



浙江大学  
ZHEJIANG UNIVERSITY

# WELCOME TO TUTORIAL

## Session 5.1 Janus-SAT: A Hybrid Quantum-Classical Solver for 3-SAT Problems



JanusQ  
Cloud



<https://janusq.github.io/tutorials/>

College of Computer Science and  
Technology,  
Zhejiang University

# Outline of Presentation

---



- **Background and challenges**
- HyQSAT overview
- Frontend
- Backend
- Experiment
- API of HyQSAT

# Applications of SAT Problem



浙江大學  
ZHEJIANG UNIVERSITY

## Propositional satisfiability problem (SAT)



Cryptography



Software Testing



Planning

Protein structure analysis  
Knowledge inference .....



Artificial Intelligence



浙江大學  
ZHEJIANG UNIVERSITY

# Example: A Motor Vehicle Parts Production Line



浙江大學  
ZHEJIANG UNIVERSITY

A product line can produce **1,000** products per day.

**20,000** products need to be produced, including **A, B, C, D....**

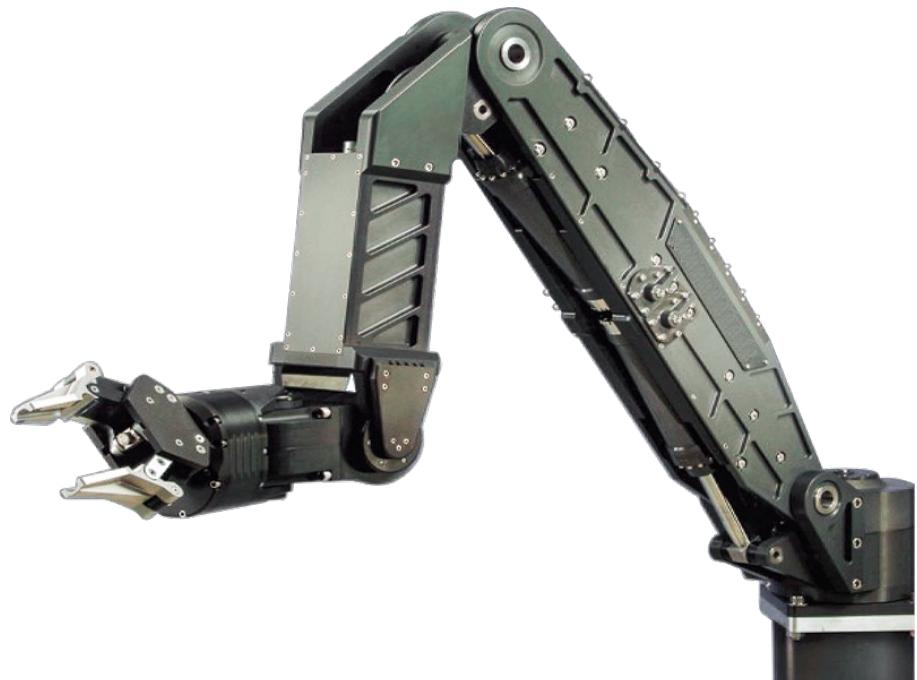
## Constraints:

A and B must be produced together;

B must be produced together with one of E, F or G;

C cannot be produced with E together ;

.....



The optimal classical algorithm takes **3** days to find the optimal schedule.

# Formulation the SAT Problem: An Example



浙江大學  
ZHEJIANG UNIVERSITY

A SAT problem  $C$  in a **conjunctive normal form** with variables  $x_1, x_2, x_3, x_4$ :

$$C = c_1 \wedge c_2$$

$$c_1 = x_1 \vee x_2 \vee x_3$$

$$c_1 = \neg x_1 \vee x_2 \vee x_3$$

The **clause** means :  $x_1$  or  $x_2$  or  $x_3$

A **solution** of the given problem:

$$x_1 = x_2 = 0$$

$$x_3 = x_4 = 1$$

All clauses need to be satisfied.

**3-SAT problem:** each clause has no more than 3 variables. **The first NP-complete problem.**



浙江大學  
ZHEJIANG UNIVERSITY

# Optimal Classical Algorithm: CDCL Algorithm



浙江大學  
ZHEJIANG UNIVERSITY

Tree search



$$\begin{aligned}c_1 &= x_1 \vee x_2 \vee x_3 \\c_2 &= x_2 \vee \neg x_3 \vee x_4 \\c_3 &= x_2 \vee \neg x_4\end{aligned}$$

Input problem



浙江大學  
ZHEJIANG UNIVERSITY

# Optimal Classical Algorithm: CDCL Algorithm



Apply a tree search strategy with tree steps:

