Energy Efficient Network Routing using Hybrid Quantum Algorithm

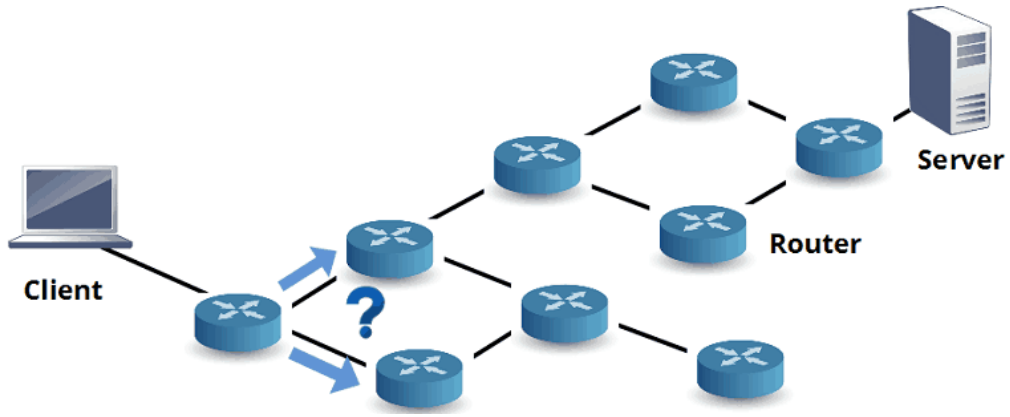
Jie Chen

Email : cencen.cj2021@protonmail.com

[arxiv: 2110.06321]

Table of Contents

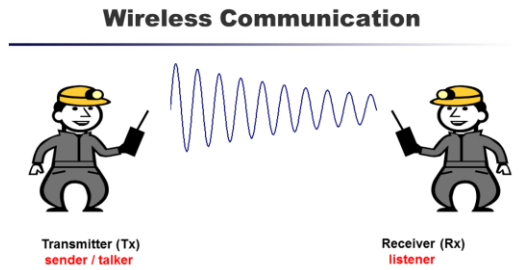
- Motivation
  - Energy Efficient Routing overall
  - Conventional Approaches
- Problem Formulation
- Results
- Future work



Energy Efficient Routing Overall

- Use Scenarios
  - Path loss, Doppler effect, etc attenuate and distort the signal
  - Devices are powered to process and hold the data

- Application Scenarios
  - Smart Home/City applications
  - Wireless Body Network (Health Sector)
  - Wireless Underwater Network
  - Embedded devices working in extreme environment



Conventional Approaches and Challenges

- Legacy High Performance Computers
  - Parallel/Distributed Computing – multiple cores
  - Software Algorithm Design – the search for the ground truth(s)
    - Bio-intelligence
      - Game Theory, Neural Network, Swarm Intelligence
    - Artificial Intelligence
      - Learning Theories
- Softwarisation raises the challenge to legacy computational power
  - Higher volume of data to process
    - Various hardware network topologies, ideal or abnormal
    - Unstoppable human innovations, in terms of software applications
  - Lower barriers to enter the playground – pushing a pursuit of extremity

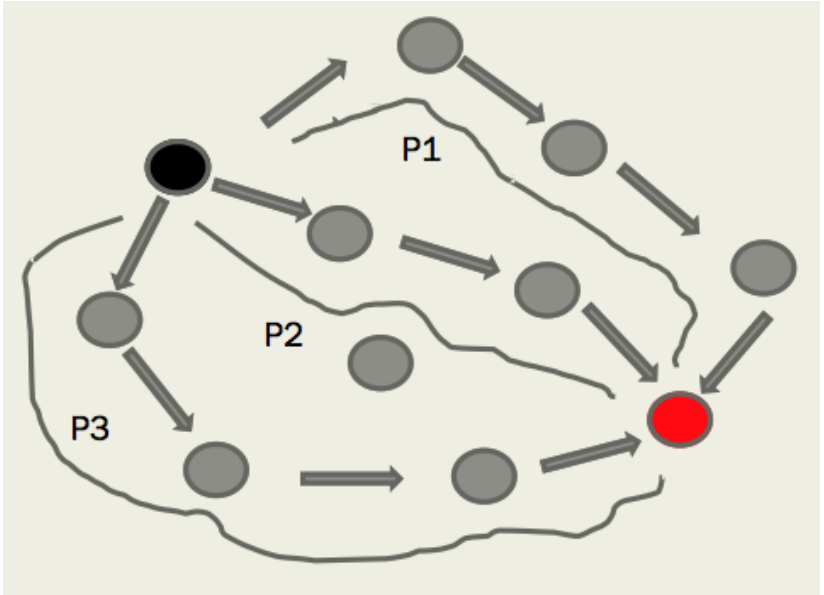
The Project Highlights

- The first to apply the computational power of a QPU (quantum processor unit) to network design, particularly in the utterly important direction of energy conservation

- The first to compare the 2000Q and Advantage_System1.1 processor performance in the application of network design

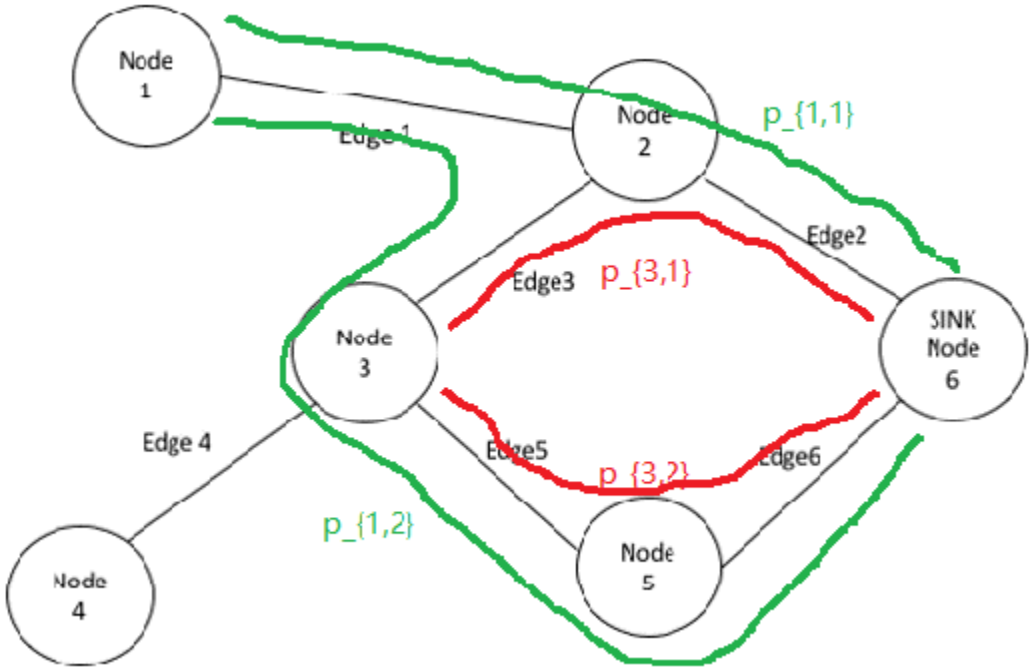
- The first to apply the Domain Wall Encoding scheme for QPU in practical engineering problem

