Solving Graph Coloring with 256 qubit neutral atoms platform

A Hybrid Quantum-Classical approach

MATTEO FLOCCO – FONDAZIONE LINKS / POLITECNICO DI TORINO

HPCQC23 - CASALECCHIO DI RENO

Background: PCI assignment problem

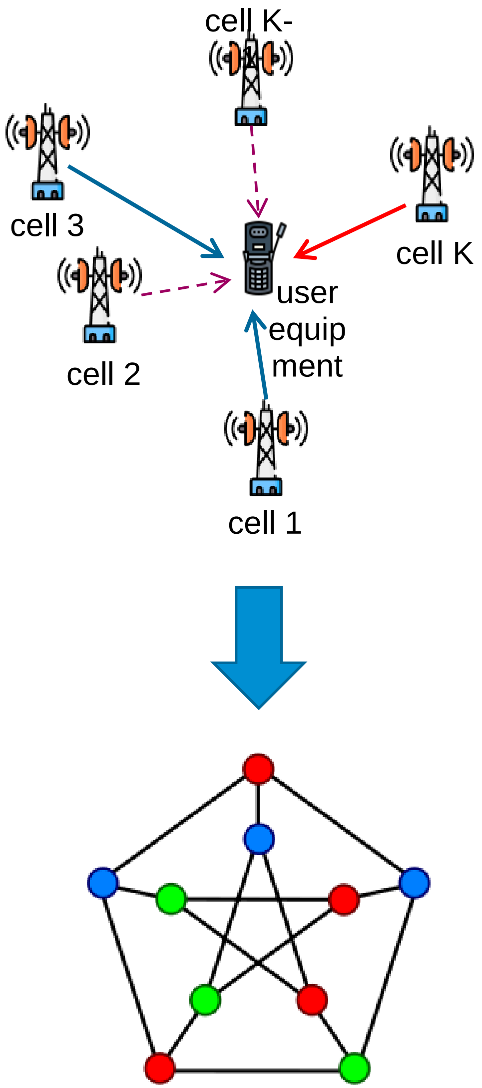
Optimization problems

Quantum Computing (QC) promises disruptive impact on hard computational problems and related applications.

<u>Physical Cell Identifiers (PCIs) problem</u>:
assign PCIs "*label*" to the nodes of a cellular communication network

- **Goal:** maximise data throughput by avoiding conflicts and confusion between nodes and end terminal (cell phone)

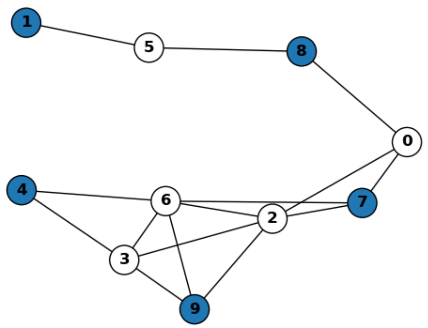
PCI assignment translate to a graph format and to one or more instances of **Graph Coloring (GC) problem**.



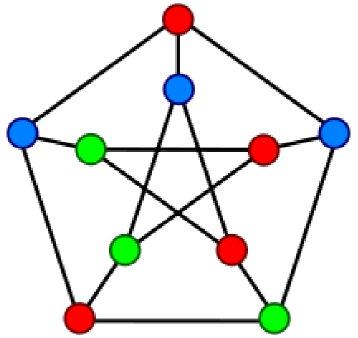
MIS & GC problems

Definitions

G(V,E) graph



χ(G) = 3



Maximal vs. Maximum Independent Set (IS):

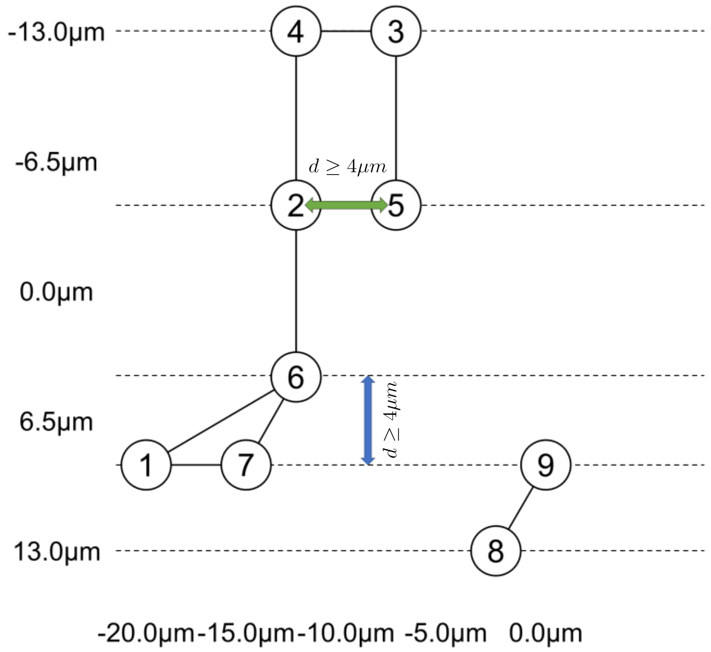
- Maximal (m)IS: cannot add a vertex to the set without violating the independence constraint;
- Maximum (M)IS: the globally largest maximal independent set that can be identified on a G.

Graph Coloring problem:

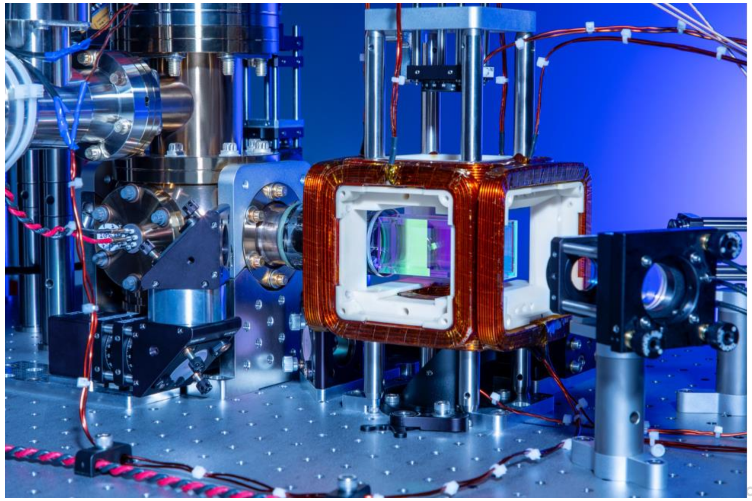
- Feasible coloring of G: color/vertex assignment s.t. vertexes that share an edge have different colors;
- Graph coloring problem: minimize colors required for coloring G (chromatic number **χ(G)**).

