Distributing the Simulated Annealing workload for Quantum Unfolding in HEP

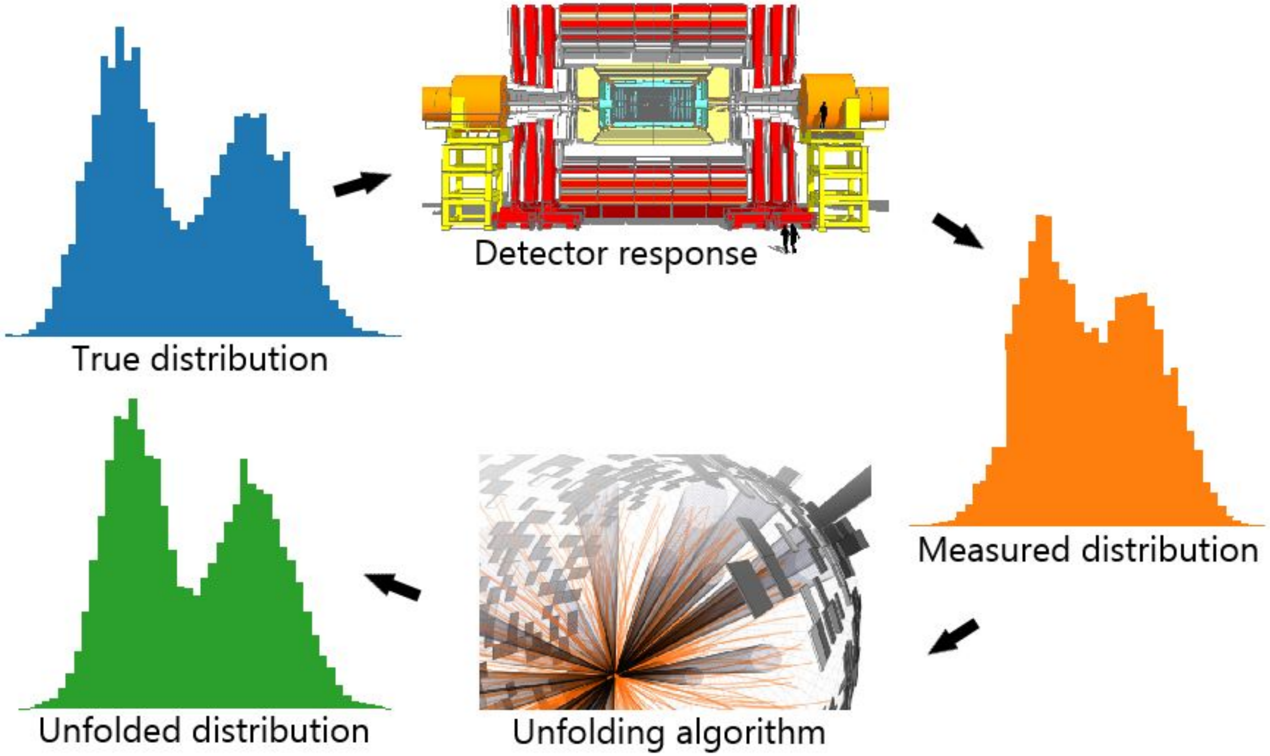
International Symposium on Grids & Clouds 2025 (ISGC25) *18 March 2025, Taipei*

Simone Gasperini, *PhD student in Data Science and Computation (UniBO & INFN Bologna)*

Tommaso Diotalevi (*UniBO & INFN Bologna*)
Marco Lorusso (*UniBO & INFN CNAF*)

Unfolding problem

- In **High-Energy Physics** (HEP) experiments each measurement apparatus has a unique signature in terms of *detection efficiency*, *resolution,* and *geometric acceptance*
- The overall effect is that the distribution of some measured observable in a physical process is *biased* and *distorted*
- **Unfolding** is the mathematical technique to correct for this distortion and recover the original distribution



$$\boxed{\vec{\mu} = R\vec{x}}$$

reconstructed histogram (≈ measured histogram)

response matrix

unfolded histogram (≈ true histogram)

Unfolding techniques

Classical methods used in HEP:

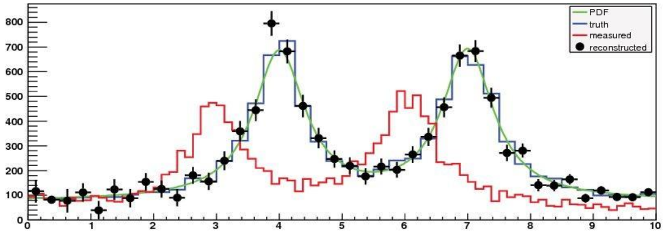
- Standard matrix inversion (never used in practice) ➡ $\vec{x} = R^{-1}\vec{d}$ with $\vec{d} \sim \vec{\mu}$
- Bin-by-bin unfolding (never used in practice)

