Introducing Qibo

from quantum circuits to machine learning arXiv:2009.01845

Stefano Carrazza

29th November 2021

ACAT 2021

Introduction

Introduction

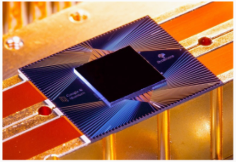
From a practical point of view, we are moving towards new technologies, in particular hardware accelerators:

CPU

GPU

FPGA/ASIC

Quantum chip



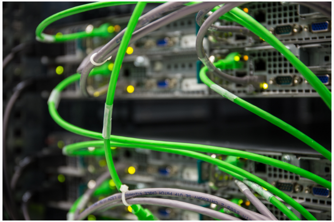
Moving from general purpose devices $\Rightarrow$ application specific

Challenges

However, there are several challenges:

- simulate efficiently algorithms on classical hardware for QPU?
- control, send and retrieve results from the QPU?
- error mitigation, keep noise and decoherence under control?

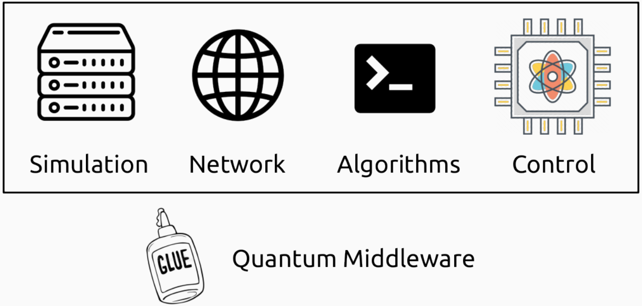




How can we interact with QPU?

Solution:

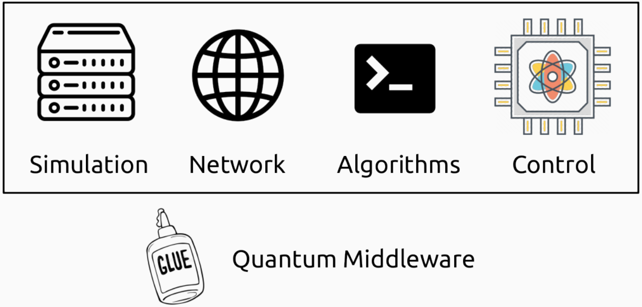
Construct a Quantum Middleware:



How can we interact with QPU?

Solution:

Construct a Quantum Middleware:

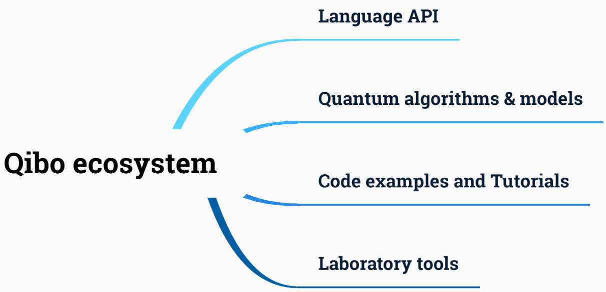


⇒ **Qibo: an open-source full-stack middleware.**

Introducing Qibo

Introducing Qibo

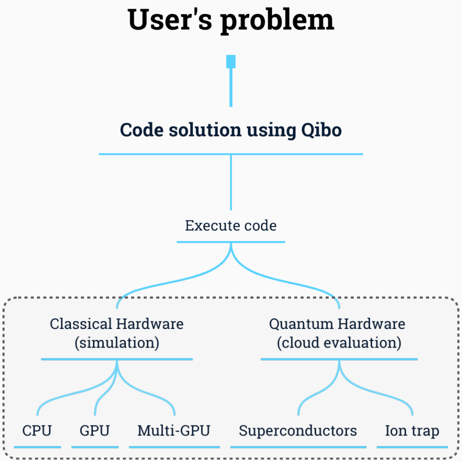
Qibo is an open-source full stack **API** for quantum simulation and hardware control.
It is platform **agnostic** and supports **multiple backends**.



