddot{\bar{x}}_i = -m\lambda_i \bar{x}_i; \quad i = 1, \ldots, n \tag{2.44}$$

*This is a system of decoupled linear equations so for each component we have the solution depending on the sign of $\lambda_i$.*

*If $\lambda_i > 0$, then*

$$\bar{x}_i(t) = \frac{1}{\sqrt{m\lambda_i}}\dot{\bar{x}}_i(0)\sin\left(\sqrt{m\lambda_i}t\right) + \bar{x}_i(0)\cos\left(\sqrt{m\lambda_i}t\right) \tag{2.45}$$

*which means that this component periodically oscillates around 0.*

*If $\lambda_i = 0$, then*

$$\bar{x}_i(t) = \dot{\bar{x}}_i(0)t + \bar{x}_i(0) \tag{2.46}$$

*which means that this component linearly expands.*

*If $\lambda_i < 0$,*

$$\bar{x}_i(t) = \left(\frac{\bar{x}_i(0)}{2} + \frac{\dot{\bar{x}}_i(0)}{2\sqrt{-m\lambda_i}}\right)e^{\sqrt{-m\lambda_i}t} + \left(\frac{\bar{x}_i(0)}{2} - \frac{\dot{\bar{x}}_i(0)}{2\sqrt{-m\lambda_i}}\right)e^{-\sqrt{-m\lambda_i}t} \tag{2.47}$$

*which means that this component exponentially expands.*

*By changing the reference frame back to the original $\mathbf{x} = U\bar{\mathbf{x}}$, we can obtain the exact solution of the dynamical system. However, the qualitative behavior of this dynamical system can be understood by analyzing each solution component of the system in reference frame which diagonalizes A. Each component in this reference frame either oscillates around zero, linearly expands, or exponentially expands, with respect to the sign of the corresponding eigenvalue.*

## 2.2 Numerical simulations

In this section, two numerical methods for solving ordinary differential equations are presented. Those are Euler method and symplectic Euler method.

Euler method is a numerical method which can be used to approximately solve any initial-value problem and thus simulate any dynamical system. Although being very simple to state and implement, its weakness is being less precise than some other numerical methods such as various other Runge-Kutta methods.

Symplectic Euler method is a modification of the Euler method used for simulating special kind of ordinary differential equations, namely Hamiltonian dynamical systems. Although the error bound is of the same order as for the Euler method, symplectic Euler method conserves the energy (for time-independent Hamiltonian) much better than the Euler method. Thus, symplectic Euler method is more appropriate for simulating Hamiltonian dynamics.

Many other numerical methods for solving various ordinary differential equations can be found in [17] and [31], each with certain advantages and disadvantages. However, only these two methods will be used in chapter 3.

## Euler method

This section is based on [17]. Consider a given initial value problem

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(t, \mathbf{x}) \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \end{aligned} \tag{2.48}$$

The Euler method for approximately solving this initial value problem is given by the iterative formula

$$\begin{aligned} \mathbf{x}^{(0)} &= \mathbf{x}_0 \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \mathbf{f}(t_k, \mathbf{x}^{(k)}) \cdot \Delta_t, \quad k = 0, \ldots, N_{\text{iter}} - 1 \end{aligned} \tag{2.49}$$

where $\Delta_t = \frac{T}{N_{\text{iter}}}$, $t_k = k\Delta_t$, and $T$ is the time interval on which we need to approximate the solution. For each $k$, $\mathbf{x}^{(k)}$ is the approximate value for $\mathbf{x}(t_k)$.

Euler method is a first-order method, meaning that the bound for error between the approximate and exact solutions is proportional to $\Delta_t$, and is not proportional to any higher power of $\Delta_t$. Details can be found in [17].

## Symplectic Euler method

This section is based on [9]. Symplectic Euler method (sometimes also called semi-implicit Euler method) can be used for approximately solving the initial value problem of the form

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(t, \mathbf{y}) \\ \dot{\mathbf{y}} &= \mathbf{g}(t, \mathbf{x}) \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \\ \mathbf{y}(t_0) &= \mathbf{y}_0 \end{aligned} \tag{2.50}$$

This system typically arises in Hamiltonian dynamics if the Hamiltonian has separable variables

$$H(t, \mathbf{x}, \mathbf{y}) = E(t, \mathbf{x}) + T(t, \mathbf{y})$$

Often $E$ is the potential energy of the system, while $T$ is the kinetic energy.

Symplectic Euler method for approximately solving this system is given by the iterative formula

$$\begin{aligned} \mathbf{x}^{(0)} &= \mathbf{x}_0 \\ \mathbf{y}^{(0)} &= \mathbf{y}_0 \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \mathbf{f}(t_k, \mathbf{y}^{(k)}) \cdot \Delta_t \\ \mathbf{y}^{(k+1)} &= \mathbf{y}^{(k)} + \mathbf{g}(t_k, \mathbf{x}^{(k+1)}) \cdot \Delta_t, \quad k = 0, \ldots, N_{\text{iter}} - 1 \end{aligned} \tag{2.51}$$

where $\Delta_t = \frac{T}{N_{\text{iter}}}$, $t_k = k\Delta_t$, and $T$ is the time interval on which we need to approximate the solution. For each $k$, $\mathbf{x}^{(k)}$ and $\mathbf{y}^{(k)}$ are respectively approximate values for $\mathbf{x}(t_k)$ and $\mathbf{y}(t_k)$. The difference between original Euler method is that $\mathbf{x}^{(k+1)}$ is used for calculating $\mathbf{y}^{(k+1)}$, instead of $\mathbf{x}^{(k)}$.

A second variant of this method is given by *reversing the order of calculations*, i.e.

$$\begin{aligned}
\mathbf{x}^{(0)} &= \mathbf{x}_0 \\
\mathbf{y}^{(0)} &= \mathbf{y}_0 \\
\mathbf{y}^{(k+1)} &= \mathbf{y}^{(k)} + \mathbf{g}(t_k, \mathbf{x}^{(k)}) \cdot \Delta_t \\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \mathbf{f}(t_k, \mathbf{y}^{(k+1)}) \cdot \Delta_t, \quad k = 0, \ldots, N_{\text{iter}} - 1
\end{aligned} \tag{2.52}$$

Although symplectic Euler method is also a first-order method, i.e. the error is proportional to $\Delta_t$, it is usually a more appropriate method for simulating Hamiltonian dynamical systems compared to the original Euler method. This is due to the fact that symplectic Euler method conserves energy (for time-independent Hamiltonians) better than Euler method. In fact, Euler method often persistently increases the energy, making it less accurate. Details about this method can be found in [9] and [31].

## 2.3 Other

This section serves only as a reference for some arguments used in the rest of the thesis.

**Theorem 2.3.1** (Weyl's inequality [12])**.** *Let $M = N + R$, $N$, and $R$ be $n \times n$ Hermitian matrices, with their respective eigenvalues $\mu_i, \eta_i, \rho_i$ ordered in descending order as follows:*

$$\begin{aligned}
M: &\quad \mu_1 \geq \cdots \geq \mu_n, \\
N: &\quad \eta_1 \geq \cdots \geq \eta_n, \\
R: &\quad \rho_1 \geq \cdots \geq \rho_n
\end{aligned}$$

*Then the following inequalities hold:*

$$\eta_i + \rho_n \leq \mu_i \leq \eta_i + \rho_1, \quad i = 1, \ldots, n$$

**Proposition 2.3.2.** *Let $\Omega \subset \mathbb{R}^n$ be an open, convex and connected set. Let $E \in C^2(\Omega)$ such that the second differential $\frac{\partial^2 E}{\partial \mathbf{x}^2}$ is positive semidefinite, i.e. all of its eigenvalues are greater than 0, at each point of $\Omega$.*

*Then $E$ is a convex function.*

**Definition 2.3.3.** *[2] Consider a second-order differential equation*

$$\ddot{x}(t) = tx(t) \tag{2.53}$$

*Two linearly independent solutions of this equation are called* Airy functions *and are denoted by* $A(t), B(t)$.

**Proposition 2.3.4.** *Let* $D \subset \mathbb{R}^n$ *be a compact and convex set. Let* $f \in C^1(D)$.
*Then,* $f$ *is Lipscitz continuous.*

*Proof.* Choose arbitrary $\mathbf{x}, \mathbf{y} \in D$ and consider the line between them parametrized by $\gamma : [0,1] \to D, \gamma(t) = (1-t)\mathbf{x} + t\mathbf{y}$. Since $D$ is convex, $\gamma$ is indeed contained in $D$. Define $g : [0,1] \to \mathbb{R}, g = f \circ \gamma$. Then, by the mean value theorem there exists $\xi \in \langle 0, 1 \rangle$ such that

$$\frac{\partial f}{\partial \mathbf{x}}(\gamma(\xi)) \cdot (\mathbf{y} - \mathbf{x}) = g'(\xi) = g(1) - g(0) = f(\mathbf{y}) - f(\mathbf{x})$$

Thus,

$$|f(\mathbf{y} - \mathbf{x})| \leq [\text{Cauchy-Schwartz}] \leq \left\| \frac{\partial f}{\partial \mathbf{x}}(\gamma(\xi)) \right\| \cdot \|\mathbf{y} - \mathbf{x}\| \leq L \cdot \|\mathbf{y} - \mathbf{x}\|$$

where

$$L = \max_{\mathbf{x} \in D} \left\| \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}) \right\| < \infty$$

Thus, $f$ is Lipschitz with a Lipschitz constant $L$. □

# Chapter 3

# Algorithms

This chapter presents and analyses physics-inspired dynamical systems and their corresponding numerical simulations, which together comprise algorithms for heuristically solving QUBO problem. These algorithms are namely Coherent Ising Machine (CIM) 3.2, Simulated Bifurcation (SB) 3.4, Simulated Coherent Ising Machine (SimCIM) 3.8, and Ballistic Simulated Bifurcation (bSB) 3.9. There are other similar variants of these physics-inspired algorithms, such as Discrete Simulated Bifurcation (dSB) [19] and heated versions of simulated bifurcation [27], but those are not covered in the thesis.

These physics-inspired algorithms were proposed as adaptations of quantum algorithms (algorithms designed for running on a quantum computer) for solving QUBO and other combinatorial optimization problems. These adaptations are done by approximating the expected value of annihilation operator $a$, present in quantum Hamiltonian $H$, by a complex number $x + iy$ where $x, y \in \mathbb{R}$. What is left after this *dequantization* are the equations of motion for variables $x, y$. This is where physics-inspired dynamical systems for solving combinatorial optimization problems came from — the details can be found in [40], [23], and [22]. Modifying these dynamical systems further enabled to design algorithm which are easier to simulate, while preserving, and even improving the solution quality [19] [37].

Although motivated by quantum systems, these dynamical systems are completely classical, often even deterministic. Thus, those systems will be analyzed in what follows from a mathematical perspective of dynamical systems.

First, in order to understand how a dynamical system generates candidate solutions of QUBO problem, consider a state vector which evolves in time $\mathbf{x} : [0, \infty\rangle \to \mathbb{R}^n$. Taking the sign of each component at some time instance $t$, $\sigma_i \coloneqq x_i(t)$ provides a candidate solution for the QUBO task $(\sigma_1, \ldots, \sigma_n)$. The goal is to obtain a dynamical system which *attracts* various trajectories towards such areas of space that provide high-quality approximate solutions of the QUBO problem, according to the mapping mentioned above $\sigma_i \coloneqq x_i(t)$.

Finally, CIM and SB were chronologically developed earlier, and they are respectively a

gradient and a Hamiltonian system over the same energy/landscape function $E$. Motivated by CIM and SB, and by modifying the function $E$, new algorithms were proposed. This is how bSB and SimCIM were born. They are also respectively a gradient and a Hamiltonian systems over this new energy/landscape function.

## 3.1 Introduction to CIM and SB

Let us define the following energy functional

$$E(\mathbf{x}; \alpha, \beta, \mu) = \frac{\alpha}{4} \sum_{i=1}^{n} (x_i^2 - \mu)^2 - \frac{\beta}{2} \sum_{i,j=1}^{n} J_{ij} x_i x_j \tag{3.1}$$

which will be used both in sections 3.2 and 3.4. This energy functional is a function of position $\mathbf{x}$ and parameters $\alpha, \beta, \mu$ denoted after the semicolon. It is essential to analyze this function with respect to parameters because it will be governing the vector fields defined in the following dynamical systems. This energy function will vary in time itself because the parameters will vary in time.

The given coupling matrix $J$ is symmetric (if not, we may always symmetrize it by taking coefficients $\frac{1}{2}(J_{ij} + J_{ji})$ instead of $J_{ij}$ and $J_{ji}$). $\alpha(t) > 0$, $\beta(t) > 0$ and $\mu(t) \in \mathbb{R}$ are parameters which (might) vary over time and thus produce different landscape over which the point $\mathbf{x}$ moves. However, both in CIM and SB algorithm, parameters $\alpha$ and $\beta$ are constant in time. The crucial part is that $\mu(t)$ is time-dependent which causes the bifurcations of the system. When talking about CIM and SB, a bifurcation refers to the emergence of new or vanishing of some existing local minima of energy functional $E$. The second part, $-\frac{\beta(t)}{2} \sum_{i,j} J_{ij} x_i x_j$, corresponds to the value of the QUBO task and is the actual function that we want to minimize but over the set $\{-1, +1\}^n$. The first part, $\frac{\alpha(t)}{4} \sum_i (x_i^2 - \mu(t))^2$, thus might be viewed as a penalty term. It penalizes $x_i$'s whose magnitude is different than $\sqrt{\mu}$ (whenever $\mu \geq 0$). More about penalty terms can be found in [41].

When $\mu$ is small enough, the only local and global minimum of $E$ is at the origin. As $\mu$ increases, new local minima occur. For large enough $\mu$, $E$ will have $2^n$ local minima corresponding to (existing in the close vicinity of) the points $\{-\sqrt{\mu}, +\sqrt{\mu}\}^n$. The value of the energy functional $E$ at some of these points $\mathbf{x} = \sqrt{\mu}\boldsymbol{\sigma}, \boldsymbol{\sigma} \in \{-1, +1\}^n$ is given by

$$E(\sqrt{\mu}\boldsymbol{\sigma}; \alpha, \beta, \mu) = -\frac{\beta\mu}{2} \sum_{i,j} J_{ij} \sigma_i \sigma_j$$

which is proportional to the function that we are minimizing over the discrete set. Increasing the parameter $\mu$ from former to latter value, new local minima emerge – a process which we refer to as *bifurcations*. If we evolve the system long enough, i.e. until $\mu$ crosses certain

value, we will have $2^n$ local minima of $E$, each corresponding to one feasible solution from $\{-1,+1\}^n$. This is the statement of the next theorem.

**Theorem 3.1.1.** *Let* $D \coloneqq \max_i \sum_{j=1}^{n} \left|\beta J_{ij}\right|$ *For* $\mu > \frac{1}{2\alpha}(1+3\sqrt{3})D$*, there exist* $2^n$ *local minima of function E, which are all in a one-to-one correspondence with the feasible set* $\{-1,+1\}^n$*. This correspondence is given by taking the sign of each component — for a local minimum* $\mathbf{x}$*, the corresponding element of the feasible set is* $\boldsymbol{\sigma} = \operatorname{sign}\mathbf{x}$*.*

*Proof.* The proof can be found in [39]. □

Let us calculate the first and second differential of $E$.

$$\frac{\partial E}{\partial \mathbf{x}} = \alpha[x_1^3,\ldots,x_n^3] - \alpha\mu\mathbf{x}^T - \beta\mathbf{x}^T J \tag{3.2}$$

$$\frac{\partial^2 E}{\partial \mathbf{x}^2} = 3\alpha\,\operatorname{diag}(x_1^2,\ldots,x_n^2) - \alpha\mu I - \beta J \tag{3.3}$$

**Proposition 3.1.2.** *The origin is a stationary point of E, meaning that* $\frac{\partial E}{\partial \mathbf{x}}(\mathbf{0}) = \mathbf{0}^T$*, for all* $\mu \in \mathbb{R}$*. For*

$$\mu < -\frac{\beta}{\alpha}\lambda_{\max}(J) \tag{3.4}$$

*the origin is a local minimum of E. For*

$$\mu > -\frac{\beta}{\alpha}\lambda_{\max}(J) \tag{3.5}$$

*the origin is not a local minimum of E.*

*Proof.* Use formula (3.2) to see that the origin is stationary. The second differential of $E$ at the origin is

$$\frac{\partial^2 E}{\partial \mathbf{x}^2}(\mathbf{0}) = -\alpha\mu I - \beta J$$

If (3.4) holds, than $\lambda_{\min}(\frac{\partial^2 E}{\partial \mathbf{x}^2}(\mathbf{0})) = -\alpha\mu - \beta\lambda_{\max}(J) > \alpha\frac{\beta}{\alpha}\lambda_{\max}(J) - \beta\lambda_{\max}(J) = 0$ which means that all eigenvalues of $\frac{\partial^2 E}{\partial \mathbf{x}^2}$ are positive which implies that $E$ is a local minimum. Analogous argument shows that when (3.5) holds, the origin is not a local minimum because at least one eigenvalue is negative. □

For

$$\mu \leq -\frac{\beta}{\alpha}\lambda_{\max}(J) \tag{3.6}$$

we can actually analyze the second differential of $E$ to see that

$$\begin{aligned}
\lambda_{\min}\left(\frac{\partial^2 E}{\partial \mathbf{x}^2}\right) &\geq [\text{Weyl's inequality (2.3.1) }] \\
&\geq \lambda_{\min}(3\alpha \operatorname{diag}(x_1^2, \ldots, x_n^2)) + \lambda_{\min}(-\alpha\mu I - \beta J) \\
&= 3\alpha \min_{i=1,\ldots,n} x_i^2 - \alpha\mu - \beta\lambda_{\max}(J) \\
&\geq -\alpha\mu - \beta\lambda_{\max}(J) \\
&\geq \alpha\frac{\beta}{\alpha}\lambda_{\max}(J) - \beta\lambda_{\max}(J) = 0
\end{aligned} \tag{3.7}$$

So, when condition (3.6) is satisfied, we know from 2.3.2 and 3.1.2 that $E$ is a convex function with its global minimum centered at the origin.

