Practical QC School 22

Combinatorial Optimization, Variational algorithms and QAOA through Atos's environment

Agostino Maria Cassese
HPC, AI & Quantum Computing Consultant
01/12/2022
Today's agenda

01. A dive into the Atos Quantum Program

02. Combinatorial Optimization, Variational algorithms and their Quantum Applications

03. QAOA and Max-Cut, simulation through a QLM
01. A dive into the Atos Quantum Program
Who we are and what we do
Atos
European at heart

- Founded in 1997, started in France, quickly became a global reality
- With 150,000+ employees
- Billing more than 11 billions yearly and investing 235m in R&D
- We deliver in 71 countries all across the globe
Atos Italia
Excellence & Acceleration

- Founded in 2011, focusing on innovation and development of the country
- With 1500+ employees in 5 locations
- Billing more than 300 millions yearly
- Organized in 6 industries to ensure excellence for our customers

Manufacturing

Resources, Utilities, Transport & Logistics

Financial & Insurance

Telecom, Media & Technology

Public sector & defence

Healthcare & lifestyle
Our customers
Big Data and Security – BDS Italy

We build realities with our clients

High Performance Computing

Fourth most powerful HPC with Leonardo CINECA
Leonardo Finmeccanica Da Vinci’s cluster

Quantum Computing

Q@TN with UniTrento and FBK
PQCS CINECA

Cybersecurity

ENI SAP management
Fastweb Law Enforcement Architecture

Artificial Intelligence

Automation and efficiency with Generali
Computer Vision
Leader in decarbonization
Improving energy efficiency and carbon emissions

Reducing carbon emissions by 50% by 2025, offsetting by 2028

Choosing hardware that ensure energy saving

Helping companies deal with sustainability challenges
Atos’s plan
Simulations to simplify real problems

“Let’s look at Quantum Computers as accelerators, not as independent systems”

- Hardware agnostic
- Simulators for an experimental approach
- Based on our HPC and simulations experience
- “All inclusive”
- Open platform
Atos’s Quantum Roadmap approach

- **2016**: Launch of Quantum program
- **2017**: Release and installation of the first QLM
- **2018**: Noisy simulations
- **2019**: myQLM & QLM user club
- **2020**: Simulated Annealing
- **2023+**: QPU offering
The Quantum Program in 5 steps

- Quantum Programming Platform
- Quantum Algorithms
- New Generation Architecture
- Quantum Safe Cryptography
- Quantum Expert Consulting Services
The Atos Quantum Program
Empowering international research on Quantum Computing

Quantum applications

Quantum algorithms

Next-generation architectures
A choice made by many

Thanks to our openness and scalability, we work with scientific communities, companies and universities to bring the future a step closer.
The Atos QLM: our appliance
The result of our commitment

Built to face real problems, now

Designed to interface with real QPUs

Universal programming environment
myQLM
The entry level of our Quantum offering

Freeware desktop solution
Entry level simulation
Scalability ~20 qubits
A new quantum metrics reference, applicable to all quantum circuits

Measures the efficiency to run a representative quantum application, instead of pure technical KPIs

Offering universal and free access to Q-score
What does it offer?
A complete programming environment

**Programming**
- **AQASM**
  - Assembly language to build quantum circuits
- **pyAQASM**
  - Python extension to AQASM
- **CIRC**
  - Binary format of quantum circuits
- **QLIB**
  - AQASM & pyAQASM libraries
- **CO problems class**
  - Describe any Combinatorial Optimization problem
- **INTEROP**
  - Connectors with other frameworks

**Optimization**
- **PBO**
  - Pattern based optimizer
- **NNIZER**
  - Topology constraint solver
- **Circuit Optimizer**
  - Generic circuit optimizer

**Simulation**
- **Simulators**
  - Digital QC Simulators
  - Quantum-Inspired Simulators
- **Physics**
  - Physical Noise models

**ProjectQ**

---
What does it offer?