Solving MIS on Neutral Atoms

Building a register



- Current set-up of the QuEra's Aquila platform allows for:
  - **Analog operations**
  - **Global Rydberg pulses**
  - **256 qubits on 2D arrays**
- Interactions between qubits of a register can be represented as a **Unit-Disk graph (UD)**.
  - Graph must be **embedded** into a Unit-Disk graph.



Source:AWS white paper

Solving MIS on Neutral Atoms

Solving algorithm (Quantum Kernel)

- MIS problem can be mapped to the **Ising and machine Hamiltonians** and solved with algorithms such as Quantum Adiabatic Algorithm (**QAA**):
  - System in ground state at t=0;
  - "Slow" evolution to final Hamiltonian (mapped to the problem cost function).

$$\min_{\delta_i} \quad -\sum_{i \in V} \delta_i + P \sum_{(i,j)\in E} \delta_i \delta_j$$

$$\text{s.t.} \quad \delta_i \in \{0,1\}$$

$$H_{Ising} = -\sum_{ij} J_{ij}\sigma_i\sigma_j + B\sum_i \sigma_i$$

$$\sigma|\uparrow\rangle = 1|\uparrow\rangle \qquad \sigma|\downarrow\rangle = -1|\downarrow\rangle$$

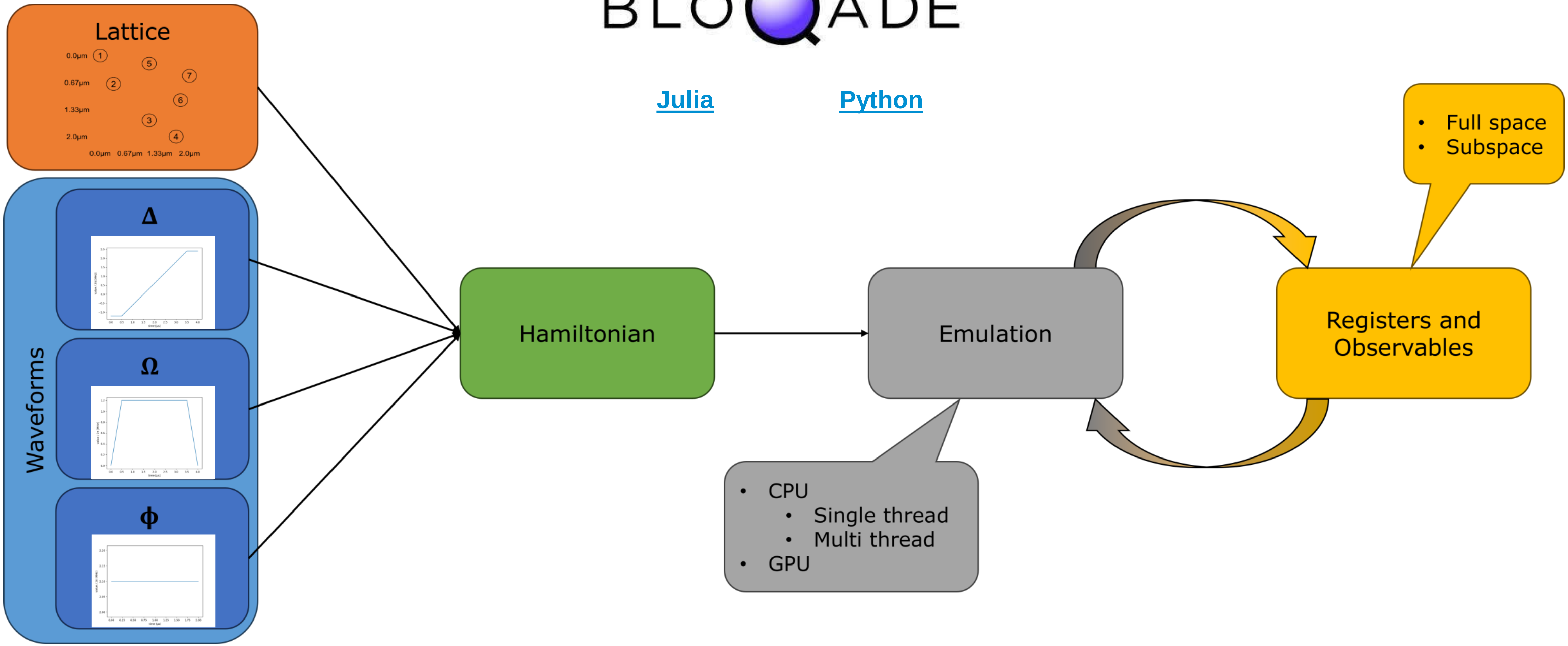
$$\mathcal{H} = \frac{\Omega}{2}\sum_i (|g_i\rangle\langle r_i| + |r_i\rangle\langle g_i|) - \Delta\sum_i n_i + \sum_{i<j} V_{ij} n_i n_j$$