The condition

$$\mu = -\frac{\beta}{\alpha}\lambda_{\max}(J) \tag{3.8}$$

is thus called the *first bifurcation point*.

It thus makes sense to increase $\mu$, for example linearly, from certain starting value $\mu_0$ to certain ending value $\mu_1$, while taking

$$\mu_0 \geq -\frac{\beta}{\alpha}\lambda_{\max}(J) \tag{3.9}$$

because this is the point when bifurcations in landscape function $E$ start to emerge. Selecting the concrete value of $\mu_0$ and $\mu_1$ should be based on empirical evidence i.e. by experimenting which parameter provides the best result.

**Example 3.1.3.** *In this example we will observe what happens for the case $n = 2$. Since the coupling matrix is symmetric and with zero diagonal elements, there are, up to scale, two possibilities for the coupling matrix, and those are given by $J = \begin{bmatrix} 0 & \epsilon \\ \epsilon & 0 \end{bmatrix}$, $\epsilon \in \{-1, +1\}$. Consider the QUBO task given by*

$$\min_{\sigma_1,\sigma_2\in\{-1,+1\}} -\sum_{i,j=1}^{2} J_{ij}\sigma_i\sigma_j \tag{3.10}$$

*Diagonalization of $J$ provides us with eigenvectors $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ corresponding respectively to eigenvalues $-\epsilon, +\epsilon$. First, we will determine stationary points of $E$ for some fixed $\mu$.*

$$\begin{aligned}
\alpha x_1(\mu - x_1^2) + \beta\epsilon x_2 &= 0 \\
\alpha x_2(\mu - x_2^2) + \beta\epsilon x_1 &= 0
\end{aligned} \tag{3.11}$$

*which is, by summing and subtracting these equations, equivalent to*

$$\begin{aligned} \alpha\mu(x_1 + x_2) - \alpha(x_1 + x_2)(x_1^2 - x_1x_2 + x_2^2) + \beta\epsilon(x_1 + x_2) = 0 \\ \alpha\mu(x_1 - x_2) - \alpha(x_1 - x_2)(x_1^2 + x_1x_2 + x_2^2) - \beta\epsilon(x_1 - x_2) = 0 \end{aligned} \tag{3.12}$$

***Case 1:*** $x_1 + x_2 = 0, x_1 - x_2 = 0$
*This is the trivial case* $x_1 = x_2 = 0$ *which is the stationary point for all* $\mu$.

***Case 2:*** $x_1 + x_2 = 0, x_1 - x_2 \neq 0$
*From first equation we have* $x_1 = -x_2$ *so plugging it into the above we get*

$$\begin{aligned} 2\alpha\mu x_1 - 2\alpha x_1^3 - 2\beta\epsilon x_1 = 0 \\ \Leftrightarrow x_1^2 = \mu - \frac{\beta\epsilon}{\alpha} \end{aligned} \tag{3.13}$$

*From here we conclude that*

$$-x_2 = x_1 = \pm\sqrt{\mu - \frac{\beta\epsilon}{\alpha}}, \quad \text{for } \mu \geq \frac{\beta\epsilon}{\alpha} \tag{3.14}$$

***Case 3:*** $x_1 + x_2 \neq 0, x_1 - x_2 = 0$
*From second equation we have* $x_1 = x_2$ *so plugging it into the above we get*

$$\begin{aligned} 2\alpha\mu x_1 - 2\alpha x_1^3 + 2\beta\epsilon x_1 = 0 \\ \Leftrightarrow x_1^2 = \mu + \frac{\beta\epsilon}{\alpha} \end{aligned} \tag{3.15}$$

*From here we conclude that*

$$x_2 = x_1 = \pm\sqrt{\mu + \frac{\beta\epsilon}{\alpha}}, \quad \text{for } \mu \geq -\frac{\beta\epsilon}{\alpha} \tag{3.16}$$

***Case 4:*** $x_1 + x_2 \neq 0, x_1 - x_2 \neq 0$
*This provides*

$$\begin{aligned} \alpha\mu - \alpha(x_1^2 - x_1x_2 + x_2^2) + \beta\epsilon = 0 \\ \alpha\mu - \alpha(x_1^2 + x_1x_2 + x_2^2) - \beta\epsilon = 0 \end{aligned} \tag{3.17}$$

$$\begin{aligned} & & x_1^2 - x_1x_2 + x_2^2 &= \mu + \frac{\beta\epsilon}{\alpha} \\ \Leftrightarrow & & x_1^2 + x_1x_2 + x_2^2 &= \mu - \frac{\beta\epsilon}{\alpha} \end{aligned} \tag{3.18}$$

$\Leftrightarrow$ [ *by introducing* $y_1 = x_1 + x_2, y_2 = x_1 - x_2$]

$$\begin{aligned} & & y_1^2 + 3y_2^2 &= 4(\mu + \frac{\beta\epsilon}{\alpha}) \\ \Leftrightarrow & & 3y_1^2 + y_2^2 &= 4(\mu - \frac{\beta\epsilon}{\alpha}) \end{aligned} \tag{3.19}$$

*which is a full-rank linear system with quadratic indeterminates whose solution is then*

$$\begin{aligned} y_1^2 &= \frac{1}{2}[-(\mu + \frac{\beta\epsilon}{\alpha}) + 3(\mu - \frac{\beta\epsilon}{\alpha})] = \mu - 2\frac{\beta\epsilon}{\alpha} \\ y_2^2 &= \frac{1}{2}[3(\mu + \frac{\beta\epsilon}{\alpha}) - (\mu - \frac{\beta\epsilon}{\alpha})] = \mu + 2\frac{\beta\epsilon}{\alpha} \end{aligned} \tag{3.20}$$

*leading to 4 solutions*

$$\begin{aligned} x_1 &= \frac{1}{2}[(-1)^k \sqrt{\mu - \frac{2\beta\epsilon}{\alpha}} + (-1)^l \sqrt{\mu + \frac{2\beta\epsilon}{\alpha}}] \\ x_2 &= \frac{1}{2}[(-1)^k \sqrt{\mu - \frac{2\beta\epsilon}{\alpha}} - (-1)^l \sqrt{\mu + \frac{2\beta\epsilon}{\alpha}}] \end{aligned}, \quad k, l \in \{0, 1\} \tag{3.21}$$

*which holds whenever* $\mu \geq |\epsilon| \frac{2\beta}{\alpha}$.

*First two non-trivial cases are local minima, which can be seen by taking the second partial derivatives of* $E$*:*

$$-D^2E = \begin{bmatrix} \alpha\mu - 3\alpha x_1^2 & \beta\epsilon \\ \beta\epsilon & \alpha\mu - 3\alpha x_2^2 \end{bmatrix} \tag{3.22}$$

*If* $-D^2E$ *is negative-definite (all of its eigenvalues are less than zero), then we have a local minimum. Calculating the eigenvalues is straightforward for this* $2 \times 2$ *symmetric matrix, and plugging in from first two cases the fact that* $x_1^2 = x_2^2$ *we obtain that* $E$ *has local minimum when*

$$\beta\epsilon < \alpha(3x_1^2 - \mu) \tag{3.23}$$

*For the case 3.1.3,* (3.23) *becomes equivalent to*

$$\mu > \frac{2\beta\epsilon}{\alpha} \tag{3.24}$$

*while for the case 3.1.3,* (3.23) *becomes equivalent to*

$$\mu > \frac{-\beta\epsilon}{\alpha} \tag{3.25}$$

*Now we have all pieces to form a conclusion of this example. Depending on the sign of* $\epsilon$*, the solution of the original QUBO problem* (3.10) *is*

$$\begin{aligned} \sigma_1 &= -\sigma_2, \quad \textit{for } \epsilon < 0 \\ \sigma_1 &= \sigma_2, \quad \textit{for } \epsilon > 0 \end{aligned} \tag{3.26}$$

*Note that in each case we can choose the sign of one* $\sigma$ *arbitrarily.*

*In case that* $\epsilon > 0$*, we have that* (3.1.3) *are the stationary points which bifurcate earlier i.e. for smaller* $\mu$ *and those are also the local minima. This is the case where* $x_1 = x_2$ *so following the minima which bifurcate earlier provides an exact global minimum for the QUBO problem.*

*In case that* $\epsilon < 0$*, we have that* (3.1.3) *are stationary points which bifurcate earlier i.e. for smaller* $\mu$ *and those are local minima as well because* $\frac{2\beta\epsilon}{\alpha} < \frac{\beta\epsilon}{\alpha}$*. This is the case where* $x_1 = -x_2$ *so following the minima which bifurcate earlier provides an exact global minimum for the QUBO problem. In figure 3.1 we can see how these local minima start to emerge as* $\mu$ *increases.*

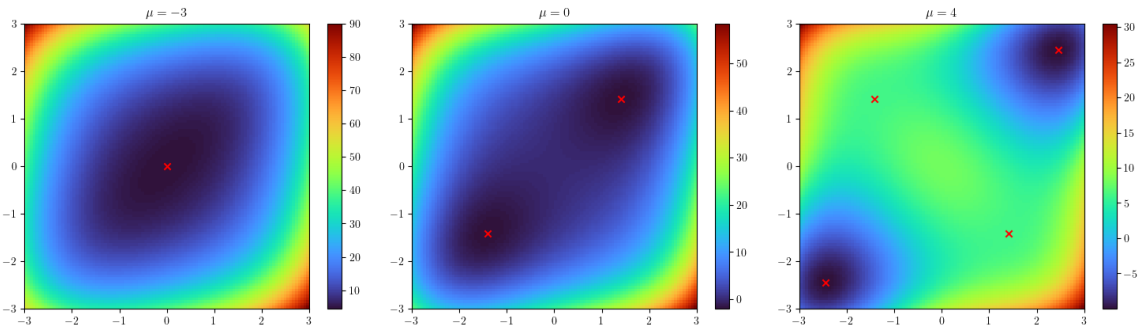


Figure 3.1: Visualization of energy landscape $E$ for CIM and SB

**Overview:** This is a visual representation of energy landscape $E$ for a system with two variables and coupling term $J_{12} = J_{21} = 1$. Optimum solutions of the corresponding QUBO problem are (+,+) and (-, -). As the parameter $\mu$ increases, it causes the qualitative appearance of landscape function $E$ to change — new local minima emerge, and they correspond to candidate solutions of QUBO.

**Details:** The parameters are $\alpha = 1, \beta = 2$. The values of parameter $\mu$ are denoted on the top of each subplot. Red crosses denote local minima of $E$.

## 3.2 Coherent Ising machine (CIM)

This section is based on [32] and [39].

### Dynamics

CIM is a gradient dynamical system with time-dependent landscape function 3.1. The vector field describing this dynamical system is thus given by the negative gradient of the energy function 3.1

$$\mathbf{f}(\mathbf{x};\alpha,\beta,\mu) = -\frac{\partial E}{\partial \mathbf{x}}^T(\mathbf{x};\alpha,\beta,\mu) \tag{3.27}$$

Parameters are written after the semicolon and those are $\alpha,\beta,\mu$. Time dependence is achieved by varying parameters in time. In this algorithm only parameter $\mu$ is monotonically increased from certain initial value to certain final value. Parameters $\alpha,\beta > 0$ are kept constant.

Written out component-wise, the dynamical system describing CIM is given by a set of coupled differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t);\alpha,\beta,\mu(t)) \tag{3.28}$$

or written out further, without denoting the dependence on time,

$$\dot{x}_i = f_i = \alpha x_i(\mu - x_i^2) + \beta \sum_j J_{ij}x_j \quad ; \quad i = 1,\dots,n \tag{3.29}$$

### Algorithm

CIM algorithm is a simulation of the described dynamical system on a computer using Euler method.

**Algorithm 1** CIM

1: Initialize vector $\mathbf{x}$ randomly around $\mathbf{0}$
2: **for** $k$ in range$(0, N_{\text{iter}})$ **do**
3: $\quad x_i \leftarrow \left[\alpha x_i(\mu(t_k) - x_i^2) + \beta \sum_j J_{ij}x_j\right] \cdot \Delta_t, \quad i = 1,\dots,n$
4: **end for**

On figure 3.2 we observe that as local minima bifurcate away from the origin, the system follows some of these local minima. Local minima corresponding to $(+,+)$ and

$(-,-)$ are lower than those corresponding to $(+,-)$ and $(-,+)$ so it is more likely that the system will converge towards $(+,+)$ or $(-,-)$ that the other two minima. Also, local minima corresponding to $(+,+)$, $(-,-)$ bifurcate earlier, i.e. for smaller values of $\mu$, than the other two local minima. That's why in the middle subplot the system will surely converge to the optimal solution, since other local minima have not yet even emerged.

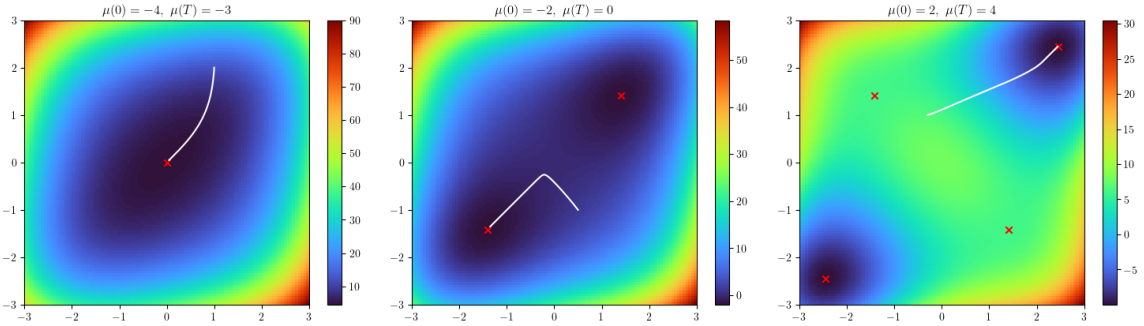


Figure 3.2: CIM algorithm

**Overview:** This is a visual representation of CIM algorithm on a system with two variables with coupling term $J_{12} = J_{21} = 1$. Optimum solutions of the corresponding QUBO problem are (+,+) and (-, -). In the left subplot, the parameter $\mu$ did not cross the first bifurcation point, so the dynamical system collapses towards the origin (starting far away from it). As the time passes, $\mu$ increases, causing a change in the qualitative appearance of landscape $E$. In the middle subplot, the origin splits into two local minima (red crosses) which correspond to the solution of QUBO problem. In the right subplot, two new local minima emerge, corresponding to suboptimal solutions of QUBO. However, they are not as *attractive* for the dynamical system as the optimal solutions. In both cases, the dynamical system ends up converging towards a local minimum corresponding to an optimal solution of QUBO.

**Details:** The white line represents the trajectory obtained by running the CIM algorithm. Red crosses denote local minima of $E$. Each subplot corresponds to a new run of the algorithm. The parameters are $\alpha = 1, \beta = 2$. The energy landscape $E$ is plotted only for the final value of $\mu$, that is $E(\cdot; \alpha, \beta, \mu(T))$ is plotted on each of the subplots. In each subplot, $\mu$ increases linearly from starting to ending value. Starting and ending values are respectively denoted as $\mu(0)$ and $\mu(T)$ on top of each subplot. Number of iterations is 400, while $\Delta_t = 0.01$. Initial positions are $\mathbf{x}(0) = (1, 2)$ for the left subplot, $\mathbf{x}(0) = (0.5, -1.0)$ for the middle subplot, and $\mathbf{x}(0) = (-0.3, 1.0)$ for the right subplot.

## 3.3 Gradient descent and momentum

Gradient descent algorithms are used extensively for various continuous optimization tasks, such as for training neural networks in machine learning models. These algorithms could be understood as Euler method applied to the gradient system where the gradient is taken of the function which is being optimized. Suppose that we want to find a (local) minimum of a function $E : \mathbb{R}^n \to \mathbb{R}$. A gradient descent algorithm is then

$$\begin{aligned} \mathbf{x}^{(0)} &= \mathbf{x}_0 \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} - \frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}^{(k)}) \cdot \Delta_t, \quad k = 0, \ldots, N_{\text{iter}} - 1 \end{aligned} \tag{3.30}$$

where $\Delta_t$ is a parameter called *learning rate*, $N_{\text{iter}}$ is the number of iterations, and $\mathbf{x}_0$ is some initial position which is often randomly chosen, ideally in a close vicinity of the target (local) minimum.

On top of this, there is a technique which is often used for improving the convergence rate of such optimization algorithms, and it is called the *momentum* [35]. The momentum acts in such a way that it accumulates the previous displacements and takes those into account when performing the next update. The algorithm is the following

$$\begin{aligned} \mathbf{x}^{(0)} &= \mathbf{x}_0 \\ \mathbf{x}^{(1)} &= \mathbf{x}^{(0)} - \frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}^{(0)}) \cdot \Delta_t \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} - \frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}^{(k)}) \cdot \Delta_t + \gamma \left( \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)} \right), \quad k = 1, \ldots, N_{\text{iter}} - 1 \end{aligned} \tag{3.31}$$

where all parameters are as earlier, and $\gamma \in [0, 1]$ is the momentum parameter. Adding the momentum has been shown to increase the convergence rate significantly. According to [35], adding momentum is actually equivalent to the numerical simulation of a physical situation in which a Newtonian particle moves through a viscous medium under the influence of a conservative force field. This interpretation is of special interest to us because we are already dealing with continuous dynamical systems.

Now let's return to the CIM algorithm.

### Adding momentum to CIM

Since CIM is essentially a gradient descent algorithm, it might make sense to apply the momentum to it for improving its performance.

Adding momentum to the algorithm above, we get the following: First, choose the momentum parameter $0 \leq \gamma \leq 1$. Momenta will be stored in vector $\mathbf{y}$. We could also initialize the momentum vector randomly.