- 1) Decision
- 2) Propagation
- 3) Conflict resolving

3) Conflict resolving

3) Conflict  
resolving

$$c_1 = x_1 \vee x_2 \vee x_3$$

$$c_2 = x_2 \vee \neg x_3 \vee x_4$$

$$c_3 = x_2 \vee \neg x_4$$

Input problem

$$x_1 = 0$$

1) Decision

$$c_1 = x_2 \vee x_3$$

$$c_2 = x_2 \vee \neg x_3 \vee x_4$$

$$c_3 = x_2 \vee \neg x_4$$

$$x_2 = 0$$

$$\underline{c_1 = x_3}$$

$$c_2 = \neg x_3 \vee x_4$$

$$c_3 = \neg x_4$$

$$x_3 = 1$$

$$\underline{c_2 = x_4}$$

$$c_3 = \neg x_4$$

$$x_4 = 1$$

$$c_3 = 0$$

1) Decision

$$x_2 = 1$$

$$c_1 = 1 \quad \checkmark$$

$$c_2 = 1$$

$$c_3 = 1$$

2) Propagation

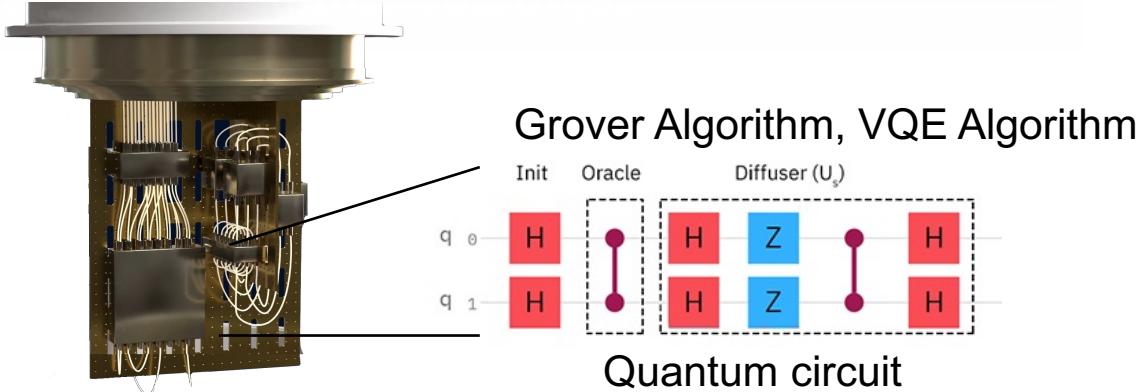
Conflict

# Solving 3-SAT Problems by Quantum Computing

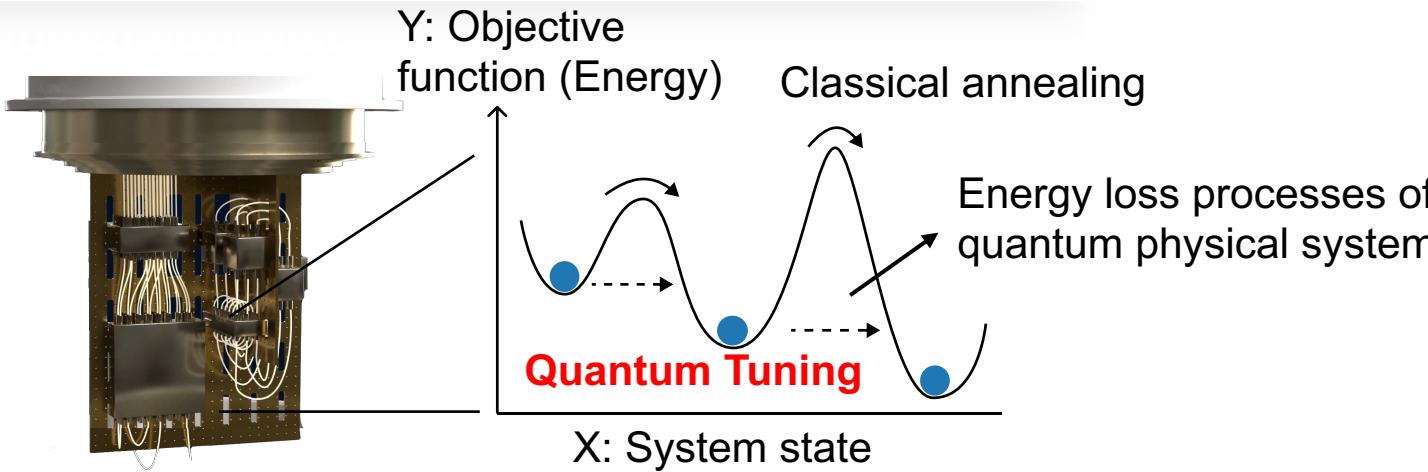


浙江大學  
ZHEJIANG UNIVERSITY

## Gate-based quantum computer



## Quantum annealer



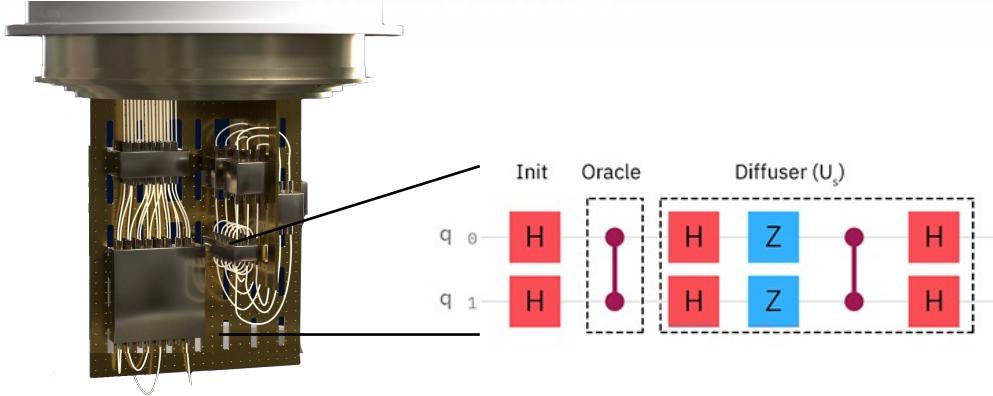
浙江大學  
ZHEJIANG UNIVERSITY

# Solving 3-SAT Problems by Quantum Computing

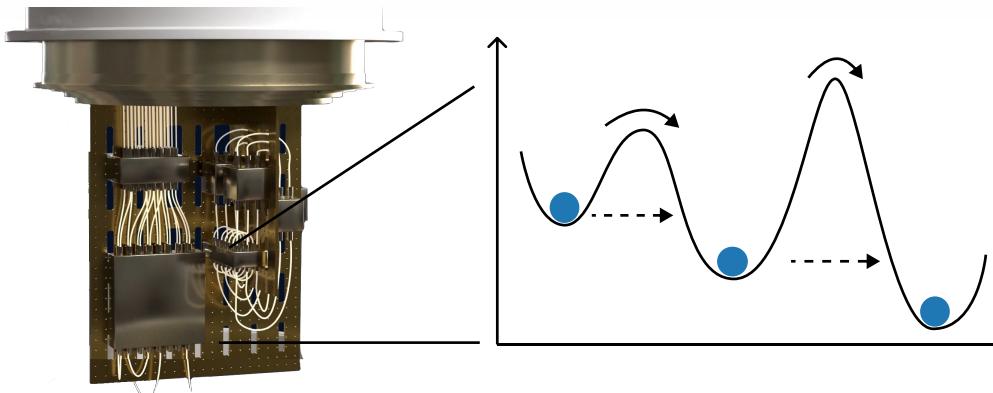


浙江大学  
ZHEJIANG UNIVERSITY

## Gate-based quantum computer



## Quantum annealer



Quantum	Classical	
Gate-based	Quantum annealing (QA)	CDCL
Digital	Simulated	Digital
Quantum superposition	Quantum tunneling	Classical physics
$O(\sqrt{L})$	$O(e^{\sqrt{L}})$	$O(e^L)$
$\sim 100$ qubits	$\sim 2000$ qubits	$>2^{30}$ bits
$\sim 10$ variables	$\sim 50$ variables	$\sim 1000$ variables

# Deploying 3-SAT Problem to Quantum Annealer



浙江大學  
ZHEJIANG UNIVERSITY

The 3-SAT problem first should be transferred into the **minimization problem of a quadratic polynomial objective function**, formulated as:

**Configured**

$$\arg \min_X H_C(X) = I + \sum_{i=1}^L B_i x_i + \sum_{i=1}^L \sum_{j=i+1}^L J_{i,j} x_i x_j,$$

## Example:

$$\begin{aligned} C &= c_1 \wedge c_2 \\ c_1 &= x_1 \vee x_2 \vee x_3, \\ c_2 &= \neg x_2 \vee \neg x_3 \vee x_4 \end{aligned}$$



$$\begin{aligned} H_C(X, A) = & 3 + x_1 + 3x_2 - 2x_3 \\ & - x_4 - 2a_2 + x_1 x_2 \\ & - x_2 x_3 - 2a_1 x_1 - 2a_1 x_2 \\ & + a_1 x_3 - 2a_2 x_2 + 2a_2 x_3 \\ & + a_2 x_4 \end{aligned}$$



浙江大學  
ZHEJIANG UNIVERSITY

# Deploying 3-SAT Problem to Quantum Annealer



## Step 1

$$\begin{aligned}C &= c_1 \wedge c_2 \\c_1 &= x_1 \vee x_2 \vee x_3, \\c_2 &= \neg x_2 \vee \neg x_3 \vee x_4\end{aligned}$$

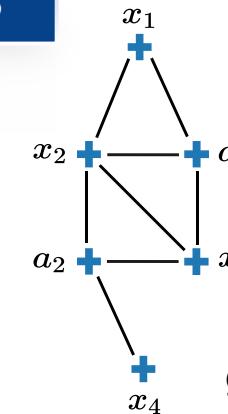
3-SAT problem

## Step 2

$$\begin{aligned}H_C(X, A) = & 3 + x_1 + 3x_2 - 2x_3 \\& - x_4 - 2a_2 + x_1x_2 \\& - x_2x_3 - 2a_1x_1 - 2a_1x_2 \\& + a_1x_3 - 2a_2x_2 + 2a_2x_3 \\& + a_2x_4\end{aligned}$$