Bloqade

QuEra's Julia (and Python) package for neutral atoms quantum computation

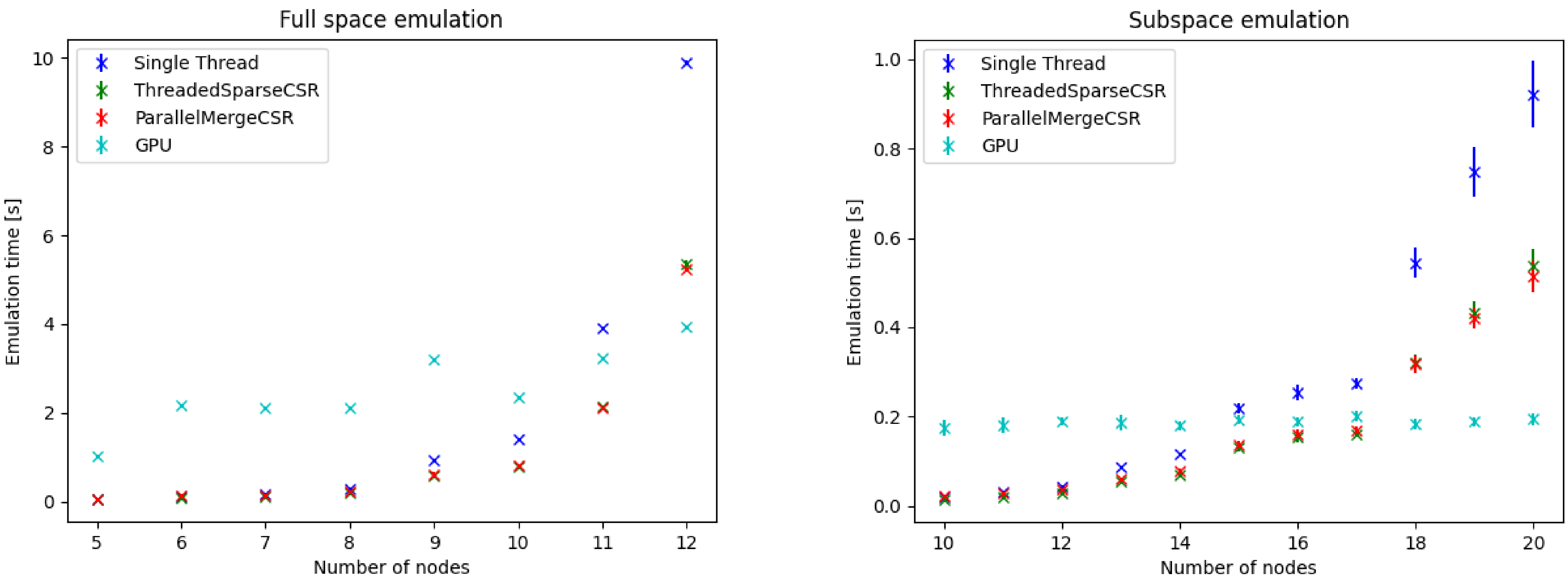
BLOQADE

[Julia]   [Python]



Bloqade performances

Bloqade emulation on 28-cores - one NVIDIA A30 GPU server



The tests have been performed using the BenchmarkTool Julia library, using 10 minutes or 10k simulations (whichever happens first) as stopping criteria and one graph per data point

GC = BB on mIS

BBQ-mIS

Notation:

- Vertex = vertex of the graph G
- Node = node of the Branch&Bound (BB) process
- mIS = maximal Independent Set

Theorem: Every graph G has an optimal coloring in which (at least) one of the colors is an mIS.

Greedy-it-MIS issue: consider just one MIS solution at each iteration.

→

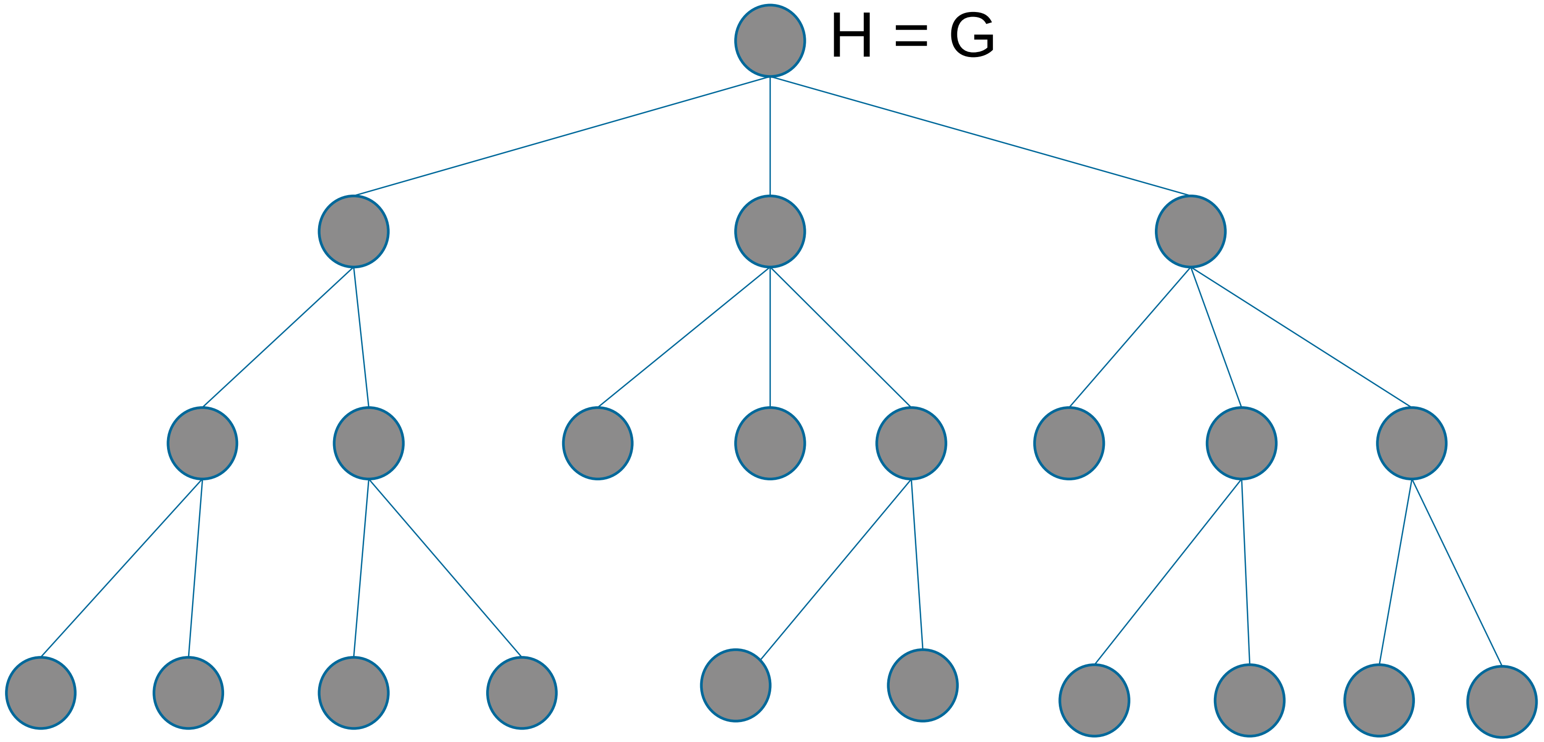
BBQ-mIS improvement: outer optimization loop that considers multiple mIS solutions at each iteration.