**Algorithm 2** CIM with momentum

1: Initialize vectors $\mathbf{x}, \mathbf{y}$ randomly around $\mathbf{0}$
2: **for** $k$ in range$(0, N_{\text{iter}})$ **do**
3: $\quad y_i \leftarrow \gamma \cdot y_i + \alpha x_i(\mu(t_k) - x_i^2) + \beta \sum_j J_{ij} x_j, \quad i = 1, \ldots, n$
4: $\quad x_i \leftarrow x_i + y_i \cdot \Delta_t, \quad i = 1, \ldots, n$
5: **end for**

For momentum parameter $\gamma = 0$, CIM with momentum 2 becomes precisely CIM 1.

**Remark 3.3.1.** *'CIM' and 'CIM with momentum' will in the rest of the thesis often be used interchangeably, referring to CIM with momentum, unless specified otherwise.*

## 3.4 Simulated Bifurcation (SB)

This section is based on [23].

### Dynamics

Simulated Bifurcation (SB) is a Hamiltonian dynamical system with time-dependent Hamiltonian given by

$$H(\mathbf{x}, \mathbf{y}; \alpha, \beta, m, \mu) = \frac{m}{2} \sum_{i=1}^{n} y_i^2 + E(\mathbf{x}; \alpha, \beta, \mu) \tag{3.32}$$

where $E$ is the potential function given by 3.1. The first part of the summation corresponds to kinetic energy depending only on momenta $\mathbf{y}$ while the second part corresponds to potential energy depending only on positions $\mathbf{x}$. Parameters $\alpha, \beta, m > 0$ will be kept constant over time, while the parameter $\mu(t) \in \mathbb{R}$ will monotonically increase from certain initial to certain final value. The system of differential equations governing the dynamics for this Hamiltonian system is thus given by

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \frac{\partial H}{\partial \mathbf{y}}(\mathbf{x}(t), \mathbf{y}(t); \alpha, \beta, m, \mu(t)) \\ \dot{\mathbf{y}}(t) &= -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}(t), \mathbf{y}(t); \alpha, \beta, m, \mu(t)) \end{aligned} \tag{3.33}$$

Written out further component-wise and omitting the time variable we get

$$\begin{aligned} \dot{x}_i &= \frac{\partial H}{\partial y_i} = m y_i \\ \dot{y}_i &= -\frac{\partial H}{\partial x_i} = \alpha x_i(\mu - x_i^2) + \beta \sum_j J_{ij} x_j \end{aligned} \tag{3.34}$$

## Algorithm

SB algorithm is a numerical simulation of the above dynamical system on a computer. The simulation is performed by symplectic Euler method rather than standard Euler method because it is more stable for simulating Hamiltonian systems. The update step with standard symplectic Euler method would thus be

$$\begin{aligned} x_i^{(k+1)} &= x_i^{(k)} + m y_i^{(k)} \cdot \Delta_t \\ y_i^{(k+1)} &= y_i^{(k)} + [\alpha x_i^{(k+1)} \cdot (\mu(t_k) - (x_i^{(k+1)})^2) + \beta \sum_j J_{ij} x_j^{(k+1)}] \cdot \Delta_t \end{aligned} \tag{3.35}$$

where $t_k = \Delta_t k$ is the time discretization, having $\Delta_t$ fixed.

Additionally, as the authors proposed in the article [23], a modified symplectic Euler method provides even better results either in terms of solution quality or computation speed. Since the computation of $\sum_j J_{ij} x_j^{(k+1)}$ is computationally the most expensive part, the Hamiltonian is split into two parts

$$\begin{aligned} H(\mathbf{x}, \mathbf{y}; \alpha, \beta, m, \mu) &= M \frac{H_1(\mathbf{x}, \mathbf{y}; \alpha, \beta, m, \mu)}{M} + H_2(\mathbf{x}, \mathbf{y}; \alpha, \beta, m, \mu) \\ H_1(\mathbf{x}, \mathbf{y}; \alpha, \beta, m, \mu) &:= \frac{m}{2} \sum_i y_i^2 + \frac{\alpha}{4} \sum_i (x_i^2 - \mu)^2 \\ H_2(\mathbf{x}, \mathbf{y}; \alpha, \beta, m, \mu) &:= -\frac{\beta}{2} \sum_{i,j} J_{ij} x_i x_j \end{aligned} \tag{3.36}$$

$M$ is some positive integer.

Applying arguments about symplectic maps, a modified explicit symplectic Euler method for Simulated Bifurcation is obtained, which is called the SB algorithm.

**Algorithm 3** SB

Initialize vectors $\mathbf{x}, \mathbf{y}$ randomly around 0
**for** $k$ in range$(0, N_{\text{iter}})$ **do**
  **for** $l$ in range$(0, M)$ **do**
    $x_i \leftarrow x_i + m y_i \delta_t, \quad i = 1, \ldots, n$
    $y_i \leftarrow y_i + \left[\alpha x_i \cdot (\mu(t_k) - x_i^2)\right] \cdot \delta_t, \quad i = 1, \ldots, n$
  **end for**
  $x_i \leftarrow x_i, \quad i = 1, \ldots, n$
  $y_i \leftarrow y_i + \beta \sum_{j=1}^n J_{ij} x_j \cdot \Delta_t, \quad i = 1, \ldots, n$
**end for**

$\Delta_t$ is fixed discretized time interval and $\delta_t$ is its smaller refinement given by $\delta_t = \frac{\Delta_t}{M}$. Updates with respect to the first part of the Hamiltonian are refined by iterating $m = 0, \ldots, M-1$ sub-updates with smaller time step $\delta_t$.

In figure 3.3 we see visualization of SB algorithm on system with two variables.

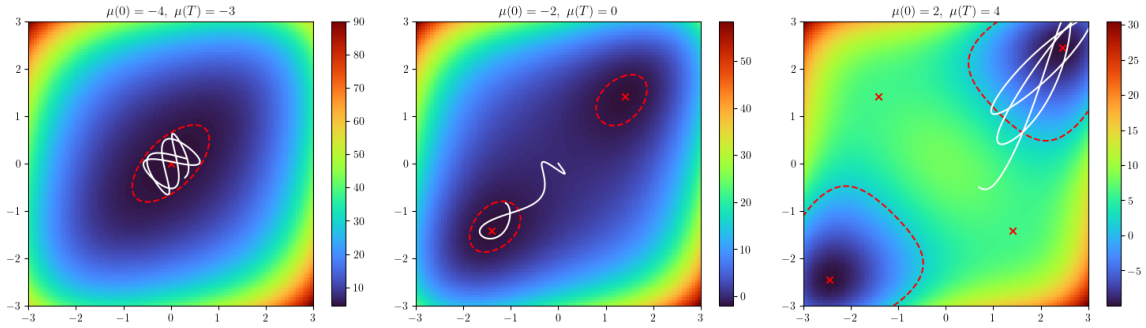


Figure 3.3: SB algorithm

**Overview:** This is a visual representation of SB algorithm on a system with two variables with coupling term $J_{12} = J_{21} = 1$. Optimum solutions of the corresponding QUBO problem are (+,+) and (-, -). In the left subplot, the parameter $\mu$ did not cross the first bifurcation point, so the dynamical system revolves around the origin. As the time passes, $\mu$ increases, causing a change in the qualitative appearance of landscape $E$. In the middle subplot, the origin splits into two local minima (red crossses) which correspond to the solution of QUBO problem. In the right subplot, two new local minima emerge, corresponding to suboptimal solutions of QUBO. However, they are not as *attractive* for the dynamical system as the optimal solutions. In both cases, the dynamical system ends up circulating through the area (red dashed line) corresponding to an optimal solution of QUBO.

**Details:** The white line represents the trajectory obtained by running the SB algorithm. Red crosses denote local minima of $E$. The red dotted line represents the boundary for the feasible region of the Hamiltonian system, that is the level set of $E$ at value equals to the total energy of the system at that time instance. The parameters are $\alpha = 1, \beta = 2$. Each subplot corresponds to a new run of the algorithm. The energy landscape $E$ is plotted only for the final value of $\mu$, that is $E(\cdot; \alpha, \beta, \mu(T))$ is plotted on each of the subplots. In each subplot, $\mu$ increases linearly from starting to ending value. Starting and ending values are respectively denoted as $\mu(0)$ and $\mu(T)$ on top of each subplot. Initial positions are $\mathbf{x} = (0, 0)$ in each subplot. For the leftmost subplot, initial positions are $\mathbf{x} = [0.3, -0.3]$, initial momenta are $\mathbf{y} = [0.5, 0.5]$, $\Delta_t = 0.1$, and the number of iterations is 100. For the middle subplot, initial positions are $\mathbf{x} = [0, 0]$, initial momenta are $\mathbf{y} = [0.3, -0.4]$, $\Delta_t = 0.05$, and the number of iterations is 150. For the rightmost subplot, initial positions are $\mathbf{x} = [0, 0]$, initial momenta are $\mathbf{y} = [0.3, -0.2]$, $\Delta_t = 0.05$, and the number of iterations is 150.

## 3.5 Mechanism of CIM and SB

Let us explain the mechanism of CIM and SB dynamical systems and corresponding algorithms. Although CIM and SB are nonautonomous dynamical systems, by the connection theorem 2.1.9, in order to analyze such system locally (in time), it is sufficient to consider it as an autonomous system with frozen vector field at some time instance of interest.

The behaviour of these two systems around the first bifurcation point could be understood in the light of examples 2.1.11 and 2.1.10 which provide solutions of linear gradient and Hamiltonian systems. Indeed, by the Hartman-Grobman theorem 2.1.7, we have that the qualitative behaviour of CIM and SB around the origin is equivalent to the dynamics of the corresponding linearized systems. Thus, we need to analyze $\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = -3\alpha \operatorname{diag}(x_1^2, \dots, x_n^2) + \alpha\mu I + \beta J$ which at the origin is $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{0}) = \alpha\mu I + \beta J$. To understand this we need to observe what happens in the reference frame which diagonalizes $J$.

Before the first bifurcation point, all eigenvalues of $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{0})$ are smaller than 0, so CIM will collapse towards the origin, while SB will revolve around it.

As the time passes, and $\mu$ increases, some of the largest eigenvalues of $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{0})$ will cross 0 and become positive. Both the CIM and SB will start to rapidly expand towards these directions, while staying bounded or even contract in all other directions. This is why both CIM and SB provide good approximate solutions of the QUBO problem at early stages of the algorithm.

After the system expands far enough from the origin, and as $\mu$ continues to increase, the behaviour around origin stops being relevant. Instead, new local minima of $E$ emerge and the system starts either converging towards them (CIM) or revolving around them (SB). Under the assumption that local minima corresponding to better QUBO solutions emerge earlier (for smaller $\mu$) than other local minima, or that these local minima will have higher *attractivity*, we expect that the system will be able to converge towards these local minima and thus provide high-quality QUBO solutions.

As discussed in [39] for CIM, those variables which bifurcate from the origin at early stages *usually* do not return to the origin nor change their sign anymore. This potentially allows us to consider only the system reduced to those variables which did not bifurcate yet, while keeping others freezed.

Analyzing the bifurcation pattern (the relative order of emergence of local minima) would be essential for understanding the behavior of CIM. This would enable us to predict the performance of CIM and SB algorithms — provide us a way to tell when will these algorithms provide good solutions of QUBO problem and when it will fail to do so. However, it is not clear how to systematically treat and analyze this bifurcation pattern or develop theory about it. Analyzing this further is out of the scope of this thesis.

## 3.6 Comparing CIM and SB

### Bifurcation pattern

In the figure 3.4 we compare the bifurcation behaviour of a single-variable system and the evolution of the dynamics through time.

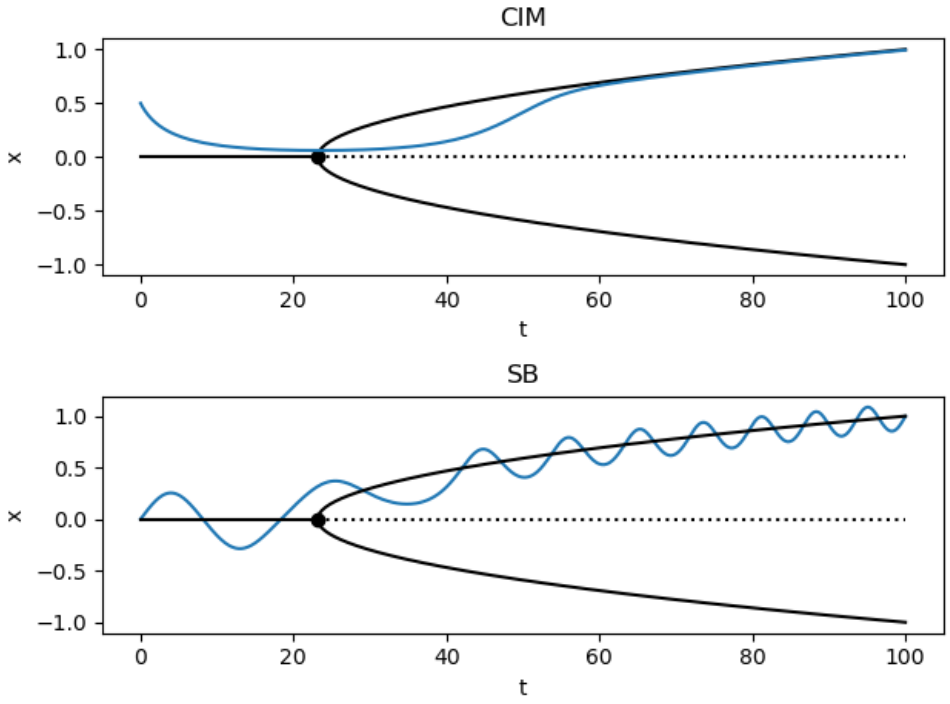


Figure 3.4: Comparison of CIM and SB with one variable

**Overview:** This is a visual representation of CIM and SB algorithms performing on only one variable. As the time evolves, and $\mu$ crosses the first bifurcation point (black dot), the landscape function $E$ changes its qualitative appearance and thus forces the dynamical system to bifurcate towards one of the two stable branches (solid black lines).
**Details:** Solid black lines represent the positions of local minima of landscape function $E$. Dashed line represents the position of an unstable stationary point. Black dot is the bifurcation point. Blue lines represent the evolution of CIM and SB algorithms. These are locally a good approximation of the exact solutions of CIM and SB dynamical systems. The following parameters have been used. $J = [0]$, $\alpha = 0.5, \beta = 0.5, \mu_0 = -0.3, \mu_1 = 1.0$, $\Delta_t = 0.1$. Number of iterations is 1000. Initial conditions for CIM are $x_1(0) = 0.5$, while for SB are $x_1(0) = 0.0, y_1(0) = 0.1$. Momentum is not used for this CIM simulation.

## Relation between CIM and SB

Although the momentum parameter is usually kept smaller than 1, it is very interesting what happens when $\gamma = 1$. In this case, CIM with momentum 2 potentially becomes a simulation of SB dynamical system. That is, we can choose hyperparameters in CIM with momentum in such a way that the update step becomes a symplectic Euler method applied to SB dynamics.

Let us distinguish between CIM's and SB's parameters with a superscript CIM and SB.

For given parameters for SB algorithm $\alpha^{\text{SB}}, \mu_0^{\text{SB}}, \mu_1^{\text{SB}}, \beta^{\text{SB}}, \Delta_t^{\text{SB}}, m^{\text{SB}}$, and assuming that $M^{\text{SB}} = 1$, the SB algorithm could be written like this (notice that the order of equations for $x_i$ and $y_i$ is flipped but this remains a correct numerical simulation of the dynamics described in the section above)

**Algorithm 4** variant of SB with $M = 1$

Initialize vectors $\mathbf{x}, \mathbf{y}$ randomly around 0
**for** $k$ in range$(0, N_{\text{iter}})$ **do**
    $y_i \leftarrow y_i + \left[\alpha^{\text{SB}} x_i \cdot (\mu^{\text{SB}}(t_k) - x_i^2) + \beta^{\text{SB}} \sum_{j=1}^n J_{ij} x_j\right] \cdot \Delta_t^{\text{SB}}, \quad i = 1, \dots, n$
    $x_i \leftarrow x_i + m^{\text{SB}} y_i \Delta_t^{\text{SB}}, \quad i = 1, \dots, n$
**end for**

Define the parameters for CIM as follows

$$\begin{aligned} \gamma^{\text{CIM}} &= 1 \\ \alpha^{\text{CIM}} &= \alpha^{\text{SB}} \Delta_t^{\text{SB}} \\ \beta^{\text{CIM}} &= \beta^{\text{SB}} \Delta_t^{\text{SB}} \\ \mu^{\text{CIM}} &= \mu^{\text{SB}} \\ \Delta_t^{\text{CIM}} &= m^{\text{SB}} \Delta_t^{SB} \end{aligned} \tag{3.37}$$

Plugging in the above parameters into CIM algorithm with momentum 2 we get

**Algorithm 5** CIM with momentum mimicking SB

Initialize vectors $\mathbf{x}, \mathbf{y}$ randomly around $\mathbf{0}$
**for** $k$ in range$(0, N_{\text{iter}})$ **do**
    $y_i \leftarrow y_i + \alpha^{\text{SB}} \Delta_t^{\text{SB}} x_i (\mu^{\text{SB}}(t_k) - x_i^2) + \beta^{\text{SB}} \Delta_t^{\text{SB}} \sum_{j=1}^n J_{ij} x_j, \quad i = 1, \dots, n$
    $x_i \leftarrow x_i + y_i \cdot m^{\text{SB}} \Delta_t^{\text{SB}}, \quad i = 1, \dots, n$
**end for**

which is precisely the above variant of SB algorithm 3.6. Although these implementations of CIM and SB are not completely equivalent, CIM with momentum has the ability to

simulate the dynamics of SB if the parameters are selected appropriately (especially $\gamma = 1$). In this context, CIM with momentum is in some sense a generalization of SB algorithm.

