LIST

Principles of quantum computing based on Hamiltonian approaches

Applications to optimisation

S. Louise, S. Deleplanque, D. Vert

stephane.louise@cea.fr

17-21 April 2023,

Evolution of an isolated quantum system

Schödinger Equation (Dirac notation): $E = E_c + E_p$

$$i\hbar\frac{d}{dt}\left|\psi(t)\right\rangle = \frac{\hat{P}^2}{2m}\left|\psi(t)\right\rangle + V(\hat{X},t)\left|\psi(t)\right\rangle = \hat{\mathcal{H}}\left|\psi(t)\right\rangle \tag{1}$$

Stationary case:

- Eigen function of the Hamiltonien: $\hat{\mathcal{H}}\left|\varphi_n\right\rangle = E_n\left|\varphi_n\right\rangle$
- $\left|\psi\right\rangle = \sum_n \sum_j c_{n,j}\left|\varphi_{n,j}\right\rangle exp\left(\frac{-iE_n t}{\hbar}\right)$

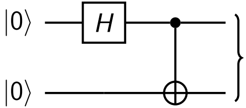
$$\left|\psi(t)\right\rangle = exp\left(-i\frac{\hat{\mathcal{H}}}{\hbar}t\right)\left|\psi(0)\right\rangle = \hat{U}(t)\left|\psi(0)\right\rangle \tag{2}$$

For Quantum Computing: $\left|\psi\right\rangle = \left|q_0\right\rangle \otimes \left|q_1\right\rangle \otimes \cdots \otimes \left|q_{n-1}\right\rangle$ with $\left|q_i\right\rangle = \alpha\left|0\right\rangle + \beta\left|1\right\rangle$
For spin states we have: $\left|\psi\right\rangle = \left|s_0\right\rangle \otimes \left|s_1\right\rangle \otimes \cdots \otimes \left|s_{n-1}\right\rangle$ with $\left|s_i\right\rangle = a\left|-1\right\rangle + b\left|1\right\rangle$

$$s_i = 2q_i - 1 \tag{3}$$

Gate-based Quantum Computing Vs Analog QC

$$|\psi_f\rangle = \hat{U}\,|\psi_i\rangle$$

- Gate-based computing
  - $|\psi_f\rangle = u_{cnot}\,u_{h\otimes 1}\,|0\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
  - 

$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
  - "Universal" quantum computing
- Analog Quantum Computing
  - More limited number of operators U: e.g. only 2 limits operators
  - Adiabatic Theorem

Theorem

A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum. (from Wikipedia)

Principes du calcul quantique adiabatique

Bases of adiabatic computing:

- $\mathcal{H}(t) = (1 - \frac{t}{\tau})\mathcal{H}_i + \frac{t}{\tau}\mathcal{H}_f$
- $\mathcal{H}(0) = \mathcal{H}_i$ et $\mathcal{H}(\tau) = \mathcal{H}_f$
- τ is large "enough"
- $\psi(t = 0) = \varphi_{i,0}$ base state of the hamiltonian $\mathcal{H}_i$
- $\Rightarrow$ *(th. adiabatique)* $\psi(\tau) = \varphi_{f,0}$

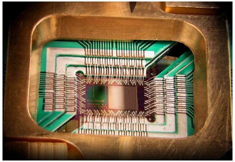
Theorem

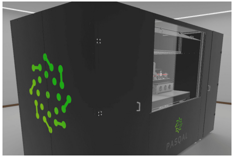
Given a "slow enough" evolution of the hamiltonian of the quantum computer between an initial hamiltonian $\mathcal{H}_i$ and a final hamiltonian $\mathcal{H}_f$ and initializing the system in the base state $\varphi_{i,0}$ of the initial hamiltonian, the system ends in the base state $\varphi_{f,0}$ of the final hamiltonian.

An usual case: spin systems

- $\mathcal{H} = \mathcal{P}(\sigma_0, \sigma_1, \cdots, \sigma_{n-1})$
- $|\psi\rangle = |s_0\rangle \otimes |s_1\rangle \otimes \cdots \otimes |s_{n-1}\rangle$ avec $s_i \in \{-1, 1\}$

D-Wave, Pasqal, Ising Hamiltonian





Canadian Enterprise funded in 1999. Provider of quantum computing solutions since 2009

- Superconducting flux qubits (nobium)
- 5 generations of QPU: 128, 1152, 2048, 5000+
- next generation: Advantage 2, 7440 qubits
- Principle: Quantum Annealing (QA)

- Rydberg Atoms (Rubidium)
- Prototype: 128 qubits
- Recently demonstrated +300 qubits
- Principle: Quantum evolution kernel

