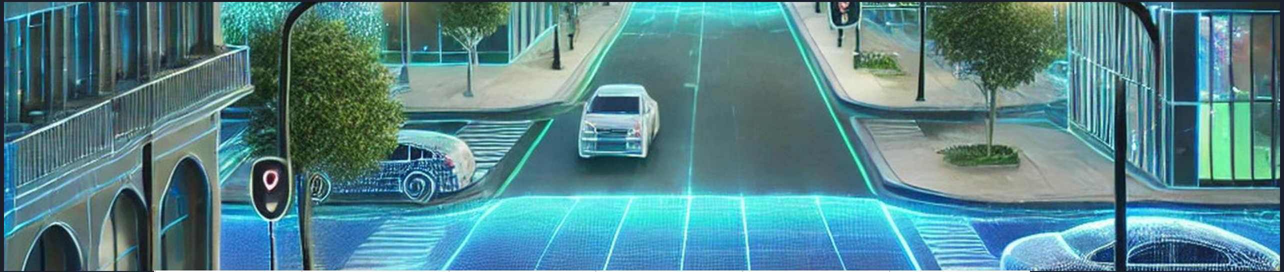
Quark

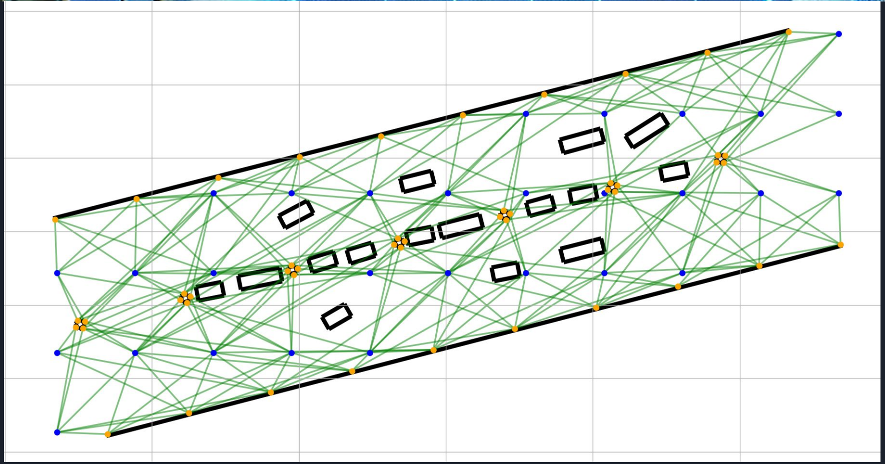
Sensor Positioning
Aqarios | PushQuantum

Ashish, Vishnuthirtha , Adrian Raiser, Marvin Raiser

The Problem

Problem Description

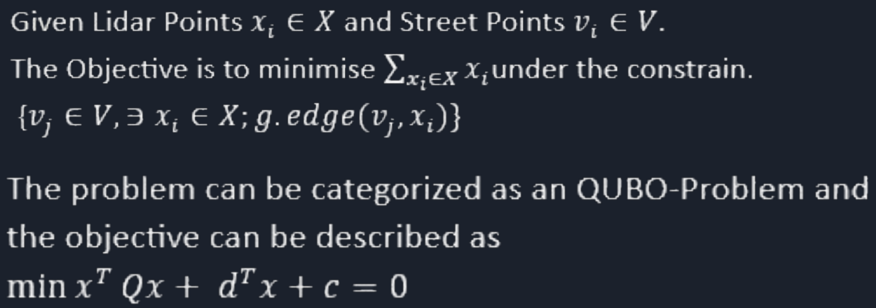




Problem Description

Given a set of Lidar Placement Points X
and a set of Points of Interest (Street Points) V,

The objective is to minimize the amount of installed Lidar sensor, while every Point V has to be seen by at least one Lidar:



