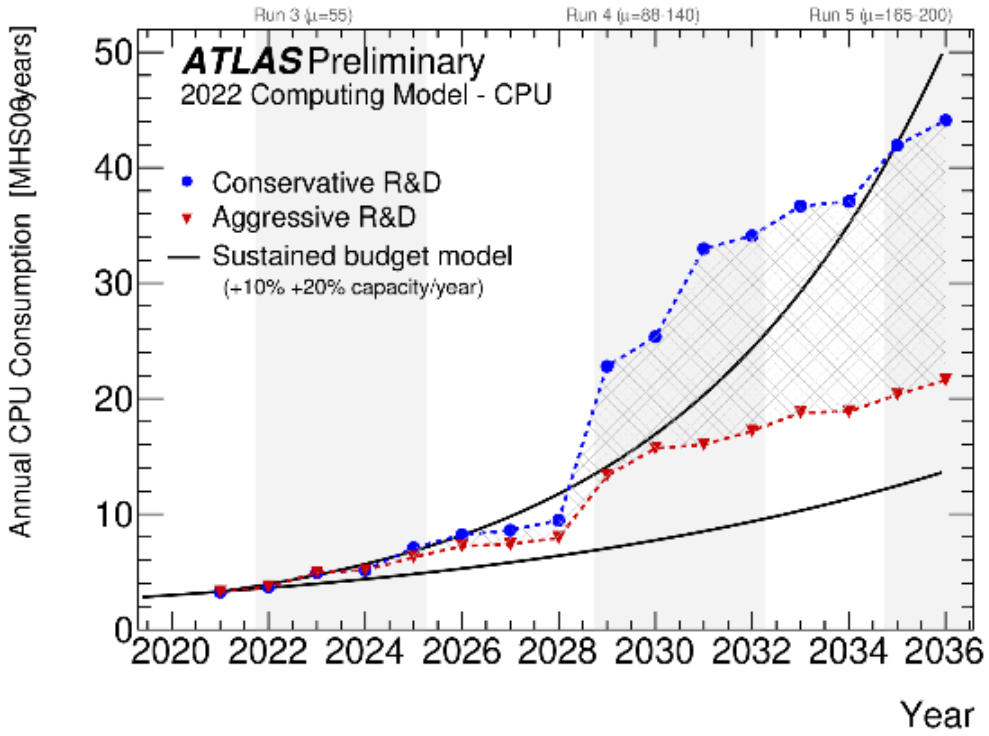
Quantum Algorithms for Track Reconstruction at High Energy Colliders

Workshop of Tracking in Particle Physics Experiments, May 17-19, 2024

大川(Okawa) 英希(Hideki)

Institute of High Energy Physics, Chinese Academy of Sciences

Why Quantum Computing for HEP?

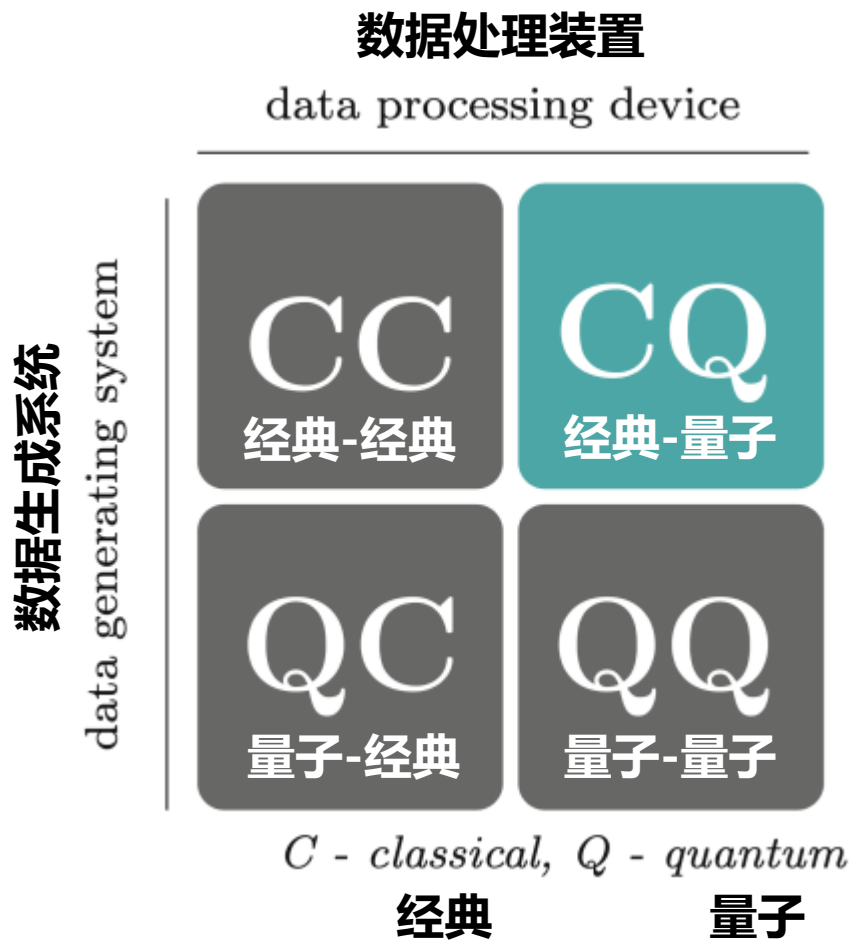


- At the HL-LHC, **we will enter the "Exa-byte" era. Annual computing cost will increase by a factor of 10-20**
- **<u>Without various innovations, the experiment will not be able to operate</u>.** GPUs and other state-of-the-art technologies will be the baseline at the HL-LHC.
- **Quantum computing may bring another "leap".**

- **Tracking is the highest CPU consuming reconstruction task.**
- Tackling these challenges will also be useful for future colliders: e.g. CEPC & SppC

Quantum Computer + Machine Learning

- **Quantum computer + machine learning → Quantum machine learning (QML)**
  "the best of both worlds"?
- Why do we bother? → Rich Hilbert space, superposition, entanglement (& tunneling) could lead to improvement in learning data
- 4 types of QML exist
- However, quantum advantage is not yet fully established in practical implementations.
- In this talk, I will present recent results utilizing CQ & CC approaches.



Tracking w/ CQ, CC

CQ approach (classical data + quantum computer)

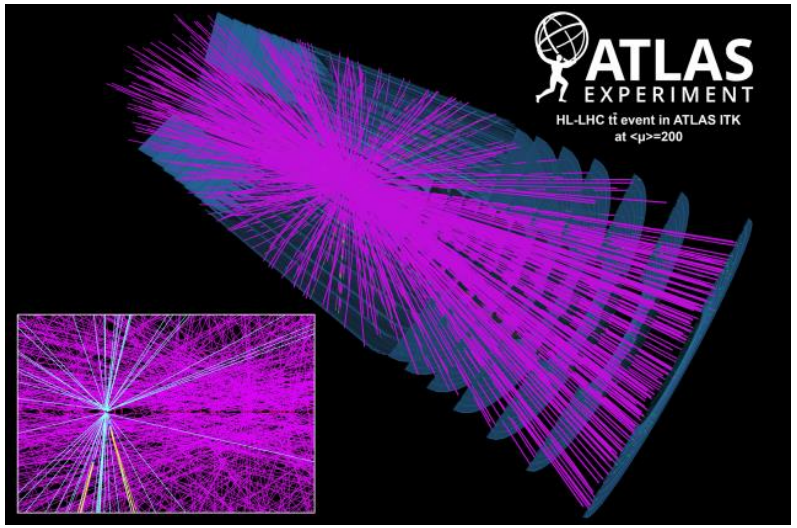
1. **H. Okawa**, “Charged particle reconstruction for future high energy colliders with Quantum Approximate Optimization Algorithm,” Springer Communications in Computer and Information Science, 2036 (2024) 272–283, arXiv:2310.10255

NEW!

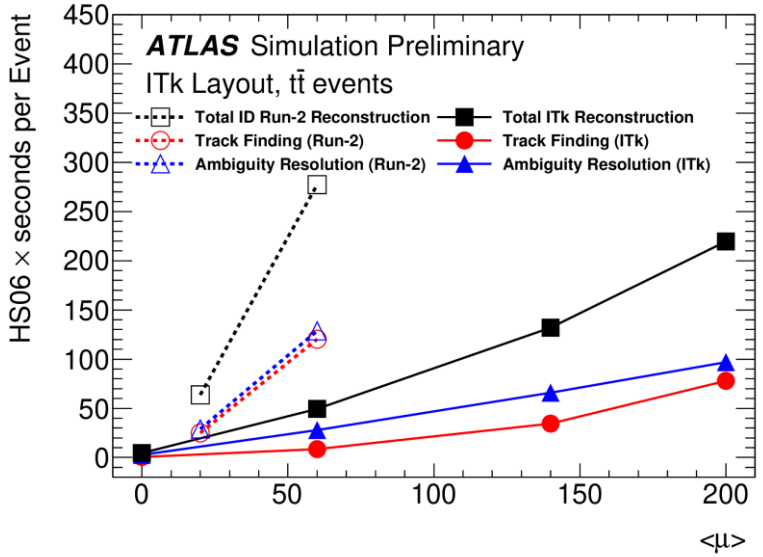
CC approach (fully classical but quantum-inspired algorithm)

2. **H. Okawa**, Q.-G. Zeng, X.-Z. Tao, M.-H. Yung, “Quantum Annealing Inspired Algorithms for Track Reconstruction at High Energy Colliders,” arXiv:2402.14718 (2024).