## 3.7 Introduction to SimCIM and bSB

Let us define the following energy functional

$$E(\mathbf{x};\alpha,\beta) = \frac{\alpha}{2}\sum_i x_i^2 - \frac{\beta}{2}\sum_{i,j} J_{ij}x_i x_j, \qquad \text{for } \mathbf{x}\in[-1,+1]^n \tag{3.38}$$

$$E(\mathbf{x};\alpha,\beta) = +\infty, \qquad \text{otherwise} \tag{3.39}$$

which will be used both in sections 3.8 and 3.9.

Let us define the corresponding vector field $\mathbf{f}$ as a negative gradient of the *regular* part of $E$ (3.38)

$$\mathbf{f}(\mathbf{x};\alpha,\beta) \coloneqq -\frac{\partial E}{\partial \mathbf{x}}^T(\mathbf{x};\alpha,\beta) = \alpha\mathbf{x} - \beta J\mathbf{x} \tag{3.40}$$

This energy functional is a function of position $\mathbf{x}$ and parameters $\alpha,\beta$ denoted after the semicolon.

**Remark 3.7.1.** *Sometimes when we choose and fix some concrete values for $\alpha$ and $\beta$, we will pretend that $E$ is only a function of position $\mathbf{x}$ and thus write $E(\mathbf{x}) \equiv E(\mathbf{x};\alpha,\beta)$.*

For SimCIM and bSB we will only be interested in the dynamics inside the region $[-1,+1]^n$ and we actually want to prevent the system from going outside of this region. That's why we formally define that $E(\mathbf{x}) = +\infty$ outside of $[-1,+1]^n$. This potentially complicates the mathematical formalism but it will not be an issue for the analysis presented here.

It is essential to analyze energy functional $E$ with respect to parameters because it will be governing the vector fields defined in the following dynamical systems. This energy function will vary in time itself because the parameters will vary in time.

Concretely, $\beta > 0$ will be constant over time, while $\alpha \equiv \alpha(t)$ will decrease monotonically from some initial value $\alpha_0$ to some final value $\alpha_1$. This decreasing regime varies among algorithms. In bSB, $\alpha(t)$ changes linearly, i.e. $\alpha(t) = \alpha_0 + \frac{\alpha_1-\alpha_0}{T}t$. In the original SimCIM algorithm, $\alpha(t)$ is a sigmoid function given by the hyperbolic tangent law $\alpha(t) = -(\alpha_1-\alpha_0)\tanh(A(t-\frac{T}{2})) + \alpha_1$, where $A > 0$ is some positive constant, for example $A = 3$. Although different regimes for $\alpha(t)$ produce different solutions and some regimes might be better than the others, we cannot predict which regime would be the best. The choice of the regime is not carved into stone and it even seems arbitrary to certain degree. Nevertheless, what is important is that $\alpha(t)$ continuously and monotonically (slowly) decreases in time starting from $\alpha_0$ and ending with $\alpha_1$. Thus, we will usually make $\alpha$ linearly decrease from $\alpha_0$ to $\alpha_1$.

Let us explain the behaviour of energy functional $E$ for arbitrary $\alpha \in \mathbb{R}$. Since $J$ is symmetric, it is thus orthogonally diagonalizable. Let $U^T JU = \Lambda = \mathrm{diag}(\lambda_1,\dots,\lambda_n)$ be

the diagonalization of $J$ such that eigenvalues of $J$ are sorted in descending way $\lambda_{\max} = \lambda_1 \geq \cdots \geq \lambda_n = \lambda_{\min}$. Denote change of variables $\bar{\mathbf{x}} \coloneqq U^T\mathbf{x}$. We have that

$$E(\mathbf{x}) = \frac{\alpha}{2}\mathbf{x}^T\mathbf{x} - \frac{\beta}{2}\mathbf{x}^T J\mathbf{x} = \frac{\alpha}{2}\bar{\mathbf{x}}^T\bar{\mathbf{x}} - \frac{\beta}{2}\bar{\mathbf{x}}^T\Lambda\bar{\mathbf{x}} = \bar{\mathbf{x}}^T(\frac{\alpha}{2}I - \frac{\beta}{2}\Lambda)\bar{\mathbf{x}} \tag{3.41}$$

Here the nice thing is that $(\frac{\alpha}{2}I - \frac{\beta}{2}\Lambda)$ is a diagonal matrix so $E$ represented in these transformed coordinates is more comprehensible.

Let us calculate the first and the second differential of $E$.

$$\frac{\partial E}{\partial \mathbf{x}}(\mathbf{x};\alpha,\beta) = \alpha\mathbf{x}^T - \beta\mathbf{x}^T J \tag{3.42}$$

$$\frac{\partial^2 E}{\partial \mathbf{x}^2} = \alpha I - \beta J \tag{3.43}$$

**Lemma 3.7.2.** *For $\alpha > \beta\lambda_{\max}$, origin $\mathbf{x}_0 = \mathbf{0}$ is the global minimum of $E$*

*Proof.* On one hand, $E(\mathbf{0}) = 0$. Write $E$ in the form $E(\mathbf{x}) = \frac{\alpha}{2}\mathbf{x}^T\mathbf{x} - \frac{\beta}{2}\mathbf{x}^T J\mathbf{x}$ so because $\mathbf{x}^T J\mathbf{x} \leq \lambda_{\max}\mathbf{x}^T\mathbf{x}$, we have

$$E(\mathbf{x}) \geq \frac{\alpha}{2}\mathbf{x}^T\mathbf{x} - \frac{\beta}{2}\lambda_{\max}\mathbf{x}^T\mathbf{x} = \frac{1}{2}(\alpha - \beta\lambda_{\max})\mathbf{x}^T\mathbf{x} \tag{3.44}$$

So, for $\mathbf{x} \neq \mathbf{0}$, this implies $E(\mathbf{x}) > 0$. □

**Lemma 3.7.3.** *For each $\alpha \in \mathbb{R}$ such that $\alpha \neq \beta\lambda_i, \forall i$; origin is the only stationary point of $E$. Consequently, for such $\alpha$ there are no local minima of $E$ except maybe $\mathbf{0}$.*

*Proof.* To see that there are no stationary points except the origin, we need to look at the first differential of $E$. $\frac{\partial E}{\partial \mathbf{x}} = 0$ is equivalent to

$$\frac{\alpha}{\beta}\mathbf{x} = J\mathbf{x} \tag{3.45}$$

implying that, other than the origin, for such $\mathbf{x}$ to exist, it needs to be an eigenvector of $J$ corresponding to eigenvalue $\frac{\alpha}{\beta}$ which by assumption is not. □

We could state few more facts about stationary points of $E$ for these special $\alpha$'s but it is not relevant for the upcoming sections. One more important thing is to analyze the second differential of $E$ for different values of $\alpha$. Since $D^2E = \alpha I - \beta J$, the eigenvalues of $D^2E$ are $\alpha - \beta\lambda_1, \ldots, \alpha - \beta\lambda_n$ with corresponding eigenvectors being same as $J$'s. The sign of these eigenvalues will locally determine the qualitative behavior of the dynamical systems which follow in the next sections.

One more question which should be addressed is starting and ending values $\alpha_0$ and $\alpha_1$. There is some degree of freedom in choosing them but there are some facts which should

be considered. Nothing interesting happens when $\alpha > \beta\lambda_{\max}$ since $E$ is a quadratic function with all eigenvalues positive. For $\alpha \leq \beta\lambda_{\max}$, $E$ starts to change because some eigenvalues for it start being negative.

The condition

$$\alpha = \beta\lambda_{\max}(J) \tag{3.46}$$

is thus called the *first bifurcation point*.

Thus, both for gradient system and for Hamiltonian system with landscape $E$, it makes sense starting the dynamics with

$$\alpha_0 \leq \beta\lambda_{\max}(J) \tag{3.47}$$

Selecting the concrete value of $\alpha_0$ and $\alpha_1$ should be based on empirical evidence i.e. by experimenting which parameter provides the best result.

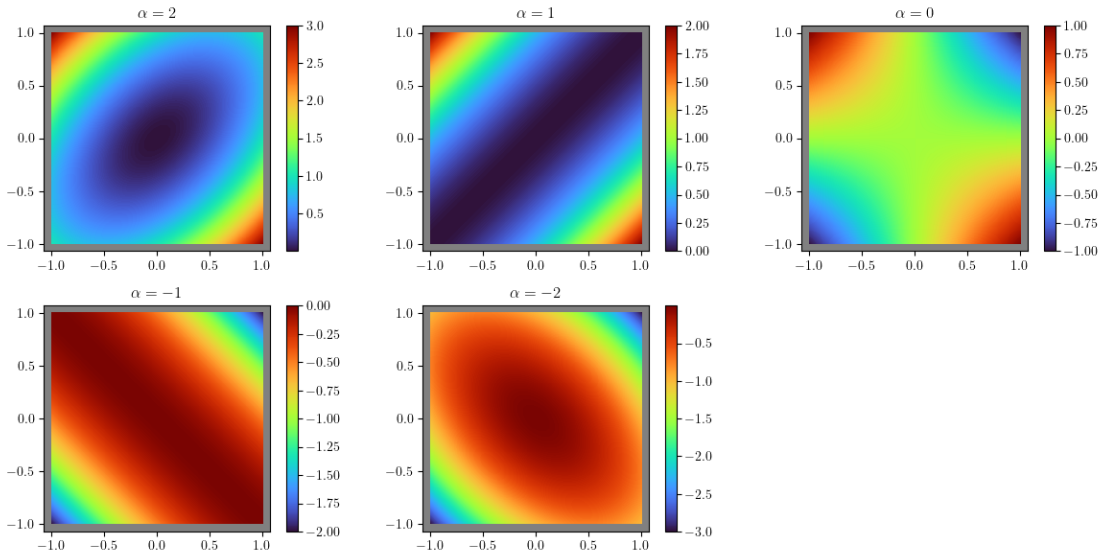


Figure 3.5: Visualization of energy landscape $E$ for SimCIM and bSB

**Overview:** This is a visual representation of energy landscape $E$ for a system with two variables and coupling term $J_{12} = J_{21} = 1$. Optimum solutions of the corresponding QUBO problem are (+,+) and (-, -). The landscape undergoes a qualitative change as $\alpha$ decreases.

**Details:** $\beta = 1$. Values of $\alpha$ are denoted on top of each subplot. $\alpha = 1$ and $\alpha = -1$ are critical values where some eigenvalues of $D^2E$ change their sign, thus qualitatively changing the landscape. The gray border represents the wall beyond which the dynamics cannot occur.

## 3.8 Simulated Coherent Ising machine (SimCIM)

This section is based on [37].

### Dynamics

SimCIM is a gradient dynamical system with time-dependent energy function 3.38. The vector field describing this dynamical system is thus given by the negative gradient of the energy function and is given by (3.40)

$$\mathbf{f}(\mathbf{x};\alpha,\beta) = -\frac{\partial E}{\partial \mathbf{x}}^T(\mathbf{x};\alpha,\beta)$$

The vector field is precisely equals $\mathbf{f}$ for points in the interior region, i.e. for $\mathbf{x} \in \langle -1,+1\rangle^n$. We want to constrain the dynamics to the region $[-1,+1]^n$ so we will stop the dynamics along certain axis if we hit the wall, i.e. the boundary of $[-1,+1]^n$. Writing this formally we get that vector field is

$$\mathbf{g}(\mathbf{x};\alpha,\beta) \coloneqq A(\mathbf{x};\alpha,\beta)\cdot\mathbf{f}(\mathbf{x};\alpha,\beta) \tag{3.48}$$

where diagonal matrix $A$ regulates which components should evolve and which should be paused because they hit the boundary. That is, we have $A(\mathbf{x};\alpha,\beta) = diag(a_1,\ldots,a_n)$; $a_i = 1$ if $x_i \in \langle -1,+1\rangle$ or $(x_i \in \{-1,+1\}\ \&\ x_i f_i < 0)$, $a_i = 0$ otherwise. This gives the dynamical system

$$\dot{\mathbf{x}}(t) = \mathbf{g}(\mathbf{x}(t);\alpha(t),\beta(t)) \tag{3.49}$$

which written out component-wise without denoting the dependence on time reads

$$\begin{cases} \dot{x}_i = f_i = -\alpha x_i + \beta\sum_j J_{ij}x_j, & \text{when } x_i \in \langle -1,+1\rangle \\ & \quad \text{or } (x_i \in \{-1,+1\}\ \&\ x_i f_i < 0) \\ \dot{x}_i = 0, & \text{otherwise} \end{cases} \tag{3.50}$$

Parameter $\beta > 0$ is kept constant through time but it is essential that parameter $\alpha(t)$ is monotonically decreased in time as it will be responsible for generating bifurcations.

### Algorithm

SimCIM algorithm is a numerical simulation of the above dynamical system by Euler method.

**Algorithm 6** SimCIM

Initialize vector $\mathbf{x}^{(0)}$ randomly around $\mathbf{0}$
**for** $k$ in range(0, $N_{\text{iter}}$) **do**
    $x_i^{(k+1)} \leftarrow x_i^{(k)} + \left(-\alpha(t_k)x_i^{(k)} + \beta \sum_{j=1}^{n} J_{ij}x_j^{(k)}\right)\Delta_t, \quad i = 1, \ldots, n$
    $x_i^{(k+1)} \leftarrow \phi(x_i^{(k+1)}), \quad i = 1, \ldots, n$
**end for**

In algorithm 6, $\phi$ is the clamping function given by

$$\phi(x) = \begin{cases} -1, & \text{for } x < -1 \\ x, & \text{for } -1 \leq x \leq 1 \\ 1, & \text{for } x > -1 \end{cases} \tag{3.51}$$

On figure 3.6 we observe that until $\alpha$ crosses the first bifurcation point ($\alpha = 1$), the origin is a stable fixed point which attracts the system. As $\alpha$ crosses the first bifurcation point, the landscape function $E$ changes such that the directions corresponding to $(+,+)$ and $(-,-)$ become directions where $E$ decreases, while directions $(+,-)$ and $(-,+)$ stay directions of increasing $E$. That's why in the top-right and bottom-left subplots we have that the system evolves straight to the point $(-1,-1)$. In the bottom-middle figure the system also evolves towards the $(-,-)$ configuration, but it hits the wall $y = -1$ in the meantime and after that continues evolving only in $x$ component, and finally ends up in the state $(-1,-1)$.

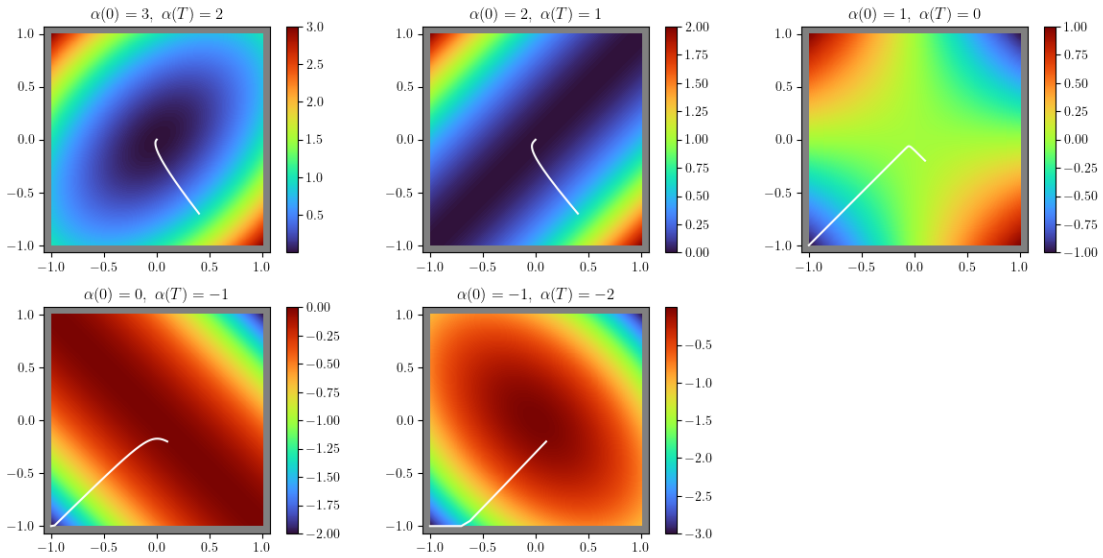


Figure 3.6: SimCIM algorithm

**Overview:** This is a visual representation of SimCIM algorithm on a system with two variables with coupling term $J_{12} = J_{21} = 1$. Optimum solutions of the corresponding QUBO problem are (+,+) and (-, -). In the top-left and top-right subplots, the parameter $\alpha$ did not cross the first bifurcation point so the dynamical system collapses towards the origin (starting far away from it). In all other subplots, the dynamical system starts close to the origin, but converges to QUBO optimum solutions.

**Details:** The gray border represents the wall beyond which the dynamics cannot occur. The white line represents the trajectory obtained by running the SimCIM algorithm. Each subplot corresponds to a new run of the algorithm. The energy landscape $E$ is plotted only for the final value of $\alpha$, that is $E(\cdot; \alpha(T), \beta)$ is plotted on each of the subplots. $\beta = 1$. In each subplot, $\alpha$ decreases linearly from starting to ending value. Starting and ending values are respectively denoted as $\alpha(0)$ and $\alpha(T)$ on top of each subplot. Number of iterations is 200, and the step size is $\Delta_t = 0.05$. Initial positions are $\mathbf{x}(0) = (0.4, -0.7)$ for upper left and upper middle subplot, while $\mathbf{x}(0) = (0.1, -0.2)$ for the other plots.

## Adding momentum

Since SimCIM is essentially a gradient descent algorithm, it makes sense to add momentum to it in order to make the convergence faster, similarly as it was done for CIM in 3.3, based on 3.3 and [35]. Momentum is acutually proposed in the SimCIM article [37] as well (as opposed to CIM).