https://github.com/qiboteam/qibo https://arxiv.org/abs/2009.01845

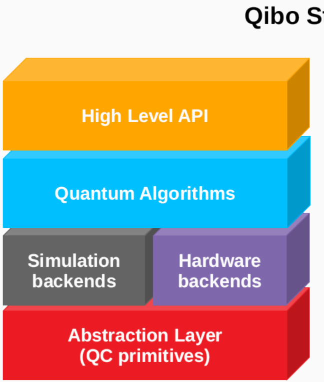
Los Alamos
NATIONAL LABORATORY
CERN
UNIVERSITÀ DEGLI STUDI DI MILANO
INFN
QILIMANJARO
QUANTUM TECH
IFAE
Institut de Física d'Altes Energies
UNIVERSITAT DE BARCELONA
BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación
TII
Technology Innovation Institute
CQT
Centre for Quantum Technologies
National University of Singapore
SG
NSCC
National Supercomputing Centre
Agency for Science, Technology and Research
SINGAPORE
Universidade Federal do Rio de Janeiro
UBA
Universidad de Buenos Aires

Abstractions in Qibo



- Single piece of code
- Automatic deployment on simulators and quantum devices
- Plugin backends mechanism

Abstractions in Qibo



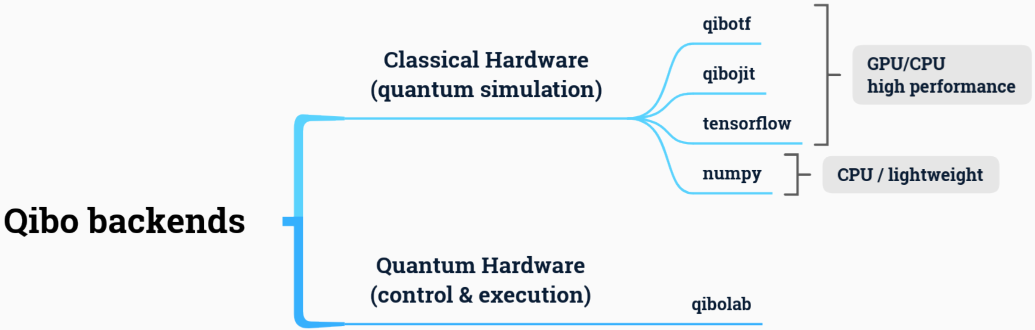
Interface for users: model definition and execution.

Implementation of algorithms based on quantum operations.

Backend specialization for classical and quantum hardware.

Code abstraction for circuit and gates representation.

Backends in Qibo



This layout opens the possibility to support:

- multiple classical and quantum hardware specifications
- hardware accelerators for simulation (single-GPU and multi-GPU)

numpy

pip install qibo

Simulator based on tensordot and linear algebra operations.

Features:

- Cross-architecture (x86, arm64, etc).
- Cross-platform.
- Fast for single-threaded operations.

tensorflow

pip install tensorflow

Simulator based on tensorflow primitives (einsum, matmul).

Features:

- Multithreading CPU.
- Single GPU.
- Gradient descent on quantum circuits.

qibotf

pip install qibotf

Simulator based on tensorflow custom operators in C++ and CUDA.

Features:

- Excellent single node performance.
- Multithreading CPU.
- Multi-GPU.
- Low memory footprint.

qibojit NEW

pip install qibojit

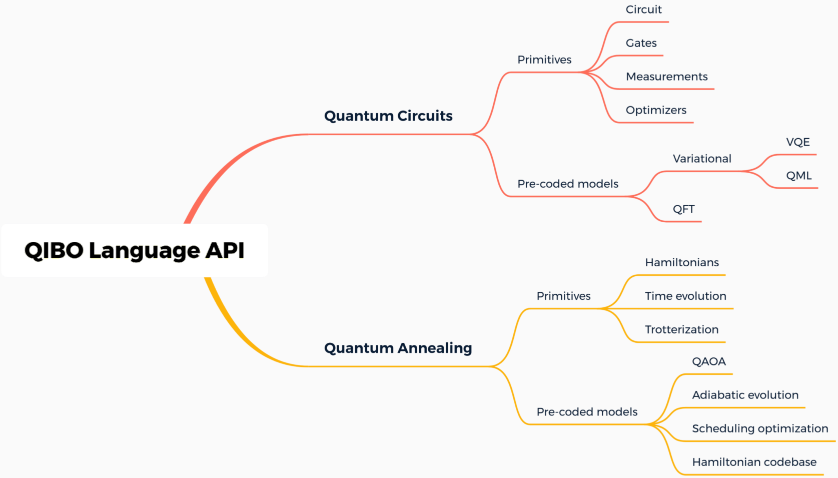
Simulator based on numba and cupy operations.