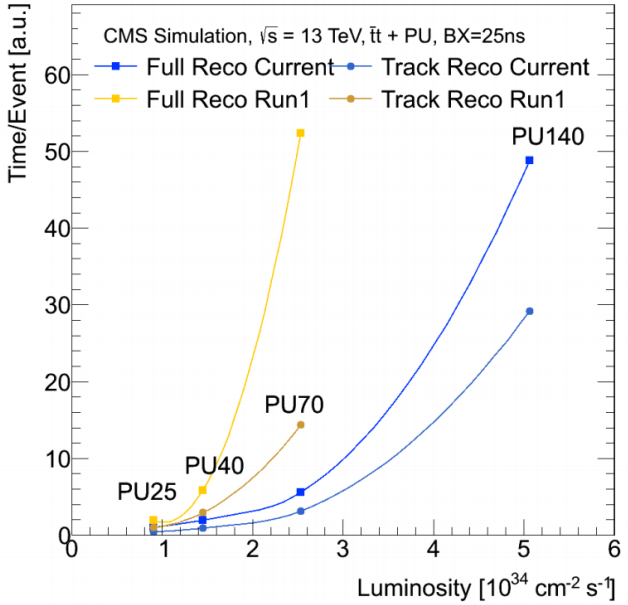
Track Reconstruction at LHC & HL-LHC



ATL-PHYS-PUB-2019-041



https://cds.cern.ch/record/1966040



	Run 1	Run 2	HL-LHC
μ	21	40	150-200
Tracks	~280	~600	**~7-10k**

- At the HL-LHC, additional interactions per bunch crossing becomes exceedingly high & **CPU time increases exponentially with more pileup.**
- GPU & ML-based approaches are actively investigated, but quantum ML may play an important role.

Quantum Approach: QUBO

- Tracks are formed by connecting silicon detector hits: e.g. triplets (segments w/ 3 hits).

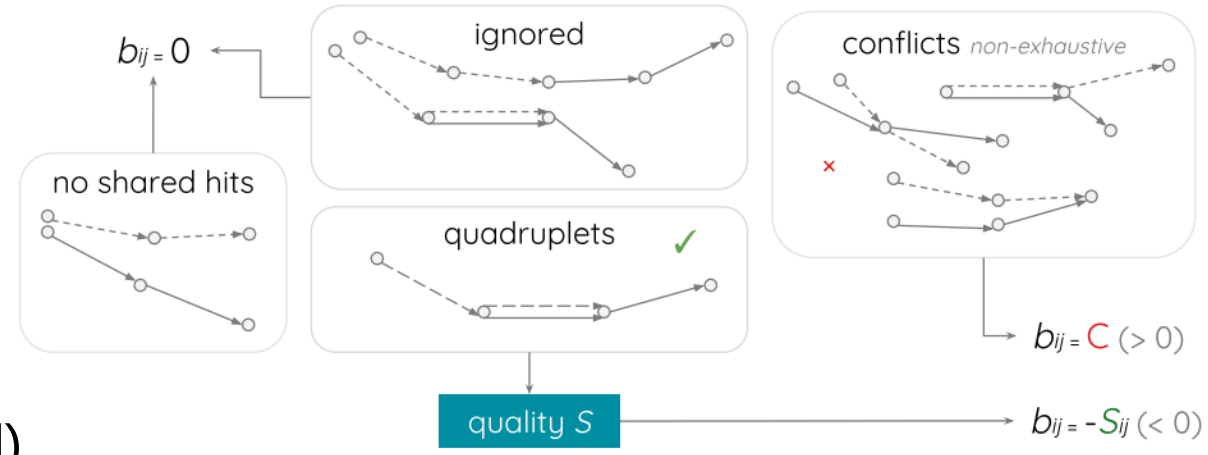
$$O(a,b,T) = \sum_{i=1}^{N} a_i T_i + \sum_{i}^{N} \sum_{j<i}^{N} b_{ij} T_i T_j$$

Quality of triplets

Compatibility b/w triplet pairs

b_{ij} = 0 (if no shared hit)
= 1 (if conflict)
= $-S_{ij}$ (if two hits are shared)

F. Bapst et al. Comp. Soft. Big Sci. 4 (2019) 1.



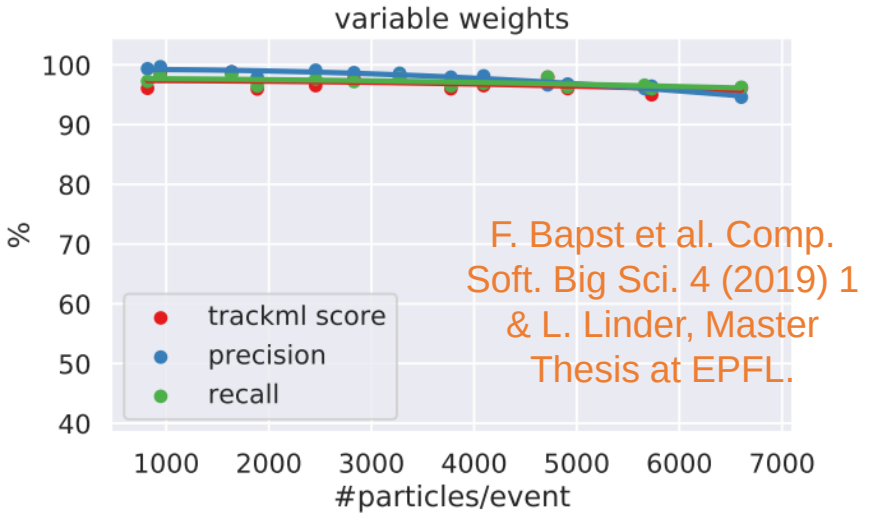
- Triplets are connected to reconstruct tracks & it can be regarded as a **quadratic unconstrained binary optimization (QUBO)** problem.
- **Minimizing QUBO is equivalent to searching for the ground state of the Hamiltonian.**

Previous Studies (not exhaustive)

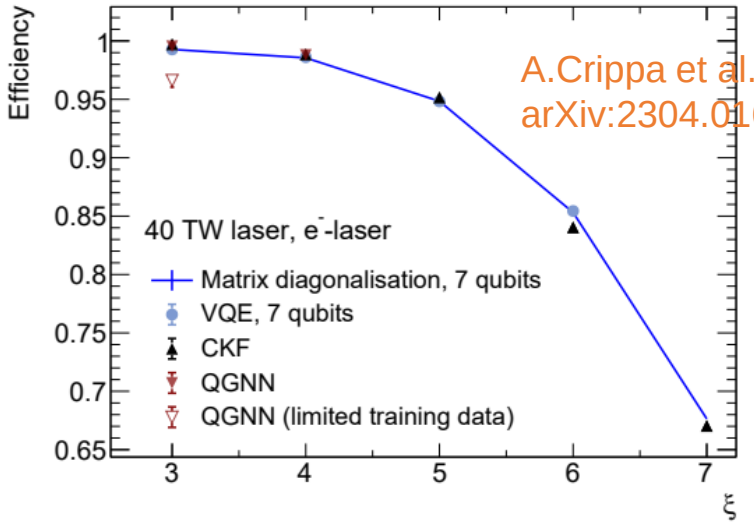
- **Quantum Annealing**
  - Quantum annealer looks for the global minimum of a given function through adiabatic theorem with quantum tunneling: a natural machine to search for the ground state of a Hamiltonian.
- **Quantum Gates**

$$\mathcal{H} = -\sum_{n=1}^{N}\sum_{m<n}\bar{b}_{nm}\sigma_n^x\sigma_m^x - \sum_{n=1}^{N}\bar{a}_n\sigma_n^x$$

  - QUBO can be mapped to Ising Hamiltonian and be solved using Variational Quantum Eigensolver (VQE), Quantum Approximate Optimization Algorithm (QAOA), or something a like.
  - There are also non-QUBO approaches such as using Quantum Graph Neural Network.



