HPC-Centric Quantum Computing に向けた サービスに向けて

○**片桐 孝洋**, 高橋 一郎, 森下 誠, 星野 哲也, 河合 直聡, 永井 亨（名古屋大学）

AXIES2024、奈良コンベンションセンター
HPCテクノロジー3、2024年12月10日（火曜） 15:00～15:15、C会場

名古屋大学 NAGOYA UNIVERSITY

目次

1. 背景
2. 事例１：コヒーレントイジングマシン
3. 事例２：CMOSアニーリングマシン
4. 予備評価
5. おわりに

目次

1. 背景
2. 事例１：コヒーレントイジングマシン
3. 事例２：CMOSアニーリングマシン
4. 予備評価
5. おわりに

量子古典ハイブリット計算環境とスーパーコンピュータ

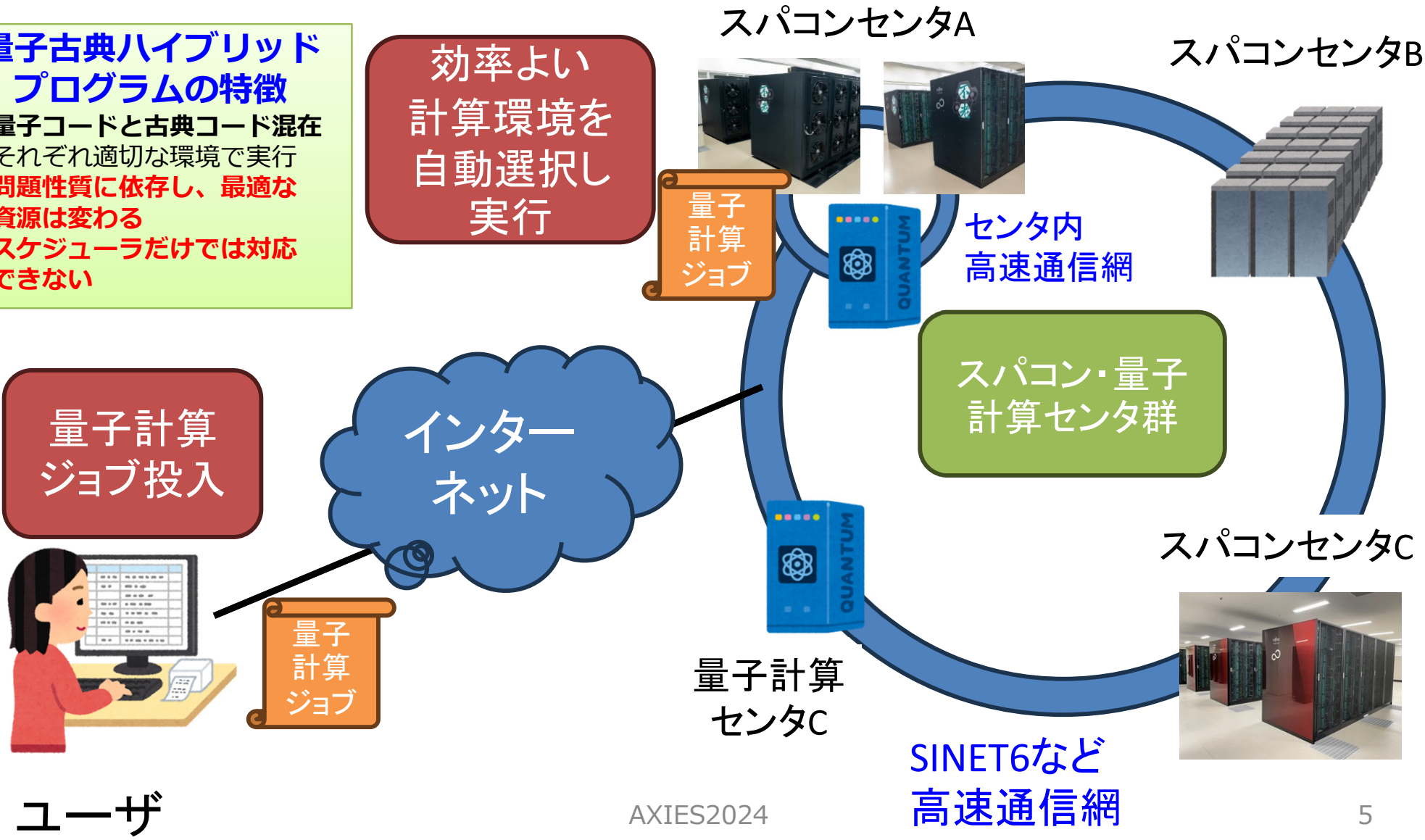
HPC-Centric Quantum Computing

- FTQC実用化までは、HPC技術中核とし量子研究の支援をする **HPC-Centric Quantum Computing** を行う

Source: T. Katagiri, HQCC-AT: An Application Programming Interface for Hybrid Quantum-Classical Computing with Auto-tuning Facility, 7th September 2024. DOI: 10.13140/RG.2.2.18404.39043

量子古典ハイブリッドプログラムの特徴
⇒量子コードと古典コード混在
⇒それぞれ適切な環境で実行
⇒問題性質に依存し、最適な資源は変わる
⇒スケジューラだけでは対応できない