Ising Hamiltonian

$$\mathcal{H}_{Is} = \sum_{i=0}^{n-1} h_i \sigma_i + \sum_i \sum_j J_{ij} \sigma_i \sigma_j \tag{4}$$

Equivalency between Ising and QUBO

- Generalized Ising problem (2D): $\mathcal{H}(\mathbf{h},\mathbf{J},\mathbf{s}) = \sum_i h_i s_i + \sum_{i<j} J_{ij}\sigma_i s_j$ with s_k spins and $J_{i,j}$ coupling constants
- QUBO problems e.g. $f = x^T Q x = \sum_{i\leq j} q_{i,j} x_i x_j$ with $x_i \in \{0,1\}, \forall i$
- Variable substitutions: $s_i = 2x_i - 1$, gives:

$$\mathcal{H}(\mathbf{x}) = f(x) = \sum_{i\leq j} q_{ij} x_i x_j + C_H \quad \text{avec} \quad q_{ij} = \begin{cases} 4J_{ij} & i \neq j \\ -\sum_k 2J_{ki} - \sum_k 2J_{ik} + 2h_i & i = j \end{cases} \tag{5}$$

$$C_H = \sum_{ij} J_{ij} + \sum_i h_i$$

- Variable substitutions: $x_i = \frac{s_i+1}{2}$

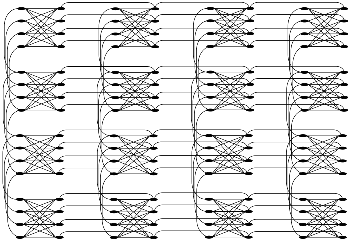
$$f(\sigma) = \mathcal{H} = \sum_{i<j} J_{ij} + \sum_i h_i + C_x \quad \text{avec} \quad J_{ij} = \frac{q_{ij}}{4}$$

$$h_i = \frac{1}{4}\left(2q_{ii} + \sum_k q_{ik} + \sum_k q_{ki}\right), C_x = \frac{1}{4}\sum_{ij} q_{ij} \tag{6}$$

D-Wave QPU architecture: limitations

- **D-Wave 1 (2011): $\leq$128 qubits, ≤ 312 couplers**
- **D-Wave 2 (2013): ≤ 512 qubits, ≤ 1472 couplers (Chimera) $\simeq 6$ couplers/qubit**
- **D-Wave 2X (2015): ≤ 1152 qubits, ≤ 3360 couplers (Chimera)**
- **D-Wave 2000Q (2017): ≤ 2048 qubits, ≤ 6016 couplers (Chimera)**
- **D-Wave Advantage (2020): ≤ 5640 qubits, ≤ 40484 couplers (Pegasus) $\simeq 15$ couplers/qubit**
- **D-Wave Advantage-2 (2023-2024): $\simeq +7000$ qubits, $\simeq 20$ couplers/qubit (Zephir)**

The Chimera graph:



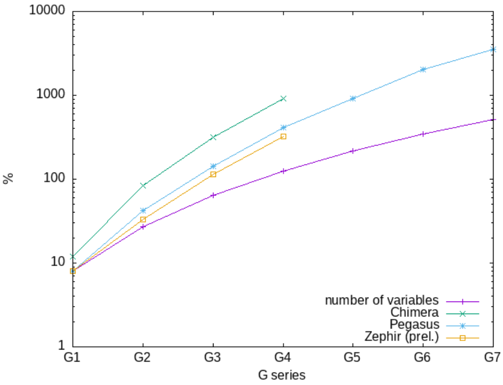
Adjacency matrix:

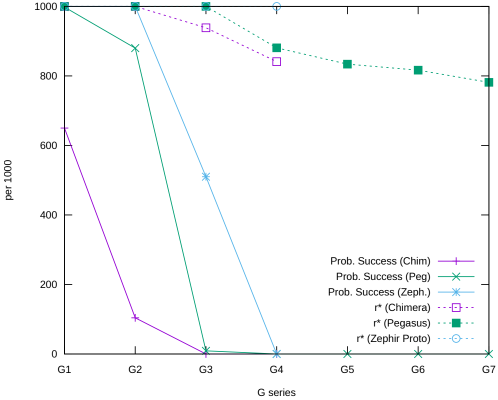
$$A_{\chi}^{32} = \left[\begin{array}{cccccccccccccccc}
1&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
&1&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
&&1&0&1&1&1&1&0&0&0&0&0&0&0&0\\
&&&1&1&1&1&1&0&0&0&0&0&0&0&0\\
&&&&1&0&0&0&0&0&0&0&1&0&0&0\\
&&&&&1&0&0&0&0&0&0&0&1&0&0\\
&&&&&&1&0&0&0&0&0&0&0&1&0\\
&&&&&&&1&0&0&0&0&0&0&0&1\\
&&&&&&&&1&0&0&0&1&1&1&1\\
&&&&&&&&&1&0&0&1&1&1&1\\
&&&&&&&&&&1&0&1&1&1&1\\
&&&&&&&&&&&1&1&1&1&1\\
&&&&&&&&&&&&1&0&0&0\\
&&&&&&&&&&&&&1&0&0\\
&&&&&&&&&&&&&&1&0\\
&&&&&&&&&&&&&&&1
\end{array}\right]$$

