量子コンピューティング Trackingへの応用

寺師 弘二
ICEPP 東京大学

テラスケール研究会
2019年6月8日

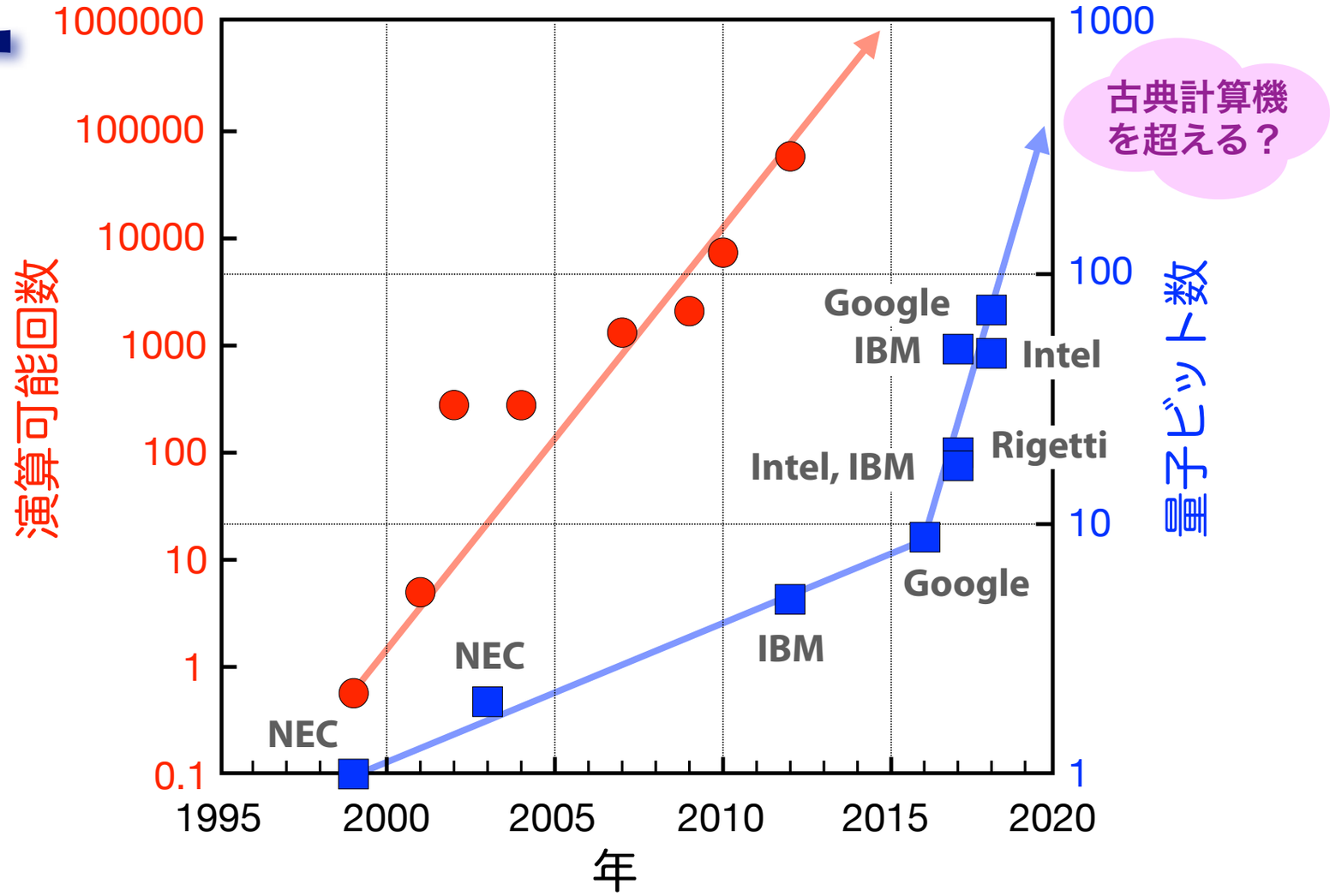
ML & QC at ICEPP (link)

量子コンピューティング

量子コンピュータ技術が近年急速に進展し、**通常計算機の性能を超える**可能性が指摘され始めている

Quantum Supremacy

*~50*量子ビットで到達*?!*



量子計算が優位性を持つアルゴリズムはいくつか知られている

- 素因数分解 (Shor 1994), データベース探索 (Glover 1996) など

➡ ハードウェア性能やアルゴリズムの実装手法によるため、実際の能力は不明

非常に多くの量子ビット($>\sim10^6$) も必要になる

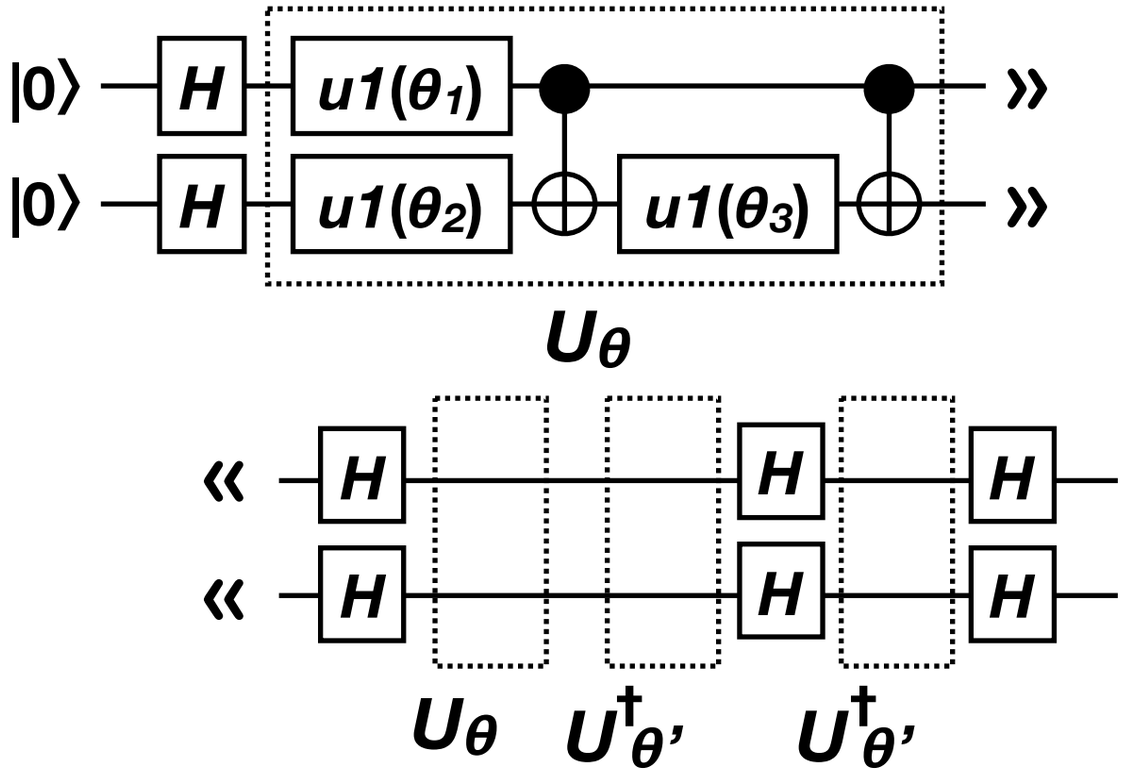
量子コンピューティングは高エネルギー実験にとって有用か？

高エネルギー実験への応用で量子加速は実現可能か？

量子コンピューティング

量子コンピュータという場合、大きく二つの計算方式に分類される

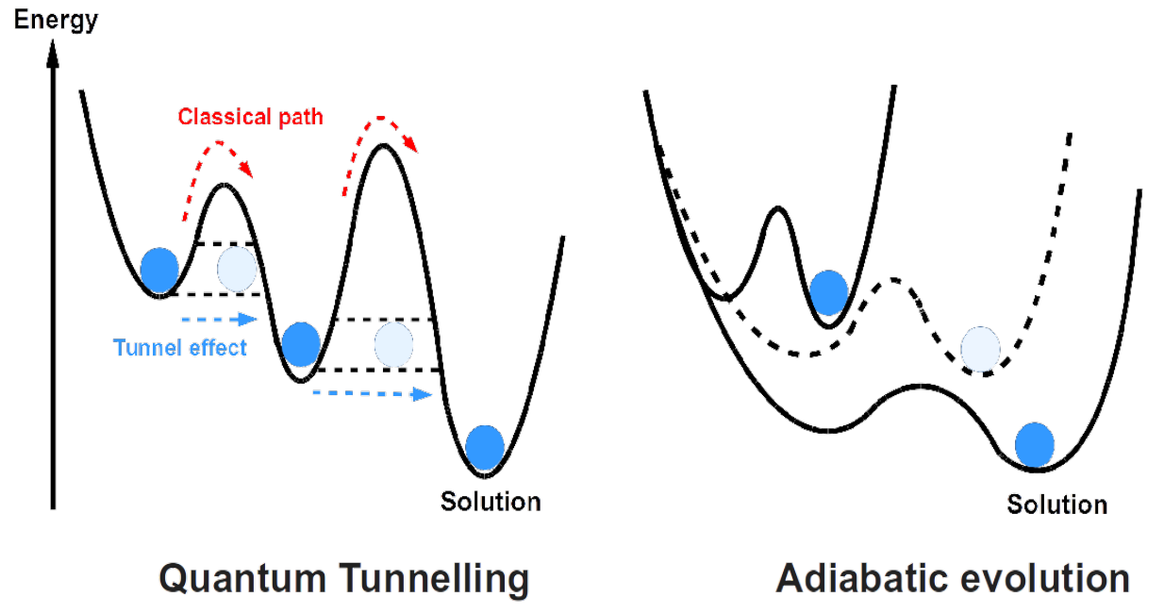
量子ゲートコンピュータ



- 解きたい問題ごとに量子回路(アルゴリズム)を構成する
- 多様な問題に対応可能

IBM, Google, Intel, …

量子アニーリング



- 系のハミルトニアンを変化させ、解 (= エネルギーの最小値) を求める
- 最適化問題に適している

D-Wave

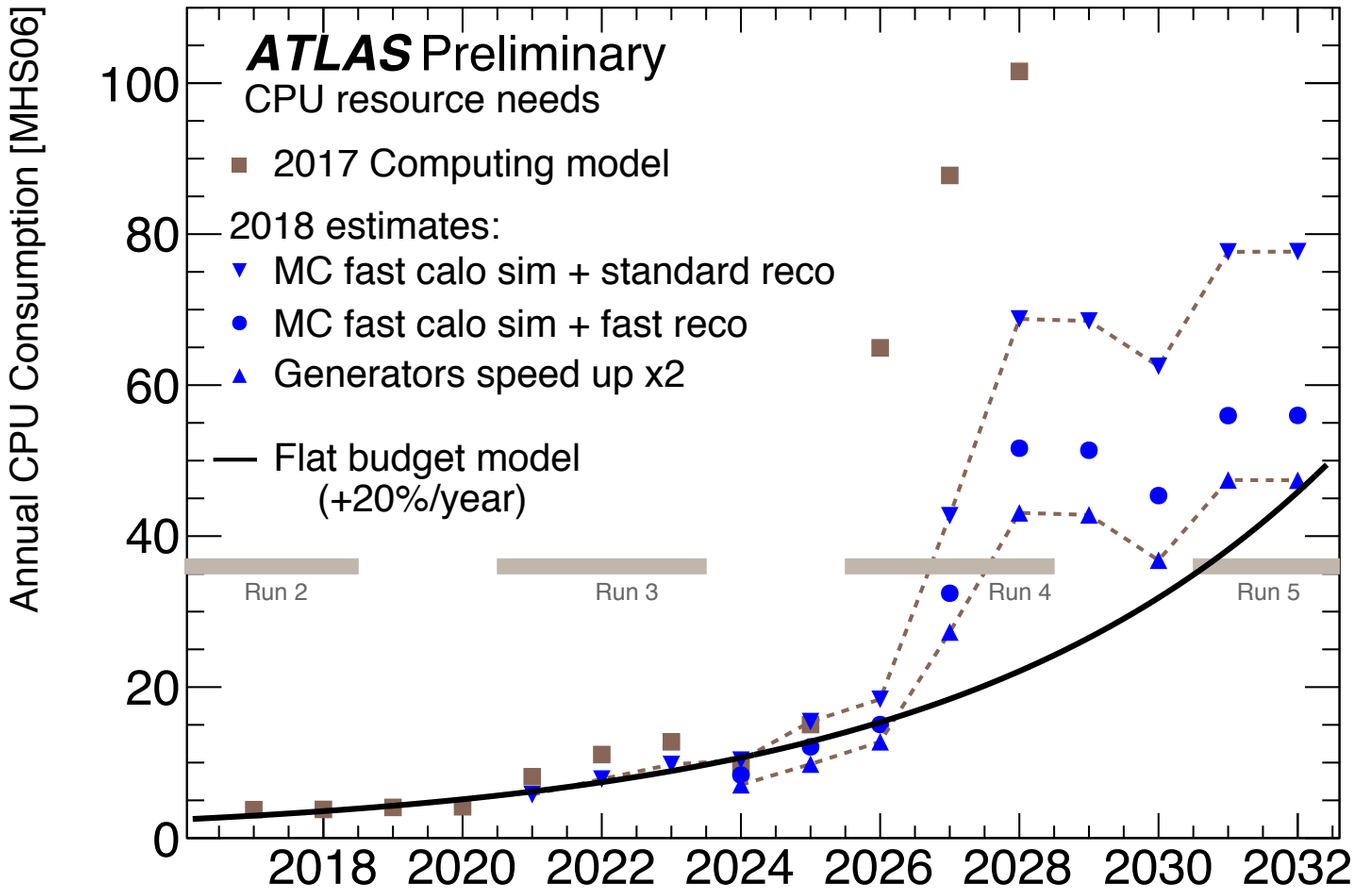
Trackingへの応用?

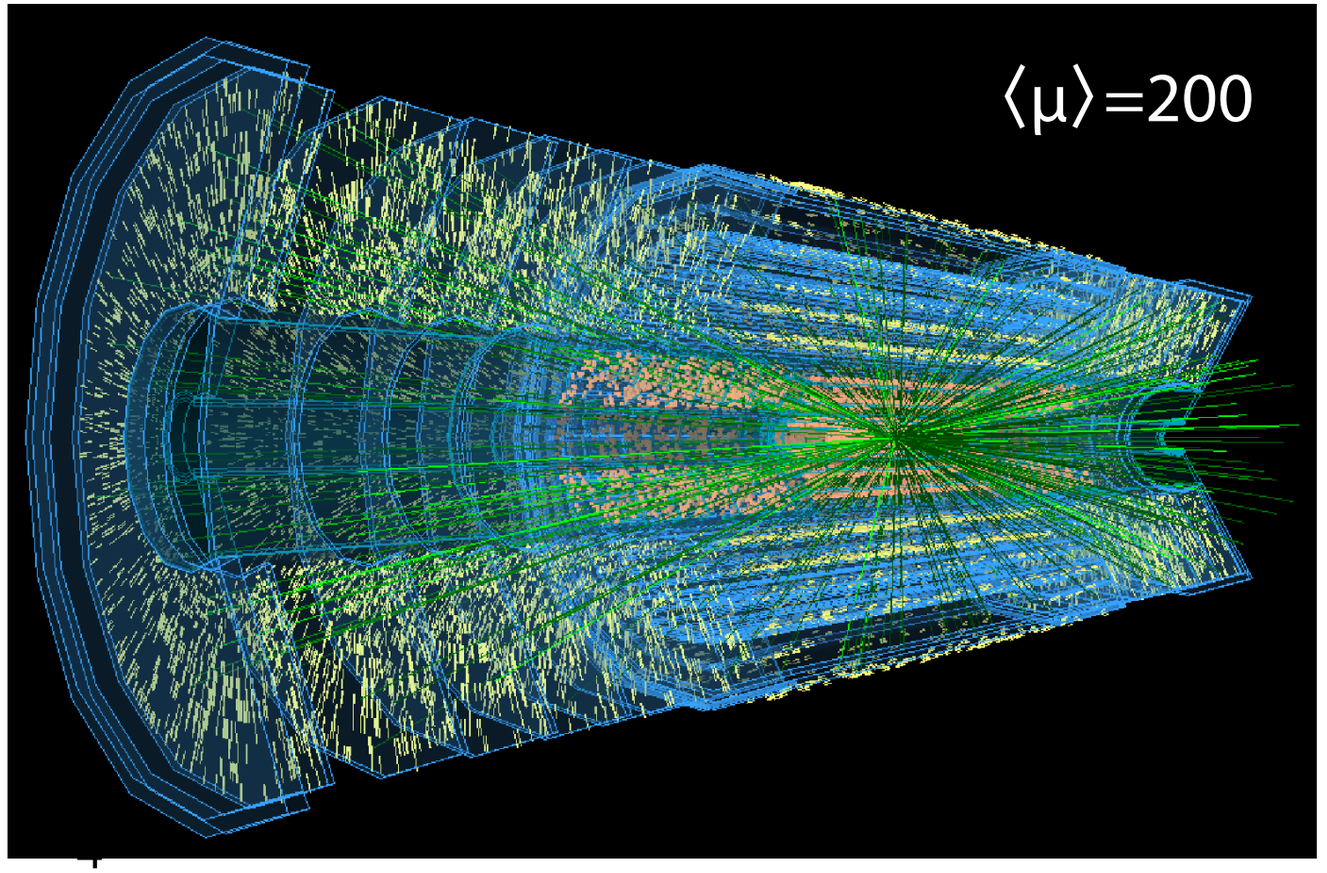
- 加速器の高輝度化
- 検出器読み出しの高精細化
  (チャンネル数の増加)

➡ HL-LHCでは、利用可能な計算機能力の~3-5倍の計算資源が必要

計算資源の多くは、飛跡を含む事象の再構成に使われる

➡ 飛跡の再構成に量子技術の応用ができないか？





Trackingへの応用?