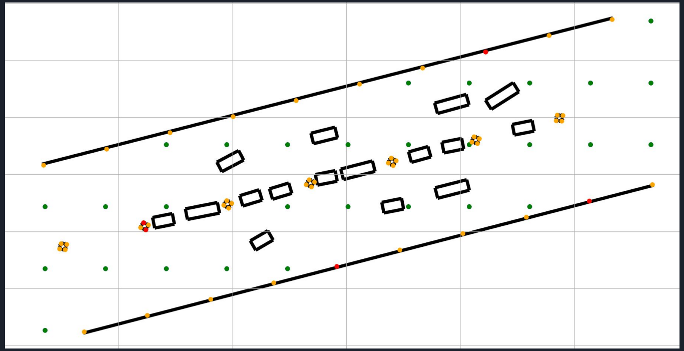
Given Lidar Points $x_i \in X$ and Street Points $v_i \in V$.

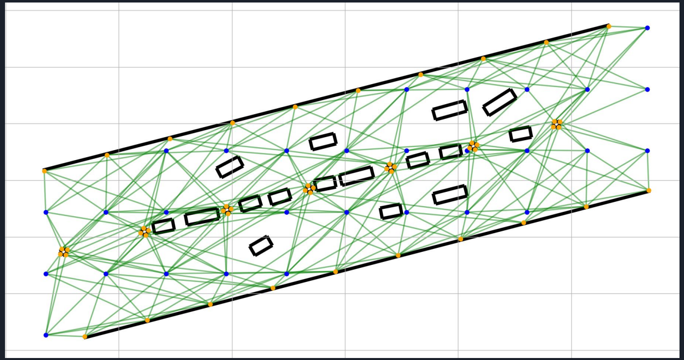
The Objective is to minimise $\sum_{x_i \in X} x_i$under the constrain.

$\{v_j \in V, \exists\, x_i \in X; g.edge(v_j, x_i)\}$

The problem can be categorized as an QUBO-Problem and the objective can be described as

$$\min x^T\, Qx + \; d^T x + c = 0$$

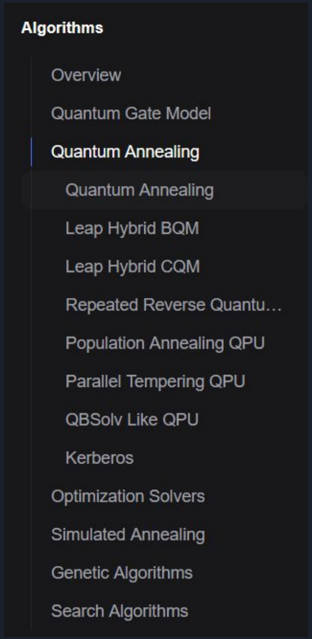




Assembling the Q matrix

- Target: min(x^T * Q * x + d^T * x + c)
- Problem is described as inequality => Introduction of “slack values”
  - Every slack value increases complexity
- Collection of Cost Functions: f(x,s) = d_j*(∑x_i - s_j - 1)^2 for every Street Point j
  - Iterate over every Street Point v_j => “Submatrices” (potentially parallelizable/GPU)
- Assemble Q, d and c by “adding” the Submatrices
- Due to the problem being binary, d can be integrated into Q
  - Added to main diagonal
- c represents an energy “base level”, moving the Hyperplane “up and down”
  - => can be ignored
- Caveats: s_j is a positive integer depending on the neighbours
  - state vector has to be adjusted accordingly

Aqarios Luna



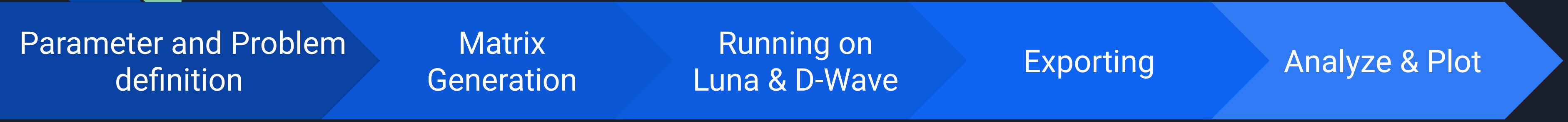
- “solving optimization problems with our advanced quantum, hybrid, and classical algorithms tailored to your unique needs.”
- Transpiler for python code to run on (simulated) quantum hardware
- connectable with D-Wave, IBM and others