Problem Formulation



- Each sensor node has up to three path options to select
- Each sensor node can select at most one path
- The routing table is update per interval, during which period, a given amount of bits is transmitted
- Each link has a uniform maximum rate

An Example Illustration



- Suppose Node 1 and Node 3 are transmitting at r1 and r3 respectively

$p_{1,1} = [E_1, E_2]$ and $p_{1,2} = [E_1, E_3, E_5, E_6]$
$p_{3,1} = [E_3, E_2]$ and $p_{3,2} = [E_5, E_6]$

- $X_1 = [xp_{1,1}, xp_{1,2}]$ and $X_2 = [xp_{3,1}, xp_{3,2}]$
- $e_{i,j} = \sum_{E_k} e_{i,j,k}$

- $E = xp_{1,1}e_{1,1} + xp_{1,2}e_{1,2} + xp_{3,1}e_{3,1} + xp_{3,2}e_{3,2}$

where $xp_{1,1} + xp_{1,2} = 1$ $xp_{3,1} + xp_{3,2} = 1$
and $xp_{i,j} \in Z$

$$E_{Tx}(l,d) = \begin{cases} lE_{elec} + l\epsilon_{fs}d^2, & d < d_0. \\ lE_{elec} + l\epsilon_{mp}d^4, & \text{else} \end{cases}$$

data processing data transmission

$$E_{Rx}(l,d) = lE_{elec}$$

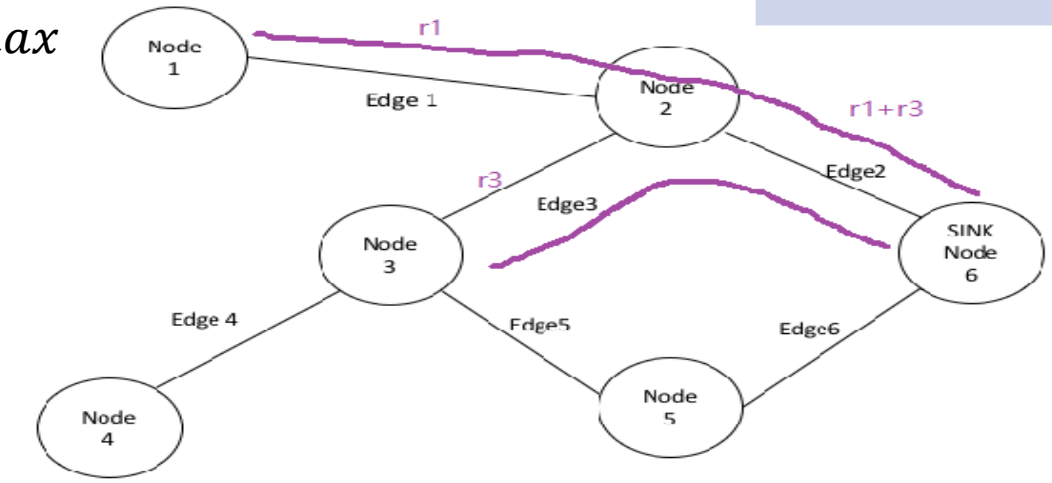
TABLE I: Parameter Value

Name	Value
E_{elec}	50 nJ/bit
ϵ_{mp}	$0.0013 \frac{pj}{bit}/m^4$
ϵ_{fs}	$10 \frac{pj}{bit}/m^2$
d_0	$\sqrt{\frac{\epsilon_{fs}}{\epsilon_{mp}}} = 87.7058$

An Example Illustration

- 4 combinations of path $[X_{1,1}, X_{3,1}]$, $[X_{1,2}, X_{3,1}]$
- $[X_{1,1}, X_{3,2}]$, $[X_{1,2}, X_{3,2}]$
- For the first combination, there will be three edges $[E_1, E_2, E_3]$ in use.
- For E_1, E_3: $r_1 < C_{max}$, $r_3 < C_{max}$
- For E_2: $r_1 + r_3 < C_{max}$

Symbol	Notation
$p_{i,j}$	the j^{th} path for node i
E_i	Edge i
$xp_{i,j}$	Is 1 if the j^{th} path for node I is selected
$e_{i,j}$	the j^{th} path of node i energy consumption
$e_{i,j,k}$	the k^{th} hop of the j^{th} path for node i energy consumption



Problem Break Down

- The energy aware problem is formulated as an integer linear programming problem as below:

$$\begin{aligned} \min_{x_i} \quad & \sum_{alledgej} \{f(d_j)(\sum_{j \in x_i} x_i r_n \Delta t) + E_{elec} * \sum_{j \in x_i} x_i r_n \Delta t\} \\ \text{s.t.} \quad & \sum_{j \in x_i} x_i * r_n \leq C_{max} \ \forall edge_j \\ & \sum_{x_i \in n} x_i = 1 \ \forall n \end{aligned}$$

- The first inequality is mitigated by using slack variable and the second equality is met by coding the problem according to the domain wall scheme

Algorithm

Algorithm 1 Hybrid Algorithm Procedure

1: Call the sub procedure to collect all feasible paths - PathCollector
2: Call the sub procedure to assign paths to respective edges - getEdgeM
3: Call the sub procedure to formulate QUBO problem - makeEffArray
4: Call the sub procedure to encode the QUBO problem - makeEncoding
5: Call the QPU API Solver

Measurements

- Correct Rate
  - Number of problem instances that reach the minimum (energy/processing time) to the overall number of problem instances
- Incorrect rate
  - Fraction of the samples which returned an solution that is not optimal
- Embedding error rate
  - Faction of the samples which failed to be embedded

Measurements

- Effectiveness Plot
  - How well/badly the respective QPU performs in terms of accuracy against the classical solvers
  - Correct Rate, Incorrect Rate, Embedding Error Rate
- Speediness Plot
  - How well/badly the respective QPU performs in terms of speed against all the other solvers
  - Correct Rate

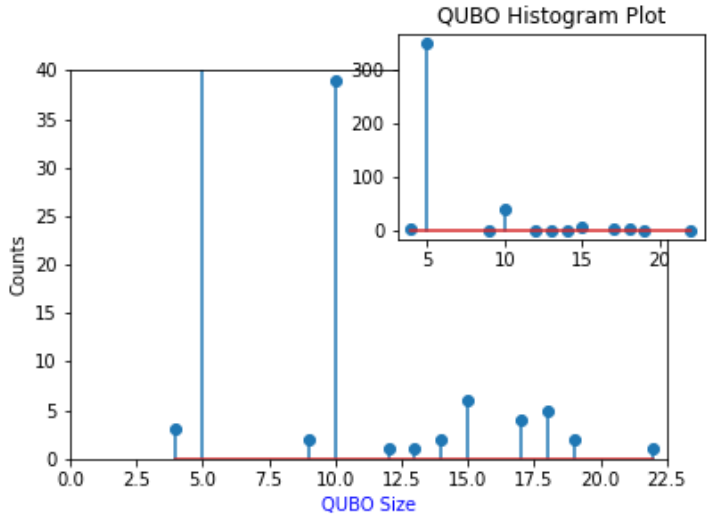
Categories of Experiments

- Go over all possible combinations of graph size and source number excluding the flow rate