- ill-conditioned problem
- large statistical fluctuations

- Likelihood-based unfolding (SVD)
- Iterative Bayesian unfolding (IBU)

RooUnfold

The ROOT Unfolding Framework



Quantum-based methods:

- First proof-of-concept attempt in 2019: the model worked only for small-size problems instances and the previous generation of D-Wave quantum annealing devices was used
- QUnfold: our open-source implementation ready to be used by HEP scientists

QUnfold QM formulation

Log-likelihood maximization unfolding: $\max_{\vec{x}} \left(\log \mathcal{L}(\vec{\mu}|\vec{d}) + \lambda \mathcal{S}(\vec{\mu}) \right)$

reconstructed histogram

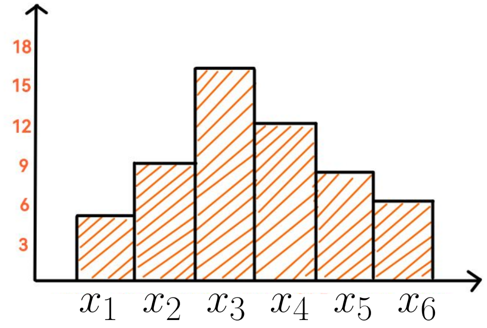
measured histogram

regularization term (avoid noise overfitting)

Quadratic Model minimization: $\min_{\vec{x}} \left(||R\vec{x} - \vec{d}||^2 + \boxed{\lambda ||G\vec{x}||^2} \right)$

Tikhonov regularization: discrete 2nd order derivative (Laplacian operator G)

- Classical minimization model of a regularized quadratic function
- $\vec{x}$ is the vector of integer numbers representing the unfolded histogram



QUnfold QUBO formulation

To get the QUBO model from the integer-variables QM, a “binarization” process based on the standard **logarithmic encoding** of the integer numbers to bit-strings is performed

$$\vec{a} = -2R^T\vec{d}$$
$$B = R^T R + \lambda G^T G$$

$$\longrightarrow$$

$$H(\vec{x}) = \sum_i a_i x_i + \sum_{i,j} b_{ij} x_i x_j = \vec{a} \cdot \vec{x} + \vec{x}^T B \vec{x}$$

The scaling of the number of binary variables (= required logical qubits) is:

- $O(N_{\text{bins}})$: linear with the number of bins in the histogram
- $O(\log(N_{\text{entries}}))$: logarithmic with the number of entries in the histogram

Example:
- Gaussian distribution
- 20 bins
- 5M entries
- $N_{qubits} \approx$ 350

QUnfold software

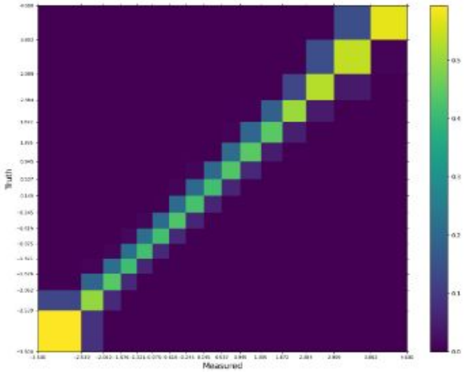
- Package implemented entirely in **Python**
- Designed to work also on real LHC data
- Based on **NumPy**, but compatible with **ROOT**
- **Support for distributed computation with Dask**
- Open-source repository on GitHub
- Available on PyPI (`pip install QUnfold`)

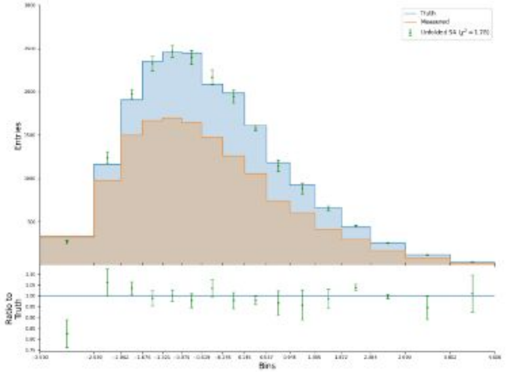
```
from qunfold import QUnfolder

# Define your input response matrix and measured histogram as numpy arrays
response = ...
measured = ...
binning = ...

# Create the QUnfolder object and initialize the QUBO model
unfolder = QUnfolder(response, measured, binning, lam=0.1)
unfolder.initialize_qubo_model()

# Run one of the available solvers to get the unfolding result
sol, cov = unfolder.solve_simulated_annealing(num_reads=100)
```





Available Solvers:

1) **Simulated Annealing**
2) D-Wave Quantum Annealing
3) D-Wave Hybrid solver
4) Classical Gurobi solver

