HPCA 2025 Tutorial

Topic 5. Choco-Q: Commute Hamiltonian-based QAOA for Constrained Binary Optimization

JanusQ Cloud

Speaker: Liqiang Lu

College of Computer Science and Technology
Zhejiang University (ZJU)

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

Outline of Presentation

- **Background and challenges**
- Overview of Choco-Q
- Choco-Q characterization and calibration
- Experiment
- API of Choco-Q

Application of Constrained Binary Optimization

Constrained Binary Optimization

objective

$$\max_{x_1,x_2,x_3} \ 3x_1x_2 - 2x_2 + x_3$$

constriants

$$\begin{aligned} s.t. \quad & x_1 + x_2 \qquad = 1 \\ & x_1 \qquad - x_3 = 0 \\ & x_j \in \{0,1\}, j = 1,2,3 \end{aligned}$$

Facility location

Routing planning

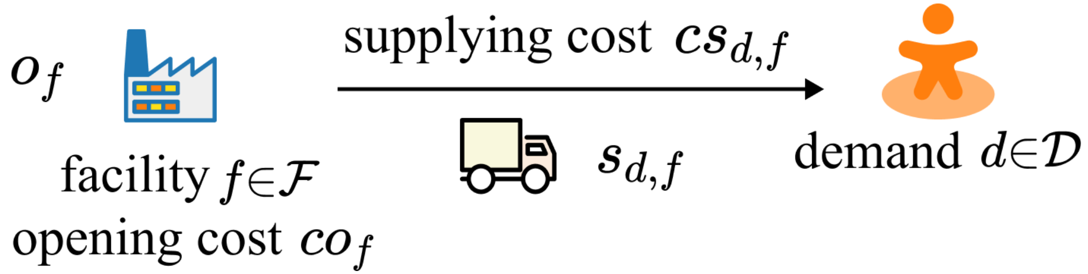
project scheduling

portfolio optimization

Example: Facility location Problem

Variable Definition

There are facilities to supply the demands.
The costs including opening the facilities and supply demands.



Whether to open facility

$$o_f = \begin{cases} 1, & \text{open facility } f \\ 0, & \text{otherwise} \end{cases}$$

How to assign demands

$$s_{d,f} = \begin{cases} 1, & \text{facility } f \text{ supplies demand } d \\ 0, & \text{otherwise} \end{cases}$$

Constraints

Each demand can only be assigned to one facility

$$\sum_{f \in \mathcal{F}} s_{d,f} = 1, \forall d \in \mathcal{D}$$

An unopened facility cannot supply any demand

$$s_{d,f} \leq o_f, \forall d \in \mathcal{D}, \forall f \in \mathcal{F}$$

objective

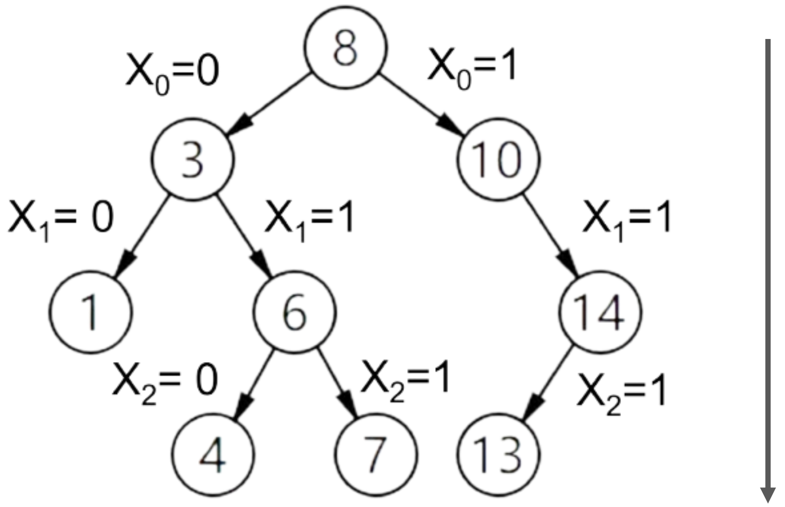
Minimize the cost of opening facilities and supplying demands

$$\min_{s_{d,f}, o_f} \underbrace{\sum_{d \in \mathcal{D}} \sum_{f \in \mathcal{F}} cs_{d,f} \cdot s_{d,f}}_{\text{supplying cost}} + \underbrace{\sum_{f \in \mathcal{F}} co_f \cdot o_f}_{\text{opening cost}}$$

Classical Solving Methods

Exact algorithms

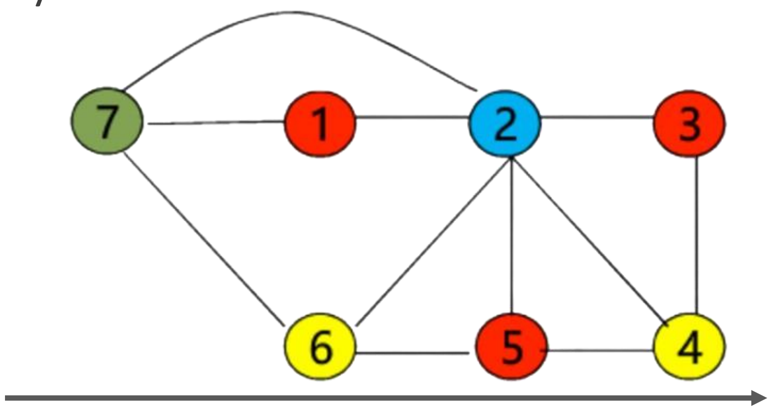
branch and bound method.



Construct a binary tree based on the variable values.

Approximation algorithms

greedy method



Color the vertices sequentially, ensuring that adjacent vertices are assigned different colors.

Solving constraint optimization problems of any form, with a worst-case complexity of 2^n.

Need artificially designed for particular problems.

No guarantee for global optimality.

NP-Hard problem, and the classical algorithms for exact solutions have exponential complexity.

Quantum approximation optimization algorithm

Adiabatic Evolution

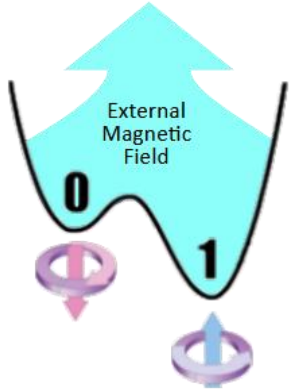
Driver Hamiltonian

Simple and easy to prepare

Adiabatic evolution

Objective Hamiltonian

The objective to be optimized



Quantum approximation optimization algorithm (QAOA)

