HPCA 2025 Tutorial

Topic 5. HyQSAT: A Hybrid Quantum-Classical Solver for 3-SAT Problems

Lab

JanusQ Cloud

Speaker: Tianyao Chu

College of Computer Science and Technology
Zhejiang University (ZJU)

https://janusq.github.io/HPCA_2025_Tutorial/

Outline of Presentation

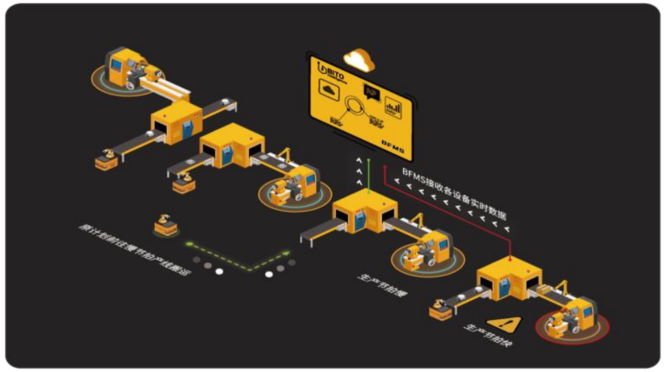
- **Background and challenges**
- HyQSAT overview
- Frontend
- Backend
- Experiment
- API of HyQSAT

Applications of SAT Problem

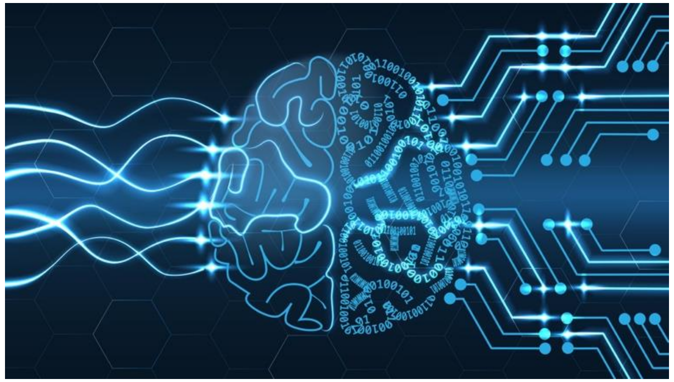
Propositional satisfiability problem (SAT)

Cryptography

Software Testing



Planning



Artificial Intelligence

Protein structure analysis
Knowledge inference ……

Example: A Motor Vehicle Parts Production Line

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

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

$$C = c_1 \wedge c_2$$

$$\boxed{c_1 = x_1 \vee x_2 \vee x_3}$$

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

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

A **solution** of the given problem:

$$\boxed{\begin{array}{l} x_1 = x_2 = 0 \\ x_3 = x_4 = 1 \end{array}}$$

All clauses need to be satisfied.

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

Optimal Classical Algorithm: CDCL Algorithm

Tree search

$c_1 = x_1 \vee x_2 \vee x_3$
$c_2 = x_2 \vee \neg x_3 \vee x_4$
$c_3 = x_2 \vee \neg x_4$

Input problem

Optimal Classical Algorithm: CDCL Algorithm

浙江大學 ZHEJIANG UNIVERSITY

Apply a tree search strategy with tree steps:

1) Decision
2) Propagation
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

$$\boxed{x_1 = 0}$$

1) Decision

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

$$c_1 = x_1 \vee x_2 \vee x_3$$
$$c_2 = x_2 \vee \neg x_3 \vee x_4$$
$$c_3 = x_2 \vee \neg x_4$$

Input problem

$$\downarrow$$

$$\boxed{x_1 = 0}$$

1) Decision

$$\downarrow$$

$$c_1 = x_2 \vee x_3$$
$$c_2 = x_2 \vee \neg x_3 \vee x_4$$
$$c_3 = x_2 \vee \neg x_4$$

Optimal Classical Algorithm: CDCL Algorithm

Apply a tree search strategy with tree steps:

1) Decision
2) Propagation
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

$$c_1 = x_2 \vee x_3$$
$$c_2 = x_2 \vee \neg x_3 \vee x_4$$
$$c_3 = x_2 \vee \neg x_4$$

$x_2 = 0$

1) Decision

$$c_1 = x_3$$
$$c_2 = \neg x_3 \vee x_4$$
$$c_3 = \neg x_4$$

Optimal Classical Algorithm: CDCL Algorithm

Apply a tree search strategy with tree steps:

1) Decision
2) Propagation
3) Conflict resolving

$$c_1 = x_1 \lor x_2 \lor x_3$$
$$c_2 = x_2 \lor \neg x_3 \lor x_4$$
$$c_3 = x_2 \lor \neg x_4$$

Input problem

$$x_1 = 0$$

1) Decision

$$c_1 = x_2 \lor x_3$$
$$c_2 = x_2 \lor \neg x_3 \lor x_4$$
$$c_3 = x_2 \lor \neg x_4$$

$$x_2 = 0$$

$$c_1 = x_3$$
$$c_2 = \neg x_3 \lor x_4$$
$$c_3 = \neg x_4$$

$$\boxed{x_3 = 1}$$

2) Propagation

Optimal Classical Algorithm: CDCL Algorithm

ZHEJIANG UNIVERSITY

Apply a tree search strategy with tree steps:

1) Decision
2) Propagation
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

$\downarrow$

$x_1 = 0$

1) Decision

$\downarrow$

$c_1 = x_2 \vee x_3$
$c_2 = x_2 \vee \neg x_3 \vee x_4$
$c_3 = x_2 \vee \neg x_4$

$\downarrow$

$x_2 = 0$

$\downarrow$

$\underline{c_1 = x_3}$
$c_2 = \neg x_3 \vee x_4$
$c_3 = \neg x_4$

$\downarrow$

$\boxed{x_3 = 1}$

2) Propagation

$\downarrow$

$\underline{c_2 = x_4}$
$c_3 = \neg x_4$

ZHEJIANG UNIVERSITY

Optimal Classical Algorithm: CDCL Algorithm

Apply a tree search strategy with tree steps:

1) Decision
2) Propagation
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

$$c_1 = x_2 \vee x_3$$
$$c_2 = x_2 \vee \neg x_3 \vee x_4$$
$$c_3 = x_2 \vee \neg x_4$$

$$x_2 = 0$$

$$c_1 = x_3$$
$$c_2 = \neg x_3 \vee x_4$$
$$c_3 = \neg x_4$$

$$x_3 = 1$$

2) Propagation

$$c_2 = x_4$$
$$c_3 = \neg x_4$$

$$x_4 = 1$$

$$c_3 = 0$$ ✕

Conflict

Optimal Classical Algorithm: CDCL Algorithm

ZHEJIANG UNIVERSITY

Apply a tree search strategy with tree steps:

1) Decision
2) Propagation
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

$\downarrow$

$\boxed{x_1 = 0}$

1) Decision

$\downarrow$

$$c_1 = x_2 \vee x_3$$
$$c_2 = x_2 \vee \neg x_3 \vee x_4$$
$$c_3 = x_2 \vee \neg x_4$$

$\downarrow$

$\boxed{x_2 = 0}$

$\downarrow$

$$c_1 = x_3$$
$$c_2 = \neg x_3 \vee x_4$$
$$c_3 = \neg x_4$$

$\downarrow$

$\boxed{x_3 = 1}$

2) Propagation

$\downarrow$

$$c_2 = x_4$$
$$c_3 = \neg x_4$$

$\downarrow$

$\boxed{x_4 = 1}$

$\downarrow$

$c_3 = 0$ ✕

Conflict

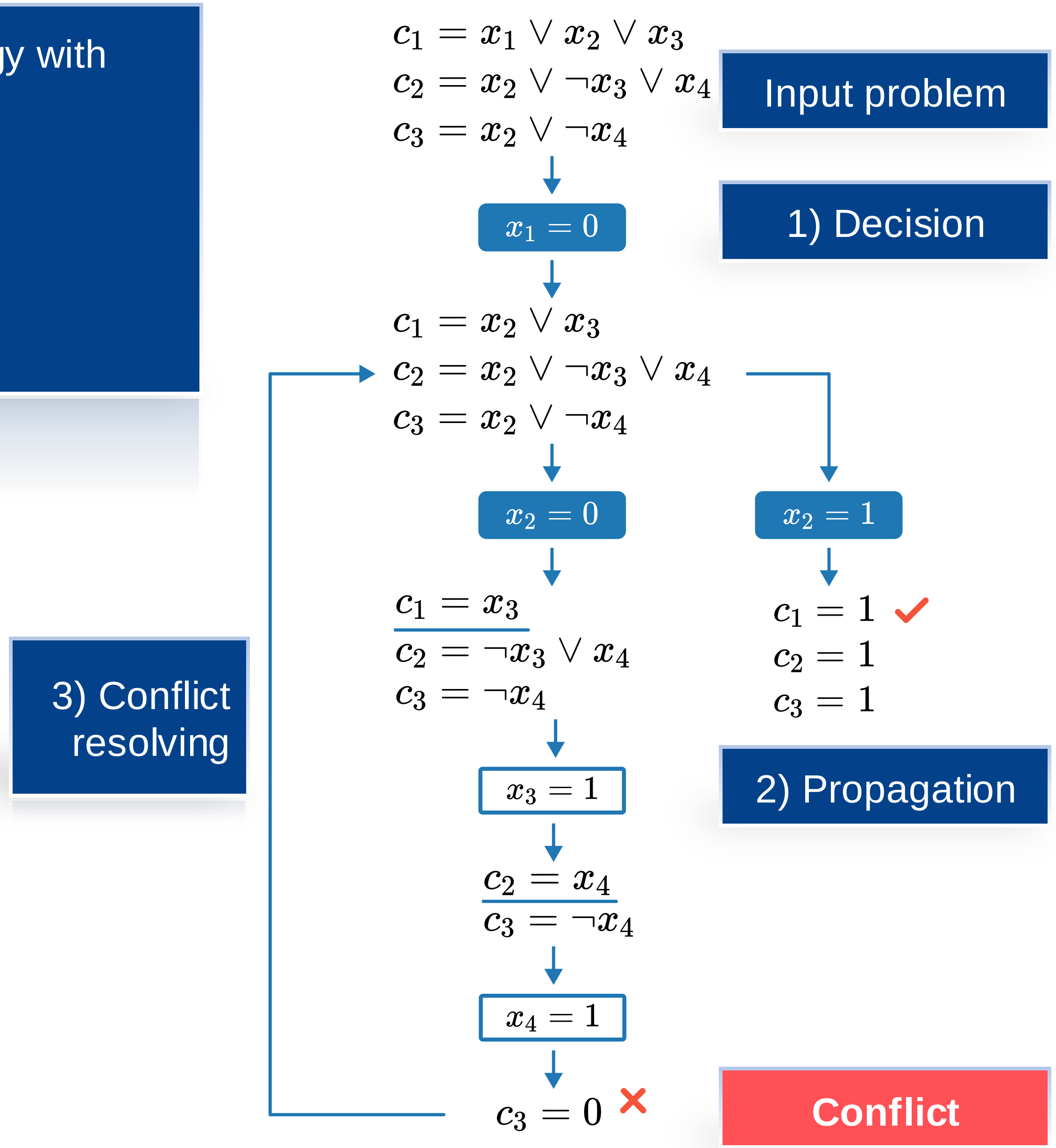
3) Conflict resolving

ZHEJIANG UNIVERSITY

Optimal Classical Algorithm: CDCL Algorithm

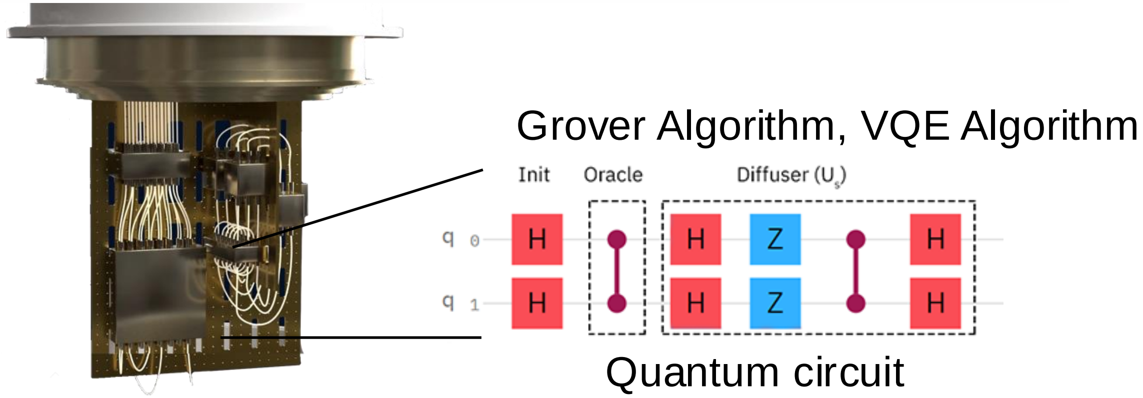
Apply a tree search strategy with tree steps:

1) Decision
2) Propagation
3) Conflict resolving

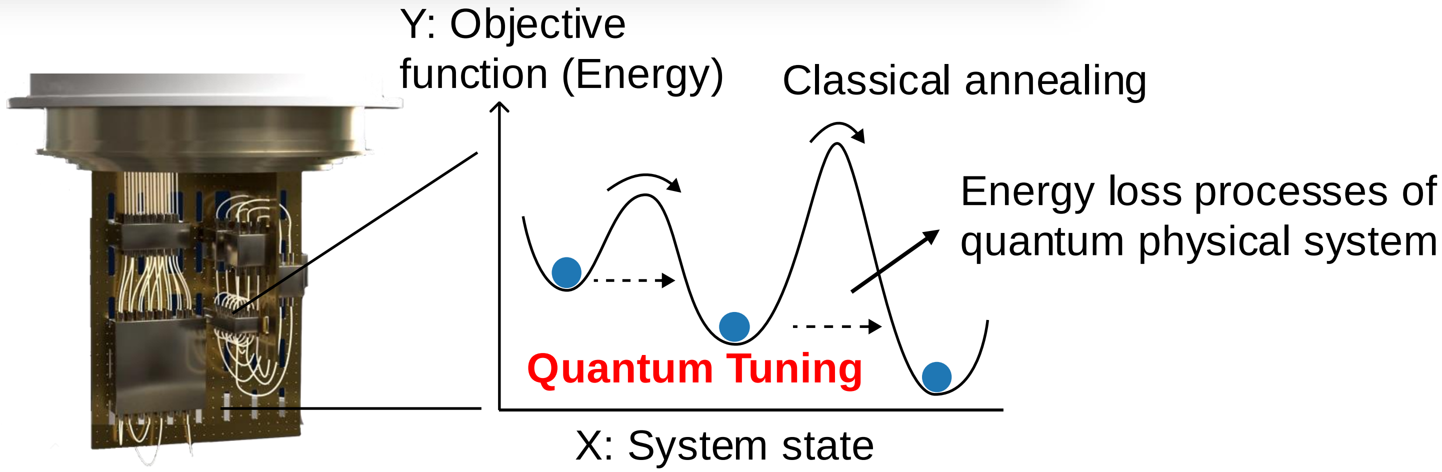


Solving 3-SAT Problems by Quantum Computing

Gate-based quantum computer

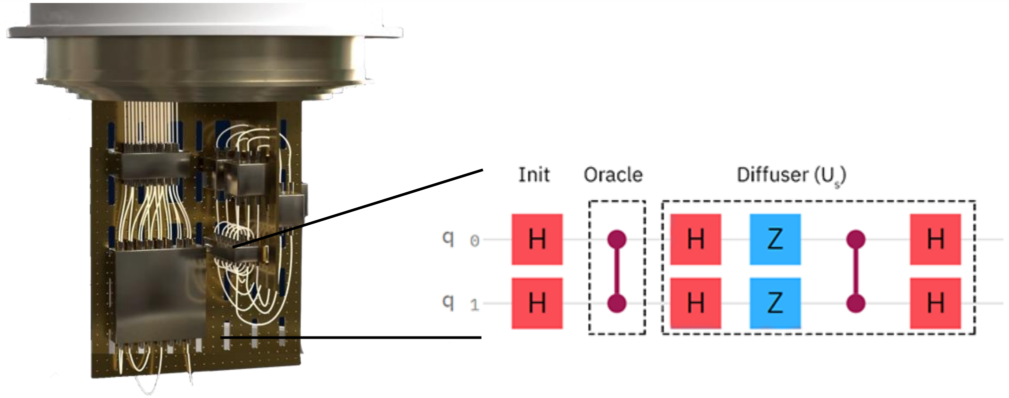


Quantum annealer

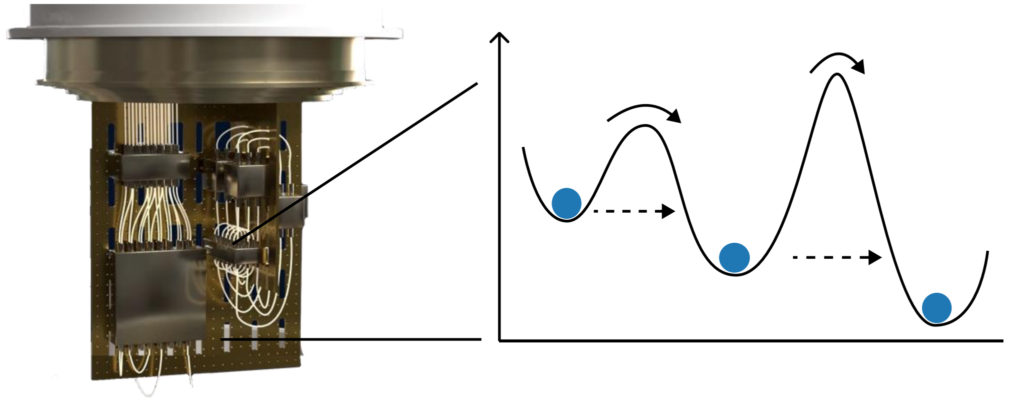


Solving 3-SAT Problems by Quantum Computing

Gate-based quantum computer



Quantum annealer



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

Deploying 3-SAT Problem to Quantum Annealer

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

Configured

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

Example:

$$\begin{aligned} C &= c_1 \wedge c_2 \\ c_1 &= x_1 \vee x_2 \vee x_3, \\ c_2 &= \neg x_2 \vee \neg x_3 \vee x_4 \end{aligned} \quad \Rightarrow \quad \begin{aligned} H_C(X, A) = 3 &+ x_1 + 3x_2 - 2x_3 \\ &- x_4 - 2a_2 + x_1 x_2 \\ &- x_2 x_3 - 2a_1 x_1 - 2a_1 x_2 \\ &+ a_1 x_3 - 2a_2 x_2 + 2a_2 x_3 \\ &+ a_2 x_4 \end{aligned}$$

Deploying 3-SAT Problem to Quantum Annealer

Step 1

$$C = c_1 \wedge c_2$$
$$c_1 = x_1 \vee x_2 \vee x_3,$$
$$c_2 = \neg x_2 \vee \neg x_3 \vee x_4$$

3-SAT problem

Deploying 3-SAT Problem to Quantum Annealer

Step 1

$$C = c_1 \wedge c_2$$
$$c_1 = x_1 \vee x_2 \vee x_3,$$
$$c_2 = \neg x_2 \vee \neg x_3 \vee x_4$$

3-SAT problem

Step 2

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

Objective function

Deploying 3-SAT Problem to Quantum Annealer

Step 1

$$C = c_1 \wedge c_2$$
$$c_1 = x_1 \vee x_2 \vee x_3,$$
$$c_2 = \neg x_2 \vee \neg x_3 \vee x_4$$

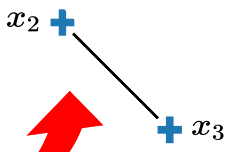
3-SAT problem

Step 2

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

Objective function

Step 3



Problem graph

Deploying 3-SAT Problem to Quantum Annealer

Step 1

$$C = c_1 \wedge c_2$$
$$c_1 = x_1 \vee x_2 \vee x_3,$$
$$c_2 = \neg x_2 \vee \neg x_3 \vee x_4$$