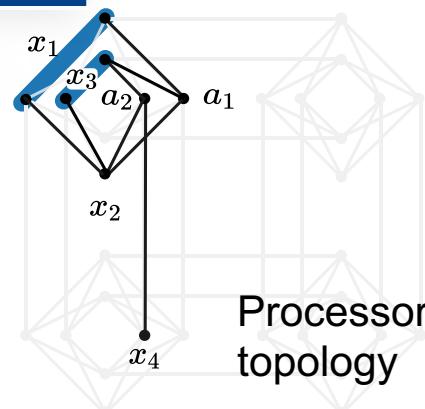
Objective function

## Step 3



Problem graph

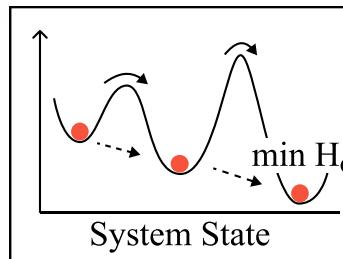
## Step 4



Processor topology

Embedding

## Step 5



## Step 6

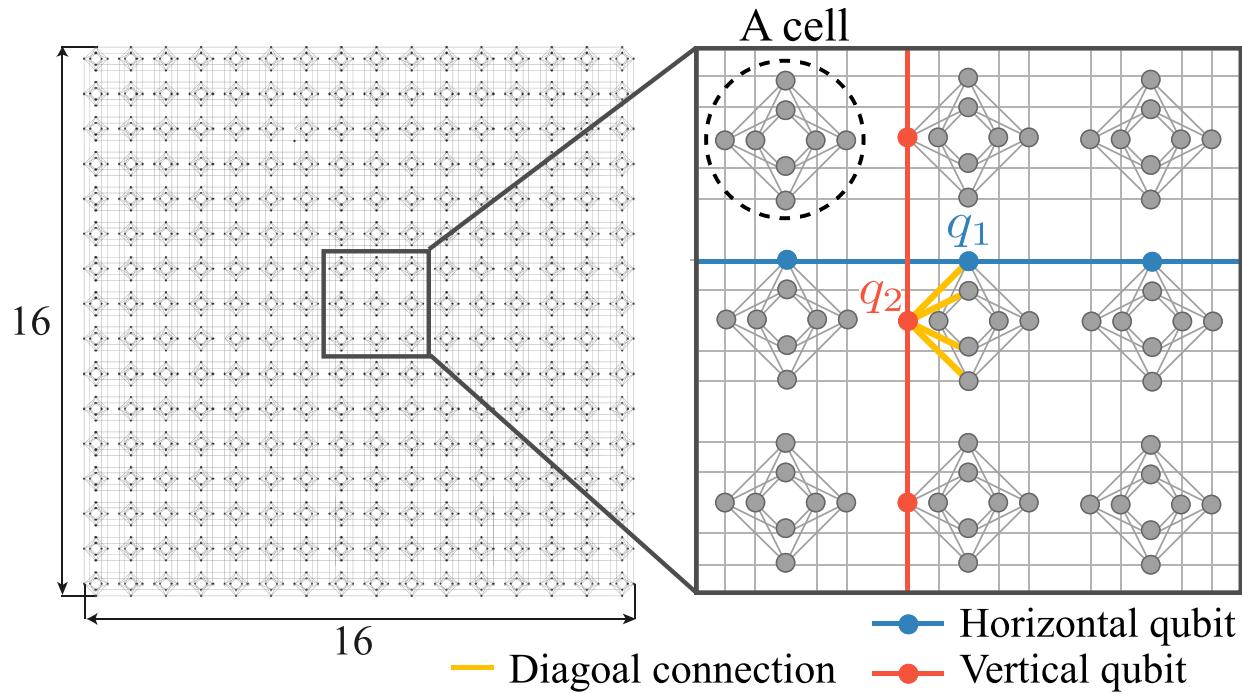
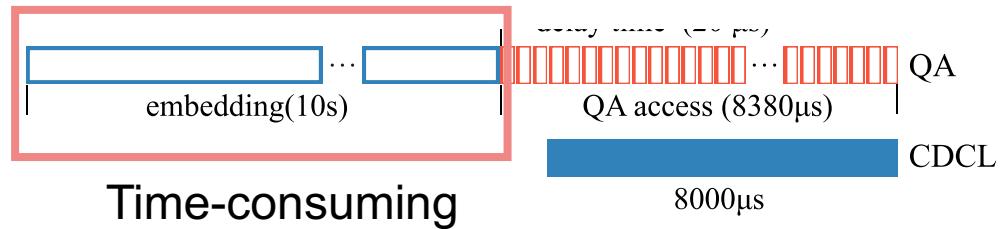
$$\begin{aligned}\min H_c(X, A) &= 0 \\x_1 &= x_2 = 0 \\x_3 &= x_4 = 1\end{aligned}$$

Solution

# Challenge 1 of Quantum Annealing



## High embedding latency

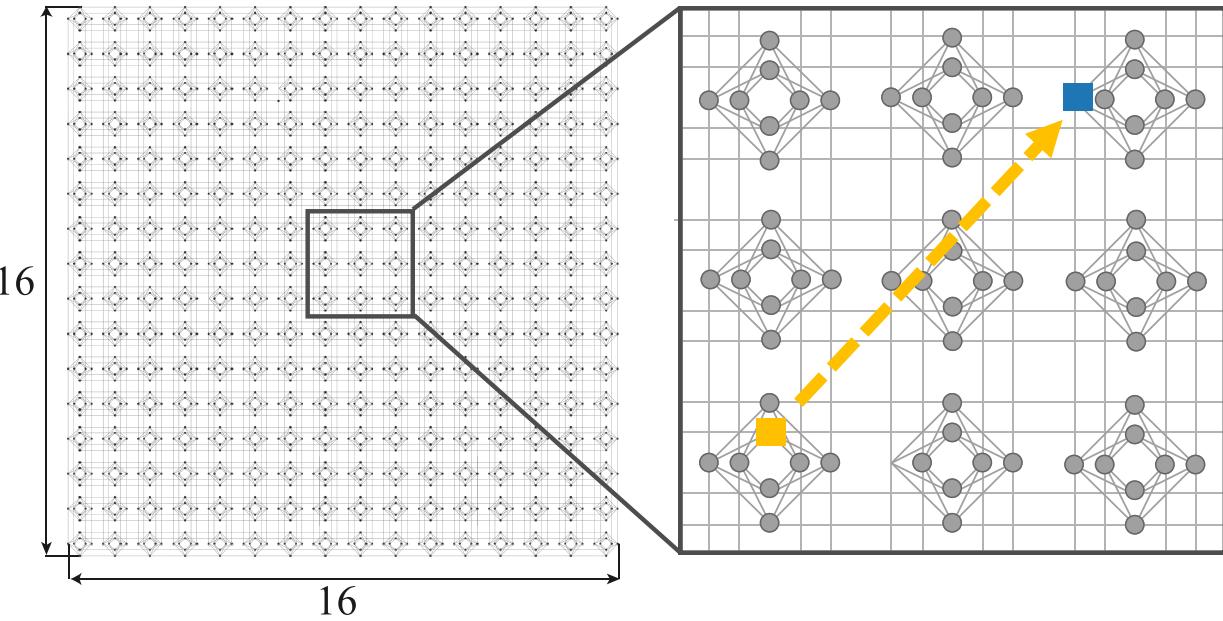
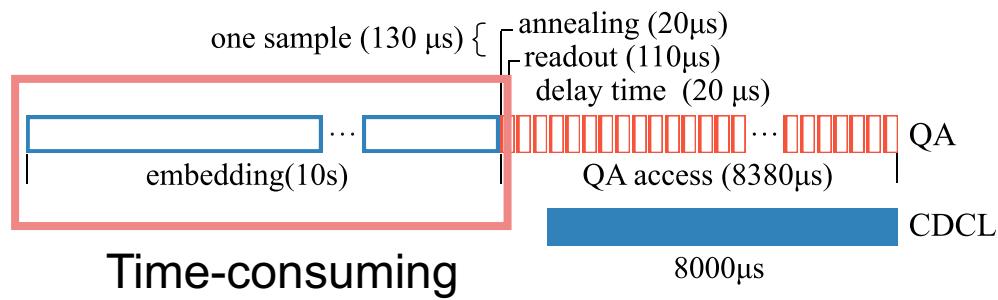