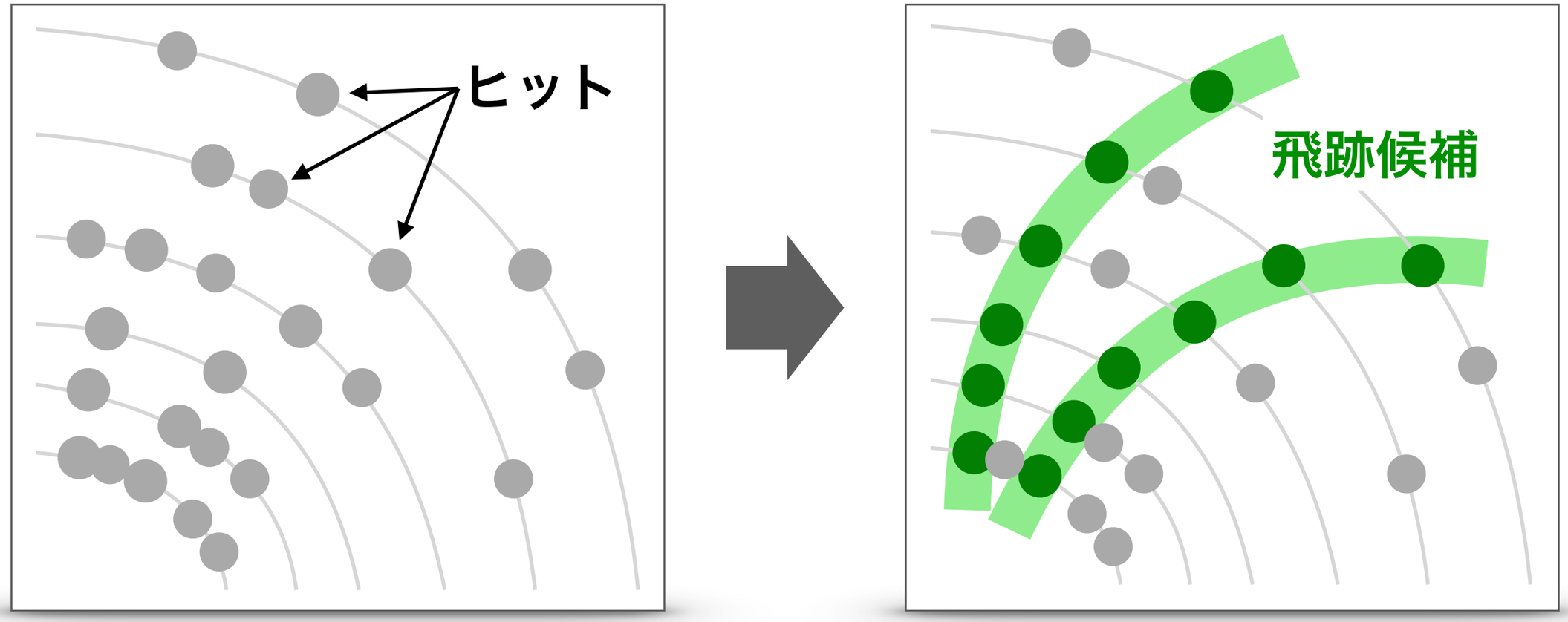
検出器が作る膨大なヒットから飛跡候補となるヒットの組み合わせを選ぶ

- HL-LHC : 10^4個の生成粒子 → O(10^5)の検出器ヒット

同じ粒子に属するヒットは一定の規則性 (パターン) を持って記録される

- **この規則性を元に、正しい組み合わせを優先的に選別する**
- **さまざまな組み合わせを並列に計算することで、高速選別が可能**

(かもしれない) ➡ **量子加速の可能性**

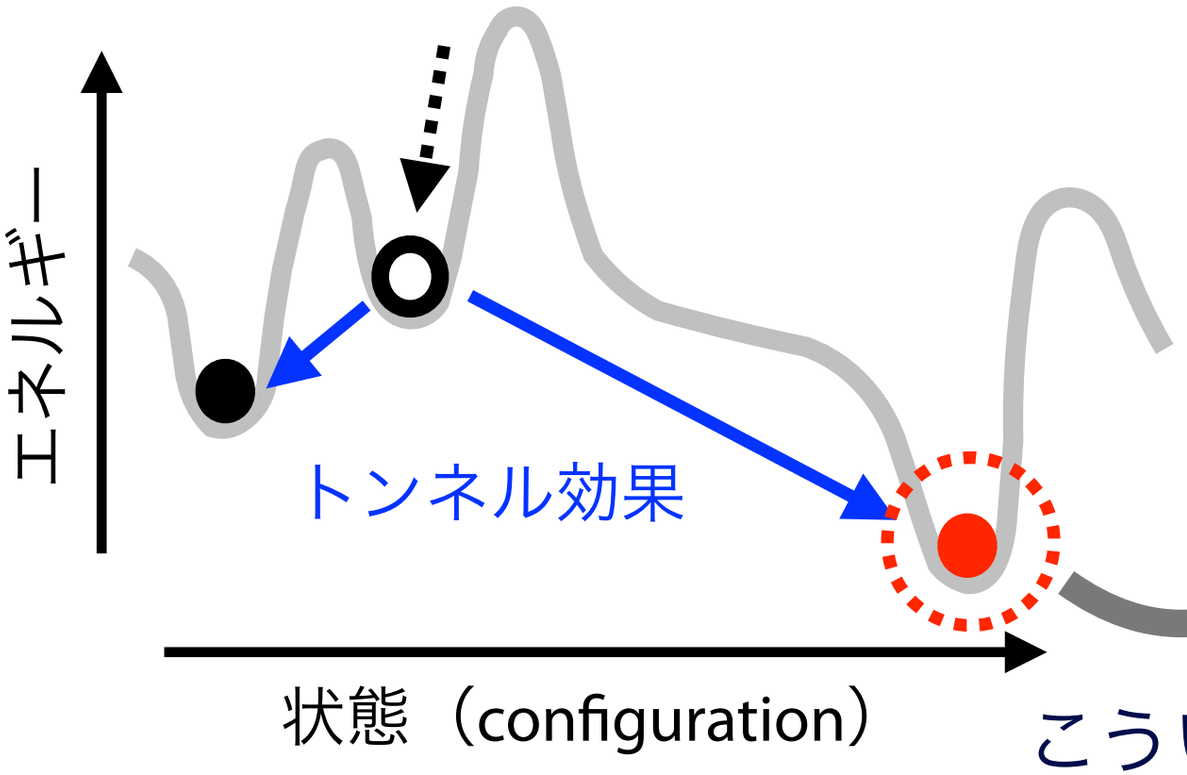


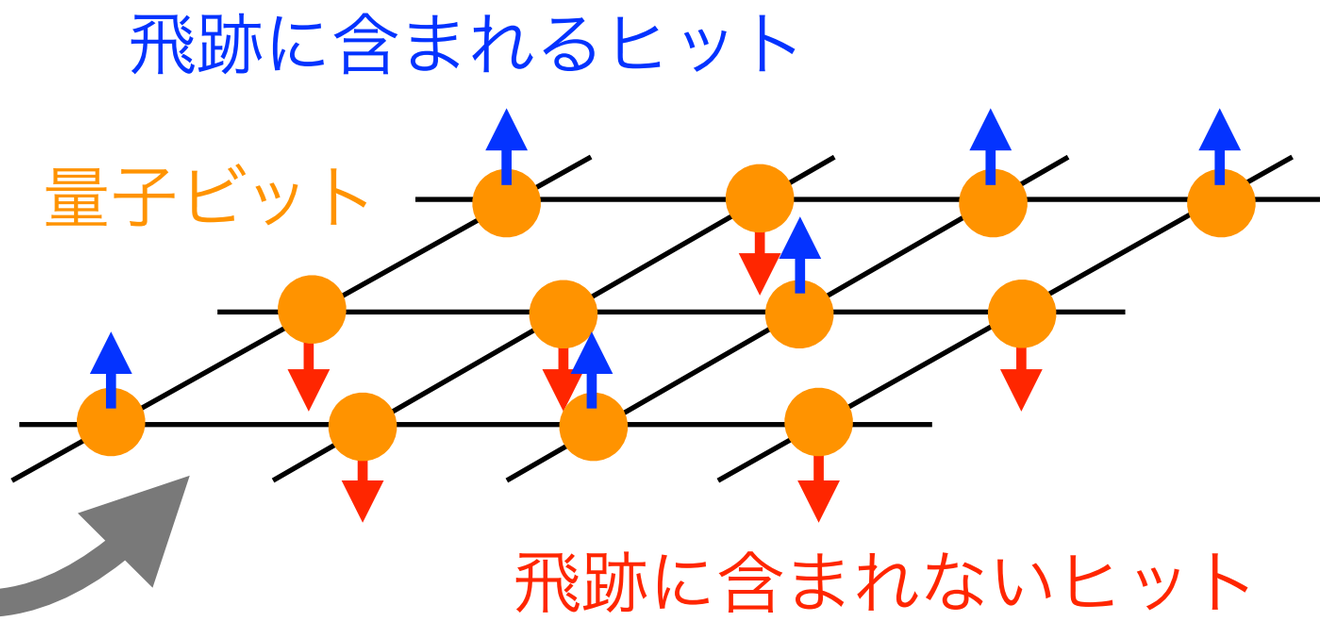
量子アニーリングでの飛跡再構成

どのように飛跡候補となるヒットの組み合わせを選別するか？

➡ **正しい組み合わせを持つ状態が最も小さいエネルギーを持つ**ような問題を設定する

量子アニーリングの場合、トンネル効果でより高い確率で最適解に到達する(と期待)





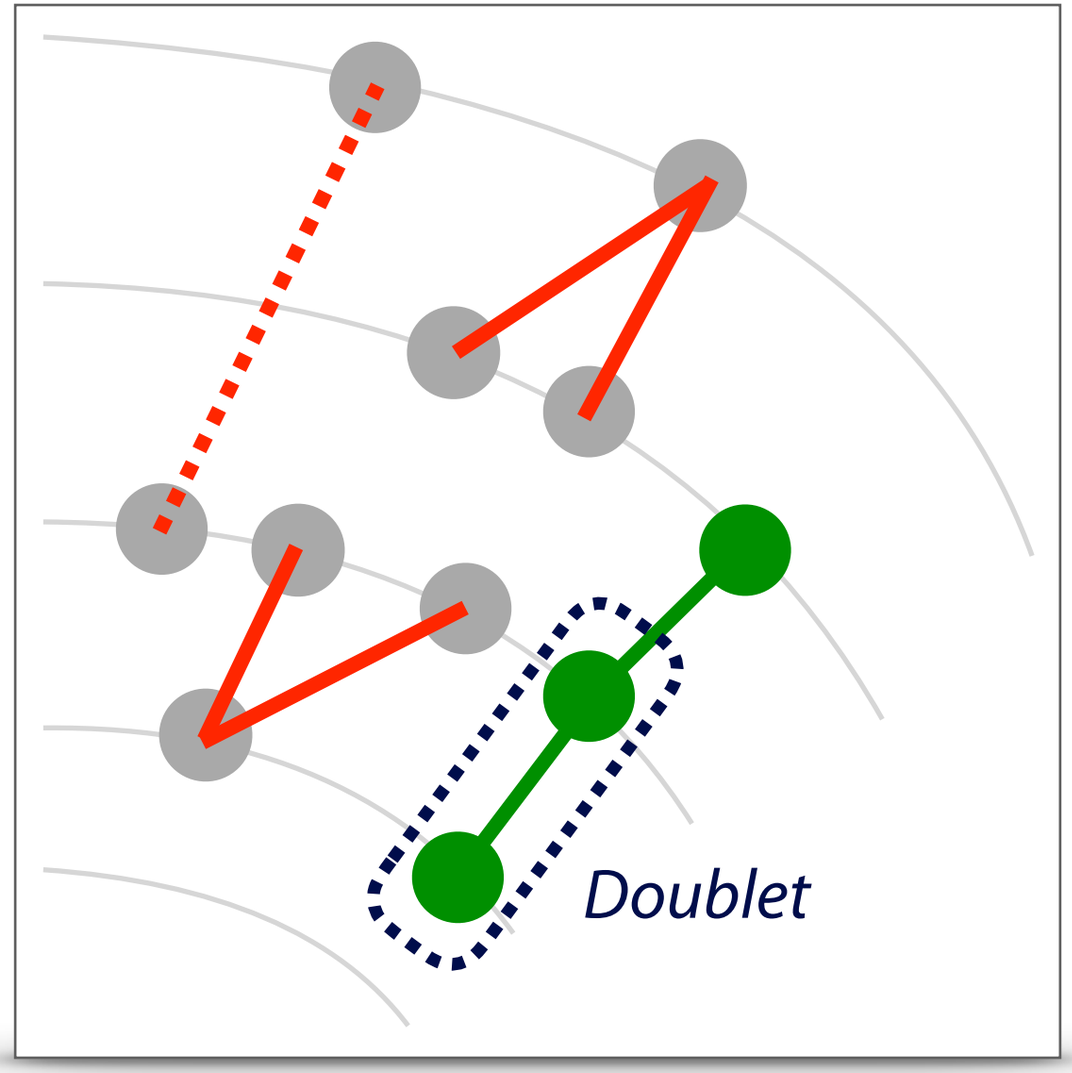
こういう状況を作りたい!

飛跡の構成要素として、何を量子ビット (スピン) に対応させるか？

ハミルトニアンモデルの設定

Stimpfl-Abele, Garridoモデル (1991)

- ニューラルネットワーク
- 飛跡の構成要素として*Doublet* (2層ヒット)を使う → *Doublet*をニューロンに設定
- ヒットを共有する*Doublet*やホールがある*Doublet*には高いコストを与える
- 連続する*Doublet*には低いコストを与える
- この条件で最小エネルギーの状態を求める



ただしこのモデルではいくつかの限界がある

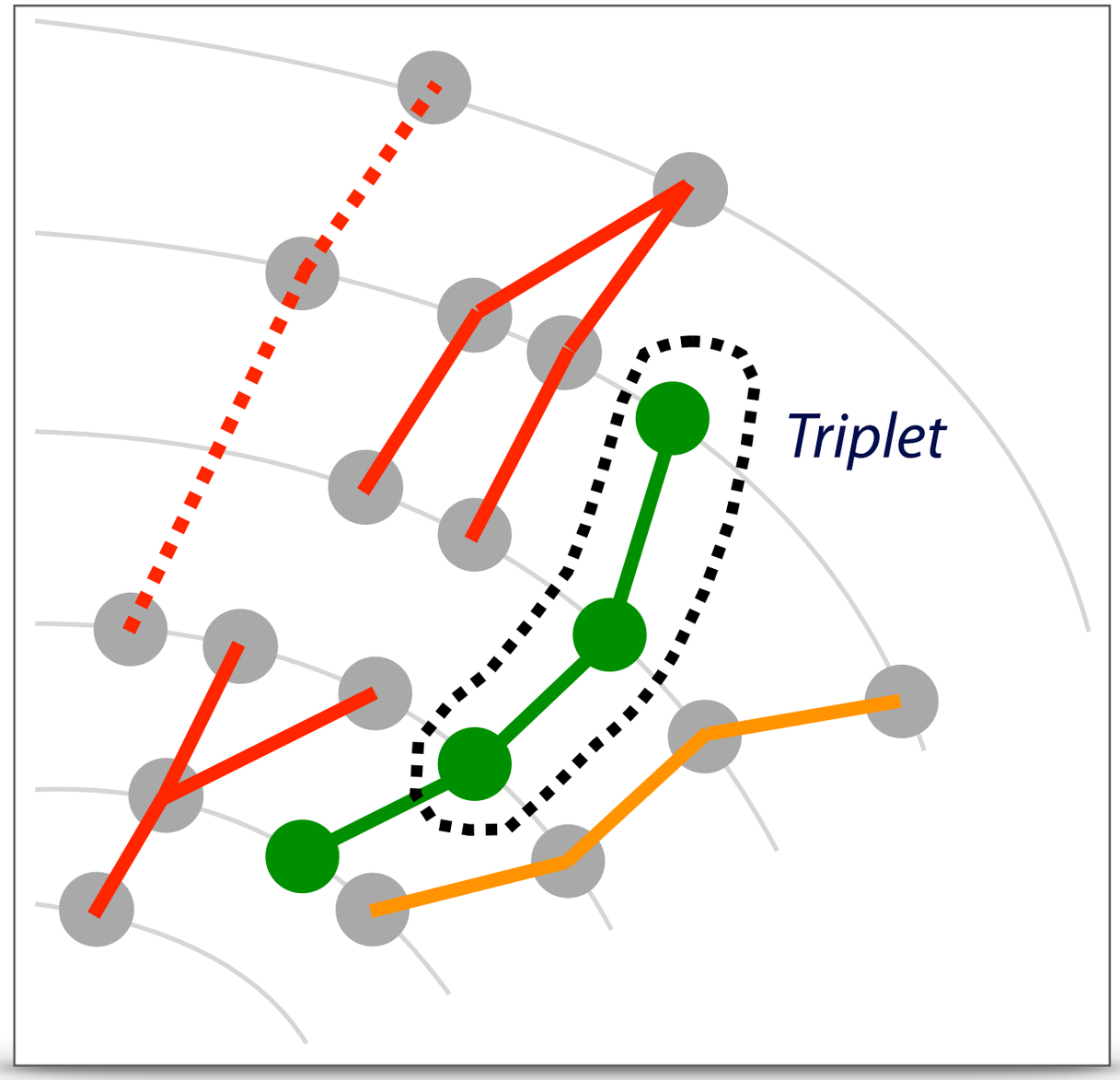
- *Doublet*数を抑えるために、Primary Vertexの位置 (0,0,0) を仮定してヒットを選別している
- 曲率が定義できないため、ジグザグの飛跡ができる

改良したアニーリングモデル

Triplet (3層ヒット) を量子ビットに設定することで問題を解消

→ **Qallse**

- L. Linder & LBNLグループが開発 (HEP.QPR)
- 共有ヒットの取り扱いはほぼ同じ
- 曲率が計算できるため、ジグザグな候補には高いコストの設定が可能



Quadratic Unconstrained Binary Optimization (QUBO)形式のハミルトニアンで表現

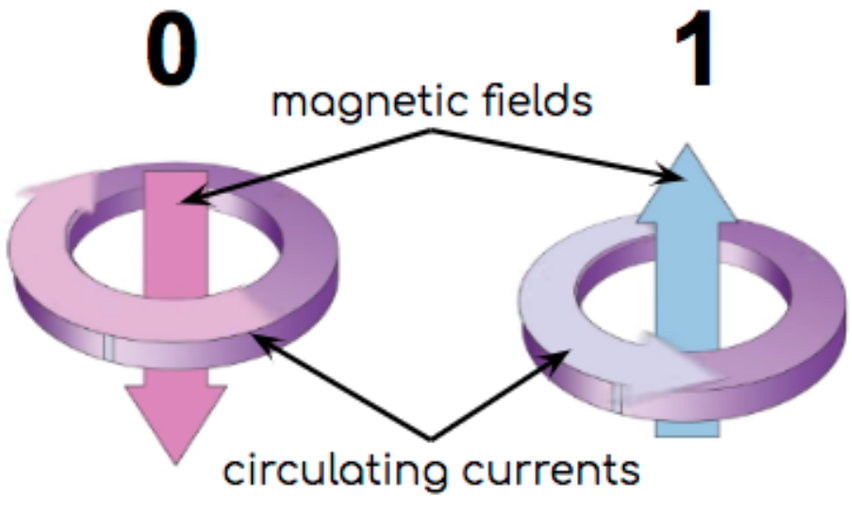
$$\mathcal{O}(a;b;T) = \sum_{i}^{N} a_i T_i + \sum_{i}^{N}\sum_{j>i}^{N} b_{ij} T_i T_j \quad T \in \{0,1\}$$

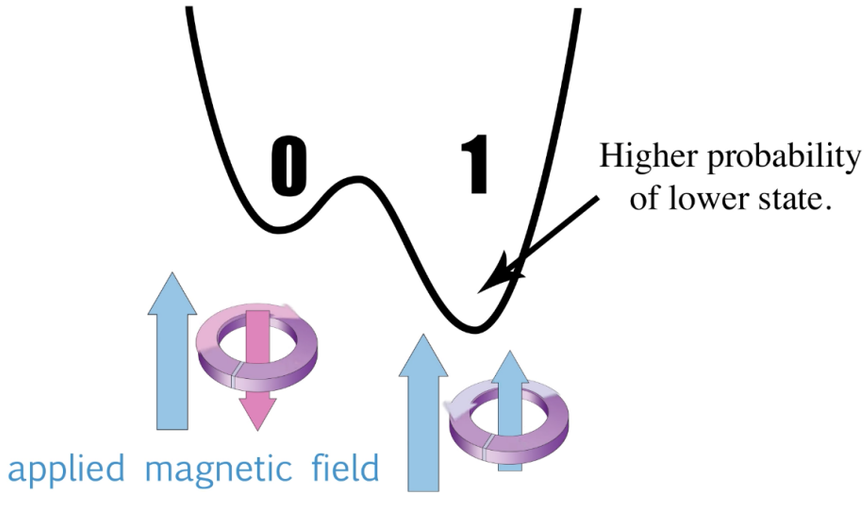
$$a_i = -0.1,\ b_{ij} = \begin{cases} -S(T_i,T_j) & \text{緑} \\ 1 & \text{赤} \\ 0 & \text{それ以外} \end{cases}$$

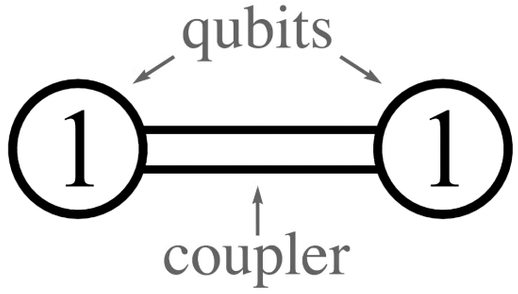
D-Wave

量子アニーリングマシン

- 超電導量子ビット
- 2048量子ビット (*D-Wave 2000Q*)
- キメラグラフ接続 :
  - *16×16 units, 8 qubits/unit → 6016 couplers*
  - *64*量子ビットの全結合に相当
- アニーリング時間 ~ *20 μs*
  (コヒーレント時間は*O(0.1) μs* 程度？)





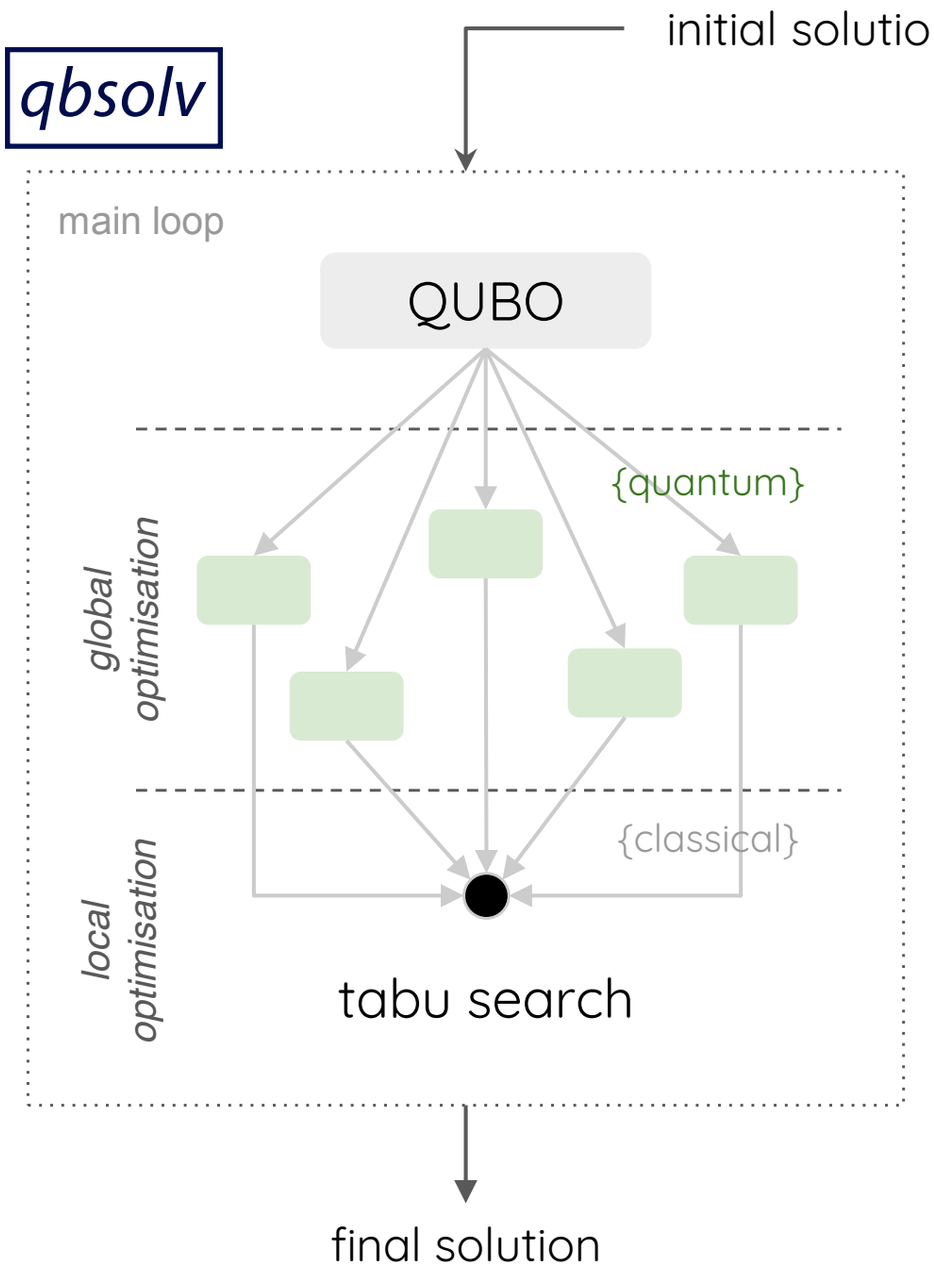


D-Waveでの飛跡再構成

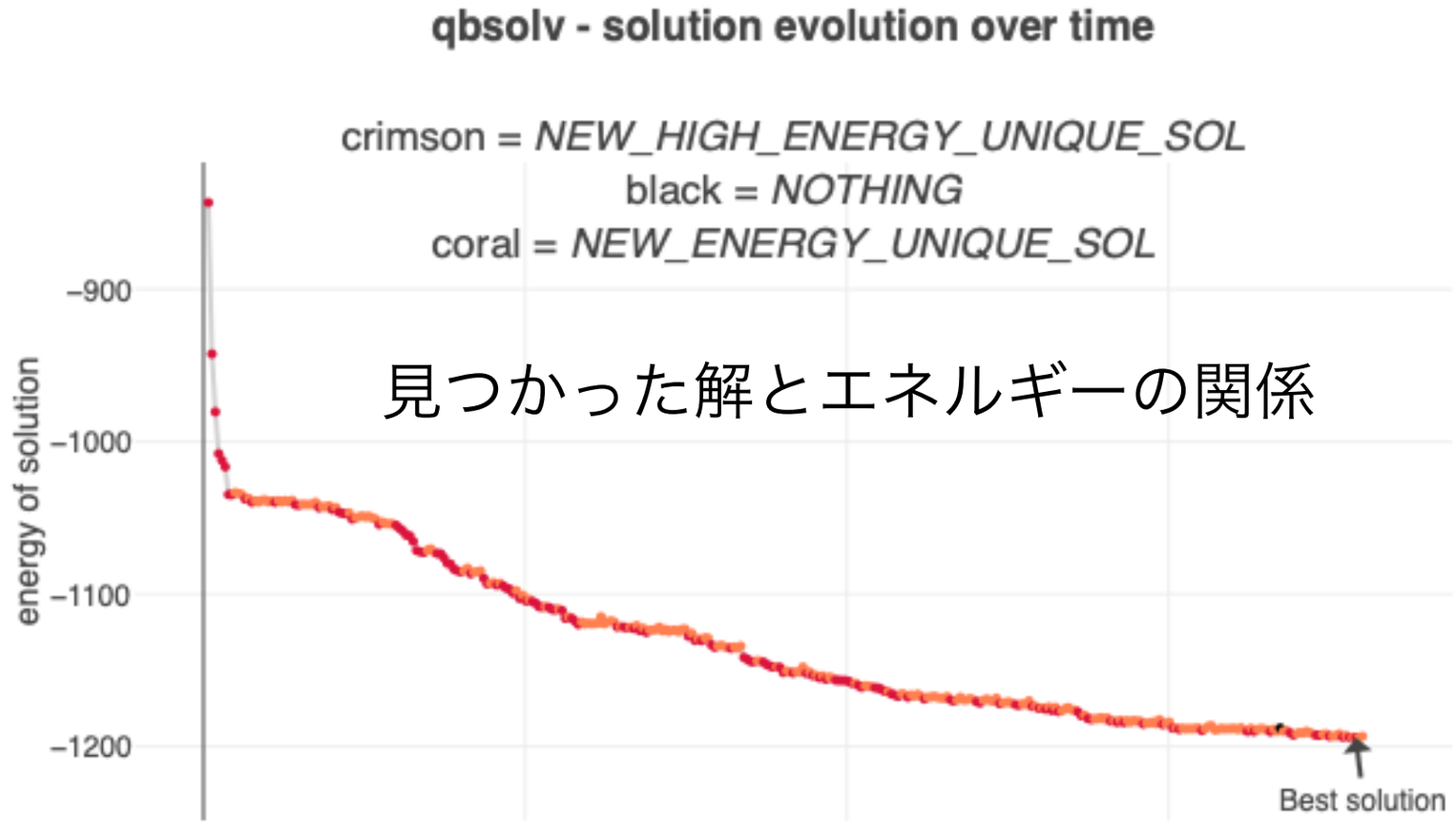
大きなQUBO問題はハードウェアに収まる大きさに分割して対応 (*qbsolv*)