BBQ-mIS algorithm:

- Start with the whole graph G in the root node of the BB tree;
- For each BB node find a set of mIS solutions and generate one branch for each solution;
- Each BB node is associated with a subgraph of G obtained by removing all the vertexes of one mIS;
- Select the solution with the lowest number of colors used.

GC = BB on mIS

BBQ-mIS



Each BB node has:

- H: induced subgraph of the original graph G
- Feasible coloring: mISs solutions + worst-case coloring[1]
- Obj. fun: colors used[2] in the feasible coloring
- LB: depth of the node + LB on χ(H)

[1] *worst-case coloring*: one color for each of the remaining vertexes in H
[2] *colors used*: depth of the node, i.e. number of mISs sets removed + worst-case coloring

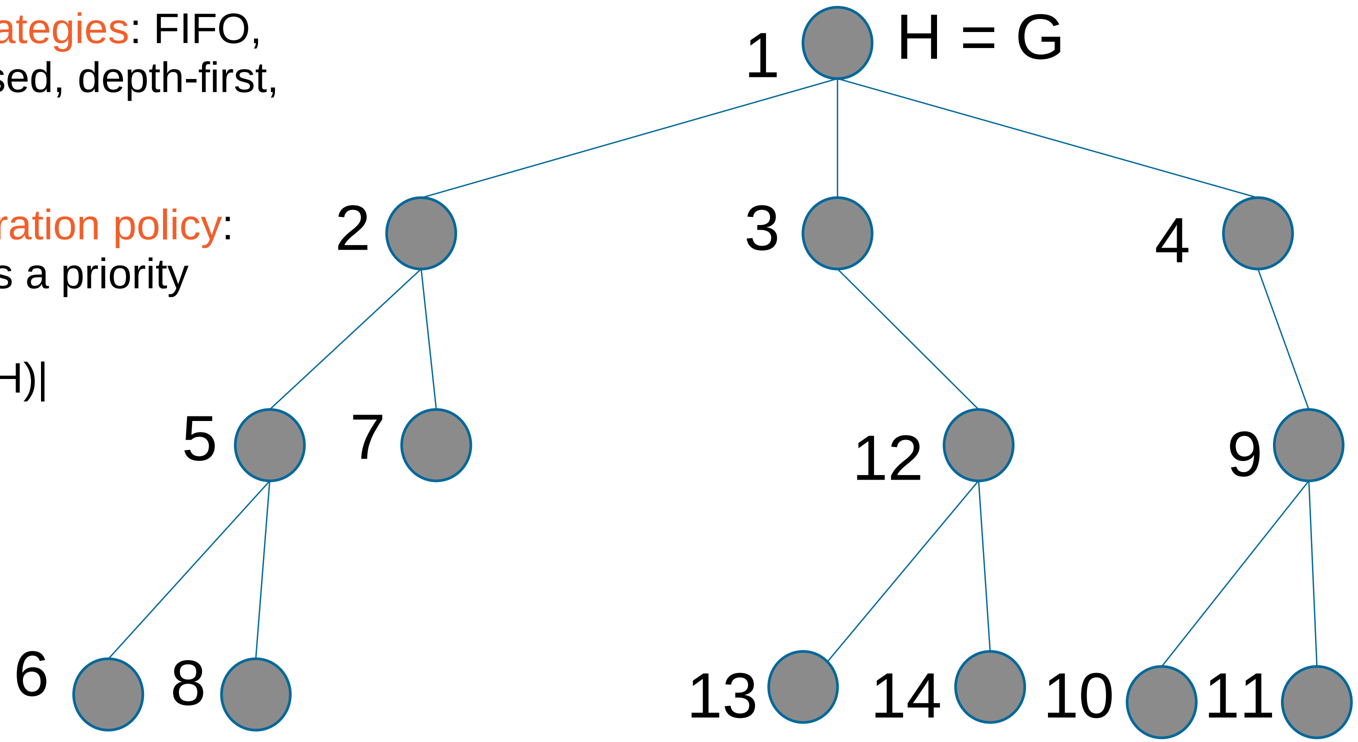
GC = BB on mIS

BBQ-mIS

BB tree exploration:

- Traditional strategies: FIFO, LIFO, gap-based, depth-first, etc...
- Custom exploration policy: each node has a priority defined as

  - UB ⨉ |edges(H)|



Pruning criteria:

- Non improving solution: LB ≥ best current objective
- Unfeasible solution:
  - Non-independent set solution
  - Non-maximal set solution
- Redundant solution: the same induced subgraph H has yet been explored previously