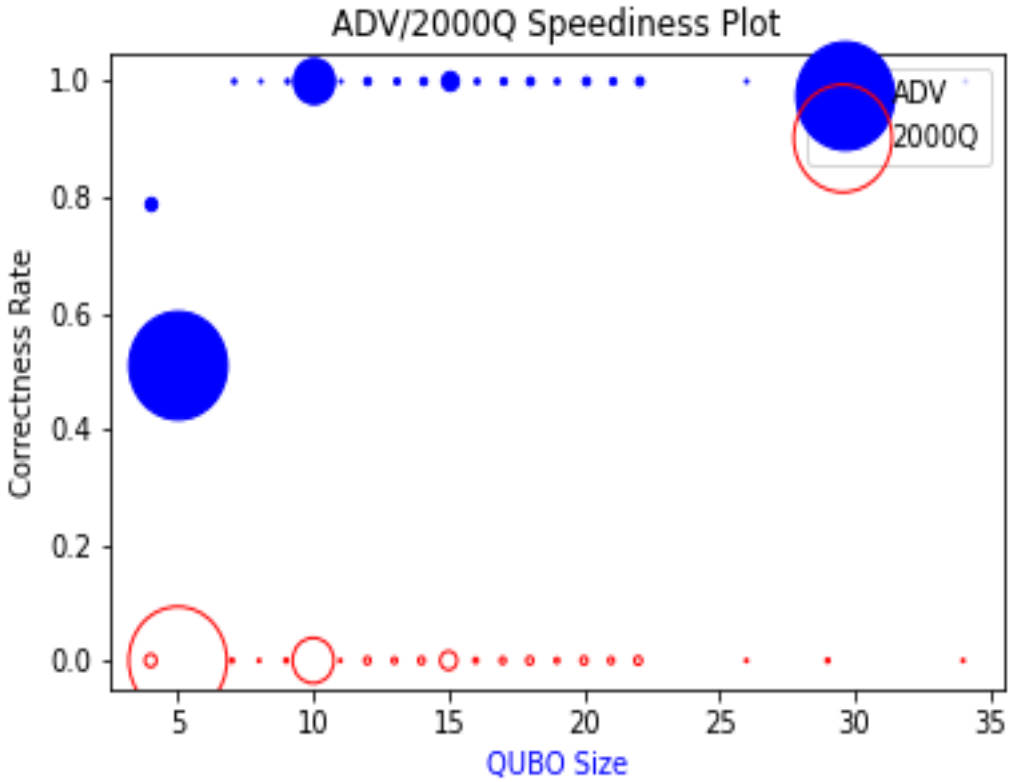
- Apply Erdos-Renyi graph generation algorithm and generate 20 problem samples each graph size from (5 to 12)

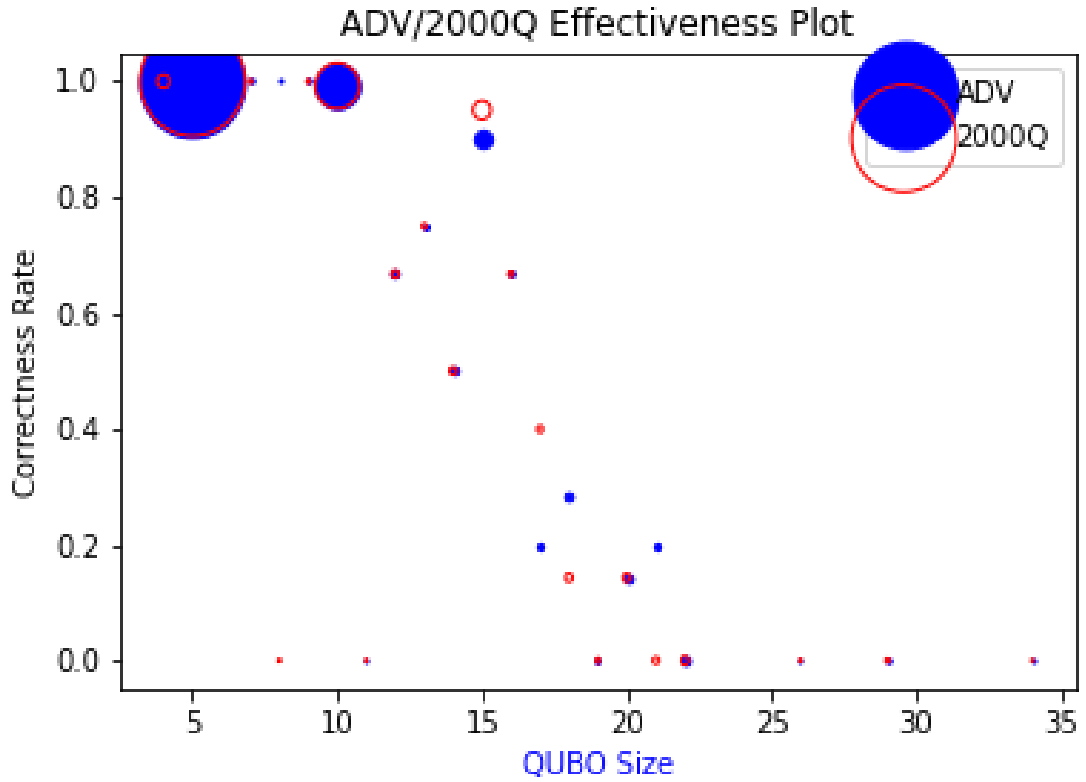
Results- Graph Size=5

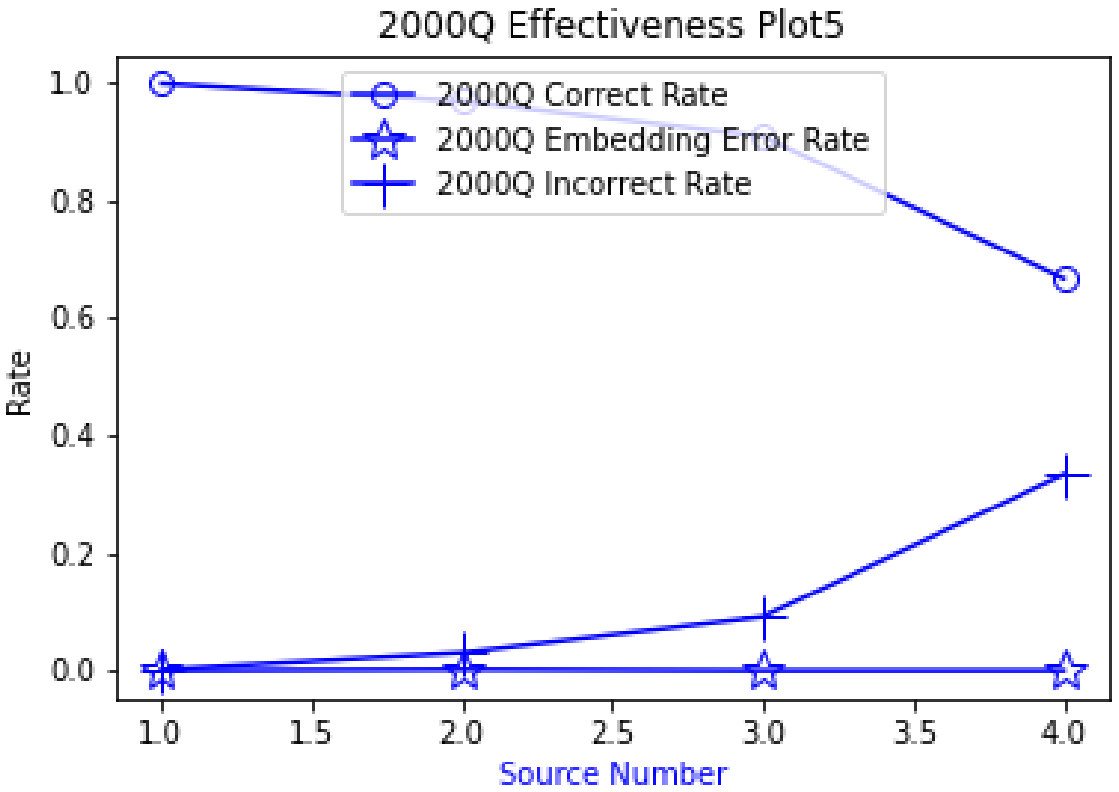
- Advantage_sys1.1 has an absolute advantage in the speed as QUBO size increases to around 10.