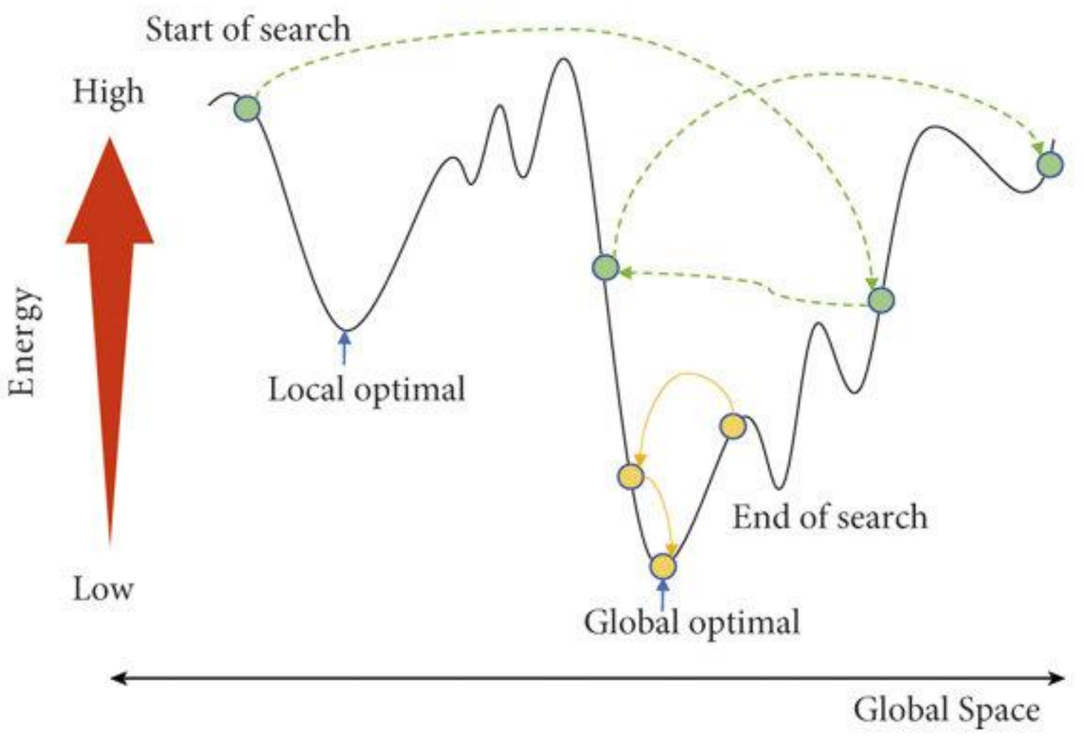
Simulated Annealing

SA is a heuristic probabilistic technique for approximating the global optimum of a given optimization problem function in a large search space

D-Wave SA implementation for **QUBO/Ising** problems:

1) Initialize the system to random configuration and high temperature
2) Apply a perturbation and compute the energy
$$\Delta E = f(S') - f(S)$$
3) Accept the new configuration with probability
$$P = \begin{cases} 1 & \text{if } \Delta E < 0 \\ e^{-\Delta E/T} & \text{if } \Delta E \geq 0 \end{cases}$$
4) Reduce the temperature of the system