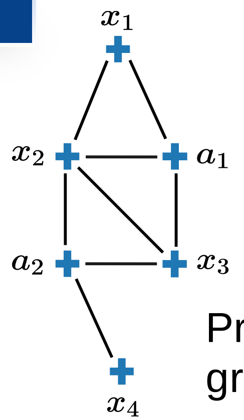
3-SAT problem

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

Deploying 3-SAT Problem to Quantum Annealer

Step 1

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

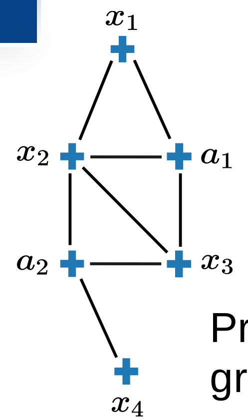
3-SAT problem

Step 2

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

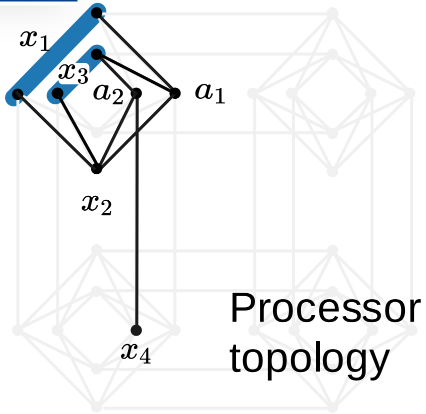
Objective function

Step 3



Problem graph

Step 4



Processor topology

Embedding

Deploying 3-SAT Problem to Quantum Annealer

ZHEJIANG UNIVERSITY

Step 1

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

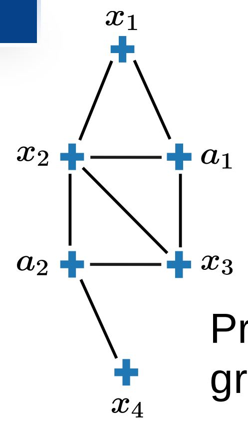
3-SAT problem

Step 2

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

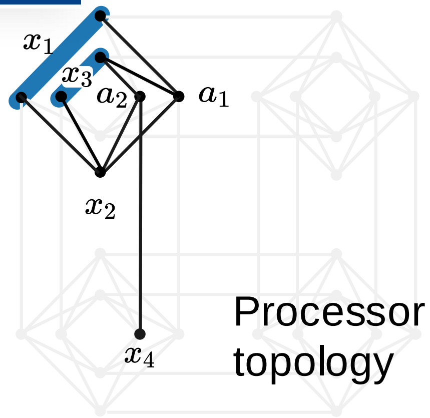
Objective function

Step 3



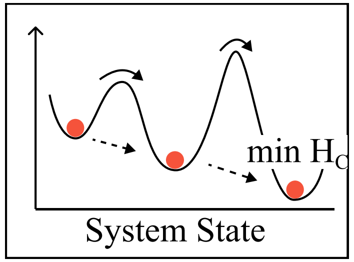
Problem graph

Step 4



Embedding

Step 5



Quantum Annealing

Deploying 3-SAT Problem to Quantum Annealer

ZHEJIANG UNIVERSITY

Step 1

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

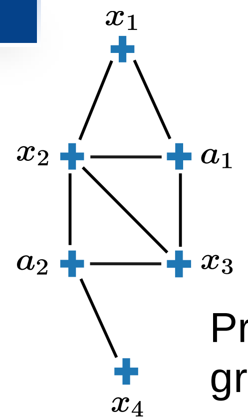
3-SAT problem

Step 2

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

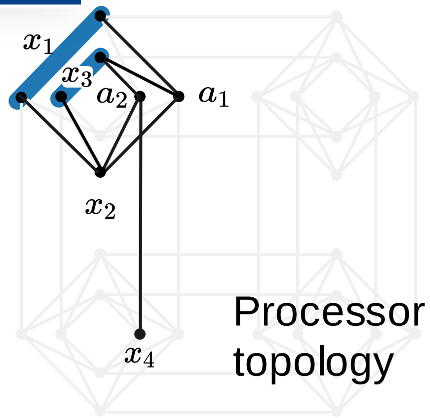
Objective function

Step 3



Problem graph

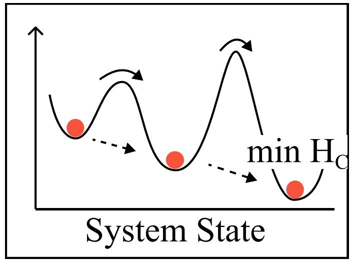
Step 4



Processor topology

Embedding

Step 5



Quantum Annealing

Step 6

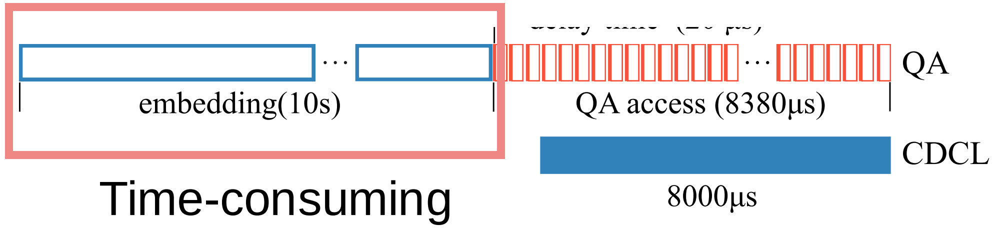
$$min\ H_c(X, A) = 0$$

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

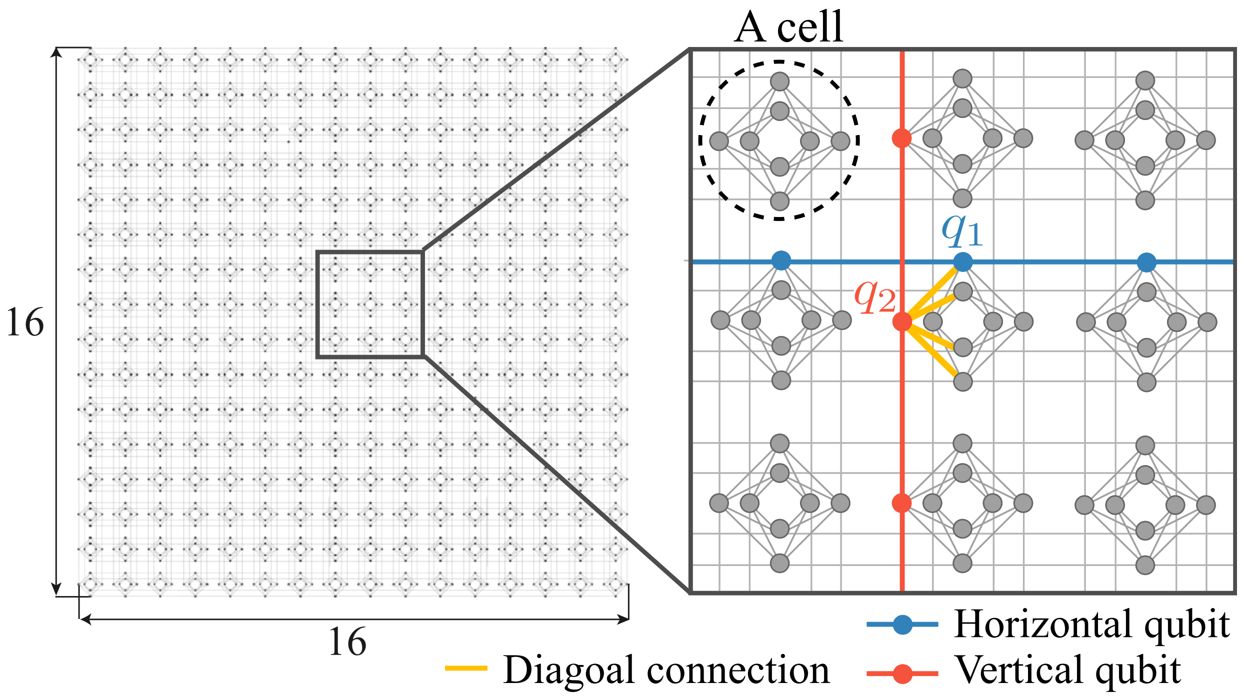
Solution

Challenge 1 of Quantum Annealing

High embedding latency



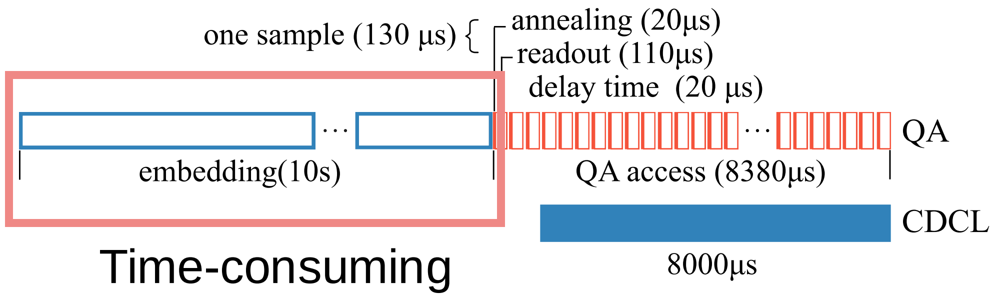
Time-consuming

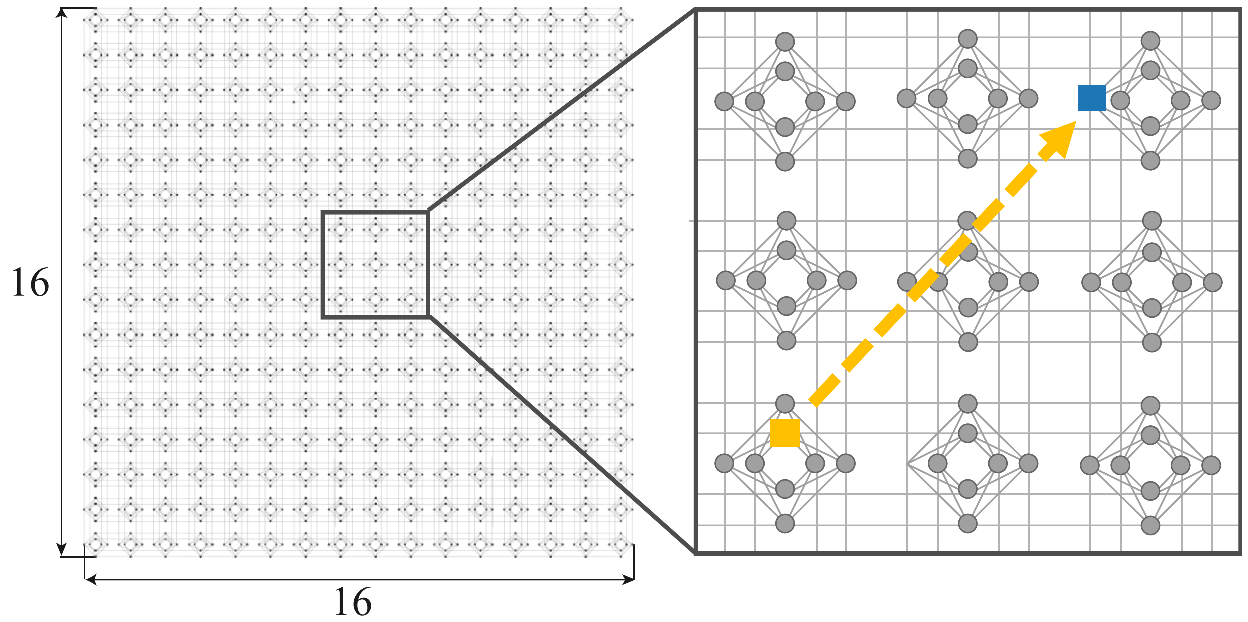


Processor topology of the D-Wave 2000Q

Challenge 1 of Quantum Annealing

High embedding latency



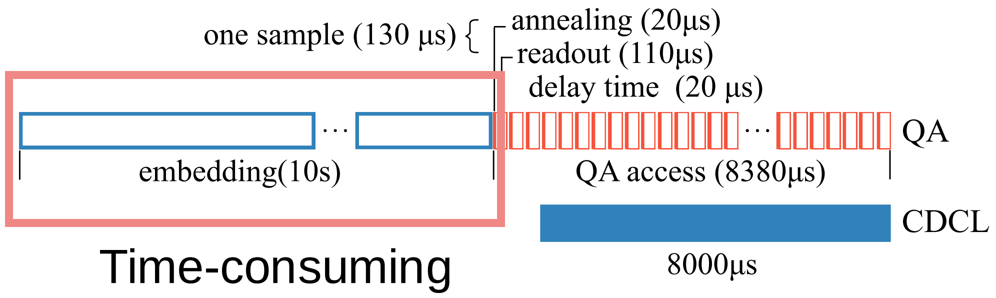


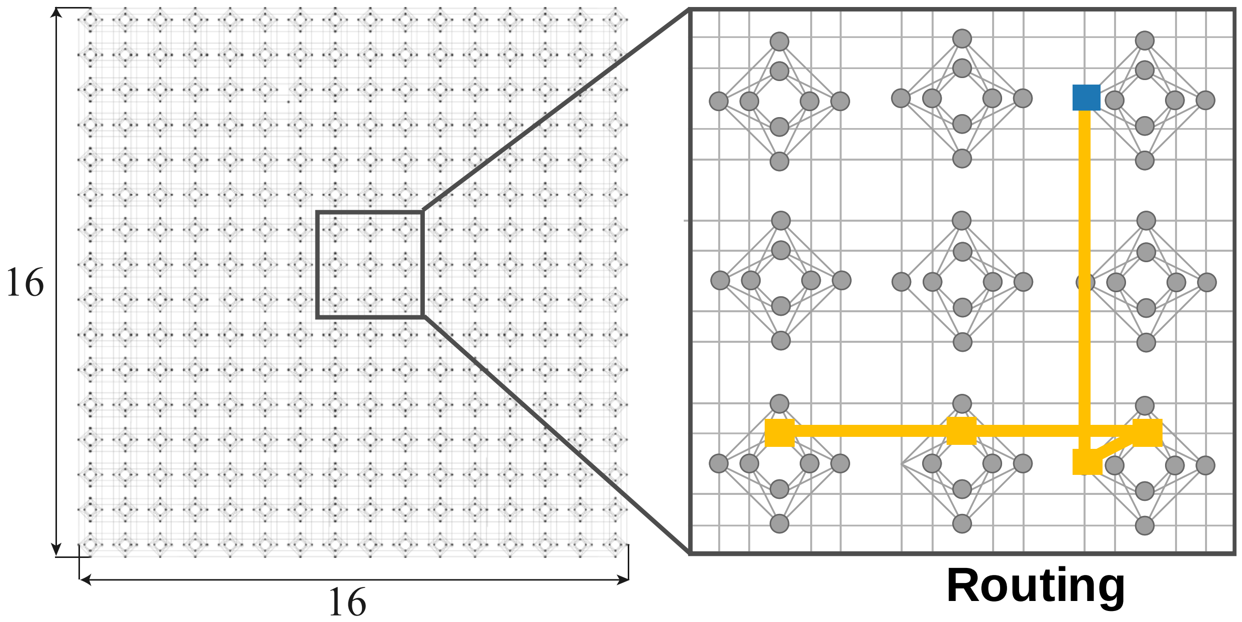
The two most time-consuming parts of the previous embedding schemes: **routing, adjustment**.

Challenge 1 of Quantum Annealing

ZHEJIANG UNIVERSITY

High embedding latency



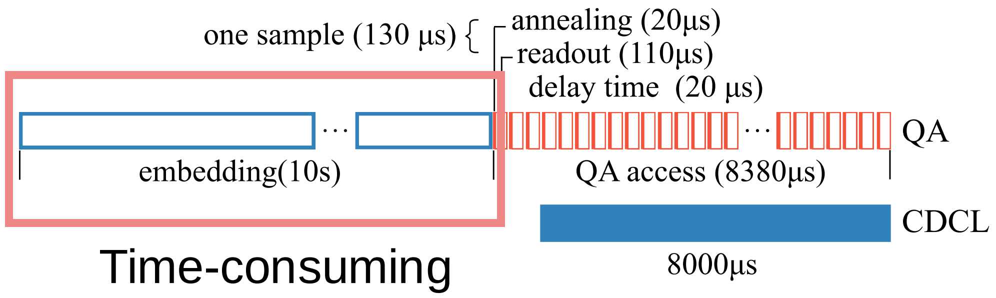


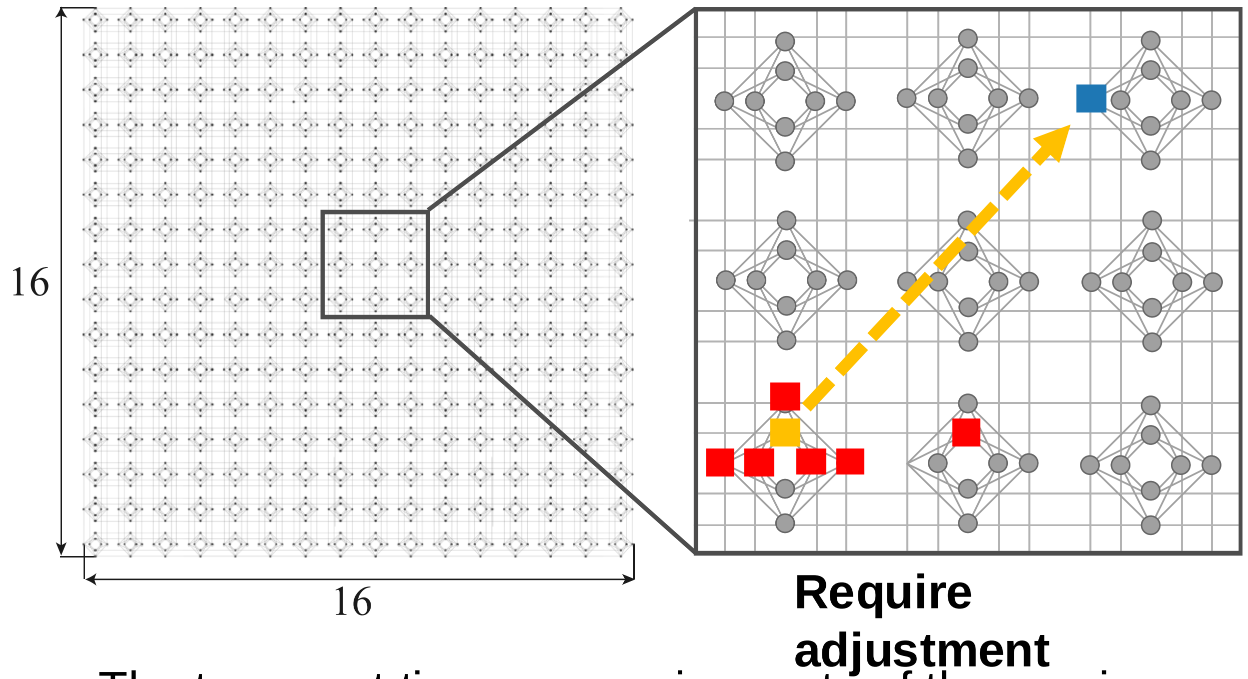
The two most time-consuming parts of the previous embedding schemes: **routing, adjustment**.