- 経験的には、~160粒子程度までは分割しなくてもOK

もうこれ以上エネルギーを更新できない状態までアニーリングを繰り返す



~4900粒子の場合



見つかった解とエネルギーの関係

最も良い解に対応する*Doublet*を選ぶことで、飛跡候補が得られる

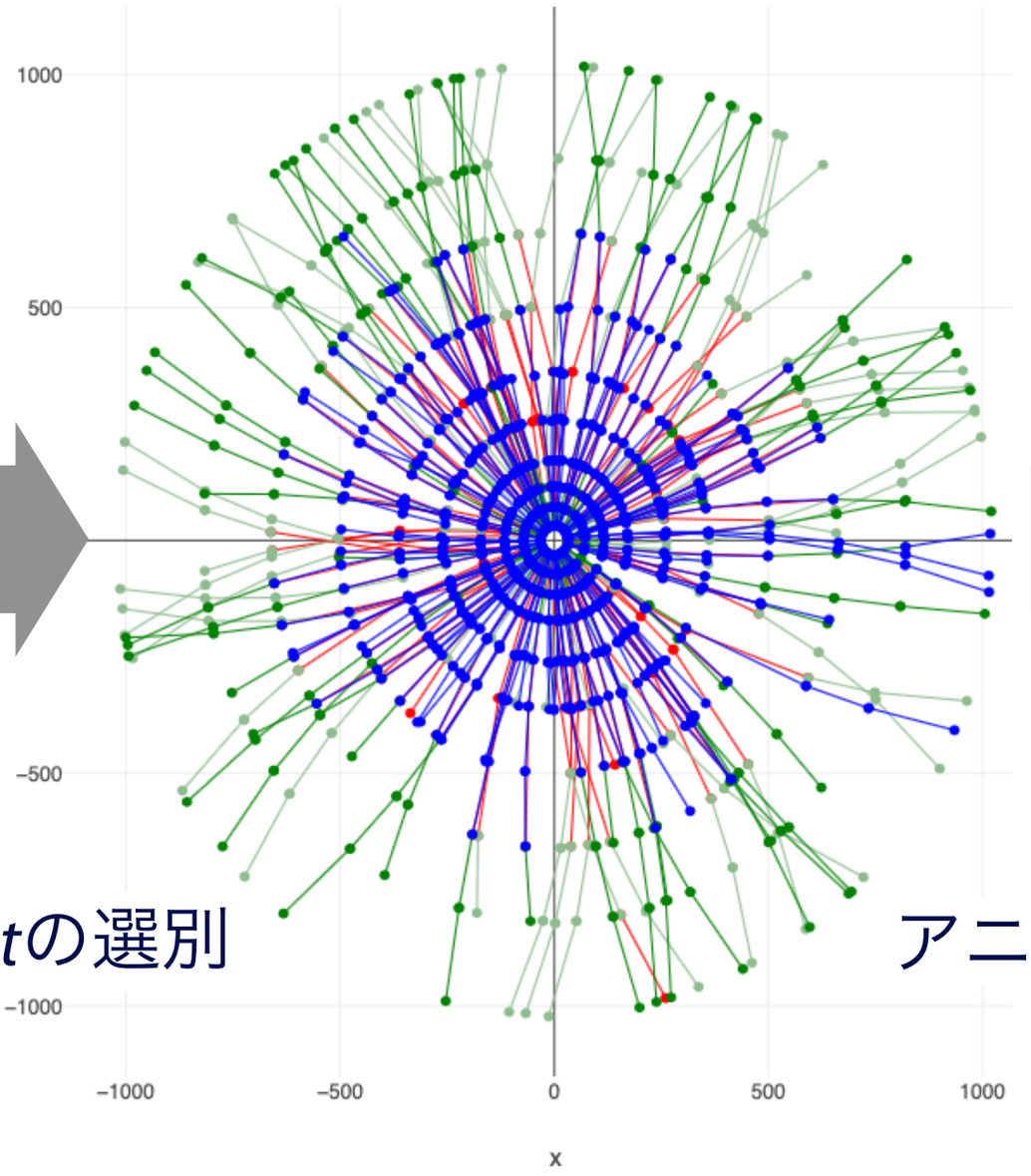
結果

~1600粒子 (11000ヒット)の場合

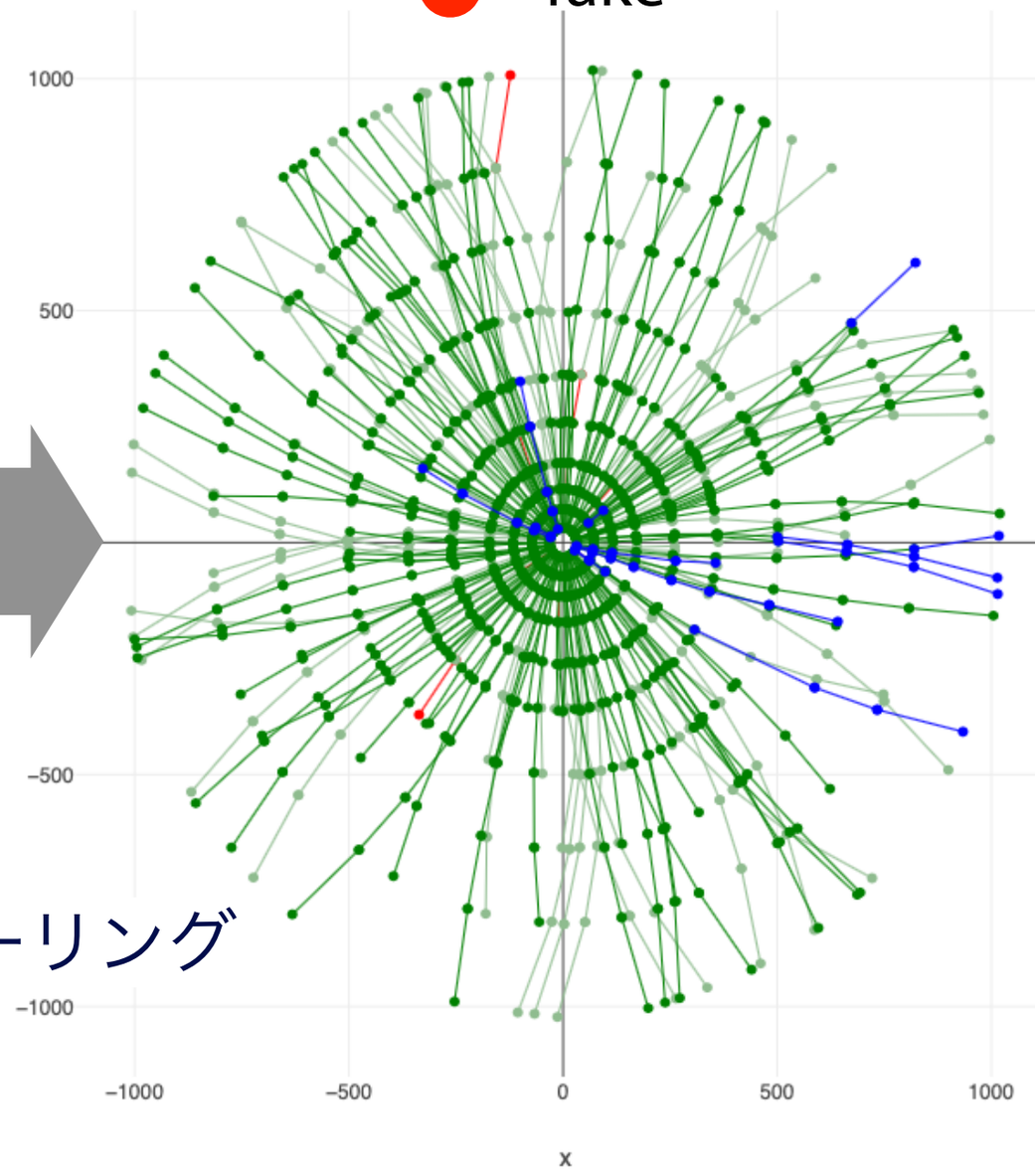
*TrackML*のデータを使用(→バックアップ)

real (focused)
real (unfocused)
missing
fake

入力



*Doublet*の選別



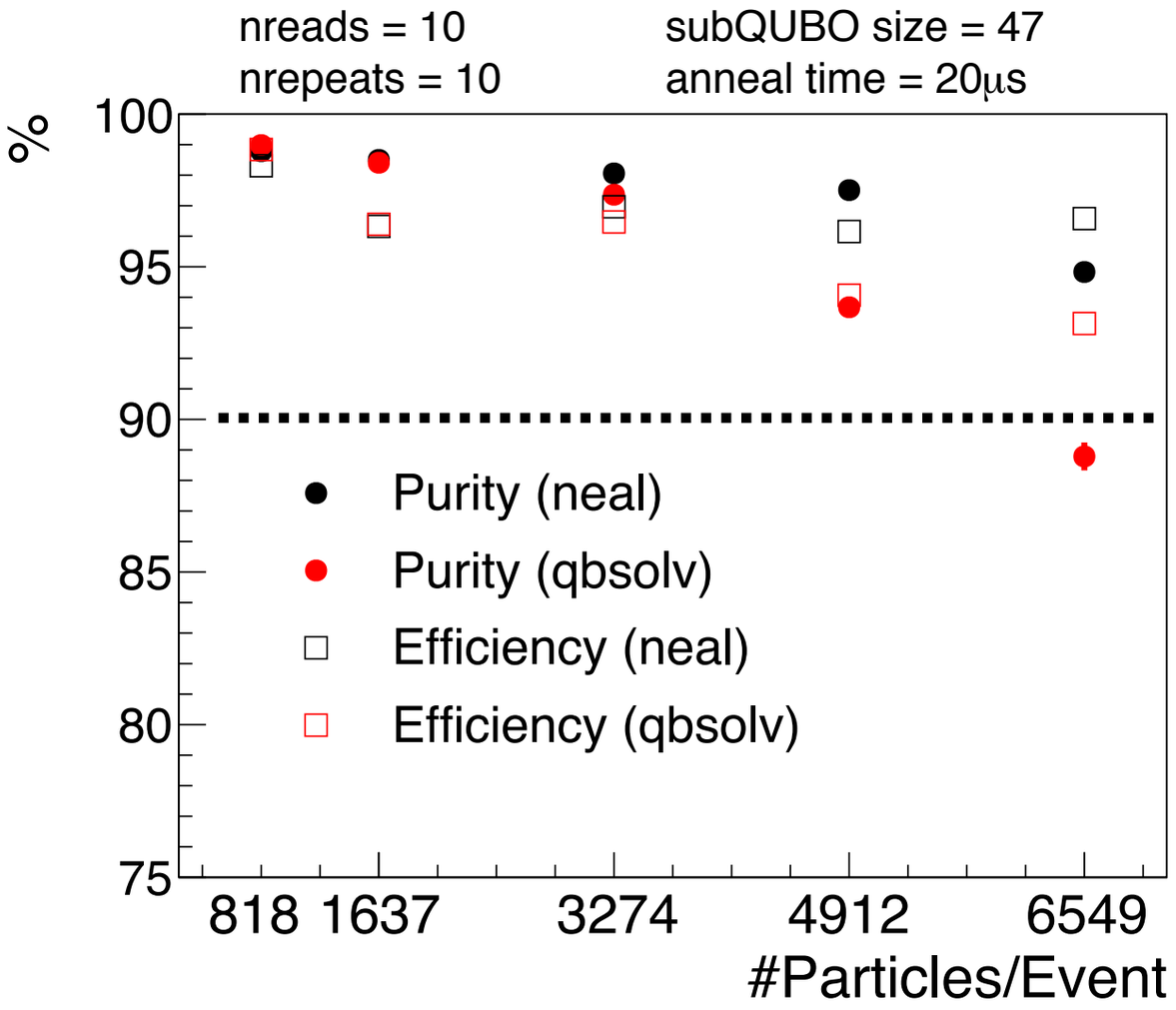
アニーリング

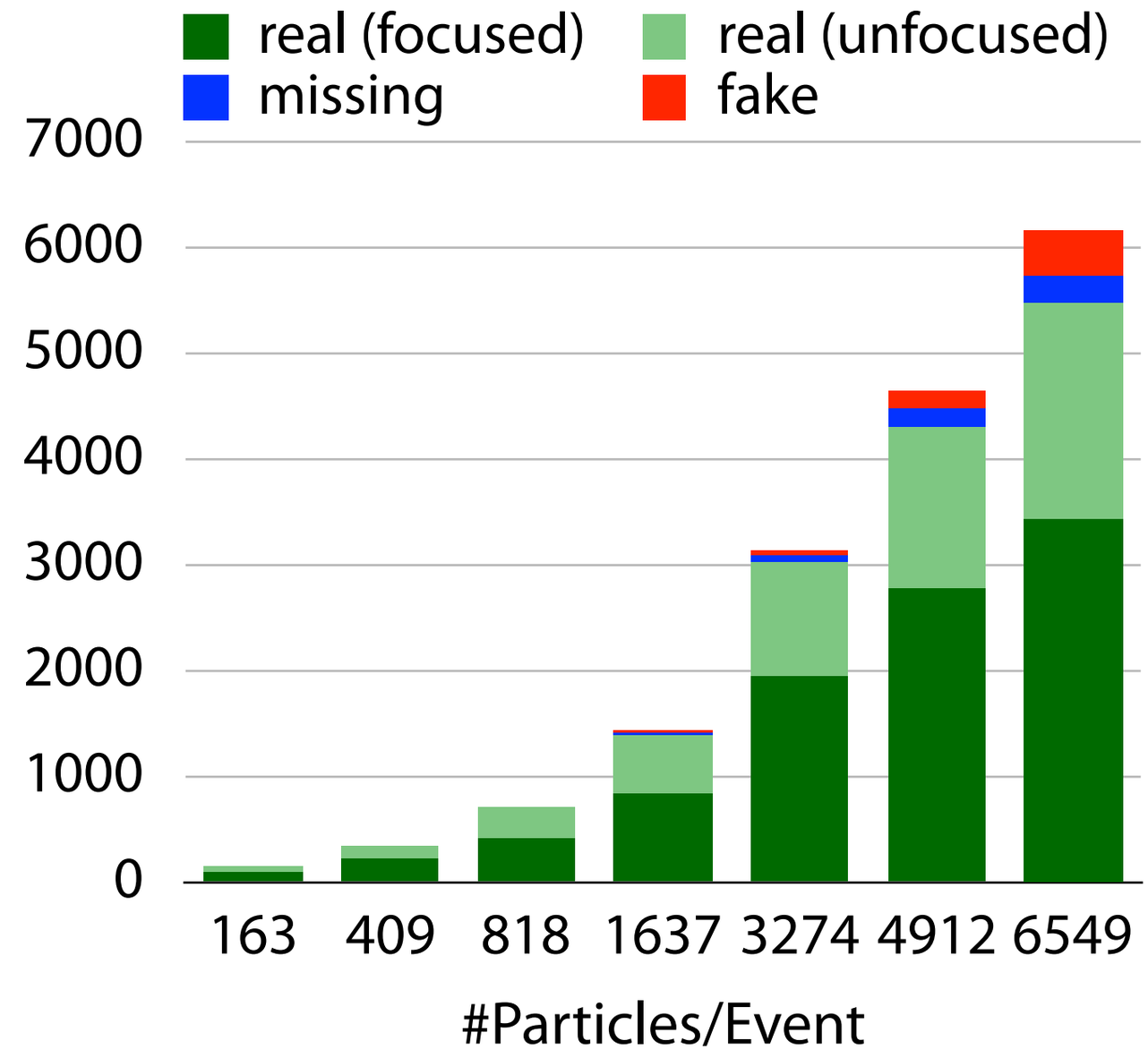
390000 *Doublets*	
Purity	0.22%
Efficiency	99.5%

2445 *Doublets*	
Purity	37.2%
Efficiency	22.6%

1424 *Doublets*	
Purity	98.5%
Efficiency	96.4%

結果





- ~6000粒子までは、90%以上の再構成効率とPurityを達成
- 熱揺らぎでアニーリングさせるSimulated Annealing (neal)は、同等かそれ以上に有効。。。