First we choose the momentum parameter $0 \leq \gamma \leq 1$. Momenta will be stored in vector $\mathbf{y}$.

**Algorithm 7** SimCIM with momentum

1: Initialize vectors $\mathbf{x}^{(0)}, \mathbf{y}^{(0)}$ randomly around $\mathbf{0}$
2: **for** $k$ in range(0, $N_{\text{iter}}$) **do**
3: $\quad y_i^{(k+1)} \leftarrow \gamma y_i^{(k)} + \left(-\alpha(t_k) x_i^{(k)} + \beta \sum_{j=1}^{n} J_{ij} x_j^{(k)}\right), \quad i = 1, \ldots, n$
4: $\quad x_i^{(k+1)} \leftarrow x_i^{(k)} + y_i^{(k+1)} \Delta_t, \quad i = 1, \ldots, n$
5: $\quad y_i^{(k+1)} \leftarrow 0, \quad \text{if } x_i^{(k+1)} \notin [-1, +1], \quad i = 1, \ldots, n$
6: $\quad x_i^{(k+1)} \leftarrow \phi(x_i^{(k+1)}), \quad i = 1, \ldots, n$
7: **end for**

For momentum parameter $\gamma = 0$, SimCIM with momentum 7 becomes precisely SimCIM 6.

**Remark 3.8.1.** *'SimCIM' and 'SimCIM with momentum' will often be used interchangeably, referring to SimCIM with momentum, unless specified otherwise.*

### Exact solution

In this section, an exact solution of SimCIM dynamical system is provided. This solution holds until the system hits the wall. After it hits the wall, a similar approach for deriving the exact solution could be used but considering only those variables that did not hit the wall, while fixing others.

Let us revise the dynamical system describing SimCIM (3.50) for the interior region i.e. until none of the components hits the wall

$$\dot{\mathbf{x}} = -\alpha\mathbf{x} + \beta J\mathbf{x} \tag{3.52}$$

Let $U^T JU = \Lambda$ be an orthogonal diagonalization of $J$ and denote the change of variables $\bar{\mathbf{x}} = U^T\mathbf{x}$. As described in the previous subsection, the system of coupled equations (3.52) is brought down to

$$\dot{\bar{x}}_i = -\alpha\bar{x}_i + \beta\lambda_i\bar{x}_i, \quad i = 1, \ldots, n \tag{3.53}$$

which is a system of uncoupled equations of the same form, so it is sufficient to consider only single one of them. So we drop the index $i$ in what follows. Suppose that $\alpha$ has the linear form $\alpha(t) = \alpha_0 - \frac{\alpha_0-\alpha_1}{T}t$. We are thus interested in solving equation of the form

$$\dot{\bar{x}}(t) = (\beta\lambda - \alpha_0)\bar{x}(t) + \frac{\alpha_0 - \alpha_1}{T}t\bar{x}(t) \tag{3.54}$$

But this is a linear equation of the first order, so it has the solution

$$\bar{x}(t) = \bar{x}(0)\exp\left(\frac{\alpha_0 - \alpha_1}{2T}t^2 + (\beta\lambda - \alpha_0)t\right) \tag{3.55}$$

Since $\alpha$ is decreasing, i.e. $\alpha_0 > \alpha_1$, we see from equation (3.55) that the solution will eventually start rapidly increasing because of the $t^2$ term. For $\alpha_0 - \beta\lambda \leq 0$ the solution will immediately start expanding, while for $\alpha_0 - \beta\lambda > 0$ the solution will first collapse towards 0 and then at some point in time start expanding.

Thus, SimCIM dynamical system has an exact component-wise solution given by (3.55) in the reference frame which diagonalizes $J$. Changing the reference frame back to $\mathbf{x} = U\bar{\mathbf{x}}$ we obtain the exact solution of SimCIM until it hits the wall.

## 3.9 Ballistic Simulated Bifurcation (bSB)

This section is based on [19].

## Dynamics

Ballistic Simulated Bifurcation (bSB) is a Hamiltonian dynamical system with time-dependent Hamiltonian given by

$$H(\mathbf{x}, \mathbf{y}; \alpha, \beta, m) = \frac{m}{2} \sum_i y_i^2 + E(\mathbf{x}; \alpha, \beta) \tag{3.56}$$

where $E$ is the potential function given by 3.38. The first part of the summation corresponds to kinetic energy depending only on momenta $\mathbf{y}$ while the second part corresponds to potential energy depending only on positions $\mathbf{x}$. Parameters $m, \beta > 0$ are constant in time, while parameter $\alpha$ decreases monotonically (usually linearly) from certain starting value $\alpha_0$ to certain ending value $\alpha_1$. The system of differential equations governing the dynamics for this Hamiltonian system is thus given by

$$\begin{aligned} \dot{\mathbf{x}}^T(t) &= \frac{\partial H}{\partial \mathbf{y}}(\mathbf{x}(t), \mathbf{y}(t); \alpha(t), \beta, m) \\ \dot{\mathbf{y}}^T(t) &= -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}(t), \mathbf{y}(t); \alpha(t), \beta, m) \\ &= -\frac{\partial E}{\partial \mathbf{x}}(\alpha(t), \beta, m) = \mathbf{f}^T(\mathbf{x}, \alpha(t), \beta) \end{aligned} \tag{3.57}$$

where

$$\mathbf{f}(\mathbf{x}; \alpha, \beta) = -\frac{\partial E}{\partial \mathbf{x}}^T (\mathbf{x}; \alpha, \beta)$$

as in (3.40).

This holds in the interior region, i.e. where $\mathbf{x} \in \langle -1, +1 \rangle^n$. We want to model the boundary region as a perfectly inelastic wall. When certain component $x_i$ hits the boundary, it will lose all kinetic energy in that direction i.e. $y_i$ will become 0 and thus prevent the dynamical system from going out of the allowed region.

Written out further component-wise and omitting the time variable we get the system describing bSB

$$\begin{cases} \dot{x}_i &= m y_i \\ \dot{y}_i &= -\alpha x_i + \beta \sum_j J_{ij} x_j, \quad \text{when } x_i \in \langle -1, +1 \rangle \text{ or } (x_i \in \{-1, +1\} \;\&\; x_i f_i < 0) \\ y_i &= 0, \quad \text{if } x_i \in \{-1, +1\} \;\&\; x_i f_i > 0 \end{cases} \tag{3.58}$$

## Algorithm

**Algorithm 8** bSB

1: Initialize vectors $\mathbf{x}^{(0)}, \mathbf{y}^{(0)}$ randomly around $\mathbf{0}$
2: **for** $k$ in range(0, $N_{\text{iter}}$) **do**
3: $\quad y_i^{(k+1)} \leftarrow y_i^{(k)} + \left(-\alpha(t_k)x_i^{(k)} + \beta \sum_{j=1}^{n} J_{ij}x_j^{(k)}\right)\Delta_t, \quad i = 1, \ldots, n$
4: $\quad x_i^{(k+1)} \leftarrow x_i^{(k)} + m y_i^{(k+1)}\Delta_t, \quad i = 1, \ldots, n$
5: $\quad y_i^{(k+1)} \leftarrow 0, \quad \text{if } x_i^{(k+1)} \notin [-1, +1], \quad i = 1, \ldots, n$
6: $\quad x_i^{(k+1)} \leftarrow \phi(x_i^{(k+1)}), \quad i = 1, \ldots, n$
7: **end for**

In algorithm 8, $\phi$ is the clamping function defined by (3.51).

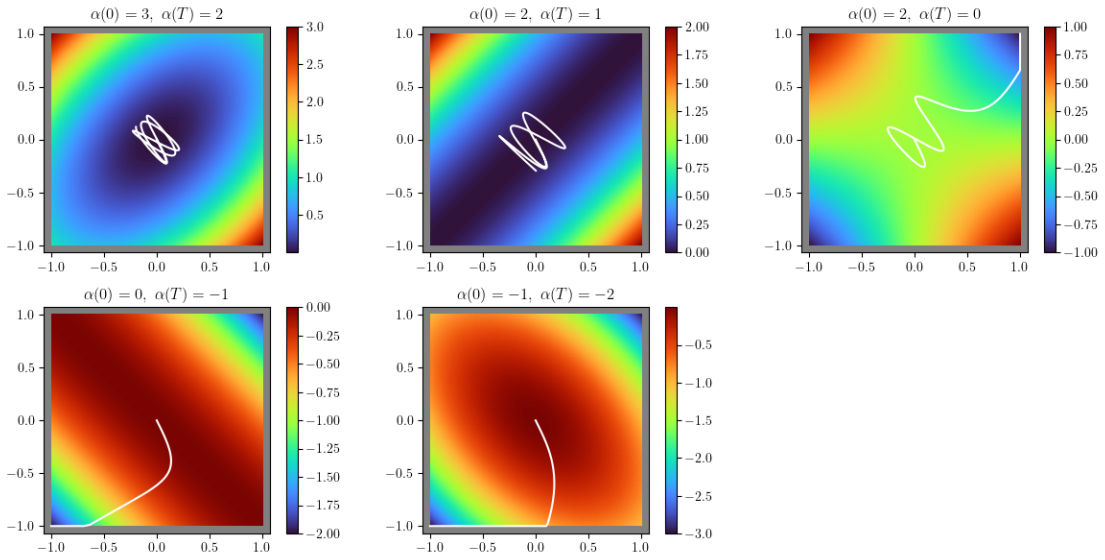


Figure 3.7: bSB algorithm

**Overview:** This is a visual representation of bSB algorithm on a system with two variables with coupling term $J_{12} = J_{21} = 1$. Optimum solutions of the corresponding QUBO problem are (+,+) and (-, -). In the top-left and top-middle subplots, the parameter $\alpha$ did not cross the first bifurcation point so the dynamical system circulates around the origin. In the top-right subplot, and in the bottom subplots, the dynamical system converges to QUBO optimum solutions.

**Details:** The gray border represents the wall beyond which the dynamics cannot occur. The white line is the trajectory obtained by running the bSB algorithm. Each subplot corresponds to a new run of the algorithm. The energy landscape $E$ is plotted only for the final value of $\alpha$, that is $E(\cdot; \alpha(T), \beta)$ is plotted on each of the subplots. $\beta = 1$. In each subplot, $\alpha$ decreases linearly from starting to ending value. Starting and ending values are respectively denoted as $\alpha(0)$ and $\alpha(T)$ on top of each subplot. Number of iterations is 250, and the step size is $\Delta_t = 0.05$. Initial positions are $\mathbf{x}(0) = (0.0, 0.0)$ and the initial momenta are $\mathbf{y} = (0.2, -0.4)$.

## Exact solution

In this section, an exact solution of the bSB dynamical system is provided. This solution holds until the system hits the wall. After it hits the wall, a similar approach for deriving the exact solution could be used but considering only those variables that did not hit the wall, while fixing others.

Let us revise the dynamical system describing bSB or the interior region i.e. until none of the components hit the wall

$$\begin{aligned} \ddot{\mathbf{x}} &= m\mathbf{y} \\ \ddot{\mathbf{y}} &= -\alpha\mathbf{x} + \beta J\mathbf{x} \end{aligned} \tag{3.59}$$

Let $U^T JU = \Lambda$ be an orthogonal diagonalization of $J$ and denote the change of variables $\bar{\mathbf{x}} = U^T\mathbf{x}$. As described in the previous subsection, the system of coupled equations (3.59) is brought down to

$$\ddot{\bar{x}}_i = m(\beta\lambda_i - \alpha)\bar{x}_i, \quad i = 1, \dots, n \tag{3.60}$$

which is a system of uncoupled equations of the same form, so it is sufficient to consider only single one of them. So we drop the index $i$ in what follows. $\alpha$ is a linear function so suppose it has the form $\alpha(t) = \alpha_0 + \frac{\alpha_1 - \alpha_0}{T}t$ . Now, we will shift and scale the solution to reduce the problem to Airy equation (2.53). Define

$$x'(t) \coloneqq \bar{x}\left(\sqrt[3]{\frac{T}{m(\alpha_0 - \alpha_1)}}t + \frac{(\beta\lambda - \alpha_0)T}{\alpha_1 - \alpha_0}\right) \tag{3.61}$$

Now calculate

$$\begin{aligned} \ddot{x}'(t) &= \sqrt[3]{\frac{T}{m(\alpha_0 - \alpha_1)}}^2 \ddot{\bar{x}}\left(\sqrt[3]{\frac{T}{m(\alpha_0 - \alpha_1)}}t + \frac{(\beta\lambda - \alpha_0)T}{\alpha_1 - \alpha_0}\right) \\ &= \sqrt[3]{\frac{T}{m(\alpha_0 - \alpha_1)}}^2 m\left(\beta\lambda - \alpha_0 - \frac{\alpha_1 - \alpha_0}{T}\left(\sqrt[3]{\frac{T}{m(\alpha_0 - \alpha_1)}}t + \frac{(\beta\lambda - \alpha_0)T}{\alpha_1 - \alpha_0}\right)\right)\cdot\dots \\ &\quad \cdot \bar{x}\left(\sqrt[3]{\frac{T}{m(\alpha_0 - \alpha_1)}}t + \frac{(\beta\lambda - \alpha_0)T}{\alpha_1 - \alpha_0}\right) \\ &= -\frac{\alpha_1 - \alpha_0}{T}\frac{T}{m(\alpha_0 - \alpha_1)}mtx'(t) + \cdots \\ &\quad + \sqrt[3]{\frac{T}{m(\alpha_0 - \alpha_1)}}^2 m\left(\beta\lambda - \alpha_0 - \frac{\alpha_1 - \alpha_0}{T}\cdot\frac{(\beta\lambda - \alpha_0)T}{\alpha_1 - \alpha_0}\right)x'(t) \\ &= tx'(t) \end{aligned} \tag{3.62}$$

so $x'$ solves the Airy equation (2.53).

Conversely, let $x'(t) = C_1A(t)+C_2B(t)$ be any solution of Airy equation (2.53) ($C_1, C_2 \in \mathbb{R}$ are arbitrary constants, while $A, B$ are linearly independent solutions of Airy equation). By taking

$$\bar{x}(t) = x'\left(\sqrt[3]{\frac{m(\alpha_0-\alpha_1)}{T}}\left(t-\frac{(\beta\lambda-\alpha_0)T}{\alpha_1-\alpha_0}\right)\right) \tag{3.63}$$

we get

$$\begin{aligned}
\ddot{\bar{x}}(t) &= \sqrt[3]{\frac{m(\alpha_0-\alpha_1)}{T}}^2 \ddot{x}'\left(\sqrt[3]{\frac{m(\alpha_0-\alpha_1)}{T}}\left(t-\frac{(\beta\lambda-\alpha_0)T}{\alpha_1-\alpha_0}\right)\right) \\
&= \sqrt[3]{\frac{m(\alpha_0-\alpha_1)}{T}}^2 \cdot \dots \\
&\quad \cdot\left(\sqrt[3]{\frac{m(\alpha_0-\alpha_1)}{T}}\left(t-\frac{(\beta\lambda-\alpha_0)T}{\alpha_1-\alpha_0}\right)\right)\cdot x'\left(\sqrt[3]{\frac{m(\alpha_0-\alpha_1)}{T}}\left(t-\frac{(\beta\lambda-\alpha_0)T}{\alpha_1-\alpha_0}\right)\right) \\
&= \frac{m(\alpha_0-\alpha_1)}{T}\left(t-\frac{(\beta\lambda-\alpha_0)T}{\alpha_1-\alpha_0}\right)\bar{x}(t) \\
&= m\left(-\frac{(\alpha_1-\alpha_0)}{T}t+\beta\lambda-\alpha_0\right)\bar{x}(t) \\
&= m\left(\beta\lambda-\alpha(t)\right)\bar{x}(t)
\end{aligned} \tag{3.64}$$

which is exactly (3.60).

Thus, bSB dynamical system has an exact solution given by (3.63) in the reference frame which diagonalizes $J$. Choosing constants $C_1, C_2$ such that initial conditions are satisfied, and changing the reference frame back to $\mathbf{x} = U\bar{\mathbf{x}}$ we obtain an exact solution of bSB until it hits the wall.

## 3.10 Mechanism of SimCIM and bSB

Let us explain the mechanism of SimCIM and bSB. Although SimCIM and bSB are nonautonomous dynamical systems, by the connection theorem 2.1.9, in order to analyze such system locally (in time), it is sufficient to consider it as an autonomous system with frozen vector field at some time instance of interest. The mechanism of SimCIM and bSB can now be understood in the light of examples 2.1.11 and 2.1.10 which provide solutions of linear gradient and Hamiltonian systems. By changing the reference frame to the one that diagonalizes the coupling matrix $J$, one obtains a comprehensible behaviour of the dynamical system. As the parameter $\alpha$ decreases through time, the qualitative appearance of $E$ changes. That is, some of the largest eigenvalues of $\frac{\partial^2 E}{\partial \mathbf{x}^2} = -\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ cross zero and become negative. Until that first bifurcation point, SimCIM dynamical system will collapse towards

the origin, and bSB will revolve around the origin. After the first bifurcation point, the system will start to rapidly expand towards eigenvectors corresponding to smallest (now negative) eigenvalues of $\frac{\partial^2 E}{\partial \mathbf{x}^2}$ i.e. largest eigenvalues of $J$. This way, the system provides good approximate solutions of QUBO at early stages of the algorithm. This behaviour holds until some components hit the wall. After hitting the wall, this argument does not hold anymore in the reference frame which diagonalizes $J$. However, a similar argument can be applied, but one needs to consider only those components which have not hit the wall yet and diagonalize such submatrix.

As discussed in [39] for CIM, those variables which bifurcate from the origin at early stages *usually* do not return to the origin nor change their sign anymore. This seems consistent with the clipping mechanism for SimCIM and bSB, and could be potentially used for analyzing the system which is reduced to only those variables which did not hit the wall yet.