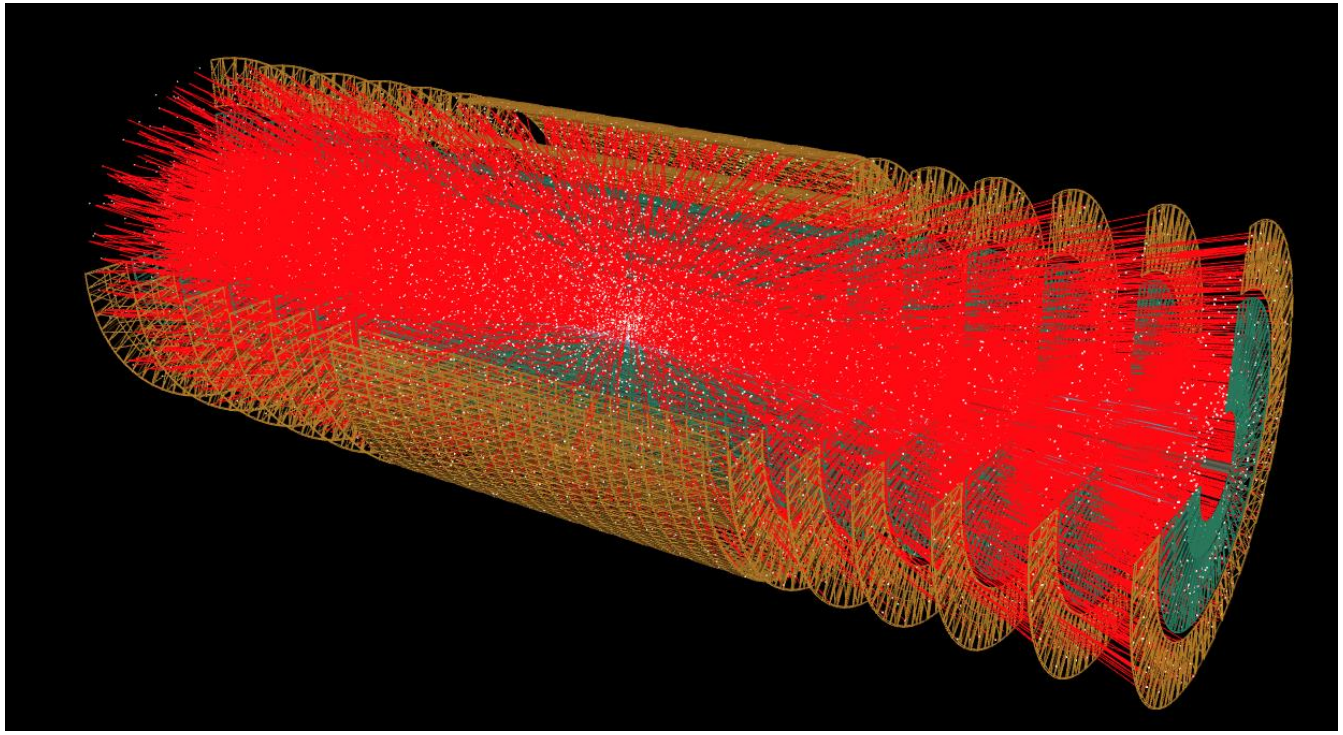
F. Bapst et al. Comp. Soft. Big Sci. 4 (2019) 1 & L. Linder, Master Thesis at EPFL.



A.Crippa et al., arXiv:2304.01690

Dataset (TrackML)

- TrackML is an open-source dataset prepared for TrackML Challenges (a competition hosted by CERN & Kaggle).
- It is **designed w/ HL-LHC conditions (200 pileup) & run w/ fast simulation (e.g. noise, inefficiency, parametrized material effects, etc.)**
- Continues to be useful for individual studies including quantum tracking.



Thanks to Andreas Salzburger for the suggestion

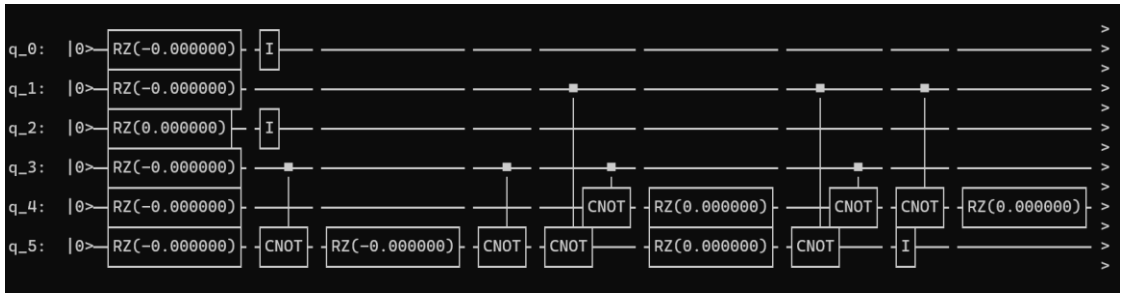
CQ Approach (经典数据+量子计算)

H. Okawa, Springer Communications in Computer and Information Science, 2036 (2024) 272–283, arXiv:2310.10255

QAOA in Origin Quantum (本源)

- VQE & QAOA libraries implemented in pyqpanda-algorithm by OriginQ (本源).
- Adopts **Quantum Alternative Operator Ansatz** for QAOA.

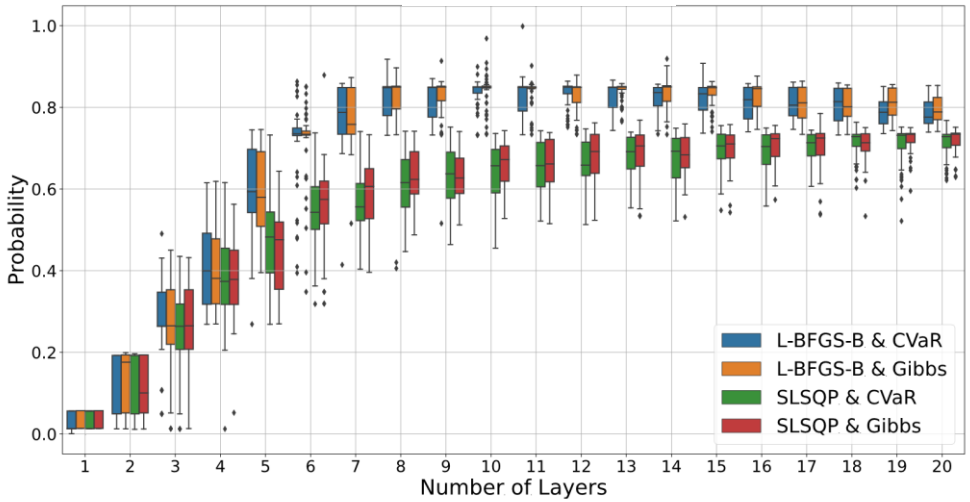
An example of circuits from the actual run



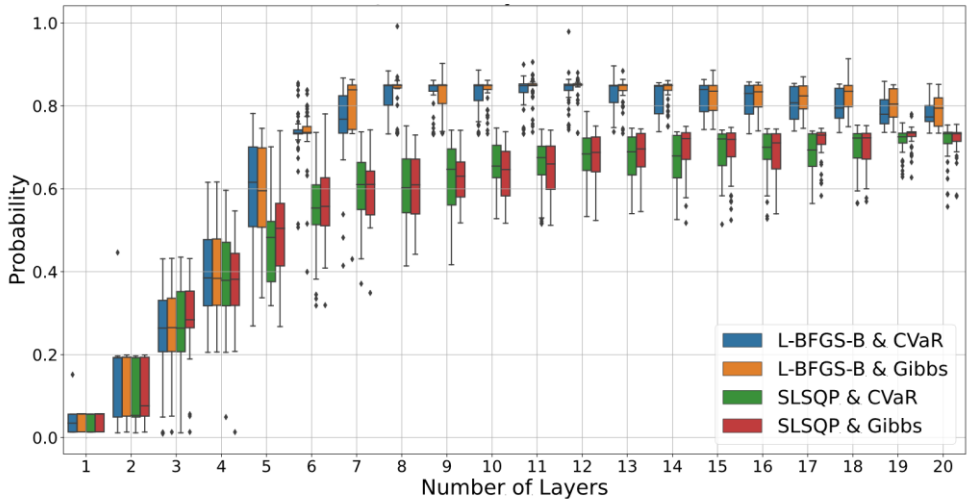
- Can utilize CVaR loss function (P. Barkoutsos et al., Quantum, 2020, 4: 256) or Gibbs optimization
- **6 qubit machine (Wuyuan 悟源)** is used for the real hardware computation in this talk.