DWave Quantum Computer

- Runs Problems on a Quantum Annealer
- Luna API as middleware between developer and DWave
- Free API provides 1min computation time for free
- 5 Tokens used/burned
- only used when Luna decides to use the qpu

Comparison between Algorithms

Benchmarking Pipeline



Parameter and Problem definition

- Solver
  - QA, QAGA+, SAGA+
  - (p_size)
  - (mut_rate)
  - (rec_rate)
- Version (row)
- num_cols
- P1
- P2
- P3

Matrix Generation

- Provide custom matrix
- generate with given scenario settings

Running on Luna & D-Wave

- Running the APIs
- Token Management
- Waiting for the results

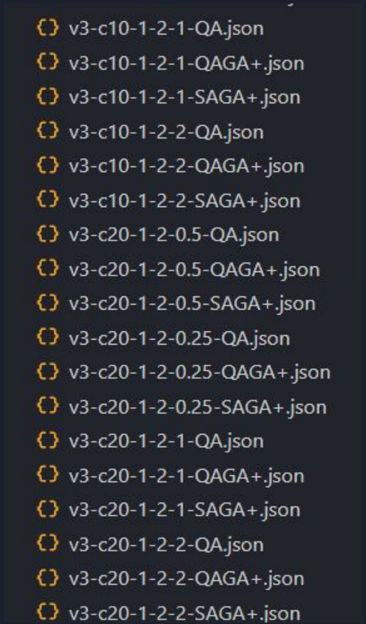
Exporting

- Recalculate objective value
- store every run in a json file

Analyze & Plot

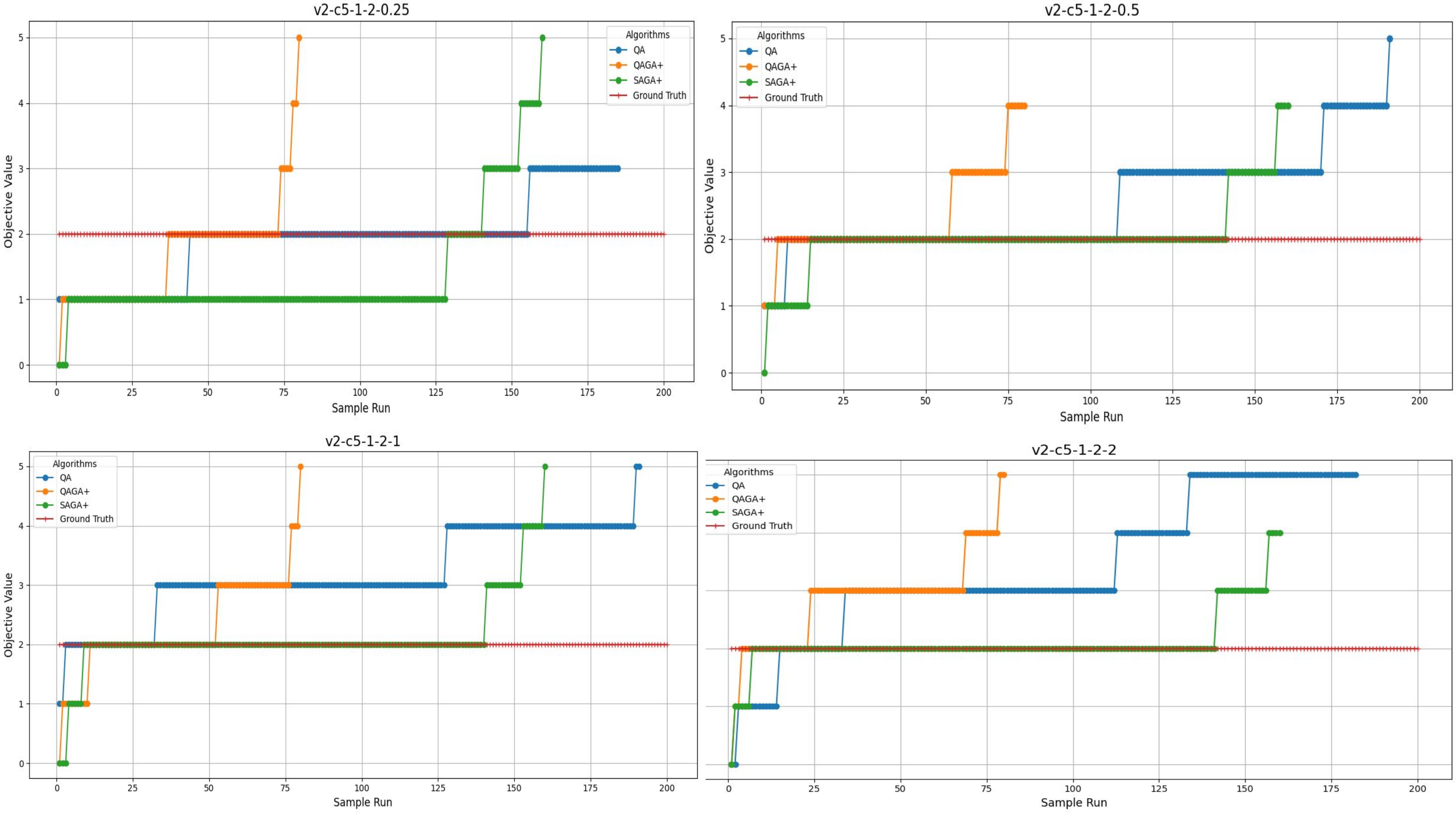
- Analyze the data by reading the json files
- plotting the results