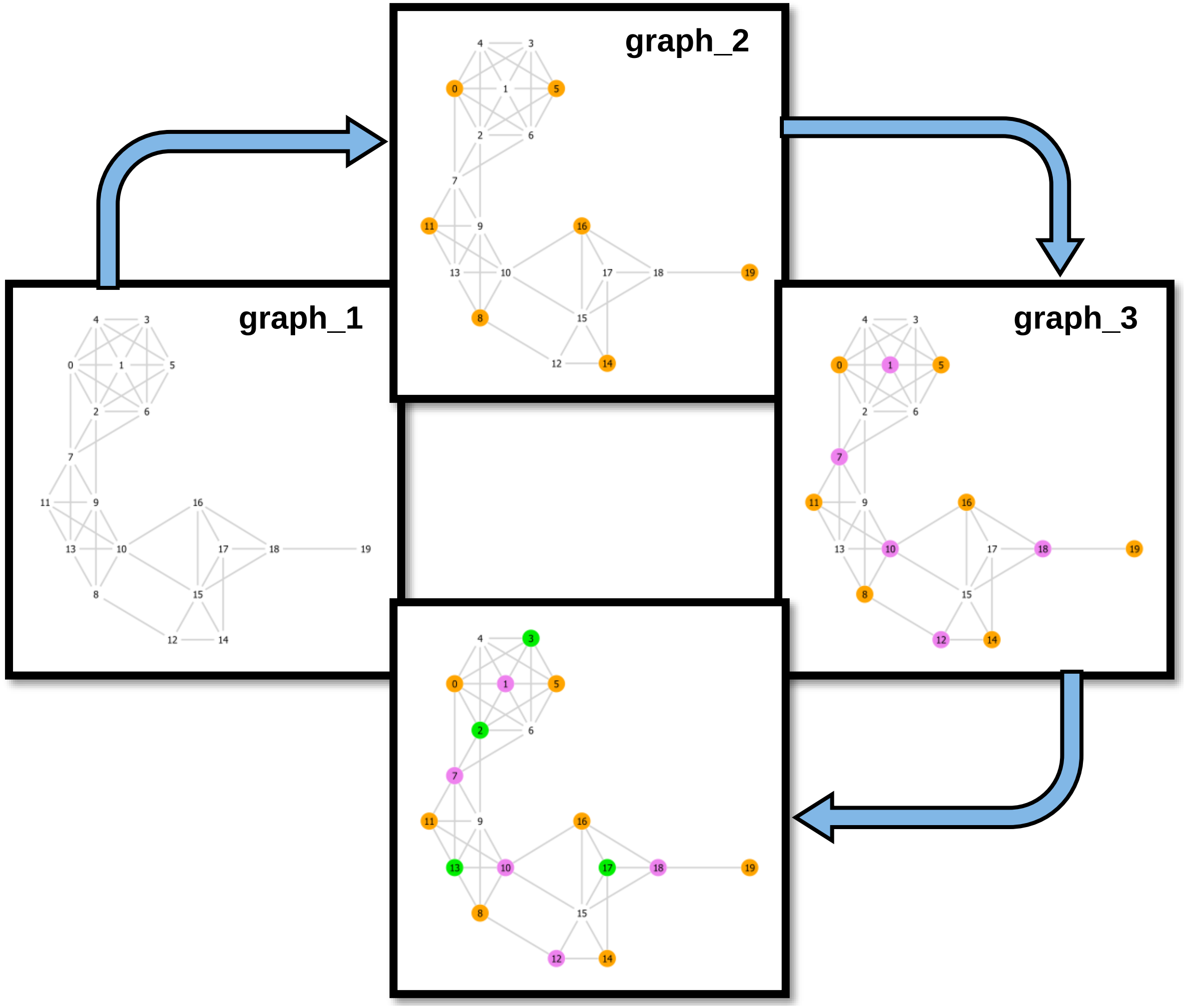
The tree exploration policy is affected by the number of MPI processes that are used to parallelise the BB procedure.

Solving the GC problem

BBQ-mIS on a 20-vertex graph

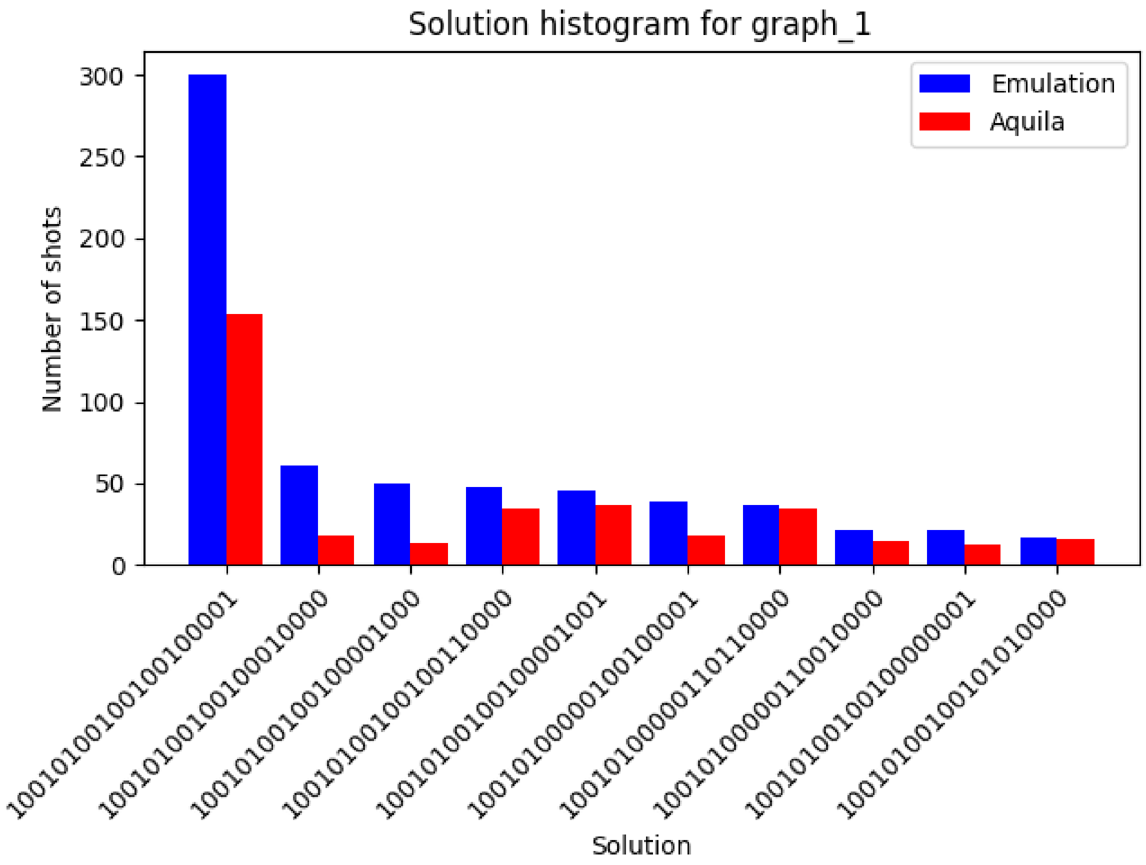
- We demonstrate the BBQ-mIS algorithm's ability to **solve unit disk-based GC problems**, exemplified by antenna positions in Turin, by embedding these graphs for Aquila.