効率よい計算環境を自動選択し実行



目的

1. 量子処理を記載することで簡便かつ高性能に実行できる量子・古典計算基盤開発

- ユーザの観点では量子・古典の計算環境を意識せず、高速かつ解の精度が高いようにジョブが自動実行され、計算結果が得られることが最重要
- 従来のジョブスケジューラが行っている、ユーザが直接どの計算機にジョブを投げるか指定した実行形態では不十分
- ユーザが扱っている問題の大きさや問題の数理的な性質を自動判定し、実行速度や解の精度を予測した最適化により、自動的に適する計算実行環境（量子計算機実機やGPU による量子回路シミュレータの実行など）を切り替える

2. スーパーコンピュータの高速化技術開発

- 古典計算環境の高速化は不可避
- 現在主流のマルチコアCPU やGPU のハードウェアを考慮した古典処理の高速化技術開発は必須
- Python環境の高速化が必須

AT技術による量子・古典計算環境

● 将来実現するFTQCへのシームレスな移行を支援

古典計算機

古典アルゴリズム

GPU並列　　マルチコア並列

スーパーコンピュータ

アニーラ型（疑似量子）

- 日立CMOSアニーリングマシン
- 富士通デジタルアニーラ
- NEC Vector Annealing

疑似量子コンピュータ（クラウド）

全自動で
ジョブ（プログラム）
が移動

自動チューニング

問題規模・特性等
に応じ
適する環境を
自動選択

量子計算機（実機）

回路型

NISQ

アニーラ型

D-waveほか

量子回路シミュレータ

回路型
（シミュレータ）

GPU並列

スーパーコンピュータ

目次

コヒーレント・イジングマシン（光量子コンピュータ）

NTT Researchとの共同研究

コヒーレント・イジングマシン(CIM)[1]

- 光パラメトリック発振器（OPO）を用いて、インコヒーレント光（位相がランダムな雑音光）を種とするレーザを利用したイジングマシン
- エラー訂正機能を持つタイプ（closed-loop CIM）、エラー訂正機能を持たないタイプ（open-loop CIM）の2つがある。
- 最近はclosed-loop方式が主体。実装は困難だが、既存のアルゴリズムを凌駕する性能が得られる。
- CIMは、実機、実機＋古典、シミュレータがある。シミュレータはCyber-CIMと呼ばれる。
- カオス的振る舞いを入れることで、局所解に陥ることを防ぐ工夫がされている（Chaotic Amplitude Control (CAC) algorithm, CIM-CAC)

[1]山本喜久:最適化問題解決の未来：コヒーレントイジングマシン（CIM）, NTT Research : Upgrade Realityをめざした3つのオープンコラボレーション (2021.11)
https://journal.ntt.co.jp/article/16174

CIM-CACm[2]

- Chaotic Amplitude Controlを用いたCIM
- シミュレータ（Pythonプログラム）がある。
- 以下のIsing Hamiltonianの低エネルギ状態を見つけることができる

$$\mathcal{H}(\sigma) = \frac{1}{2}\sigma^T \Omega \sigma$$

ここで、$\sigma \ni -1,1^N, \Omega$はinstance-dependent Ising coupling

- Wishart planted instancesをテストするベンチマーク提供
- 評価基準はTime To Solution (TTS)

TTS = log (1 – 0.99) / log (1 – *p0*) *T*

ここで、*p0* は ground-stateを見つける確率。

[2] GitHub: NTTRI-PHI-Algorithms/CACm, https://github.com/NTTRI-PHI-Algorithms/CACm

CIM-CACm[2]のモデル

$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + \Delta t(-\beta(t)\boldsymbol{x}(t) + \alpha \boldsymbol{e}(t) \circ (\Omega \boldsymbol{\phi}(t)) + \gamma(\boldsymbol{x}(t) - \boldsymbol{x}(t-1))), \boldsymbol{e}(t+1)$$
$$= \boldsymbol{e}(t) - \xi \boldsymbol{e}(t) \circ (\boldsymbol{x}(t)^2 - 1)$$

with

$$\frac{1}{N}\sum_i e_i(t) = 1, \forall\, t, \beta = \beta_1 + \frac{t}{T}(\beta_2 - \beta_1),$$

and

$$\phi_i(t) = \tanh(x_i(t)), \forall\, i\,.$$

[2] GitHub: NTTRI-PHI-Algorithms/CACm, https://github.com/NTTRI-PHI-Algorithms/CACm