There is one more mechanism applied both to SimCIM and bSB and it is regarding the final state of the system. The moment when $\alpha(t) = 0$ has a significant role because at that point, if the system has bifurcated enough, it will provide a 1-opt solution. This will be discussed in the following two sections. In terms of parameter settings, it would make sense to stop the dynamics at that point and thus make $\alpha_1 = 0$. However, sometimes it is found that making $\alpha_1$ slightly smaller than 0 provides better solution. It does not hurt making $\alpha_1$ smaller than 0 as long as we track the best possible solution found, so we still have a chance of capturing that 1-opt solution at $\alpha(t) = 0$, but in case that required conditions are not met at $\alpha(t) = 0$, we still have a chance of finding better solution for $\alpha(t) < 0$. Making $\alpha_1$ too small however usually is not good because we want $\alpha(t)$ to change slowly in time, in order for the system to be able to evolve properly.

Let us now explain why SimCIM and bSB have a tendency of generating 1-opt solutions for QUBO.

## Final state of SimCIM

Let us observe what happens at (usually the final) time $\alpha(t) = 0$. Assume that the system has bifurcated completely in a sense that $\forall i, x_i \in \{-1, +1\}$ and assume that the system has stopped in a sense that $\forall i, g_i = 0$. This implies $x_i f_i \geq 0$. The solution provided by SimCIM is, as usual, $\sigma_i := \text{sign}(x_i)$ which in this case provides $\sigma_i = x_i$. We have

$$\beta \sum_{j=1}^{n} J_{ij}\sigma_i\sigma_j = x_i\beta \sum_{j=1}^{n} J_{ij}x_j = x_i f_i \geq 0 \tag{3.65}$$

Flipping the $i$-th binary variable provides a change in QUBO functional (see (1.22))

$$\begin{aligned}\Delta_i Q &= Q(\sigma_1, \ldots, \sigma_{i-1}, -\sigma_i, \sigma_{i+1}, \ldots, \sigma_n) - Q(\sigma_1, \ldots, \sigma_{i-1}, \sigma_i, \sigma_{i+1}, \ldots, \sigma_n) \\ &= -4 \sum_{j=1}^{n} J_{ij} \sigma_i \sigma_j\end{aligned} \tag{3.66}$$

This, combined with (3.65), provides

$$\Delta_i Q \leq 0 \tag{3.67}$$

so for the maximization task which we are interested in here, flipping the value of $\sigma_i$ cannot improve the solution. If (3.69) holds for each $i$, then the solution obtained by SimCIM algorithm is, by definition 1.1.3, a 1-opt solution.

## Final state of bSB

Let us assume that at the final time, each $x_i$ has obtained a discrete value of either $-1$ or $+1$. That is, $x_i \in \{-1, +1\}, \forall i = 1, \ldots, n$. The vector field corresponding to momentum at the final time is, because $\alpha(T) = 0$,

$$\dot{y}_i = f_i \coloneqq \beta \sum_{j=1}^{n} J_{ij} x_j \tag{3.68}$$

Assume for some $i$ that its acceleration is pointing out of the bounded region, meaning that $x_i f_i > 0$. The solution provided by bSB is, as usual, $\sigma_i \coloneqq \text{sign}(x_i)$ which in this case provides $\sigma_i = x_i$. From (3.68) we have

$$\beta \sum_{j=1}^{n} J_{ij} \sigma_i \sigma_j = x_i \beta \sum_{j=1}^{n} J_{ij} x_j = x_i f_i > 0 \tag{3.69}$$

Flipping the $i$-th binary variable provides a change in QUBO functional

$$\begin{aligned}\Delta_i Q &= Q(\sigma_1, \ldots, \sigma_{i-1}, -\sigma_i, \sigma_{i+1}, \ldots, \sigma_n) - Q(\sigma_1, \ldots, \sigma_{i-1}, \sigma_i, \sigma_{i+1}, \ldots, \sigma_n) \\ &= -2 \sum_{j=1}^{n} (J_{ij} + J_{ji}) \sigma_i \sigma_j \\ &= -4 \sum_{j=1}^{n} J_{ij} \sigma_i \sigma_j\end{aligned} \tag{3.70}$$

This, combined with (3.69), provides

$$\Delta_i Q < 0 \tag{3.71}$$

so for the maximization task which we are interested in here, flipping the value of $\sigma_i$ will worsen the solution. If (3.69) holds for each $i$, then the solution obtained by bSB algorithm is an 1-opt solution.

Thus, if the dynamics of the bSB has stopped at some point in time and does not change until final time, then the solution of bSB algorithm is a 1-opt solution.

## 3.11 Relation between SimCIM and bSB

The relation between SimCIM and bSB is completely analogous to the relation between CIM and SB. It will be written-out here again for completeness.

For momentum parameter $\gamma = 0$, SimCIM with momentum 7 becomes precisely SimCIM 6.

Although the momentum parameter is usually kept smaller than 1, it is very interesting what happens when $\gamma = 1$. In this case, SimCIM with momentum 7 becomes precisely bSB algorithm 8. That is, we can choose hyperparameters in SimCIM with momentum in such a way that the update step becomes exactly the same as the update step of bSB algorithm.

Let us distinguish between SimCIM's and bSB's parameters with a superscript SimCIM and bSB.

For given parameters for bSB algorithm, $\alpha_0^{\text{bSB}}, \alpha_1^{\text{bSB}}, \beta^{\text{bSB}}, \Delta_t^{\text{bSB}}, m^{\text{bSB}}$ let us choose a special set of parameters for SimCIM defined with

$$\begin{aligned} \gamma^{\text{SimCIM}} &= 1 \\ \alpha_0^{\text{SimCIM}} &= \alpha_0^{\text{bSB}} \cdot \Delta_t^{\text{bSB}} \\ \alpha_1^{\text{SimCIM}} &= \alpha_1^{\text{bSB}} \cdot \Delta_t^{\text{bSB}} \\ \beta^{\text{SimCIM}} &= \beta^{\text{bSB}} \cdot \Delta_t^{\text{bSB}} \\ \Delta_t^{\text{SimCIM}} &= m^{\text{bSB}} \Delta_T^{\text{bSB}} \end{aligned} \tag{3.72}$$

**Algorithm 9** bSB, revisited

1: Initialize vectors $\mathbf{x}^{(0)}, \mathbf{y}^{(0)}$ randomly around $\mathbf{0}$
2: **for** $k$ in range(0, $N_{\text{iter}}$) **do**
3: $\quad y_i^{(k+1)} \leftarrow y_i^{(k)} + \left(-\alpha^{\text{bSB}}(t_k) x_i^{(k)} + \beta^{\text{bSB}} \sum_{j=1}^n J_{ij} x_j^{(k)}\right) \Delta_t^{\text{bSB}}, \quad i = 1, \ldots, n$
4: $\quad x_i^{(k+1)} \leftarrow x_i^{(k)} + m^{\text{bSB}} y_i^{(k+1)} \Delta_t^{\text{bSB}}, \quad i = 1, \ldots, n$
5: $\quad y_i^{(k+1)} \leftarrow 0, \quad \text{if } x_i^{(k+1)} \notin [-1, +1], \quad i = 1, \ldots, n$
6: $\quad x_i^{(k+1)} \leftarrow \phi(x_i^{(k+1)}), \quad i = 1, \ldots, n$
7: **end for**

Plugging in the above parameters into SimCIM algorithm with momentum 7 we get

**Algorithm 10** SimCIM with momentum mimicking bSB

1: Initialize vectors $\mathbf{x}^{(0)}, \mathbf{y}^{(0)}$ randomly around $\mathbf{0}$
2: **for** $k$ in range(0, $N_{\text{iter}}$) **do**
3: $\quad y_i^{(k+1)} \leftarrow y_i^{(k)} + \left(-\alpha^{\text{bSB}}(t_k) \Delta_t^{\text{bSB}} x_i^{(k)} + \beta^{\text{bSB}} \Delta_t^{\text{bSB}} \sum_{j=1}^n J_{ij} x_j^{(k)}\right), \quad i = 1, \ldots, n$
4: $\quad x_i^{(k+1)} \leftarrow x_i^{(k)} + y_i^{(k+1)} m^{\text{bSB}} \Delta_t^{\text{bSB}}, \quad i = 1, \ldots, n$
5: $\quad y_i^{(k+1)} \leftarrow 0, \quad \text{if } x_i^{(k+1)} \notin [-1, +1], \quad i = 1, \ldots, n$
6: $\quad x_i^{(k+1)} \leftarrow \phi(x_i^{(k+1)}), \quad i = 1, \ldots, n$
7: **end for**

This shows that SimCIM with momentum is able to exactly reproduce bSB steps if parameters are chosen accordingly. In this context, SimCIM is a generalization of bSB algorithm.

## 3.12 First bifurcation point

As it was discussed earlier in this chapter, we want to chose parameters such that at the beginning of the dynamics we are already at the first bifurcation point or have even crossed

it. Although the best parameter setting should be determined experimentally, by observing empirical evidence for which parameter provides the best solution, we expect that it will be something around the first bifurcation point.
Let us revise the first bifurcation points.
For CIM and SB this is 3.8 i.e.

$$\mu = -\frac{\beta}{\alpha}\lambda_{\max}(J)$$

For SimCIM and bSB this is 3.46 i.e.

$$\alpha = \beta\lambda_{\max}(J)$$

We see that both of these depend on the largest eigenvalue of the coupling matrix $J$. So, how do we determine the first bifurcation point exactly?

The first approach would be to use some standard library with numerical algorithms and use it to numerically approximate the largest eigenvalue of $J$. However, for real applications this might be undesirable (for example if the matrix is very large this will be time consuming).

The workaround, proposed in [23], is to *guess* the approximation of the largest eigenvalue. As stated in [23], according to Wigner's semicircle law in random matrix theory, $\lambda_{\max}$ is approximately given by

$$\lambda_{\max} \approx 2\sqrt{n}\sigma \tag{3.73}$$

where $n$ is the number of variables and $\sigma$ is the standard deviation of the nondiagonal elements of $J$, which can be easily calculated from $J$.

# Chapter 4

# Experiments

Physics-inspired dynamical systems presented in the previous chapter can be efficiently simulated on conventional hardware devices such as a CPU. Furthermore, since they almost completely comprise of a vast amount of simple add/multiply operations, they can be simulated even more efficiently on parallelized hardware devices such as GPUs or FPGAs. This enables them to easily scale to large instances of QUBO, MAX-CUT, and other problems of interest.

This chapter focuses on presenting the results of various experiments, explaining them, and discussing them further, all in order to investigate the performance of algorithms presented in chapter 3. These simulations include Coherent Ising Machine (CIM), Simulated Bifurcation (SB), Simulated Coherent Ising Machine (SimCIM), and Ballistic Simulated Bifurcation (bSB) algorithms. Besides these algorithms, some results obtained by classical algorithms are provided for comparison.

Finally, a new technique called *dropout* is presented. It acts as adding noise in a *meaningful* way to the above algorithms, often providing improvement in solution quality.

## 4.1 GSet Dataset

GSet is a benchmark dataset publicly available on the link [4]. It consists of many weighted graph instances with different graph topologies. On some instances, the edges' weight is always 1, while on other instances, the edges' weight can be either $+1$ or negative $-1$.

Since the MAX-CUT problem is equivalent to QUBO problem 1.2.5, this dataset in fact serves as a QUBO benchmark. The best known cut value for each graph instance is provided according to the article [16], and will be referred to as 'max known cut' in the following tables. These solutions are, in fact, also obtained by their heuristic algorithm called Breakout Local Search (BLS).

The experiments will be run on first 21 instances of GSet, G1-G21. Every instance G1-G21 has 800 nodes. The number of edges is denoted in the 'edge' column. Weights can be either always +1 or combined +1 and -1 which is denoted in the column 'weights'.

## 4.2 Method

This section explains details regarding experiment setup and execution.

Each experiment consists first of parameter fine-tuning and then testing. For each algorithm, and for each GSet instance this process is started from scratch.

In order to fine-tune the parameters, some predetermined set of candidates is chosen for each parameter. This set of candidates is chosen based on manual experimenting, and determining which range for each parameter provides sensible behaviour in terms of stability and solution quality. For example, in order to fine-tune the SimCIM or bSB algorithm, we need to choose starting coefficient $\alpha_0 \in \mathbb{R}$. If we choose $\beta = 1$, it makes sense to choose $\alpha_0$ to be at the first bifurcation point $\beta\lambda_{\max}(J)$, or slightly less. So, for example, the candidates for $\alpha_0$ could be chosen to be $\beta\lambda_{\max}(J) \cdot \eta, \eta \in \{1.0, 0.9, 0.8, 0.5, 0.1\}$.

Note that the momentum was used both for CIM and SimCIM algorithms, with the possibility for the momentum parameter $\gamma$ to be fine-tuned, including set to 0 (which for CIM is the original algorithm without momentum). For example, for SimCIM the set of possible momentum parameters used was $[0, 0.8, 0.9, 0.99, 1.0]$. Both for SimCIM and CIM it turned out that the algorithm consistently provided better results with momentum strictly larger than 0.

After choosing the possible candidates for each parameter, the fine-tuning process begins. For every parameter combination, selected in a grid-search fashion, the experiment is run 5 times using different initial conditions. The result from each run is taken and the mean value of these results is recorded. A parameter combination — which turned out to be stable (in a sense that it succeeded in all 5 tries), and with the highest mean value obtained — is chosen for testing, and will be referred to as *fine-tuned parameters*.

Next, a proper testing is performed. Using these fine-tuned parameters, the experiment is run 50 times with various initial conditions. The result from each run is taken and the minimum, mean, and maximum values are recorded, and presented in the following tables in section 4.3 under the columns 'min', 'mean', and 'max'.

In order to remove the possible advantage of certain parameter combinations just because of luck, i.e. having better initial conditions, each parameter combination receives the same initial conditions. In fact, 5 initial conditions are generated at the beginning, and their copy is provided to each parameter combination at the beginning of their run.

In order to remove the effect of approximating $\lambda_{\max}$ by some proposed ways in chapter 3, for all of these experiments the value of $\lambda_{\max}$ is calculated in advance using standard numerical algorithms for obtaining spectral decomposition of a symmetric matrix.

## 4.3 Benchmarking

The following tables represent results obtained by each algorithm on instances G1-G21 of GSet. For each algorithm, for each graph instance, first the parameters are fine-tuned as described in the previous section. Then, these fine-tuned parameters are selected and a new experiment is performed with 50 tries. In each try, different initial conditions (initial positions and initial momenta) are chosen. After performing these 50 tries, the minimum, mean and maximum cut values found by the algorithm are denoted in the corresponding columns 'min', 'mean', 'max'.

The following figures represent a single run of the experiment of each algorithm on GSet instance G18. This shows how the algorithm performs through time (or actually number of iterations). On $x$-axis, the number of iterations is plotted. On the *Cut value* subplot, the value of the cut is plotted obtained by taking sign($\mathbf{x}$) as the partition of the graph. The red dotted line in this plot represents the maximum known cut from the table. For G18 it is equals to 992. On *Landscape value E* and *Potential energy E* subplots, the value of the function $E$ (defined in chapter 3) is plotted at each iteration step $E(\mathbf{x}(t); \cdot(t))$. Saturation graph represents how many variables have absolute value larger than 0.98. It shows how many variables have moved from the origin. For SimCIM and bSB it actually represents how many variables have (probably) stopped their motion. The subplots denoted with $\alpha$ or $\mu$ represent the value of corresponding parameters changing through time. The subplot denoted with *Spins norm* is the norm of vector $\mathbf{x}$. The red dashed line represents the norm of feasible set $\{-1, +1\}^n$.

For SimCIM and CIM, the coupling matrices are normalized (not a necessary step) in a sense that $J/\|J\|$ is taken in plance of $J$. That's why the $\alpha$ and $E$ might be scaled differently than expected. The purpose of these plots is to see the overall behavior — the trend of each function.