Features:

- Excellent single node performance.
- Multithreading CPU, single GPU and multi-GPU
- Cross-platform (just-in-time compilation)
- Works on NVIDIA and AMD GPUs.

Computational models in Qibo

Computational models in Qibo



Quantum Circuits

Quantum circuits

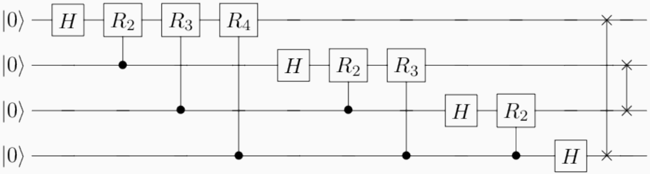
The quantum circuit model considers a sequence of unitary quantum gates:

$$\left|\psi'\right\rangle = U_2 U_1 \left|\psi\right\rangle \quad \rightarrow$$

The quantum circuit model considers a sequence of unitary quantum gates:

$$|\psi'\rangle = U_2 U_1 |\psi\rangle \quad \rightarrow$$

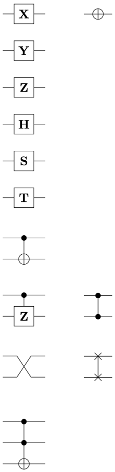
For example a Quantum Fourier Transform with 4 qubits is represented by



Models based on Grover's algorithms and Shor's factorization algorithms.

Quantum gates

- **Single-qubit gates**
    - Pauli gates
    - Hadamard gate
    - Phase shift gate
    - Rotation gates
- **Two-qubit gates**
    - Conditional gates
    - Swap gate
    - fSim gate
- Special gates: Toffoli



Operator	Gate(s)	Matrix
Pauli-X (X)	X	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$
Pauli-Z (Z)	Z	$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)	H	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Phase (S, P)	S	$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$
$\pi/8$ (T)	T	$\begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$
Controlled Z (CZ)	Z	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$
SWAP		$\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$

Quantum circuit simulation

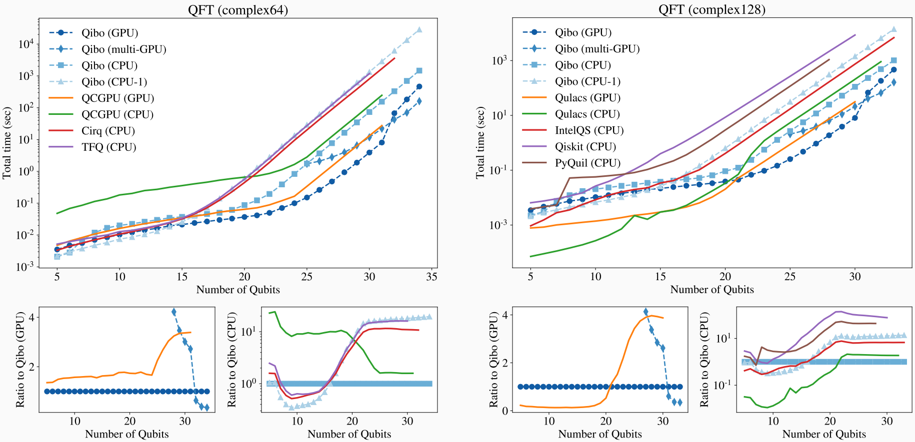
The final state of circuit evaluation is given by:

$$\psi'(\sigma_1, \ldots, \sigma_N) = \sum_{\boldsymbol{\tau}'} G(\boldsymbol{\tau}, \boldsymbol{\tau}')\psi(\sigma_1, \ldots, \boldsymbol{\tau}', \ldots, \sigma_N),$$

where the sum runs over qubits targeted by the gate.