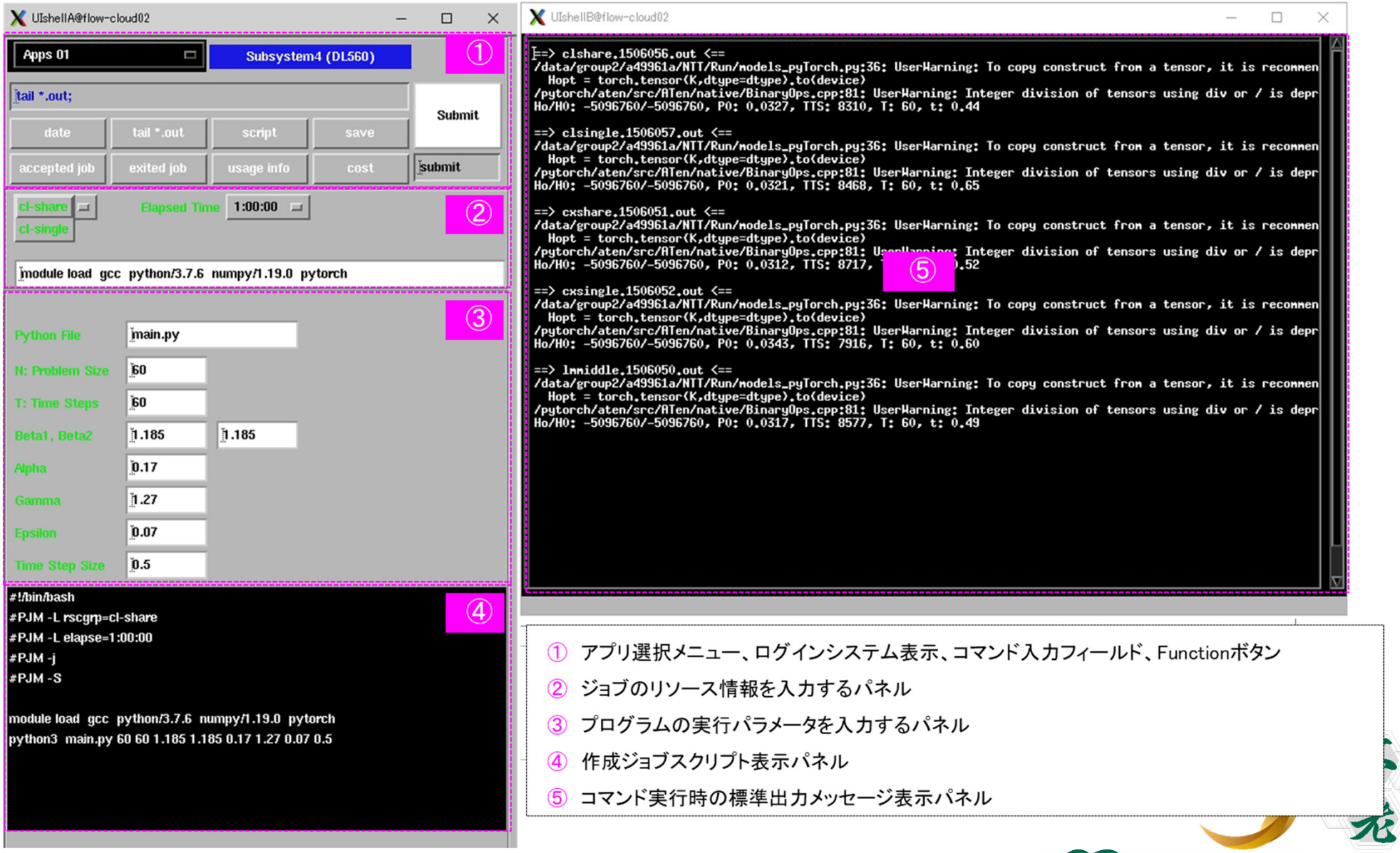
CIM-CACm[2]のパラメタ

Parameter	Interpretation
T	Number of time steps
$\beta\ 1$	Initial decay rate
$\beta\ 2$	Final decay rate
a	Coupling strength
γ	Momentum term strength
ξ	Rate of change of auxiliary variables
Δ	Time step size

[2] GitHub: NTTRI-PHI-Algorithms/CACm, https://github.com/NTTRI-PHI-Algorithms/CACm

(名大ITC独自開発)
コヒーレント・イジングマシン（CIM）用GUI（概要）

※光量子コンピュータによるイジングマシン。NTT Researchとの共同研究



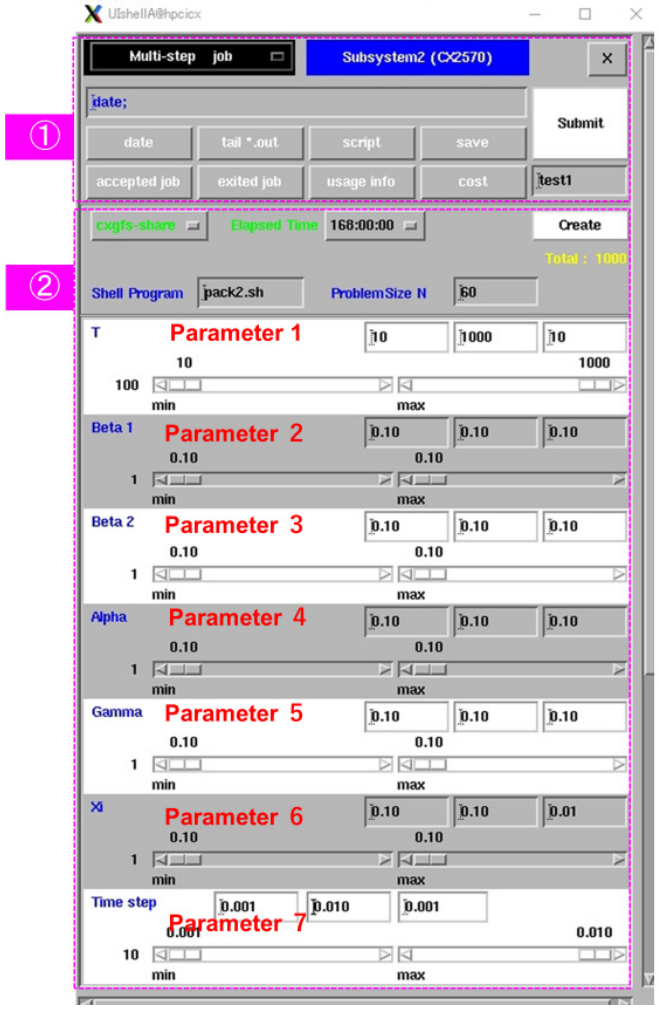
① アプリ選択メニュー、ログインシステム表示、コマンド入力フィールド、Functionボタン
② ジョブのリソース情報を入力するパネル
③ プログラムの実行パラメータを入力するパネル
④ 作成ジョブスクリプト表示パネル
⑤ コマンド実行時の標準出力メッセージ表示パネル