Processor topology of the D-Wave 2000Q

# Challenge 1 of Quantum Annealing



## High embedding latency

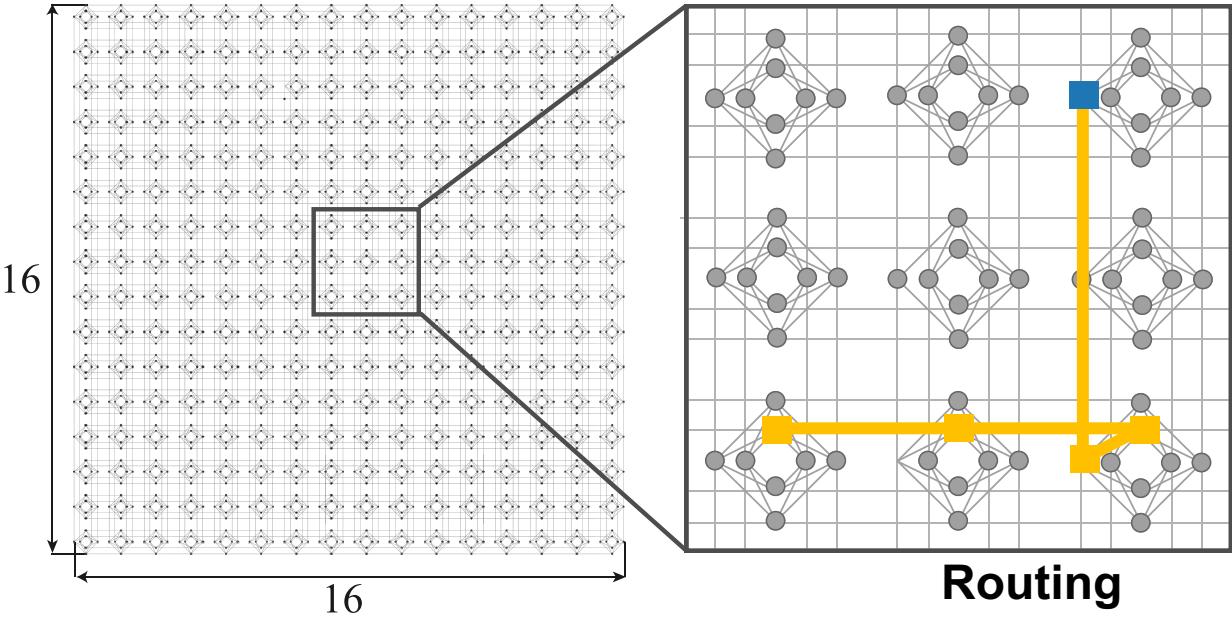
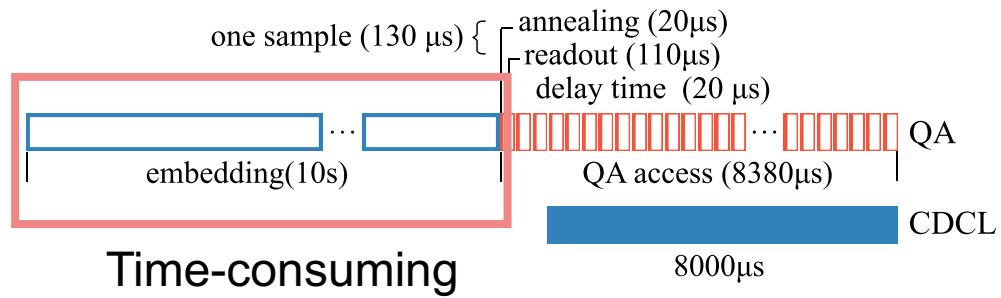


The two most time-consuming parts of the previous embedding schemes: **routing, adjustment**.

# Challenge 1 of Quantum Annealing



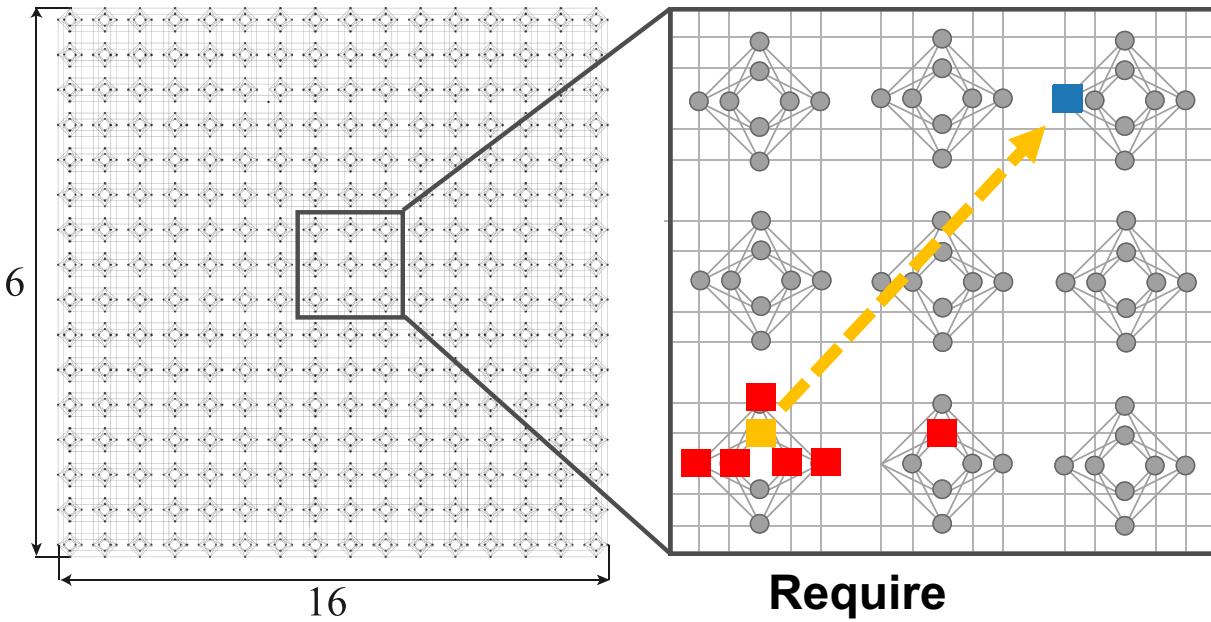
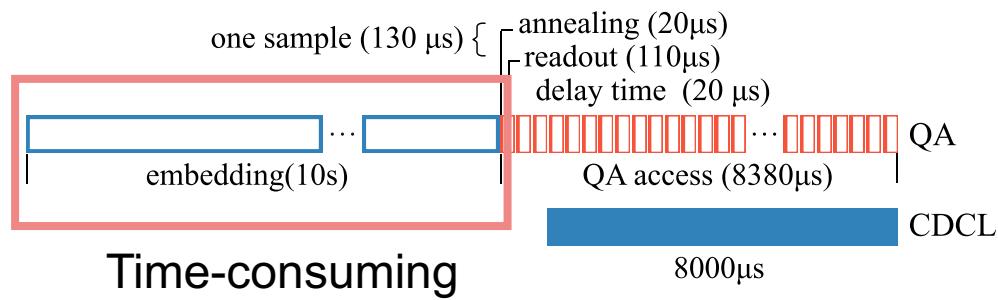
## High embedding latency



The two most time-consuming parts of the previous embedding schemes: **routing, adjustment**.