Interoperability with python based frameworks

- Providing binders to translate Quantum circuits
- Back and fourth from QLM to QPUs and simulators
What are the possible ways to use the Atos QLM?
A multi-purpose system

**Learn**
Get acquainted with quantum computing

**Optimize**
Select the best quantum technology to solve your problem

**Atos QLM**

**Test**
Conceive new programs and debug them

**Run hybrid code**
Off-load the quantum-accelerable parts to the simulated QPU
Optimal battery operation
Batteries on a national scale for renewables energy

Smart charging for electric vehicles
Positioning of load-station on a wide-range map

Probabilistic Risk and Safety assessment
Probability estimation and application of optimal strategies
BMW Group

QC in manufacturing

Process scheduling
Optimization of major shops to maximize throughput

Quantum Circuits for ML training
Faster and optimal. Computer vision for inspections

Optimal order of production
Lower the production cost with best quality and lowest rework

21
Total Energies

QC in Oil&Gas Industry

Solving complex partial differential equations
Different applications in many camps of interest

Tackling decarbonization
Simulation of large complex molecules for efficient adsorbents

Supporters of the Atos Quantum Approach
Co-chair of the Atos QLM User Club
Reuse method for quantum circuit synthesis – AMMCS 2017
C. Allouche, M. Baboulin, T. Goubault de Brugière, and B. Valiron

Electron-Phonon Systems on a Universal Quantum Computer – Phys Rev Lett 2018
A. Macridin, P. Spentzouris, J. Amundson, and R. Harnik

Digital quantum computation of fermion-boson interacting systems – Phys Rev A 2018
A. Macridin, P. Spentzouris, J. Amundson, and R. Harnik

Synthesizing Quantum Circuits via Numerical Optimization – ICCS 2019
T. Goubault de Brugière, M. Baboulin, B. Valiron, and C. Allouche

q-means: A quantum algorithm for unsupervised machine learning – NIPS 2019
I. Kerenidis, J. Landman, A. Luongo, and A. Prakash

Function Maximization with Dynamic Quantum Search – QTOP 2019
C. Moussa, H. Calandra, and T. S. Humble

Methods for Classically Simulating Noisy Networked Quantum Architectures – QST 2019
I. Vankov, D. Mills, P. Wallden, and E. Kashefi

Practical implementation of a quantum backtracking algorithm – Sofsem 2020
S. Martiel, M. Remaud
To quantum or not to quantum: towards algorithm selection in near-term quantum optimization – QST 2020
C. Moussa, H. Calandra, and V. Dunjko

Arrays vs. Decision Diagrams: A Case Study on Quantum Circuit Simulators – ISMVL 2020
T. Grurl, J. Fuß, S. Hillmich, L. Burgholzer, and R. Wille

Considering decoherence errors in the simulation of quantum circuits using decision diagrams – ICCAD39 2020
T. Grurl, J. Fuß, and R. Wille

M. F. Serret, B. Marchand, and T. Ayral

Classification of the MNIST data set with quantum slow feature analysis – Phys Rev A 2020
I. Kerenidis and A. Luongo

Quantum Divide and Compute: Hardware Demonstrations and Noisy Simulations – IVLSI 2020
T. Ayral, F. -M. L. Régent, Z. Saleem, Y. Alexeev, and M. Suchara

Quantum CNOT Circuits Synthesis for NISQ Architectures Using the Syndrome Decoding Problem – Rev Comp 2020
T. G. de Brugiére, M. Baboulin, B. Valiron, S. Martiel, and C. Allouche

Quantum circuits synthesis using Householder transformations – CPC 2020
T. Goubault de Brugiére, M. Baboulin, B. Valiron, and C. Allouche
Stochastic Quantum Circuit Simulation Using Decision Diagrams – arXiv 2020
T. Grurl, R. Kueng, J. Fuß, and R. Wille

Qualifying quantum approaches for hard industrial optimization problems. A case study in the field of smart-charging of electric vehicles – arXiv 2020
C. Dalyac et al.

Practical Quantum Computing: Solving the Wave Equation Using a Quantum Approach – ACM Transactions on QC 2021
A. Suau, G. Staffelbach, and H. Calandra

Benchmarking quantum co-processors in an application-centric, hardware-agnostic and scalable way – arXiv 2021
S. Martiel, T. Ayral, and C. Allouche

Quantum Divide and Compute: Exploring the Effect of Different Noise Sources – SN COMPUT. SCI. 2021
T. Ayral, F.-M. L. Régent, Z. Saleem, Y. Alexeev, and M. Suchara

Quantum Computing: Towards Industry Reference Problems – Digitale Welt 2021
A. Luckow, J. Klepsch, and J. Pichlmeier

Qualifying quantum approaches for hard industrial optimization problems. A case study in the field of smart-charging of electric vehicles – arXiv 2020
C. Dalyac et al.
Click on the picture or visit myqlm.github.io
Combinatorial Optimization and Variational Algorithms
A first approach and applications in the Quantum Realm
Combinatorial Optimization

What are we talking about?