repeat for a fixed number of iterations

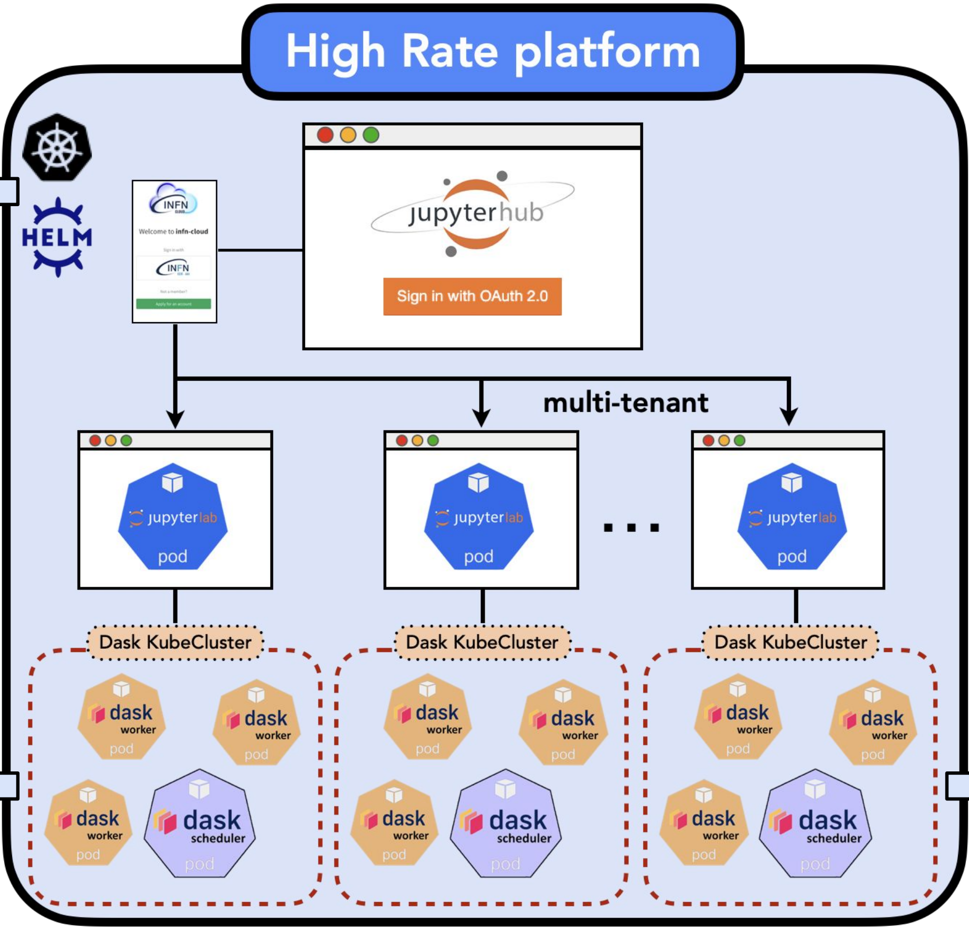


→ probabilistic guarantee to converge to the global optimal solution for long smooth temperature schedules

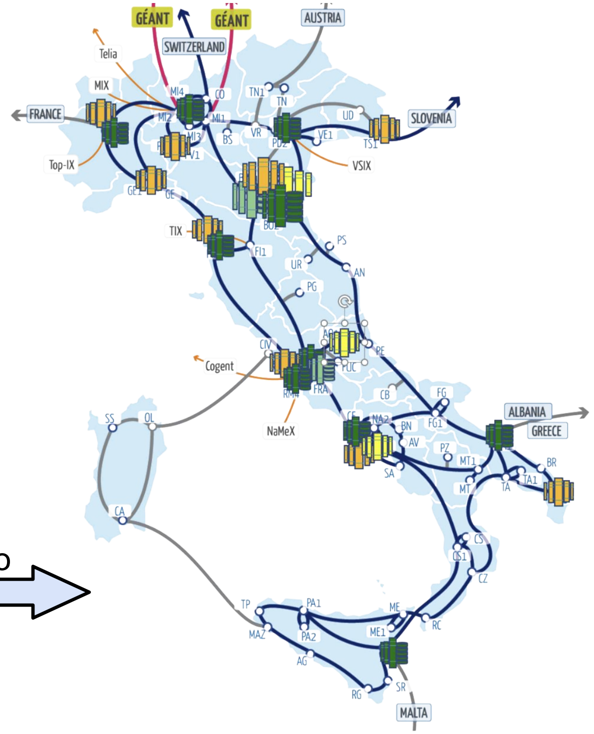
Distributed SA workload - 1

Deployment of the Kubernetes scalable resources handled via HELM Charts

Users can choose how many **Dask worker nodes** to use for distributing the SA workload