# Challenge 1 of Quantum Annealing

## High embedding latency

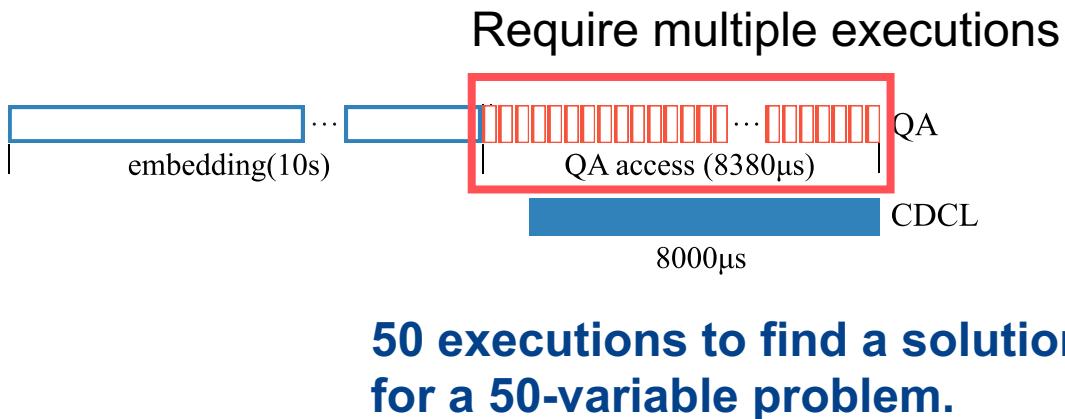


The two most time-consuming parts of the previous embedding schemes: **routing, adjustment**.

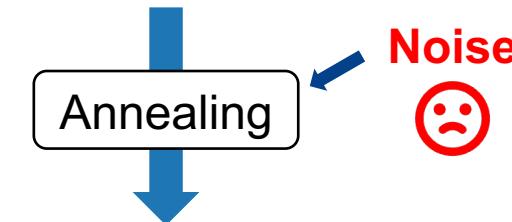
# Challenge 2 of Quantum Annealing



## Noise



$$\begin{aligned}C &= c_1 \wedge c_2 \\c_1 &= x_1 \vee x_2 \vee x_3, \\c_2 &= \neg x_2 \vee \neg x_3 \vee x_4\end{aligned}$$



$$\begin{aligned}x_1 &= x_2 = 0 \\x_3 &= x_4 = 1\end{aligned}$$

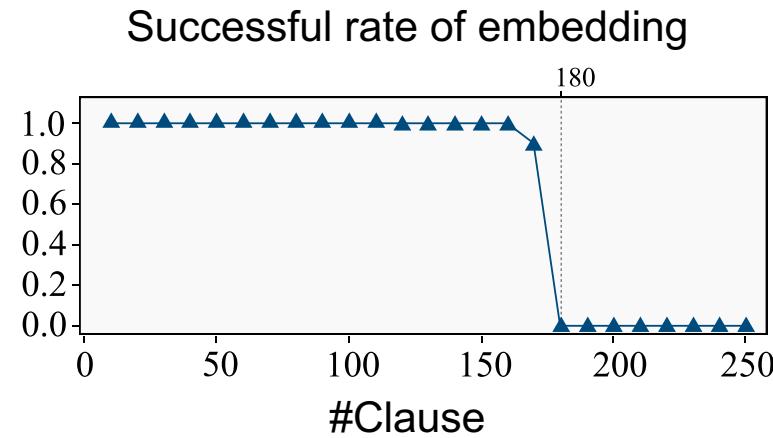
Some variables get wrong assignment.

# Challenge 3 of QA: Limited Problem Scale

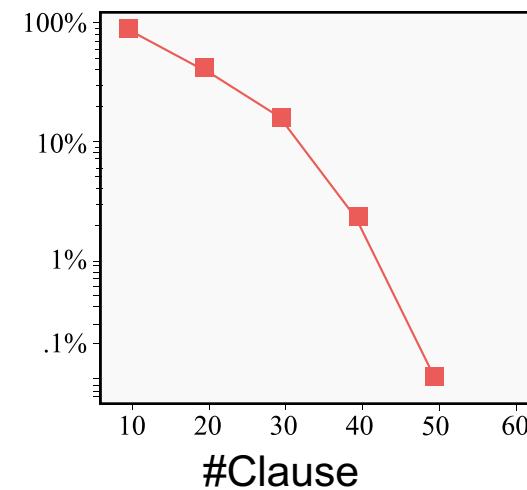


浙江大學  
ZHEJIANG UNIVERSITY

## Limited Problem Scale



Success rate of finding solution



Limited by both the **number of qubits** and **noise**.

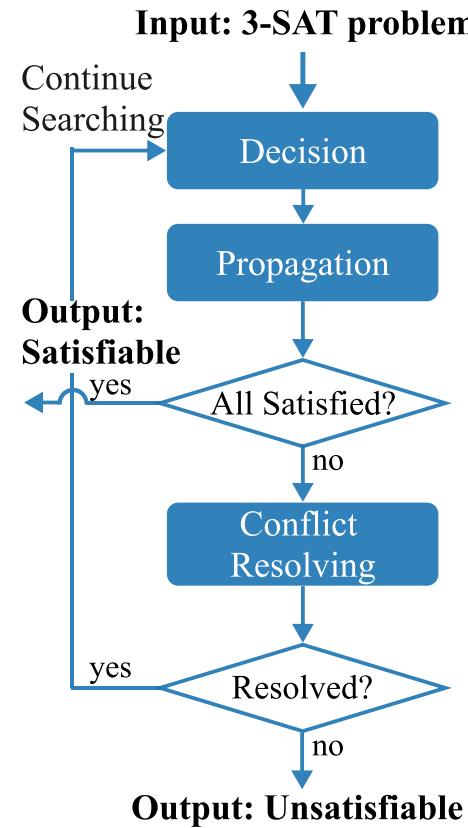
The success rate decreases **exponentially** as the problem size increases.

# Outline of Presentation

---



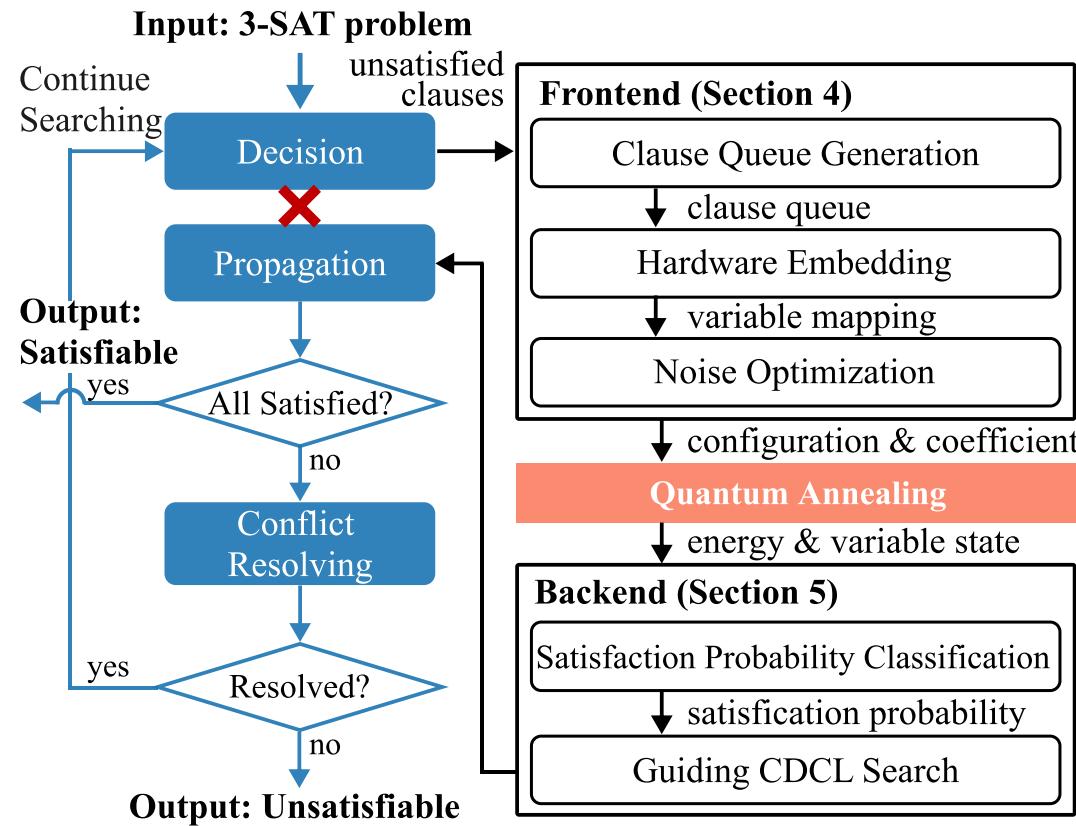
- Background and challenges
- **HyQSAT overview**
- Frontend
- Backend
- Experiment
- API of HyQSAT



