# Mapping MAX-2-SAT to Ising model

Marcin Wierzbiński [1 2]

[1]University of Warsaw, Faculty of Mathematics, Informatics and Mechanics
[2]AstroCENT, Nicolaus Copernicus Astronomical Center, Polish Academy of Sciences

December 15, 2020

ASTROCENT
European Funds Smart Growth
Republic of Poland
FNP Foundation for Polish Science
European Union European Regional Development Fund
CAMK PAN

# Acknowledgments


- prof. Henryk Michalewski (University of Warsaw, Google)
- prof. Piotr Gawron (AstroCENT, Nicolaus Copernicus Astronomical Center, Polish Academy of Sciences)
- prof. Karol Życzkowski (UJ Atomic Physics Department)
- prof. Alexandra Lvovskiego (University of Oxford and Russian Quantum Center).

# Introduction

- Motivation

# Introduction

- Motivation
- Showing that 2-SAT problem is solvable in polynomial time.

# Introduction

- Motivation
- Showing that 2-SAT problem is solvable in polynomial time.
- A short introduction to tensor product space and Ising model.

# Introduction

- Motivation
- Showing that 2-SAT problem is solvable in polynomial time.
- A short introduction to tensor product space and Ising model.
- Reduction of MAX-2-SAT to the Ising model.

# Introduction

- Motivation
- Showing that 2-SAT problem is solvable in polynomial time.
- A short introduction to tensor product space and Ising model.
- Reduction of MAX-2-SAT to the Ising model.
- Results.

# Motivation

- Ising formulations of many NP-Hard problems [1]

[1] Frontiers in Physics http://dx.doi.org/10.3389/fphy.2014.00005

# Motivation

- Ising formulations of many NP-Hard problems [1]
- The coefficients of Ising model will define our calculation problem

[1] Frontiers in Physics `http://dx.doi.org/10.3389/fphy.2014.00005`

# Motivation

- Ising formulations of many NP-Hard problems [1]
- The coefficients of Ising model will define our calculation problem
- SAT – Automated theorem proving, CSP(Constraint satisfiability problem)

[1] Frontiers in Physics http://dx.doi.org/10.3389/fphy.2014.00005

# Motivation

- Ising formulations of many NP-Hard problems [1]
- The coefficients of Ising model will define our calculation problem
- SAT – Automated theorem proving, CSP(Constraint satisfiability problem)
- 3-SAT – A very important problem in the world of in computational complexity theory

[1] Frontiers in Physics http://dx.doi.org/10.3389/fphy.2014.00005

# NP-Hard problems

- The class of problems for which there is an algorithm for solving them in polynomial time is denoted by **P**.

[2] We don't know about NP $\subset P$.

# NP-Hard problems

- The class of problems for which there is an algorithm for solving them in polynomial time is denoted by **P**.
- There are problems that are not known if they belong to this class, but you can check the correctness of the solution in polynomial time.

[2] We don't know about NP $\subset P$.

# NP-Hard problems

- The class of problems for which there is an algorithm for solving them in polynomial time is denoted by **P**.
- There are problems that are not known if they belong to this class, but you can check the correctness of the solution in polynomial time.
- We say such problems belong to **class NP**.

[2] We don't know about NP $\subset P$.

# NP-Hard problems

- The class of problems for which there is an algorithm for solving them in polynomial time is denoted by **P**.
- There are problems that are not known if they belong to this class, but you can check the correctness of the solution in polynomial time.
- We say such problems belong to **class NP**.
- $P \subset NP$[2]. We can check the correctness of the solution in polynomial time, simply by solving it ourselves.

[2] We don't know about NP $\subset P$.

# NP-Hard problems

- The class of problems for which there is an algorithm for solving them in polynomial time is denoted by **P**.
- There are problems that are not known if they belong to this class, but you can check the correctness of the solution in polynomial time.
- We say such problems belong to **class NP**.
- $P \subset NP$[2]. We can check the correctness of the solution in polynomial time, simply by solving it ourselves.
- A problem $H$ is **NP-Hard** when each $L \in NP$ can be reduced in polynomial time to $H$;

[2] We don't know about NP $\subset P$.

# 2-SAT Problem

$$F = (x_0 \vee x_1) \wedge (x_0 \vee \neg x_1) \wedge (\neg x_0 \vee x_1)$$

# 2-SAT Problem

$$F = (x_0 \vee x_1) \wedge (x_0 \vee \neg x_1) \wedge (\neg x_0 \vee x_1) = \Omega_1 \vee \Omega_2 \vee \Omega_3 \quad (1)$$

For a given Boolean formula $F$ in the **2-CNF** form, determine whether there is a boolean variable assignment such that formula is TRUE. Otherwise, we say that $F$ is unsatisfiable.