量子アニーリング：次のステップ

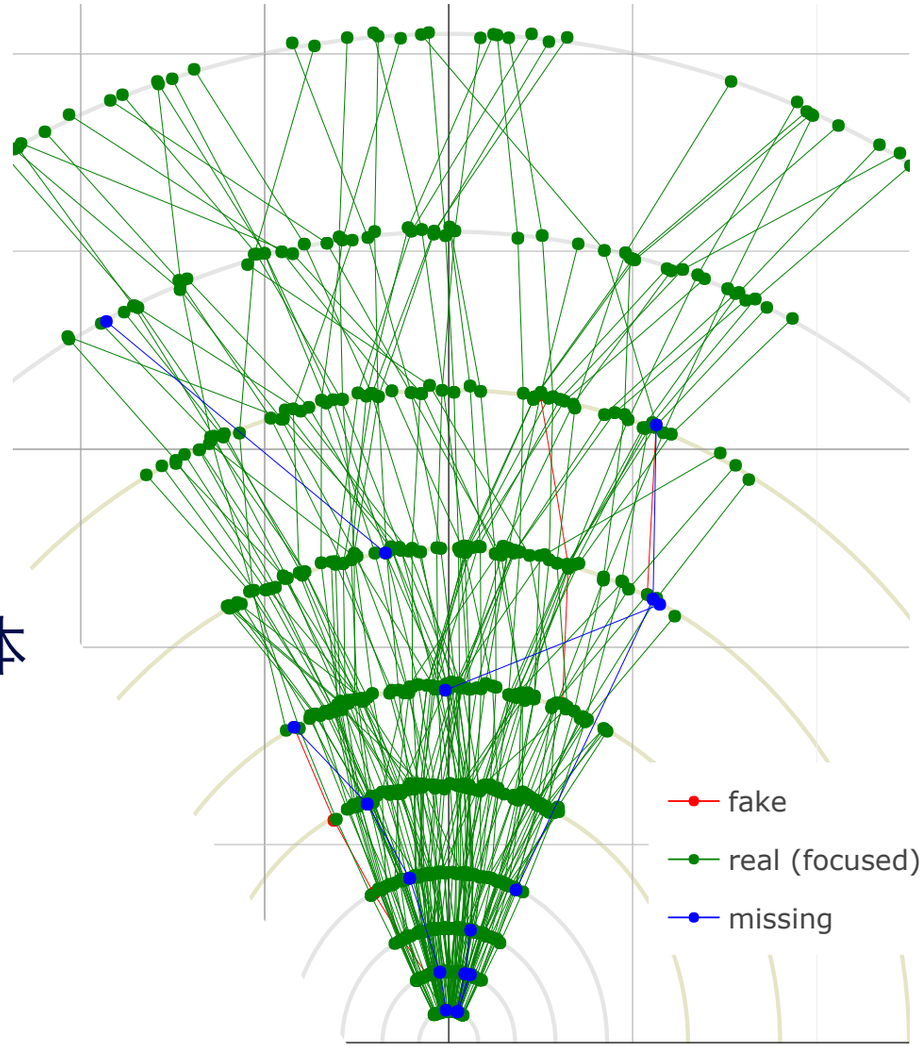
より現実に近い環境でのトラッキング

- 共有ヒットの取り扱い
- Endcap領域
- Low p_T (<1 GeV) への拡張

検出器の狭い範囲に限れば、HL-LHC環境でも問題自体はまだスパースである可能性が高い

➡ 通常のトラッキングが難しい状況への応用

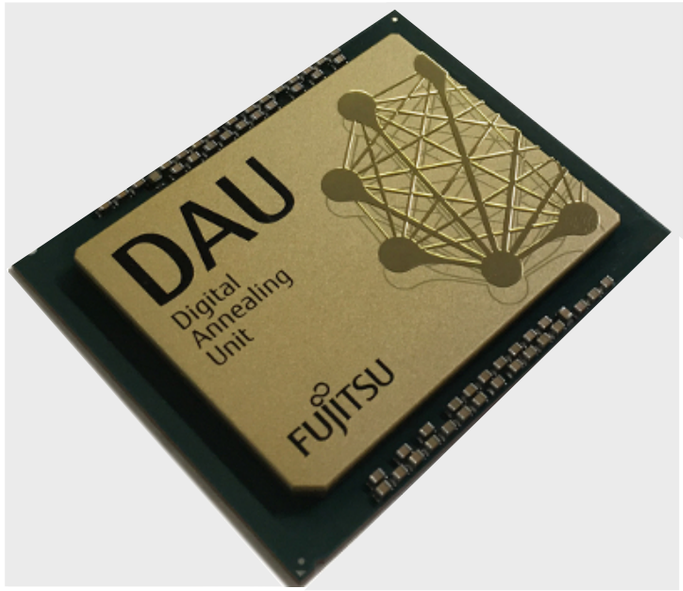
例) ジェット中の飛跡のような高密度環境



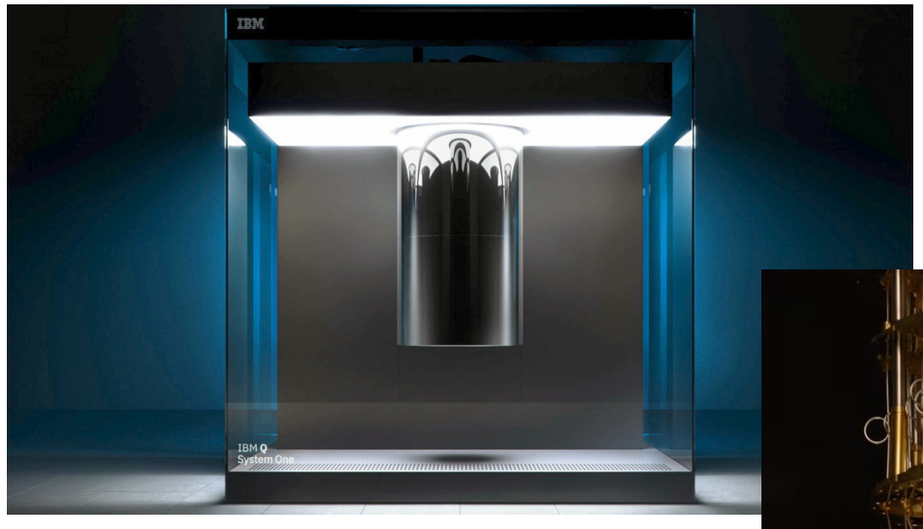
量子ビット数の制約やネットワーク利用のため、大規模化は容易ではない

➡集積回路を使ったアニーリングの応用

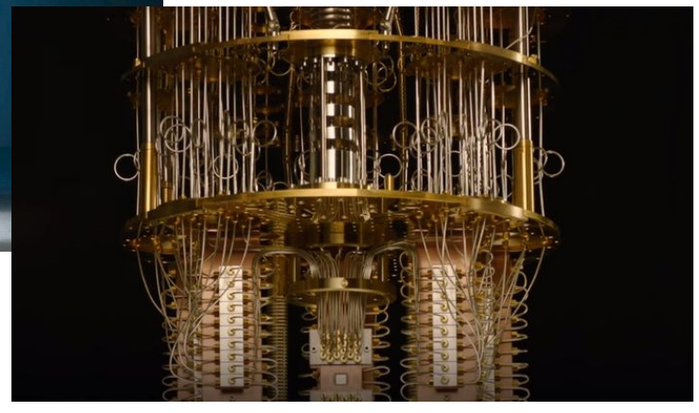
例) 富士通・日立デジタルアニーラー



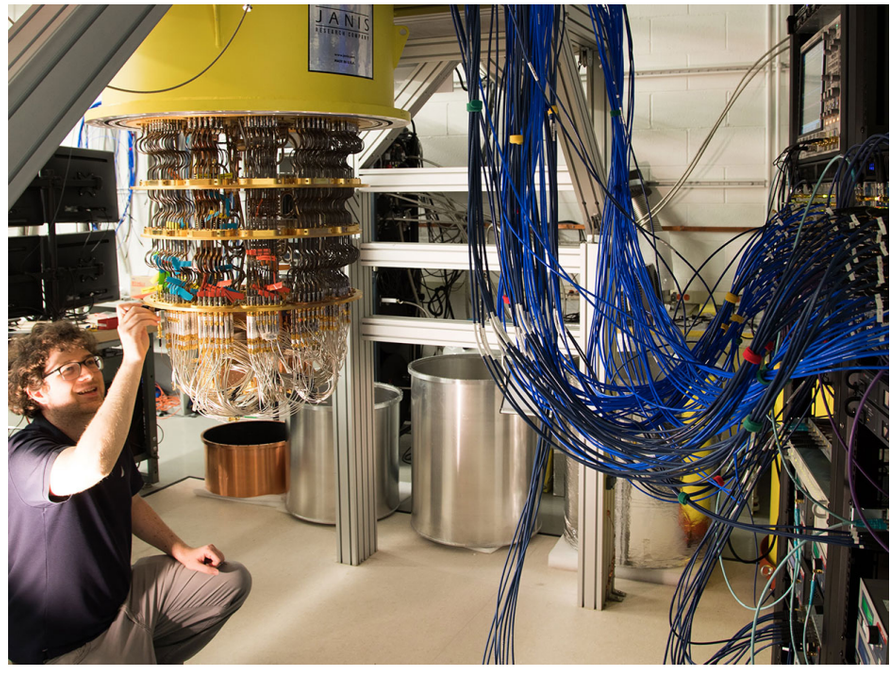
量子ゲートコンピュータでは…



IBM



Google



- 量子ゲートコンピュータで飛跡の再構成を行うのは非現実的かも。。
  - 量子ビット数などのハードウェアの制約、回路構成の複雑さなど
- ただし、アイデア次第でより効率的なハミルトニアンモデルの探索も可能

現在は、高度なアルゴリズムの実装には「古典・量子計算のハイブリッド手法」が主流

- 量子計算が得意なところだけ量子コンピュータが担当、それ以外は通常計算にまかせる

量子ゲート方式での再構成

まずイジング模型のハミルトニアンの場合を考える

QUBOハミルトニアンをイジング模型のハミルトニアンへ変換

$$\mathcal{O}(a;b;T) = \sum_{i}^{N} a_i T_i + \sum_{i}^{N}\sum_{j>i}^{N} b_{ij} T_i T_j \quad T \in \{0,1\}$$

$$T_i = \frac{1 - s_i}{2}$$

$$\Rightarrow \quad H = \sum_{i}^{N} h_i s_i + \sum_{i}^{N}\sum_{j>i}^{N} J_{ij} s_i s_j \quad s_i \in (-1,1)$$

このハミルトニアンの元で、最小エネルギーになる「*Triplet*の組み合わせ＝基底状態」を優先的に選び出す

➡ 量子回路への実装はどうするか？

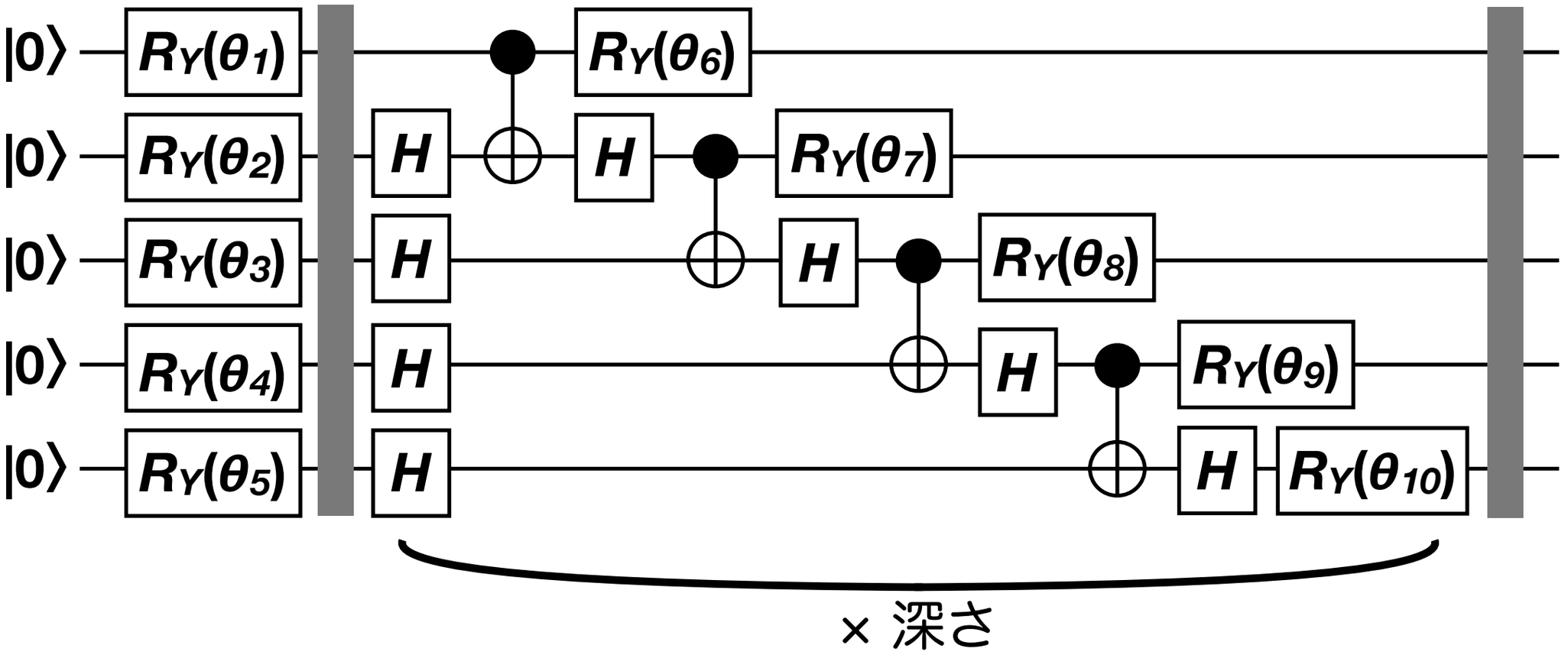
量子回路への実装

どのように「基底状態」を与える回路を実装するか？

- 量子ビットにパラメータ化した量子ゲートを作用させ、その出力状態のエネルギー期待値を測定する（➔ 量子計算が担当）
- 量子パラメータをランダムに更新し、最小エネルギーになる状態 ＝「基底状態の近似解」を探索する（➔ 古典計算が担当）

➡ **変分量子回路**

5量子ビットの場合

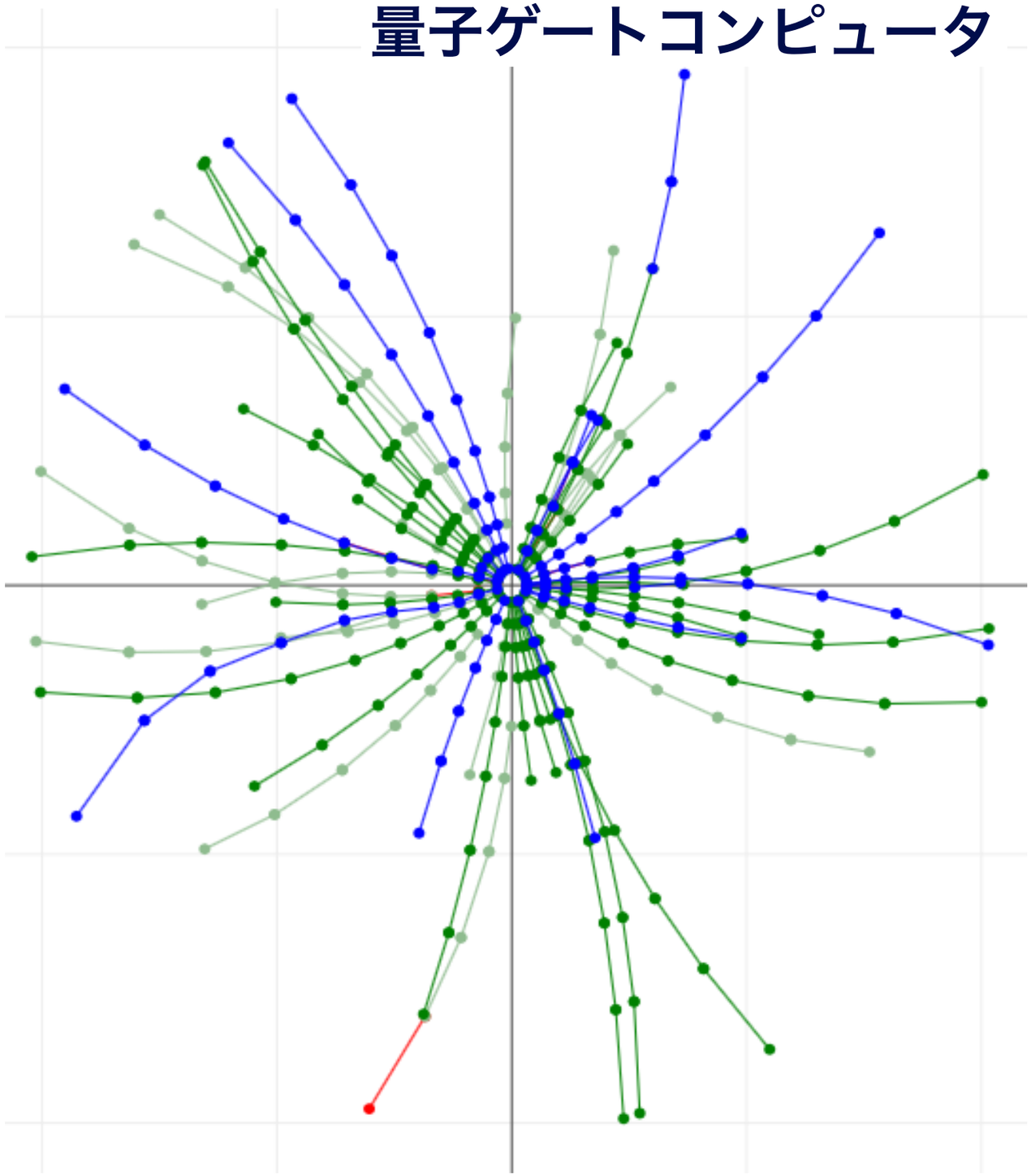


量子回路での再構成

- 量子ビット数を抑えるために、
  検出器を$\Delta\eta$=0.1ごとの領域に分割
  - 各領域で$N_{qubits} < 25$を要求
  - $N_{qubits} > 25$の場合は、ϕ方向にさらに分割
- 各領域ごとにパラメータを決定
  (回路構成は共通)
- 最後に全η領域の結果を統合
- 高速量子回路シミュレータQulacsで実行 (でないと時間がかかってしょうがない。。。)

~400粒子の場合

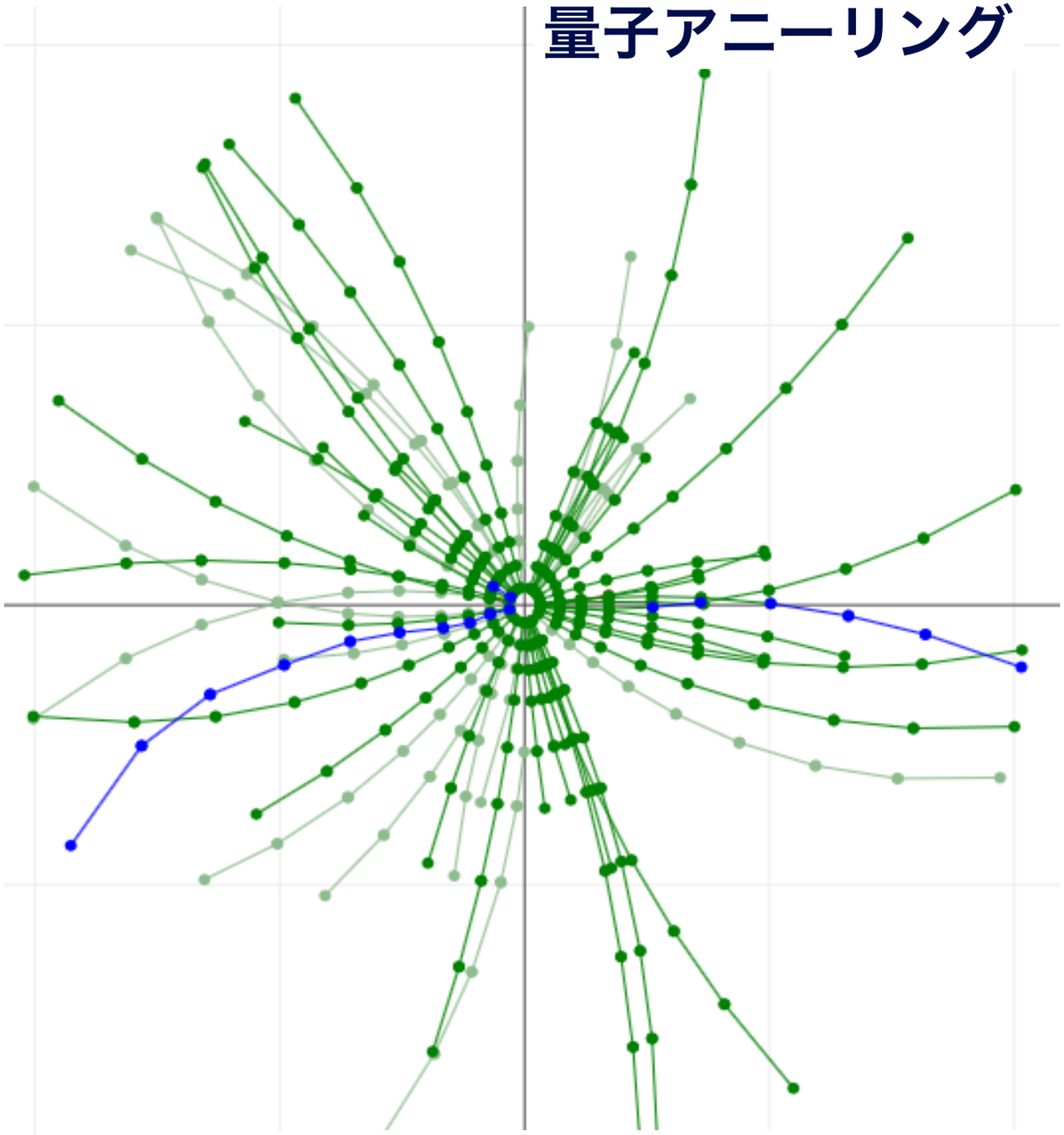
量子ゲートコンピュータ



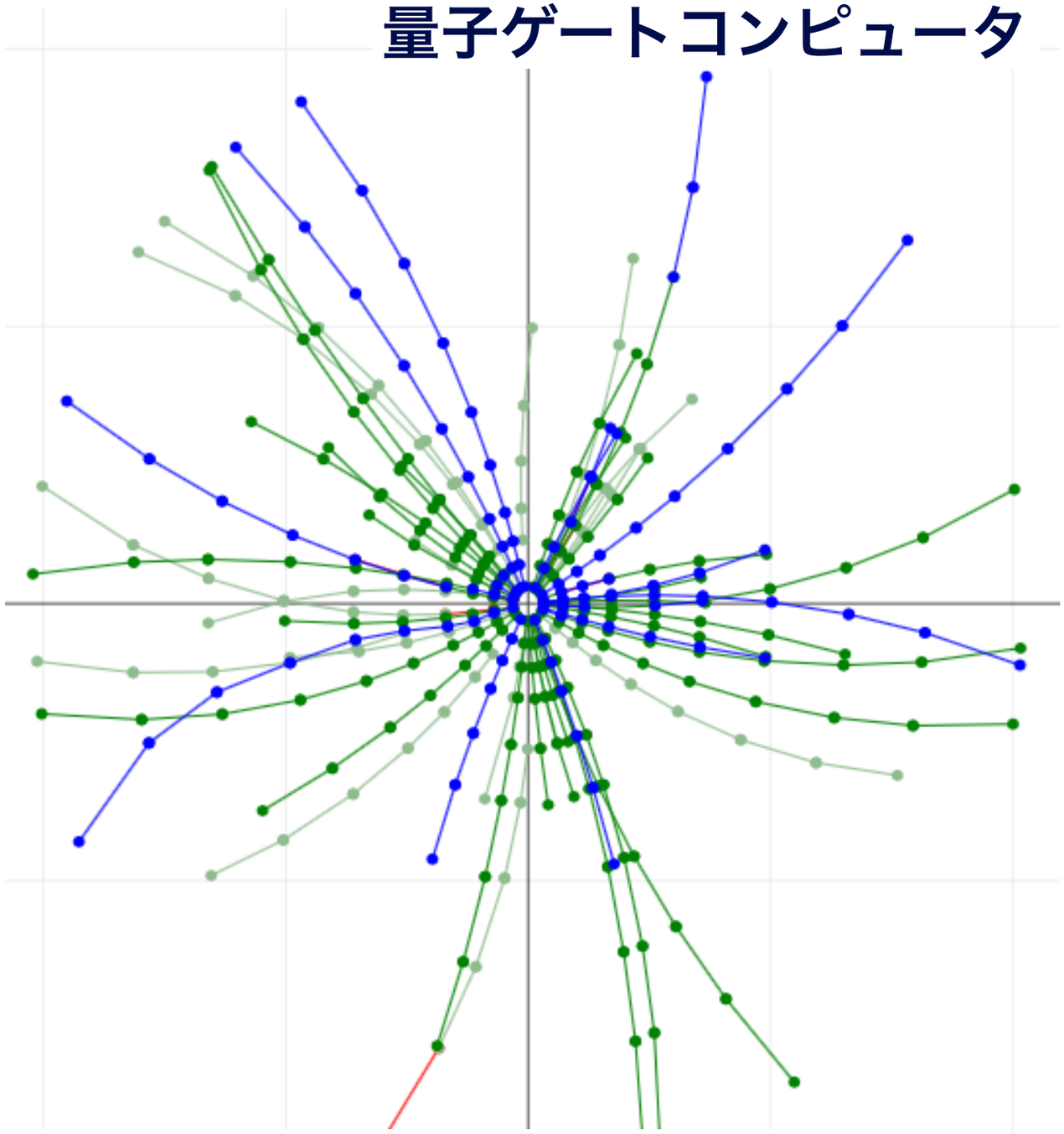
量子回路での再構成

~400粒子の場合

量子アニーリング



量子ゲートコンピュータ



- 量子アニーリングの結果をほぼ再現できる
  (失われた飛跡は領域分割のため？)
- 実際の量子ゲートコンピュータでは高速化できるか？ (今後の課題)