Limitation D-Wave computers (noise excepted)

To drive around limitations in number of couplers per qubit: utilization of logical qubits

- Complexify seriously the problem to solve
- Requires to find a good (the best?) embedding which is usually costly





Utilizing D-Wave computer for optimization, and beyond

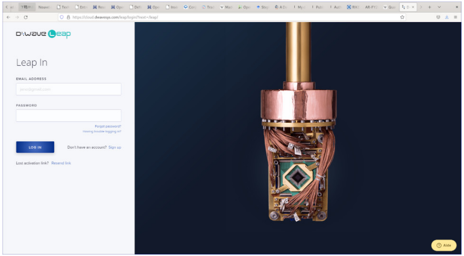
The QUBO/Ising formalism allows to address most of the optimization problems:

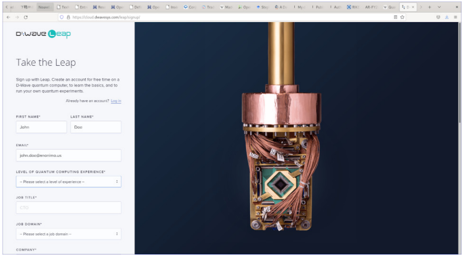
- Traveling Salesman Problem,
- Logistic problems,
- Graph coloring,
- ...

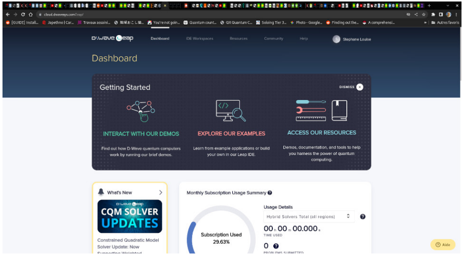
We can also apply it to other kinds of problems with a bit of imagination:

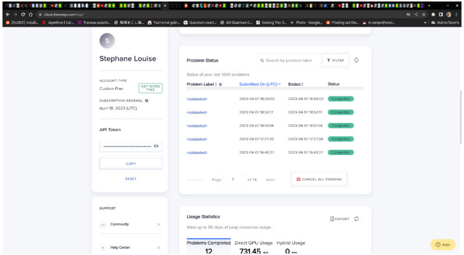
- Factorizing integers,
- Machine Learning,
- Linear algebra,
- ...

Setting a D-Wave Leap account and installing Ocean 1/3









Setting a D-Wave Leap account and installing Ocean 2/3

```
louise:~$ python3 -m venv ocean
louise:~$ . ocean/bin/activate
```

Now ocean is installed, but not yet ready to run

```
(ocean) louise:~$ dwave config create
Using the simplified configuration flow.
Try 'dwave config create --full' for more options.
(ocean) louise:~$ pip install dwave-ocean-sdk
```

Still not complete

Setting a D-Wave Leap account and installing Ocean 3/3

```
(ocean) louise:~$ dwave setup
Optionally install non-open-source packages and configure your environment.

Do you want to select non-open-source packages to install (y/n)? [y]:
...
The terms of the license are available online: https://docs.ocean.dwavesys.com/eula
Install (y/n)? [y]:
Installing: D-Wave Drivers
...
Install (y/n)? [y]:
Installing: D-Wave Problem Inspector
Successfully installed D-Wave Problem Inspector.

Creating the D-Wave configuration file.
Using the simplified configuration flow.
Try 'dwave config create --full' for more options.

Updating existing configuration file: /home/louise/.config/dwave/dwave.conf
Available profiles: defaults
Profile (select existing or create new) [defaults]:
Authentication token [skip]: [copy_token_from_dashboard]
Configuration saved.
```

Verifying D-Wave Ocean configuration

We can probe the available QPUs and solvers

```
(ocean) louise:~$ dwave solvers --list --all
DW_2000Q_6
DW_2000Q_VFYC_6
hybrid_binary_quadratic_model_version2p
hybrid_discrete_quadratic_model_version1p
hybrid_constrained_quadratic_model_version1p
Advantage_system6.1
Advantage2_prototype1.1
Advantage_system4.1
```

If you obtain this output, the configuration is done. To run a python script on a QPU, simply use

```
(ocean) louise:~$ python3 myPythonScript.py
...