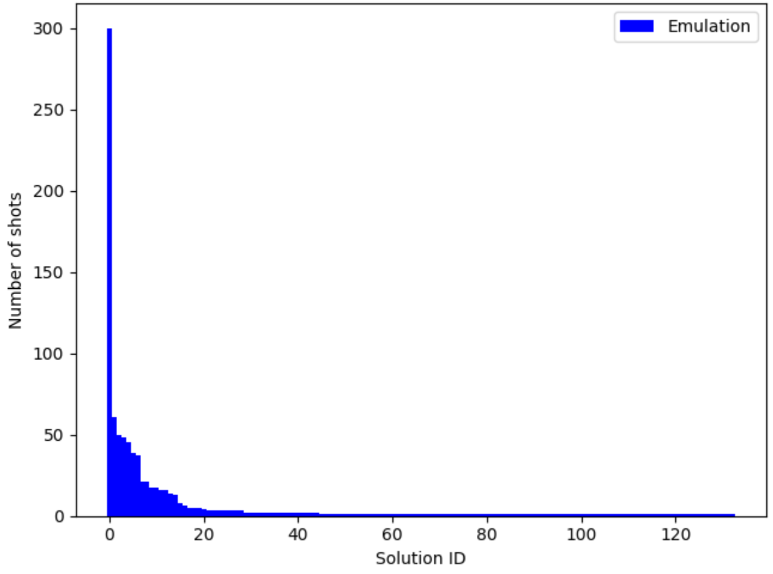
- To assess Aquila's results against noiseless emulations, we executed three tasks on **Aquila** (via AWS Braket), mirroring the emulations, to solve a GC problem on a twenty-node graph.

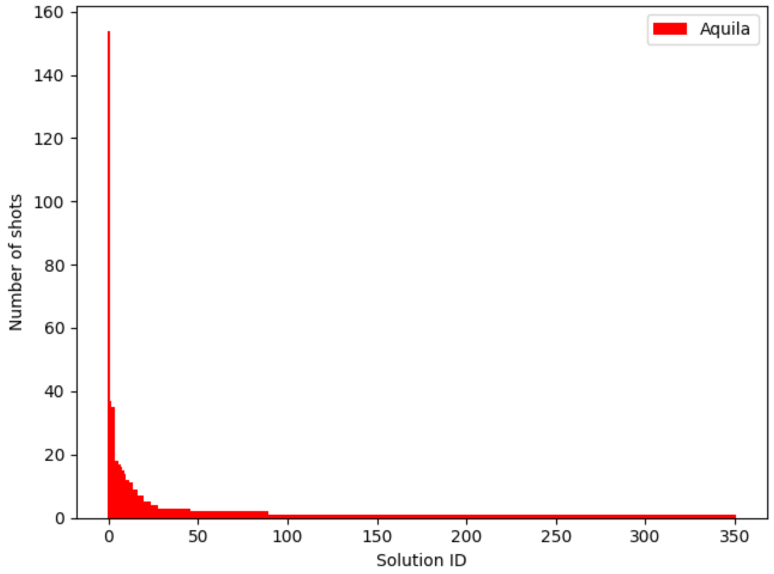


Runs on Aquila

Some real data on a 20-vertex graph (1000 shots)







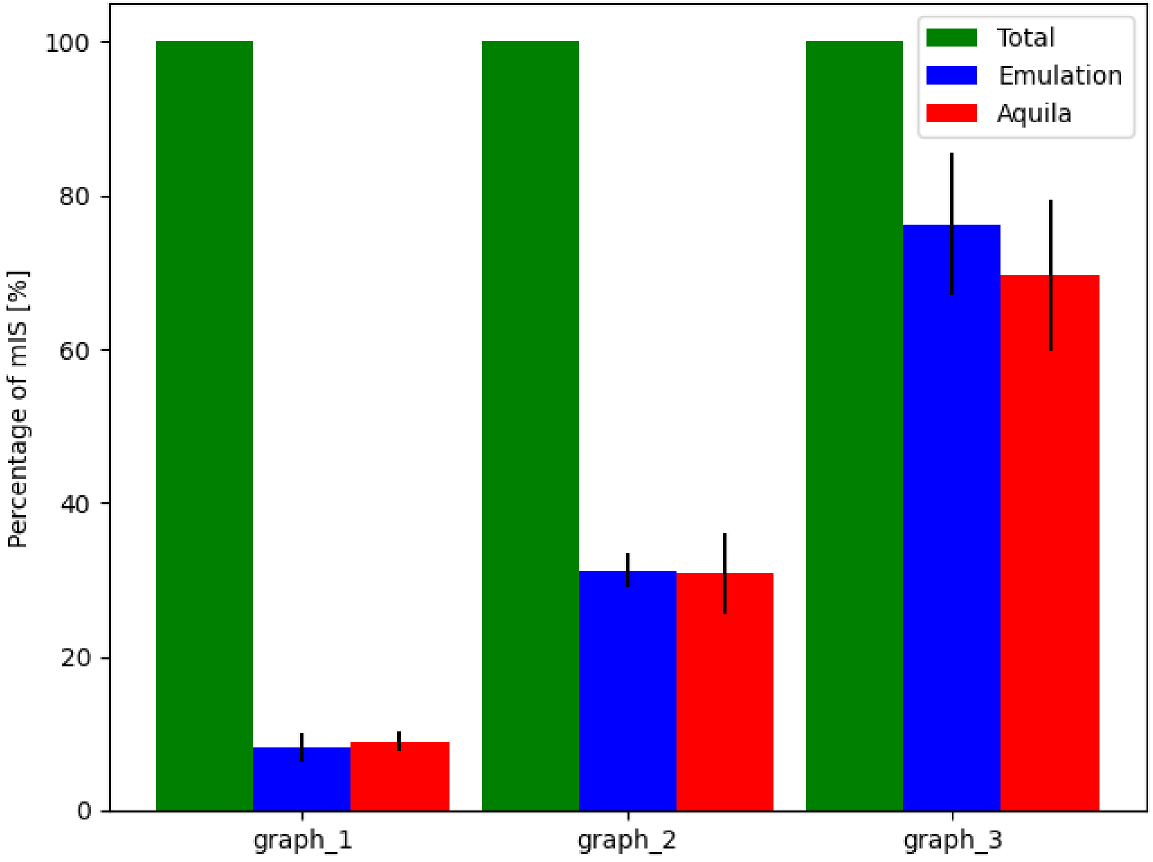
Runs on Aquila

Some real data on 20 vertex graph

- Percentage of mIS found is important → **better exploration** of solution space.