Operations Research
- Deals with development and application of analytical methods, to improve decision-making
- Employing techniques such as modeling, statistics and optimization
- Emphasis on practical applications

Optimization problems
- Find the best solution among many others
- Applied on real world scenarios but as mathematical models
- Algorithmic approach to solutions
- Becomes combinatorial when such variables are discrete
Combinatorial optimization
Applications

- Industrial problems
  Production planning
  Localization of facilities
  Stock management

- Organization problems
  Routing
  Scheduling of work shifts
  Management of water resources

- Optimal planning
  Network planning
  Structures planning
  VLSI design

- Economics decision-making problems
  Capitals allocation
  Purchase/Sell of products
  Choose investment
Combinatorial optimization
Modelling approach

1. Analysis of the problem
2. Construction of the model
3. Analysis of the model
4. Numeric solution
5. Validation of the problem
Combinatorial optimization
Structure of decision-making problems

Objective

Decision-making variables

Constraints

Mathematical model
A company sells 3 types of cars: *suv*, *convertible* and *minivan*. They use 2 type of production machines, *M1* and *M2*. To make a *suv*, they need 4 hours on *M1* and 3 on *M2*; for a *convertible* they need 8 hours on *M2* and for a *minivan* they need 2 hours on *M1* and 5 on *M2*. *M1* is available for 120 hours a week, while *M2* is available for 90 hours a week. The company wants to produce at least 3 *convertible* a week. They sell a *suv* for 1200€, a *convertible* for 1500€ and a *minivan* for 1800€. And then, they ask us to gain as much as possible from this production system.

**Objective**
- Maximize profit

**Variables**
- *Suv*, *convertible*, *minivan*
- *M1*, *M2*

**Constraints**
- *M1* for 120 hours
- *M2* for 90 hours
- At least 3 *convertibles*
Combinatorial optimization
Example of mathematical model

**Objective**
- Maximize profit

**Variables**
- Suv, convertible, minivan
  - M1, M2

**Constraints**
- M1 for 120 hours
- M2 for 90 hours
  - At least 3 convertibles

Maximization function

\[
maz Z = 1200x_1 + 1500x_2 + 1800x_3
\]

\[
\begin{align*}
\Sigma M1 & \leq 120; \\
\Sigma M2 & \leq 90; \\
x_2 & \geq 3
\end{align*}
\]

\[
\begin{align*}
4x_1 + 2x_3 & \leq 120 \\
3x_1 + 8x_2 + 5x_3 & \leq 90 \\
x_2 & \geq 3
\end{align*}
\]

\[
x_1, x_2, x_3 \geq 0
\]
Combinatorial optimization
Example of mathematical model

\[ \text{max } Z = 1200x_1 + 1500x_2 + 1800x_3 \]
\[ 4x_1 + 3x_1 + 8x_2 \geq 3 \]
\[ 2x_3 \leq 120 \]
\[ 5x_3 \leq 90 \]
\[ x_2 \geq 3 \]
\[ x_1, x_2, x_3 \geq 0 \]
Combinatorial optimization

The Simplex and why we are searching for new algorithms

- Used to resolve linear programming problems
- Utilizes a polytope to optimize problems
- Find solutions in the corners
- Widely used even if polynomial in worst case scenario
Combinatorial optimization
Solutions landscape, how many people to change a lightbulb?

- There are multiple algorithms and methods for CO and OR
- Sometimes even for the same problem

- A lot of problems require gargantuan calculus
- Approximation for heuristic solutions
Variational algorithms

Quantum evolution

- Encode the problem in the energy landscape
- Find a trial lowest energy point
- Exclude points from landscape
- Iterate until you find the global minimum
Variational algorithms
A NISQ optimization routine

A working application for the current NISQ systems

- Classical optimizer to leverage on quantum properties
- Loop feedback
Variational algorithms
Some examples

- VQE
- VITE
- VQF

- Chemistry related, finds the ground state
- Another promising way to find ground states
- Breakdown factoring in a NISQ system
Quantum Approximate Optimization Algorithm

Formulating combinatorial problems

Our problems can be formulated as a cost function

Finding the ground state, is a minimization problem

We use this similarity to resolve QUBO and Ising problems
Quantum Approximate Optimization Algorithm
Solving combinatorial problems

- Introduced in 2014
- QUBO encoding
Quantum Approximate Optimization Algorithm

QAOA Circuit
Quantum Approximate Optimization Algorithm

QAOA in practice

- Each vertex a qubit
- Each qubit in a partition
- Value of an edge based on his neighbours
Combinatorial problems
Solving combinatorial problems