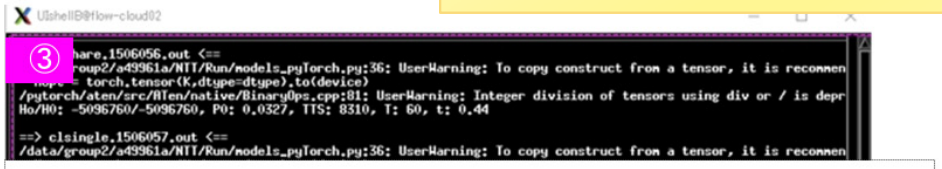
(名大ITC独自開発)

コヒーレント・イジングマシン（CIM）用GUI（パラメタサーベイ用API）

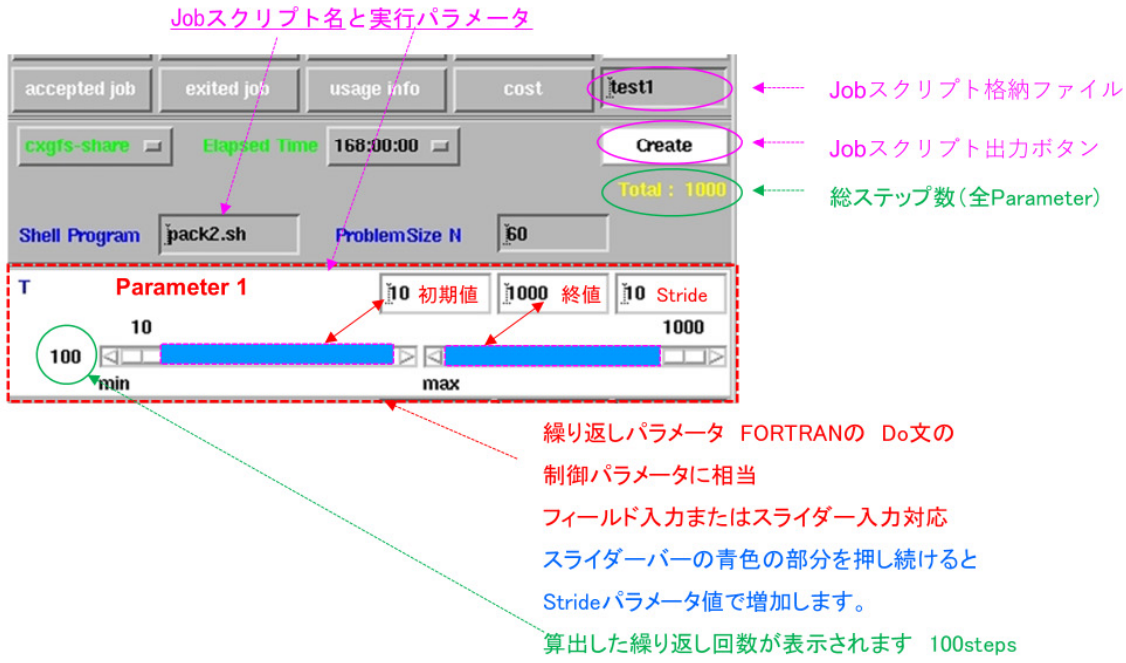
「不老」TypeI、II、クラウド上で動作

〈操作パネルのレイアウト〉





① アプリ選択メニュー、ログインシステム表示、コマンド入力フィールド、Functionボタン
② ジョブのリソース情報とプログラムの実行パラメータを入力するパネル
③ コマンド実行時の標準出力メッセージ表示パネル



※現在、パラメタ探索空間を効率的に探索する自動チューニング機能を実装中

目次

1. 背景
2. 事例１：コヒーレントイジングマシン
3. 事例２：CMOSアニーリングマシン
4. 予備評価
5. おわりに

背景

- Googleや中国科学技術大学で量子超越性の主張[1]
  - アルゴリズムを改良して高速化し、かつ、最新のスーパーコンピュータを使って超並列化
    →同程度の時間、もしくは、より短い時間で求解できる例を提示
    →量子超越性は現在無い　→それより「量子有用性」の実証
- 疑似量子アニーラ が我が国では活発に開発
  - 例）
    1. **CMOSアニーリングマシン（日立）[2]**
    2. デジタルアニーラ（富士通）
    3. シミュレーテッド分岐マシン（東芝）