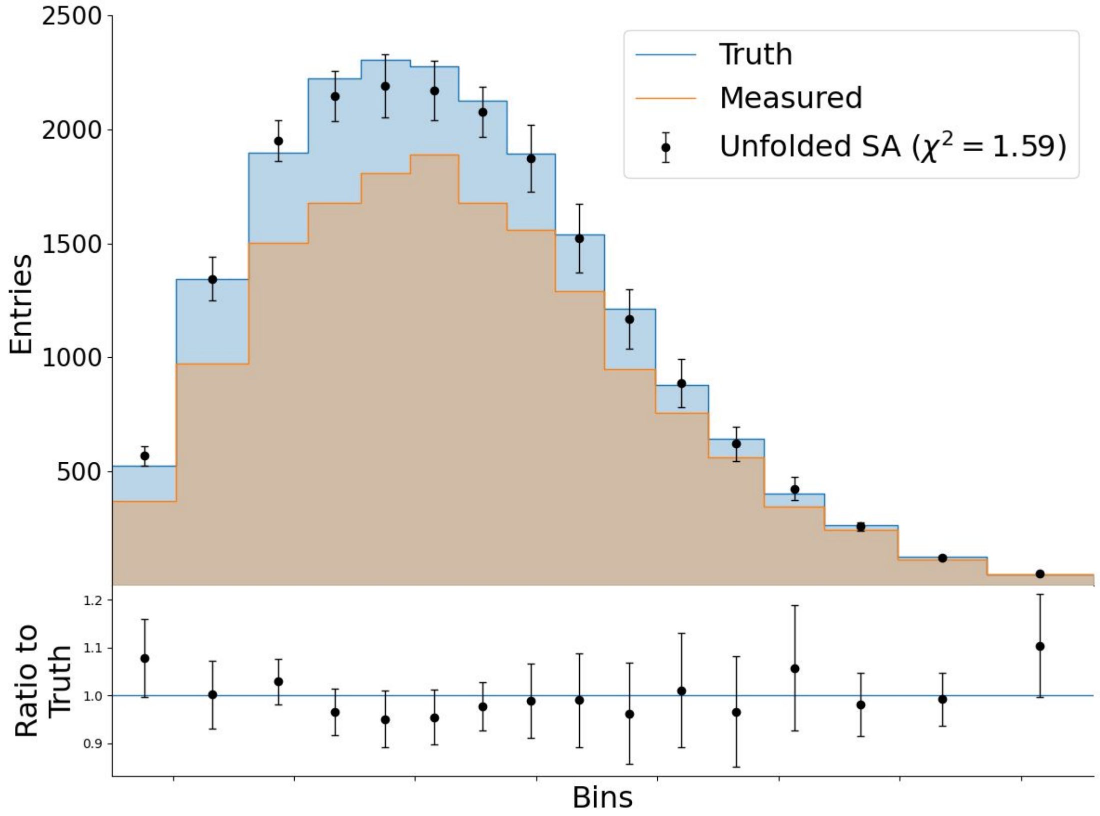


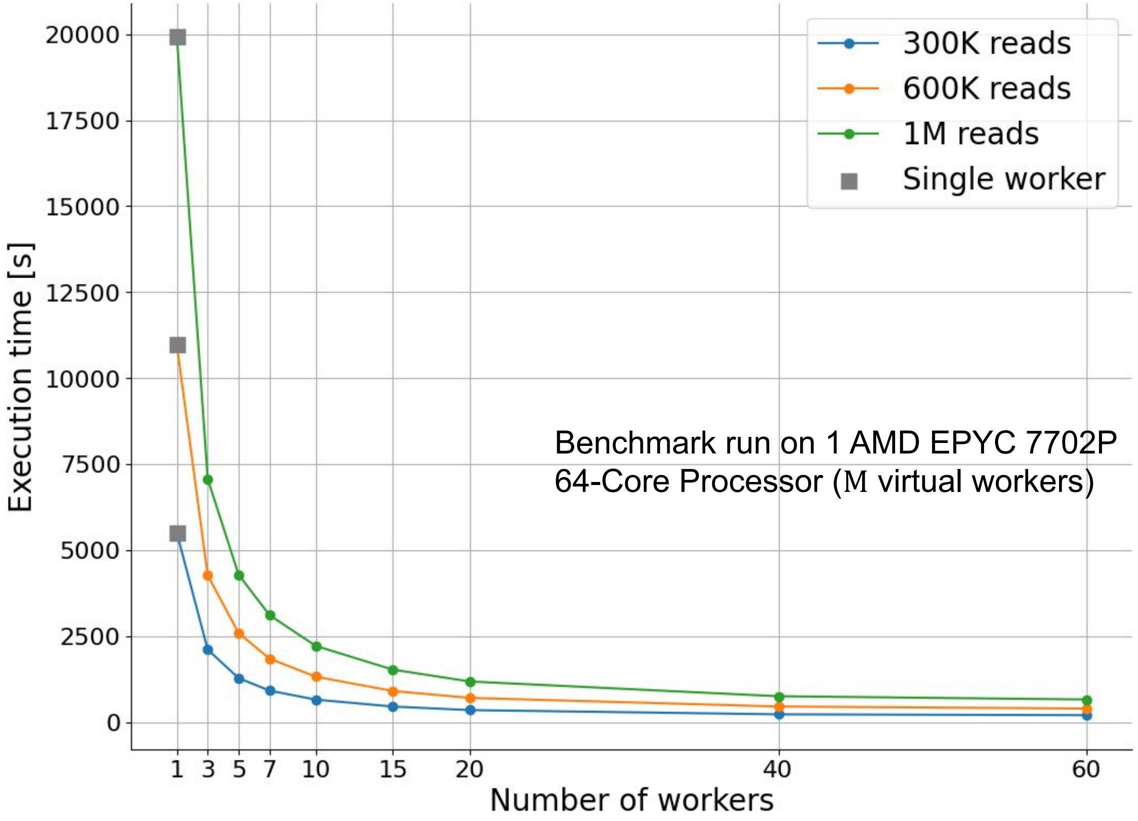
offloaded to



Distributed SA workload - 2

The unfolding solution is computed as an **average over N independent reads** of the final outcome returned by the SA algorithm: the workload is distributed by running N/M reads on each one of the M Dask worker nodes





Benchmark run on 1 AMD EPYC 7702P 64-Core Processor (M virtual workers)

Conclusion

- The proposed **optimization-based approach** represents an interesting framework to formulate and solve the statistical unfolding problem in HEP (and beyond)
- QUnfold offers a simple set of tools to test this innovative unfolding technique by using both **classical heuristics** and **real hybrid/quantum solvers**
- **Distributing the Simulated Annealing workload** over a high-throughput infrastructure can bring a significative speed-up in terms of execution time

GitHub

https://github.com/Quantum4HEP/QUnfold

Thank you for your attention!