# 2-SAT Problem

$$F = (x_0 \vee x_1) \wedge (x_0 \vee \neg x_1) \wedge (\neg x_0 \vee x_1) = \Omega_1 \vee \Omega_2 \vee \Omega_3 \quad (1)$$

For a given Boolean formula $F$ in the **2-CNF** form, determine whether there is a boolean variable assignment such that formula is TRUE. Otherwise, we say that $F$ is unsatisfiable.

| $x_0$ | $x_1$ | $(x_0 \vee x_1) \wedge (x_0 \vee \neg x_1) \wedge (\neg x_0 \vee x_1)$ |
|---|---|---|
| F | F | F |
| F | T | F |
| T | F | F |
| T | T | T |

# 2-SAT Problem

$$F = (x_0 \vee x_1) \wedge (x_0 \vee \neg x_1) \wedge (\neg x_0 \vee x_1) = \Omega_1 \vee \Omega_2 \vee \Omega_3 \quad (1)$$

For a given Boolean formula $F$ in the **2-CNF** form, determine whether there is a boolean variable assignment such that formula is TRUE. Otherwise, we say that $F$ is unsatisfiable.

| $x_0$ | $x_1$ | $(x_0 \vee x_1) \wedge (x_0 \vee \neg x_1) \wedge (\neg x_0 \vee x_1)$ |
|---|---|---|
| F | F | F |
| F | T | F |
| T | F | F |
| T | T | T |

When the number of Boolean variable is $n$ then number of different assignments are $2^n$

# 2-SAT Problem

$$F = (x_0 \vee x_1) \wedge (x_0 \vee \neg x_1) \wedge (\neg x_0 \vee x_1) = \Omega_1 \vee \Omega_2 \vee \Omega_3 \quad (1)$$

For a given Boolean formula $F$ in the **2-CNF** form, determine whether there is a boolean variable assignment such that formula is TRUE. Otherwise, we say that $F$ is unsatisfiable.

| $x_0$ | $x_1$ | $(x_0 \vee x_1) \wedge (x_0 \vee \neg x_1) \wedge (\neg x_0 \vee x_1)$ |
|---|---|---|
| F | F | F |
| F | T | F |
| T | F | F |
| T | T | T |

When the number of Boolean variable is $n$ then number of different assignments are $2^n$ Important: $\Omega_i$ we denote as clause.

# MAX-2-SAT optimisation problem

### Theorem

MAX-2-SAT is NP-Complete.

# MAX-2-SAT optimisation problem

## Theorem

MAX-2-SAT is NP-Complete.

NP-completeness reduction: Any 3-SAT problem can be reduced to MAX-2-SAT

We transform 3-SAT instance to 2-CNF form by replacing each clause $\Omega_i = (x_1 \vee x_2 \vee x_3)$

# MAX-2-SAT optimisation problem

## Theorem

MAX-2-SAT is NP-Complete.

NP-completeness reduction: Any 3-SAT problem can be reduced to MAX-2-SAT

We transform 3-SAT instance to 2-CNF form by replacing each clause $\Omega_i = (x_1 \vee x_2 \vee x_3)$

$$F = (x_1 \vee x_1) \wedge (x_2 \vee x_2) \wedge (x_3 \vee x_3) \wedge (\Omega_i \vee \Omega_i)$$
$$\wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_1 \vee \neg \Omega_i)$$
$$\wedge (x_3 \vee \neg \Omega_i)$$

# MAX-2-SAT optimisation problem

## Theorem

MAX-2-SAT is NP-Complete.

NP-completeness reduction: Any 3-SAT problem can be reduced to MAX-2-SAT

We transform 3-SAT instance to 2-CNF form by replacing each clause $\Omega_i = (x_1 \vee x_2 \vee x_3)$

$$F = (x_1 \vee x_1) \wedge (x_2 \vee x_2) \wedge (x_3 \vee x_3) \wedge (\Omega_i \vee \Omega_i) \\ \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_1 \vee \neg \Omega_i) \\ \wedge (x_3 \vee \neg \Omega_i)$$

- If an assignment satisfies $\Omega_i$, then exactly 7 of 10 clause in $F$ are satisfiable.

# MAX-2-SAT optimisation problem

## Theorem

MAX-2-SAT is NP-Complete.

NP-completeness reduction: Any 3-SAT problem can be reduced to MAX-2-SAT

We transform 3-SAT instance to 2-CNF form by replacing each clause $\Omega_i = (x_1 \vee x_2 \vee x_3)$

$$F = (x_1 \vee x_1) \wedge (x_2 \vee x_2) \wedge (x_3 \vee x_3) \wedge (\Omega_i \vee \Omega_i) \\ \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_1 \vee \neg \Omega_i) \\ \wedge (x_3 \vee \neg \Omega_i)$$

- If an assignment satisfies $\Omega_i$, then exactly 7 of 10 clause in $F$ are satisfiable.
- If an assignment does not satisfies $\Omega_i$, then exactly 6 of the 10 clause in $F$ are satisfiable.

# What we know about 2-SAT?

# What we know about 2-SAT?

**Theorem**

The 2-SAT is solvable in polynomial time.

# Intuition of reduction

# Intuition of reduction

Create directed graph $G = (V, E)$, with $2n$ vertex.