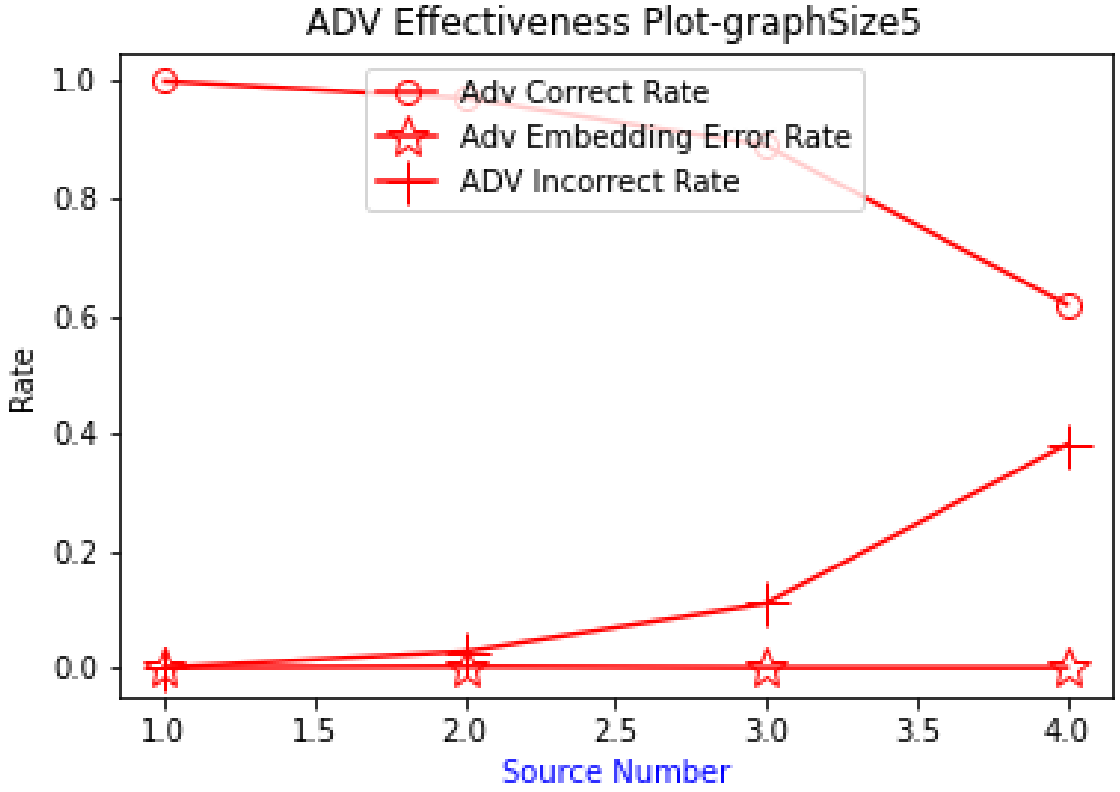


- QPU stops functioning when QUBO size reaches 20 and above;
- A dominating effective accuracy with QUBO size below 10;
- The size of the bubble indicates the amount of solutions within the same QUBO size