ZHEJIANG UNIVERSITY

Challenge 1 of Quantum Annealing

High embedding latency



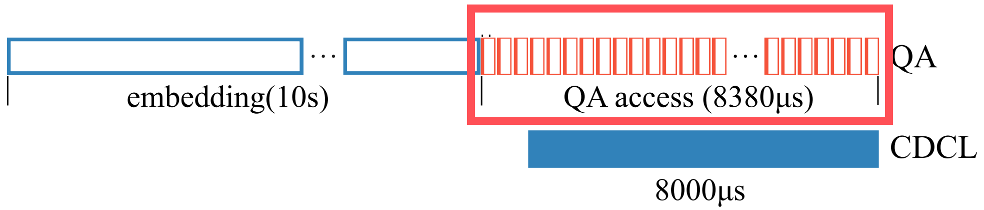


The two most time-consuming parts of the previous embedding schemes: **routing, adjustment**.

Challenge 2 of Quantum Annealing

Noise

Require multiple executions



50 executions to find a solution for a 50-variable problem.

$$C = c_1 \wedge c_2$$
$$c_1 = x_1 \vee x_2 \vee x_3,$$
$$c_2 = \neg x_2 \vee \neg x_3 \vee x_4$$

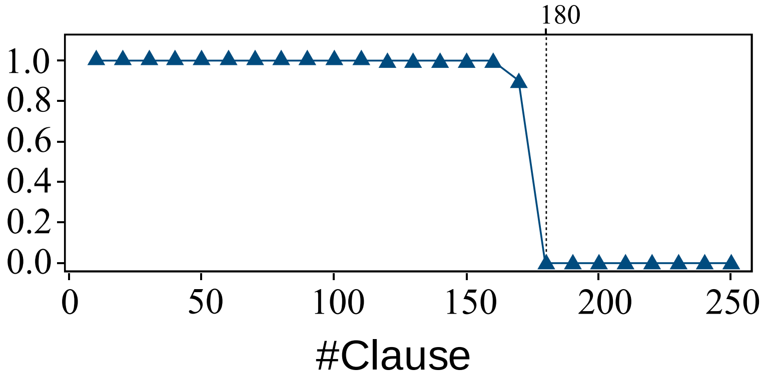
$$x_1 = x_2 = 0$$
$$x_3 = \boxed{x_4} = 1$$

Some variables get wrong assignment.

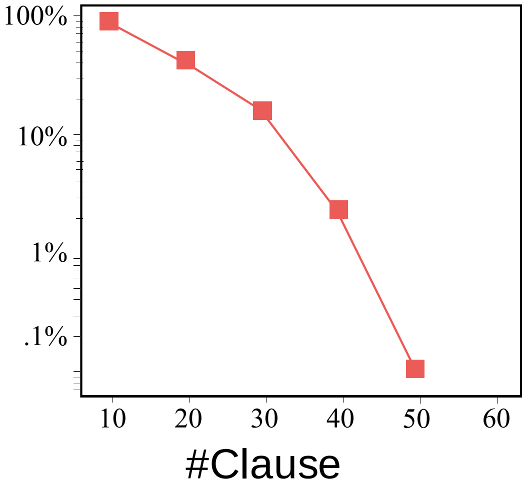
Challenge 3 of QA: Limited Problem Scale

Limited Problem Scale

Successful rate of embedding



Success rate of finding solution



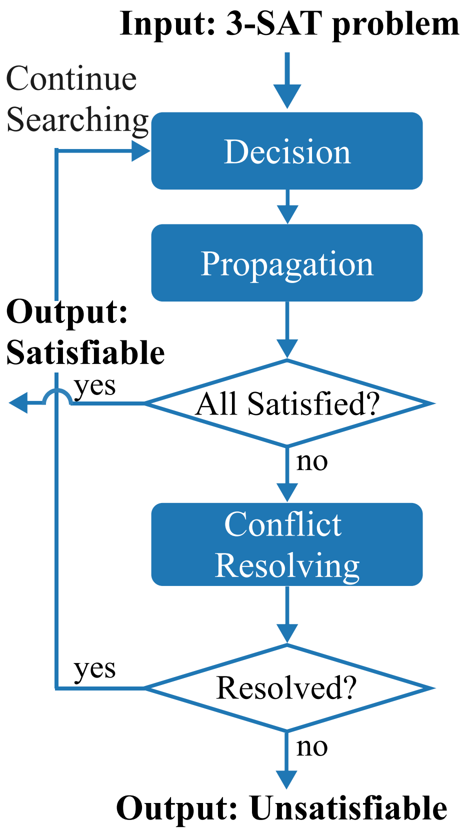
Limited by both the **number of qubits** and **noise.**

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

Outline of Presentation

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

HyQSAT Workflow

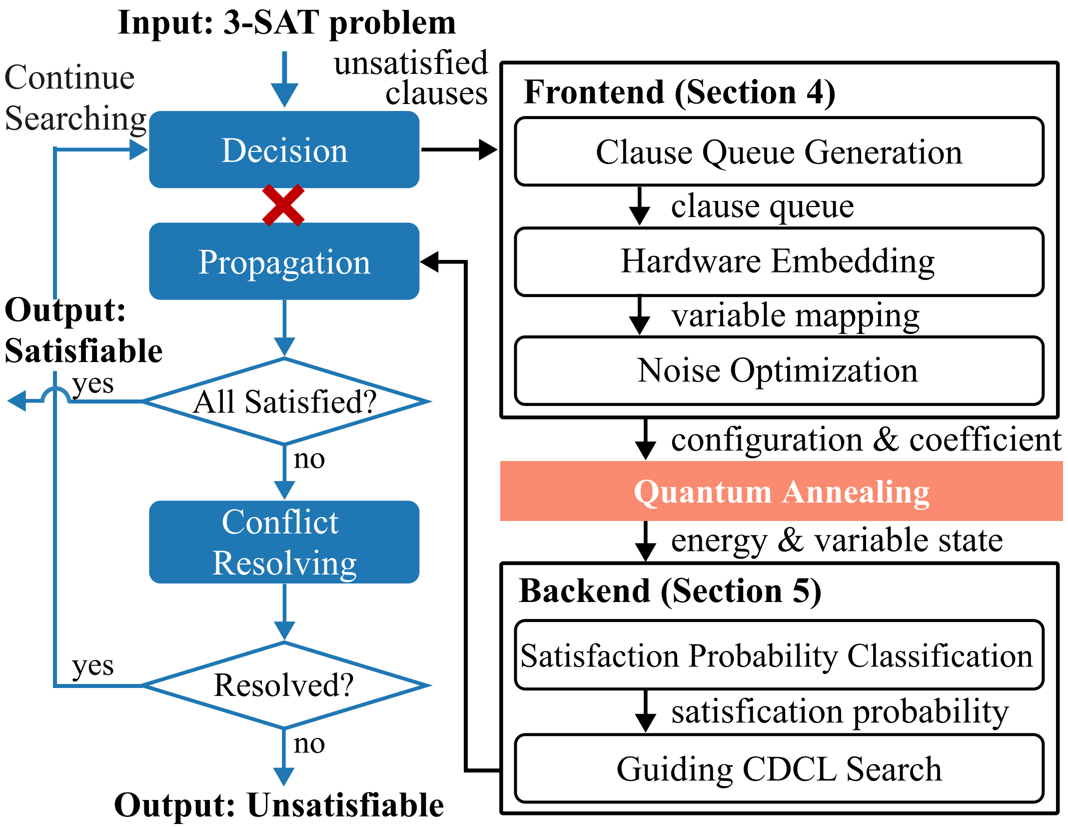


CDCL workflow

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

Caused by embedding

HyQSAT Workflow



HyQSAT workflow

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

Move critical clauses from CDCL to annealing:

1. **difficult for CDCL**
2. **efficiently embedded for quantum annealer**

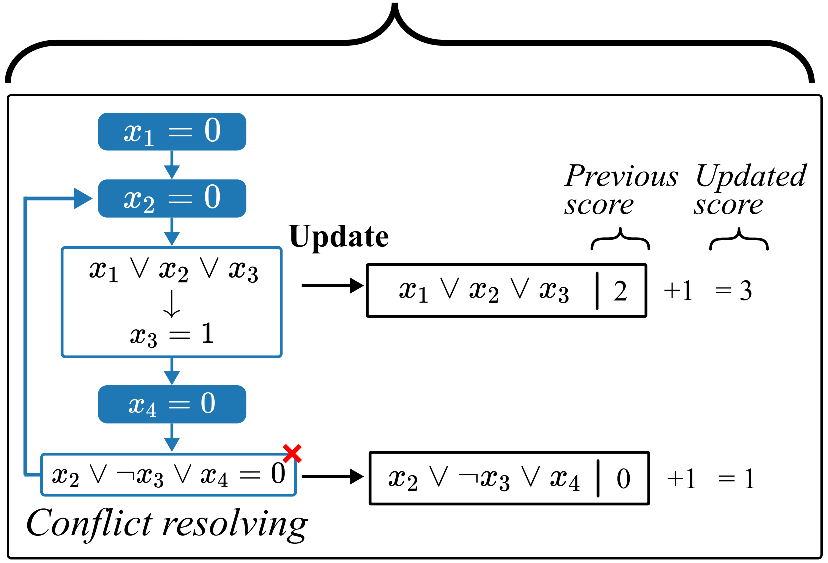
Outline of Presentation

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

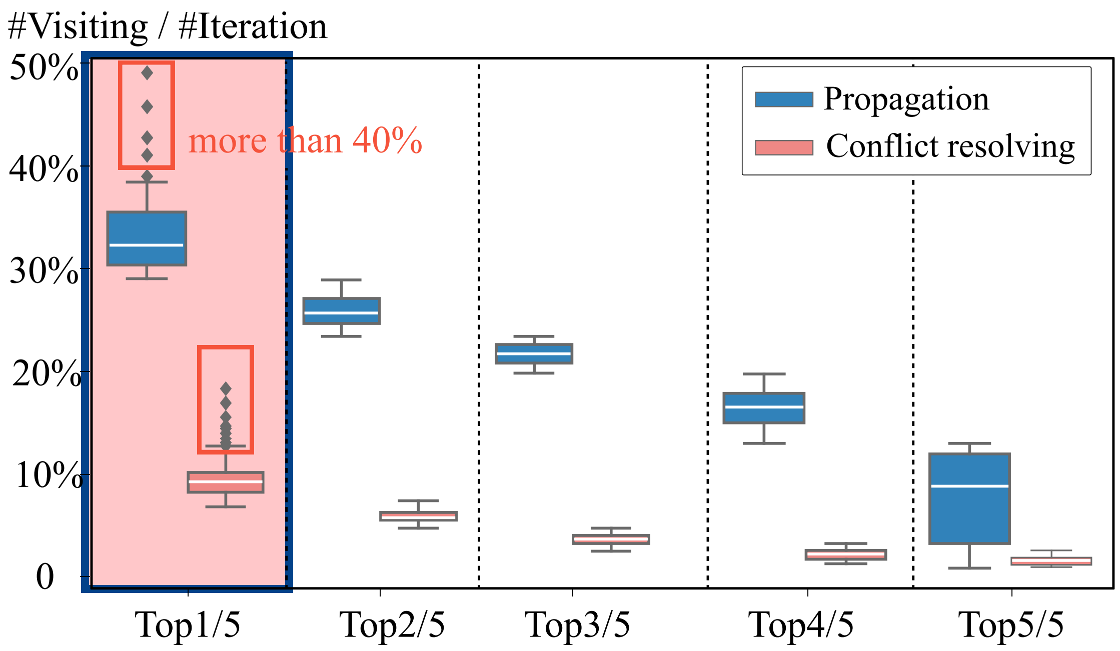
HyQSAT Frontend: Identify Difficult Clauses