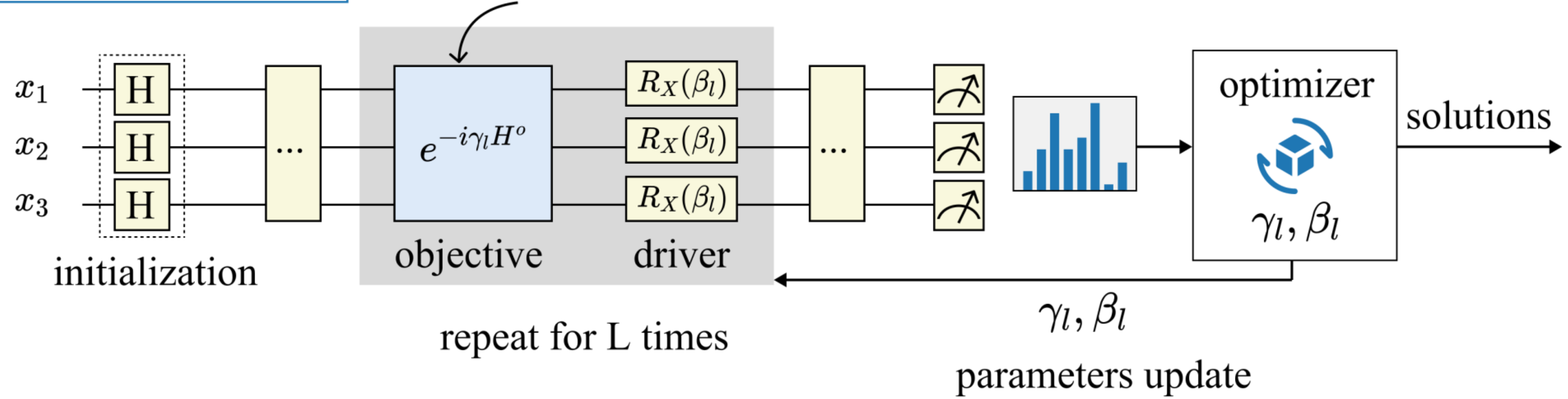
objective

$$\max_{x_1,x_2,x_3} 3x_1x_2 - 2x_2 + x_3$$

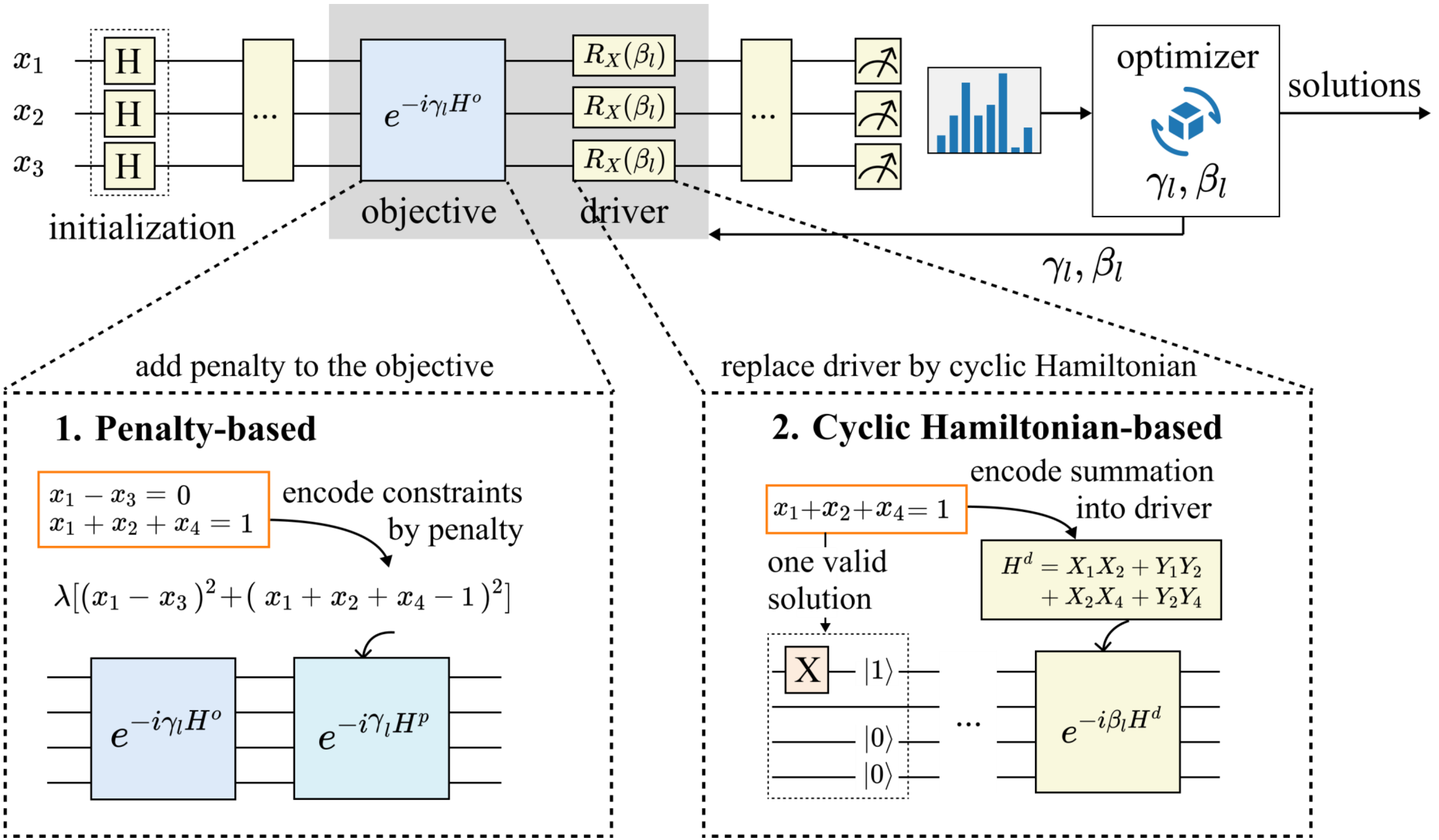
$$x_j \rightarrow \frac{I - Z_j}{2}$$

Construct H^o

$$H^o = 3\frac{I-Z_1}{2}\frac{I-Z_2}{2} - 2\frac{I-Z_2}{2} + \frac{I-Z_3}{2}$$

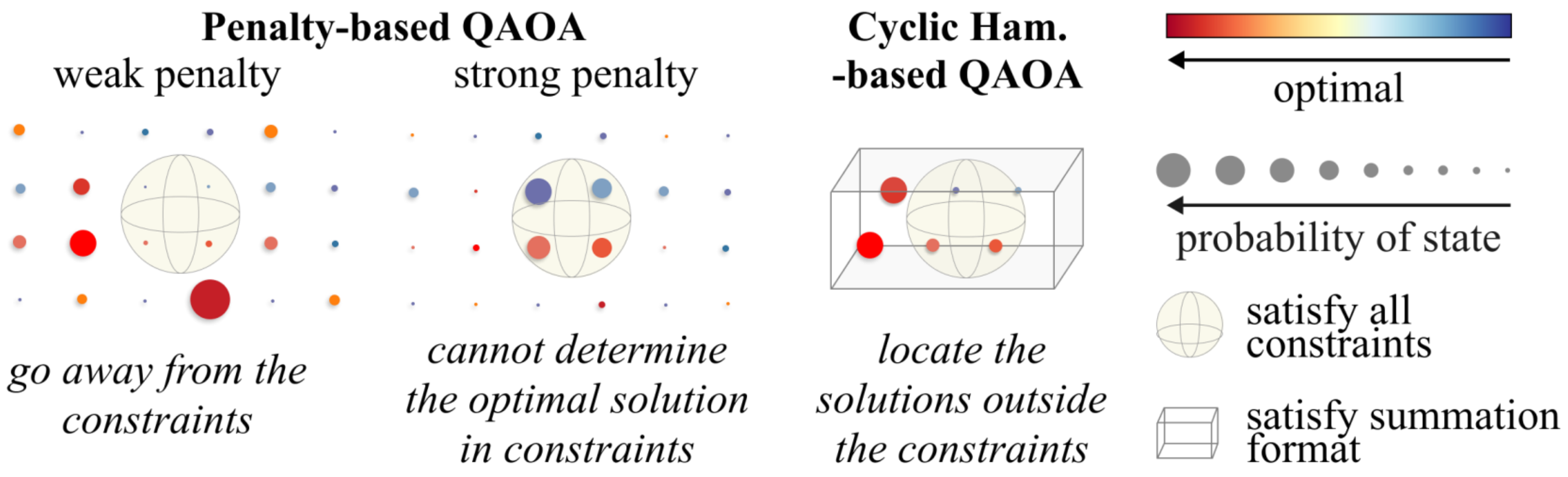


QAOA for constrained optimization



Limitations of Current Methods

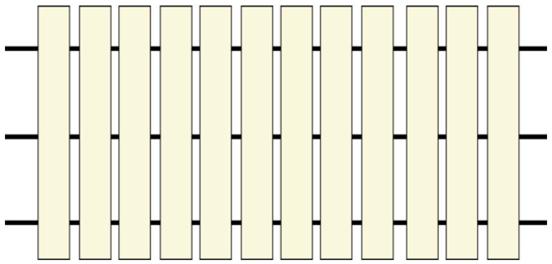
Constraint encoding	Penalty-term-based		Driver-Hamiltonian-based	
Methods	A. Verma et al. [44]	Red-QAOA [45]	Yoshioka et al. [47] *cyclic Hamilt.*	Choco-Q *commute Hamilt.*
Universality	soft const.	soft const.	hard const. only part of linear	hard const. arbitrary linear
In-constraints rate	0.03%	0.07%	0.67%	100%
Success rate	0.02%	0.03%	0.14%	67.1%
End-to-end latency	16.6s	16.7s	19.6s	7.07s



Limitations of Current Methods

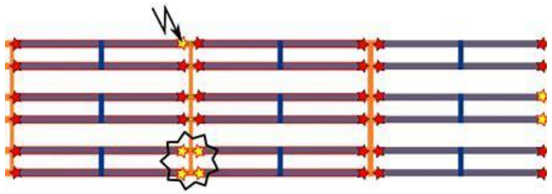
Challenges in real word computer

Large circuit depth



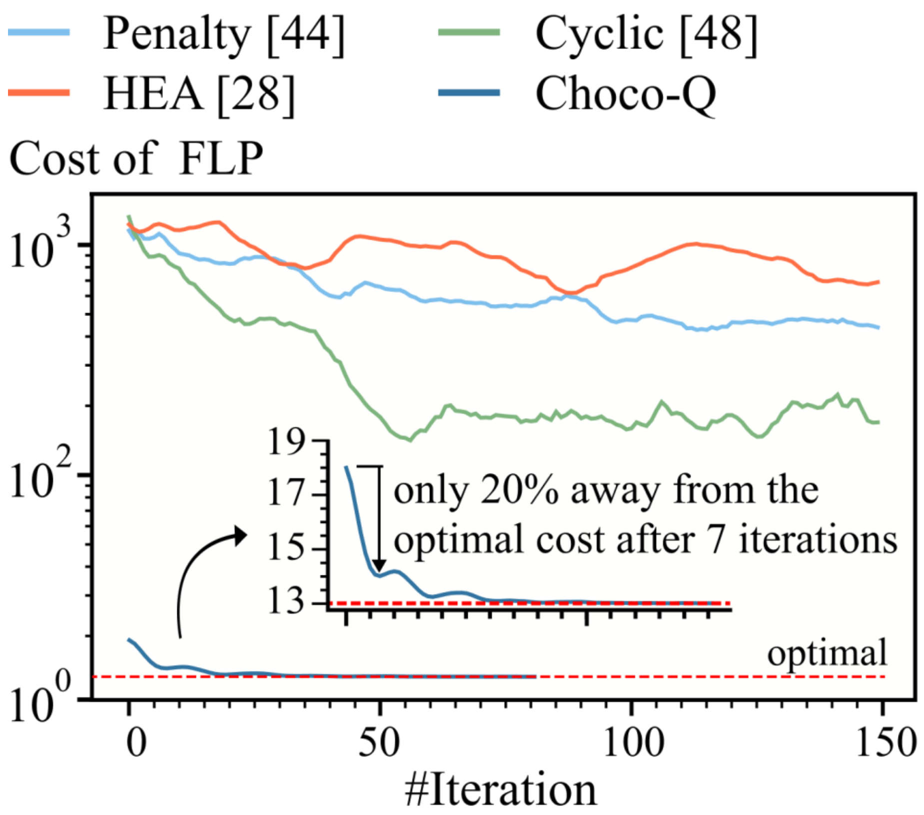
50 bits yielding more than 1040 layers of quantum gates

Real-world noise



The noise further exacerbates the difficulty of optimizing the parameters

experimental result



The optimization process is extremely unstable and difficult to converge to the optimal solution

Outline of Presentation

- Background and challenges
- **Overview of Choco-Q**
- Choco-Q characterization and calibration
- Experiment
- API of Choco-Q