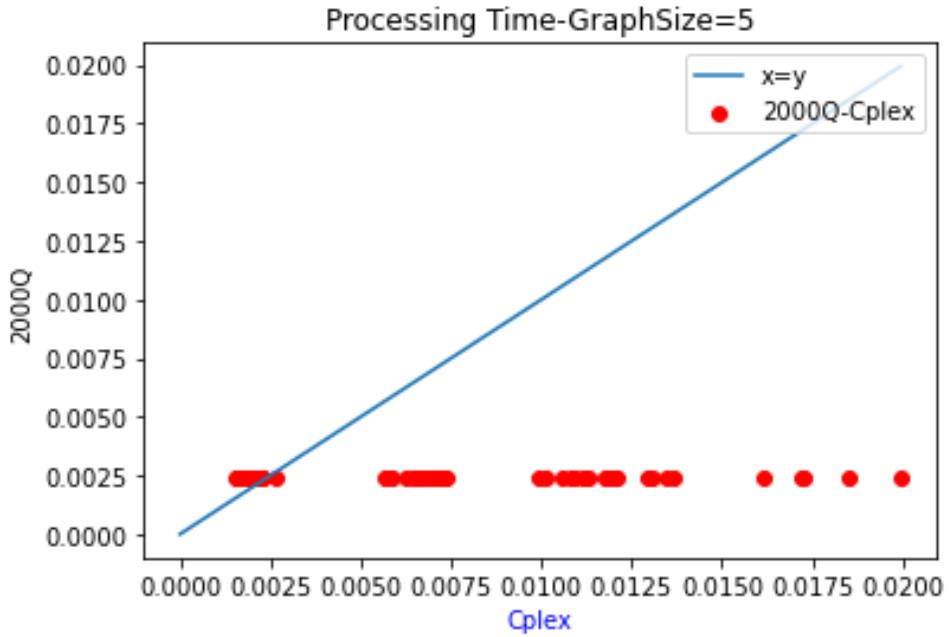


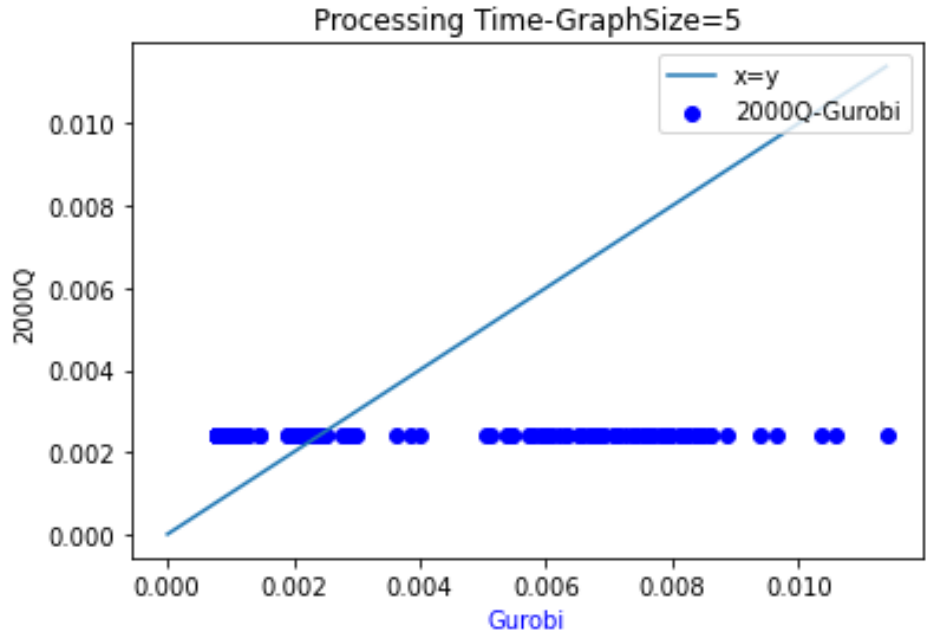
- Correct Rate decreases piecewise linearly as the source number increases

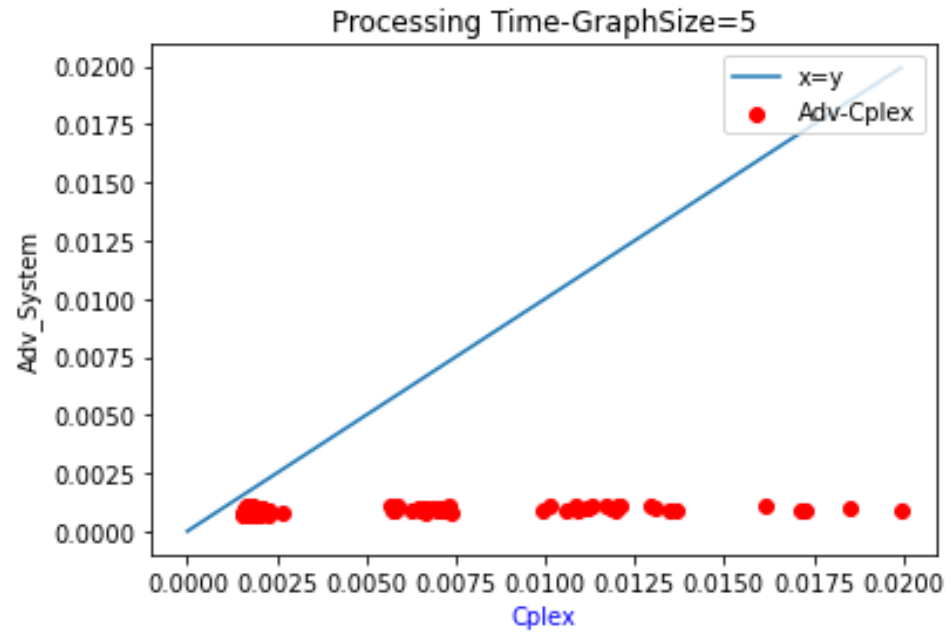
- 60% and above samples are solved faster by Advantage_sys1.1 than 2000Q and classical solvers

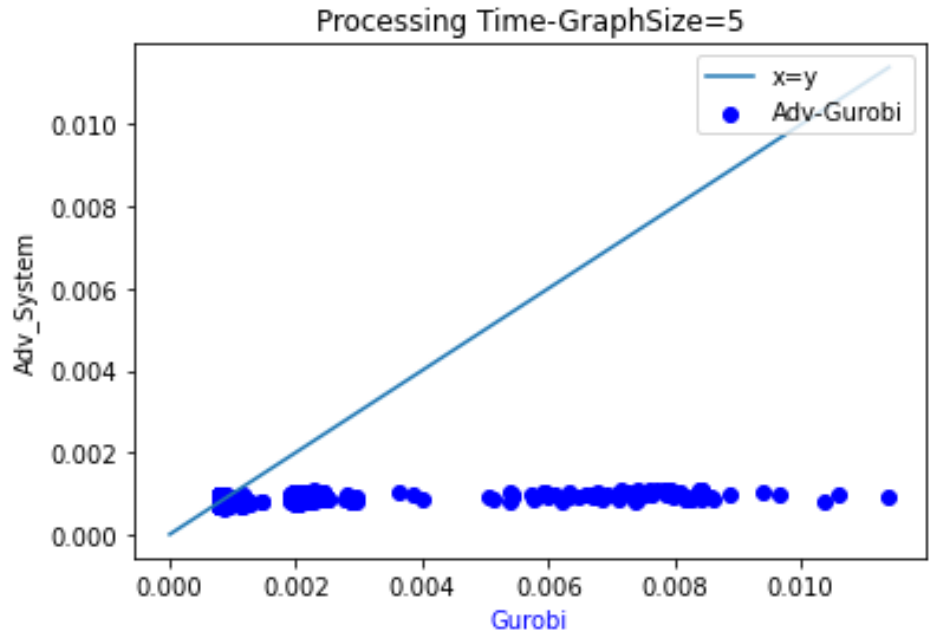
- No embedding error exists across all the samples

Processing Time Comparison









Results – Graph Size =4

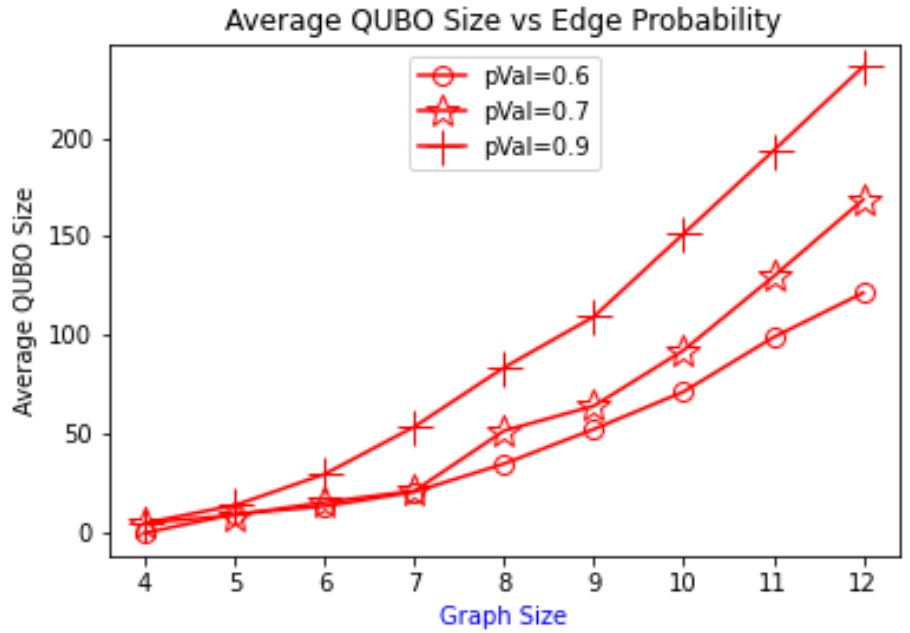
- Problem instances that are feasible for submission to the QPU. QUBO size is 5