☆ **Clause visiting frequency = Difficulty**

Predicting visiting frequencies

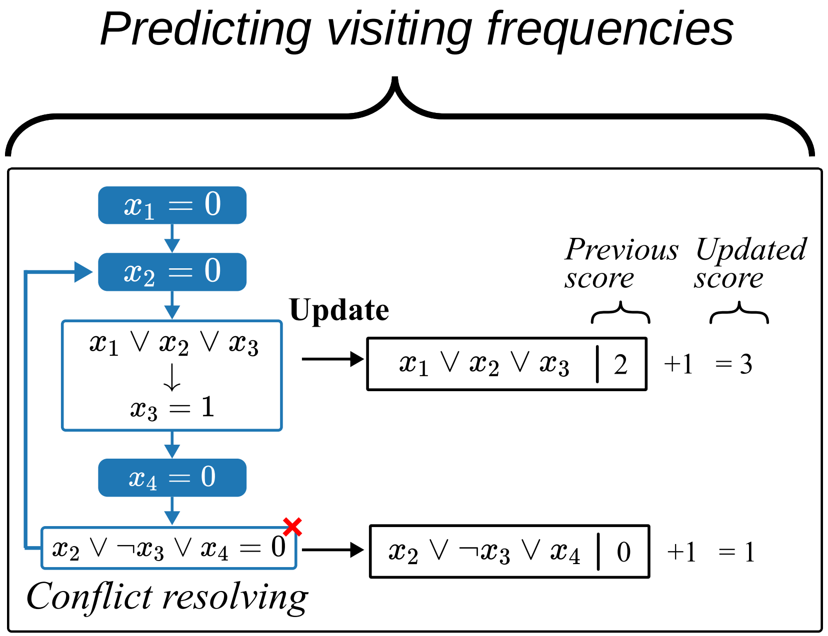


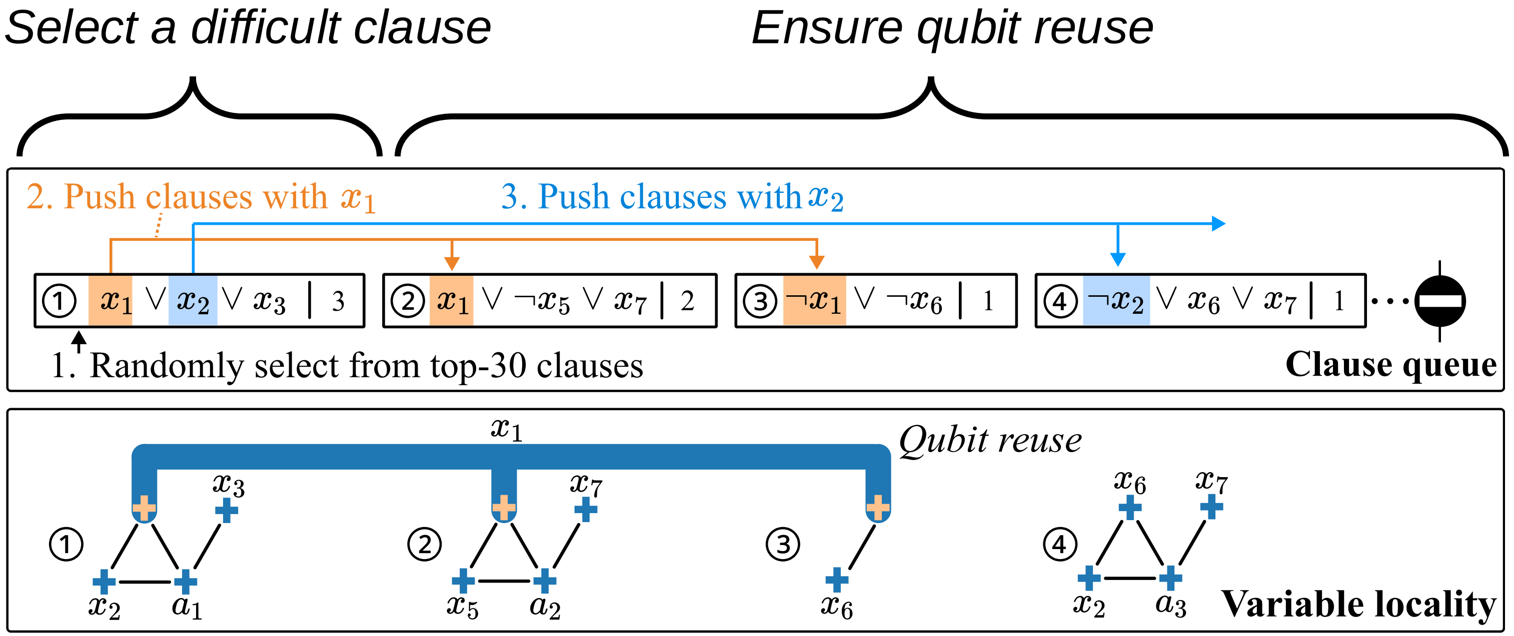
Define activity scores



Difficult clauses

HyQSAT Frontend: Identify Difficult Clauses





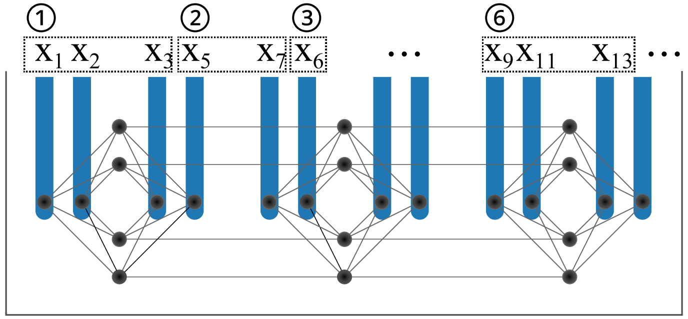
Define activity scores

Apply breadth-first search among clauses with same variables

HyQSAT Frontend: Fast Embedding

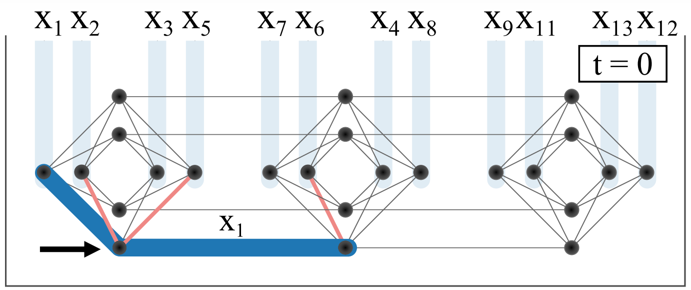
Step 1: Allocate variables to qubits of **vertical lines** according to their order in the clause queue

Clause:	① $x_1 \vee x_2 \vee x_3$	② $x_1 \vee \neg x_5 \vee x_7$	③ $\neg x_1 \vee \neg x_6$	…	⑥ $x_9 \vee x_{11} \vee x_{13}$	……
Problem graph:	x_1, x_3, x_2, a_1	x_1, x_7, x_5, a_2	x_6, x_1	…	x_9, x_{13}, x_{11}, a_5	……
Connection requirement list:	x_1:{x_2} a_1:{x_1,x_2, x_3}	x_1:{x_2,x_5} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}	x_1:{x_2,x_5,x_6} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}	…	x_1:{x_2,x_5,x_6} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7} … a_5:{x_9,x_{11},x_{13}}	

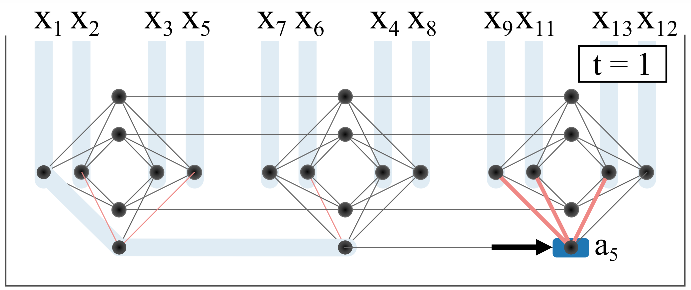


gathered for allocation together

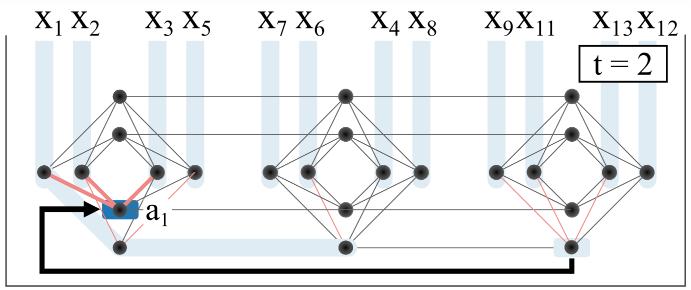
Step 2: Allocate variables to qubits of **horizontal lines**



x_1 :{x_2, x_5, x_6}



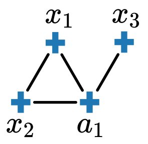
a_5 :{x_9, x_{11}, x_{13}}



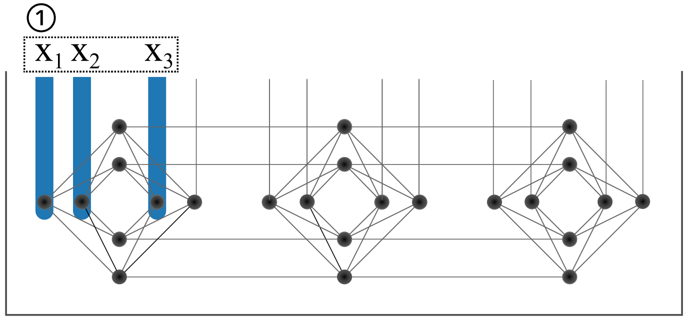
a_1 :{x_1, x_2, x_3}

HyQSAT Frontend: Fast Embedding

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

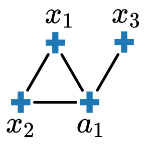
Clause:	① $x_1 \vee x_2 \vee x_3$
Problem graph:	
Connection requirement list:	x_1:{x_2} a_1:{x_1,x_2, x_3}

gathered for allocation together

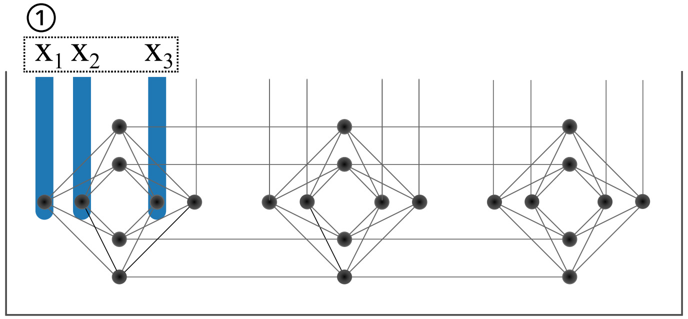


HyQSAT Frontend: Fast Embedding

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

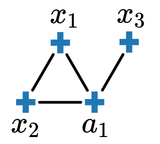
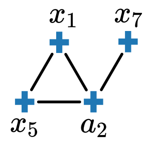
Clause:	① $x_1 \vee x_2 \vee x_3$
Problem graph:	
Connection requirement list:	x_1:{x_2} a_1:{x_1,x_2, x_3}

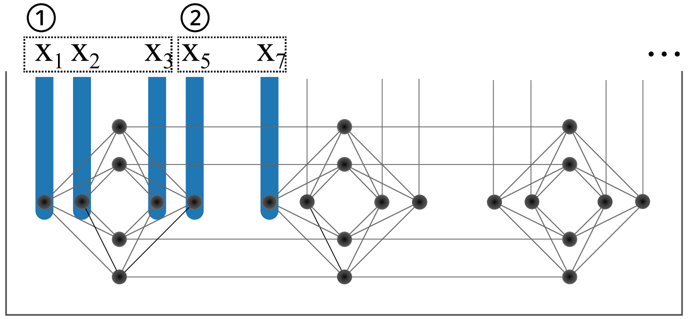
gathered for allocation together



HyQSAT Frontend: Fast Embedding

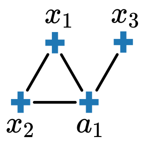
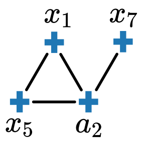
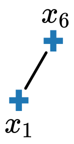
Step 1: Allocate variables to qubits of **vertical lines** according to their order in the clause queue

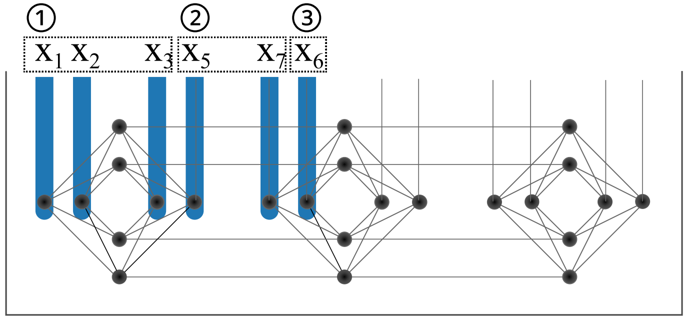
Clause:	① $x_1 \vee x_2 \vee x_3$	② $x_1 \vee \neg x_5 \vee x_7$
Problem graph:		
Connection requirement list:	x_1:{x_2} a_1:{x_1,x_2, x_3}	x_1:{x_2,x_5} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}