Heisenberg picture

Heisenberg Picture

Variation

System Hamiltonian

$$\frac{dA}{dt} = \frac{i}{\hbar}[A, H]$$

Arbitrary operator

$$[A, H] = AH - HA$$

If A Commutes with H

$$[A, H] = 0$$

$$\frac{dA}{dt} = 0$$

The operator is unchanged

Heisenberg picture

ZHEJIANG UNIVERSITY

Heisenberg Picture

Apply to constrained binary optimization

Variation

System Hamiltonian

Object Hamiltonian + Driver HamiltonianH^d

Always commute

Expect to commute

$$\frac{dA}{dt} = \frac{i}{\hbar}[A, H]$$

Arbitrary operator

Constraints operator C

$$[A, H] = AH - HA$$

If A Commutes with H

$$[A, H] = 0$$

$$[C, H^d] = 0$$

$$\frac{dA}{dt} = 0$$

$$\frac{dC}{dt} = 0$$

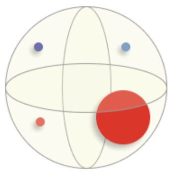
The operator is unchanged

Constraints are satisfied

Commute Hamiltonian formulation

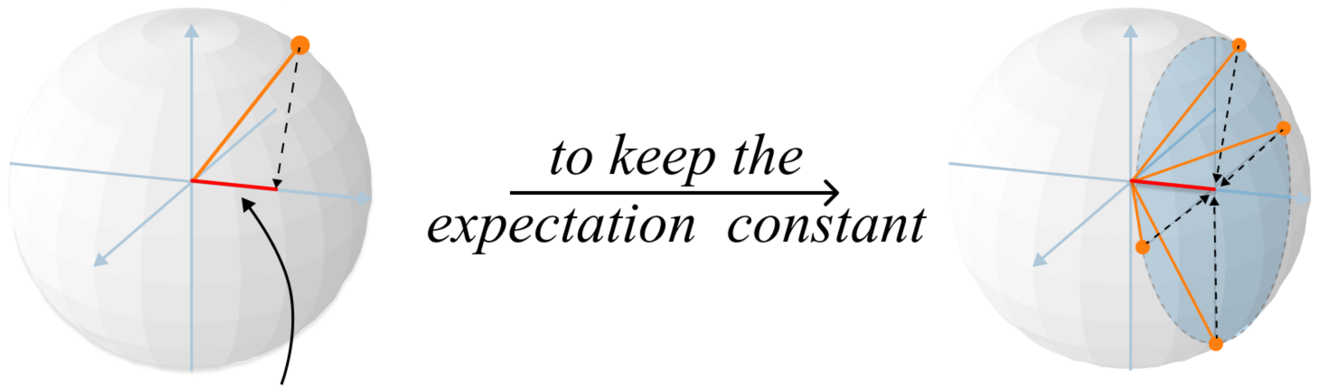
The Hamiltonian commutes with constraints operator

Commute Ham. -based Choco-Q



find the optimal solution in constraints

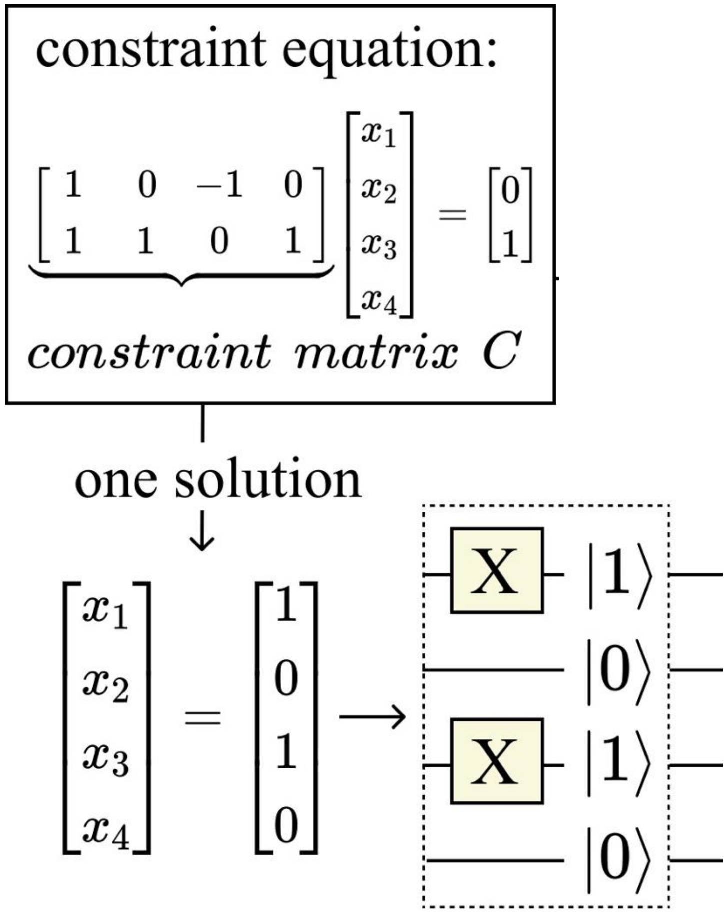
Encoding constraints via commute Hamitonain



expectation is the projection onto the constraints operator

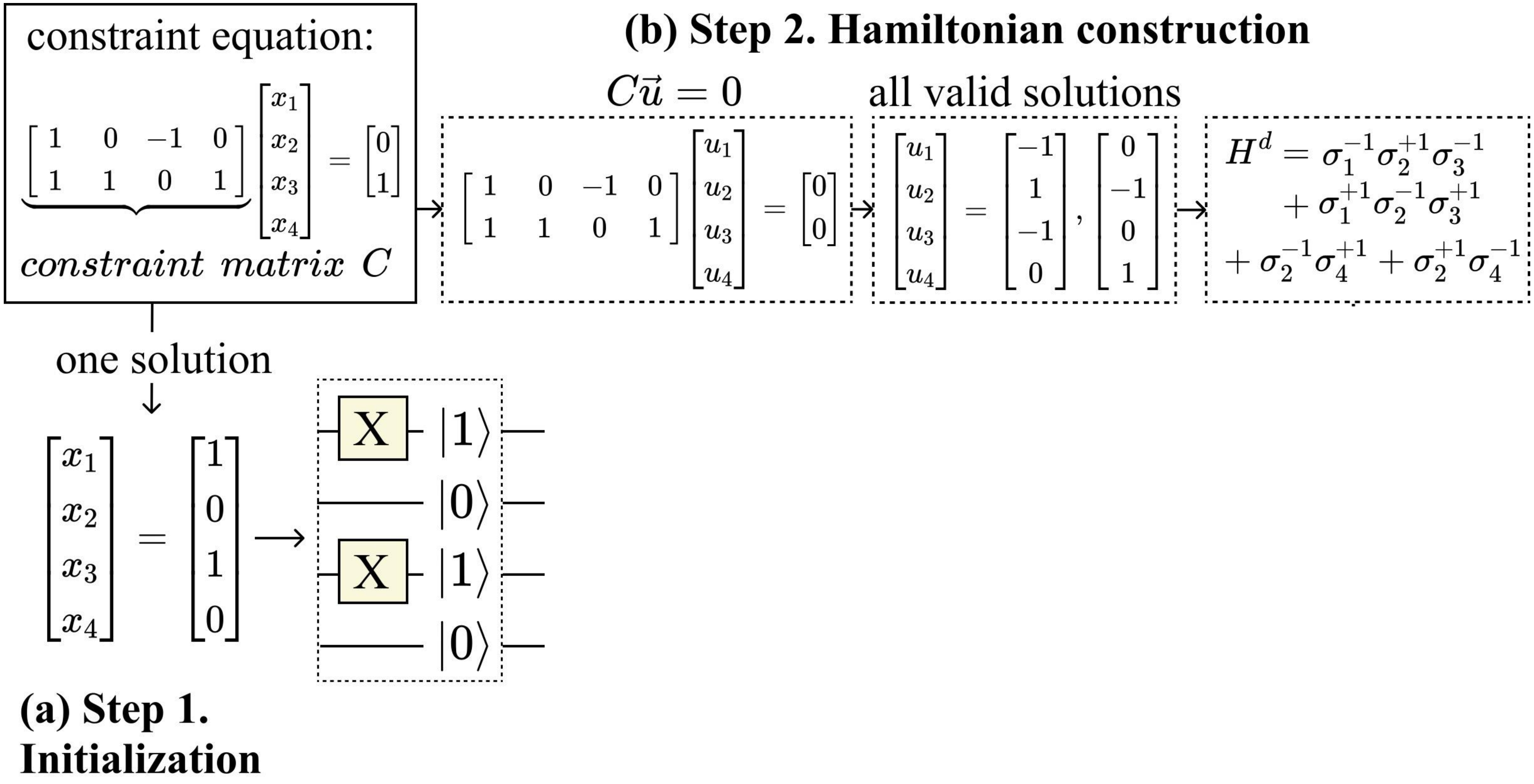
the state is perfectly constrained on the plane