    →実性能は？　性能向上の問題点は？

[1] Elizabeth Gibney, ”Hello quantum world! Google publishes landmark quantum supremacy claim”, Nature, 574(7779):pp.461-462
[2] Masanao Yamaoka, et al. "20k-spin Ising chip for combinational optimization problem with CMOS annealing", ISSCC, 2015

CMOS アニーリングマシン (1/3)

「名刺サイズ」
CMOSアニーラ実機 (ASIC, 4 bits)

- 疑似量子アニーラの位置づけ

量子コンピュータ

回路型
- IBM Quantum
- System One

アニーラ型
- D-Wave 2000Q
- D-Wave Advantage

冷却器が必要。量子ビット数は少ない。

疑似量子アニーラ

アニーラ型
- **CMOSアニーリングマシン**
- デジタルアニーラ
- シミュレーテッド分岐マシン

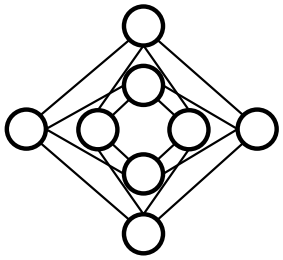
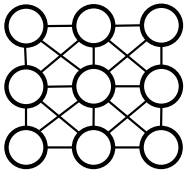
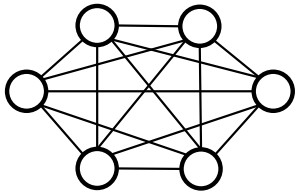
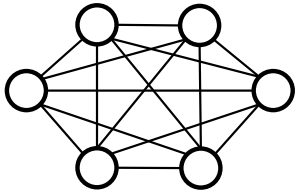
室温動作。量子ビット（相当）数が多い。

- CMOSアニーリングマシンとは？
  - 日立製作所が2015年に開発
  - イジングモデルに対するアニーリングを行うために、記憶素子であるSRAMの構造を活用して開発
  - **Annealing Cloud Web**[3] でサービス提供中(GPU version, 32bits, Float)

[3] https://annealing-cloud.com/ja/index.html

CMOS アニーリングマシン (2/3)

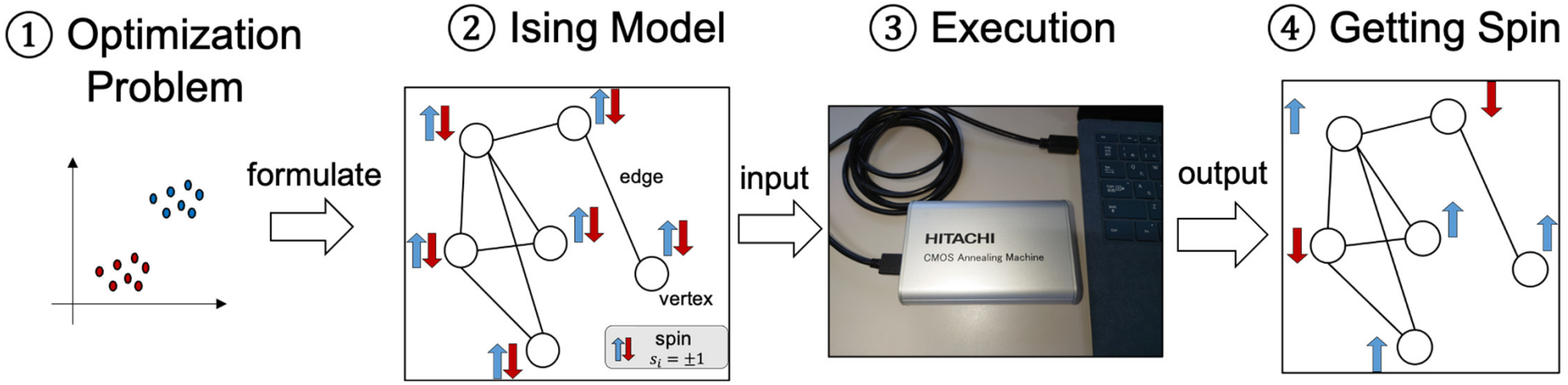
- アニーリングマシンの実装

	D-Wave 2000Q	CMOSアニーリングマシン		デジタルアニーラ	シミュレーテッド分岐マシン
		ASIC版	GPU版		
ハード	QPU	デジタル回路	GPU	デジタル回路	GPU
量子ビット	2,048	61,952	262,144	8,192	10,000
結合グラフ	キメラグラフ 	キングスグラフ (ASICのみ)  *GPU版は完全結合		完全結合グラフ 	完全結合グラフ 

名古屋大学 NAGOYA UNIVERSITY

CMOS アニーリングマシン (3/3)