量子ゲートコンピュータ：次のステップ

実際の量子ゲートコンピュータでの検証

ただし標準的なトラッキングとして使うには制約が多い。。

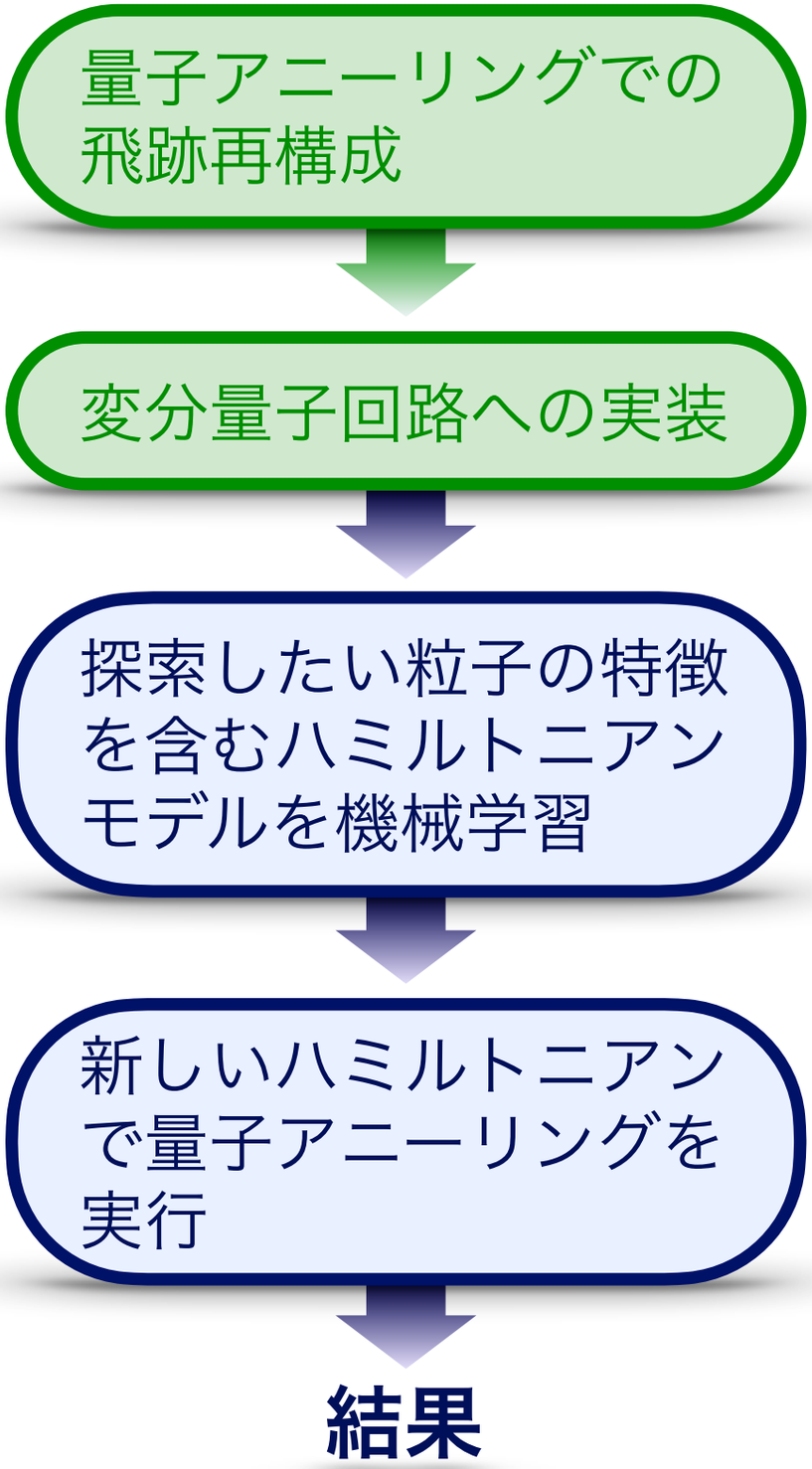
➡ 特殊なトラッキングに活用する？

- 例えば、検出器に特徴的なヒットを残す粒子の探索
- ヒットの特徴まで含めたハミルトニアンモデルを、量子回路を使って学習させる

まだ雑談レベルの話ですが。。

機械学習への応用

— 次の大きなチャレンジ



量子機械学習

- 新現象の探索には、信号と背景事象の特徴を学習し、分類に最適な変数と閾値を決定する必要がある

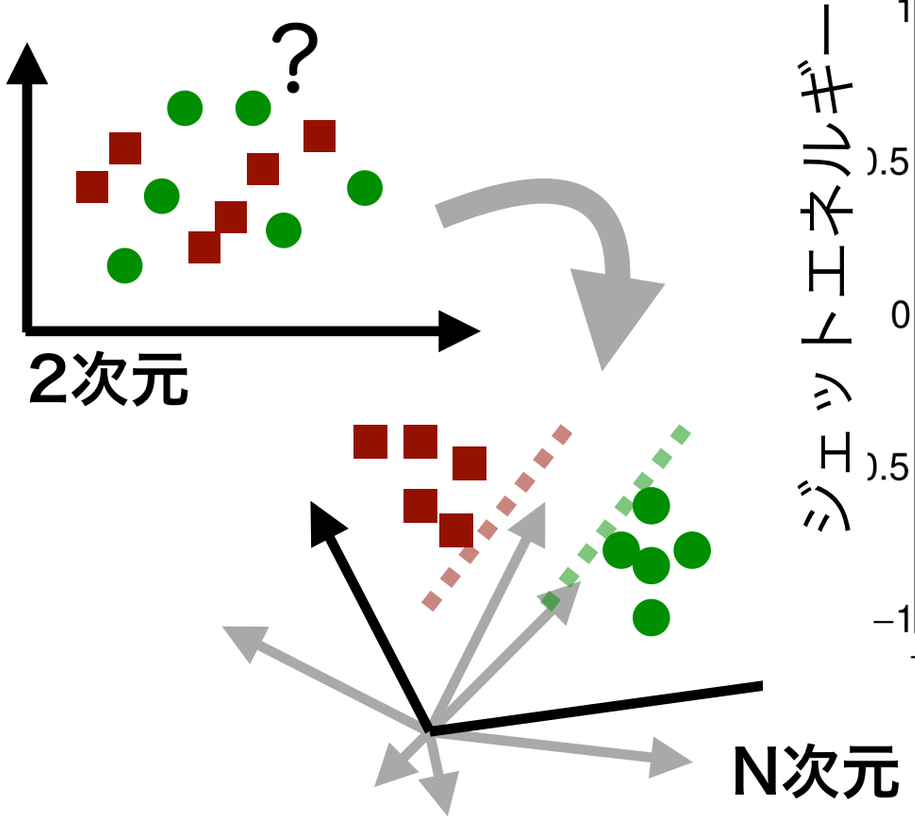
➡ **膨大な量子空間を活用し、効率の良い分類を目指す** (→ 量子機械学習)

- 50量子ビットの場合、約10^{14}次元空間が使える!!

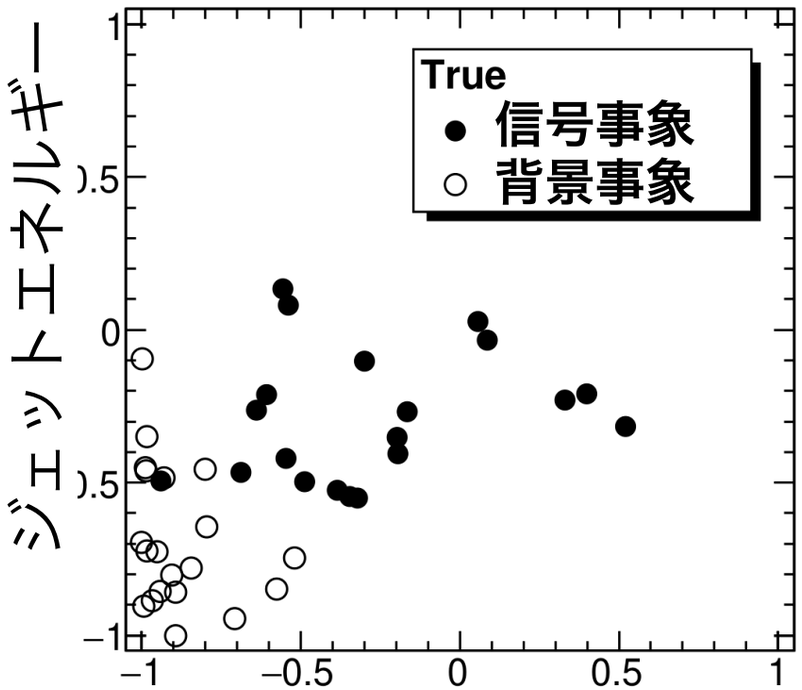
量子サポートベクターマシン

- 量子空間で信号と背景事象を　**信号(SUSY)の分類**

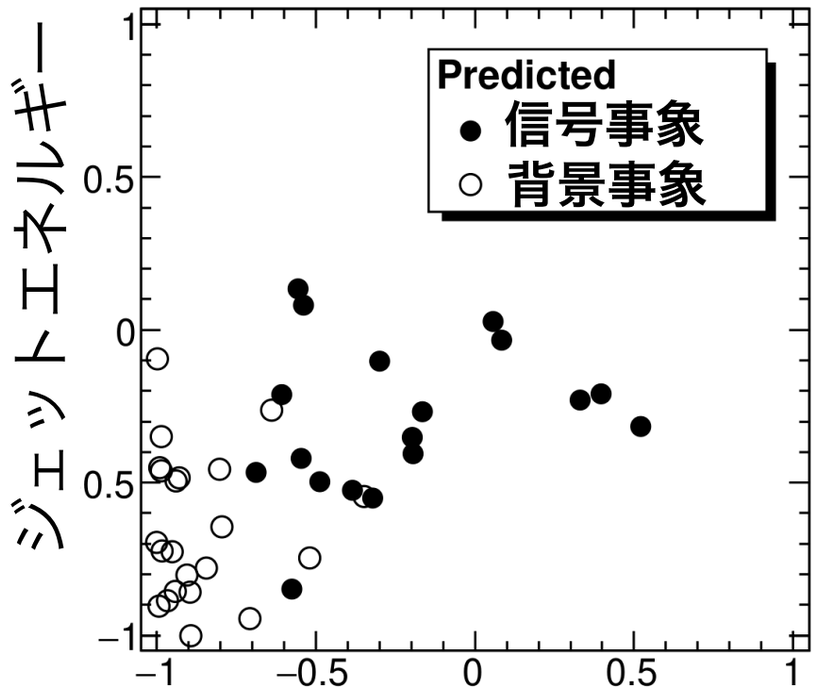
分類する平面を学習する



真の分類(教師データ)



量子SVMの分類



まとめ

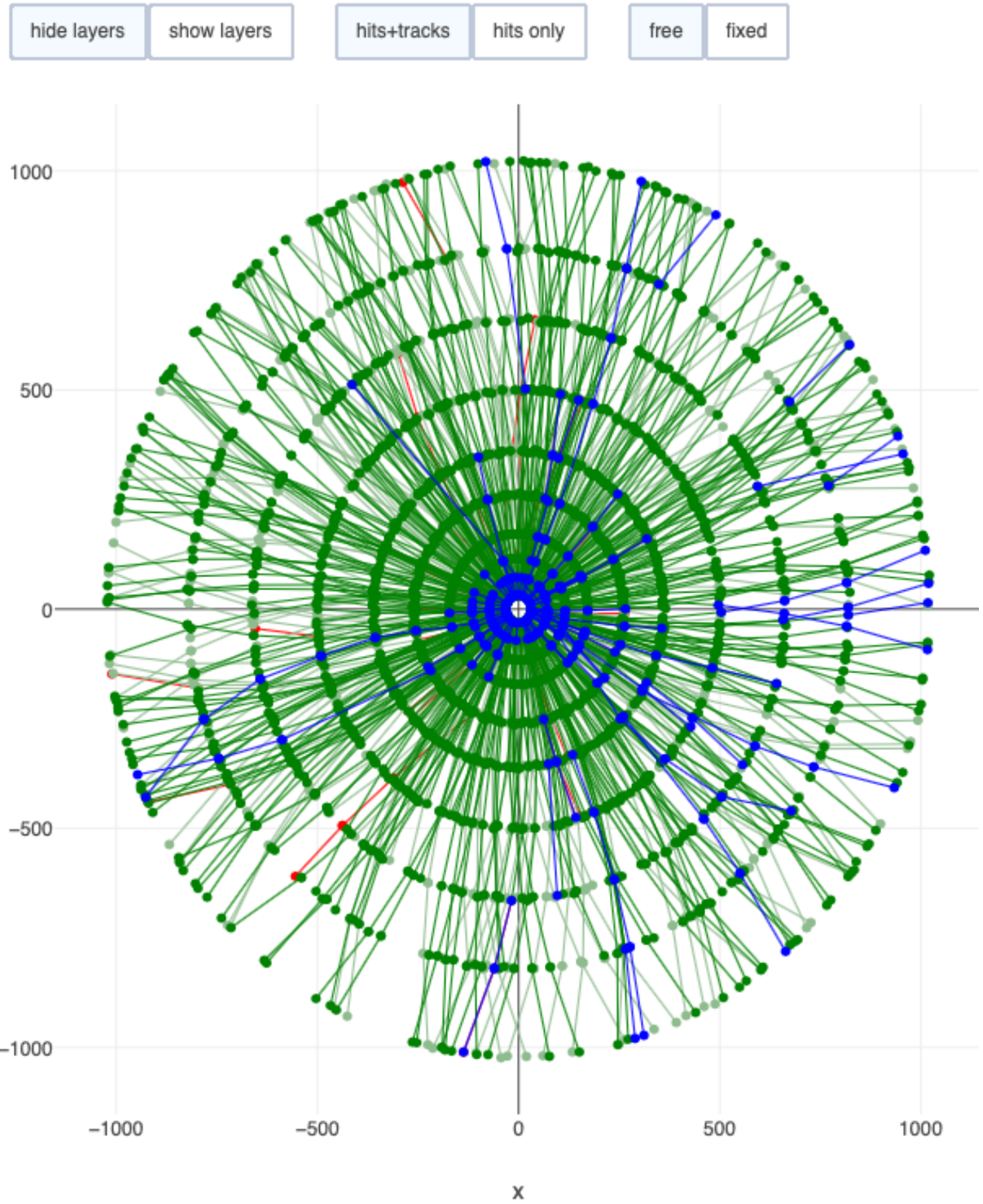
- 飛跡の再構成に向けた量子コンピューティング技術の応用に取り組んでいる。
- 量子アニーリングの初期成果はまずまず有望に見える。
  - 今後はより実機に近い環境への拡張・大規模化への取り組みへ
- 量子ゲートコンピュータでも可能だが、現実的には？
  - より特殊なトラッキングへ？
- 量子機械学習への取り組みも開始したところ。。

さらに幅広い応用を考えて、積極的に展開して行きたい。
興味がある方は是非一緒にやりましょう!!

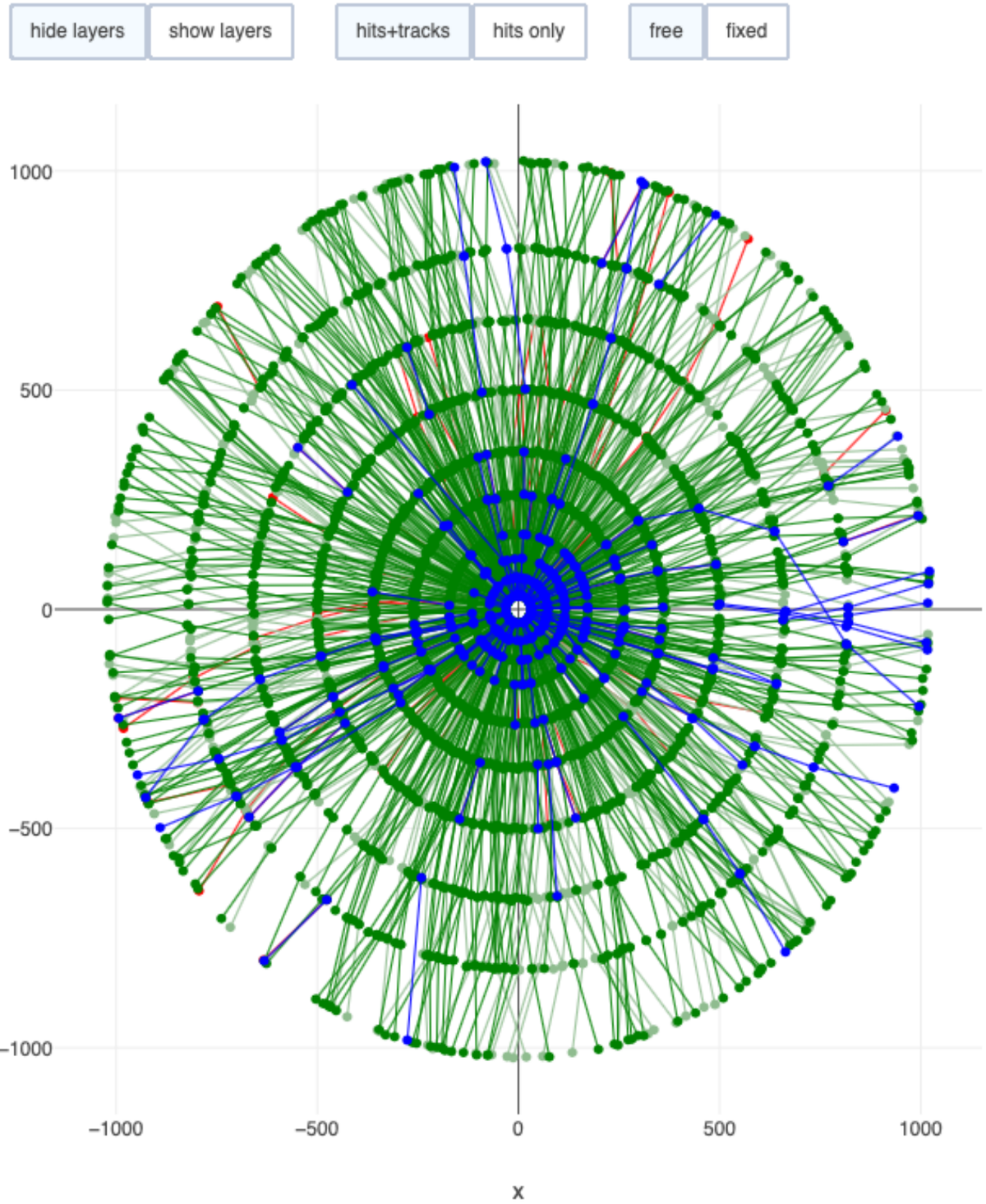
LBNL-ICEPP QCミニワークショップを2019年2月に開催しました。
解析ソフトウェアの詳細などはリンクを参照してください。