QAOA Optimization

Simulator



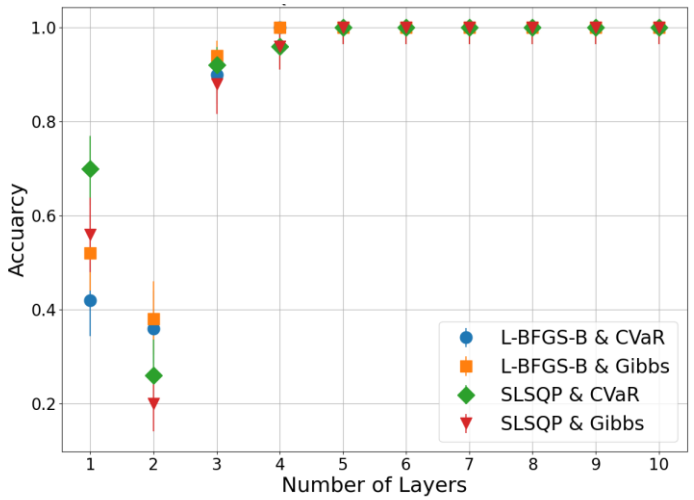
Wuyuan Hardware



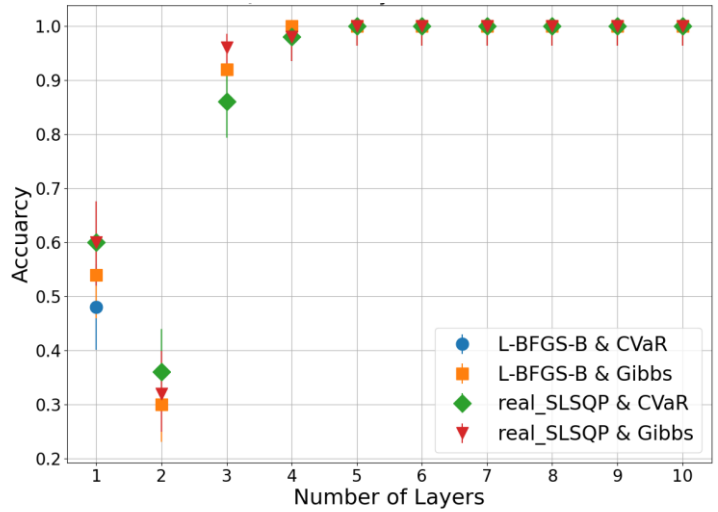
- **QAOA does not perform well w/ shallow layers, but provides good performance with more layers. Compatible performance b/w hardware & simulator.**
- L-BFGS-B optimizer is better than SLSQP. TNC has degraded performance & not shown here.
- No significant difference w/ CVaR or Gibbs loss function.
- **Probability saturates around 7 layers for L-BFGS-B cases.**

QAOA Accuracy

Simulator



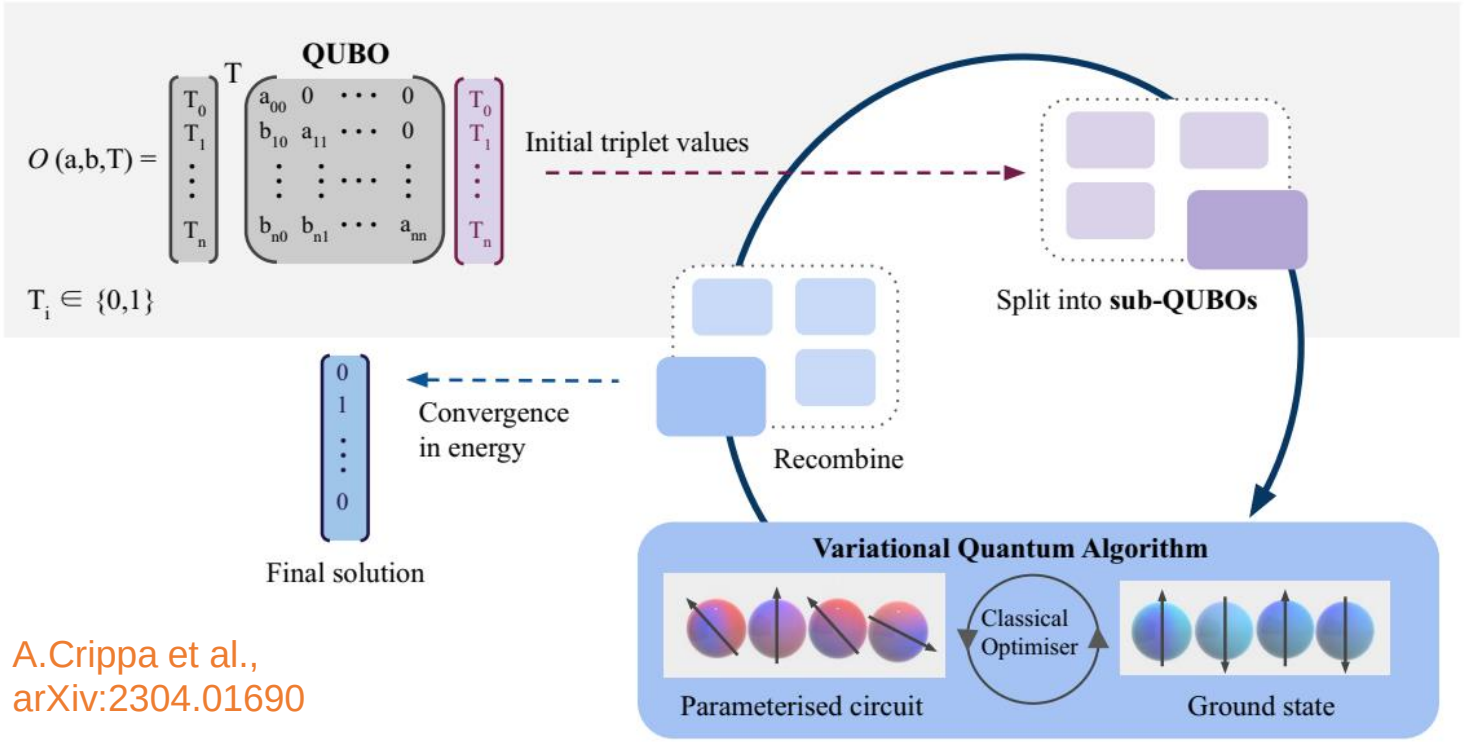
Wuyuan Hardware



- **<u>Note that the probability is NOT the accuracy of QAOA.</u>**
- A single job runs multiple measurements, ranks the answers by probability & select the highest probability state as answer.
- **<u>The accuracy already reaches 100% within the statistical uncertainty at 5 layers.</u>**
- For further studies, a conservative choice of 7 layers is used.

Sub-QUBOs

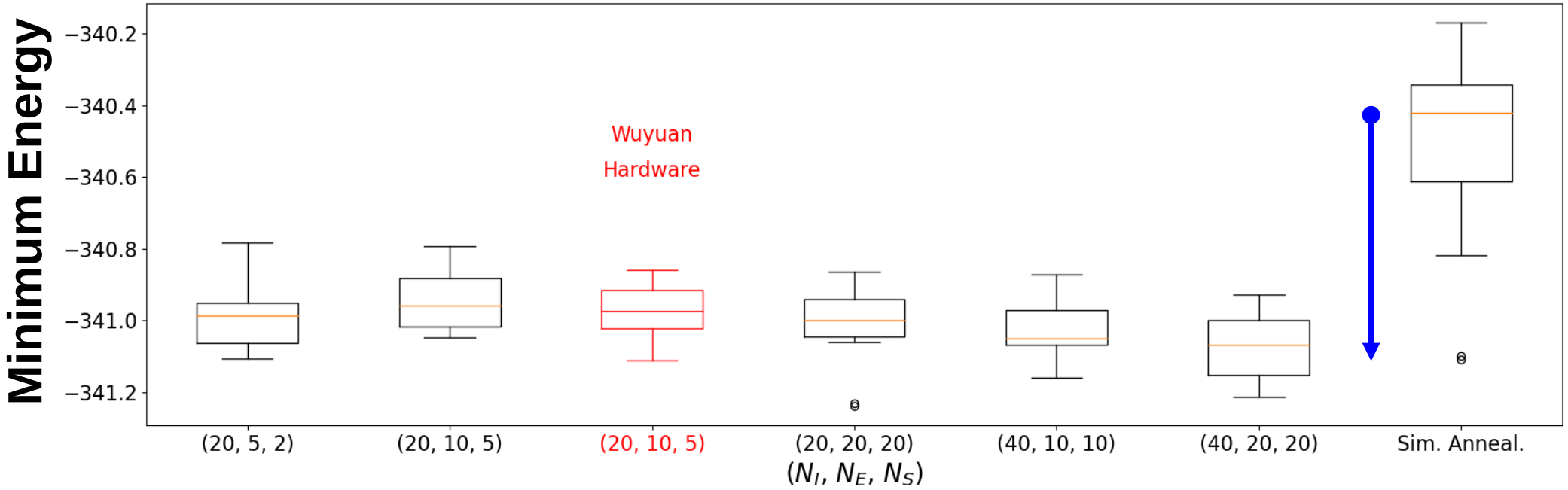
- **Number of qubits required is determined by the number of triplet candidates** → Obviously cannot cover the full QUBO [$O(10^2 \text{x} 10^2 \sim 10^5 \text{x} 10^5)$] for tracking in the NISQ era
- QUBO is split into sub-QUBOs of size **6x6 to match with OriginQ hardware.**



A.Crippa et al., arXiv:2304.01690

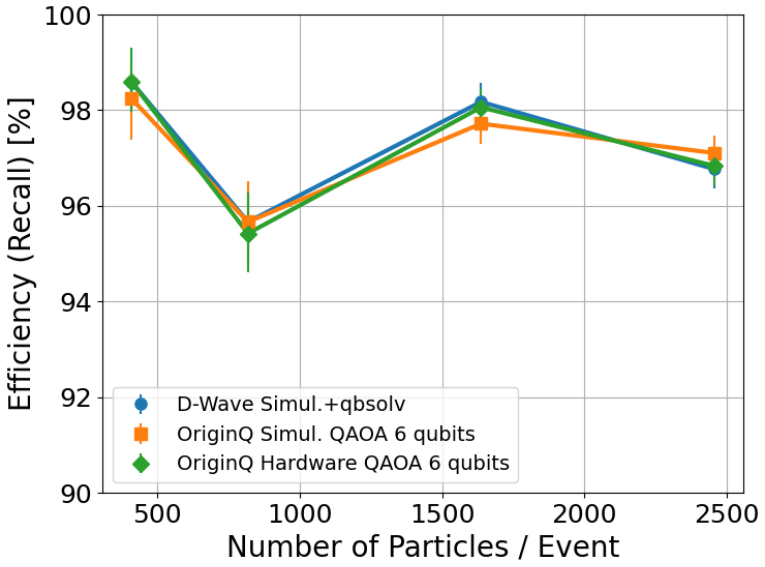
- There are various sub-QUBO algorithms proposed: qbsolv (now in dwave-hybrid library), for example.
- I adopted **a sub-QUBO method using multiple solution instances** from Y. Atobe, M. Tawada, N. Togawa, IEEE Trans. Comp. 71, 10 (2022) 2606.