# Intuition of reduction

Create directed graph $G = (V, E)$, with $2n$ vertex. Each clause we reduce as:

$$(x_i \vee x_j) \iff (\neg x_i \Rightarrow x_j) \wedge (\neg x_j \Rightarrow x_i)$$

# Intuition of reduction

Create directed graph $G = (V, E)$, with $2n$ vertex. Each clause we reduce as:

$$(x_i \vee x_j) \iff (\neg x_i \Rightarrow x_j) \wedge (\neg x_j \Rightarrow x_i)$$

Variable and their negations correspond to the vertex of this graph and the edge pointing with $x_i \rightarrow x_j$ is added if and only if the implication $x_i \Rightarrow x_j$ belongs into family $\mathcal{F}$.

# Intuition of reduction

Create directed graph $G = (V, E)$, with $2n$ vertex. Each clause we reduce as:

$$(x_i \vee x_j) \iff (\neg x_i \Rightarrow x_j) \wedge (\neg x_j \Rightarrow x_i)$$

Variable and their negations correspond to the vertex of this graph and the edge pointing with $x_i \to x_j$ is added if and only if the implication $x_i \Rightarrow x_j$ belongs into family $\mathcal{F}$.

# Intuition of reduction

Create directed graph $G = (V, E)$, with $2n$ vertex. Each clause we reduce as:

$$(x_i \vee x_j) \iff (\neg x_i \Rightarrow x_j) \wedge (\neg x_j \Rightarrow x_i)$$

Variable and their negations correspond to the vertex of this graph and the edge pointing with $x_i \rightarrow x_j$ is added if and only if the implication $x_i \Rightarrow x_j$ belongs into family $\mathcal{F}$.

Example: $F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_3)$

# Intuition of reduction

Create directed graph $G = (V, E)$, with $2n$ vertex. Each clause we reduce as:

$$(x_i \vee x_j) \iff (\neg x_i \Rightarrow x_j) \wedge (\neg x_j \Rightarrow x_i)$$

Variable and their negations correspond to the vertex of this graph and the edge pointing with $x_i \rightarrow x_j$ is added if and only if the implication $x_i \Rightarrow x_j$ belongs into family $\mathcal{F}$.

Example: $F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_3)$

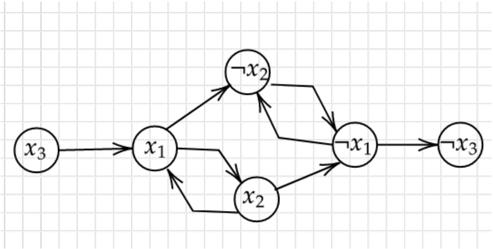



Figure: Reduction graph for $F$

# Strongly connected component

## Definition

Strongly connected component of the directed graph $G = (V, E)$ we call a subset $A \subset V$ such that $\forall_{i \neq j} \quad _{x_i, x_j \in A}$ there is a path between $x_i$ and $x_j$.

# Strongly connected component

## Definition

Strongly connected component of the directed graph $G = (V, E)$ we call a subset $A \subset V$ such that $\forall_{i \neq j} \quad {}_{x_i, x_j \in A}$ there is a path between $x_i$ and $x_j$.

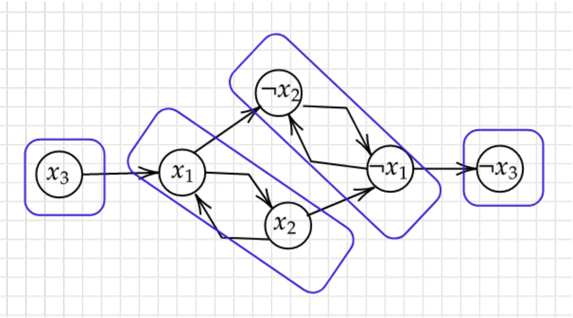



Figure: Strongly connected component

# Draft of proof for 2-SAT- lemma

## Lemma

If both vertices $x_i$ and $\neg x_i$ are in a strong connected component then the formula is unsatisfiable.

## Proof.

Recall that if $x_i$ and $\neg x_i$ are in a strong connected component then there is a pathway between $x_i \rightsquigarrow \neg x_i$ and $\neg x_i \rightsquigarrow x_i$.

# Draft of proof for 2-SAT- lemma

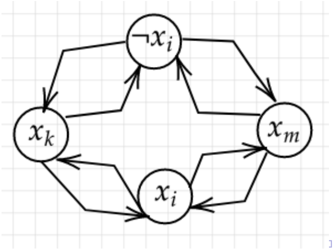
## Lemma

If both vertices $x_i$ and $\neg x_i$ are in a strong connected component then the formula is unsatisfiable.