This work is supported by ICSC - Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union - NextGenerationEU

Backup

Quantum Annealing

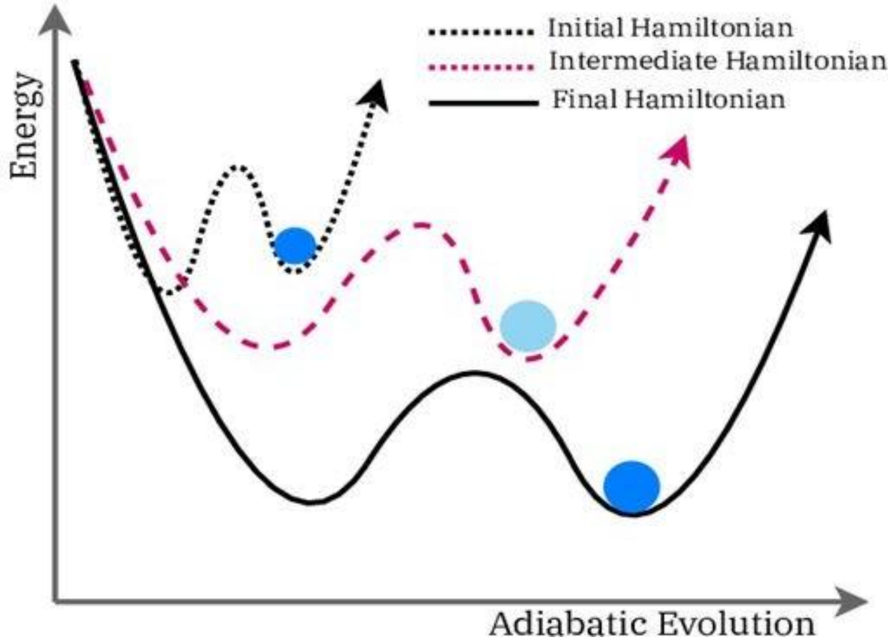
QA is a quantum optimization process to find the global minimum of an objective function $O(\vec{x})$ over a set of candidates states

1. The quantum-mechanical system is prepared in the known ground-state of an **initial Hamiltonian**

1. The problem solution is encoded in the ground-state of a **final Hamiltonian**

1. The quantum system evolution is controlled by the following time-dependent Hamiltonian:

$$H(t) = A(t)H_{init} + B(t)H_{fin} \qquad A(t)\downarrow \quad B(t)\uparrow$$

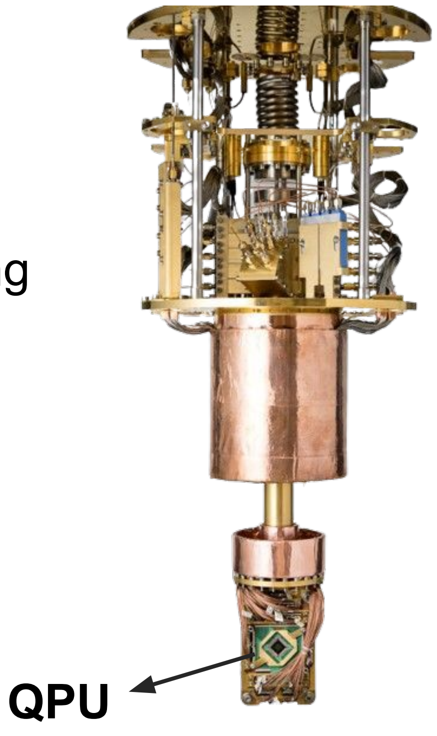


Quantum Adiabatic theorem
"If the evolution is slow enough, the quantum-mechanical system stays close the ground-state of the instantaneous Hamiltonian"