HyQSAT Frontend: Fast Embedding

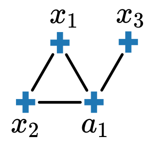
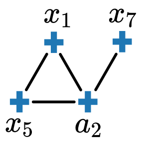
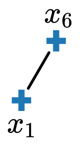
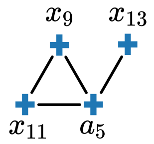
Step 1: Allocate variables to qubits of **vertical lines** according to their order in the clause queue

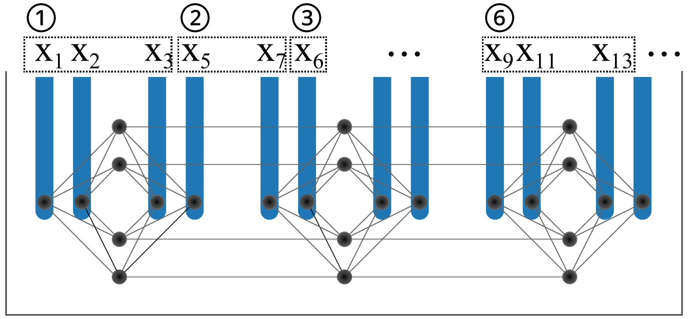
Clause:	① $x_1 \lor x_2 \lor x_3$	② $x_1 \lor \neg x_5 \lor x_7$	③ $\neg x_1 \lor \neg x_6$
Problem graph:			
Connection requirement list:	x_1:{x_2} a_1:{x_1,x_2, x_3}	x_1:{x_2,x_5} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}	x_1:{x_2,x_5,x_6} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}



HyQSAT Frontend: Fast Embedding

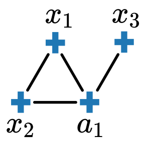
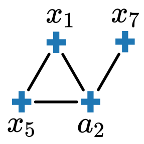
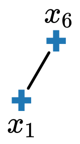
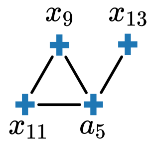
Step 1: Allocate variables to qubits of **vertical lines** according to their order in the clause queue

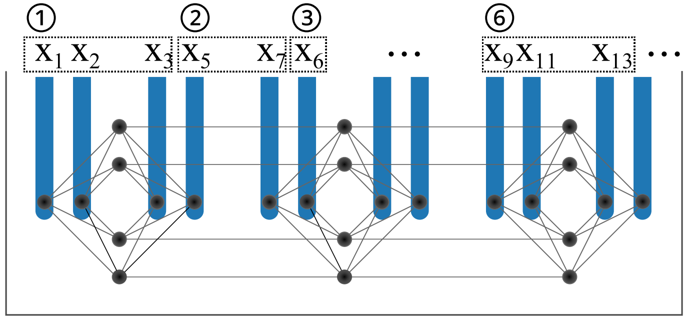
Clause:	① $x_1 \vee x_2 \vee x_3$	② $x_1 \vee \neg x_5 \vee x_7$	③ $\neg x_1 \vee \neg x_6$	…	⑥ $x_9 \vee x_{11} \vee x_{13}$	……
Problem graph:						
Connection requirement list:	x_1:{x_2} a_1:{x_1,x_2, x_3}	x_1:{x_2,x_5} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}	x_1:{x_2,x_5,x_6} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}		x_1:{x_2,x_5,x_6} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7} … a_5:{x_9,x_{11},x_{13}}	



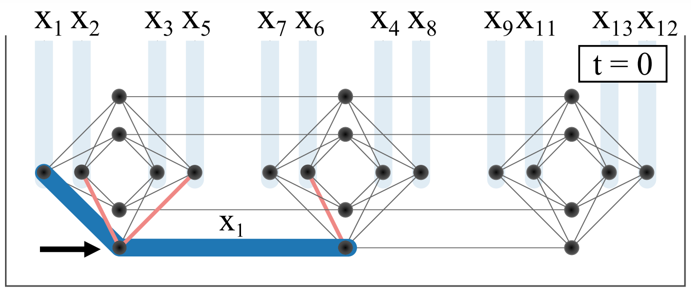
HyQSAT Frontend: Fast Embedding

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

Clause:	① $x_1 \vee x_2 \vee x_3$	② $x_1 \vee \neg x_5 \vee x_7$	③ $\neg x_1 \vee \neg x_6$	…	⑥ $x_9 \vee x_{11} \vee x_{13}$	……
Problem graph:						
Connection requirement list:	x_1:{x_2} a_1:{x_1,x_2, x_3}	x_1:{x_2,x_5} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}	x_1:{x_2,x_5,x_6} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}		x_1:{x_2,x_5,x_6} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7} … a_5:{x_9,x_{11},x_{13}}	



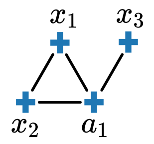
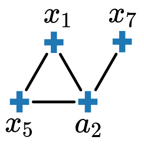
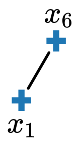
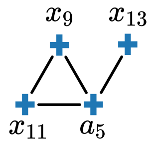
Step 2: Allocate variables to qubits of **horizontal lines**

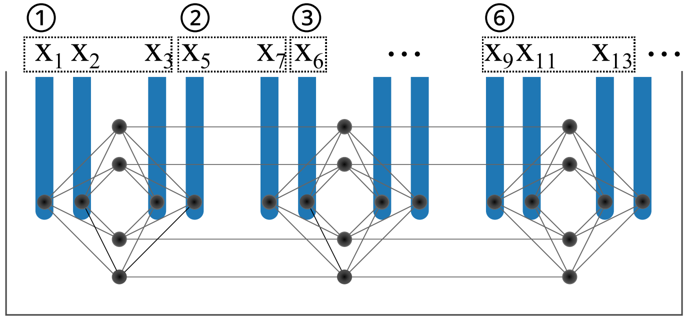


x_1 :{x_2, x_5, x_6}

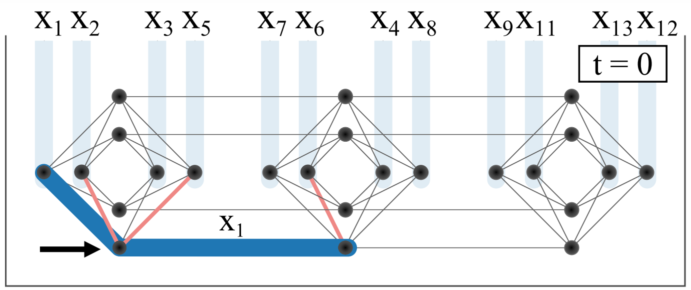
HyQSAT Frontend: Fast Embedding

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

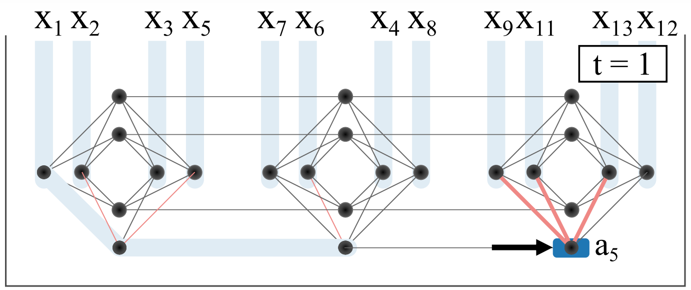
Clause:	① $x_1 \vee x_2 \vee x_3$	② $x_1 \vee \neg x_5 \vee x_7$	③ $\neg x_1 \vee \neg x_6$	…	⑥ $x_9 \vee x_{11} \vee x_{13}$	……
Problem graph:				…		……
Connection requirement list:	x_1:{x_2} a_1:{x_1,x_2, x_3}	x_1:{x_2,x_5} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}	x_1:{x_2,x_5,x_6} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}	…	x_1:{x_2,x_5,x_6} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7} … a_5:{x_9,x_{11},x_{13}}	



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



x_1 :{x_2, x_5, x_6}

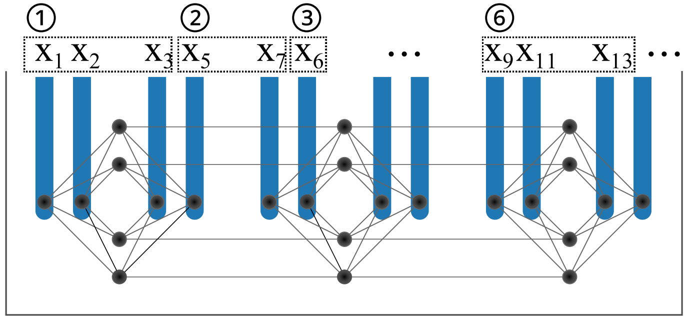


a_5 :{x_9, x_{11}, x_{13}}

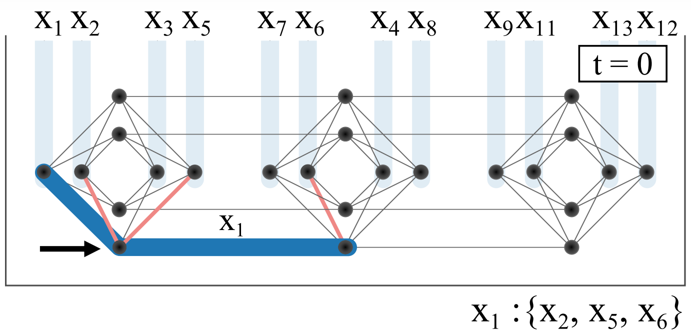
HyQSAT Frontend: Fast Embedding

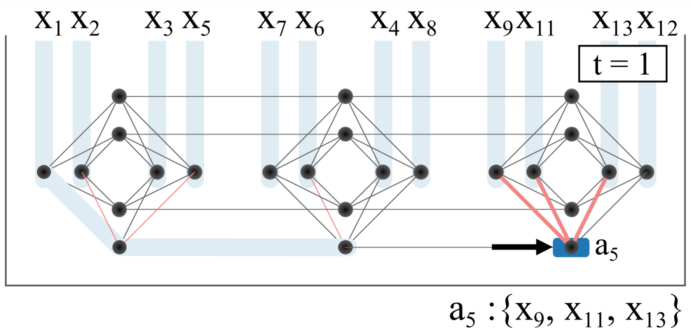
Step 1: Allocate variables to qubits of **vertical lines** according to their order in the clause queue

Clause:	① $x_1 \vee x_2 \vee x_3$	② $x_1 \vee \neg x_5 \vee x_7$	③ $\neg x_1 \vee \neg x_6$	…	⑥ $x_9 \vee x_{11} \vee x_{13}$	……
Problem graph:	x_1, x_3, x_2, a_1	x_1, x_7, x_5, a_2	x_6, x_1	…	x_9, x_{13}, x_{11}, a_5	……
Connection requirement list:	x_1:{x_2} a_1:{x_1,x_2, x_3}	x_1:{x_2,x_5} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}	x_1:{x_2,x_5,x_6} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7}		x_1:{x_2,x_5,x_6} a_1:{x_1,x_2,x_3} a_2:{x_1,x_5,x_7} … a_5:{x_9,x_{11},x_{13}}	



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





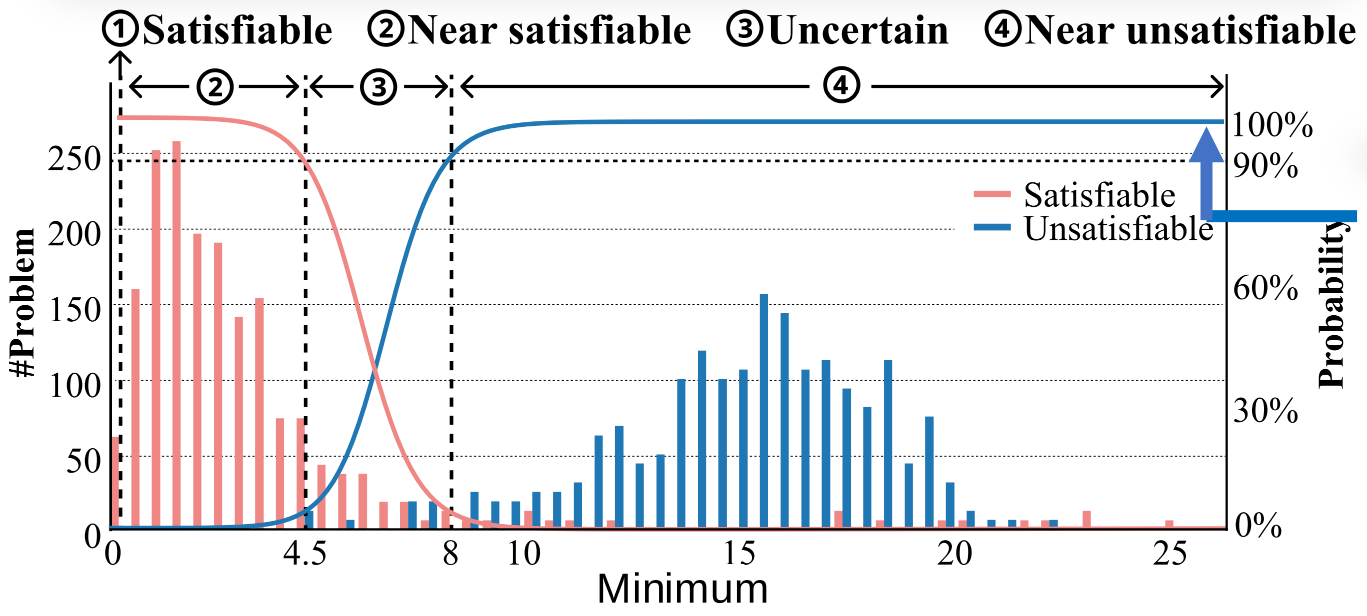
Outline of Presentation

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

HyQSAT Backend: Satisfaction Probability Classification

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, +∞].