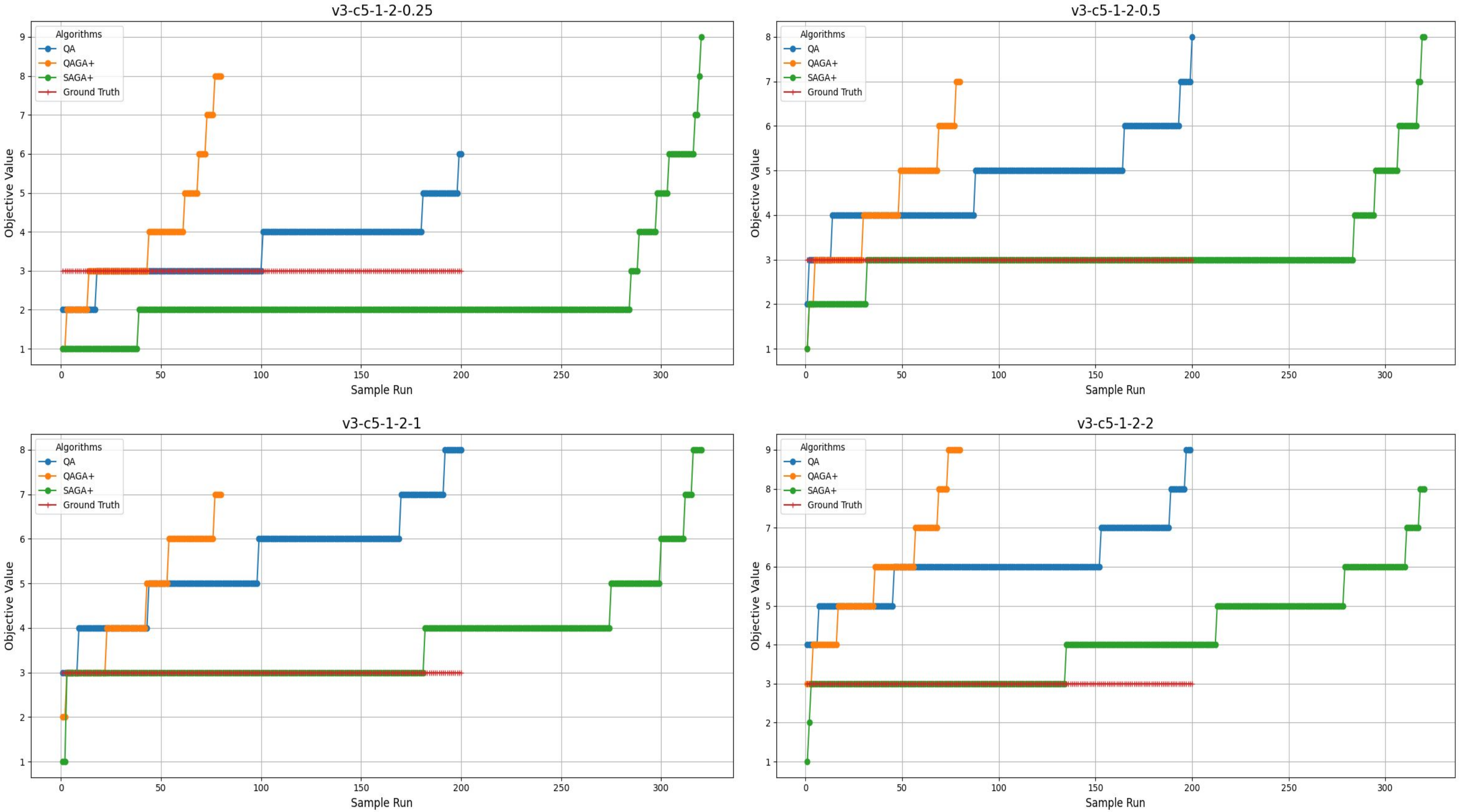
DWave Quantum Computer

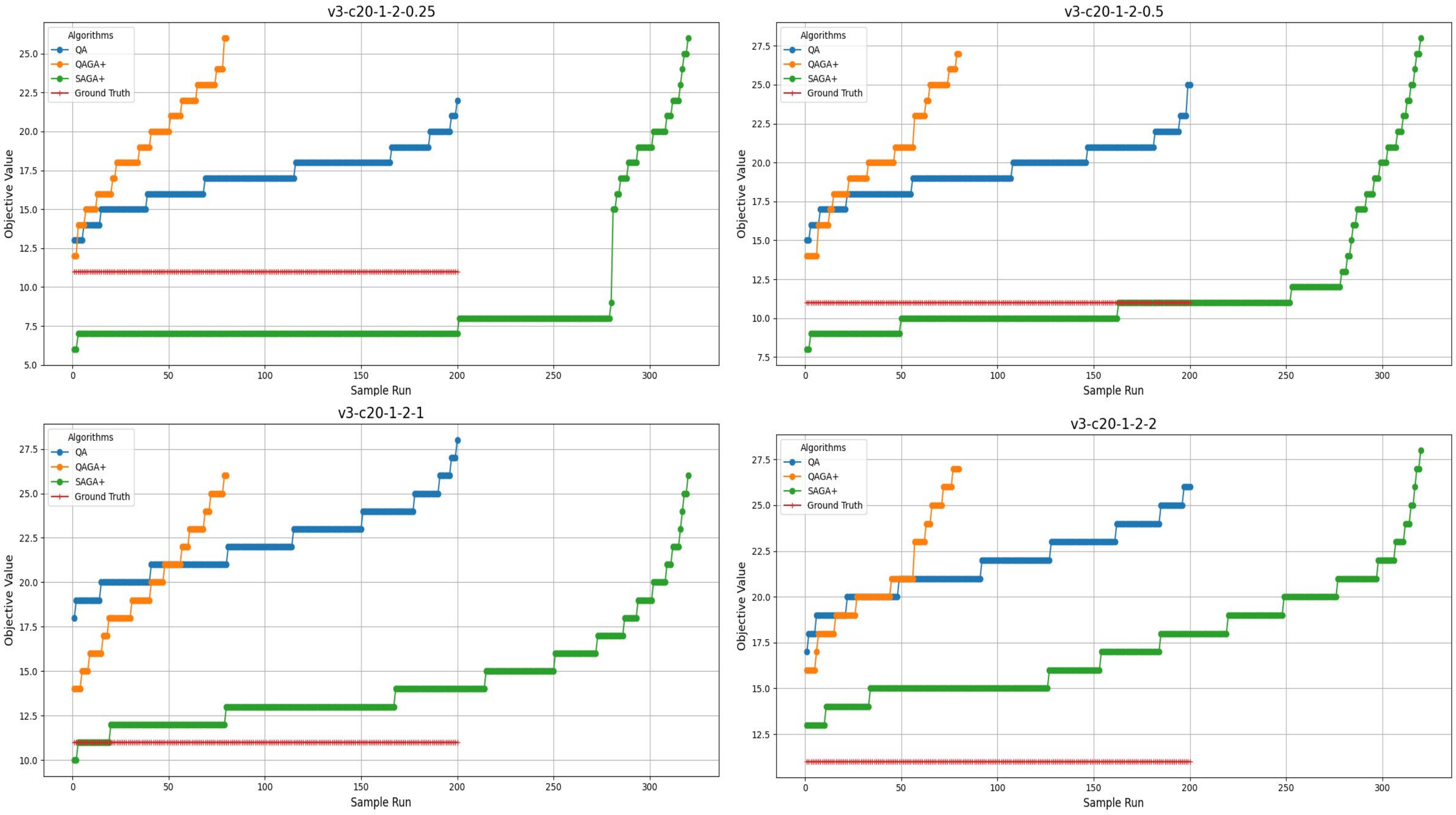


```
v3-c10-1-2-1-QA.json
v3-c10-1-2-1-QAGA+.json
v3-c10-1-2-1-SAGA+.json
v3-c10-1-2-2-QA.json
v3-c10-1-2-2-QAGA+.json
v3-c10-1-2-2-SAGA+.json
v3-c20-1-2-0.5-QA.json
v3-c20-1-2-0.5-QAGA+.json
v3-c20-1-2-0.5-SAGA+.json
v3-c20-1-2-0.25-QA.json
v3-c20-1-2-0.25-QAGA+.json
v3-c20-1-2-0.25-SAGA+.json
v3-c20-1-2-1-QA.json
v3-c20-1-2-1-QAGA+.json
v3-c20-1-2-1-SAGA+.json
v3-c20-1-2-2-QA.json
v3-c20-1-2-2-QAGA+.json