CDCL	Quantum Annealing
Large scale	Small scale
Difficulty in solving 'hard' clauses	Quantum speedup;
8000µs	10s+120µs

Caused by embedding

## CDCL workflow



HyQSAT workflow

CDCL	Quantum Annealing
Large scale	Small scale
<b>Difficulty in solving 'hard' clauses</b>	Quantum speedup;
8000µs	<b>10s+120µs</b>

Move critical clauses from CDCL to annealing:

- difficult for CDCL**
- efficiently embedded for quantum annealer**

# Outline of Presentation

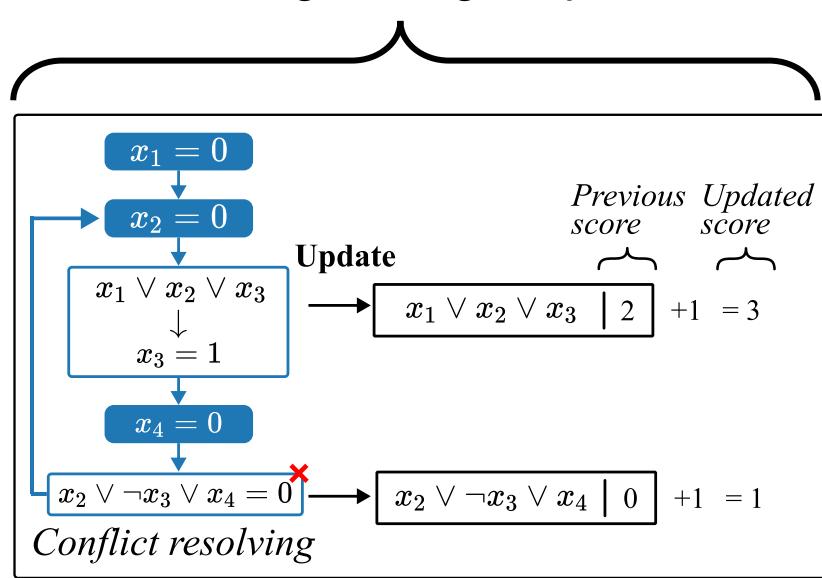
---



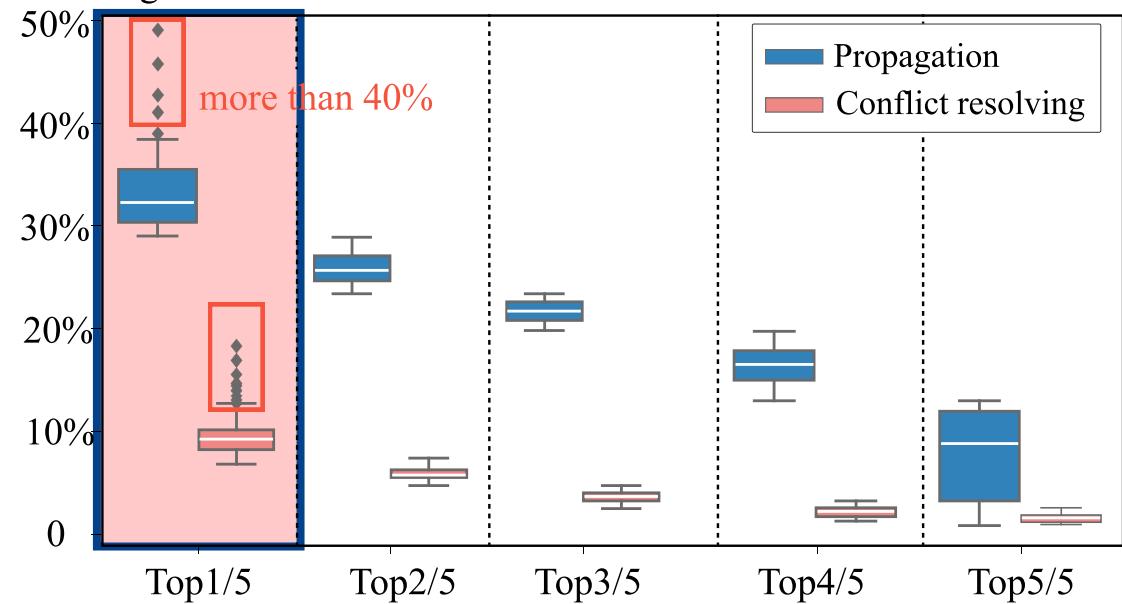
- Background and challenges
- HyQSAT overview
- **Frontend**
- Backend
- Experiment
- API of HyQSAT