Circuit Implementation

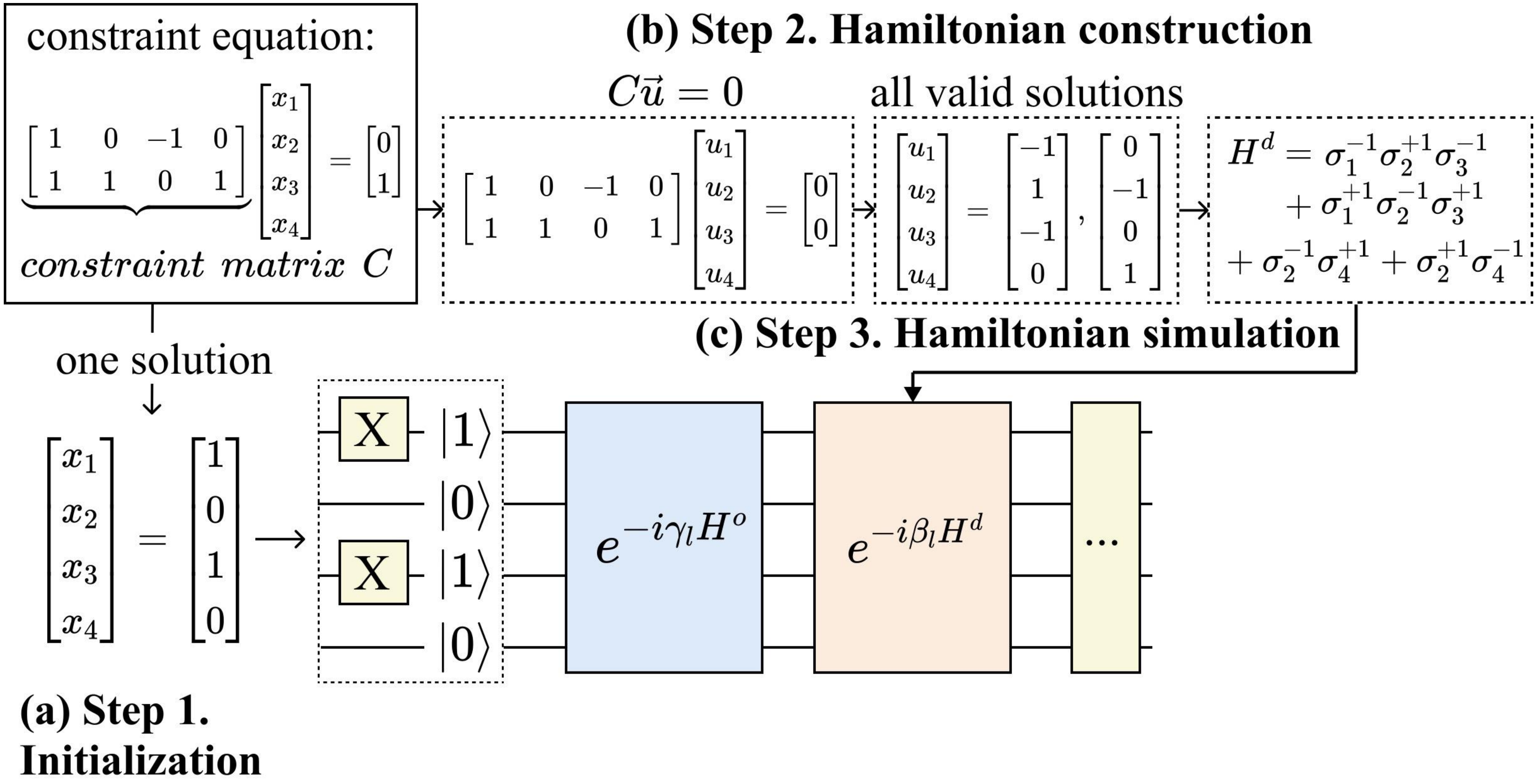


(a) Step 1. Initialization

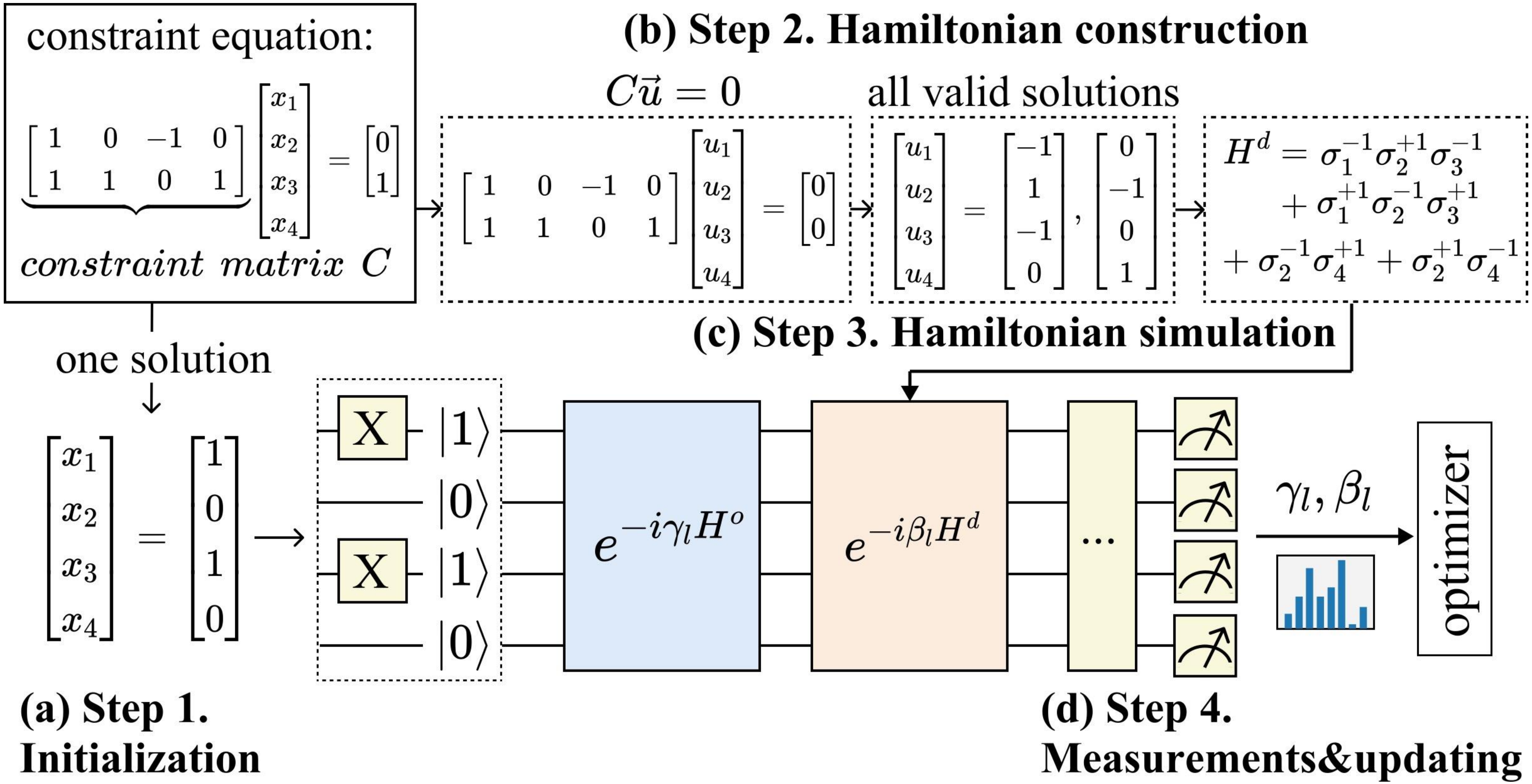
Circuit Implementation



Circuit Implementation



Circuit Implementation

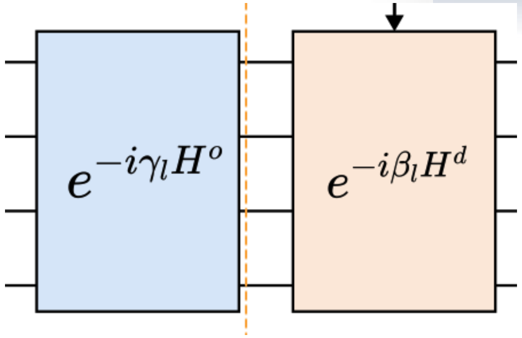


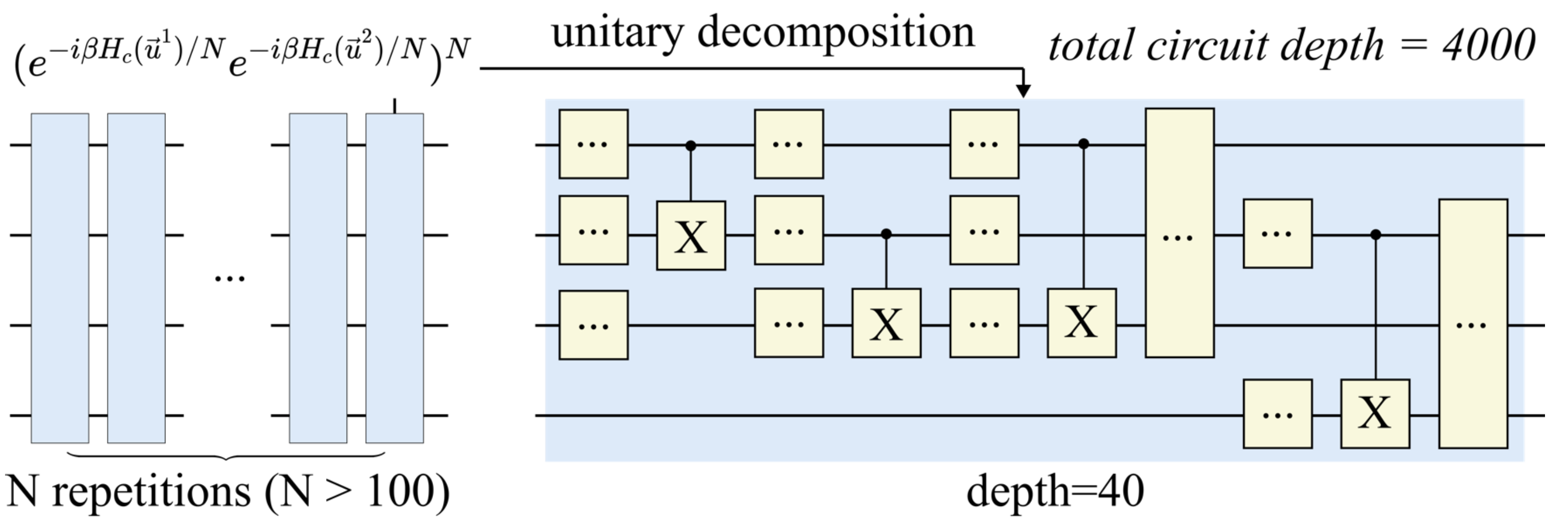
Outline of Presentation

- Background and challenges
- Overview of Choco-Q
- **Choco-Q Optimization**
- Experiment
- API of Choco-Q