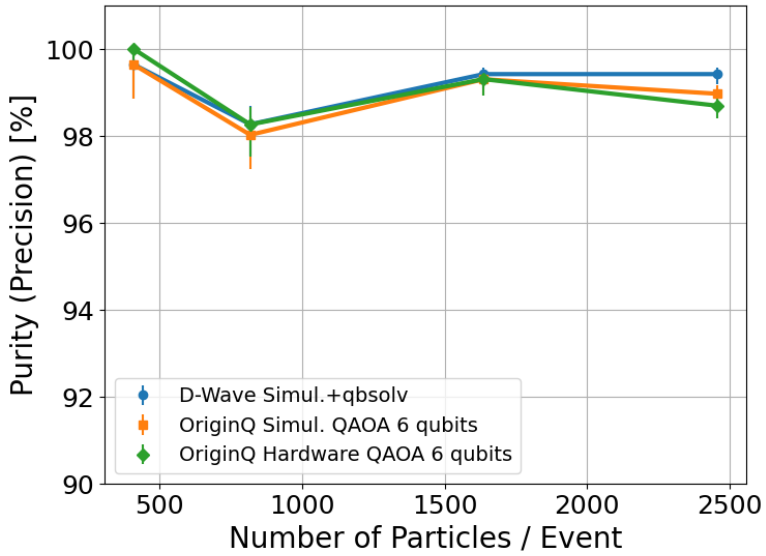
Preliminary sub-QUBO Results

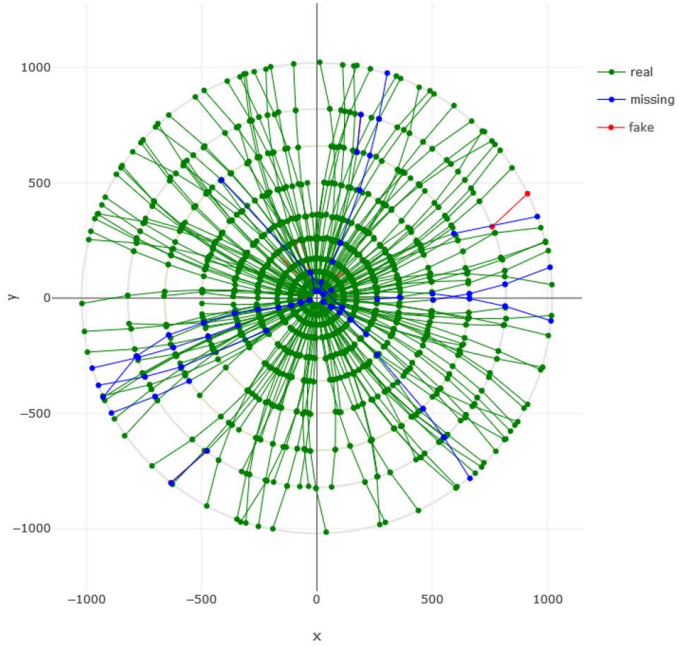


- Ran measurements to compare the performance and stability. 7 layers used in QAOA.
- No significant dependence on sub-QUBO model parameters (N_I, N_E, N_S) & **compatible performance between OriginQ simulator & actual hardware!**
- **Visible improvement w/ sub-QUBO compared to the simulated annealing only!**

WIP: Triplet Efficiency & Purity







$$\text{Efficiency} = \frac{TP}{TP+FN} = \frac{\#\text{ of matched reconstructed doublets}}{\#\text{ of true doublets}},$$

$$\text{Purity} = \frac{TP}{TP+FP} = \frac{\#\text{ of matched reconstructed doublets}}{\#\text{ of all reconstructed doublets}},$$

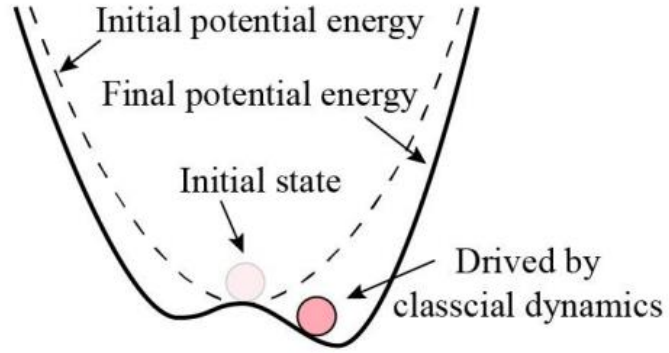
- QAOA+sub-QUBO provides **compatible performance as previous quantum annealing studies.**
- **No sign of degradation in the real hardware**

NEW!

CC Approach
(经典数据+计算, 量子启发算法)

H. Okawa, Q.-G. Zeng, X.-Z. Tao, M.-H. Yung, arXiv:2402.14718 (2024)

Simulated Bifurcation



Quantum inspired algorithm

- Quantum annealing reaches the minimum energy through adiabatic theorem
- "Quantum-inspired" algorithms search for minimum energy through the **classical time evolution of differential equations**.
- **Simulated Bifurcation (SB) in particular can run in parallel unlike simulated annealing**, in which one needs to access the full set of spins & not suitable for parallel processing

Simulated Bifurcation (SB)

➢ **adiabatic Simulated Bifurcation (aSB)**

$$\dot{x}_i = \frac{\partial H_{\mathrm{SB}}}{\partial y_i} = \Delta y_i, \qquad \dot{y}_i = \frac{\partial H_{\mathrm{SB}}}{\partial x_i} = -[Kx_i^2 + p(t) + \Delta]x_i + \xi_0 \sum_{j=1}^{N} J_{ij}x_j$$

➢ **ballistic Simulated Bifurcation (bSB)**

$$\dot{x}_i = \frac{\partial H_{\mathrm{SB}}}{\partial y_i} = \Delta y_i, \qquad \dot{y}_i = \frac{\partial H_{\mathrm{SB}}}{\partial x_i} = (p(t) - \Delta)x_i + \xi_0 \sum_{j=1}^{N} J_{ij}x_j$$

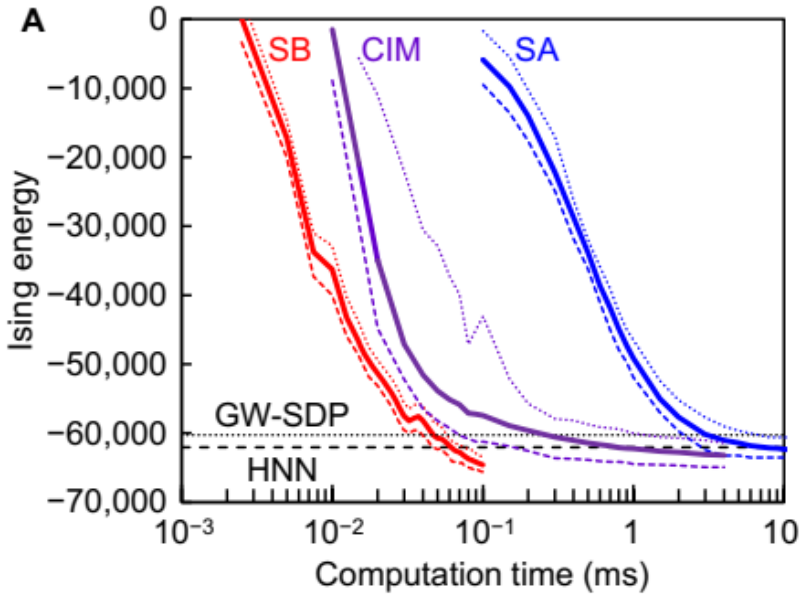
➢ **discrete Simulated Bifurcation (dSB)**

$$\dot{x}_i = \frac{\partial H_{\mathrm{SB}}}{\partial y_i} = \Delta y_i, \qquad \dot{y}_i = \frac{\partial H_{\mathrm{SB}}}{\partial x_i} = (p(t) - \Delta)x_i + \xi_0 \sum_{j=1}^{N} J_{ij}\mathrm{sign}(x_j)$$