HyQSAT Backend: Guiding CDCL Search

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				

Contribute to no acceleration for the classic CDCL.

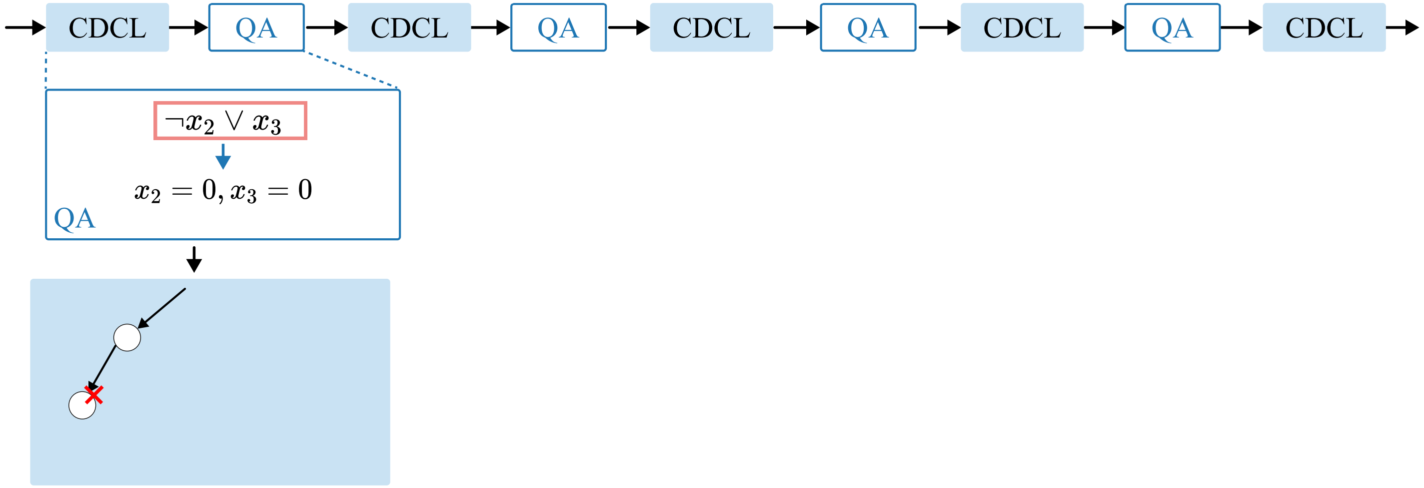
HyQSAT Backend: Guiding CDCL Search

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.

<table>
<tr><th></th><th>Satisfiable</th><th>Near satisfiable</th><th>Uncertain</th><th>Near unsatisfiable</th></tr>
<tr><td>All embedded</td><td>Strategy 1</td><td rowspan="2" colspan="1">Strategy 2</td><td rowspan="2">Strategy 3</td><td rowspan="2">Strategy 4</td></tr>
<tr><td>Not all embedded</td><td></td></tr>
</table>

Contribute to no acceleration for the classic CDCL.

Feedback strategy 1, 2, 4



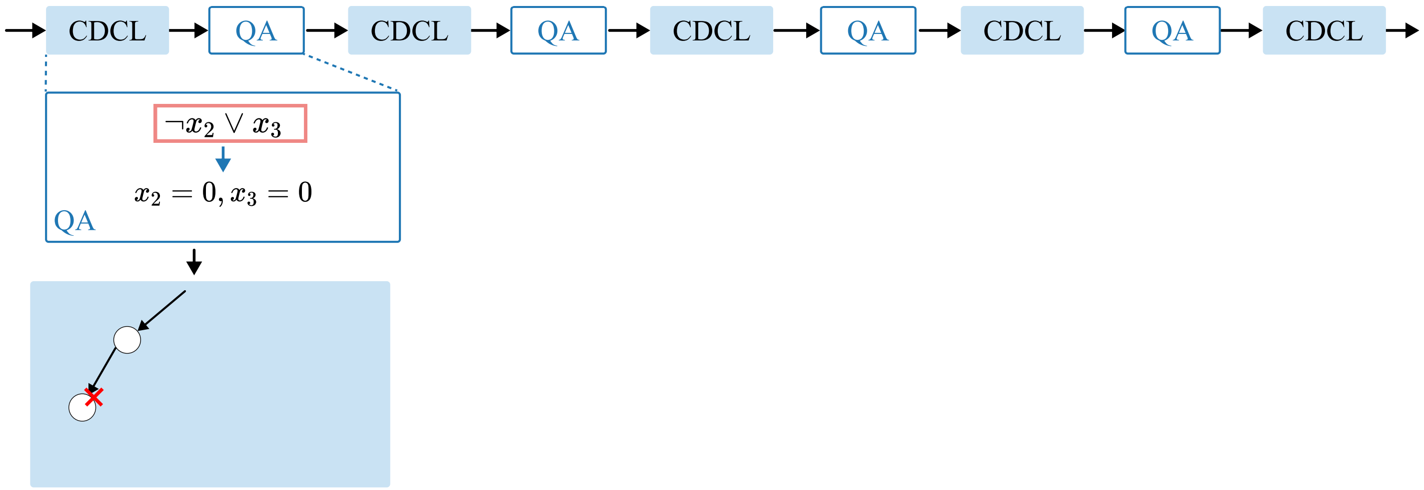
Strategy 1

HyQSAT Backend: Guiding CDCL Search

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.

<table>
<tr><th></th><th>Satisfiable</th><th>Near satisfiable</th><th>Uncertain</th><th>Near unsatisfiable</th></tr>
<tr><td>All embedded</td><td>Strategy 1</td><td rowspan="2">Strategy 2</td><td rowspan="2">Strategy 3</td><td rowspan="2">Strategy 4</td></tr>
<tr><td>Not all embedded</td><td></td></tr>
</table>

Feedback strategy 1, 2, 4



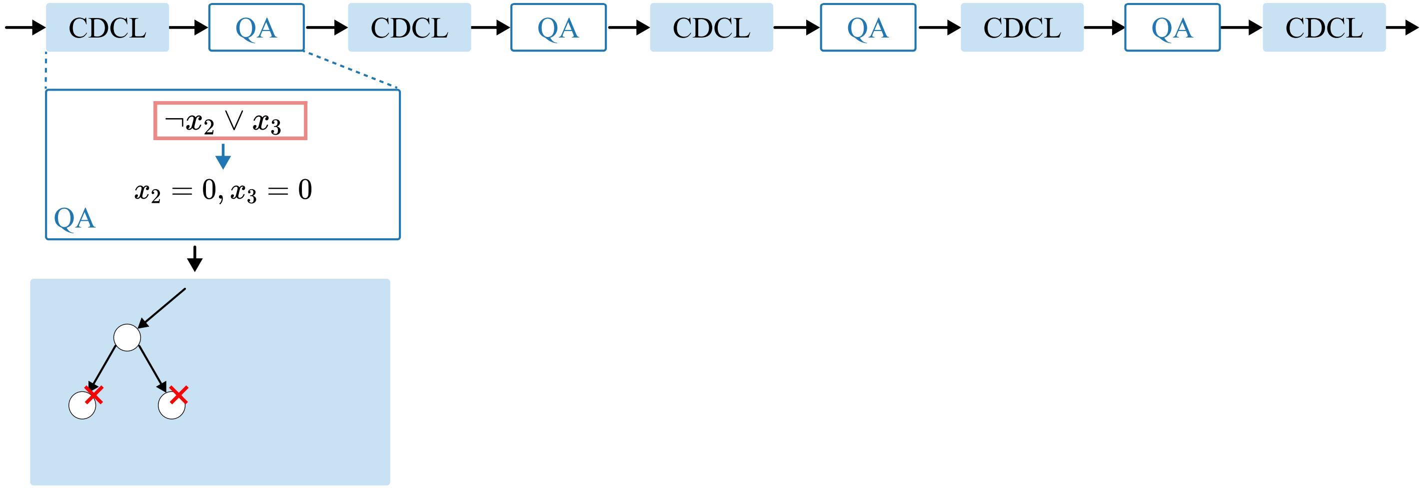
Strategy 1

HyQSAT Backend: Guiding CDCL Search

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.

<table>
<tr><th></th><th>Satisfiable</th><th>Near satisfiable</th><th>Uncertain</th><th>Near unsatisfiable</th></tr>
<tr><td>All embedded</td><td>Strategy 1</td><td rowspan="2">Strategy 2</td><td rowspan="2">Strategy 3</td><td rowspan="2">Strategy 4</td></tr>
<tr><td>Not all embedded</td><td></td></tr>
</table>