Simone Gasperini - simone.gasperini4@unibo.it

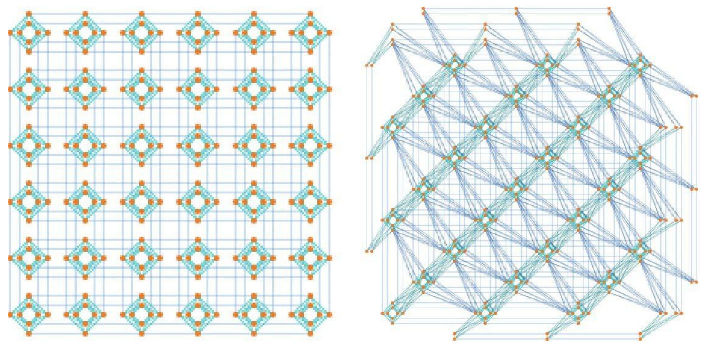
D-Wave QA system

- Shielded from external electric, magnetic, and thermal effects
- Isolated from floor vibrations
- Controlled by a specialized wiring and read-out electronics



Quantum Processing Unit (QPU)

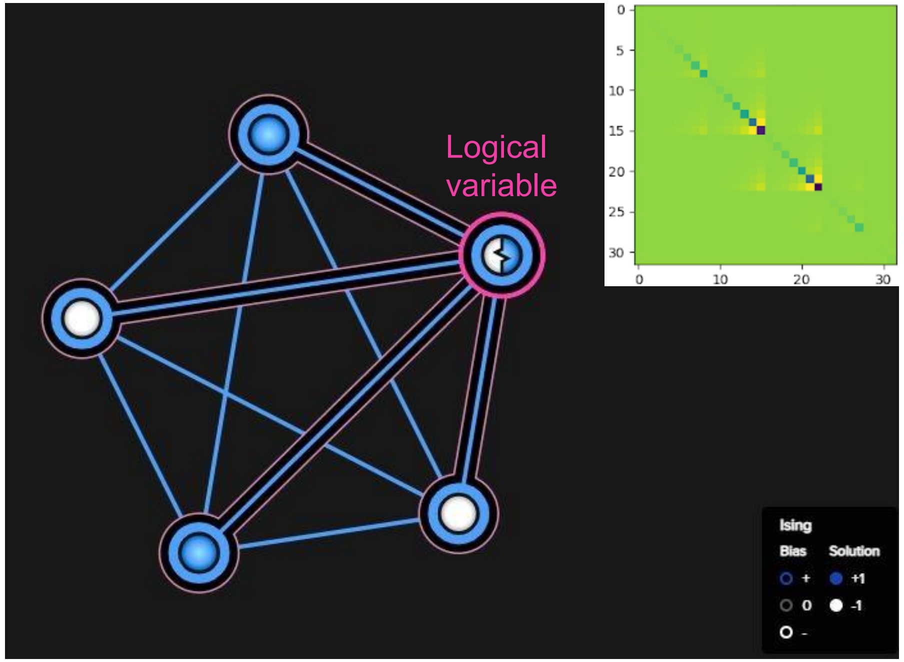
- Lattice of superconducting qubits
- Cooled down to ~15 mK by liquid Helium refrigeration system
- < 25 kW total power consumption
- <u>Physical interaction between qubits limited by a fixed topology</u>



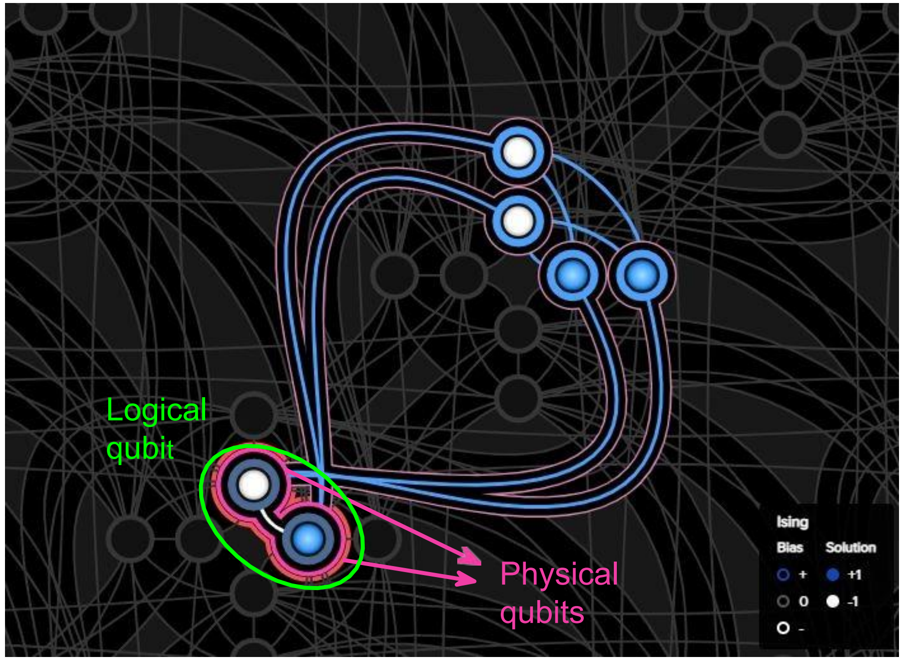
Graph embedding

Logical qubit ≠ Physical qubit

"Embedding" is the process of mapping a **source graph** (problem topology) to a **target graph** (QPU topology): <u>a cluster of physical qubits may represent a single logical qubit/variable</u>