- CMOSアニーリングマシンでの求解手順



名古屋大学
NAGOYA UNIVERSITY

目次

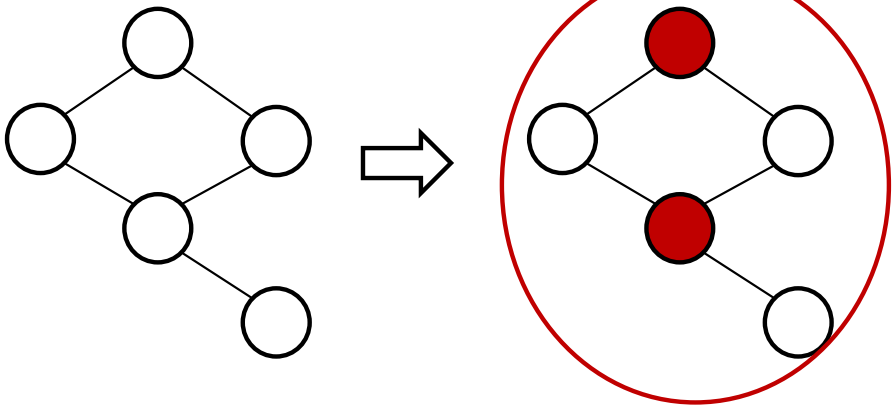
1. 背景
2. 事例１：コヒーレントイジングマシン
3. 事例２：CMOSアニーリングマシン
4. 予備評価
5. おわりに

事例：最小頂点被覆問題

問題設定（1/3）

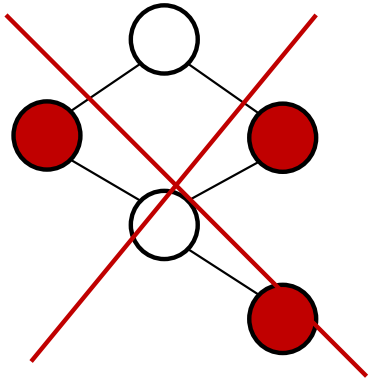
- 最小頂点被覆問題
  - グラフ $G = (V, E)$, $V' \subseteq V$) において、頂点を被覆する集合 V'を見つける（ $|V'|$ が最小となるようにする）

※頂点集合 $V' \subseteq V$はグラフ$G = (V, E)$ から
「すべての枝 $e \in E$ が、少なくとも１つの終点が V' に含まれるように取る」



例：$|V| = 5$

$|V'| = 2$
(最小頂点数)



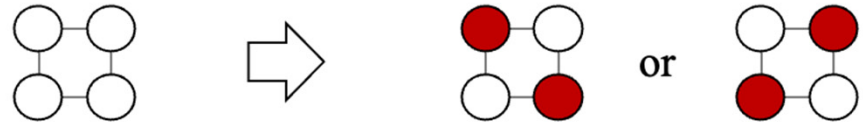
$|V'| = 3$
(最小頂点数でない)

問題設定（2/3）

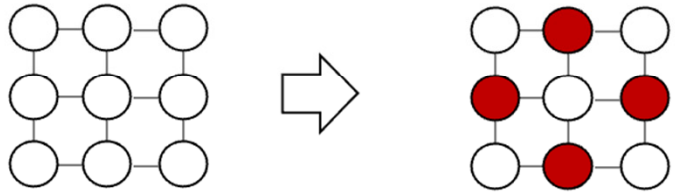
- **人工問題 (ベンチマーク問題):**
  正方格子グラフにおける最小頂点被覆問題

最小頂点被覆解が容易にわかる

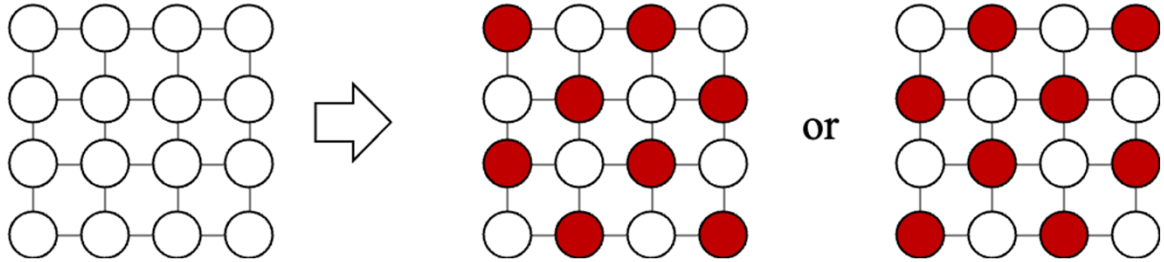
Length of one side $N = 2$



Length of one side $N = 3$



Length of one side $N = 4$



問題設定（3/3）

- 最小頂点被覆問題 (QUBO 形式)

Amplify
- フィックスターズ社開発
- イジングマシンのための Pythonライブラリ

  - エネルギー関数 (QUBO 変数 $x \in \{0,1\}$)[5]

$$H = w_a \sum_{(u,v)\in E} (1 - x_u)(1 - x_v) + w_b \sum_{v\in V'} x_v$$

制約項　　コスト項

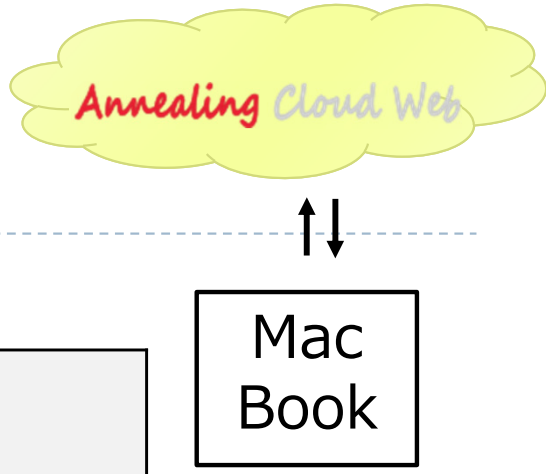
- **Amplify での実装**

```
15    # コスト関数（w_b で調整）
16    cost_function = sum([sum_poly(q[i]) for i in range(N)])
17
18    # 制約（w_a で調整）
19    constraint_x = sum([greater_equal(q[i][j] + q[i + 1][j], 1) for i in range(N - 1) for j in range(N)])
20    constraint_y = sum([greater_equal(q[i][j] + q[i][j + 1], 1) for i in range(N) for j in range(N - 1)])
21    constraint = constraint_x + constraint_y
```

```
52            # 最終的なエネルギー関数
53            energy_function = w_a*constraint + w_b*cost_function
```