```

First hand-on of an optimization problem: MaxCut

Max-cut problems aim at finding a partition of a given subset of node of a (connected) graph that cuts the highest number of edges

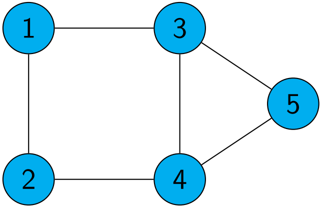
- Let $G = (V, E)$ be a graph
- $V = \{v_1, v_2, \cdots, v_n\}$ a set of nodes
- $E \in V \times V$ a set of edges
- MaxCut provides a partition that cut the highest number of edges in E

Solving with binary variables: $x_i \in \{0, 1\}$

- then $e_{ij} \in E, c_{i,j} = x_i + x_j - 2x_i x_j = \begin{cases} 0 & \text{if } v_i \text{ and } v_j \text{ in the same partition} \\ 1 & \text{if they end in different partitions} \end{cases}$
- To solve MaxCut it suffice to minimize $\mathcal{C} = \sum_{e_{ij} \in E} (-c_{ij})$

N.B.: $x_i^2 = x_i$

First D-Wave utilization for an optimization problem



- $c_{1,2} = x_1 + x_2 - 2x_1x_2$
- $c_{1,3} = x_1 + x_3 - 2x_1x_3$
- $c_{2,4} = x_2 + x_4 - 2x_2x_4$
- $\dots$
- $\mathcal{C} = -2x_1 - 2x_2 - 3x_3 - 3x_4 - 2x_5 + 2x_1x_2 + 2x_1x_3 + 2x_2x_4 + 2x_2x_4 + 2x_3x_5 + 2x_4x_5$

$$Q = \begin{pmatrix} -2 & 2 & 2 & 0 & 0 \\ 0 & -2 & 0 & 2 & 0 \\ 0 & 0 & -3 & 2 & 2 \\ 0 & 0 & 0 & -3 & 2 \\ 0 & 0 & 0 & 0 & -2 \end{pmatrix}$$

$$\mathcal{C} = \mathbf{x}^T Q \mathbf{x}$$

Execution on D-Wave QPU

The solution obtained by D-Wave's quantum annealer Advantage2_prototype1.1 is

	0	1	2	3	4	energy	num_oc.	chain_.
0	0	1	1	0	0	-5.0	335	0.0
1	0	1	1	0	1	-5.0	58	0.0
2	1	0	0	1	1	-5.0	92	0.0
3	1	0	0	1	0	-5.0	515	0.0

['BINARY', 4 rows, 1000 samples, 5 variables]

Code Leap IDE (D-Wave cloud)

```python
import numpy as np
import dimod

J= { (0,1):2.0, (0,2):2, (1,3):2, \
(2,3):2.0, (2,4):2.0, \
(3,4):2.0}
h= { 0:-2.0, 1:-2.0, 2:-3.0, 3:-3.0, 4:-2.0 }
model = dimod.BinaryQuadraticModel(h, J, 0.0, dimod.BINARY)

from dwave.system.samplers import DWaveSampler
from dwave.system.composites import EmbeddingComposite

sampler = EmbeddingComposite(DWaveSampler(solver='Advantage2_prototype1.1'))
sampler_name = sampler.properties['child_properties']['chip_id']
response = sampler.sample(model, num_reads=1000)
print("The solution obtained by D-Wave's quantum annealer",sampler_name,"is")
print(response)
```

Main messages

Analog quantum machines (a.k.a. "simulators") have pros:

- Have a high (-ish) number of qubits
- Well fitted to optimization problems
- Are also amenable to other kinds of problems (with a bit of imagination)

They also have cons:

- Non universal machines
- No known result of quantum advantage

On the power consumption side of things, is it nonetheless possible they can present some kind of advantage

N.B.: There exists a gate-based quantum computing simulation of the adiabatic theorem: the Quantum Approximate Optimization Algorithm (QAOA) [Farhi et Al., 2014]

Publications of our group:

- Vert, Sirdey, Louise, **Quantum Annealers are Cursed by their Qubits Interconnection Topologies.** ISVLSI 2020: 282-287
- Vert, Sirdey, Louise, **Benchmarking Quantum Annealing Against "Hard" Instances of the Bipartite Matching Problem.** Springer Nature Computer Science 2(2): 106 (2021)
- Louise, Sirdey, **A First Attempt at Cryptanalyzing a (Toy) Block Cipher by Means of by Means of Quantum Optimization approaches.**, Elsevier Comp. Science (4) 2023

Resources

https://github.com/stlouise/MaxMatch/blob/main/max-cut-school.py

Merci

CEA SACLAY
91191 Gif-sur-Yvette Cedex
France

Standard. + 33 1 69 08 60 00