- Quantity of mIS found using quantum hardware is in line with noiseless emulation performed on classical devices → **noise-resistant algorithm**

Number of mIS	graph_1	graph_2	graph_3
Total	124	31	8
Emulated	10.10 ± 2.33	9.70 ± 0.67	6.10 ± 0.74
Aquila	11.17 ± 1.60	9.57 ± 1.62	5.57 ± 0.79



Going forward

Next steps

- **Hyperparameter/pulse characterization**: will improve the QAA efficiency and enhance the likelihood of finding a minimum GC.

- **Closed Loop Hybrid Execution**: run the BBQ-mIS into full hybrid mode (closed loop, i.e., mIS on Aquila submitted by B&B on classical platform), leveraging hybrid programming models/tools (such as AWS Braket's **Hybrid Jobs**).

- **Benchmark**: evaluate BBQ-mIS performance using larger set of graphs and comparing it to state-of-the-art classical algorithms for graph coloring (related to closed loop execution).

- **Embedding** methodology: key to represent any graph as a unit disk graph respecting the HW constraints.

Many thanks to:

- QuEra for the support and useful discussions
- AWS for the research credits

Questions?

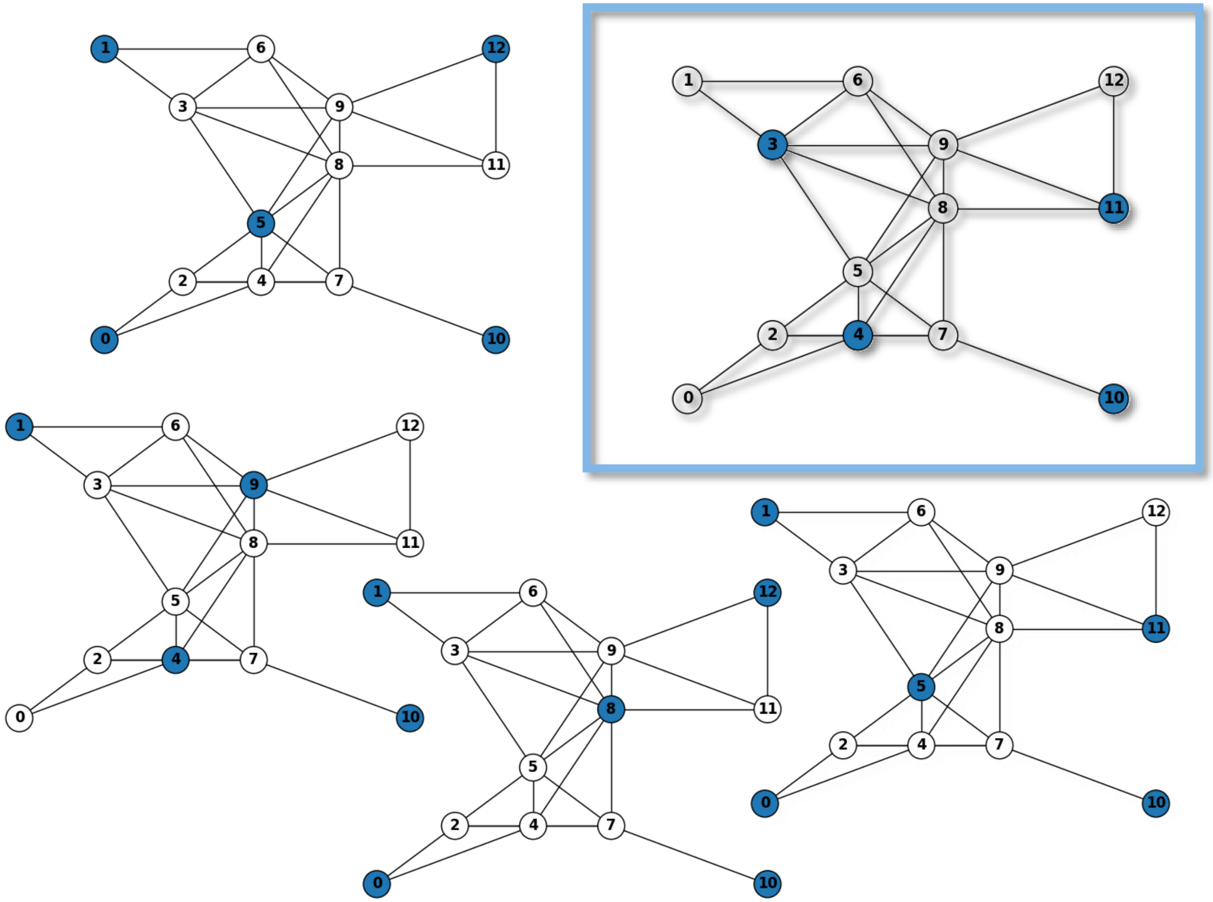
MATTEO FLOCCO – FONDAZIONE LINKS / POLITECNICO DI TORINO