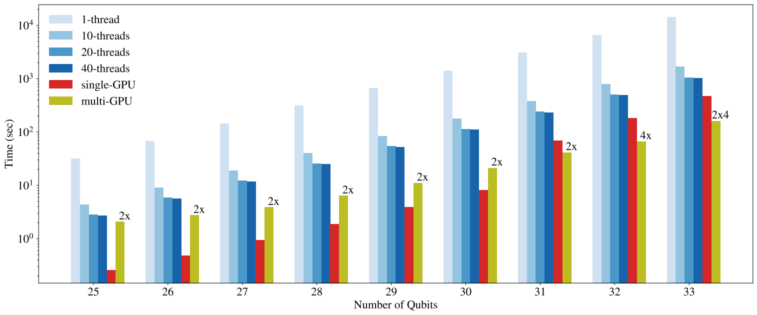
- Linear algebra approach.
- Possibility to parallelize and optimize operations.

Quantum circuit performance results



Quantum Fourier Transform performance.

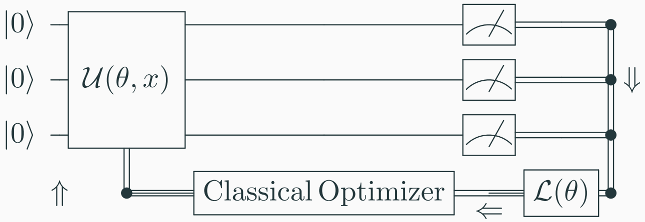
Multi-GPU trade-off



Quantum Fourier Transform performance.

Variational Quantum Circuits

Typical variational quantum circuits and data re-uploading algorithms:



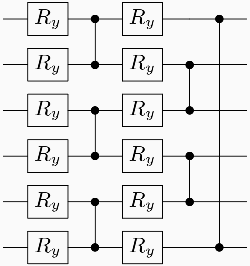
Define new parametric model architectures for quantum hardware:

$\Rightarrow$ **Variational Quantum Circuits & Quantum Machine Learning**

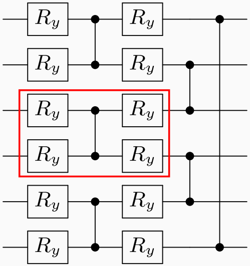
Variational circuit

Variational circuits are inspired by the structure of variational circuits used in **quantum machine learning**.