All of these plots could be used to analyze the performance of each algorithm and potentially improve algorithms' bottlenecks.

| instance | edges | weights | max known cut | min | mean | max |
|---|---|---|---|---|---|---|
| G1 | 19176 | 1 | 11624 | 11525 | 11561.2 | 11594 |
| G2 | 19176 | 1 | 11620 | 11531 | 11565.9 | 11590 |
| G3 | 19176 | 1 | 11622 | 11525 | 11566.8 | 11606 |
| G4 | 19176 | 1 | 11646 | 11567 | 11608.3 | 11627 |
| G5 | 19176 | 1 | 11631 | 11566 | 11594 | 11602 |
| G6 | 19176 | +1,-1 | 2178 | 2093 | 2142.7 | 2147 |
| G7 | 19176 | +1,-1 | 2006 | 1921 | 1955 | 1976 |
| G8 | 19176 | +1,-1 | 2005 | 1916 | 1950.3 | 1973 |
| G9 | 19176 | +1,-1 | 2054 | 1939 | 1990.9 | 2032 |
| G10 | 19176 | +1,-1 | 2000 | 1918 | 1947.4 | 1982 |
| G11 | 1600 | +1,-1 | 564 | 544 | 550.3 | 554 |
| G12 | 1600 | +1,-1 | 556 | 534 | 543.1 | 550 |
| G13 | 1600 | +1,-1 | 582 | 558 | 567.5 | 574 |
| G14 | 4694 | 1 | 3064 | 2966 | 2986.2 | 3010 |
| G15 | 4661 | 1 | 3050 | 2955 | 2975.4 | 2985 |
| G16 | 4672 | 1 | 3052 | 2959 | 2974.2 | 2984 |
| G17 | 4667 | 1 | 3047 | 2958 | 2975.4 | 2988 |
| G18 | 4694 | +1,-1 | 992 | 898 | 924.7 | 945 |
| G19 | 4661 | +1,-1 | 906 | 815 | 853.1 | 854 |
| G20 | 4672 | +1,-1 | 941 | 852 | 889.5 | 891 |
| G21 | 4667 | +1,-1 | 931 | 814 | 850.6 | 887 |

Table 4.1: CIM results on GSet

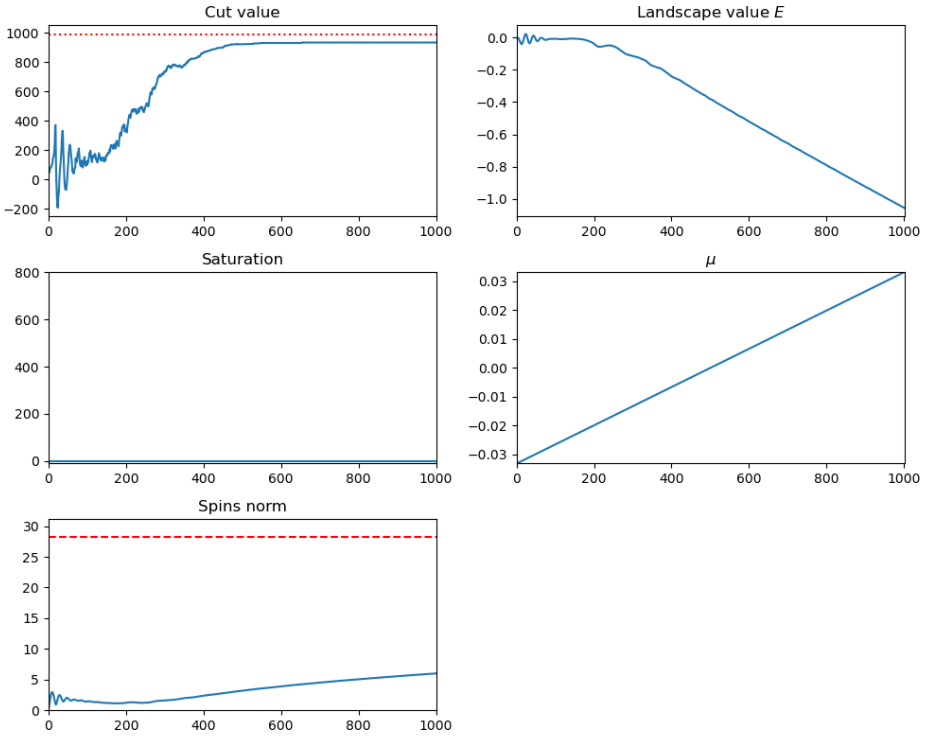


Figure 4.1: Single run of CIM algorithm on GSet instance G18

| instance | edges | weights | max known cut | min | mean | max |
|---|---|---|---|---|---|---|
| G1 | 19176 | 1 | 11624 | 11550 | 11582.2 | 11615 |
| G2 | 19176 | 1 | 11620 | 11569 | 11580 | 11599 |
| G3 | 19176 | 1 | 11622 | 11563 | 11589.8 | 11617 |
| G4 | 19176 | 1 | 11646 | 11587 | 11614.5 | 11632 |
| G5 | 19176 | 1 | 11631 | 11595 | 11607.2 | 11618 |
| G6 | 19176 | +1,-1 | 2178 | 2149 | 2158.7 | 2168 |
| G7 | 19176 | +1,-1 | 2006 | 1960 | 1978.1 | 1998 |
| G8 | 19176 | +1,-1 | 2005 | 1936 | 1977.8 | 1990 |
| G9 | 19176 | +1,-1 | 2054 | 2000 | 2020.7 | 2038 |
| G10 | 19176 | +1,-1 | 2000 | 1944 | 1968.6 | 1986 |
| G11 | 1600 | +1,-1 | 564 | 548 | 553.6 | 558 |
| G12 | 1600 | +1,-1 | 556 | 542 | 549.8 | 554 |
| G13 | 1600 | +1,-1 | 582 | 558 | 571.2 | 578 |
| G14 | 4694 | 1 | 3064 | 3004 | 3016.5 | 3033 |
| G15 | 4661 | 1 | 3050 | 2980 | 2996 | 3011 |
| G16 | 4672 | 1 | 3052 | 2980 | 3000.7 | 3018 |
| G17 | 4667 | 1 | 3047 | 2981 | 2993.7 | 3007 |
| G18 | 4694 | +1,-1 | 992 | 914 | 946.4 | 972 |
| G19 | 4661 | +1,-1 | 906 | 473 | 818 | 871 |
| G20 | 4672 | +1,-1 | 941 | 874 | 899.5 | 917 |
| G21 | 4667 | +1,-1 | 931 | 890 | 893.1 | 899 |

Table 4.2: SB results on GSet

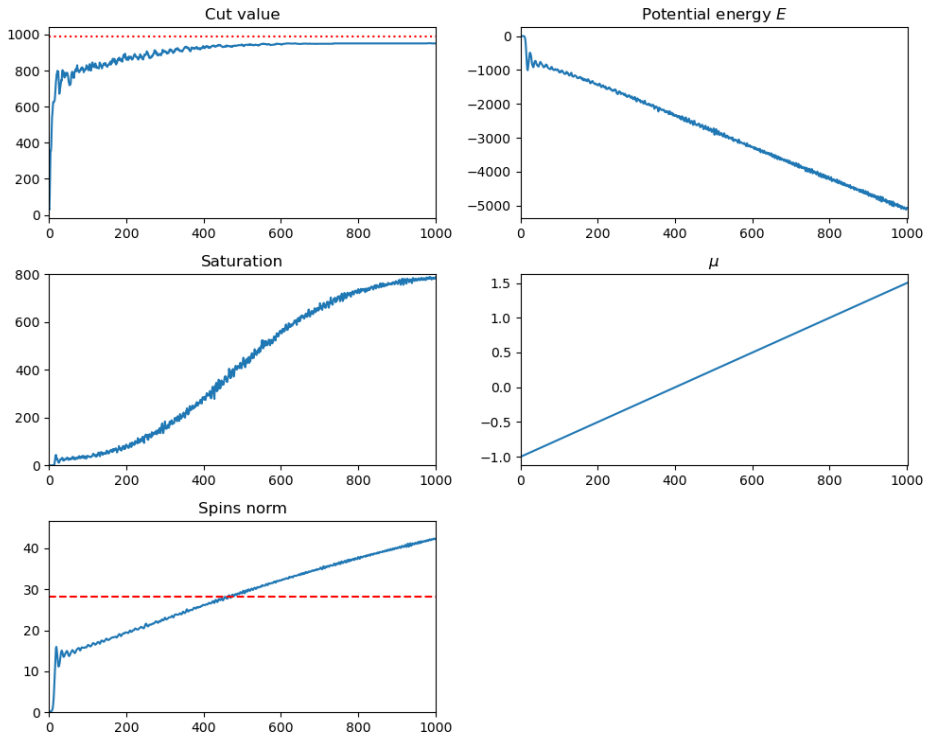


Figure 4.2: Single run of SB algorithm on GSet instance G18

| instance | edges | weights | max known cut | min | mean | max |
|---|---|---|---|---|---|---|
| G1 | 19176 | 1 | 11624 | 11566 | 11608.8 | 11623 |
| G2 | 19176 | 1 | 11620 | 11580 | 11595 | 11609 |
| G3 | 19176 | 1 | 11622 | 11568 | 11618.7 | 11622 |
| G4 | 19176 | 1 | 11646 | 11617 | 11636.8 | 11642 |
| G5 | 19176 | 1 | 11631 | 11599 | 11624.2 | 11625 |
| G6 | 19176 | +1,-1 | 2178 | 2139 | 2172.6 | 2174 |
| G7 | 19176 | +1,-1 | 2006 | 1975 | 1986.9 | 2002 |
| G8 | 19176 | +1,-1 | 2005 | 1969 | 1986.6 | 2005 |
| G9 | 19176 | +1,-1 | 2054 | 2012 | 2035.9 | 2046 |
| G10 | 19176 | +1,-1 | 2000 | 1952 | 1975.6 | 1992 |
| G11 | 1600 | +1,-1 | 564 | 544 | 549.6 | 558 |
| G12 | 1600 | +1,-1 | 556 | 536 | 544.5 | 550 |
| G13 | 1600 | +1,-1 | 582 | 558 | 569.2 | 576 |
| G14 | 4694 | 1 | 3064 | 3031 | 3033.6 | 3034 |
| G15 | 4661 | 1 | 3050 | 3022 | 3022 | 3022 |
| G16 | 4672 | 1 | 3052 | 3026 | 3027.3 | 3029 |
| G17 | 4667 | 1 | 3047 | 3021 | 3021 | 3021 |
| G18 | 4694 | +1,-1 | 992 | 964 | 964.6 | 965 |
| G19 | 4661 | +1,-1 | 906 | 869 | 878.3 | 880 |
| G20 | 4672 | +1,-1 | 941 | 926 | 926.1 | 931 |
| G21 | 4667 | +1,-1 | 931 | 907 | 913.6 | 917 |

Table 4.3: SimCIM results on GSet

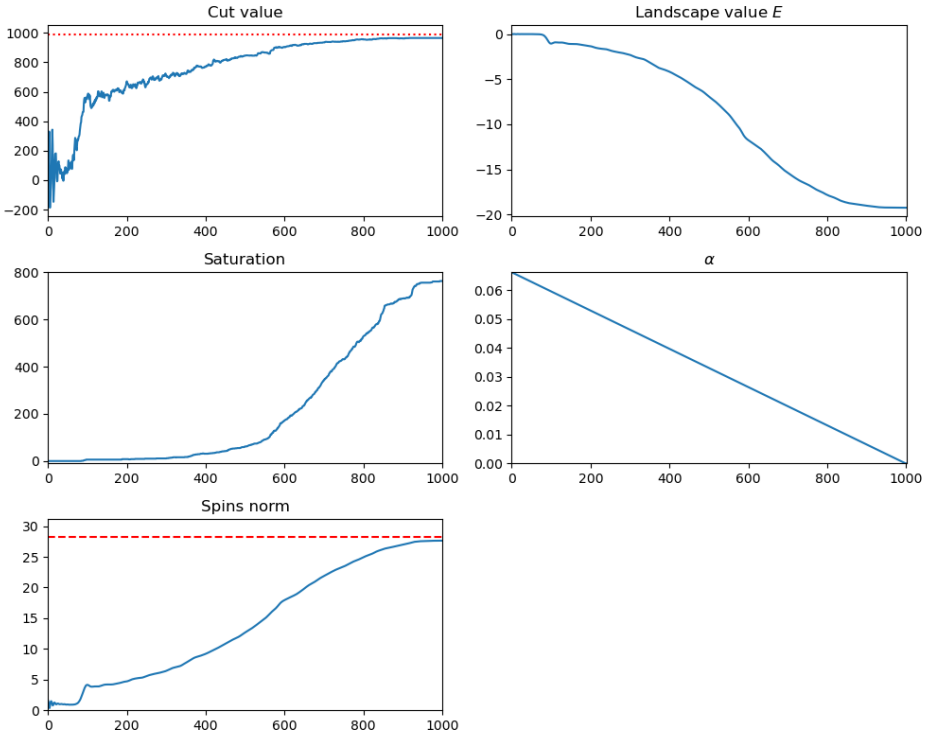


Figure 4.3: Single run of SimCIM algorithm on GSet instance G18

| instance | edges | weights | max known cut | min | mean | max |
|---|---|---|---|---|---|---|
| G1 | 19176 | 1 | 11624 | 11582 | 11608.5 | 11623 |
| G2 | 19176 | 1 | 11620 | 11569 | 11595.6 | 11612 |
| G3 | 19176 | 1 | 11622 | 11573 | 11619.1 | 11622 |
| G4 | 19176 | 1 | 11646 | 11623 | 11637.6 | 11638 |
| G5 | 19176 | 1 | 11631 | 11625 | 11625 | 11625 |
| G6 | 19176 | +1,-1 | 2178 | 2143 | 2171.3 | 2174 |
| G7 | 19176 | +1,-1 | 2006 | 1958 | 1985.2 | 1998 |
| G8 | 19176 | +1,-1 | 2005 | 1959 | 1985.7 | 2004 |
| G9 | 19176 | +1,-1 | 2054 | 2012 | 2031.9 | 2046 |
| G10 | 19176 | +1,-1 | 2000 | 1969 | 1982.9 | 1991 |
| G11 | 1600 | +1,-1 | 564 | 546 | 552.7 | 556 |
| G12 | 1600 | +1,-1 | 556 | 538 | 545.5 | 550 |
| G13 | 1600 | +1,-1 | 582 | 562 | 569.1 | 578 |
| G14 | 4694 | 1 | 3064 | 3020 | 3028.6 | 3038 |
| G15 | 4661 | 1 | 3050 | 3011 | 3021.3 | 3036 |
| G16 | 4672 | 1 | 3052 | 3016 | 3025.3 | 3038 |
| G17 | 4667 | 1 | 3047 | 3003 | 3017 | 3033 |
| G18 | 4694 | +1,-1 | 992 | 953 | 963.7 | 976 |
| G19 | 4661 | +1,-1 | 906 | 855 | 874.5 | 879 |
| G20 | 4672 | +1,-1 | 941 | 881 | 912.5 | 930 |
| G21 | 4667 | +1,-1 | 931 | 875 | 906.2 | 917 |

Table 4.4: bSB results on GSet

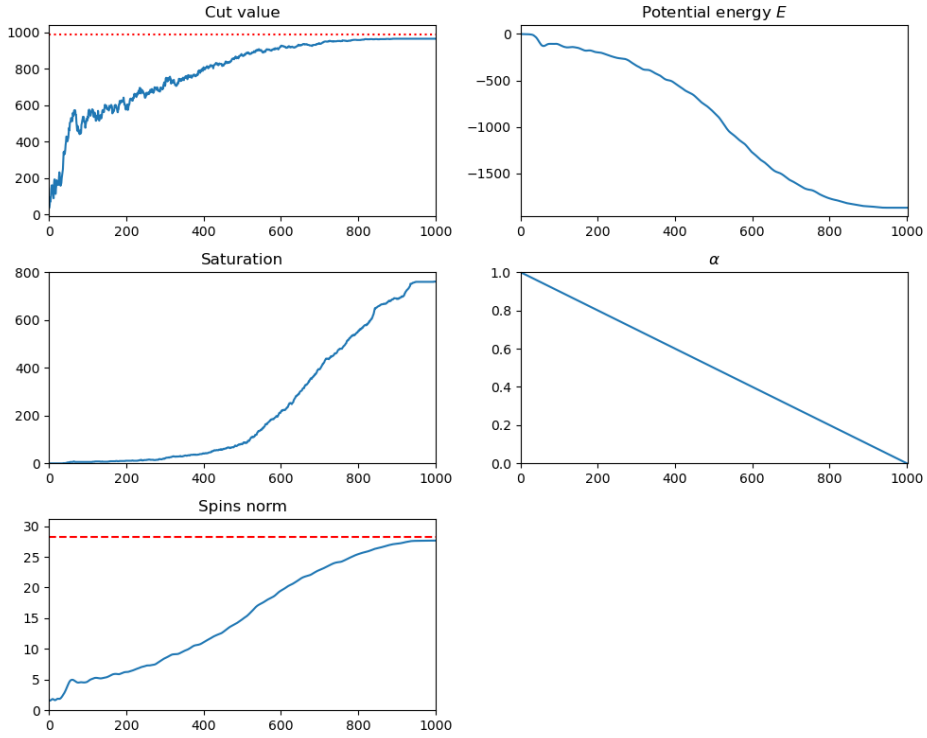


Figure 4.4: Single run of bSB algorithm on GSet instance G18

| instance | edges | weights | max known cut | CIM max | SB max | SimCIM max | bSB max | Eigen cut | Random cuts (1k) |
|---|---|---|---|---|---|---|---|---|---|
| G1 | 19176 | 1 | 11624 | 11594 | 11615 | 11623 | 11623 | 11265 | 9785 |
| G2 | 19176 | 1 | 11620 | 11590 | 11599 | 11609 | 11612 | 11248 | 9819 |
| G3 | 19176 | 1 | 11622 | 11606 | 11617 | 11622 | 11622 | 11264 | 9794 |
| G4 | 19176 | 1 | 11646 | 11627 | 11632 | 11642 | 11638 | 11290 | 9806 |
| G5 | 19176 | 1 | 11631 | 11602 | 11618 | 11625 | 11625 | 11326 | 9771 |
| G6 | 19176 | +1,-1 | 2178 | 2147 | 2168 | 2174 | 2174 | 1905 | 329 |
| G7 | 19176 | +1,-1 | 2006 | 1976 | 1998 | 2002 | 1998 | 1659 | 151 |
| G8 | 19176 | +1,-1 | 2005 | 1973 | 1990 | 2005 | 2004 | 1640 | 135 |
| G9 | 19176 | +1,-1 | 2054 | 2032 | 2038 | 2046 | 2046 | 1776 | 176 |
| G10 | 19176 | +1,-1 | 2000 | 1982 | 1986 | 1992 | 1991 | 1596 | 178 |
| G11 | 1600 | +1,-1 | 564 | 554 | 558 | 558 | 556 | 466 | 94 |
| G12 | 1600 | +1,-1 | 556 | 550 | 554 | 550 | 550 | 470 | 62 |
| G13 | 1600 | +1,-1 | 582 | 574 | 578 | 576 | 578 | 508 | 86 |
| G14 | 4694 | 1 | 3064 | 3010 | 3033 | 3034 | 3038 | 2635 | 2462 |
| G15 | 4661 | 1 | 3050 | 2985 | 3011 | 3022 | 3036 | 2550 | 2443 |
| G16 | 4672 | 1 | 3052 | 2984 | 3018 | 3029 | 3038 | 2644 | 2442 |
| G17 | 4667 | 1 | 3047 | 2988 | 3007 | 3021 | 3033 | 2493 | 2435 |
| G18 | 4694 | +1,-1 | 992 | 945 | 972 | 965 | 976 | 590 | 154 |
| G19 | 4661 | +1,-1 | 906 | 854 | 871 | 880 | 879 | 514 | 55 |
| G20 | 4672 | +1,-1 | 941 | 891 | 917 | 931 | 930 | 600 | 84 |
| G21 | 4667 | +1,-1 | 931 | 887 | 899 | 917 | 917 | 564 | 77 |

Table 4.5: Performance comparison of various algorithms on GSet

Table 4.5 compares performance of physics-inspired algorithms presented in this thesis. For all physics-inspired algorithms (CIM, SB, SimCIM, bSB), the maximum value is shown, which is found among 50 tries with various initial conditions with fine-tuned parameters as described earlier. Column 'max known cut', as earlier, represents the best known solution on particular instance, according to [16], which is also the solution found by BLS heuristic. The column 'Eigen cut' represents the cut value of the partition obtained by taking the sign of each component of the eigenvector corresponding to smallest eigenvalue of the adjacency matrix (this is an obvious heuristic based on 1.1.2). The columnt 'Random cuts (1k)' represents the maximum cut found among 1000 random cuts. These last two columns are here as a reference for the lower bound which we definitely want to beat.