Problem graph topology



Embedding on QPU Pegasus graph topology

QUnfold for HEP data

Dataset:

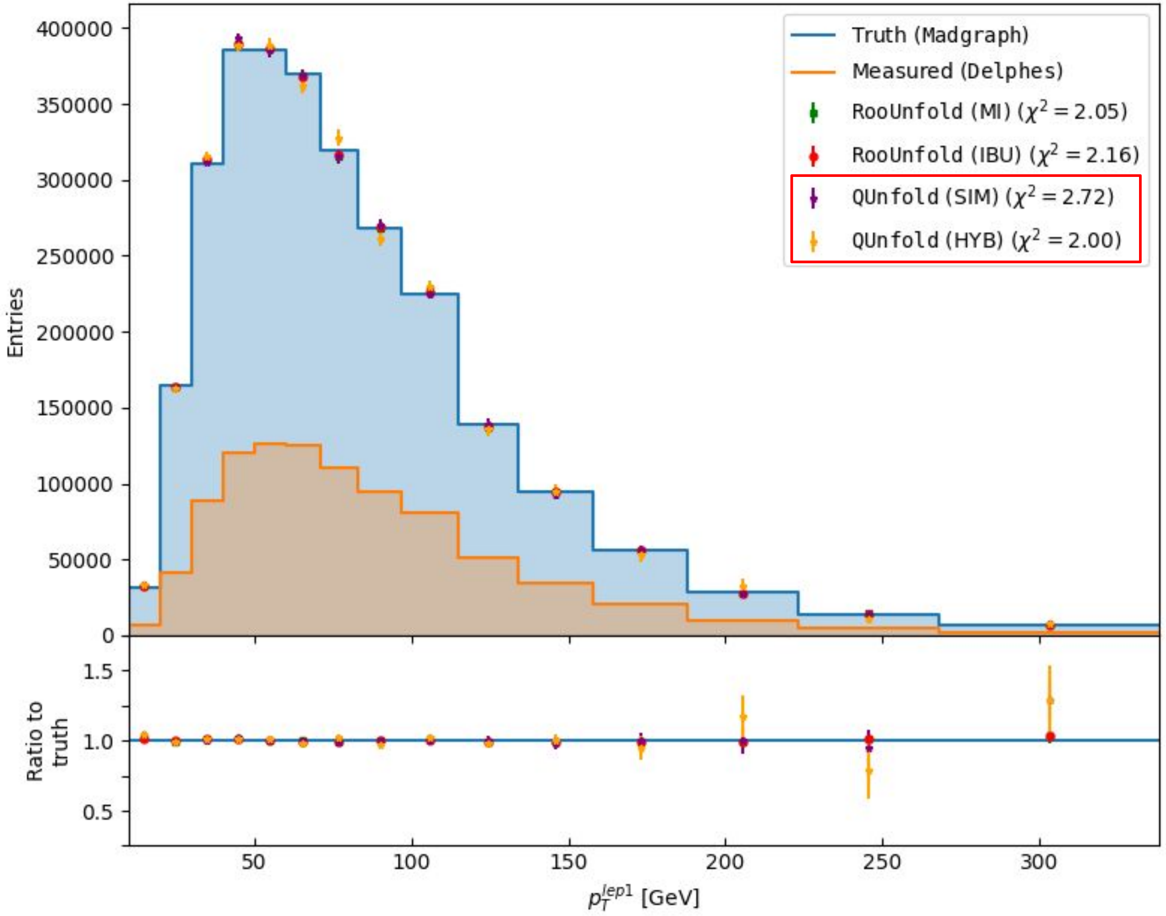
- Simulate $t\bar{t}$ **process** in the *dileptonic channel* (**2 leptons** and at least **2 *b*-jets** required in the final state)
- Generete detector-level ATLAS data (***Delphes*** sim.) and truth-level samples (***MadGraph*** gen.)
- $\approx$ 2.5M entries data are used to generate reco/truth level

Technique:

- The simulated annealing and hybrid solvers are used (quantum annealer solver is work in progress)
- The results are compared to classical HEP unfolding methods: *MI* and *IBU*, using the RooUnfold framework
- Toy Monte Carlo experiments are run to compute the covariance matrix for evaluating the quality of the result (X^2 test) and the statistical errors associated to the unfolding method

Unfolding results

Leading lepton p_T



Subleading lepton p_T