## Proof.

Recall that if $x_i$ and $\neg x_i$ are in a strong connected component then there is a pathway between $x_i \rightsquigarrow \neg x_i$ and $\neg x_i \rightsquigarrow x_i$. From transitive property of implication:

$$[(x_i \Rightarrow x_m) \wedge (x_m \Rightarrow \neg x_i)] \Rightarrow (x_i \Rightarrow \neg x_i)$$



# Phase transition

### Definition

The clause density for the CNF formula: 2-CNF is defined as ratio:

$$\alpha = \frac{M}{n}, \tag{2}$$

where $M$ number of clauses and $n$ number of variables in $F$

- $F = (x_1 \vee x_2) \wedge (x_3 \vee x_2)$

# Phase transition

## Definition

The clause density for the CNF formula: 2-CNF is defined as ratio:

$$\alpha = \frac{M}{n}, \tag{2}$$

where $M$ number of clauses and $n$ number of variables in $F$

- $F = (x_1 \vee x_2) \wedge (x_3 \vee x_2)$
- For $\alpha = \frac{2}{3}$ and 5 of the interpretations satisfible $F$

# Phase transition

## Definition

The clause density for the CNF formula: 2-CNF is defined as ratio:

$$\alpha = \frac{M}{n}, \tag{2}$$

where $M$ number of clauses and $n$ number of variables in $F$

- $F = (x_1 \vee x_2) \wedge (x_3 \vee x_2)$
- For $\alpha = \frac{2}{3}$ and 5 of the interpretations satisfible $F$
- $F = (x_1 \vee x_2) \wedge (x_3 \vee x_2) \wedge (x_3 \vee x_1) \wedge (\neg x_3 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$

# Phase transition

## Definition

The clause density for the CNF formula: 2-CNF is defined as ratio:

$$\alpha = \frac{M}{n}, \tag{2}$$

where $M$ number of clauses and $n$ number of variables in $F$

- $F = (x_1 \vee x_2) \wedge (x_3 \vee x_2)$
- For $\alpha = \frac{2}{3}$ and 5 of the interpretations satisfible $F$
- $F = (x_1 \vee x_2) \wedge (x_3 \vee x_2) \wedge (x_3 \vee x_1) \wedge (\neg x_3 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$
- For $\alpha = \frac{5}{3}$ only one of the interpretation satisfiable $F$.

# Phase transition for 2-SAT and 3-SAT

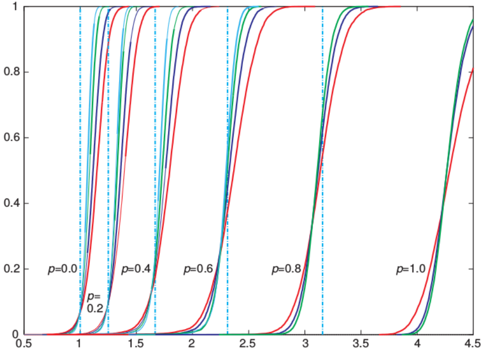



Figure: Phase transition for 2-SAT$(p = 0)$ from work [3] on the Y axis percentage of unfulfilled formula (UNSAT), on the X axis the clause density

[3] Determining computational complexity from characteristic 'phase transitions' Monasson, Rémi and Zecchina, Riccardo and Kirkpatrick, Scott and Selman, Bart and Troyansky, Lidror

# Tensor product of two vector space

## Definition

The tensor product $V \otimes W$ of two vector spaces $V$ and $W$ over the same field is a vector spaces, endowed with a bilinear mapping

$$(v, w) : V \times W :\mapsto v \otimes w \in V \otimes W$$

[4] $\text{v} \cdot w^T$

# Tensor product of two vector space

## Definition

The tensor product $V \otimes W$ of two vector spaces $V$ and $W$ over the same field is a vector spaces, endowed with a bilinear mapping

$$(v, w) : V \times W :\mapsto v \otimes w \in V \otimes W$$

If $\{v_s : s \in S\}$ and $\{w_t, t \in T\}$ are bases of $V$ and $W$, and $dim(V) = m$ and $dim(W) = n$ then $mn$ elements form a basic of $V \otimes W$. If $v \in V$ and $w \in W$ then coordinate vector of $v \otimes w$ over this basis is the outer product[4] of the coordinate vectors of v and w over the corresponding bases.

---

[4] $\mathrm{v} \cdot w^T$

# The Pauli matrixes

## Definition

The Pauli matrixes are set of three $2 \times 2$ complex matrices which are Hermitian and ununitary, with eigenvalues $+1, -1$. They are

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Identity matrix is given as:

$$\mathbb{1}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

# The Kronecker product

## Definition

The product of Kronecker's two matrixes
$A \in M_{n\times m}(\mathbb{C}), B \in M_{p\times q}(\mathbb{C})$ is the block matrix
$A \otimes B \in M_{np\times mq}(\mathbb{C})$ defined as:

# The Kronecker product

## Definition

The product of Kronecker's two matrixes $A \in M_{n\times m}(\mathbb{C}), B \in M_{p\times q}(\mathbb{C})$ is the block matrix $A \otimes B \in M_{np\times mq}(\mathbb{C})$ defined as:

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$$

# The Kronecker product

## Definition

The product of Kronecker's two matrixes $A \in M_{n \times m}(\mathbb{C}), B \in M_{p \times q}(\mathbb{C})$ is the block matrix $A \otimes B \in M_{np \times mq}(\mathbb{C})$ defined as:

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$$

Example:

# The Kronecker product

## Definition

The product of Kronecker's two matrixes $A \in M_{n\times m}(\mathbb{C}), B \in M_{p\times q}(\mathbb{C})$ is the block matrix $A \otimes B \in M_{np\times mq}(\mathbb{C})$ defined as:

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$$

Example:

$$\mathbb{1}_2 \otimes \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

# The Ising Model – unitary matrices

### Definition

The Ising Model is a weighted graph, whose vertex are qubits. The weights of the vertices are indicated as $h_i$ for $i = 1, \ldots, |V|$, and weights of the edges are determined by $J_{i,j}$.