Feedback strategy 1, 2, 4



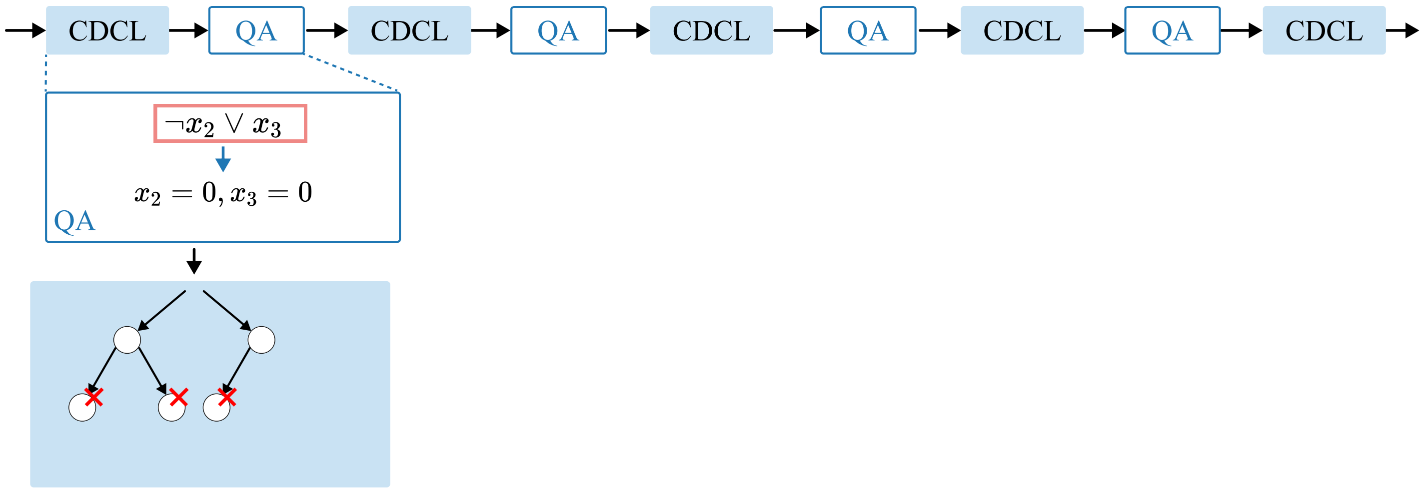
Strategy 1

HyQSAT Backend: Guiding CDCL Search

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



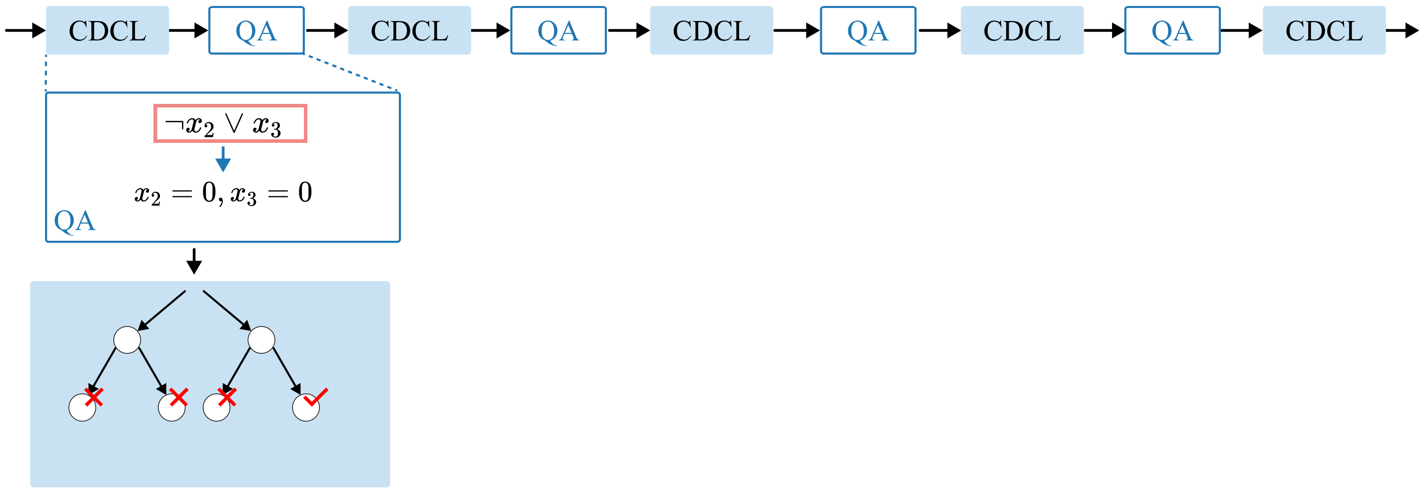
Strategy 1

HyQSAT Backend: Guiding CDCL Search

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



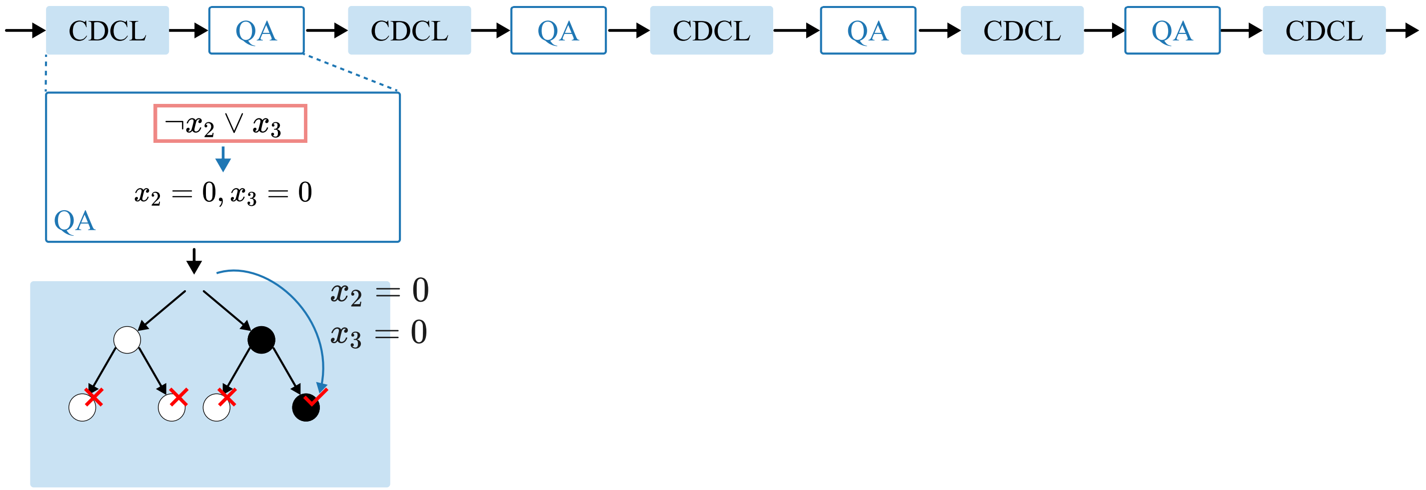
Strategy 1

HyQSAT Backend: Guiding CDCL Search

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.

<table>
<tr><th></th><th>Satisfiable</th><th>Near satisfiable</th><th>Uncertain</th><th>Near unsatisfiable</th></tr>
<tr><td>All embedded</td><td>Strategy 1</td><td rowspan="2" colspan="1">Strategy 2</td><td rowspan="2">Strategy 3</td><td rowspan="2">Strategy 4</td></tr>
<tr><td>Not all embedded</td><td></td></tr>
</table>

Feedback strategy 1, 2, 4



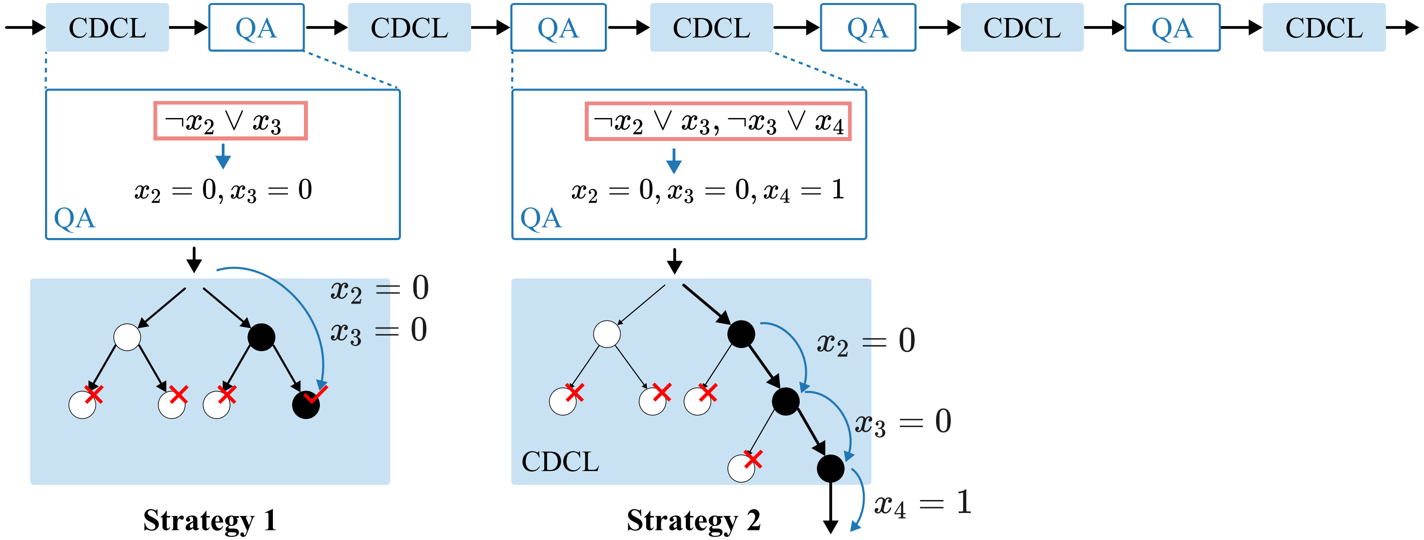
Strategy 1

HyQSAT Backend: Guiding CDCL Search

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.

<table>
<tr><th></th><th>Satisfiable</th><th>Near satisfiable</th><th>Uncertain</th><th>Near unsatisfiable</th></tr>
<tr><td>All embedded</td><td>Strategy 1</td><td rowspan="2">Strategy 2</td><td rowspan="2">Strategy 3</td><td rowspan="2">Strategy 4</td></tr>
<tr><td>Not all embedded</td><td></td></tr>
</table>

Feedback strategy 1, 2, 4

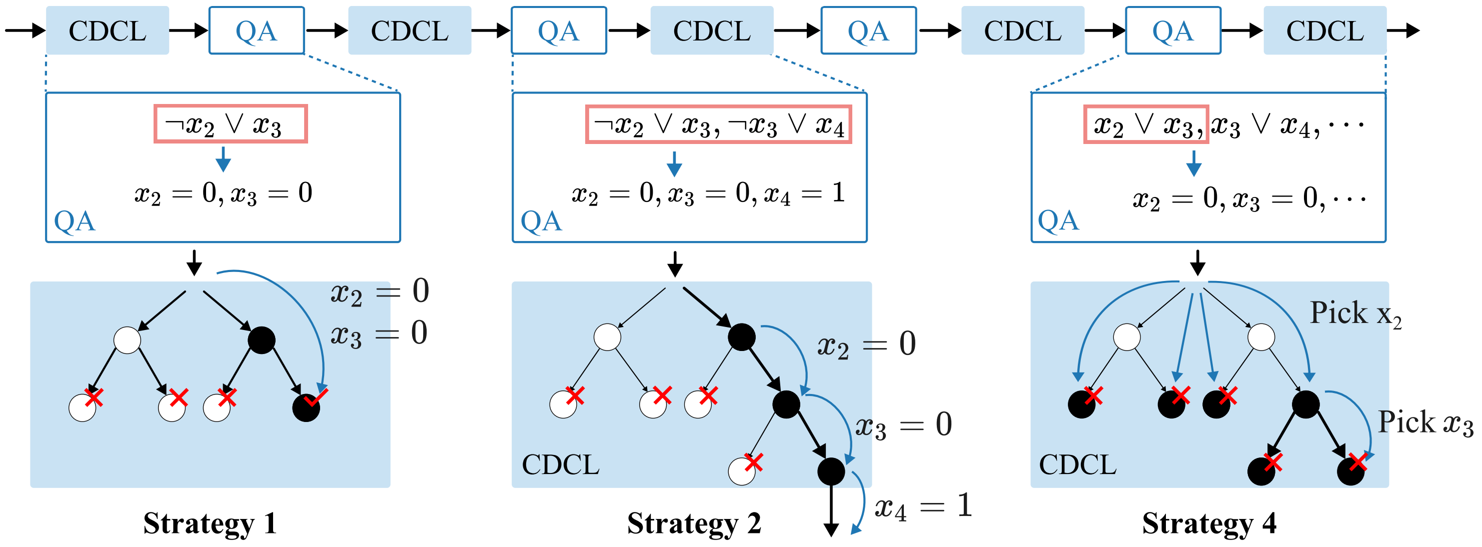


HyQSAT Backend: Guiding CDCL Search

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				

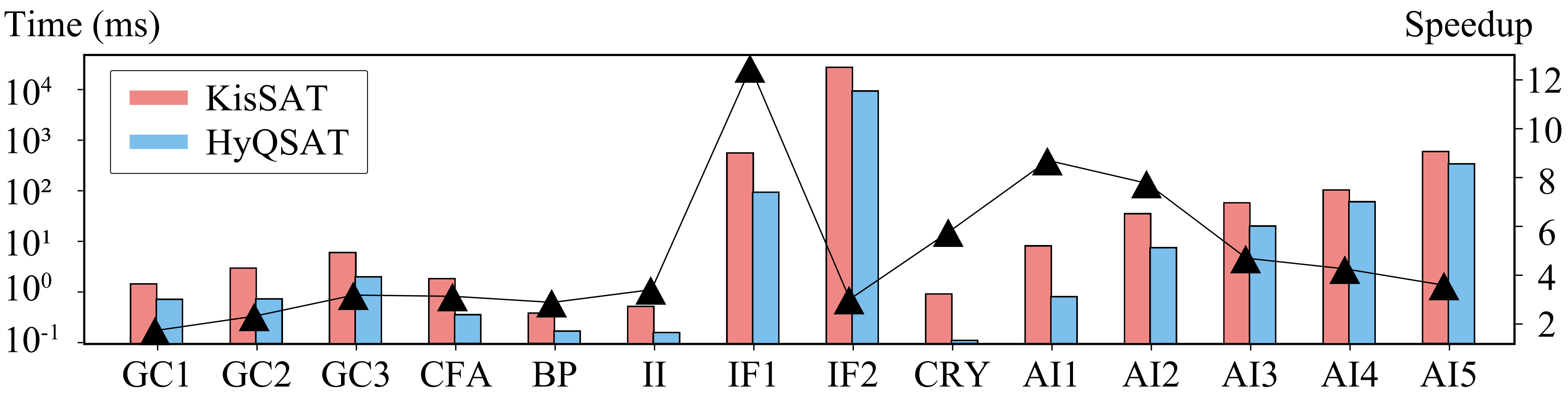
Feedback strategy 1, 2, 4



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**

API of HyQSAT

File:
- JanusQ/examples/ipynb/5_1_solve_sat_domain_problem.ipynb
- https://janusq.github.io/tutorials/demo/5_1_solve_sat_domain_problem

configure solver

solve problem

```
from janusq.hyqsat import solve_by_hyqsat

# input cnf flie
file_path = "./data/cnf_examples/test/uf100-01.cnf"

# if verbose
verbose = True
# limit the cpu time (s). 0 means infinite
cpu_lim = 0
# limit the memory. 0 means infinite
mem_lim = 0

result_janus = solve_by_hyqsat(file_path, verb=verbose,
cpu_lim=cpu_lim, mem_lim=mem_lim, use_realQC=True)
```

Use real quantum hardware
(Require API key of Dwave)

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,
    'simulation time': 1.07241,
    'quantum success number': 9,
    'quantum conflict number': 13,
    'quantum one time solve number': 0,
    ‘is satisfiable': True,
}
```

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*