- Simulated Annealing
- Simulated Quantum Annealing
- Quantum Annealing

![Graph showing solving combinatorial problems](image-url)
QAOA and Max-Cut, simulation through a QLM

Practical example on the use of our appliance
PyAQASM
A Python library to simplify

High level interface to design Quantum Circuits

Every bit of “tech” you need

Advanced usage for granular programming
Creating an EPR Pair

The circuit

- Represents a maximal entangled couple of qubits
- The output is always either $|00\rangle$ or $|11\rangle$
Creating an EPR Pair

The basics

- Program()
- .qalloc()
- .calloc()

```python
from qat.lang.AQASM import *
prog = Program()
```
Creating an EPR Pair

The basics

```
qub = prog.qalloc(2)
```
Creating an EPR Pair
The basics

```python
cb = prog.calloc(2)
```
Creating an EPR Pair

Populating the circuit

Gate calling

```python
prog.apply(H, qub[0])

prog.apply(CNOT, qub[0], qub[1])
#CNOT(qub[0], qub[1])
```
Creating an EPR Pair

A peek to our circuit

- `.to_circ()`
- `display / qat`
- `.to_job()`
- `.submit()`

```python
from qat.qpus import get_default_qpu
qpu = get_default_qpu()
result = qpu.submit(job)
```
Creating an EPR Pair

Showing the results

- Result
- Nbshots

```python
result = qpu.submit(job)

for sample in result:
    print("State %s amplitude %s" % (sample.state, sample.amplitude))
```
Creating an EPR Pair

Showing the results

```
job = circ.to_job(nbshots=10)
```
Circuits creation

The plugins

Alter the flow of our jobs, both in compiling and post process

Simple and easy to use

You can even write your owns
Circuits creation

Building stacks

Put different plugins one after another

```python
my_stack = plugin1 | plugin2 | .. | my_qpu
```
Circuits creation

The observables

- Used to determine the property of quantum states
- Automatize the sampling
- Can be manipulated

```python
from qat.core import Observable, Term

obs1 = Observable(2, pauli_terms=[Term(1., "ZZ", [0, 1])])
obs2 = Observable(2, pauli_terms=[Term(1., "X", [0])])

print(obs1 + obs2)
```
Circuit creation
Parameterized circuits

- An hybrid Machine Learning circuit
- Unfixed gates
- Evolve the gates through computation

```python
theta = prog.new_var(float, "\theta")
prog.apply(RY(theta), qubits_reg[0])
```
QAOA on myQLM

The big picture

- CombinatorialProblem Class
- From problem to variational Ansätze
- Observable synthesis
- Circuit synthesis
QAOA on myQLM
CombinatorialProblem Class

- Problem and Variables
- Clauses
- Min and Max

```python
my_problem = CombinatorialProblem()
v0 = my_problem.new_var()
v_array = my_problem.new_vars(4)
```
QAOA on myQLM

CombinatorialProblem Class

- Problem and Variables
- Clauses
- Min and Max

```python
print(v0 | v1)
print(~(v0 ^ v_array[3] | v1))
my_problem.add_clause(v0 | v1, weight=2.)
```
QAOA on myQLM
CombinatorialProblem Class

- Problem and Variables
- Clauses
- Min and Max

```python
my_maximization_problem = CombinatorialProblem(maximization=True)
```
QAOA on myQLM

CombinatorialProblem Class

```python
from qat.opt import CombinatorialProblem

my_problem = CombinatorialProblem()

v0 = my_problem.new_var()
v1 = my_problem.new_var()

v_array = my_problem.new_vars(4)

print(v0 | v1)
print(v_array[0] & v_array[2])
print(v0 ^ v_array[0])
print(~v0)
print(~(v0 ^ v_array[3]) | v1))

my_problem.add_clause(v0 ^ v1)

my_problem.add_clause(v0 | v1, weight=2.)

for clause, weight in my_problem.clauses:
    print(clause, weight)
```

```
V(0) V(1)
V(2), V(3), V(4), V(5)
V(0) | V(1)
V(2) & V(4)
V(0) ^ V(2)
~ V(0)
~ ((V(0) ^ V(5)) | V(1))
V(0) ^ V(1) 1.0
V(0) | V(1) 2.0
```
QAOA on myQLM
From problem to variational Ansätze