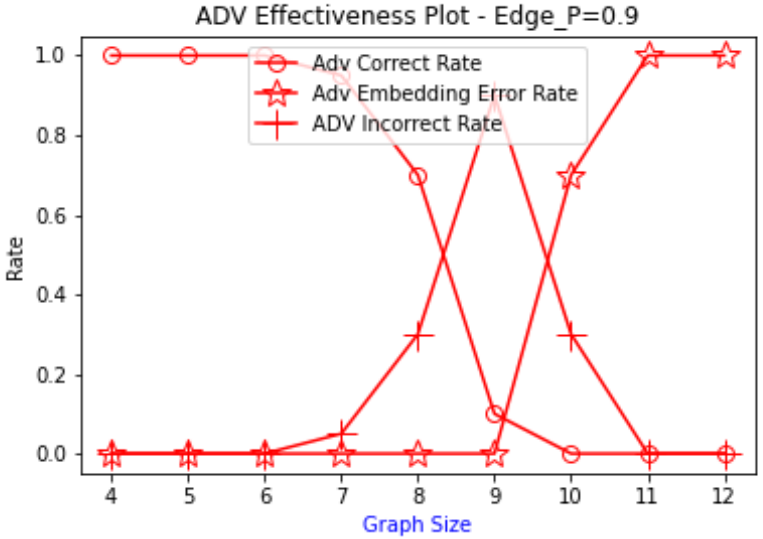
- QPU demonstrates an absolute advantage over classical solvers regarding the solution quality across the whole problem space.

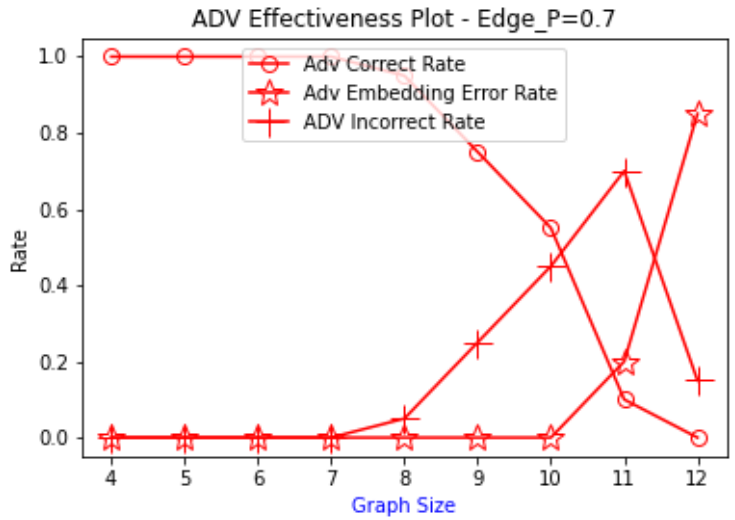
- With number of reads per run set to 3000, QPU doesn’t show an advantage in the speed while with number of reads decreased to 5, QPU speed overrides those by the classical solvers without degrading the solution quality

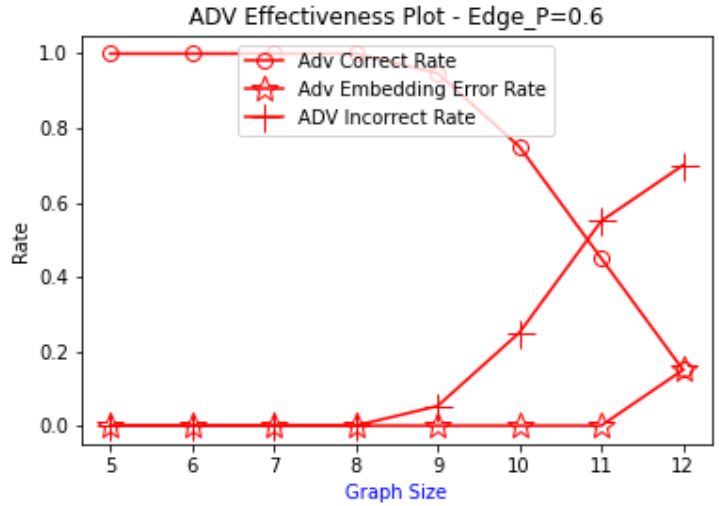
Results – Renyi-Erdos Graph Generator with probability set to 0.6,0.7 and 0.9 respectively

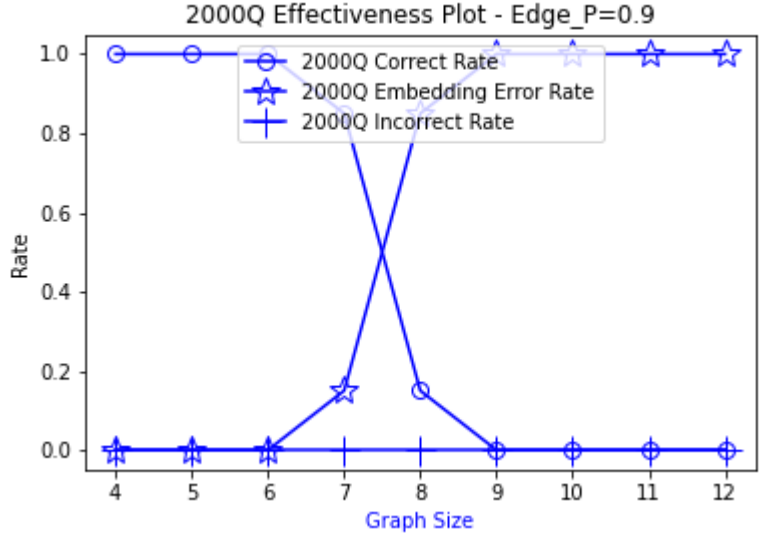
- Performance degradation point moves rightwards as the edge probability decreases;
- Average QUBO size increases as the edge probability increases;
- Embedding error appears at larger graph size with a lower value as the edge probability decreases;
- Advantage_sys1.1 shows an absolute advantage in speed over the other three solvers