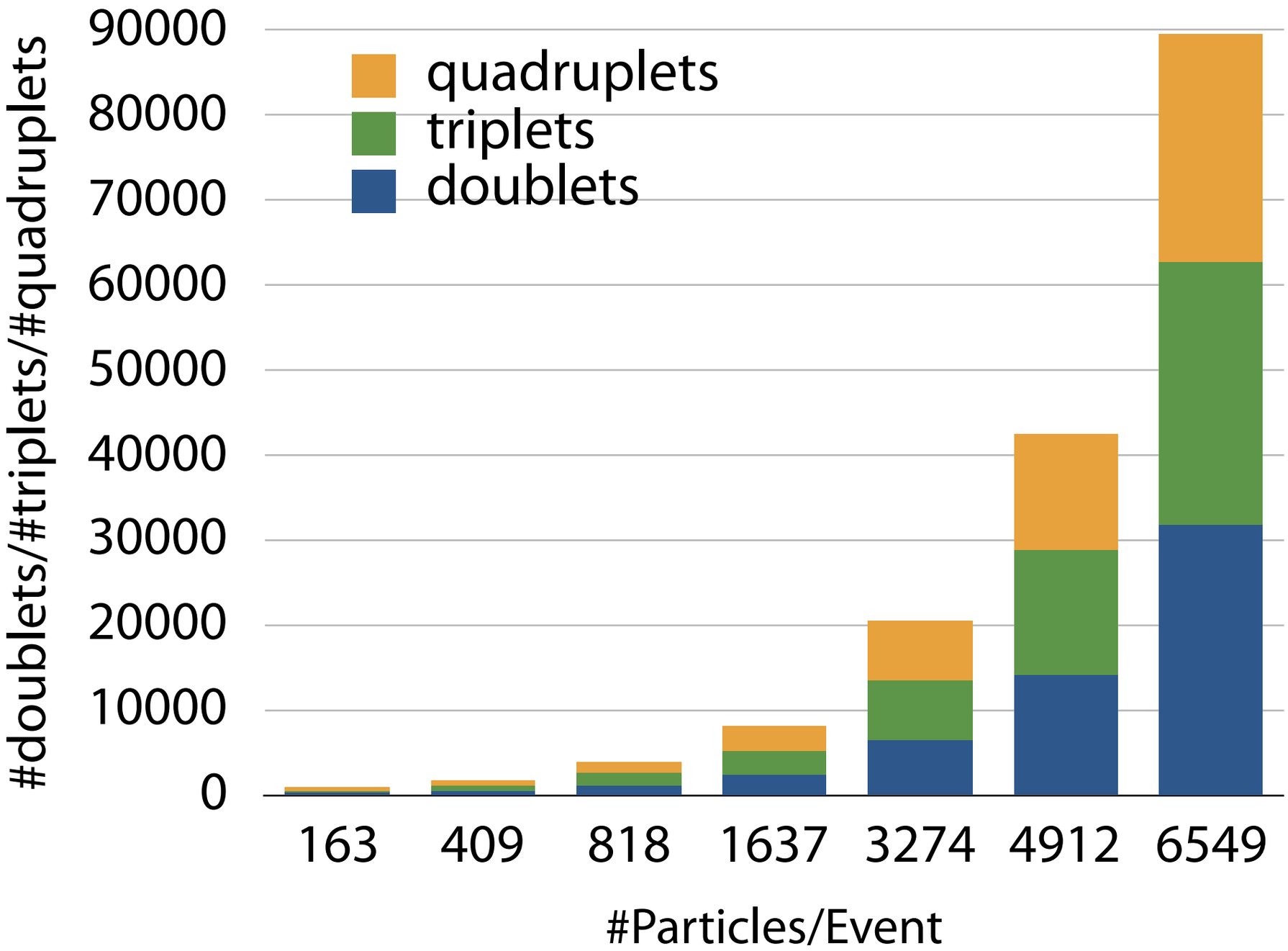
バックアップ

~4900粒子の場合



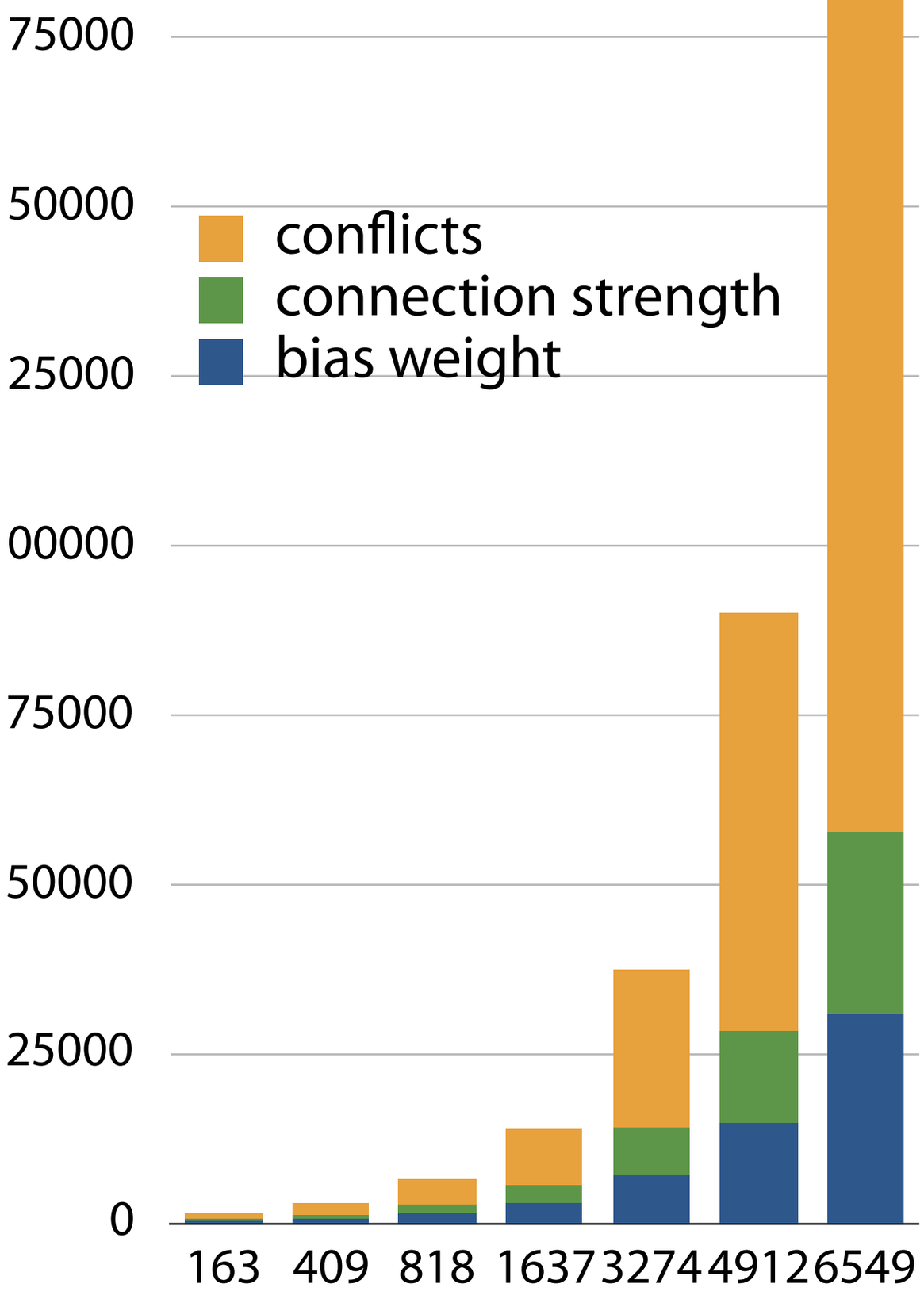
~6500粒子の場合



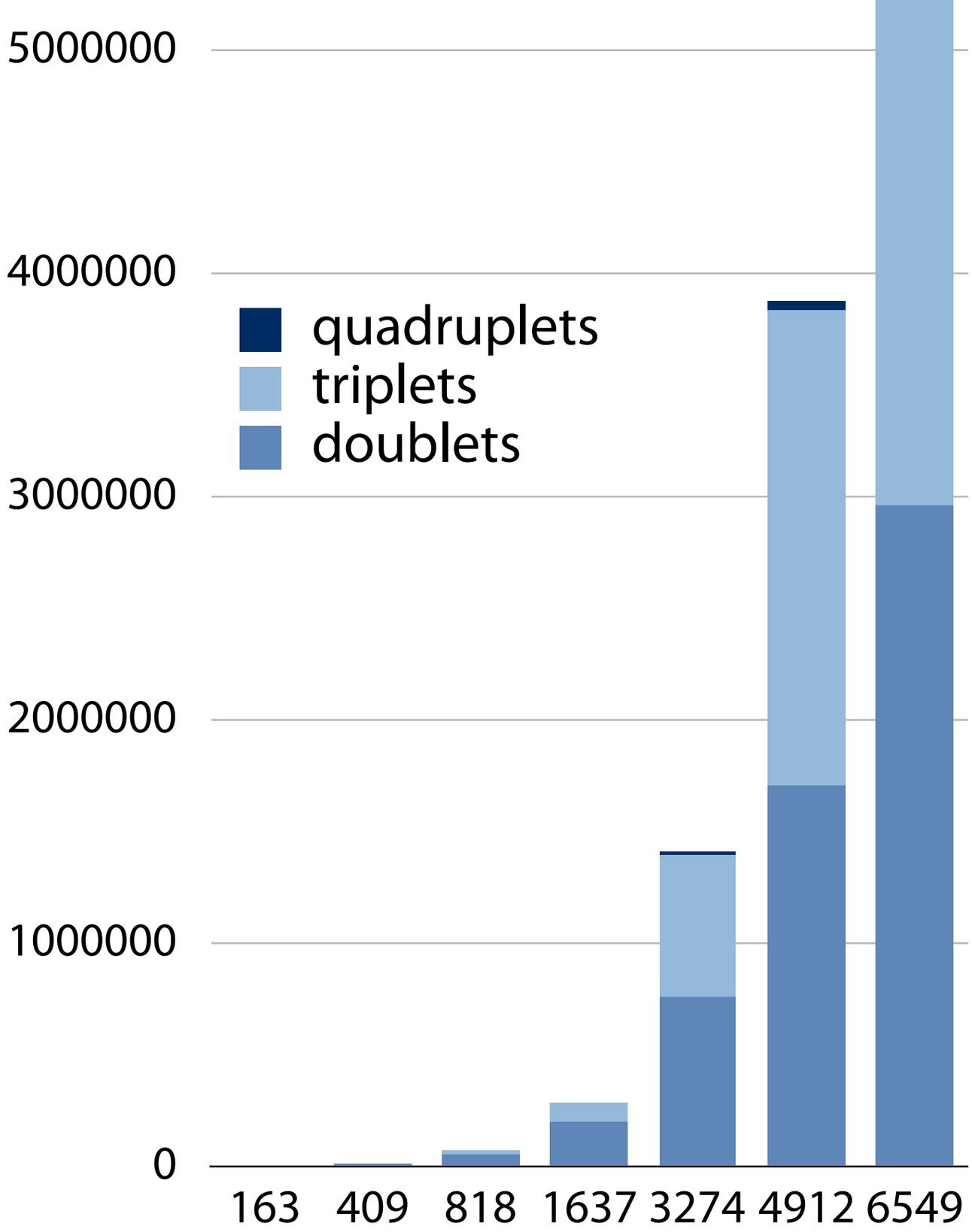


QUBOインプットでのdoublet, triplet, quadruplet

なぜdoublet : triplet : quadruplet = 3 : 2 : 1にならないか？



QUBOパラメータの内訳



Pass1で選別されたdoublet, triplet, quadruplet

Track ML

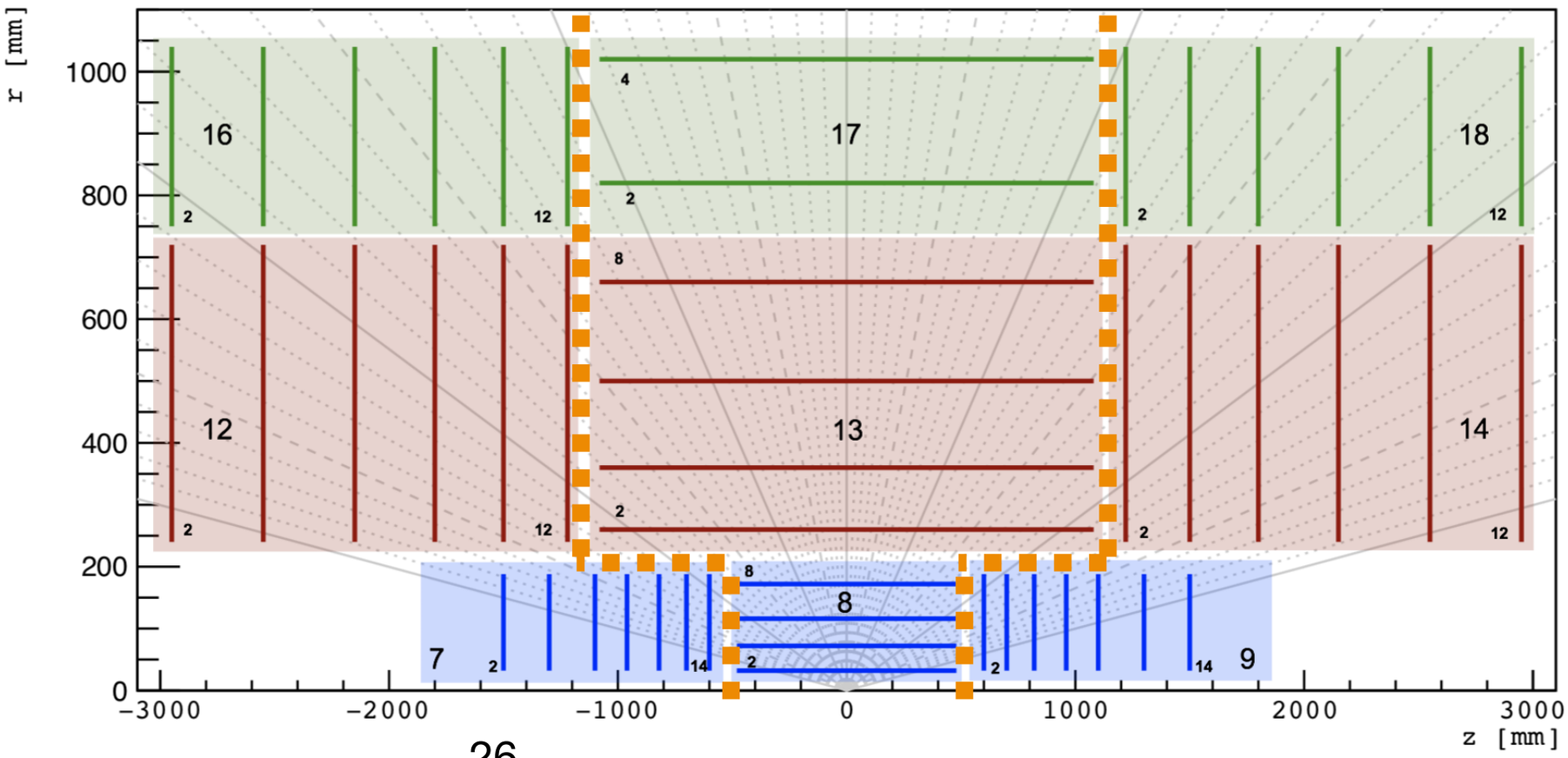
TrackML : HL-LHCでのTracking改良に向けたチャレンジ (2018年夏)

HL-LHCと実機に近い環境を提供

- ~10^4生成粒子、~1.1×10^5ヒット
- 同じ粒子が一つのレイヤーに複数のヒットを作る（約10%）
- 磁場の不均一性
- 検出器との多重クーロン散乱
- セル単位でのヒット位置の読み出し（有限の位置精度）
- ノイズの付加（約15%）

ただし、ヒットは最大で一つの粒子にしか付随しない

この解析では、バレル領域の検出器のみ使用



Understanding the qbsolv performance

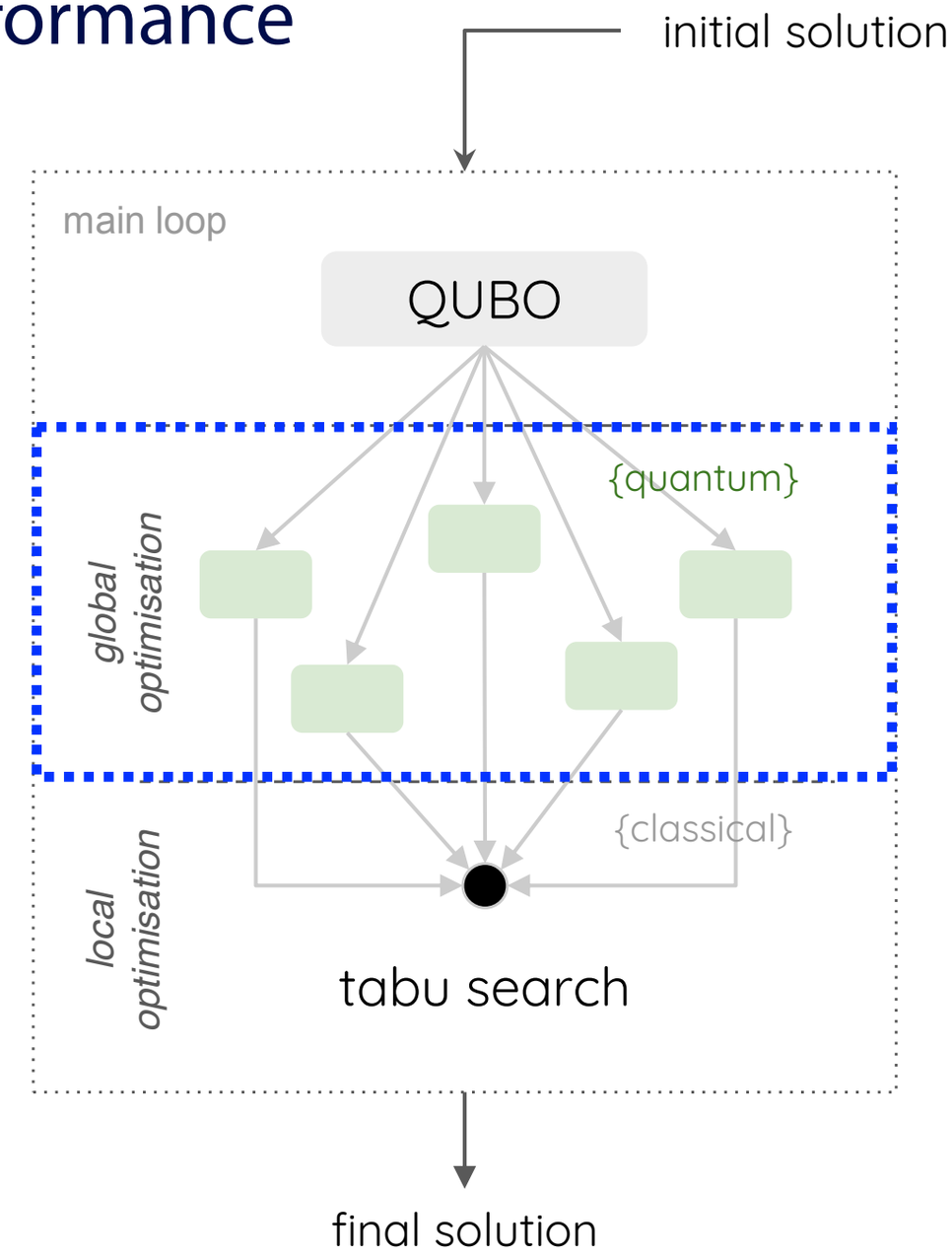
We first tried to run qbsolv with different subQUBO solvers to understand impact on the performance

Used subQUBO samplers:

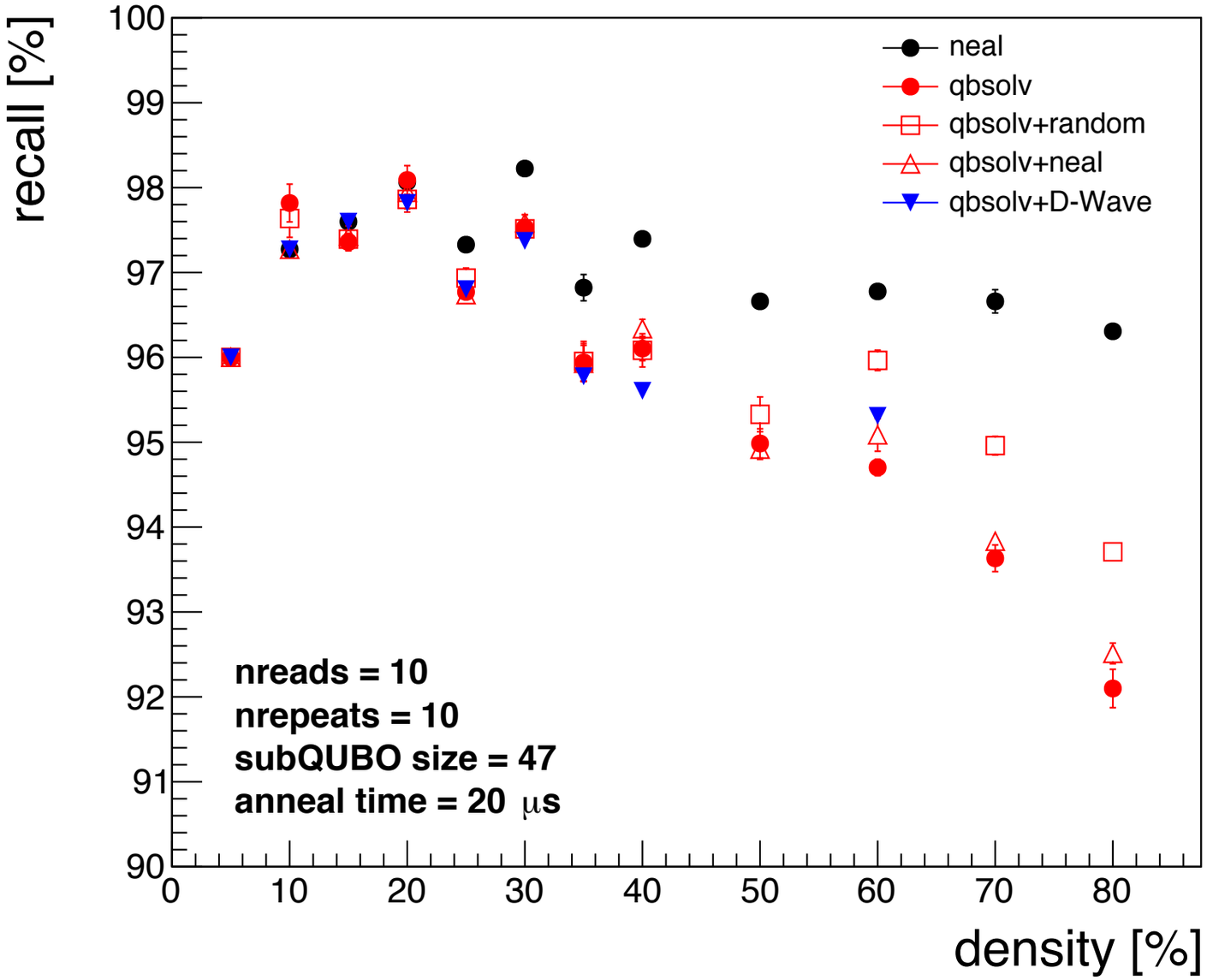
- default (tabu solver)
- dimod RandomSampler
- neal SimulatedAnnealingSampler
- D-Wave

Compared with “neal SimulatedAnnealingSampler w/o qbsolv” as a reference

tabu search in unchanged here



sub-QUBO size = 47



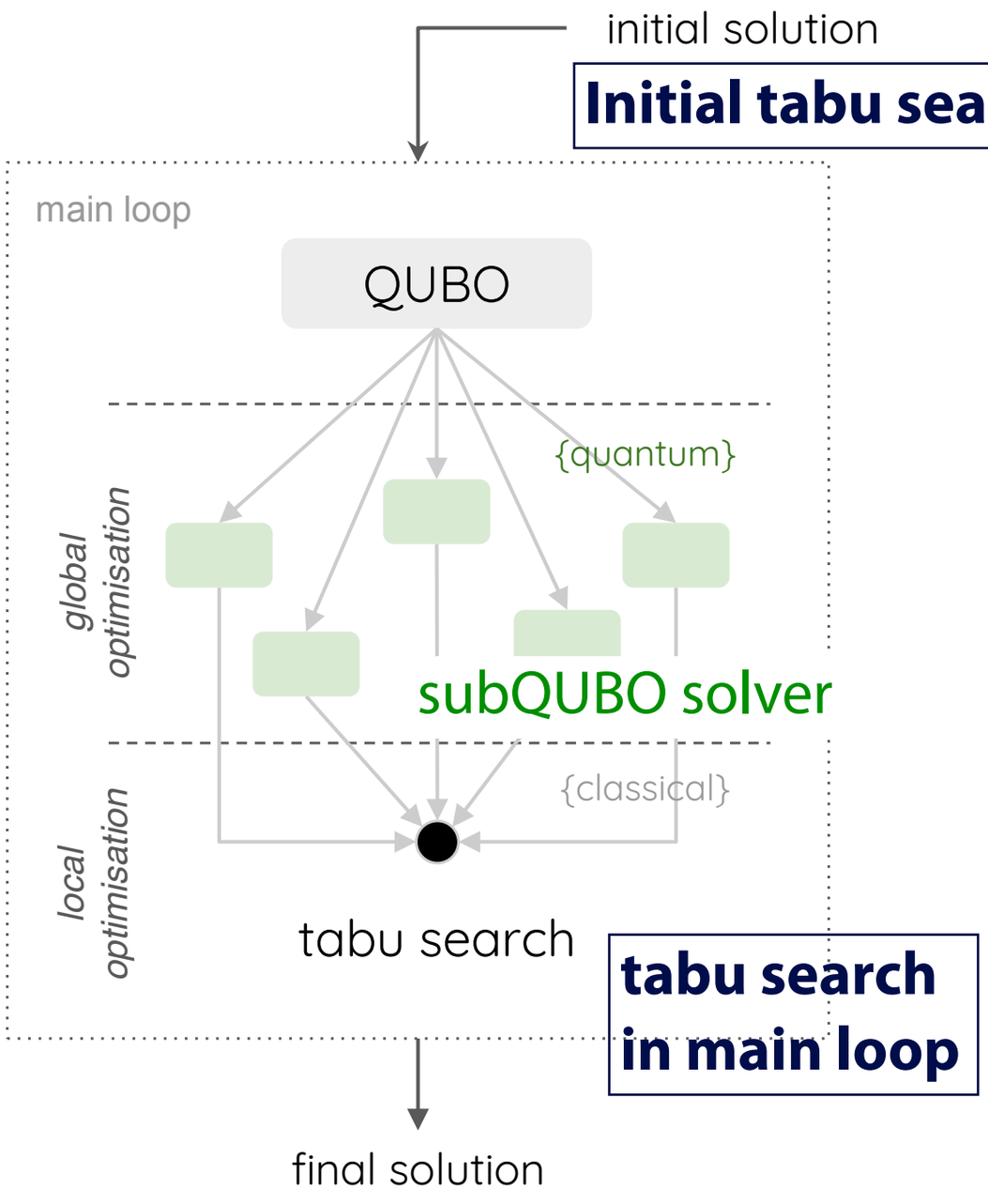
- neal SimulatedAnnealingSampler w/o qbsolv
- default (tab solver)
- dimod RandomSampler
- neal SimulatedAnnealingSampler
- D-Wave

- "recall" on the vertical axis is one of the performance measure used in tracking (higher recall → better performance)
- "density" on the horizontal axis corresponds to how many particles present in each physics event (higher density → larger # of particles)

qbsolv+random sampling looks very similar to other samplers!