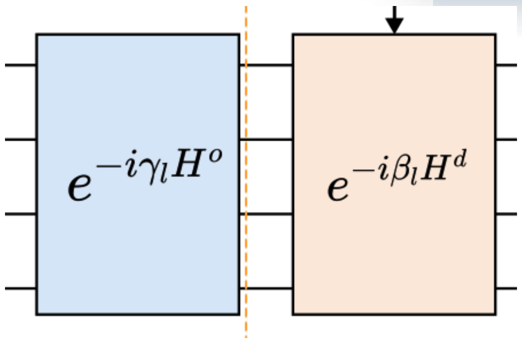
Hamiltonian Serialization

Prior method:
Trotter decomposition

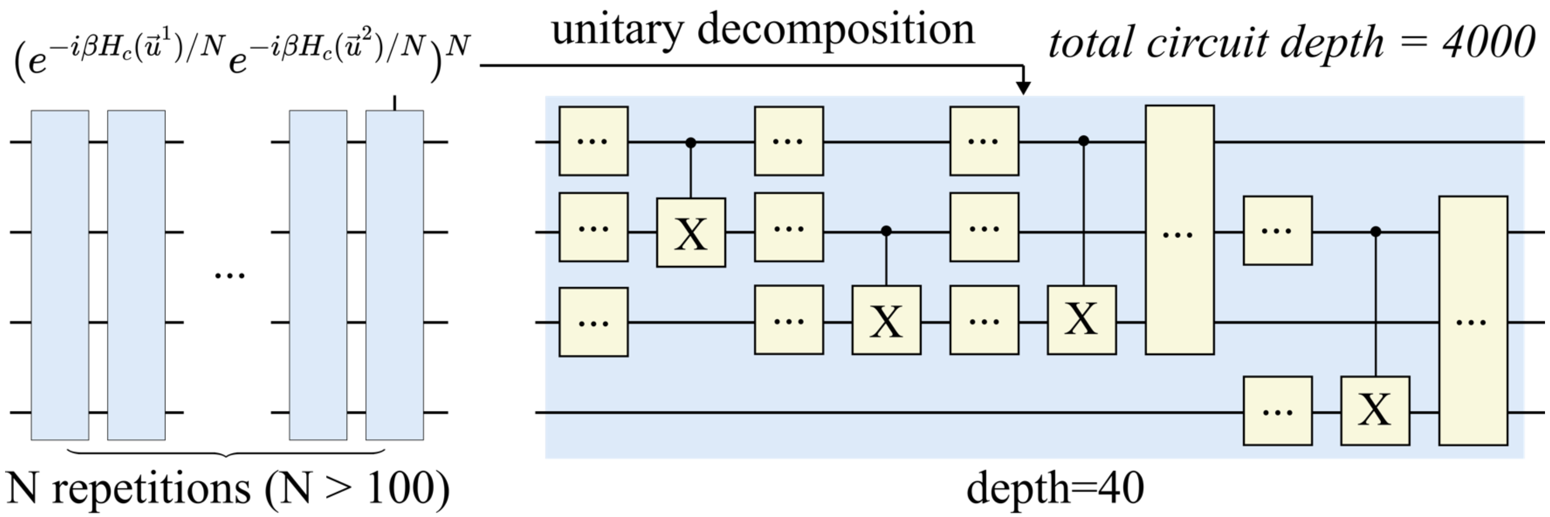




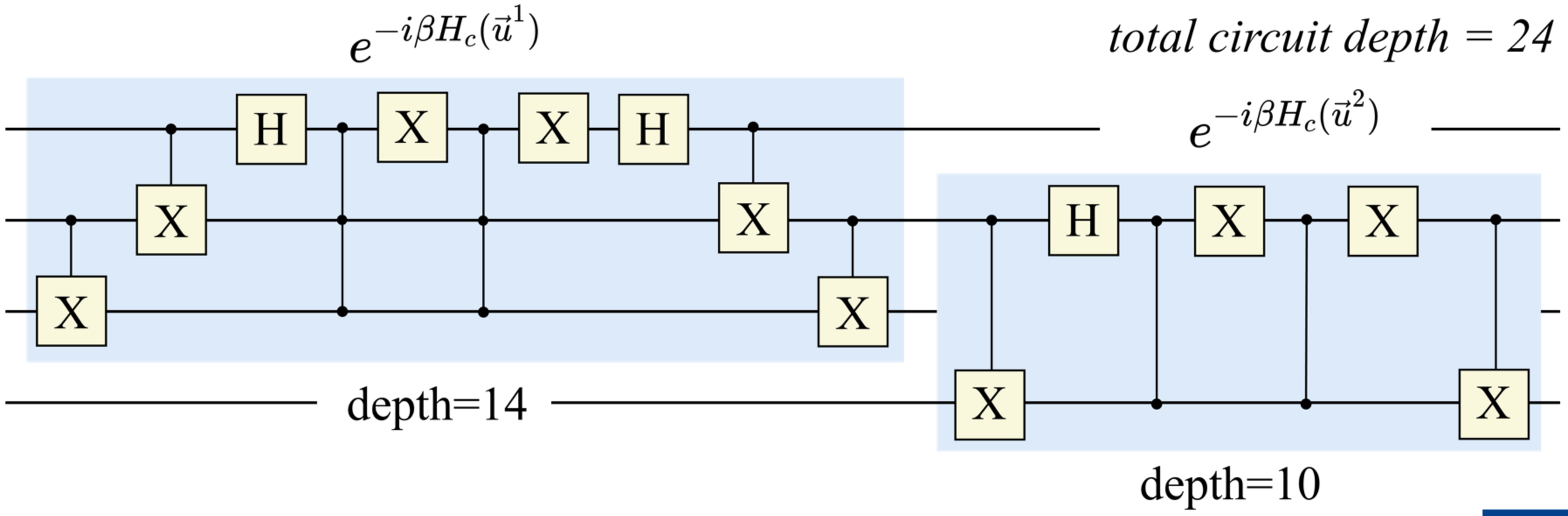
Hamiltonian Serialization



Prior method: Trotter decomposition

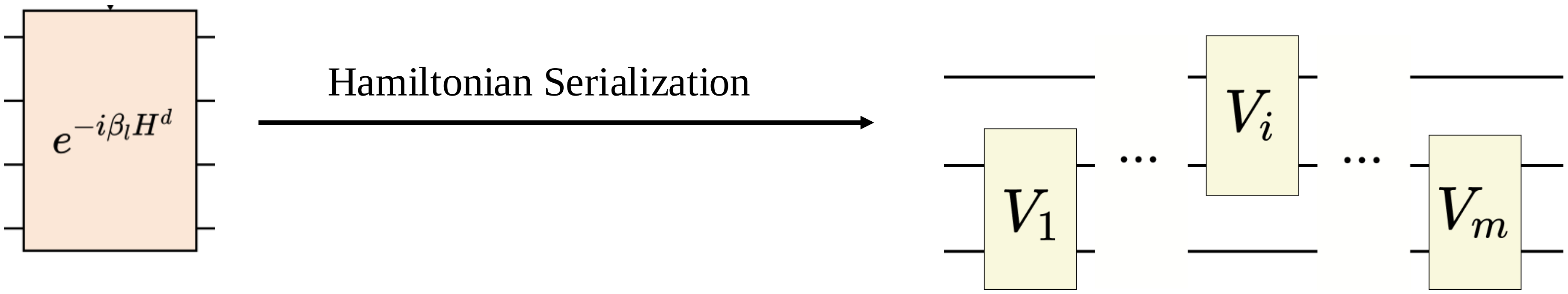


Our method: Serialization



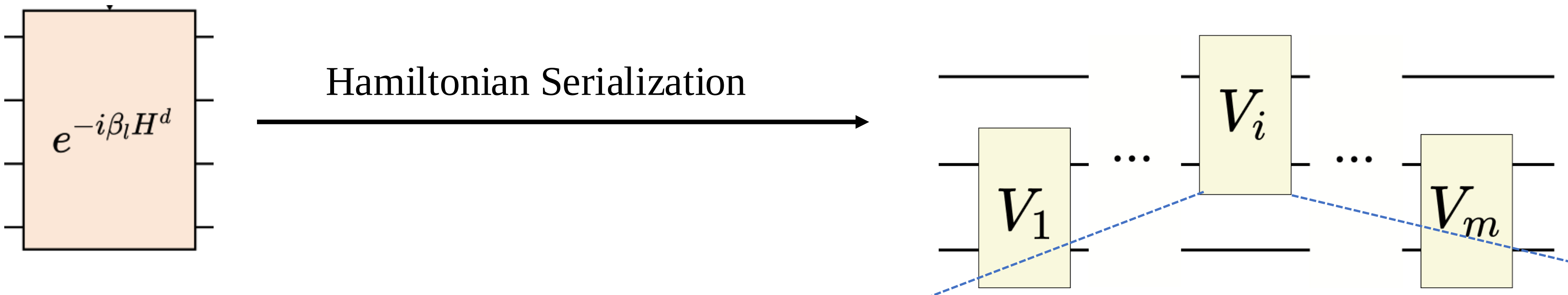
Hamiltonian Decomposition

$$V_i = e^{-i\beta_l H^d(\vec{u})}$$

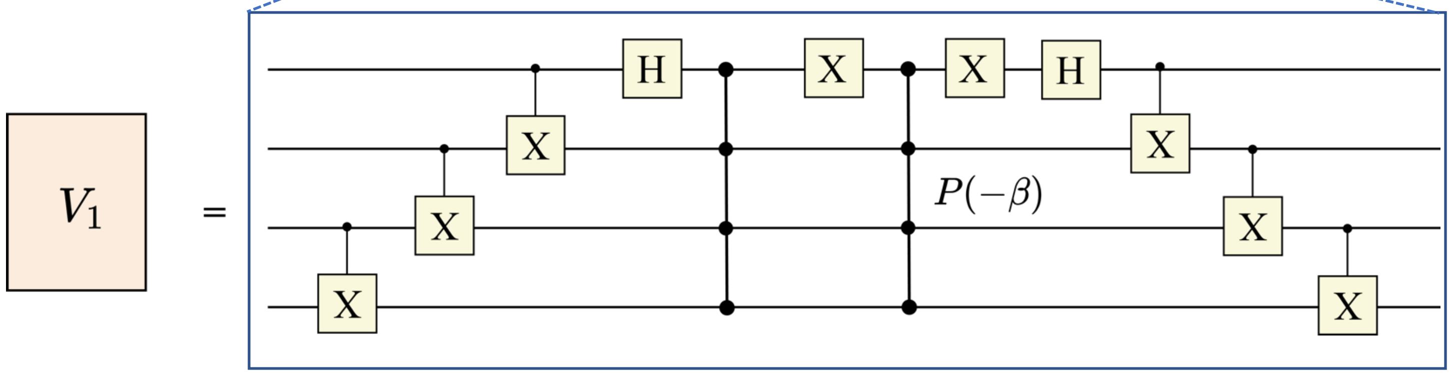


Hamiltonian Decomposition

$$V_i = e^{-i\beta_l H^d(\vec{u})}$$

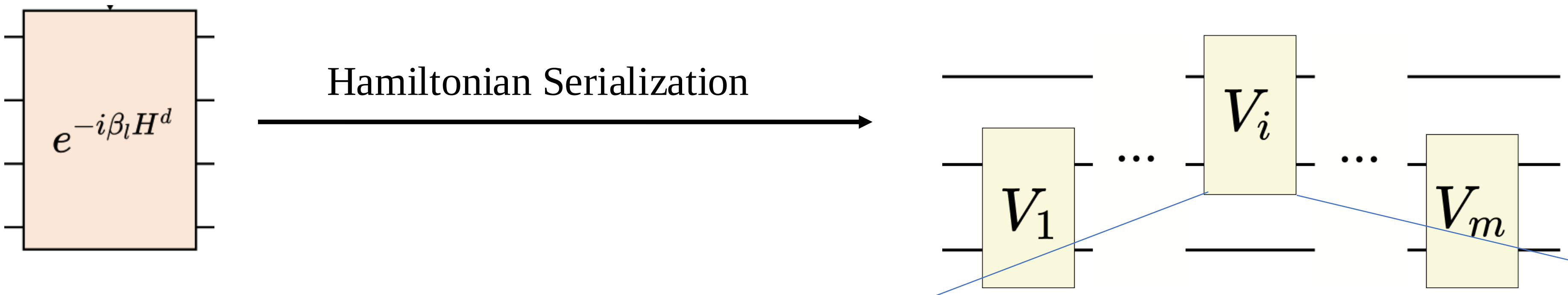


Linear decomposition by Phase gate

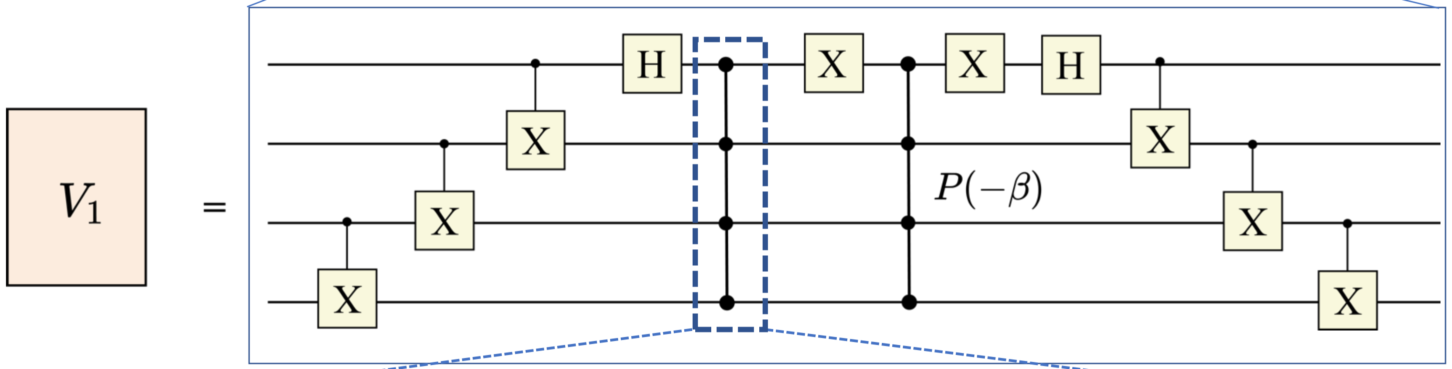


Hamiltonian Decomposition

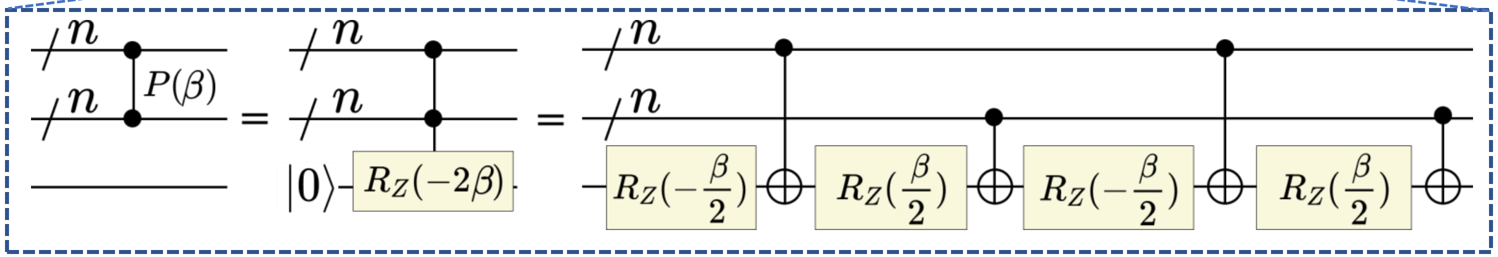
$$V_i = e^{-i\beta_l H^d(\vec{u})}$$



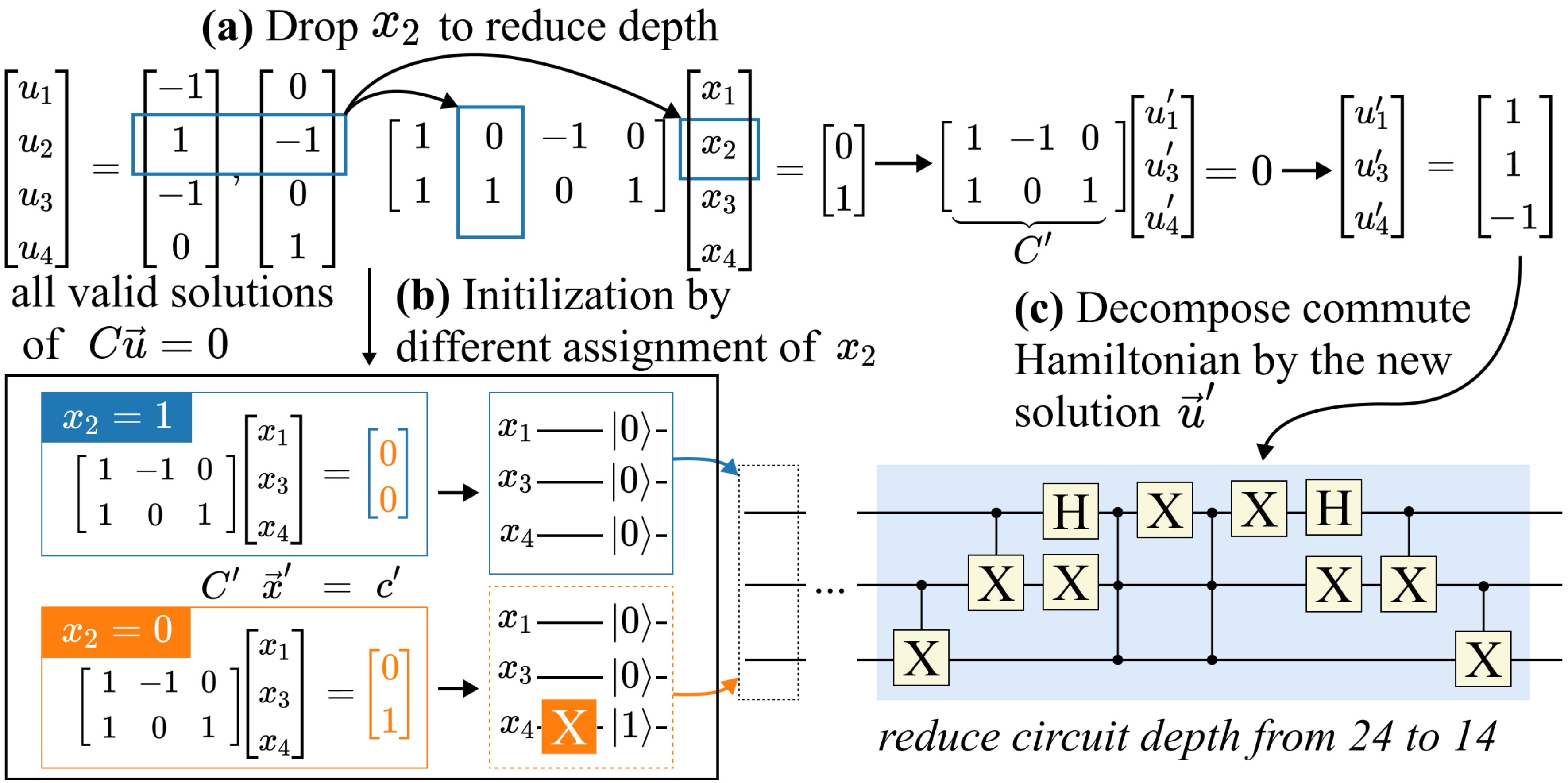
Linear decomposition by Phase gate



Linear decomposition of Phase gate



Variable Elimination



Outline of Presentation

- Background and challenges
- Overview of Choco-Q
- Choco-Q Optimization
- **Experiment**
- API of Choco-Q

Experiment: benchmarks

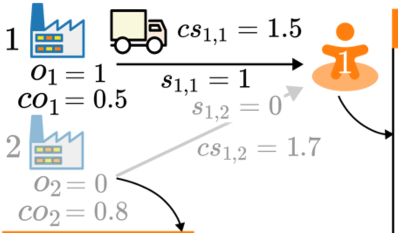
Facility location

variables