Standard Circuit



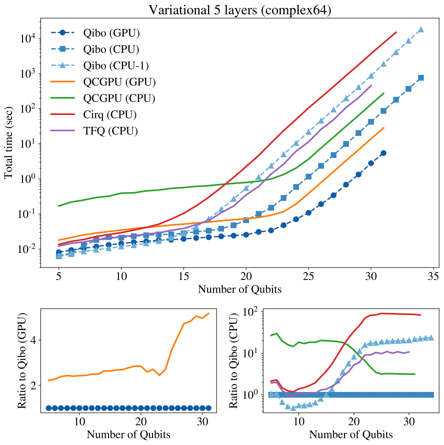
Gate fusion

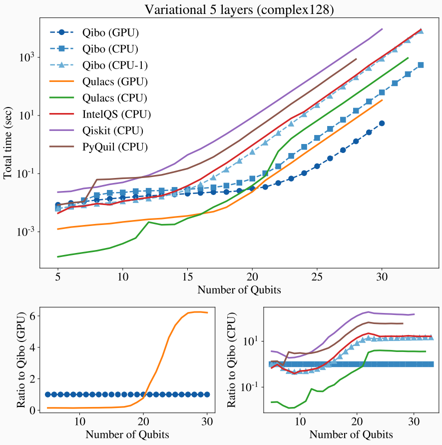


Qibo implements the gate fusion of four R_y and the controlled-phased gate, C_z

$\Rightarrow$ Qibo provides multi-qubit gate operators for CPU and GPU

Benchmarks

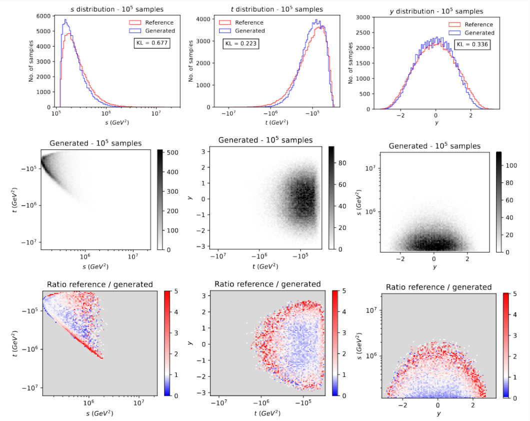




Variational circuit simulation performance comparison in single and double precision.

Summary of circuit-based built-in models in Qibo

- Variational quantum eigensolver
- Quantum approximate optimization algorithm (QAOA)
- Feedback-based algorithm for quantum optimization (FALQON)
- Quantum Neural Networks
  - Variational quantum classifier
  - Variational quantum regressor
  - Style-based quantum GAN



arXiv:2110.06933, arXiv:2011.13934

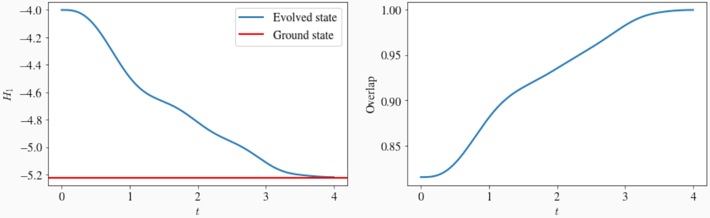
Quantum Annealing

Qibo features

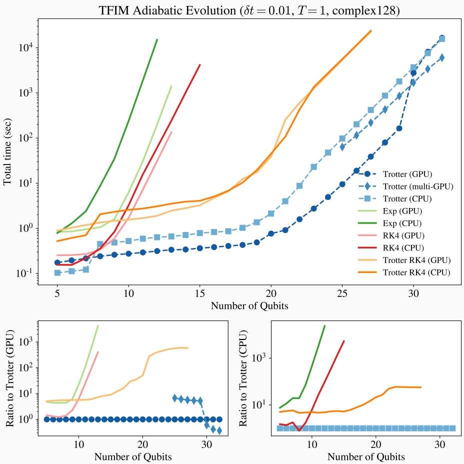
- **Annealing quantum processors**
  - Hamiltonian database
  - Time evolution of quantum states
  - Adiabatic Evolution simulation
  - Scheduling determination
  - Trotter decomposition

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H(s)|\psi(t)\rangle$$

$$H(t) = (1 - s(t))H_0 + s(t)H_1,$$



Adiabatic evolution



Adiabatic evolution performance using Qibo and TFIM for extact and Trotter solution.

Quantum hardware control

Software control for quantum hardware

Ideally, we would like to:

1. Define a circuit and/or algorithm.
2. Send and retrieve results from QPU:

Users

```python
import numpy as np
from qibo import models, gates

# create a circuit for N=3 qubits
circuit = models.Circuit(3)

# add some gates in the circuit
circuit.add([gates.H(0), gates.X(1)])
circuit.add(gates.RX(0, theta=np.pi/6))

# execute the circuit and obtain the
# final state as a tf.Tensor
final_state = circuit()
```

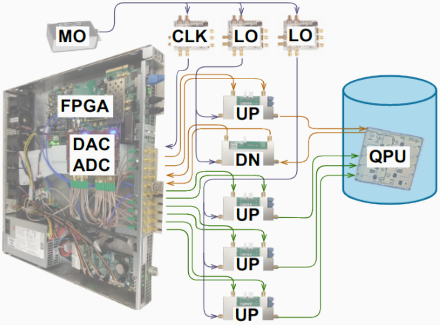
Circuit using Qibo

QPU

Software control for quantum hardware

However, from a hardware perspective this requires:

- Convert circuit into microwave pulse sequences.
- Operate multiple remote FPGA boards.
- Perform system calibration periodically.
- Schedule all operations.
- Reconstruct measurements (e.g. tomography).
- Store results / perform hybrid calculations.