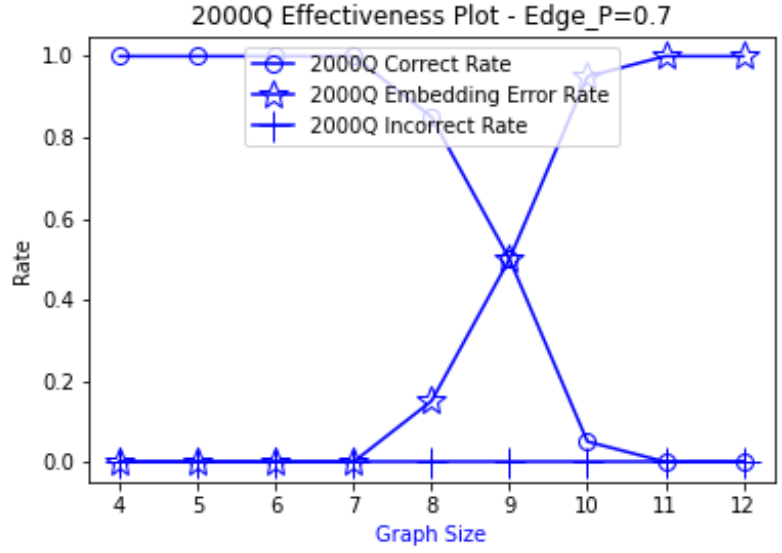


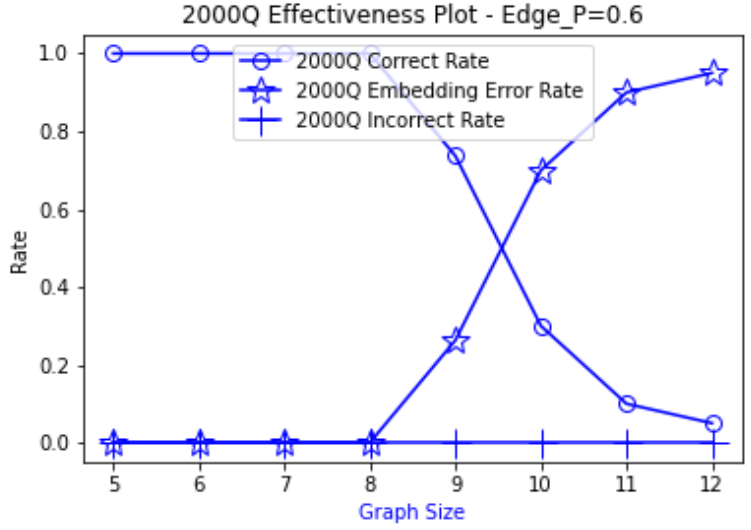

ADV Effectiveness Plot - Edge_P=0.9
Adv Correct Rate
Adv Embedding Error Rate
ADV Incorrect Rate
Rate
Graph Size

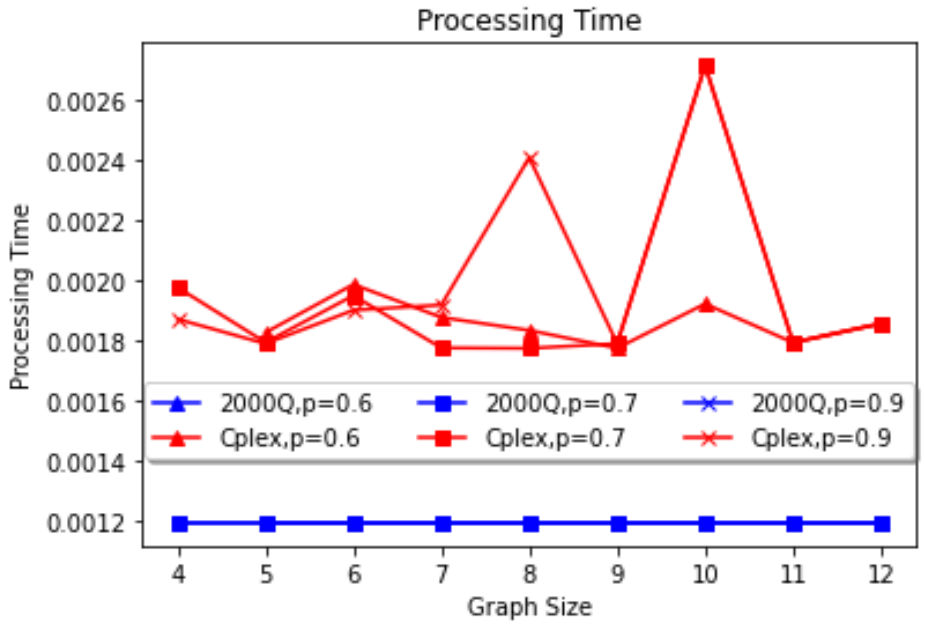

ADV Effectiveness Plot - Edge_P=0.7
Adv Correct Rate
Adv Embedding Error Rate
ADV Incorrect Rate
Rate
Graph Size

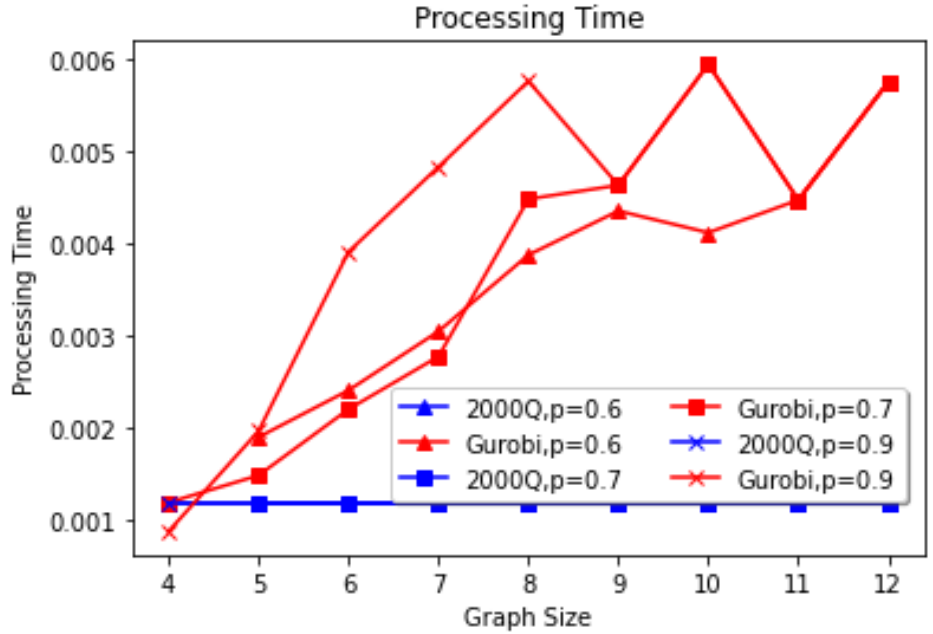

ADV Effectiveness Plot - Edge_P=0.6
Adv Correct Rate
Adv Embedding Error Rate
ADV Incorrect Rate
Rate
Graph Size

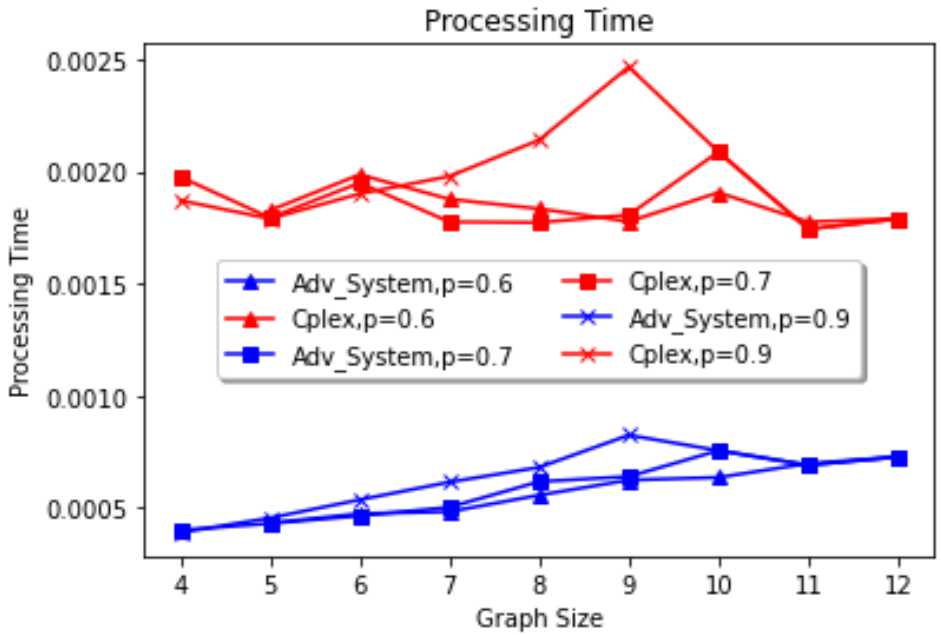

2000Q Effectiveness Plot - Edge_P=0.9
2000Q Correct Rate
2000Q Embedding Error Rate
2000Q Incorrect Rate
Rate
Graph Size