[5] https://amplify.fixstars.com/ja/techresources/research/ising-model-formulation/vertex-covering/

実験環境



ハードウェア / ソフトウェア	詳細
CMOS アニーリング マシン	• **Annealing Cloud Web API v2: GPU 版 32bit (float)**
MacBookAir (macOS Big Sur)	• Python (Version 3.8.2) • 1.6GHz Dual Core Intel Core i5 • メモリ 8GB
Amplify	• Web API経由のCMOSアニーリングマシン API • Version 0.5.13

実験結果

$$H = w_a \sum_{(u,v)\in E} (1-x_u)(1-x_v) + w_b \sum_{v\in V'} x_v$$

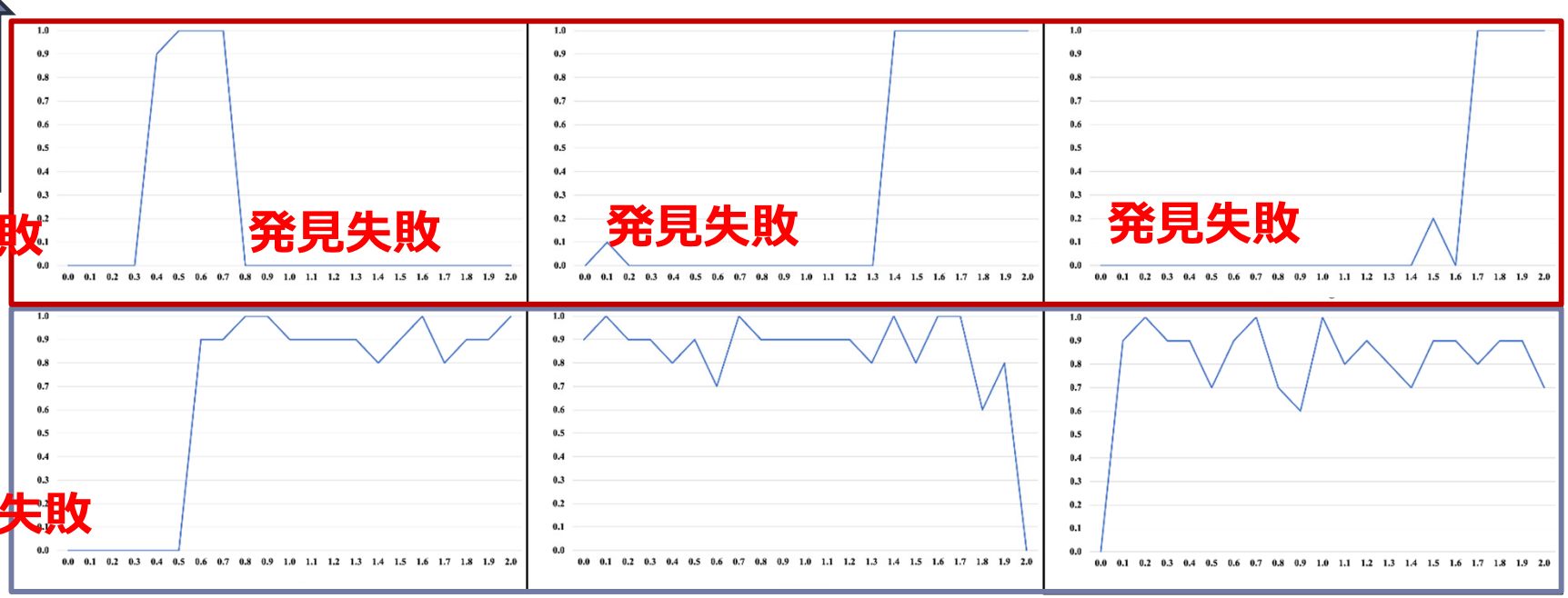
制約項　コスト項

最適解
回答率
[%]

w_a: 制約項の重み　w_b: コスト項の重み　chain_strength

$N = 7$

$N = 8$



名古屋大学
NAGOYA UNIVERSITY

疑似量子アニーラにおける Optunaによるパラメタ探索の候補

アニーリング パラメタ

パラメタ名	設定値
ステップ数	100
ステップ長	1000
初期温度	10
最終温度	0.01
実行回数	10
エネルギー値を取得するかどうか	True
スピンを取得するかどうか	True
実行時間を取得するかどうか	False