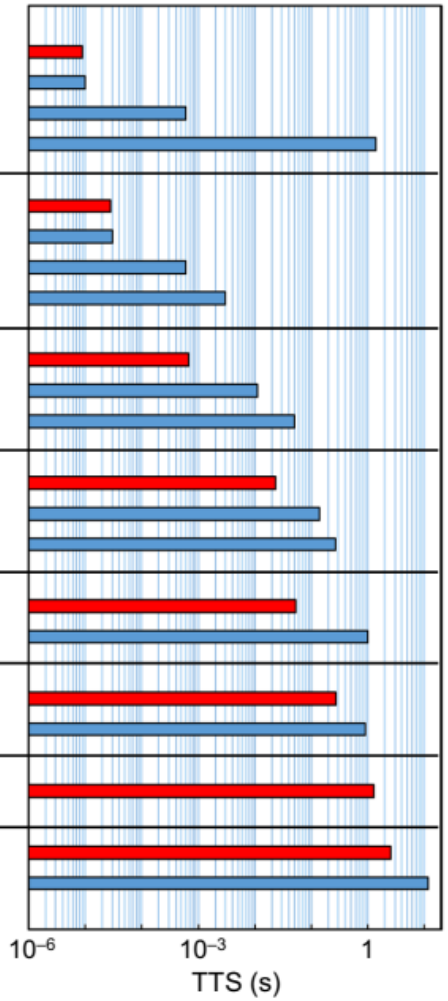
M.H. Yung

Simulated Bifurcation

Goto et al., Sci. Adv. 2019; 5 : eaav2372

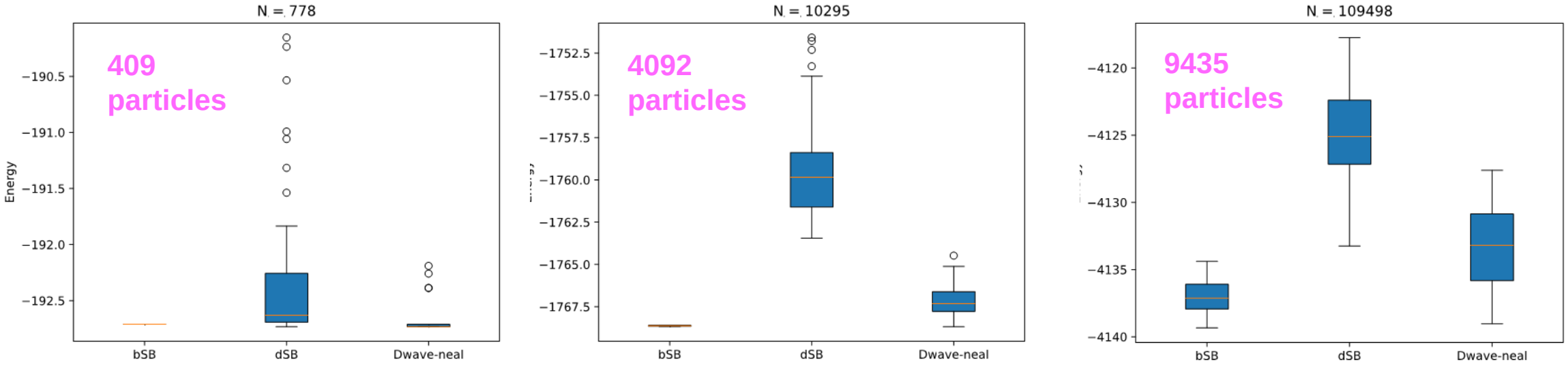


N	Connectivity	$J_{i,j}$	Machine	TTS
60	All-to-all	{±1}	**dSBM**	**9.2 μs**
			RBM	10 μs
			CIM	0.6 ms
			QA	1.4 s
100	All-to-all	{±1}	**dSBM**	**29 μs**
			RBM	30 μs
			SimCIM	0.6 ms
			CIM	3.0 ms
200	Sparse (Degree 3)	{0, −1}	**dSBM**	**0.70 ms**
			QA	11 ms
			CIM	51 ms
700	All-to-all	{±1}	**dSBM**	**25 ms**
			SimCIM	0.14 s
			DA	0.27 s
1024	All-to-all	{±1}	**dSBM**	**55 ms**
			DA	1 s
1024	All-to-all	16 bits $\{-2^{15}+1,\ldots,2^{15}-1\}$	**dSBM**	**0.29 s**
			DA	0.9 s
2000 (K_{2000})	All-to-all	{±1}	**dSBM**	**1.3 s**
2000 (G22)	Sparse (1%)	{0, −1}	**dSBM**	**2.7 s**
			SimCIM	12 s



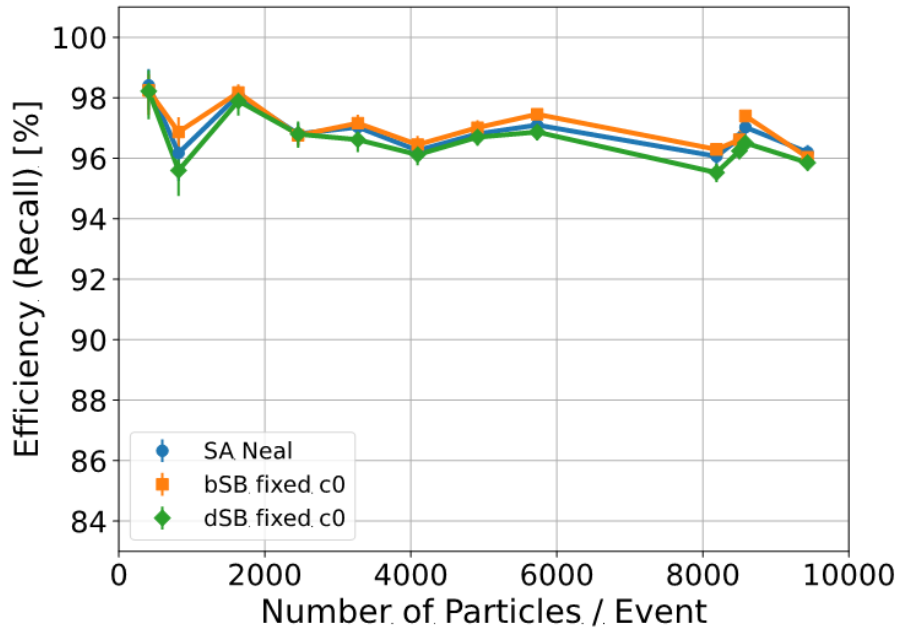
- **Simulated bifurcation is known to outperform other CC algorithms as well as quantum annealing for some problems**
- Simulated Coherent Ising Machine (CIM) had largely degraded performance in our study, so is not presented.

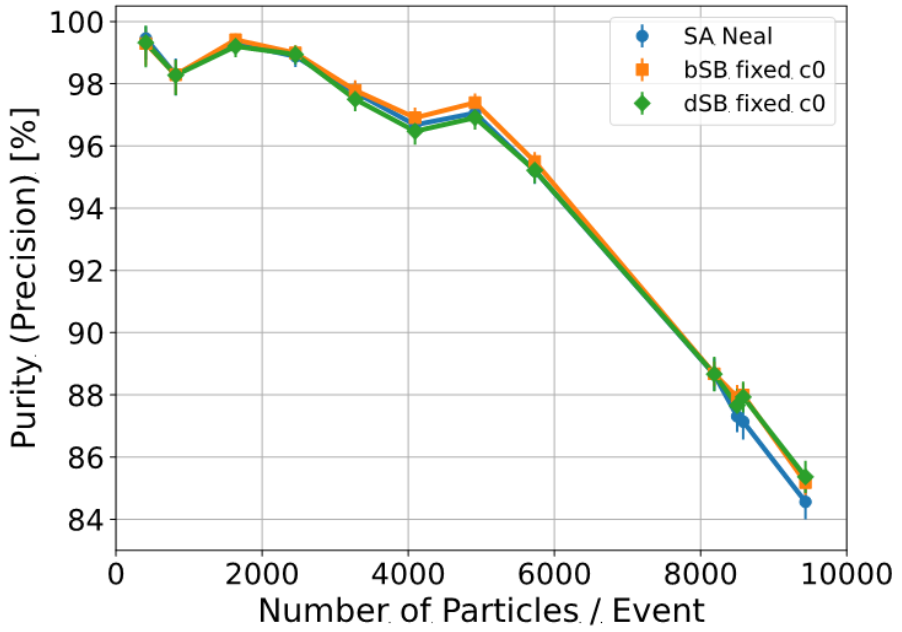
Mimimum Ising Energy Prediction



- Originally proposed adiabatic simulated bifurcation (aSB) is largely outperformed by new versions, so not shown here.
- **Ballistic simulated bifurcation (bSB) provides the best prediction of minimum energy with the least fluctuation**.
- Discrete simulated bifurcation (dSB) is not as good as the other two, but the impact on the reconstruction performance is not significant (next slide)