## 4.4 Dropout

This section describes a potential way of improving the performance of physics-inspired algorithms, presented earlier in this chapter, by adding a special type of noise. This procedure is discussed here in terms of graphs and MAX-CUT problem instead of variables and QUBO (which is equivalent 1.2.5). There are certainly many ways to add noise, including adding just white noise to the dynamical system, making it a stochastic dynamical sys-

tem. One possible way of adding noise in a sensible way is to perturb the graph somehow in each step of the algorithm, for example by removing some vertices or edges from the graph temporarily. This way, intuitively, the dynamical system should on average, at each step, follow the same path, while opening the possibility to escape local minima or even make some components bifurcate earlier by removing some of the opposite forces acting on a single variable (often called *frustration*) . Indeed, random removal of vertices often seems to improve the performance on GSet instances. This is done in the following way. At each iteration of the algorithm (CIM, SB, SimCIM, bSB), instead of working with the full adjacency matrix, one randomly and independently picks with probability $p$ for each vertex whether it will stay or it will be temporarily removed. If vertex $i$ is removed, then all edges connected to it are temporarily removed, until completing this iteration. Everything else remains the same. There is a visual example of adding dropout in figure 4.5.

Let us observe what can happen to vertex $i$. Each of the algorithms has the term $\sum_j J_{ij}x_j$ inside the update. If vertex $i$ is not removed, then the summation $\sum_j J_{ij}x_j$ will not go through all vertices, but rather through all vertices which are not removed. If vertex $i$ is removed itself, then the summation summation $\sum_j J_{ij}x_j$ will disappear. The rest of the update step for this vertex can either be performed, or not, depending on the implementation.

This procedure is similar to adding dropout to machine learning models during training [25]. This procedure will thus also be referred to as adding *dropout*.

For small dropout probabilities ($p \leq 0.1$), the algorithm often obtains an increase in performance.

In table 4.6, we see how adding dropout affects the performance of particular algorithm on GSet instances. The experiments were performed in a similar fashion as described in section 4.2. Dropout probabilities were selected from $p \in \{1\%, 5\%, 10\%, 20\%\}$. For each of these dropout probability, a parameter fine-tuning procedure is performed as described in 4.2. After parameter fine-tuning, a full experiment was run consisting of 50 tries with different initial conditions, and the obtained cut values are recorder. Mean and maximum of these recorded values is taken, for each dropout probability. The maximum among these 4 mean values is represented in the table under the column '$p > 0$ mean'. The maximum among these 4 max values is represented in the table under the column '$p > 0$ max'. Columns 'mean diff' and 'max diff' represent the difference between 'mean' and 'max' columns in this table and the corresponding columns in tables 4.1, 4.2, 4.3, and 4.4, where there is no dropout — denoted as $p = 0$. Thus, the 'diff' columns represent the difference $(p > 0) - (p = 0)$ for mean and max columns. So, if this difference is positive, it means that applying dropout increased the performance. Otherwise, adding dropout did not impact the performance, or worsened it.

In order to remove the possible advantage of dropout results due to larger number of total tries (because 4 different dropout probabilities were tried), initial conditions are taken exactly the same as for the experiment with no dropout. Thus, 50 random initial conditions

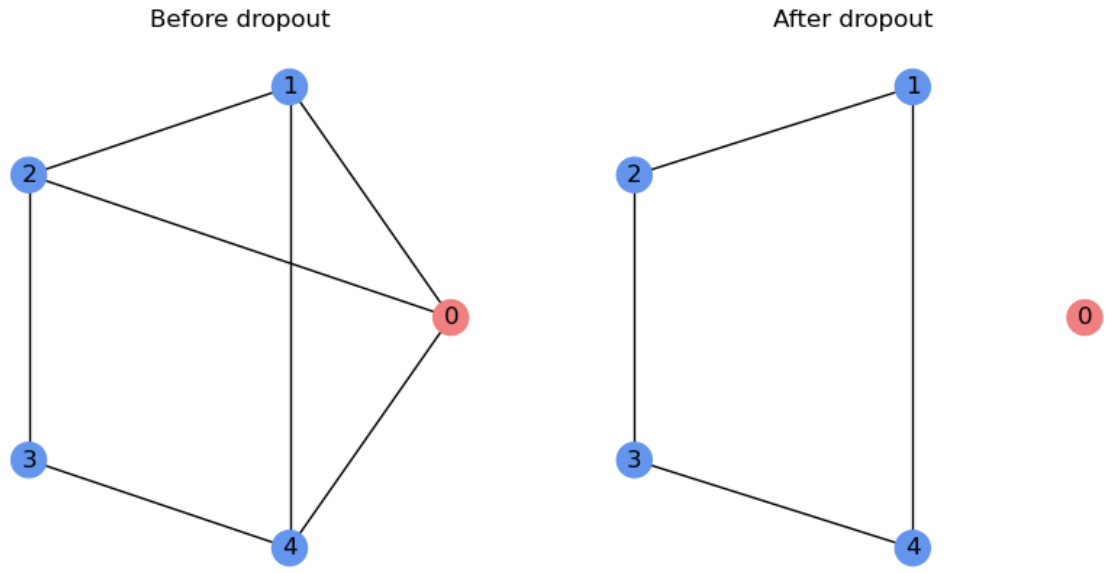


Figure 4.5: Dropout procedure

On the left we see a graph without applied dropout. Only the vertex 0 is chosen to be dropped out. The effect of dropout is that all edges connected to this vertex are removed, and it can be seen on the right.

were sampled already for experiment with no dropout, and exactly these initial conditions were used as 50 tries of each experiment with added dropout. The same was done during fine-tuning, but with 5 random initial conditions.

| General info | | CIM | | | | SB | | | | SimCIM | | | | bSB | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| instance | max known cut | $p > 0$ mean | mean diff | $p > 0$ max | max diff | $p > 0$ mean | mean diff | $p > 0$ max | max diff | $p > 0$ mean | mean diff | $p > 0$ max | max diff | $p > 0$ mean | mean diff | $p > 0$ max | max diff |
| G1 | 11624 | 11571.8 | 10.6 | 11615 | 21 | 11546.1 | -36.1 | 11586 | -29 | 11605.9 | -2.9 | 11624 | 1 | 11608.3 | -0.2 | 11624 | 1 |
| G2 | 11620 | 11567.1 | 1.2 | 11596 | 6 | 11552.2 | -27.8 | 11577 | -22 | 11600.5 | 5.5 | 11616 | 7 | 11601.1 | 5.5 | 11615 | 3 |
| G3 | 11622 | 11563.8 | -3 | 11594 | -12 | 11551 | -38.7 | 11575 | -42 | 11614.4 | -4.3 | 11622 | 0 | 11612.6 | -6.5 | 11622 | 0 |
| G4 | 11646 | 11611.2 | 2.8 | 11628 | 1 | 11593.7 | -20.8 | 11618 | -14 | 11634.6 | -2.2 | 11646 | 4 | 11637.2 | -0.4 | 11646 | 8 |
| G5 | 11631 | 11592.7 | -1.3 | 11610 | 8 | 11582.9 | -24.3 | 11608 | -10 | 11622.1 | -2.1 | 11627 | 2 | 11622.4 | -2.6 | 11631 | 6 |
| G6 | 2178 | 2141.5 | -1.2 | 2155 | 8 | 2125 | -33.7 | 2157 | -11 | 2169.6 | -3.1 | 2175 | 1 | 2172.5 | 1.2 | 2177 | 3 |
| G7 | 2006 | 1957.6 | 2.6 | 1977 | 1 | 1947.5 | -30.6 | 1969 | -29 | 1987.8 | 0.9 | 2005 | 3 | 1990.7 | 5.5 | 2006 | 8 |
| G8 | 2005 | 1949.1 | -1.3 | 1976 | 3 | 1946.8 | -31 | 1973 | -17 | 1989.5 | 2.8 | 2005 | 0 | 1992 | 6.2 | 2005 | 1 |
| G9 | 2054 | 2006.6 | 15.7 | 2026 | -6 | 1997 | -23.7 | 2024 | -14 | 2038.2 | 2.2 | 2048 | 2 | 2039.2 | 7.3 | 2051 | 5 |
| G10 | 2000 | 1954 | 6.6 | 1980 | -2 | 1948.2 | -20.4 | 1975 | -11 | 1979.4 | 3.8 | 1995 | 3 | 1983.1 | 0.1 | 1996 | 5 |
| G11 | 564 | 549.6 | -0.6 | 558 | 4 | 549.6 | -4 | 558 | 0 | 551.4 | 1.8 | 558 | 0 | 551.6 | -1.1 | 556 | 0 |
| G12 | 556 | 543.4 | 0.3 | 552 | 2 | 542.8 | -7 | 554 | 0 | 545.5 | 1 | 552 | 2 | 545.8 | 0.3 | 552 | 2 |
| G13 | 582 | 566.7 | -0.8 | 576 | 2 | 567.5 | -3.7 | 576 | -2 | 570.1 | 0.9 | 576 | 0 | 569.2 | 0.1 | 576 | -2 |
| G14 | 3064 | 2990.3 | 4.1 | 3008 | -2 | 2995.9 | -20.6 | 3016 | -17 | 3033.8 | 0.2 | 3048 | 14 | 3034.9 | 6.3 | 3049 | 11 |
| G15 | 3050 | 2979.7 | 4.3 | 2992 | 7 | 2984.4 | -11.6 | 2997 | -14 | 3021.9 | -0.1 | 3034 | 12 | 3021.1 | -0.2 | 3037 | 1 |
| G16 | 3052 | 2978.5 | 4.3 | 2999 | 15 | 2987.7 | -13 | 2999 | -19 | 3027.9 | 0.6 | 3042 | 13 | 3022.7 | -2.5 | 3040 | 2 |
| G17 | 3047 | 2978.4 | 3.1 | 2999 | 11 | 2985.3 | -8.5 | 3003 | -4 | 3019.7 | -1.3 | 3038 | 17 | 3018.7 | 1.7 | 3035 | 2 |
| G18 | 992 | 935.7 | 11.1 | 946 | 1 | 937.9 | -8.5 | 953 | -19 | 965.1 | 0.5 | 982 | 17 | 967.7 | 4 | 986 | 10 |
| G19 | 906 | 854.2 | 1.1 | 860 | 6 | 849.9 | 31.8 | 864 | -7 | 876.8 | -1.6 | 892 | 12 | 877.8 | 3.3 | 899 | 20 |
| G20 | 941 | 889.2 | -0.4 | 902 | 11 | 889.5 | -10 | 908 | -9 | 927.7 | 1.6 | 932 | 1 | 919.3 | 6.8 | 938 | 8 |
| G21 | 931 | 878.1 | 27.5 | 896 | 9 | 883.9 | -9.1 | 904 | 5 | 909.7 | -3.9 | 917 | 0 | 909.8 | 3.6 | 917 | 0 |

Table 4.6: Dropout results

The results show that for CIM, SimCIM, and bSB algorithms, adding some amount of dropout often increases the performance by some small extent. It rarely decreases the performance, and most importantly, sometimes increases the performance significantly.

To further investigate whether adding dropout makes any significant difference, the following was done. GSet instance G18 was picked, and bSB algorithm was run 10.000 times with various parameters, picked randomly from some sensible search space, and with various random initial conditions, but with no dropout. The best cut value found was 982. On the other hand, doing the same experiment with 1.000 tries, and while adding dropout (this time with decreasing schedule of dropout probability), a cut with value 989 was found, which is very close to the maximum known cut 992. Thus, it can be concluded that likely it is the dropout which improved the performance of bSB algorithm on this instance.

Besides these improvements, it can be seen that adding dropout to SB algorithm makes the opposite effect — it reduces the performance significantly. It is not clear why only the SB algorithm suffers a performance loss, while CIM, SimCIM, and bSB obtains improvement. It was observed, however, that fine-tuned parameters for CIM often had lower rates of momentum when $p > 0$ compared to the case when $p = 0$. This is consistent with the results, since SB can be seen as having fixed momentum equal to 1 3.6.

While adding dropout by removing vertices increases the performance, it has been observed that adding dropout in such a way as to remove random edges (each edge sampled independently), does not seem to improve the performance.

The question of how does adding dropout, in a way that vertices are randomly removed, actually contribute to the performance gain — is still unanswered. This question is out of the scope of this thesis, and would require further testing and analysis.

# Bibliography


[1] *Adiabatic theorem of quantum mechanics*, https://en.wikipedia.org/wiki/Adiabatic_theorem.

[2] *Airy function*, https://en.wikipedia.org/wiki/Airy_function.

[3] *Apx*, https://en.wikipedia.org/wiki/APX.

[4] *Gset dataset*, https://web.stanford.edu/~yyye/yyye/Gset/.

[5] *Maximum cut*, https://en.wikipedia.org/wiki/Maximum_cut.

[6] *Np-hardness*, https://en.wikipedia.org/wiki/NP-hardness.

[7] *Ptas*, https://en.wikipedia.org/wiki/Polynomial-time_approximation_scheme#cite_note-1.

[8] *Quadratic unconstrained binary optimization, wiki*, https://en.wikipedia.org/wiki/Quadratic_unconstrained_binary_optimization.

[9] *Semi-implicit euler method*, https://en.wikipedia.org/wiki/Semi-implicit_Euler_method#cite_note-hairer2003-1, Accessed: 2023-11-07.

[10] *Unique games conjecture*, https://en.wikipedia.org/wiki/Unique_games_conjecture.

[11] *Universe age*, https://en.wikipedia.org/wiki/Age_of_the_universe.

[12] *Weyl's inequality*, https://en.wikipedia.org/wiki/Weyl%27s_inequality.

[13] Talal M. Alkhamis, Merza Hasan, and Mohamed A. Ahmed, *Simulated annealing for the unconstrained quadratic pseudo-boolean function*.

[14] Francisco Barahona, Martin Grötschel, and Michael Jünger, *An application of combinatorial optimization to statistical physics and circuit layout design*.

[15] J. E. Beasley, *Heuristic algorithms for the unconstrained binary quadratic programming problem*.

[16] Una Benlic and Jin Kao Hao, *Breakout local search for the max-cut problem*.

[17] J. C. Butcher, *Numerical methods for ordinary differential equations*, Wiley, 2016.

[18] Carmen Chicone, *Ordinary differential equations with applications*.

[19] Hayato Goto et al, *High-performance combinatorial optimization based on classical mechanics*.

[20] Fred Glover, Bahram Alidaee, and Gary Kochenberger, *Adaptive memory tabu search for binary quadratic programs*.

[21] Michael X. Goemans and David P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*.

[22] Hayato Goto, *Bifurcation-based adiabatic quantum computation with a onlinear oscillator network*.

[23] Hayato Goto, Kosuke Tatsumura, and Alexander R. Dixon, *Combinatorial optimization by simulating adiabatic bifurcations in nonlinear hamiltonian systems*.

[24] Peter L. Hammer and Eliezer Shlifer, *Applications of pseudo-boolean methods to economic problems*.

[25] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors*.

[26] Kayo Ide and Stephen Wiggins, *Distinguished hyperbolic trajectories in time-dependent fluid flows: Analytical and computational approach for velocity fields defined as data sets*.

[27] Taro Kanao and Hayato Goto, *Simulated bifurcation assisted by thermal fluctuation*.

[28] Gary Kochenberger, Jin Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang, *The unconstrained binary quadratic programming problem: a survey*.

[29] Jakob Krarup and Peter Mark Pruzan, *Computer-aided layout design*.

[30] D. J. Laughhunn, *Quadratic binary programming with application to capital-budgeting problems*.

[31] Benedict Leimkuhler and Sebastian Reich, *Simulating hamiltonian dynamics*, Cambridge University Press, 2004.

[32] Timothée Leleu, Yoshihisa Yamamoto, Shoko Utsunomiya, and Kazuyuki Aihara, *Combinatorial optimization using dynamical phase transitions in driven-dissipative systems*.

[33] Andrew Lucas, *Ising formulations of many np problems*.

[34] Peter Merz and Bernd Freisleben, *Genetic algorithms for binary quadratic programming*.

[35] Ning Qian, *On the momentum term in gradient descent learning algorithms*.

[36] M. R. Rao, *Cluster analysis and mathematical programming*.

[37] Egor S. Tiunov, Alexander E. Ulanov, and A. I. Lvovsky, *Annealing by simulating the coherent ising machine*.

[38] Jiahai Wang, Ying Zhou, and Jian Yin, *Combining tabu hopfield network and estimation of distribution for unconstrained binary quadratic programming problem*.

[39] Juntao Wang, Daniel Ebler, K. Y. Michael Wong, Davif Shui Wing Hui, and Jie Sun, *Bifurcation behaviors shape how continuous physical dynamics solves discrete ising optimization*.

[40] Zhe Wang, Alireza Marandi, Kai Wen, Robert L. Byer, and Yoshihisa Yamamoto, *A coherent ising machine based on degenerate optical parametric oscillators*.

[41] Stephen J. Wright and Jorge Nocedal, *Numerical optimization*.

# Sažetak

Ukratko ...

# Summary

In this ...

# Životopis

Dana ...