# The Ising Model – unitary matrices

## Definition

The Ising Model is a weighted graph, whose vertex are qubits. The weights of the vertices are indicated as $h_i$ for $i = 1, \ldots, |V|$, and weights of the edges are determined by $J_{i,j}$.

## Definition

The Hamiltonian of the Ising model in the calculation base is defined as a matrix:

# The Ising Model – unitary matrices

## Definition

The Ising Model is a weighted graph, whose vertex are qubits. The weights of the vertices are indicated as $h_i$ for $i = 1, \ldots, |V|$, and weights of the edges are determined by $J_{i,j}$.

## Definition

The Hamiltonian of the Ising model in the calculation base is defined as a matrix:

$$H_p = \sum_{\{ij\} \in E} J_{i,j} \sigma_i^{(z)} \sigma_j^{(z)} + \sum_{j \in V} h_j \sigma_j^{(z)} \tag{3}$$

# With simple agreement:

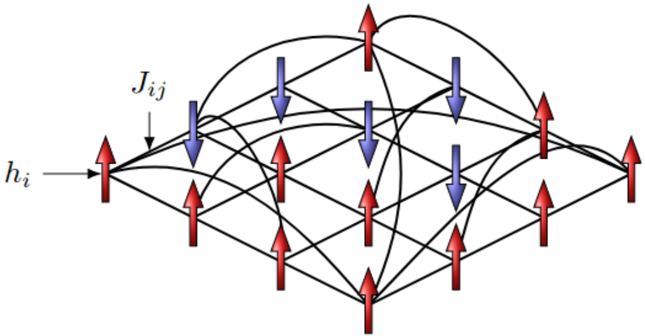
$$\sigma_i^{(z)} = \mathbb{1}_2^{\otimes(i-1)} \otimes \sigma_i^{(z)} \otimes \mathbb{1}_2^{\otimes(n-i-1)} \tag{4}$$

# With simple agreement:

$$\sigma_i^{(z)} = \mathbb{1}_2{}^{\otimes(i-1)} \otimes \sigma_i^{(z)} \otimes \mathbb{1}_2{}^{\otimes(n-i-1)} \tag{4}$$

$$\sigma_i^{(z)} \sigma_j^{(z)} = \mathbb{1}_2{}^{\otimes(j-1)} \otimes \sigma_i^{(z)} \otimes \mathbb{1}_2{}^{\otimes(j-i-1)} \otimes \sigma_j^{(z)} \otimes \mathbb{1}_2{}^{\otimes(n-i-1)} \tag{5}$$

# Ising Model – Intuition



## Definition

Eigenfunction: $\mathrm{H}_p |\psi\rangle = E_i |\psi\rangle$

[5] We often talk about the computational basis

## Definition

Eigenfunction: $\mathrm{H}_p |\psi\rangle = E_i |\psi\rangle$

## Remark

$H_p$ is a Hermitian operator. $[h_{ij}] = [\overline{h_{ji}}]$.

[5] We often talk about the computational basis

## Definition

Eigenfunction: $\mathrm{H}_p \left|\psi\right\rangle = E_i \left|\psi\right\rangle$

## Remark

$H_p$ is a Hermitian operator. $[h_{ij}] = [\overline{h_{ji}}]$.

## Proof.

It follows from the definition of Kroneker product for matrix: $\sigma_i^{(z)}$ $\square$

[5] We often talk about the computational basis

### Definition

Eigenfunction: $\mathrm{H}_p \left|\psi\right\rangle = E_i \left|\psi\right\rangle$

### Remark

$H_p$ is a Hermitian operator. $[h_{ij}] = [\overline{h_{ji}}]$.

### Proof.

It follows from the definition of Kroneker product for matrix: $\sigma_i^{(z)}$ □

### Remark

Ground state of $H_p$ is the vector of the form $|x_1\rangle \otimes \ldots \otimes |x_n\rangle$[5] for some $\{x_i\}_{i=1}^n$, where $x_i \in \{0, 1\}$.

[5] We often talk about the computational basis

# The Ising model solution:

**Remark**

$H_p$ is hermitian operator, so it has real values $(E_0 \leqslant E_1 \ldots \leqslant E_{2^n})$.
Ground state is its lowest-energy state $E_0$.

# The Ising model solution:

### Remark

$H_p$ is hermitian operator, so it has real values $(E_0 \leqslant E_1 \ldots \leqslant E_{2^n})$. Ground state is its lowest-energy state $E_0$.