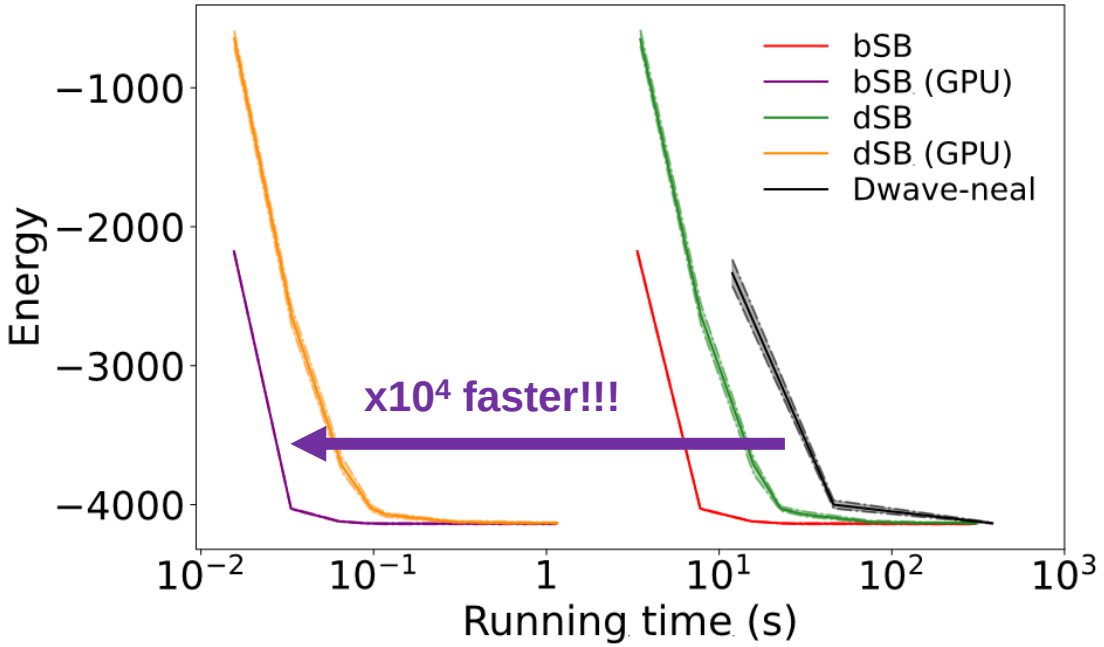
Track Efficiency & Purity w/ SB





- Simulated bifurcation provides **compatible or slightly better performance than D-Wave Neal.**
- **Track efficiency stays over 95%** for all dataset up to the highest HL-LHC conditions
- Purity degrades with track multiplicity but **>90% for <6000 particles, >84% even for ~10000 particles**.

Computation Speed



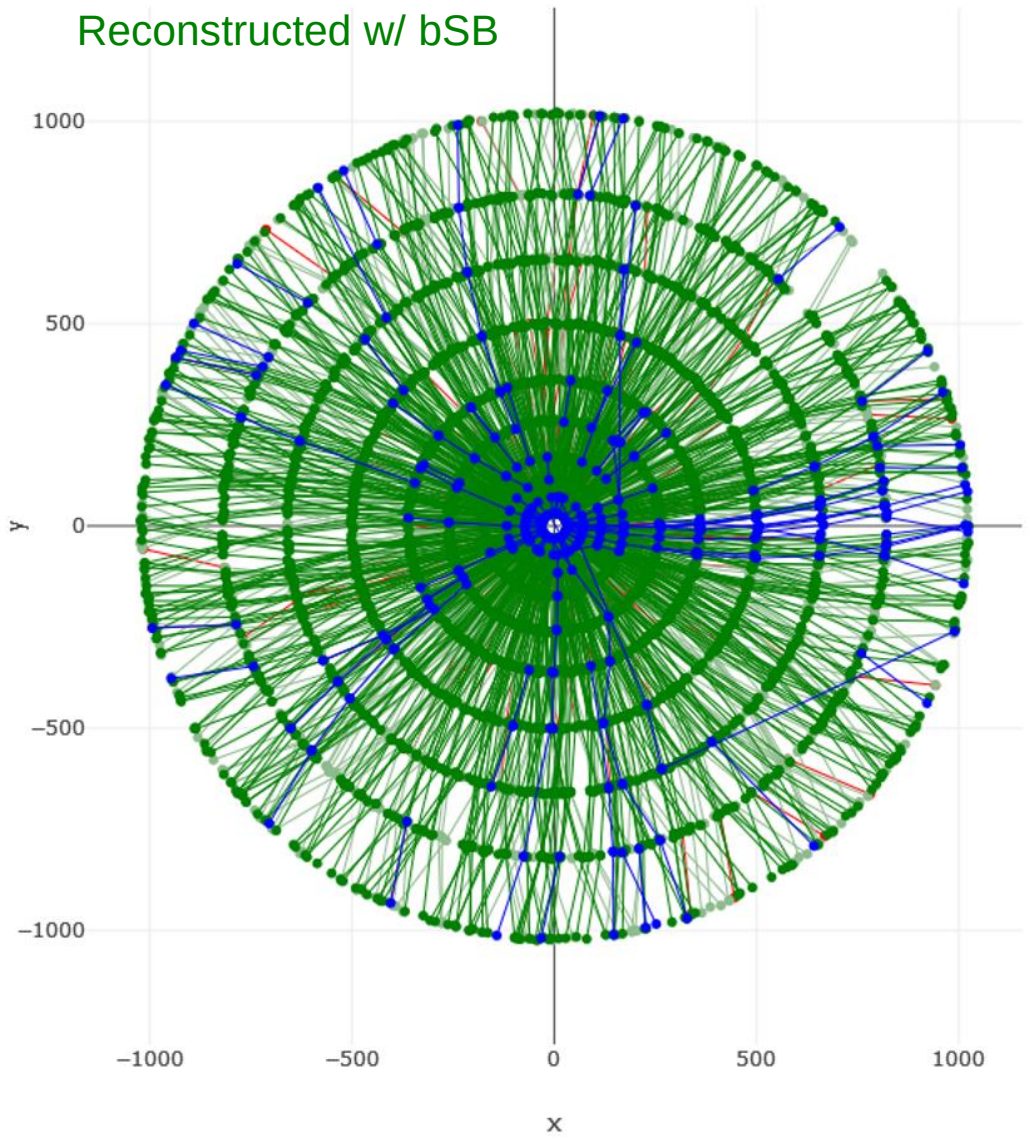
Using only 1 CPU or GPU

Data Information		Time to target [s]				
# of particles	QUBO size	bSB	bSB (GPU)	dSB	dSB (GPU)	D-Wave Neal
409	778	0.007	0.021	0.032	0.092	0.060
818	1431	0.012	0.019	0.293	0.478	0.169
1637	2904	0.012	0.019	0.293	0.478	0.169
2456	4675	0.014	0.017	–	–	0.479
3274	6945	0.032	0.022	–	–	1.229
4092	10295	0.005	0.022	0.015	0.065	0.030
4912	14855	0.027	0.016	–	–	2.165
5730	22022	0.109	0.042	–	–	3.853
8187	67570	0.488	0.028	–	–	404.297
8500	78812	1.899	0.108	–	–	785.732
8583	80113	1.321	0.067	–	–	93.782
9435	109498	3.884	0.140	–	–	1366.808

- Ballistic simulated bifurcation provides **4 orders of magnitude speed-up (23 min → 0.14s)** at most, compared to D-Wave Neal. → More speed-up expected with larger data size.
- Unlike D-Wave Neal, **simulated bifurcation can effectively run w/ multiple processing & GPU → Perfect match with HEP computing environment!!**

Summary

- Tracking is the highest CPU-consuming reconstruction task at the HL-LHC era.
- A computing leap from quantum machine learning would be highly exciting.
- Presented recent results on the quantum tracking using two complementary approaches: CQ approach (QAOA+subQUBO) & CC approach (quantum-annealing inspired algorithms).
- CQ approach: Promising tracking performance from the real quantum hardware.
- CC approach: Quantum-annealing inspired algorithms provide **four orders of magnitude speed-up** in solving QUBO at most (& more speed-up expected w/ larger dataset) from D-Wave Neal & **can already be considered for implementation**.
- Further studies are ongoing. Stay tuned!

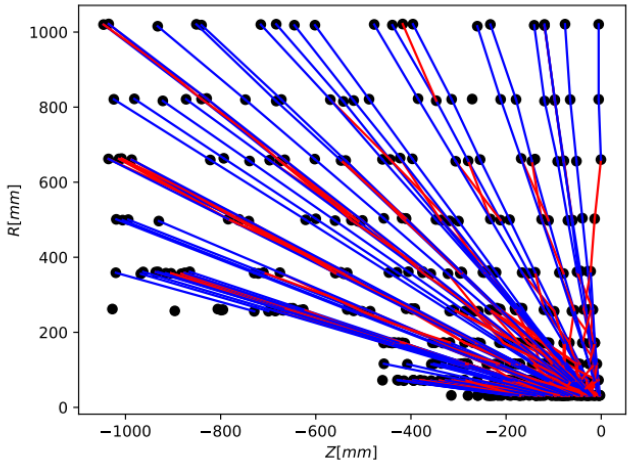


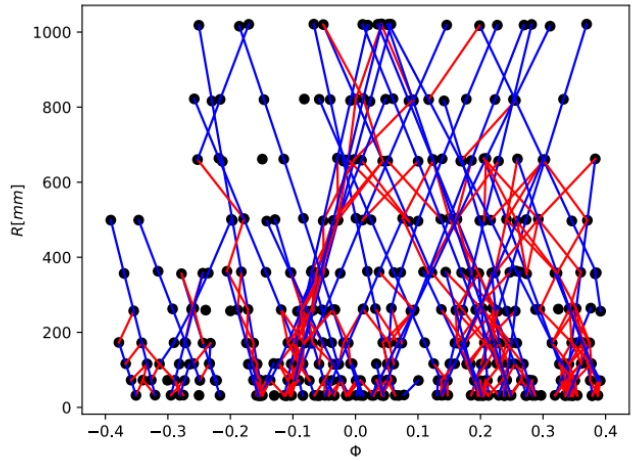
Thank you for listening!