HPCQC23 - CASALECCHIO DI RENO

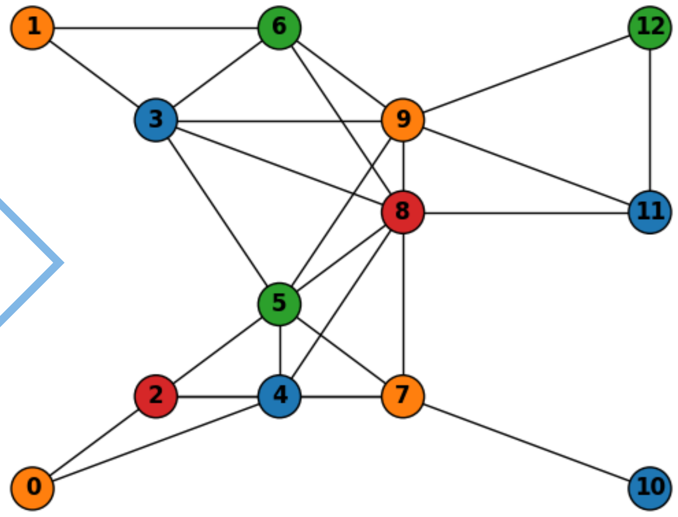
GC = BB on mIS

BBQ-mIS

mIS solutions at the BB root node



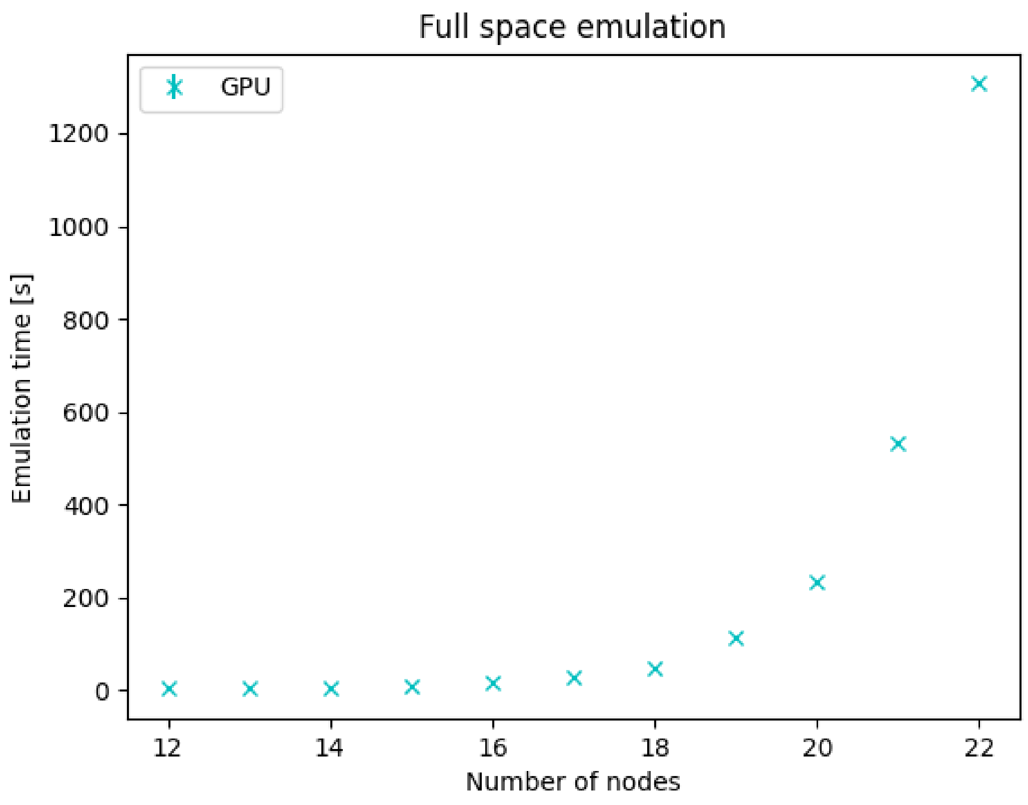
BBQ-mIS solution

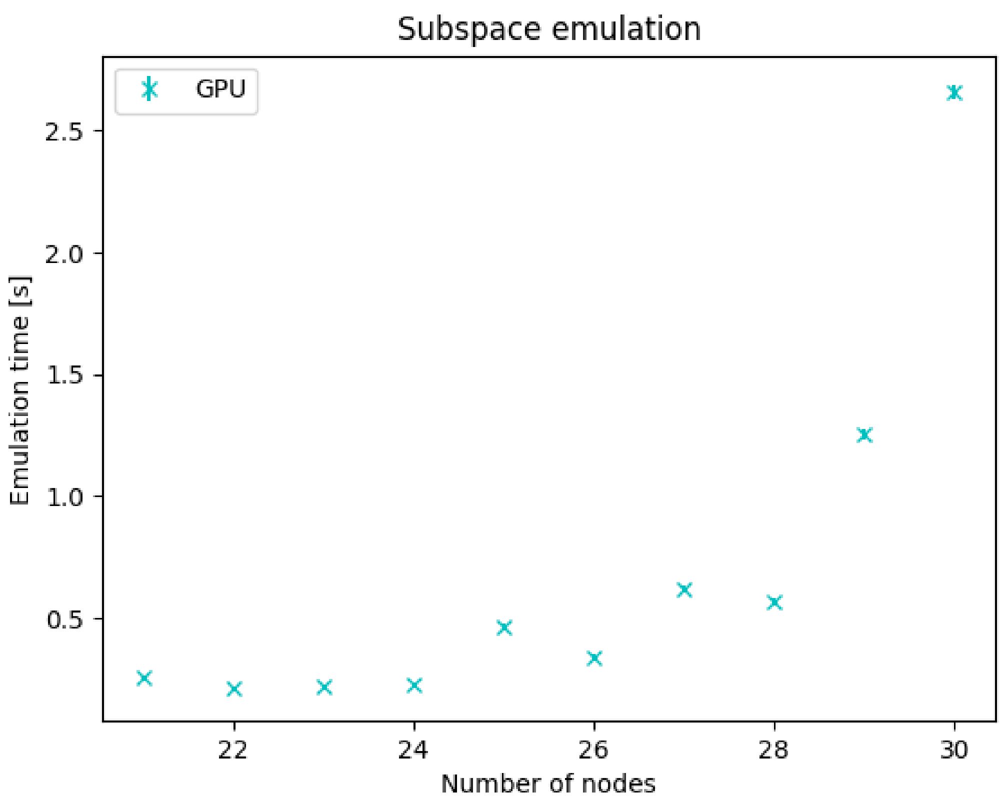


4 colors: 1 less than the Greedy-it-MIS solution

Bloqade performances

Bloqade emulation on 28-cores - one NVIDIA A30 GPU server





BBQ-mIS

Comparison of nodes exploration between BBQ-mIS and BBQ-mIS Classical