supplying cost $cs_{d,f}$; o_f; $s_{d,f}$; demand $d \in \mathcal{D}$

facility $f \in \mathcal{F}$, opening cost co_f

$o_f = \begin{cases} 1, & \text{open facility } f \\ 0, & \text{otherwise} \end{cases}$ $\quad s_{d,f} = \begin{cases} 1, & \text{facility } f \text{ supplies demand } d \\ 0, & \text{otherwise} \end{cases}$



constraint 1

Each demand can only be assigned to one facility

$$\sum_{f \in \mathcal{F}} s_{d,f} = 1, \forall d \in \mathcal{D}$$

constraint 2

An unopened facility cannot supply any demand

$$s_{d,f} \le o_f, \forall d \in \mathcal{D}, \forall f \in \mathcal{F}$$

Convert into equality by binary variable $a_{d,f}$

$$s_{d,f} + a_{d,f} = o_f, \forall d \in \mathcal{D}, \forall f \in \mathcal{F}$$

objective

Minimize the cost of supplying demands and opening facilities

$$\min_{s_{d,f}, o_f} \underbrace{\sum_{d \in \mathcal{D}} \sum_{f \in \mathcal{F}} cs_{d,f} \cdot s_{d,f}}_{\text{supplying cost}} + \underbrace{\sum_{f \in \mathcal{F}} co_f \cdot o_f}_{\text{opening cost}}$$

$$= 1.5 + 0.5$$

benchmark	#case	#variable	#constraint
F1: 2F-1D	100	6	3
F2: 3F-2D	100	15	8
F3: 3F-3D	100	21	12
F4: 4F-3D	100	28	15

2F-1D: two facilities and one demand.