### Definition

Ising's computational problem is to find the ground state $|\psi\rangle$ of Eigenfunction $H_p$ (such as in 10).

# Reduction - Step 1

The first step is transforming from Boolean $\mathbb{B} = \{T, F\}$ to binary variables $x_i = \{0, 1\}$, letting $TRUE \mapsto 0$ and $FALSE \mapsto 1$.

# Reduction - Step 1

The first step is transforming from Boolean $\mathbb{B} = \{T, F\}$ to binary variables $x_i = \{0, 1\}$, letting $TRUE \mapsto 0$ and $FALSE \mapsto 1$. The truth table of $OR$ becomes as multiplication of the binary variables:

# Reduction - Step 1

The first step is transforming from Boolean $\mathbb{B} = \{T, F\}$ to binary variables $x_i = \{0, 1\}$, letting $TRUE \mapsto 0$ and $FALSE \mapsto 1$. The truth table of $OR$ becomes as multiplication of the binary variables:

$$\begin{array}{|c|c|c|}\hline p & q & p \vee q \\ \hline T & F & T \\ \hline F & T & T \\ \hline T & T & T \\ \hline F & F & F \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|}\hline x_i \cdot x_j & x_j & x_i \\ \hline 0 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \tag{6}$$

# Reduction - Step 1

The first step is transforming from Boolean $\mathbb{B} = \{T, F\}$ to binary variables $x_i = \{0, 1\}$, letting $TRUE \mapsto 0$ and $FALSE \mapsto 1$. The truth table of $OR$ becomes as multiplication of the binary variables:

$$\begin{array}{|c|c|c|}\hline p & q & p \vee q \\ \hline T & F & T \\ \hline F & T & T \\ \hline T & T & T \\ \hline F & F & F \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|}\hline x_i \cdot x_j & x_j & x_i \\ \hline 0 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \tag{6}$$

It also define variable:

# Reduction - Step 1

The first step is transforming from Boolean $\mathbb{B} = \{T, F\}$ to binary variables $x_i = \{0, 1\}$, letting *TRUE* $\mapsto 0$ and *FALSE* $\mapsto 1$. The truth table of *OR* becomes as multiplication of the binary variables:

| $p$ | $q$ | $p \vee q$ |
|---|---|---|
| $T$ | $F$ | $T$ |
| $F$ | $T$ | $T$ |
| $T$ | $T$ | $T$ |
| $F$ | $F$ | $F$ |

$\longrightarrow$

| $x_i \cdot x_j$ | $x_j$ | $x_i$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

(6)

It also define variable:

$$v_j^k = \begin{cases} -1 & \text{if } x_j \text{ appears negated in } k\text{th clause} \\ 1 & \text{if } x_j \text{ appears unnegated in } k\text{th clause} \\ 0 & \text{if } x_j \text{ does not appear in } k\text{th clause} \end{cases} \tag{7}$$

# Reduction- Step 2

## Definition

The local Hamiltonian for the $\Omega_k$ clause is defined as

$$H_{\Omega_k} = \frac{\mathbb{1} - v^k_{j_1}\sigma^z_{j_1}}{2}\frac{\mathbb{1} - v^k_{j_2}\sigma^z_{j_2}}{2} \tag{8}$$

# Reduction- Step 2

### Definition

The local Hamiltonian for the $\Omega_k$ clause is defined as

$$H_{\Omega_k} = \frac{\mathbb{1} - v_{j_1}^k \sigma_{j_1}^z}{2} \frac{\mathbb{1} - v_{j_2}^k \sigma_{j_2}^z}{2} \tag{8}$$

### Lemma

If exist an assignment $\{x_{j_1}, x_{j_2}\} \in \{0,1\}^2$, which violate the clause $\Omega_k$ then minimal energy of $H_{\Omega_k}$ is 1. If not exist an assignment $\{x_{j_1}, x_{j_2}\} \in \{0,1\}^2$, which satisfied clause $\Omega_k$, then energy of $H_{\Omega_k}$ is 0.

# Example

Example of local Hamiltonian for

$$F = \Omega_1 = (\neg x_1 \vee x_2)$$

then for such a formula reading from $v_j$ we obtain form:

$$H_{\Omega_1} = \frac{1}{4}[\mathbb{1} - (-1 \cdot \sigma_1^z \otimes \mathbb{1}_2)] \cdot [\mathbb{1} - (\mathbb{1}_2 \otimes \sigma_2^z)] =$$

$$\frac{1}{4} \cdot \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

# Example c.d

Let us note that $H_{\Omega_1}$ has energies $E_i$ and the corresponding states $|\psi_i\rangle$ :

$$E_1 = 1 \mapsto |\psi_1\rangle = |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad E_2 = 0 \mapsto |\psi_2\rangle = |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$E_3 = 0 \mapsto |\psi_3\rangle = |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad E_4 = 0 \mapsto |\psi_4\rangle = |00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Logical assigments which, which violates the formula is: $x_1 = 0$, $x_2 = 1$, remark(we write this as $TRUE \mapsto 0$ and $FALSE \mapsto 1$

# Reduction- Step 3

Taking into account each local Hamiltonian $H_\Omega$, the problem Hamiltonian is now constructed as

# Reduction- Step 3

Taking into account each local Hamiltonian $H_\Omega$, the problem Hamiltonian is now constructed as

$$H_{Pr} = \sum_{\Omega_k \in F} H_{\Omega_k}, \tag{9}$$

# Reduction- Step 3

Taking into account each local Hamiltonian $H_\Omega$, the problem Hamiltonian is now constructed as

$$H_{Pr} = \sum_{\Omega_k \in F} H_{\Omega_k}, \tag{9}$$

that is the sum of the energy after all $M$ clause contained in the 2-SAT.