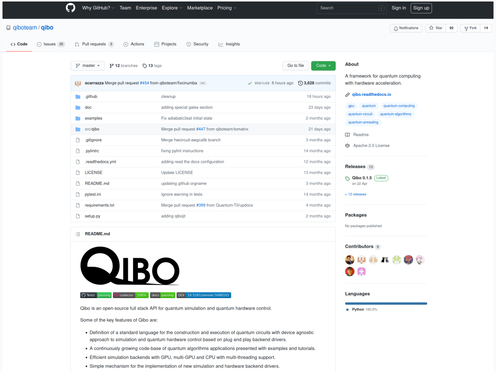
System layout from arXiv:2101.00071

Outlook

Outlook

Qibo is currently a framework for research:

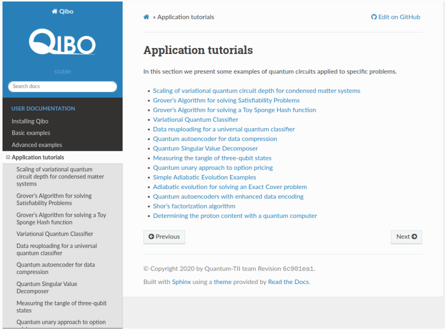
1. publicly available as an open-source code:
   https://github.com/qiboteam/qibo
2. Designed with several abstraction layers.
3. For fast prototyping of quantum algorithms.



Qibo tutorials

We provide several tutorials for:

- Variational circuits
- Grover's algorithm
- Adiabatic evolution
- Quantum Singular Value Decomposer
- ...



Visit:

https://qibo.readthedocs.io/en/stable/code-examples/applications.html

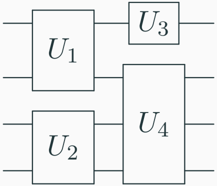
Thank you for your attention.

Rational for Variational Quantum Circuits

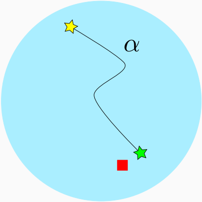
Rational:

Deliver variational quantum states $\rightarrow$ explore a large Hilbert space.

$$U(\vec{\alpha}) = U_n \ldots U_2 U_1$$



Near optimal solution

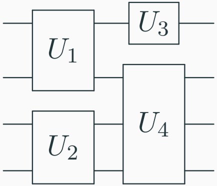


Rational for Variational Quantum Circuits

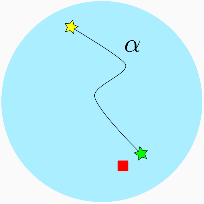
Rational:

Deliver variational quantum states → explore a large Hilbert space.

$$U(\vec{\alpha}) = U_n \dots U_2 U_1$$



Near optimal solution



Idea:

Quantum Computer is a machine that generates variational states.

⇒ **Variational Quantum Computer!**

Solovay-Kitaev Theorem

Let $\{U_i\}$ be a dense set of unitaries.

Define a circuit approximation to V:

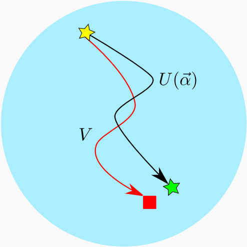
$$|U_k \dots U_2 U_1 - V| < \delta$$

Scaling to best approximation

$$k \sim \mathcal{O}\left(\log^c \frac{1}{\delta}\right)$$

where $c < 4$.

Optimal solution



$\Rightarrow$ The approximation is efficient and requires a finite number of gates.

Example for adiabatic quantum computation:

Lets consider the evolution Hamiltonian:

$$H(t) = (1 - s(t))H_0 + s(t)H_1,$$

where

- H_0 is a Hamiltonian whose ground state is easy to prepare and is used as the initial condition,
- H_1 is a Hamiltonian whose ground state is hard to prepare
- $s(t)$ is a scheduling function.

According to the adiabatic theorem, for proper choice of $s(t)$ and total evolution time T, the final state $|\psi(T)\rangle$ will approximate the ground state of the "hard" Hamiltonian H_1.