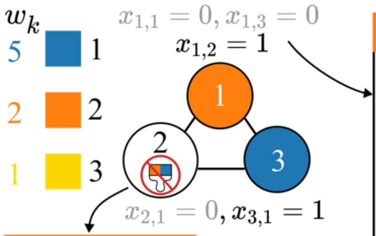
Graph coloring

variables

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $(i,j) \in \mathcal{E}$; coloring $x_{i,k}$; color weight w_k

vertex $i \in \mathcal{V}$; color $k \in \mathcal{C}$

$x_{i,k} = \begin{cases} 1, & \text{vertex } i \text{ is colored } k \\ 0, & \text{otherwise} \end{cases}$



constraint 1

Each vertex can only be colored by one color

$$\sum_{k \in \mathcal{C}} x_{i,k} = 1, \forall i \in \mathcal{V}$$

constraint 2

Adjacent vertexes cannot be colored with same color

$$x_{i,k} + x_{j,k} \le 1, \forall (i,j) \in \mathcal{E}, \forall k \in \mathcal{C}$$

Convert into equality by binary variable $a_{(i,j),k}$

$$x_{i,k} + x_{j,k} + a_{(i,j),k} = 1, \forall (i,j) \in \mathcal{E}, \forall k \in \mathcal{C}$$

objective

Minimize the weights of color used

$$\min_{x_{i,k}} \underbrace{\sum_{k \in \mathcal{C}} w_k \Big(1 - \prod_{i \in \mathcal{V}} \underbrace{(1 - x_{i,k})}_{0,\text{ if vertex } i \text{ is colored by } k}\Big)}_{0,\text{ if color } k \text{ is used}}$$

$$= 5 + 2 + 1$$
$$= 8$$

benchmark	#case	#variable	#constraint
G1: 3V-1E	100	12	6
G2: 3V-2E	100	15	9
G3: 4V-2E	100	24	12
G4: 4V-3E	100	28	16

3V-1E: a graph with three vertexes and one edge

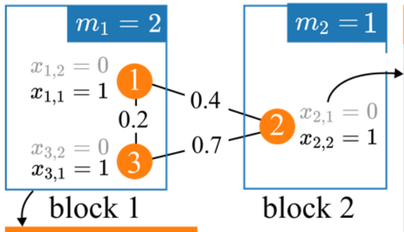
K-partitioning

variables

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$; edge weight $w_{i,j}$, $(i,j) \in \mathcal{E}$; partitioned into $x_{i,k}$; capacity m_k

vertex $i \in \mathcal{V}$; block $k \in \mathcal{B}$

$x_{i,k} = \begin{cases} 1, & \text{vertex } i \text{ is assigned to block } k \\ 0, & \text{otherwise} \end{cases}$



constraint 1

Each vertex can only be housed in one block

$$\sum_{k \in \mathcal{B}} x_{i,k} = 1, \forall i \in \mathcal{V}$$

constraint 2

The size of vertexes in a block equals its capacity

$$\sum_{i \in \mathcal{V}} x_{i,k} = m_k, \forall k \in \mathcal{B}$$

It is an equality and needs no further conversion

objective

Maximize the weight of cut edges between blocks

$$\max_{x_{i,k}} \sum_{(i,j) \in \mathcal{E}} w_{i,j} \Big(1 - \underbrace{\sum_{k \in \mathcal{B}} \underbrace{x_{i,k} x_{j,k}}_{0,\text{ if vertex } i,j \text{ in block } k}}_{0,\text{ if vertex } i,j \text{ in diffrent blocks}}\Big)$$

$$= 0.4 + 0.7$$

benchmark	#case	#variable	#constraint
K1: 4V-3E-2B	100	8	6
K2: 6V-5E-3B	100	18	9
K3: 8V-7E-3B	100	24	11
K4: 9V-8E-3B	100	27	12

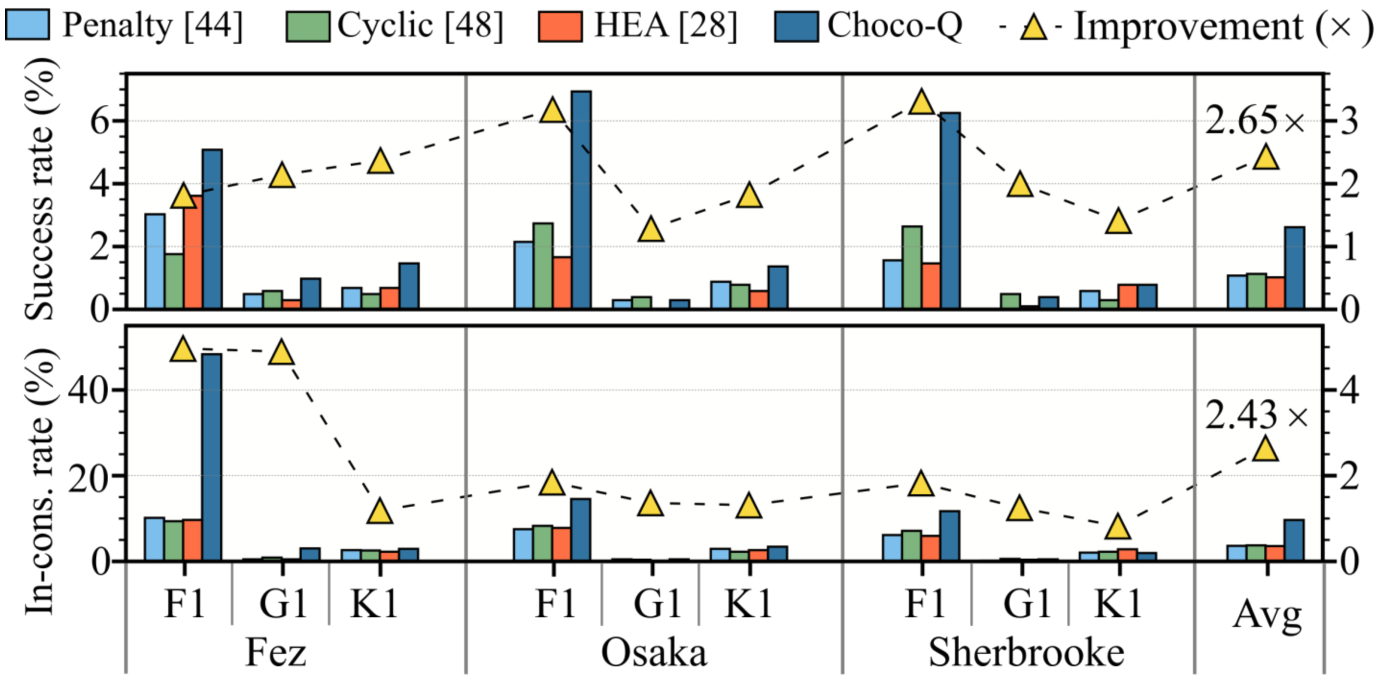
4V-3E-2B: four vertexes, three edges, and two blocks.

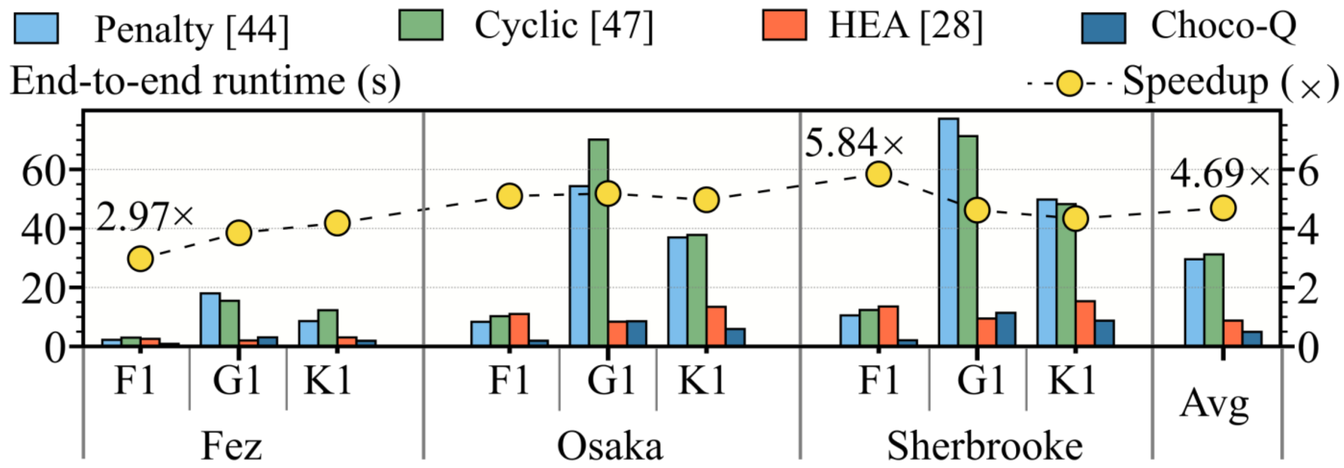
Facility location — Graph coloring — K-partitioning