# Reduction – Step 4

$$H_{Pr} = \sum_{\Omega_k \in F} H_{\Omega_k}, \tag{10}$$

Summing up after all clauses local Hamiltonians:

$$H_{Pr} = \frac{1}{4} \sum_{\Omega_k \in F} \mathbb{1} - v_{j_1}^k \sigma_{j_1}^z - v_{j_2}^k \sigma_{j_2}^z + v_{j_1}^k v_{j_2}^k \sigma_{j_1}^z \sigma_{j_2}^z \tag{11}$$

After rescaling by a factor of 4 and dropping the constant term we obtain:

$$h_{j_i} = -\sum_k v_{j_i}^k, \quad J_{j_1 j_2} = \sum_k v_{j_1}^k \cdot v_{j_2}^k, \tag{12}$$

# Reduction – algorithm

**procedure** InitIsing(F)

$M \leftarrow$ no. of clauses for formula$F$

$N \leftarrow$ no. of variables for formula $F$

$h \leftarrow$ create a zero vector size N

$J \leftarrow$ create zero marix size $N \times N$

$v \leftarrow$ create zero matrix size $M \times N$

**return** $N, m, J, h, v$

```
procedure 2CNFtoIsing(φ)
    for j = 1..M do
        get index of variables i_1, i_2
        var_1, var_2 ← +1, -1, 0 as in v_j^k 7
        v[j, i_1], v[j, i_2] ← var_1 ← var_2
    for j = 1..n do
        for i = 1..m do
            j_1 ← −1
            j_2 ← 0
            if v[j, i] <> 0 & j_1 == −1 then
                j_1 ← i + 1 continue
            if v[j, i] <> 0 & j_1 <> −1 then
                j_2 ← i + 1 break
        J[j_1, j_2] ← J[j_1, j_2] + v[j, j_1] · v[j, j_2]
        h[j_1] ← h[j_1] − v[j, j_1]
        h[j_2] ← h[j_2] − v[j, j_2]
    return J, h
```

# Reduction- step 5

## Remark

Note that we can equally transform the equation as:

$$H_{Pr} = \frac{1}{4} M \mathbb{1} + \frac{1}{4} H_p$$

# Reduction- step 5

## Remark

Note that we can equally transform the equation as:

$$H_{Pr} = \frac{1}{4} M \mathbb{1} + \frac{1}{4} H_p$$

Example:

$$F = (x_1 \vee x_2) \wedge (x_2 \vee \neg x_1) \wedge (x_3 \vee x_1) \wedge (\neg x_2 \vee \neg x_1)$$

# Reduction- step 5

## Remark

Note that we can equally transform the equation as:

$$H_{Pr} = \frac{1}{4} M \mathbb{1} + \frac{1}{4} H_p$$

Example:

$$F = (x_1 \vee x_2) \wedge (x_2 \vee \neg x_1) \wedge (x_3 \vee x_1) \wedge (\neg x_2 \vee \neg x_1)$$

The result is $v, J$ and $h$, whereby $v$ is a size matrix $4 \times 3$

$$v = \begin{array}{c} \begin{array}{ccc} v_1^k & v_2^k & v_3^k \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix} \end{array} \begin{array}{l} \\ \leftarrow \Omega_1 \\ \leftarrow \Omega_2 \\ \leftarrow \Omega_3 \\ \leftarrow \Omega_4 \end{array}$$

# Theorem

The main results of my work:

# Theorem

The main results of my work:

### Theorem

For a given formula $\phi$ in the form of 2-CNF with $M$ clauses, the number of clauses to be satisfiable is $K$ if and only if when $E_{Ising} = 4(M - K) + M$, where $E_{Ising}$ is solution of the Ising model with coefficients $J, h$.