Not expected behavior for us …

➔ Started more qbsolv studies (next slides)



Modified qbsolv code to understand the behavior:

1) exactly 1 loop: both initial tabu and tabu search in main loop performed
2) exactly 1 loop: only initial tabu search performed; no tabu search in main loop
3) exactly 1 loop: neither initial tabu nor tabu search in main loop

Still used 4 subQUBO solvers
- a) default (tabu solver)
- b) dimod RandomSamper
- c) neal SimulatedAnnealingSampler
- d) D-Wave

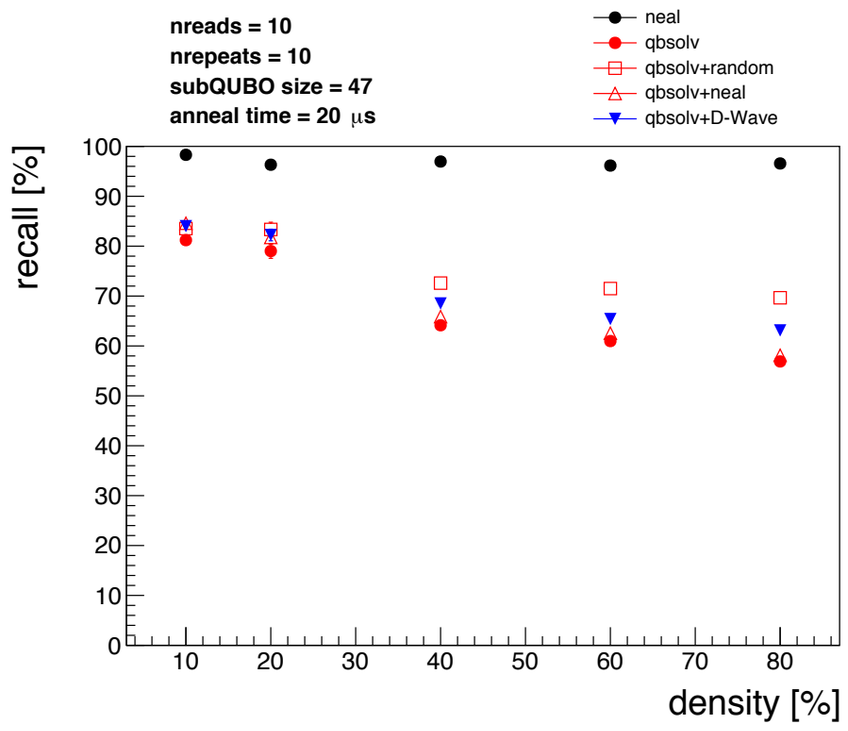
Here "exactly 1 loop" means that the algorithm is processed only once (no repetition is performed in the main loop)

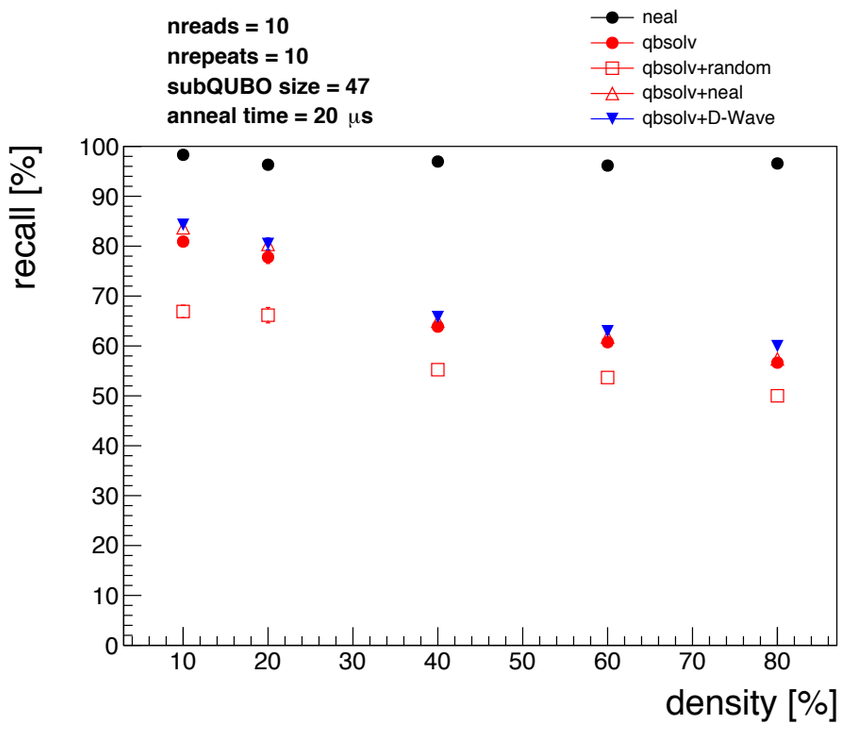
subQUBO size = 47

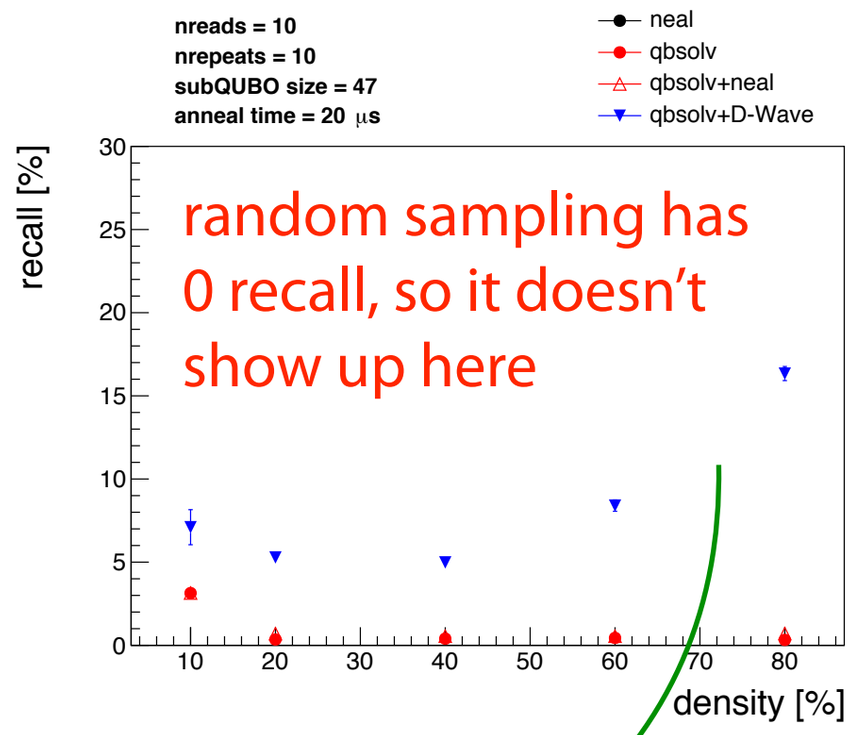
1) exactly 1 loop:
both initial tabu and tabu search in main loop performed

2) exactly 1 loop:
only initial tabu search performed; no tabu search in main loop

3) exactly 1 loop:
neither initial tabu nor tabu search in main loop

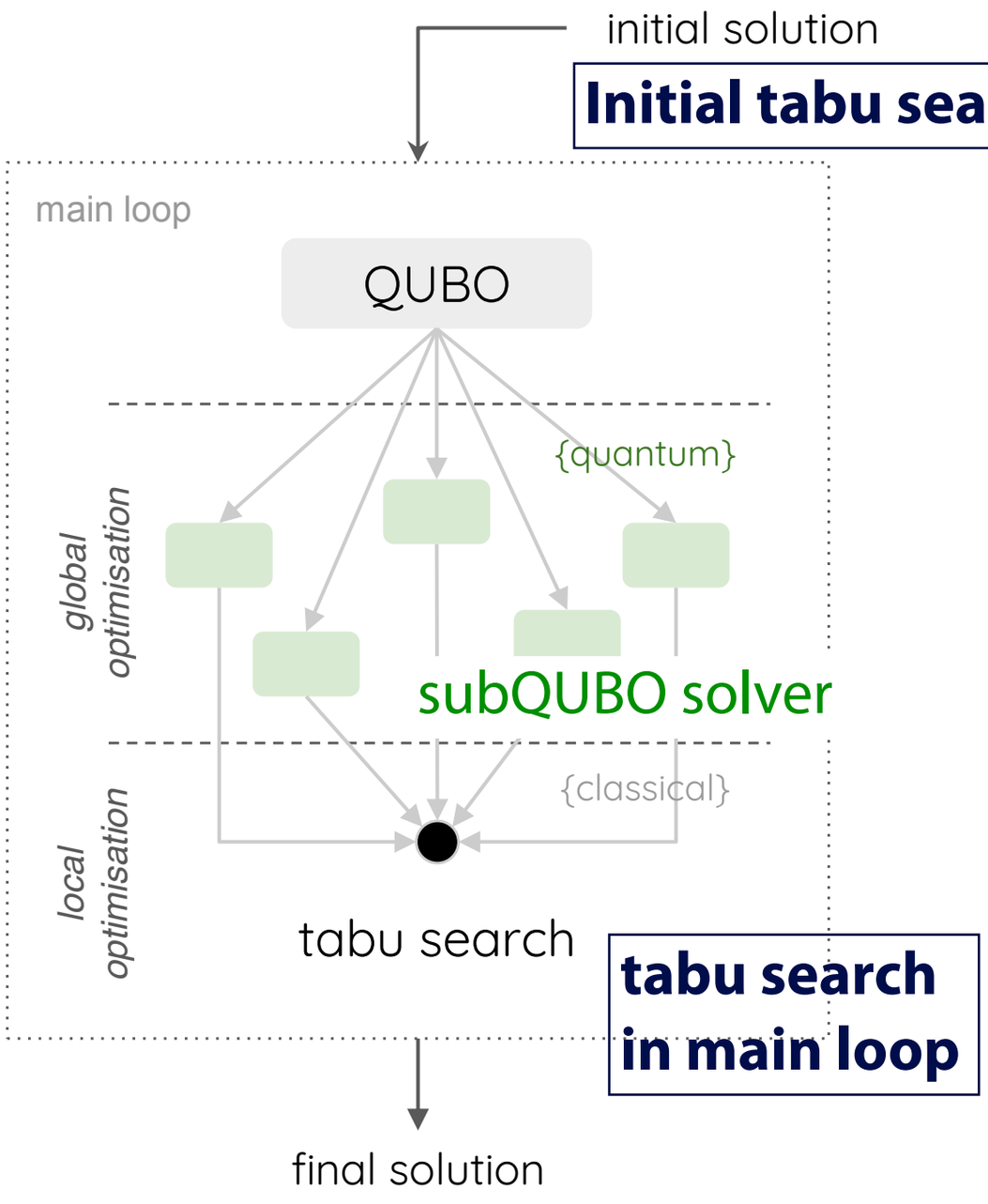






random sampling has 0 recall, so it doesn't show up here

D-Wave appears to work better if no tabu search is used



Modified qbsolv code to understand the behavior

1) exactly 1 loop: both initial tabu and tabu search in main loop performed
   Performance of random sampling similar to other samplers (→ unexpected)
2) exactly 1 loop: only initial tabu search performed; no tabu search in main loop
   Performance of random sampling clearly gets worse (→ expected)
3) exactly 1 loop: neither initial tabu nor tabu search in main loop
   Performance of random sampling is the worst (→ expected)

Confirmed that the qbsolv performance determined by tabu search

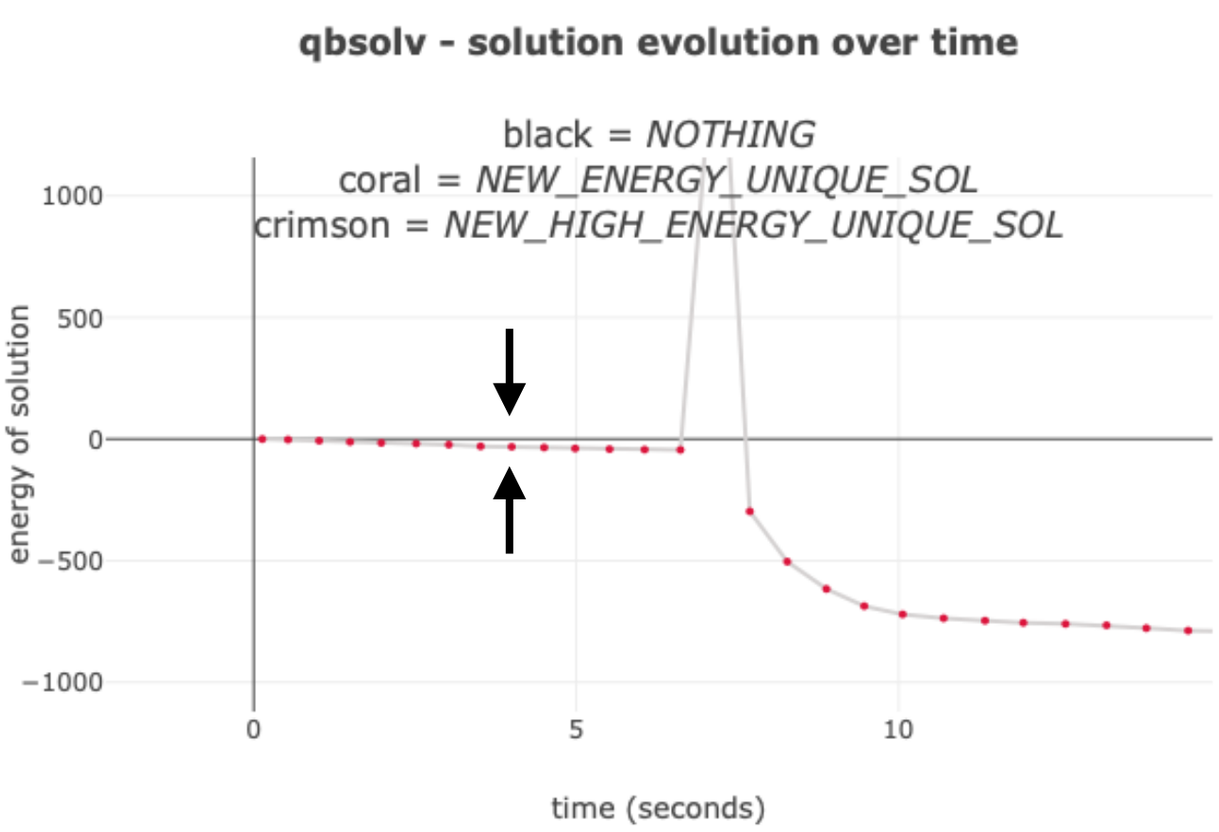
Given that D-Wave can perform better than other samplers w/o tabu search (right plot in p.7),

default qbsolv code is modified to be

4) #repeats=10; neither initial tabu nor tabu search in main loop

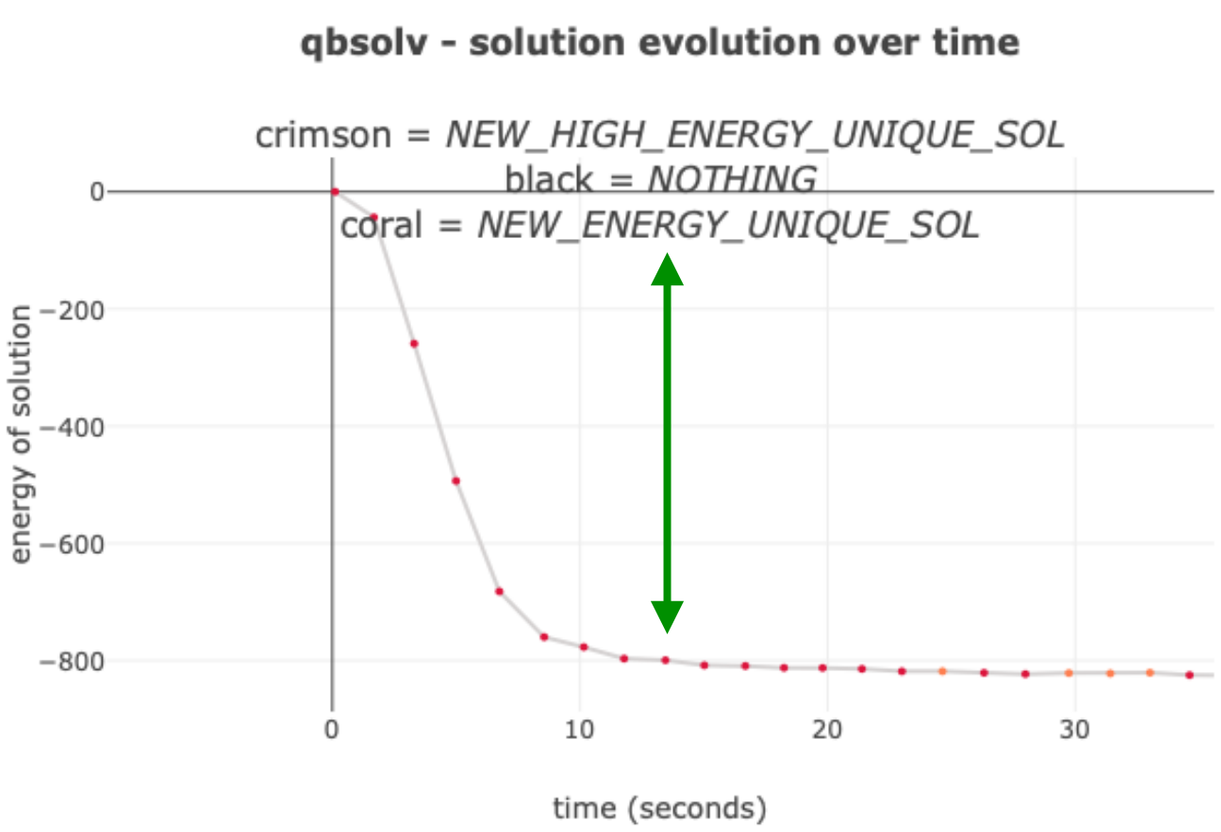
then runs with the default and D-Wave sub-QUBO samplers

default (tabu solver as subQUBO solver)



D-Wave as subQUBO solver

40% density
subQUBO size = 47



Energy drops faster for D-Wave with # of main loops

➔ D-Wave can reach better approximate solution with less # of main loops?

Given that D-Wave can reach lower energy state than other samplers (right plot in p.10),

default qbsolv code is further modified to be

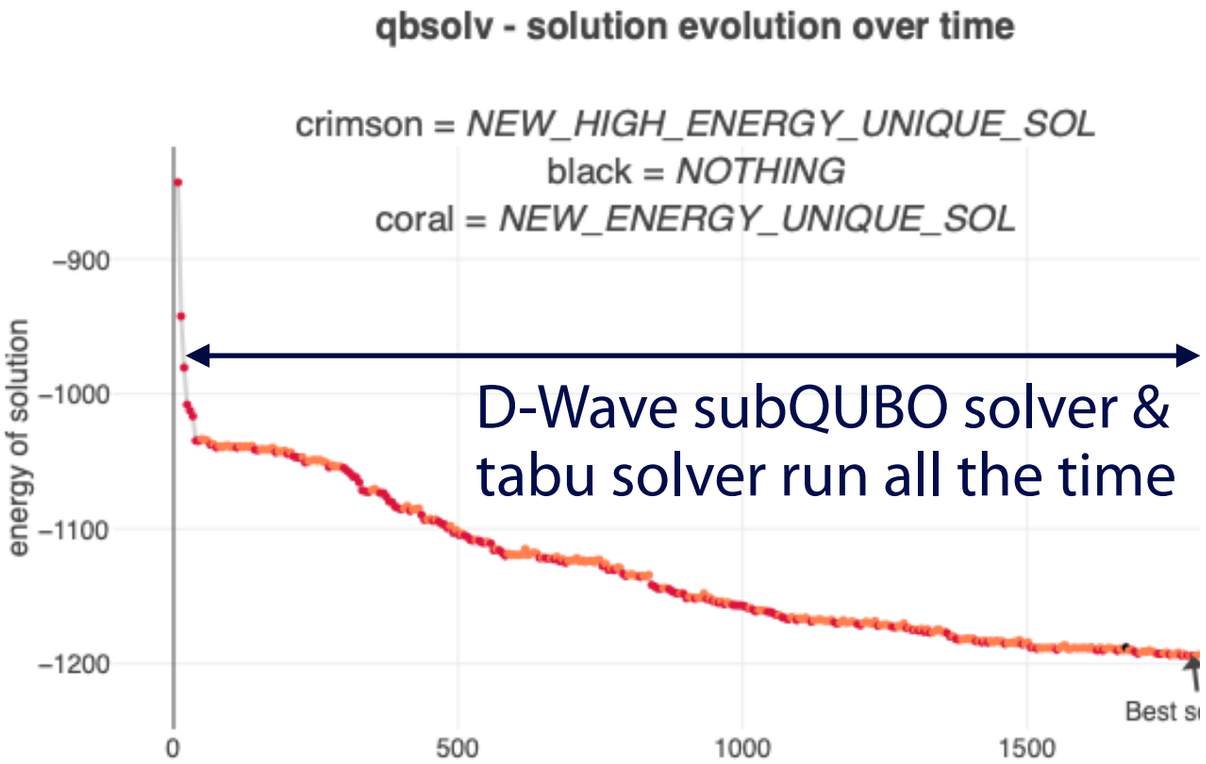
5) #repeats=10; no initial tabu; run subQUBO solver for certain # of loops w/o tabu search in main loop, then stop subQUBO solver; only tabu search runs in main loop after stopping subQUBO solver

Aim is to get intermediate energy quickly with D-Wave (→ suitable for global optimization), then reach the lowest energy state only using tabu solver (→ suitable for local minimization)

subQUBO solver runs only for the specified # of loops (e.g, 8 in next slides), then forced to stop after that in the main loop. No tabu search is performed in the main loop while the subQUBO solver runs. After stopping the subQUBO solver, only tabu solver runs in the main loop until the lowest energy state is obtained with #repeats = 10.