v3-c20-1-2-2-SAGA+.json
```

- 150 Benchmark Runs
- 5 D-Wave Tokens burned


v2-c5-1-2-0.25
Algorithms
QA
QAGA+
SAGA+
Ground Truth
Objective Value
0
1
2
3
4
5
Sample Run
0
25
50
75
100
125
150
175
200
v2-c5-1-2-0.5
Algorithms
QA
QAGA+
SAGA+
Ground Truth
Objective Value
0
1
2
3
4
5
Sample Run
0
25
50
75
100
125
150
175
200
v2-c5-1-2-1
Algorithms
QA
QAGA+
SAGA+
Ground Truth
Objective Value
0
1
2
3
4
5
Sample Run
0
25
50
75
100
125
150
175
200
v2-c5-1-2-2
Algorithms
QA
QAGA+
SAGA+
Ground Truth
Sample Run
0
25
50
75
100
125
150
175
200


v3-c5-1-2-0.25
v3-c5-1-2-0.5
v3-c5-1-2-1
v3-c5-1-2-2
Algorithms
QA
QAGA+
SAGA+
Ground Truth
Objective Value
Sample Run
0
50
100
150
200
250
300
1
2
3
4
5
6
7
8
9


v3-c20-1-2-0.25
v3-c20-1-2-0.5
v3-c20-1-2-1
v3-c20-1-2-2
Algorithms
QA
QAGA+
SAGA+
Ground Truth
Objective Value
Sample Run
0
50
100
150
200
250
300
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5

Thanks for taking a look at our Quark

Link to our Repo: https://github.com/Raisierer/PushQuantumSP