## ★ Clause visiting frequency = Difficulty

Predicting visiting frequencies



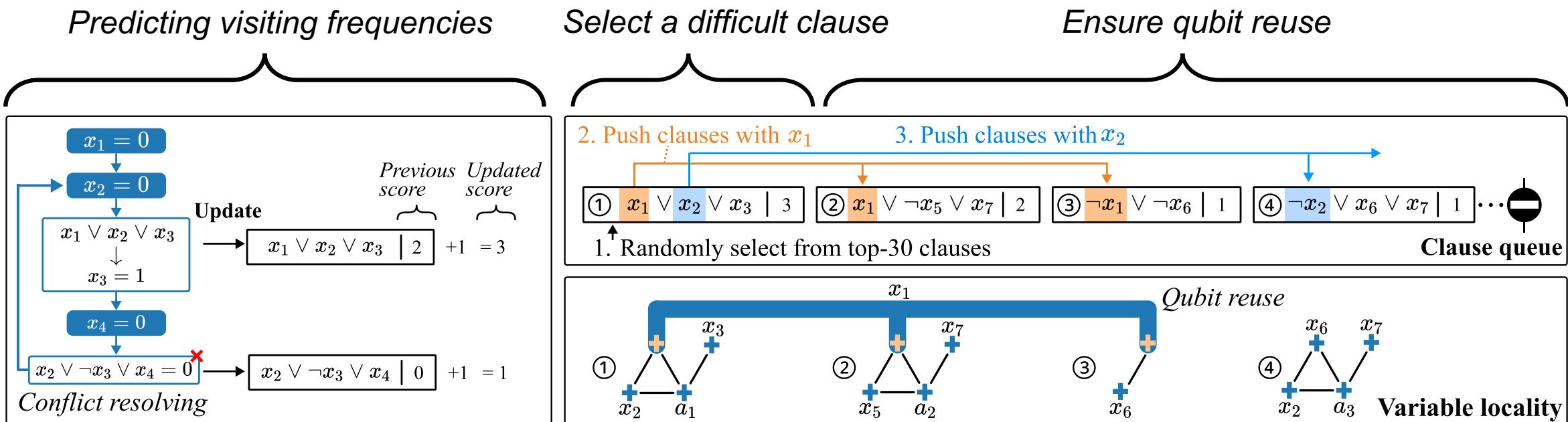
#Visiting / #Iteration



Define activity scores

Difficult clauses

# HyQSAT Frontend: Identify Difficult Clauses



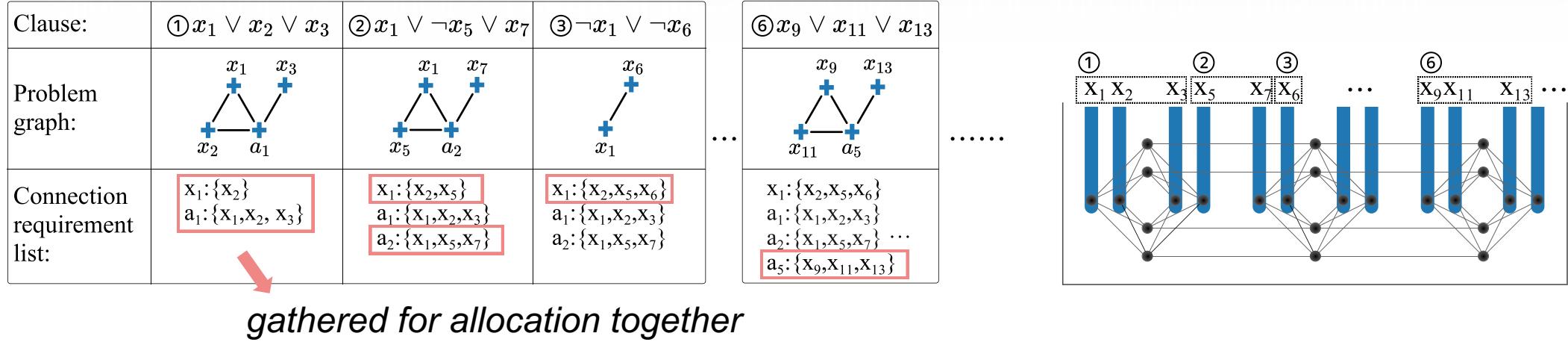
Define activity scores

Apply breadth-first search among clauses with same variables

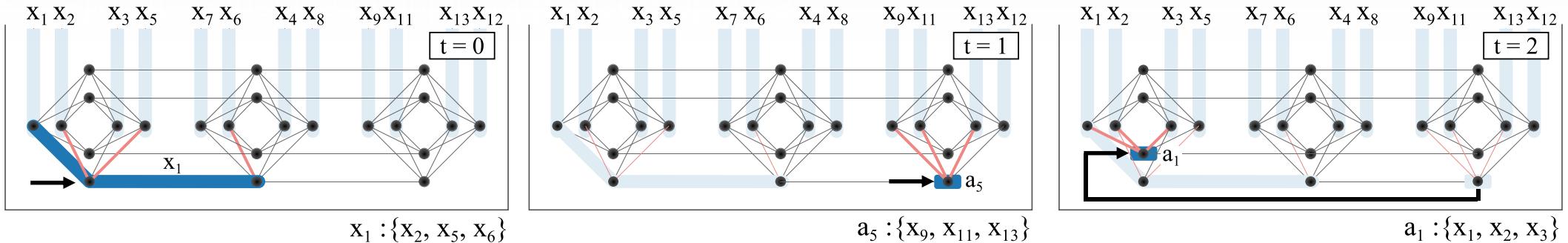
# HyQSAT Frontend: Fast Embedding



## Step 1: Allocate variables to qubits of **vertical lines** according to their order in the clause queue



## Step 2: Allocate variables to qubits of **horizontal lines**



# Outline of Presentation

---

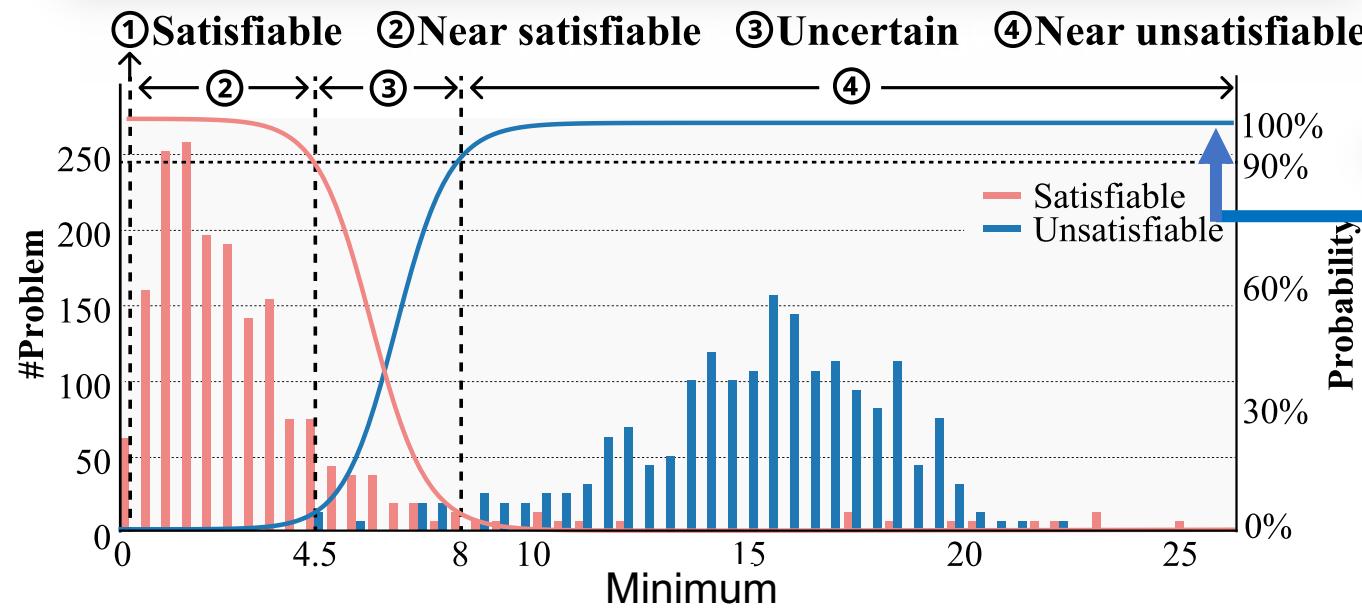


- Background and challenges
- HyQSAT overview
- Frontend
- **Backend**
- Experiment
- API of HyQSAT