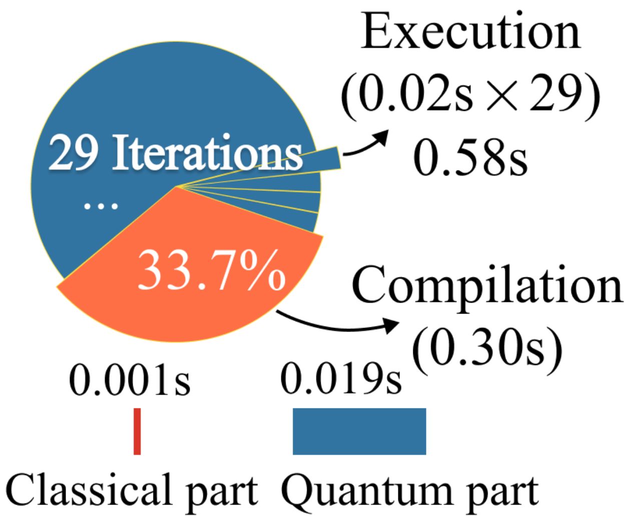
Noise-free Evaluation

Benchmark		Circuit depth					Success rate (%)					In-constraints rate (%)			
		Penalty [48]	Cyclic [53]	HEA [32]	Choco -Q	Choco -Q*	Penalty [48]	Cyclic [53]	HEA [32]	Choco -Q	Choco -Q*	Penalty [48]	Cyclic [53]	HEA [32]	Choco -Q
FLP [41]	F1	56	91	42	44	43	3.79	21.4	8.91	**60.1**	**99.8**	22.0	38.7	26.4	**100.0**
	F2	88	99	98	341	221	0.14	0.09	✗	**46.0**	**54.0**	0.47	1.37	0.10	**100.0**
	F3	92	134	147	665	275	✗	0.03	✗	**10.4**	**30.0**	0.04	0.79	✗	**100.0**
	F4	108	162	196	989	421	✗	✗	✗	**12.2**	**13.3**	✗	0.74	✗	**100.0**
GCP [30]	G1	188	230	84	168	124	0.15	4.74	0.26	**32.7**	**69.7**	0.50	10.6	0.36	**100.0**
	G2	221	263	105	519	358	0.03	0.14	0.03	**19.4**	**67.1**	0.07	0.67	0.03	**100.0**
	G3	654	708	168	769	586	✗	✗	✗	**2.75**	**17.1**	0.02	0.27	✗	**100.0**
	G4	683	737	196	1115	906	✗	✗	✗	**0.26**	**9.50**	✗	✗	✗	**100.0**
KPP [11]	K1	115	150	56	115	79	1.66	38.2	1.04	**56.3**	**86.1**	5.18	84.8	4.46	**100.0**
	K2	202	244	126	385	324	0.01	14.2	✗	**31.2**	**52.6**	0.05	39.7	0.04	**100.0**
	K3	264	306	168	538	464	0.01	2.59	✗	**3.64**	**21.1**	0.01	31.6	✗	**100.0**
	K4	296	338	189	614	534	✗	0.45	✗	**1.82**	**13.3**	✗	8.23	✗	**100.0**
JSP [50]	J1	72	168	49	92	67	5.15	13.1	3.54	**69.4**	**84.1**	15.7	38.2	9.17	**100.0**
	J2	118	224	98	280	184	0.12	2.17	0.02	**16.6**	**24.7**	1.68	20.4	0.28	**100.0**
	J3	134	243	126	395	298	0.02	1.03	✗	**6.92**	**9.78**	0.53	10.0	0.04	**100.0**
	J4	166	291	189	676	556	✗	✗	✗	**1.00**	**1.47**	0.04	9.12	✗	**100.0**
TSP [51]	T1	192	376	77	242	121	0.44	0.37	0.06	**78.4**	**99.0**	0.96	1.42	0.11	**100.0**
	T2	538	828	189	1122	493	✗	✗	✗	**11.1**	**30.7**	✗	✗	✗	**100.0**
SCP [12]	S1	179	198	65	99	80	4.22	5.53	1.61	**36.8**	**77.6**	11.9	14.2	6.61	**100.0**
	S2	223	250	76	162	154	0.59	1.24	0.02	**18.7**	**26.1**	1.17	2.43	0.30	**100.0**
	S3	449	485	129	359	231	0.22	0.41	✗	**10.3**	**17.9**	0.63	0.92	0.03	**100.0**
Improv.(×)		-	-	-	**0.96**	**1.34**	-	-	-	**>81.3**	**>157**	-	-	-	**>60.0**

Evaluation on quantum computer



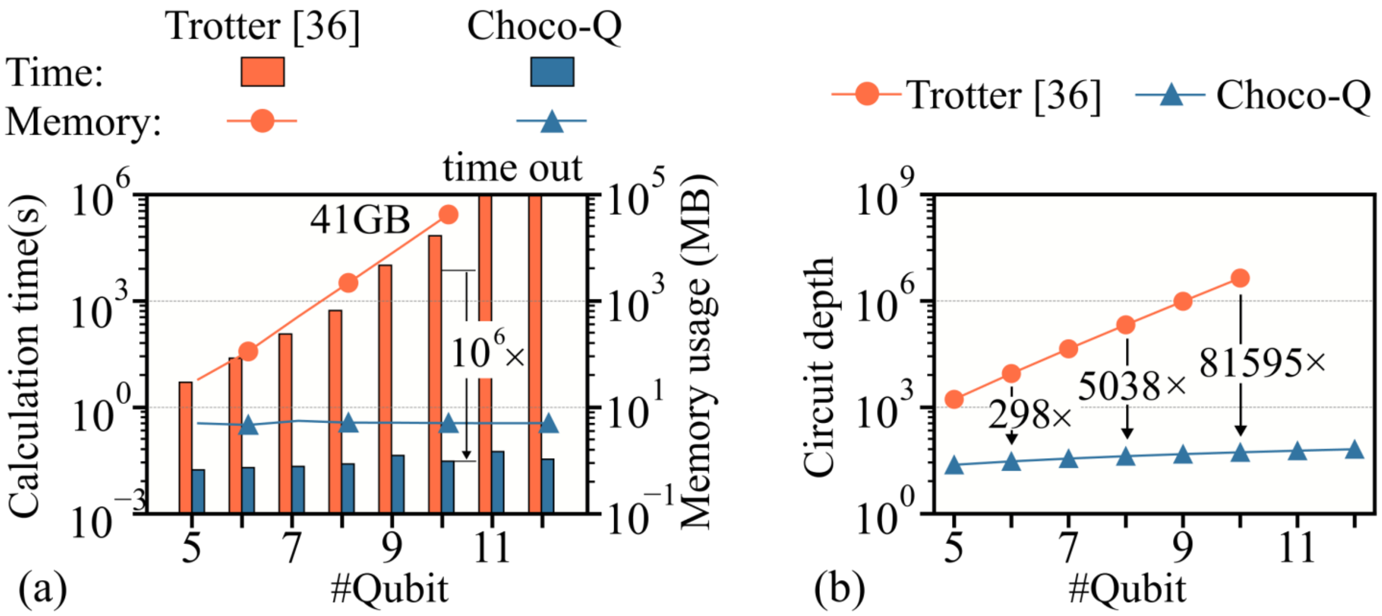




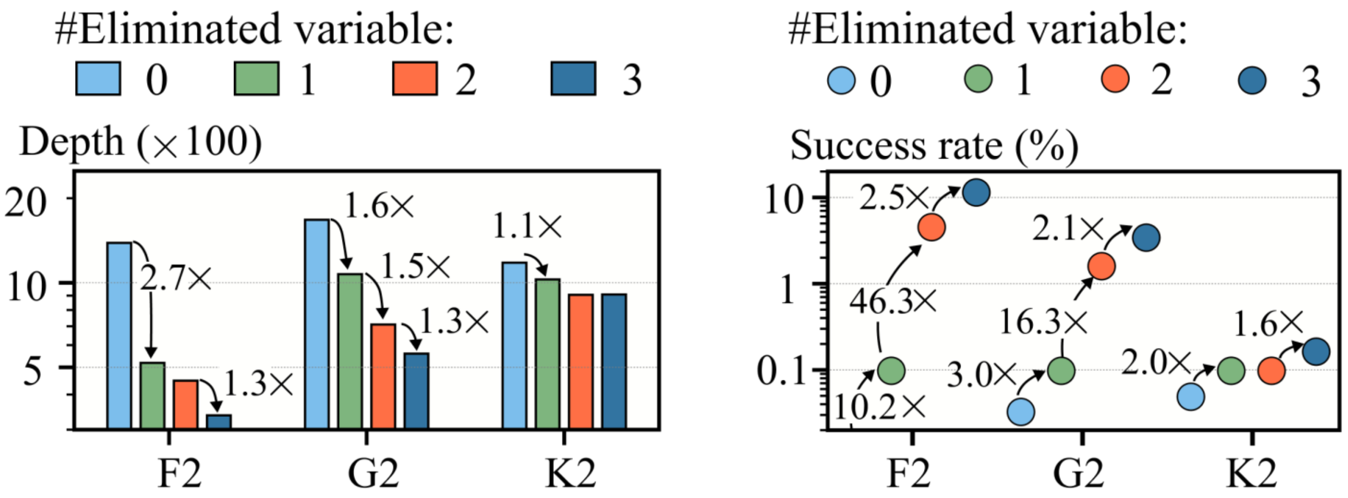
End-to-end Latency breakdown for F1 benchmarks on the Fez device.

Detailed evaluation of techniques

Hamiltonian serialization &decomposition



Variable elimination



Outline of Presentation

API of Choco-Q

File:
- JanusQ/examples/ipynb/6_1_constrained_binary_optimization.ipynb
- https://janusq.github.io/tutorials/demo/6_1_constrained_binary_optimization

```
from janusq.application.chocoq.chocoq.model import LinearConstrainedBinaryOptimization
from janusq.application.chocoq.chocoq.solvers.optimizers import CobylaOptimizer
from janusq.application.chocoq.chocoq.solvers.qiskit import ChocoSolver, DdsimProvider

m = LinearConstrainedBinaryOptimization()
x = m.addVars(5, name="x")
m.setObjective((x[0] + x[1])* x[3] + x[2], "max")
m.addConstr(x[0] + x[1] - x[2] == 0)
m.addConstr(x[2] + x[3] - x[4] == 1)

opt = CobylaOptimizer(max_iter=200)
aer = DdsimProvider()
solver = ChocoSolver(prb_model=m, optimizer=opt, provider=aer, num_layers=1)

state_list, prob_list, iter_count = solver.solve()
result_dict = {prob: state for state, prob in zip(state_list, prob_list)}
best_solution_prob, in_constraints_prob, ARG, iter_count = solver.evaluation()
```

configure the optimization problem

construct solver

solve by Choco-Q

Thanks for listening

Choco-Q: Commute Hamiltonian-based QAOA for Constrained Binary Optimization

Debin Xiang†, Qifan Jiang†, Liqiang Lu*, Siwei Tan, and Jianwei Yin*