感谢聆听！

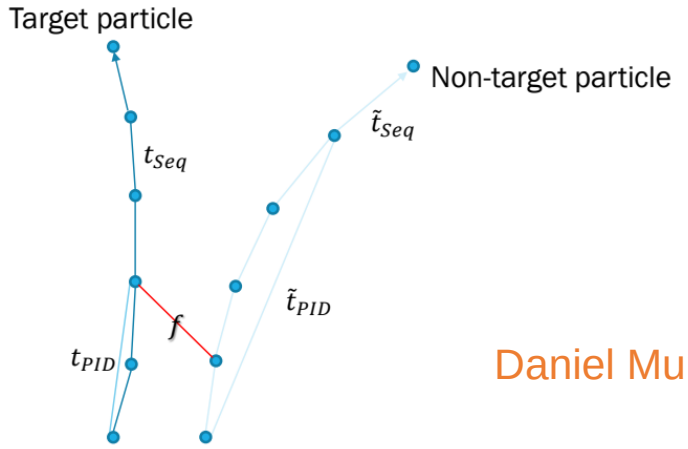
Backup

Classical ML Approaches



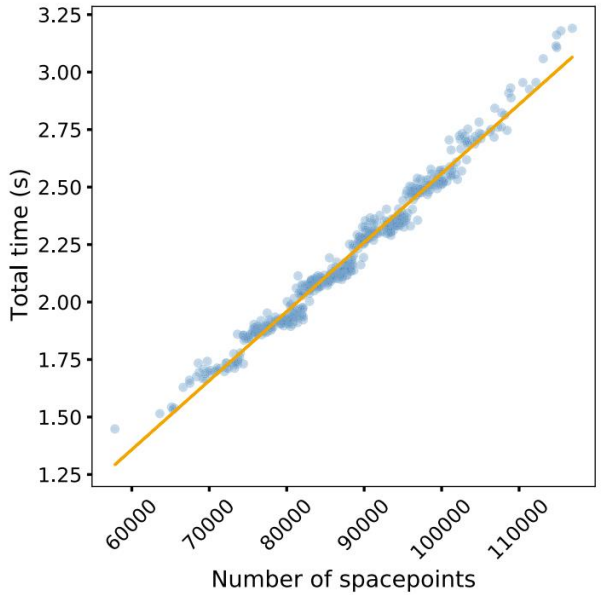


Cenk Tueysuez



Daniel Murnane

- Graph neural network (GNN) is actively investigated in the LHC [Project Exa.TrkX] & BES-III communities.
  - There are also studies using CNN & Point Net at BES-III
- Silicon hits can be regarded as "**nodes**" & connected segments as **"edges"**
- Computing time scales linearly with number of tracks



QUBO

Lucy Linder's Master thesis

$$O(a,b,T) = \sum_{i=1}^{N} a_i T_i + \sum_{i=1}^{N} \sum_{j<i}^{N} b_{ij} T_i T_j,$$

$$S_{ij} = \frac{1 - \frac{1}{2}(|\delta(q/p_{Ti}, q/p_{Tj})| + max(\delta\theta_i, \delta\theta_j))}{(1 + H_i + H_j)^2},$$

$$a_i = \alpha \left(1 - e^{\frac{|d_0|}{\gamma}}\right) + \beta \left(1 - e^{\frac{|z_0|}{\lambda}}\right),$$

- b_{ij} = 0 (if no shared hit), = 1 (if conflict), = -S_{ij} (if two hits are shared)
- α, β , γ and λ are tunable parameters, taken to be 0.5, 0.2, 1.0 and 0.5

$$QUBO = \sum_i a_i x_i + \sum_{i>j} b_{ij} x_i x_j$$

Multiple Solution Instances

- 3 parameters (N_I, N_E, N_S) in this sub-QUBO method.
- Extract N_I quasi-optimal solutions from full-QUBO classically.
- Randomly select N_S solution instances from N_I.
- Focus on particular binary variable x_i. Rank them in accordance to how much they vary over N_S solution instances. Highly varying x_i will be included in the sub-QUBO model.
- Pick-up process of N_S solution from quantum computing is repeated N_E times & N_E sub-QUBO models are considered.
- Returns a pool of N_I solutions & the best solution will be chosen.

Y. Atobe, M. Tawada, N. Togawa, IEEE Trans. Comp. 71, 10 (2022) 2606

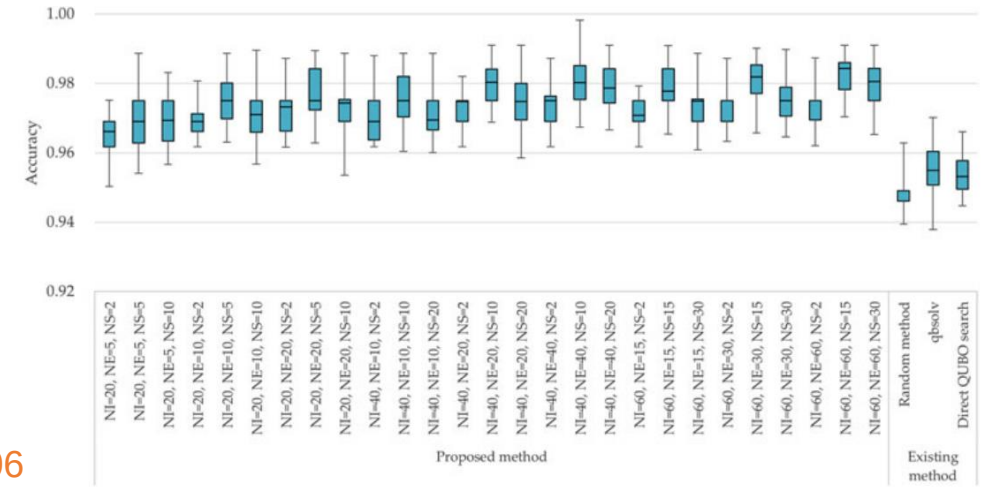
Sub-QUBO Methods

Algorithm 2. Proposed Hybrid Annealing Method

procedure PROPOSED METHOD WITH MULTI-INSTANCES
  for $(i = 1; i \leq N_I; i++)$ **do**
    $X_i \leftarrow$ Initialize(QUBO)
    Pool $\leftarrow$ AddInstancePool(Pool, X_i)
  $X_{best} \leftarrow$ FindBest(Pool)
  while not converged **do**
    for $(i = 1; i \leq N_I; i++)$ **do**
      $X_i \leftarrow$ Optimize(QUBO, X_i)
                                ▷ Using a classical computer
    for $(i = 1; i \leq N_E; i++)$ **do**
      $X_1, X_2, \cdots, X_{N_S} \leftarrow$ SelectInstance(Pool, N_S)
      for $(j = 1; j \leq n; j++)$ **do**
        for $(k = 1; k \leq N_S; k++)$ **do**
          $c_j \leftarrow c_j + x_{k,j}$
        $d_j \leftarrow |c_j - \frac{N_S}{2}|$
      subQUBO $\leftarrow$ Extract(ArgSort($d_1, d_2, \cdots, d_n$), m, X_t)
      $X' \leftarrow$ Optimize(subQUBO, X_t)
                                ▷ Using an Ising machine
      Pool $\leftarrow$ AddInstancePool(Pool, X')
    $X_{best} \leftarrow$ FindBest(Pool)
    Pool $\leftarrow$ ArrangeInstancePool(Pool, N_I)
  return $f(X_{best}), X_{best}$

Algorithm 4. Qbsolv[10]

procedure QBSOLV
  $X \leftarrow$ Initialize(QUBO)
  $X_{best} \leftarrow$ TabuSeach(QUBO, X)
  index $\leftarrow$ OrderByImpact(QUBO,X_{best})
  while not converged **do**
    for $(i = 0; i <$ Size(QUBO); $i+ =$ Size(subQUBO)) **do**
      subQUBO $\leftarrow$ Decompose(QUBO,
                index[i : i+Size(subQUBO)-1], X_{best})
      subX $\leftarrow$ Optimize(subQUBO, X_{best})
      X[index[i : i+Size(subQUBO)-1]] $\leftarrow$ subX
    $X \leftarrow$ TabuSearch(QUBO, X)
    index $\leftarrow$ OrderByImpact(QUBO, X)
    if $f(X) < f(X_{best})$ **then**
      $X_{best} \leftarrow X$
  return $f(X_{best}), X_{best}$



Y. Atobe, M. Tawada, N. Togawa, IEEE Trans. Comp. 71, 10 (2022) 2606