Quantum annealing



Minimum value of objective function, Possible solution



Based on the noise model of D-Wave 2000Q

Gaussian Naive Bayes model to estimate the probability of satisfaction

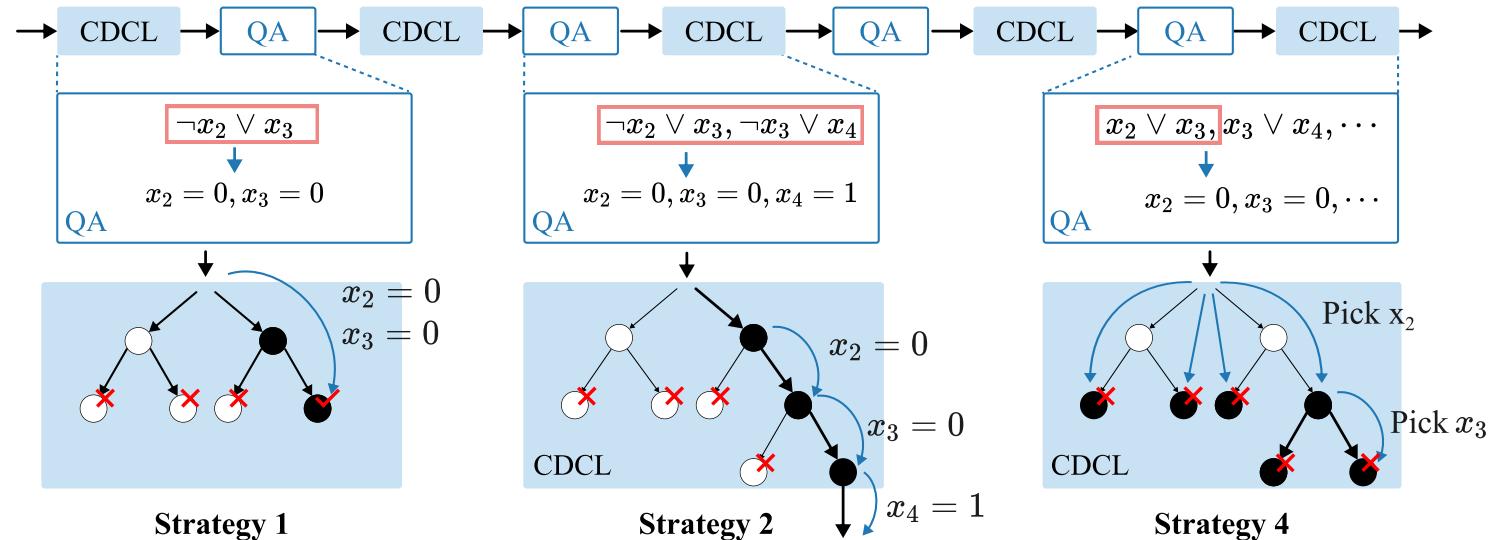
- ① Satisfiable problem: [0, 0].
- ② Near satisfiable problem: (0, 4.5].
- ③ Uncertain problem: (4.5, 8].
- ④ Near unsatisfiable problem: (8,  $+\infty$ ].



Depending on **the number of embedded clauses and their satisfaction probability**, we divide them into **four cases** and propose several feedback strategies to prune the CDCL search space.

	Satisfiable	Near satisfiable	Uncertain	Near unsatisfiable
All embedded	Strategy 1	Strategy 2	Strategy 3	Strategy 4
Not all embedded				

Feedback strategy 1, 2, 4



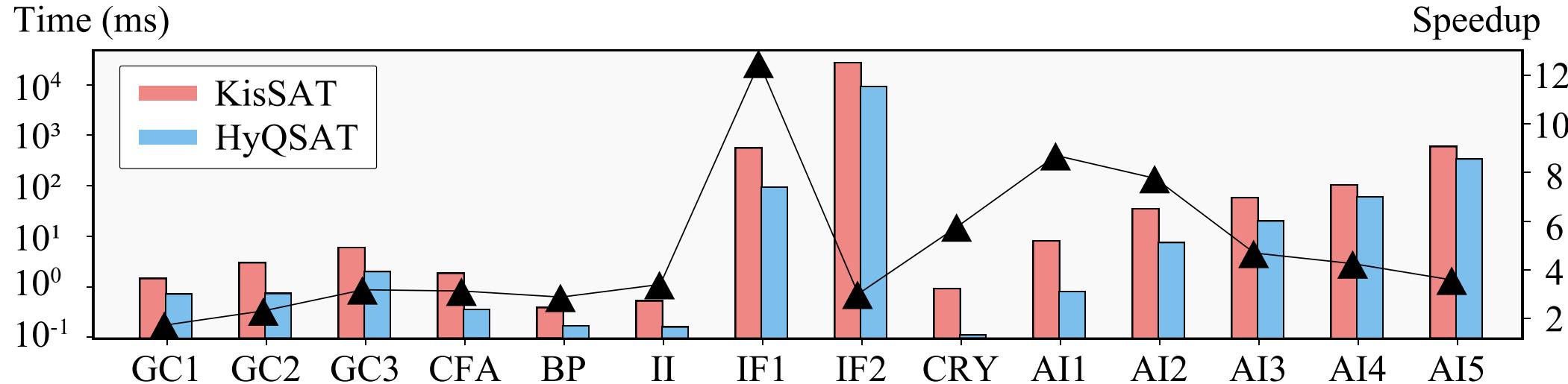
# Outline of Presentation

---



- Background and challenges
- HyQSAT overview
- Frontend
- Backend
- **Experiment**
- API of HyQSAT

# Evaluation on the Real-World Quantum Annealer



graph coloring (CG), circuit fault analysis (CFA), block planning (BP), inductive inference (II), integer factorization (IF), cryptography (CRY), and artificial intelligence (AI)

- 7 domains, 11 benchmarks
- D-Wave 2000Q real-world quantum annealer
- 4.92X speedup compared to KisSAT (win SAT competition 2022)

# Outline of Presentation

---



- Background and challenges
- HyQSAT overview
- Frontend
- Backend
- Experiment
- **API of HyQSAT**



File:

- examples/4-1.solve\_sat\_domain\_problem.ipynb
- [https://janusq.github.io/tutorials/Demonstrations/4-1.solve\\_sat\\_domain\\_problem](https://janusq.github.io/tutorials/Demonstrations/4-1.solve_sat_domain_problem)

Import package  
and data

```
from janusq.hyqsat import solve_by_janusct
```

Configure the  
solver

```
# input cnf file
file_path = "cnf_examples/uf50-01.cnf"
# if verbose
verbose = True
# cpuLim time (s). 0 means infinite
cpu_lim = 0
# memLim . 0 means infinite
mem_lim = 0
```

Solve the problem

```
result_janus = solve_by_janusct(file_path, verb=
verbose, cpu_lim=cpu_lim, mem_lim=mem_lim)
```

加下配置的解释



File:

- examples/4-1.solve\_sat\_domain\_problem.ipynb
- [https://janusq.github.io/tutorials/Demonstrations/4-1.solve\\_sat\\_domain\\_problem](https://janusq.github.io/tutorials/Demonstrations/4-1.solve_sat_domain_problem)

Import package  
and data

```
from janusq.hyqsat import solve_by_janusct
```

Configure the  
solver

```
# input cnf file
file_path = "cnf_examples/uf50-01.cnf"
# if verbose
verbose
# cpuLim . 0 means infinite
cpu_lim = 0
# memLim . 0 means infinite
mem_lim = 0
```

换个大点的数据集

Solve the problem

```
result_janus = solve_by_janusct(file_path, verb=
verbose, cpu_lim=cpu_lim, mem_lim=mem_lim)
```

Output:

```
{  
    'restarts': 1,  
    'conflicts': 9,  
    'conflict cost': 0.054,  
    'decisions': 0,  
    'propagations': 0,  
    'conflict literals': 37,  
    'solving time': 0.355,  
    'annealing time': 0.0,  
    'quantum count': 0,  
    'simulate time': 1.07241,  
    'quantum success number': 9,  
    'quantum conflict number': 13,  
    'quantum one time solve number': 0,  
    'isSatisfiable': True,  
}
```



Solving 3-SAT problems on real-world quantum annealer.

Import package  
and data

```
from janusq.hyqsat import solve_by_janusct
```

Configure the  
solver

```
# input cnf file
file_path = "cnf_examples/uf50-01.cnf"
# if verbose
verbose = True
# cpuLim . 0 means infinite
cpu_lim = 0
mem_lim = 0
```

真机的例子

Solve the problem

```
result_janus = solve_by_janusct(file_path, verb=verbose, cpu_lim=cpu_lim, mem_lim=mem_lim)
```

# Thanks for listening

## HyQSAT: A Hybrid Approach for 3-SAT Problems by Integrating Quantum Annealer with CDCL

Siwei Tan , Mingqian Yu , Andre Python , Yongheng Shang , Tingting Li , Liqiang Lu\*, and Jianwei Yin\*