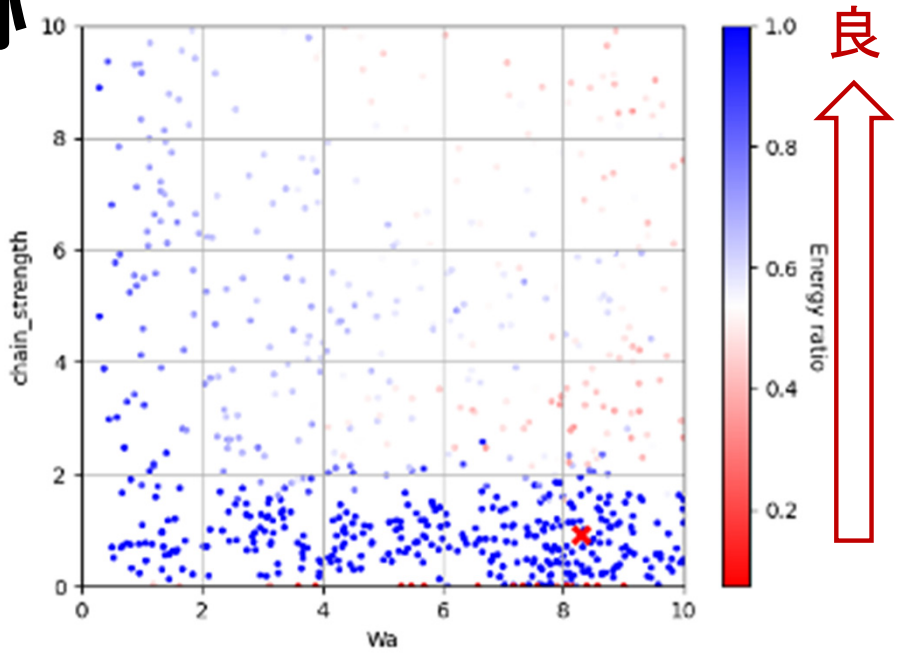
制御パラメタ

パラメタ名	設定値
正方格子グラフの１辺の長さ N	12, 13
探索アルゴリズム	Grid, Random, CMA_ES, TPE
コスト関数の重みWb	1.0
トライアル数	1000

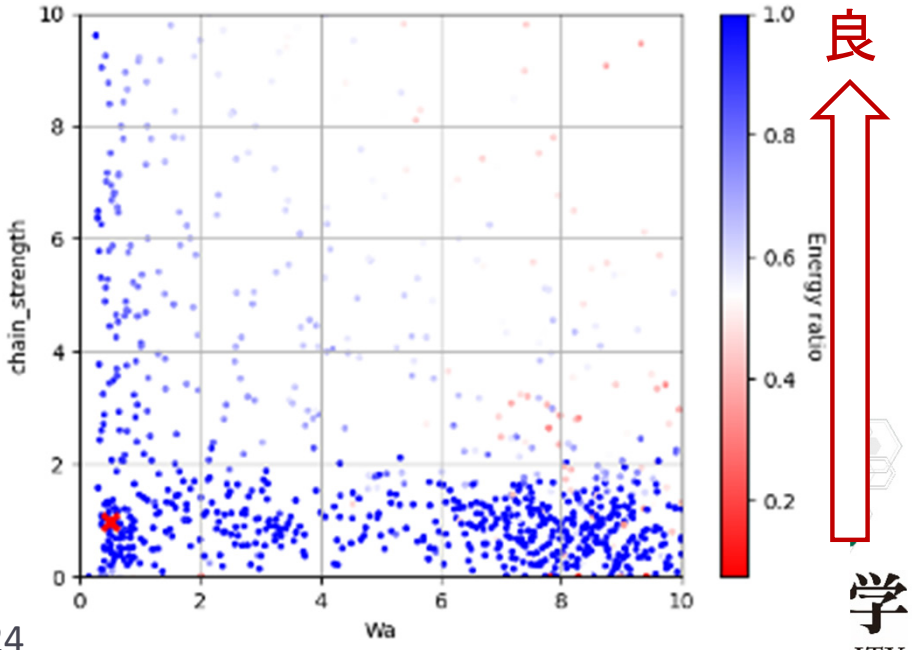
TPE (Tree-Structured Parzen Estimator) によるチューニングの軌跡

$$\text{エネルギー値の比} = \frac{\text{最適なエネルギー値}}{\text{取得したエネルギー値}}$$

Wa (横軸)とchain_strength (縦軸)
を変化させた時の
エネルギー値の比の分布
(1.0 に近いほど最適値に近い)



Optunaが返した最適値



⇒まだ探索していない範囲で、有効と思われる場所を広範囲に探索

目次

1. 背景
2. 事例１：コヒーレントイジングマシン
3. 事例２：CMOSアニーリングマシン
4. 予備評価
5. おわりに

おわりに

- HPC-Centric Quantum Computingに向け、**疑似量子アニーラや量子回路シミュレータ**のスパコンでの利用が鍵
- 利用しやすいGUIを開発
- 自動チューニング (AT) 技術 は、ここでも生産性向上の観点から必須

今後の課題

- サービスメニューの開発、システム実装、専用課金制度の検討
- 量子アニーラ上の多種の性能パラメタへのAT適用と
  ATアルゴリズム開発　→特に最適探索アルゴリズム選択
- 回路型量子コンピュータへのAT適用
  - 量子・古典量子アルゴリズム上の性能パラメタへのAT適用
- 量子回路シミュレータへのAT適用
  - 回路マージ数、GPUハード上の性能パラメタ…