- Ansätze construction

```python
depth = 3
ansatz = my_problem.qaoa_ansatz(depth).circuit
ansatz.display()

ansatz_gamma_0_pi = ansatz.bind_variables({'\gamma_0': np.pi})
```
QAOA on myQLM
From problem to variational Ansätze

```
1  depth = 3
2  ansatz = my_problem.qaoa_ansatz(depth).circuit
3  ansatz.display()

4
5  print("Variables:", ansatz.get_variables())
6
7  import numpy as np
8  ansatz_gamma_0_pi = ansatz.bind_variables({"\gamma_0": np.pi})
```
QAOA on myQLM
Observable synthesis

- Encoding in smaller Hamiltonians

\[
\begin{align*}
\text{exp} & := \text{exp} \lor \text{exp}|\text{exp} \land \text{exp}|\text{exp} \oplus \text{exp}|\neg\text{exp}|V \\
H(e_1 \lor e_2) & = H(e_1) + H(e_2) - H(e_1)H(e_2) \\
H(e_1 \land e_2) & = H(e_1) \ast H(e_2) \\
H(e_1 \oplus e_2) & = H(e_1) + H(e_2) - 2H(e_1)H(e_2) \\
H(\neg e) & = 1 - H(e) \\
H(V(i)) & = \frac{1 - \sigma_i^z}{2}
\end{align*}
\]
QAOA on myQLM
Observable synthesis

- Encoding in smaller Hamiltonians

```python
for i in range(4):
    my_problem.add_clause(variables[i]**variables[i+1])
print("Minimization:", my_problem.get.observable())
```

Minimization:

```
2.0 * I^5 +
-0.5 * (ZZ[0, 1]) +
-0.5 * (ZZ[1, 2]) +
-0.5 * (ZZ[2, 3]) +
-0.5 * (ZZ[3, 4])
```
QAOA on myQLM
Observable synthesis

- Encoding in smaller Hamiltonians

```python
my_problem = CombinatorialProblem(maximization=True)
variables = my_problem.new_vars(5)
for i in range(4):
    my_problem.add_clause(variables[i]^variables[i+1])
print("Maximization:\n", my_problem.get.observable())
```

Maximization:
-2.0 * I^5 +
0.5 * (ZZ|[0, 1]) +
0.5 * (ZZ|[1, 2]) +
0.5 * (ZZ|[2, 3]) +
0.5 * (ZZ|[3, 4])
QAOA on myQLM

Circuit synthesis

- Circuit synthesis algorithms

```python
import myQLM

# Circuit synthesis

# Default strategy
 circuit1 = my_problem.qaoa_ansatz(1, strategy="default").circuit

# Coloring strategy
 circuit2 = my_problem.qaoa_ansatz(1, strategy="coloring").circuit

# Gray synthesis strategy
 circuit3 = my_problem.qaoa_ansatz(1, strategy="gray_synth").circuit
```
A graph partitioning problem:

- Cut our graph \((G)\) in 2 subsets \((S\) and \(V)\)
- Each subset must have a number > 0 of vertexes
- Maximize the number of edges crossed by the cut
QAOA on myQLM

What exactly is a max-cut?
QAOA on myQLM
Running of a MaxCut

- Integrated wrapper
- Variational plugin to OPT

```python
import networkx as nx

graph = nx.generators.random_graphs.erdos_renyi_graph(10, 0.5)
nx.draw(graph)
from qat.vsolve.qaoa import MaxCut

problem = MaxCut(graph)
print(problem)
```
QAOA on myQLM
Running of a MaxCut

- Integrated wrapper
- Variational plugin to OPT
QAOA on myQLM
Running of a MaxCut

- Integrated wrapper
- Variational plugin to OPT

```python
from qat.qpus import get_default_qpu
from qat.plugins import ScipyMinimizePlugin
qpu = get_default_qpu()
stack = ScipyMinimizePlugin(method="COBYLA",
                           tol=1e-5,
                           options={"maxiter": 200}) | qpu

job = problem.qaoa_ansatz(3)
result = stack.submit(job)
print("Final energy:", result.value)
```
QAOA on myQLM
Running of a MaxCut

- Integrated wrapper
- Variational plugin to OPT

Most probable states are:
And as bitstrings:
Most probable cut: \{0, 1, 4, 5, 6, 8\} \{6, 2, 3, 7\}
Click on the picture or visit myqlm.github.io/notebooks
Thank you!

For any information, please contact:

Agostino Maria Cassese
agostino-maria.cassese@atos.net