# Experiments:

- D-Wave:
  https://docs.dwavesys.com/docs/latest/c_gs_2.html

[6] Neural heuristics for SAT solving Sebastian Jaszczur and Michał Łuszczyk and Henryk Michalewski [1]

# Experiments:

- D-Wave: `https://docs.dwavesys.com/docs/latest/c_gs_2.html`
- Wildqat `https://github.com/satproject/Wildqat`

[6] Neural heuristics for SAT solving Sebastian Jaszczur and Michał Łuszczyk and Henryk Michalewski [1]

# Experiments:

- D-Wave:
  `https://docs.dwavesys.com/docs/latest/c_gs_2.html`
- Wildqat `https://github.com/satproject/Wildqat`
- SIMCim `https://arxiv.org/pdf/1901.08927.pdf` [2]

[6] Neural heuristics for SAT solving Sebastian Jaszczur and Michał Łuszczyk and Henryk Michalewski [1]

# Experiments:

- D-Wave: `https://docs.dwavesys.com/docs/latest/c_gs_2.html`
- Wildqat `https://github.com/satproject/Wildqat`
- SIMCim `https://arxiv.org/pdf/1901.08927.pdf` [2]
- `akmaxsat`

[6] Neural heuristics for SAT solving Sebastian Jaszczur and Michał Łuszczyk and Henryk Michalewski [1]

# Experiments:

- D-Wave: https://docs.dwavesys.com/docs/latest/c_gs_2.html
- Wildqat https://github.com/satproject/Wildqat
- SIMCim https://arxiv.org/pdf/1901.08927.pdf [2]
- akmaxsat
- MiniSAT http://minisat.se/MiniSat+.html

[6] Neural heuristics for SAT solving Sebastian Jaszczur and Michał Łuszczyk and Henryk Michalewski [1]

## Experiments:

- D-Wave: `https://docs.dwavesys.com/docs/latest/c_gs_2.html`
- Wildqat `https://github.com/satproject/Wildqat`
- SIMCim `https://arxiv.org/pdf/1901.08927.pdf` [2]
- `akmaxsat`
- MiniSAT `http://minisat.se/MiniSat+.html`
- brute-force algorytm `https://bit.ly/2GzP4Zo`

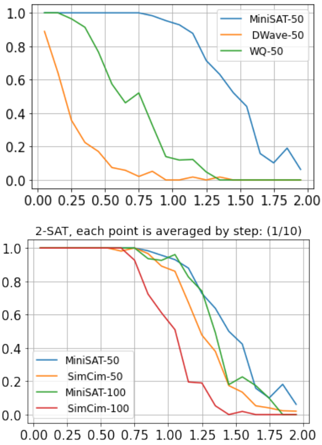
[6] Neural heuristics for SAT solving Sebastian Jaszczur and Michał Łuszczyk and Henryk Michalewski [1]

# Experiments:

- D-Wave: `https://docs.dwavesys.com/docs/latest/c_gs_2.html`
- Wildqat `https://github.com/satproject/Wildqat`
- SIMCim `https://arxiv.org/pdf/1901.08927.pdf` [2]
- `akmaxsat`
- MiniSAT `http://minisat.se/MiniSat+.html`
- brute-force algorytm `https://bit.ly/2GzP4Zo`
- Dataset generator D-SAT-(n) formulas from work [6] .

[6] Neural heuristics for SAT solving Sebastian Jaszczur and Michał Łuszczyk and Henryk Michalewski [1]

# Comparing 2-SAT with MiniSAT algorithm



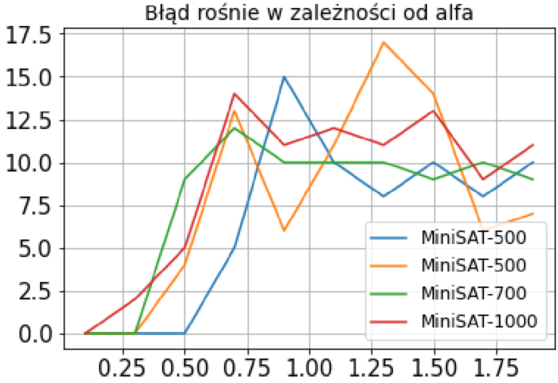

(a) The percentage of formulas satisfiable averaged over what $\frac{1}{10}$ depends on picking $\alpha$.

# Comparing MAX-2-SAT with algorithm `akmaxsat`

For selected 100 formulas with $n \in \{300, 500, 700, 1000\}$

# Comparing MAX-2-SAT with algorithm `akmaxsat`

For selected 100 formulas with $n \in \{300, 500, 700, 1000\}$




Error vs $\alpha$

# Future work

- Is it possible to calculate a probability sampling low energy state without finding a ground state?

# Future work

- Is it possible to calculate a probability sampling low energy state without finding a ground state?
- How to investigate a phase transition without finding an exact ground state?

# Future work

- Is it possible to calculate a probability sampling low energy state without finding a ground state?
- How to investigate a phase transition without finding an exact ground state?
- Provide new metrics for loss, defined as subset maximum satisfiable percent, depending on critical point $= 1$

# Future work

- Is it possible to calculate a probability sampling low energy state without finding a ground state?
- How to investigate a phase transition without finding an exact ground state?
- Provide new metrics for loss, defined as subset maximum satisfiable percent, depending on critical point $= 1$
- New experiments with Pegasus Topology

Sebastian Jaszczur, Michał Łuszczyk, and Henryk Michalewski.

Neural heuristics for sat solving, 2020.

Egor S. Tiunov, Alexander E. Ulanov, and A. I. Lvovsky.
Annealing by simulating the coherent ising machine.
27(7):10288.