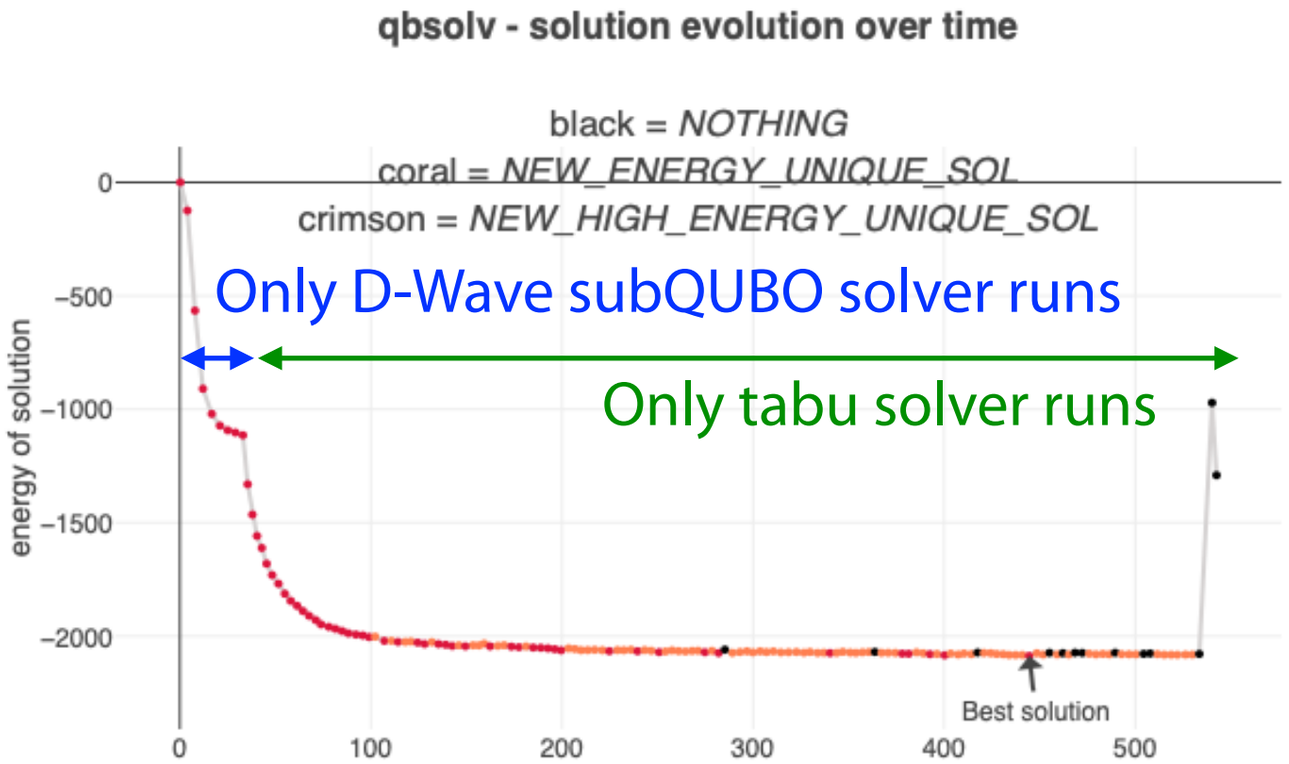
5) #repeats=10; no initial tabu; run subQUBO solver for certain # of loops w/o tabu search in main loop, then stop subQUBO solver; only tabu search runs in main loop after stopping subQUBO solver

Default qbsolv with D-Wave subQUBO solver



t_{QPU} = 49 s

Modified qbsolv with D-Wave subQUBO solver



t_{QPU} = 2.45 s

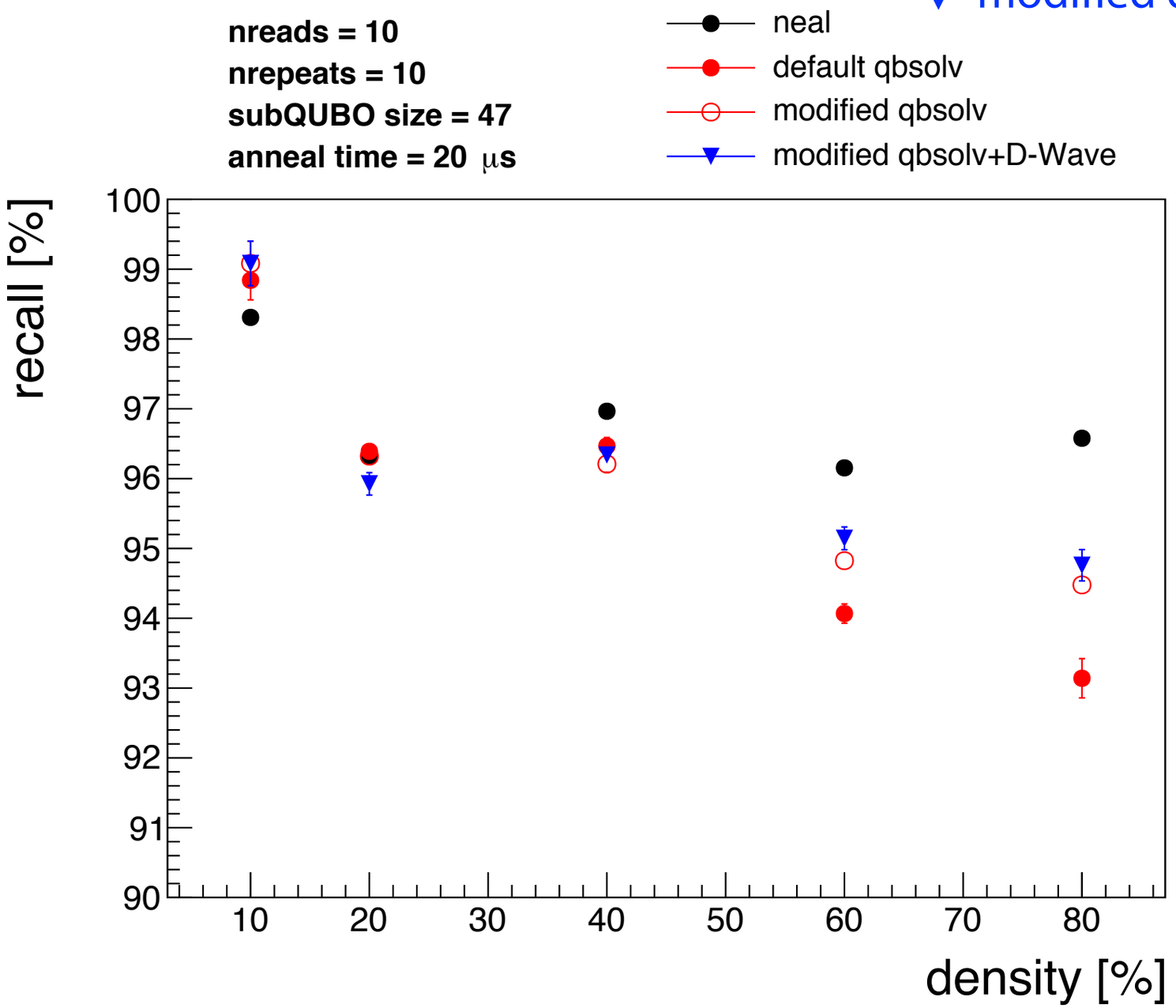
Default qbsolv:

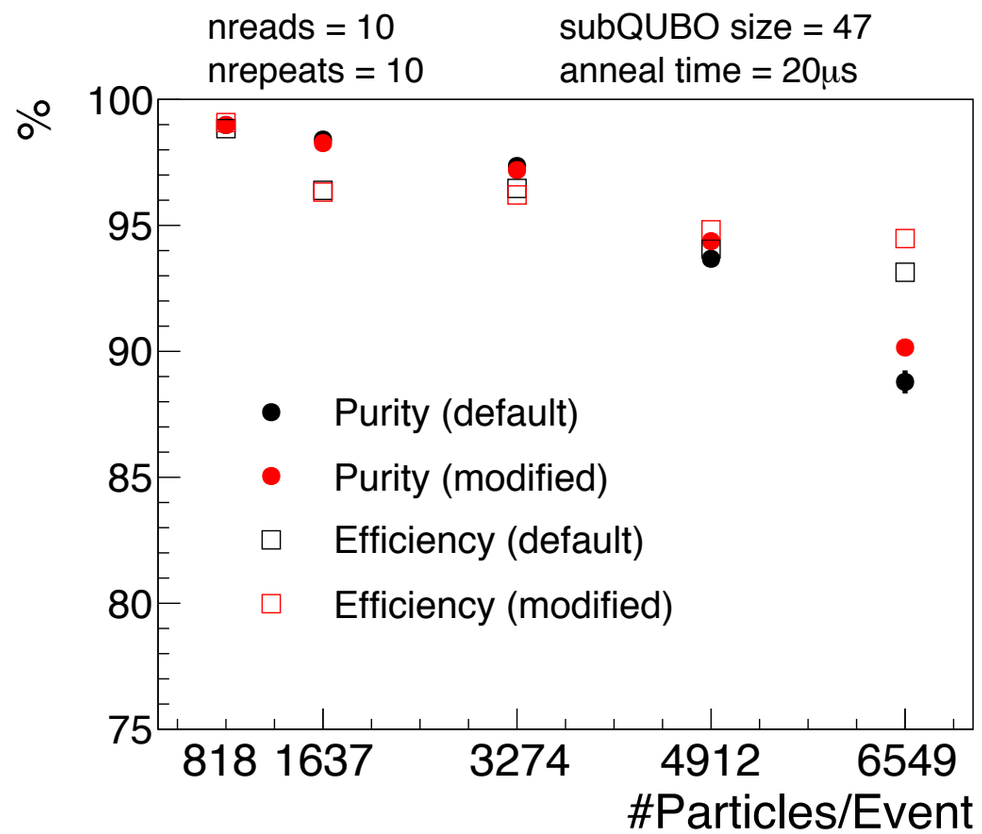
- need *very long* QPU time before converging
- quite often stopped at intermediate energy due to very small or no improvement

Modified qbsolv:

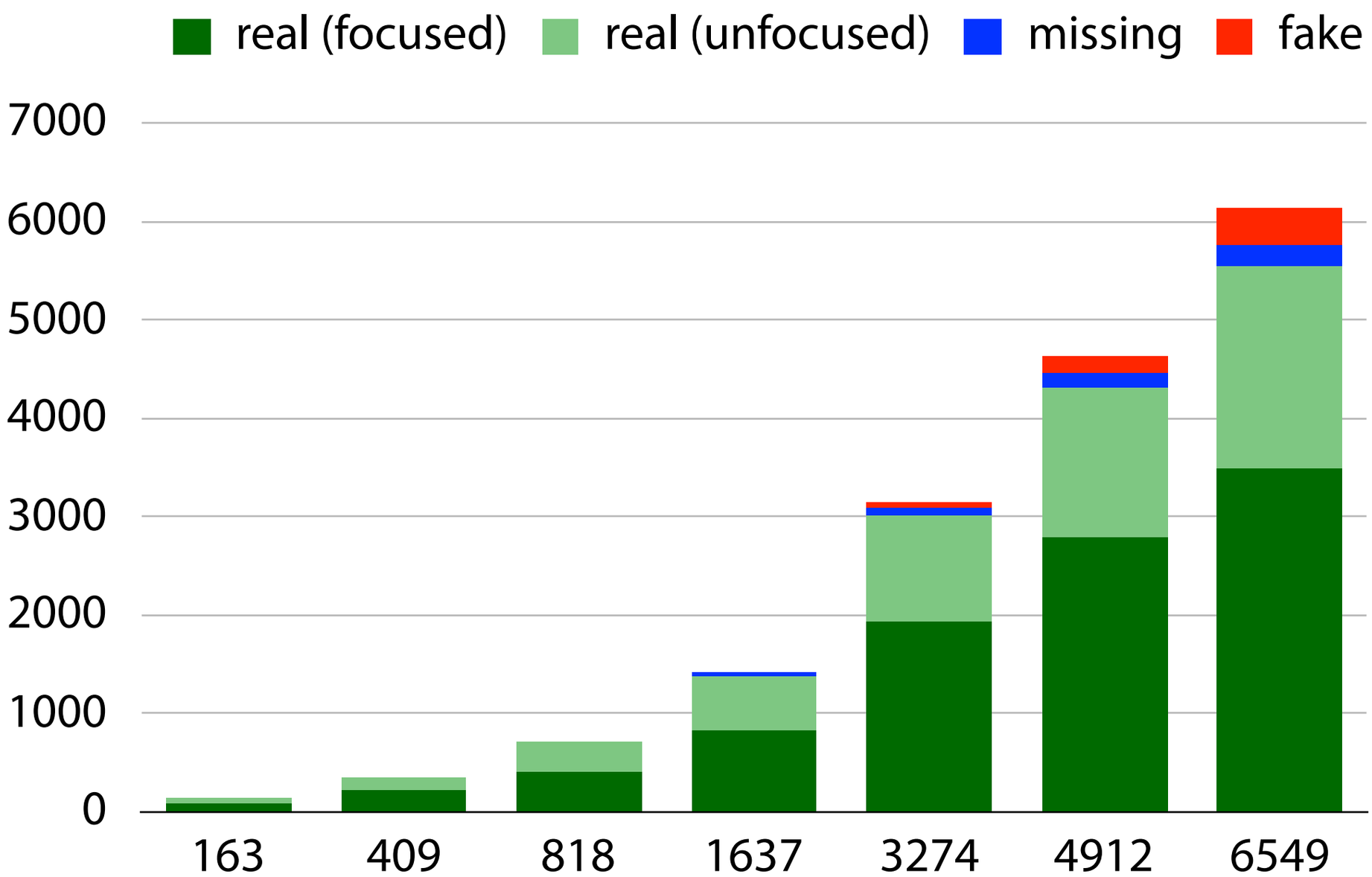
- much *shorter* QPU time
- much more stable behavior with good convergence

● neal SimulatedAnnealingSampler w/o qbsolv
● default qbsolv (tabu solver as subQUBO solver)
○ modified qbsolv (tabu solver as subQUBO solver)
▼ modified qbsolv (D-Wave as subQUBO solver)

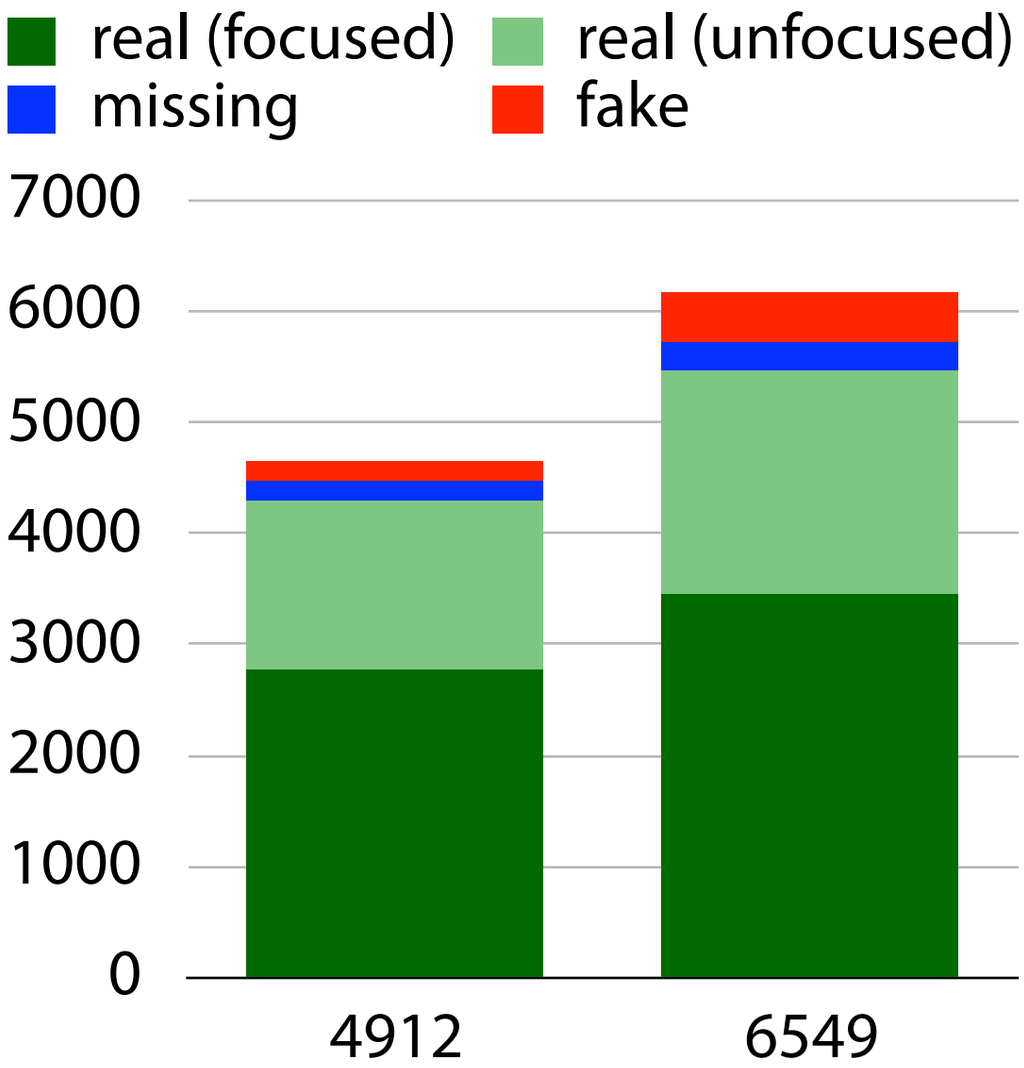




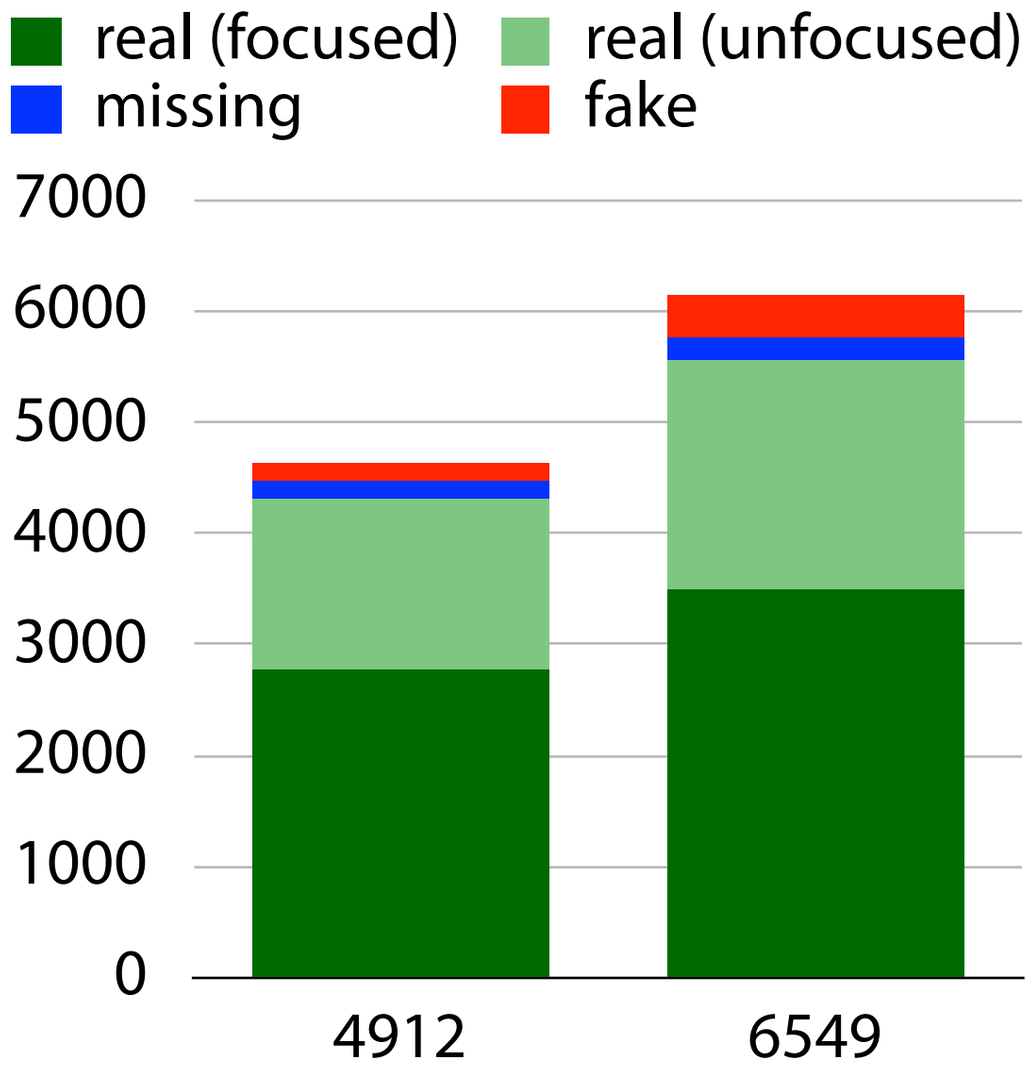
Modified qbsolv appears to perform better than default at high density


real (focused)
real (unfocused)
missing
fake
7000
6000
5000
4000
3000
2000
1000
0
163
409
818
1637
3274
4912
6549

qbsolv修正前



qbsolv修正後



fakeが13%減少

missingが19%減少