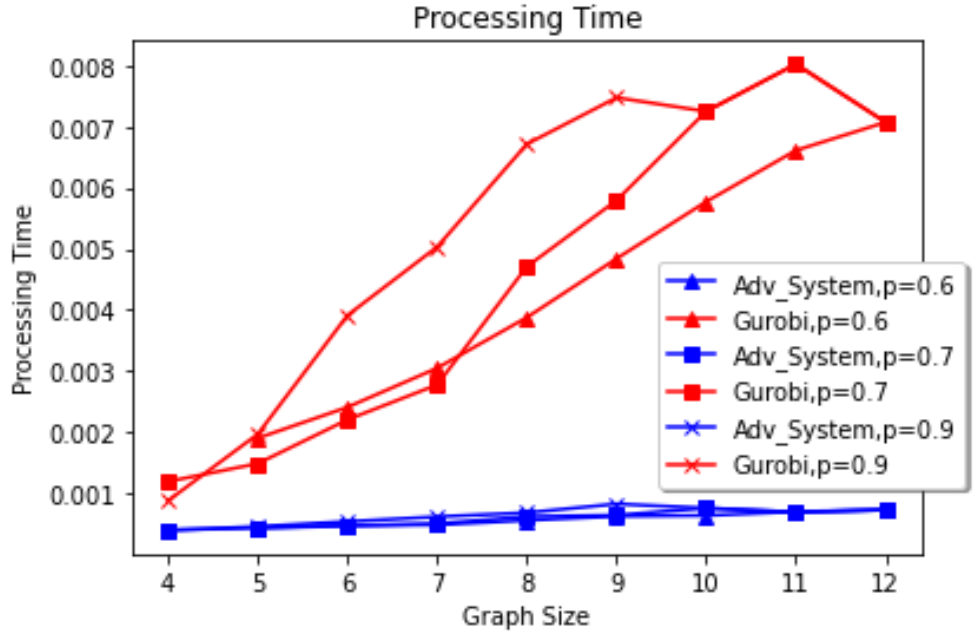

2000Q Effectiveness Plot - Edge_P=0.7
2000Q Correct Rate
2000Q Embedding Error Rate
2000Q Incorrect Rate
Rate
Graph Size


2000Q Effectiveness Plot - Edge_P=0.6
2000Q Correct Rate
2000Q Embedding Error Rate
2000Q Incorrect Rate
Rate
Graph Size

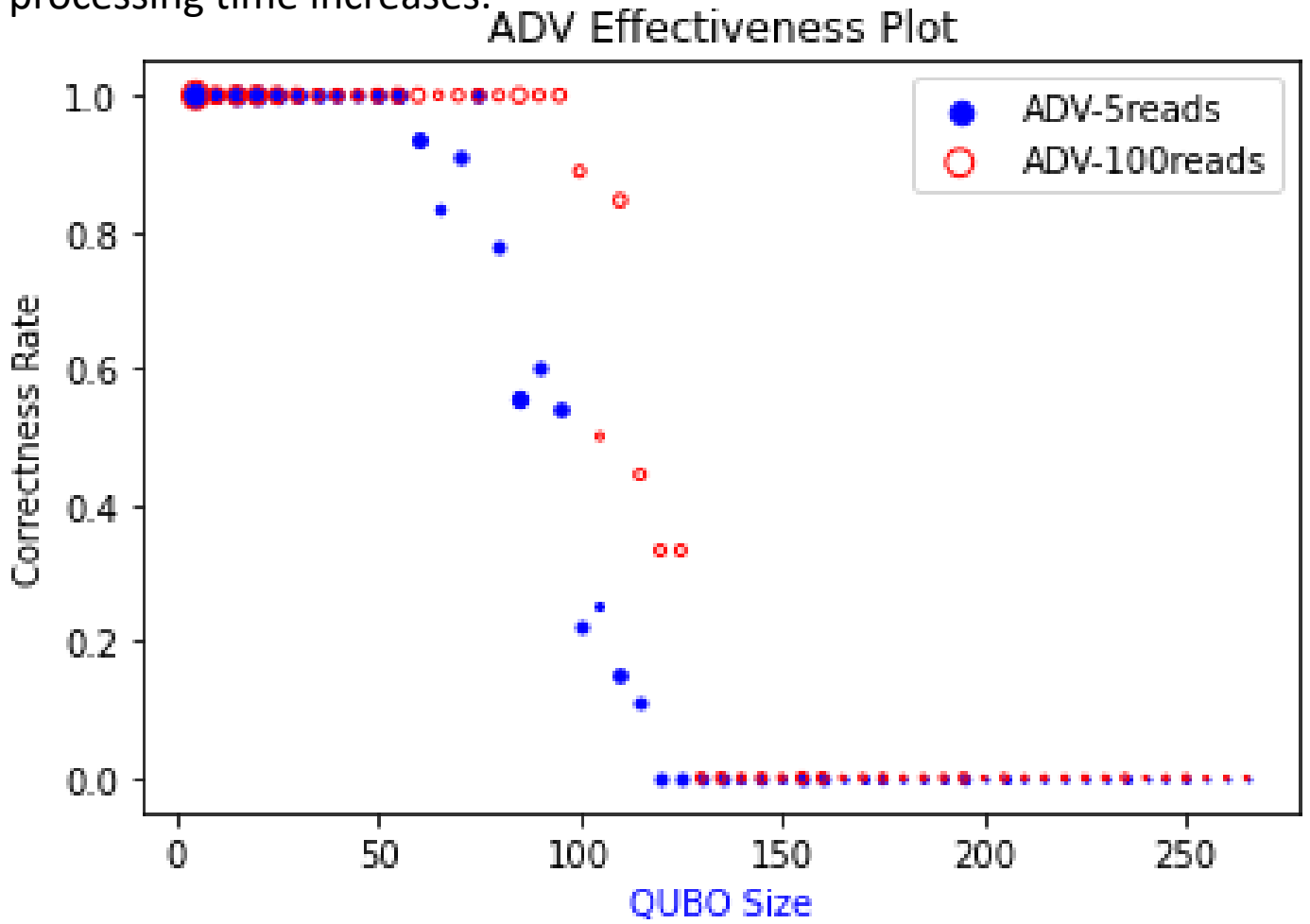

Processing Time
Processing Time
Graph Size
2000Q,p=0.6
Cplex,p=0.6
2000Q,p=0.7
Cplex,p=0.7
2000Q,p=0.9
Cplex,p=0.9

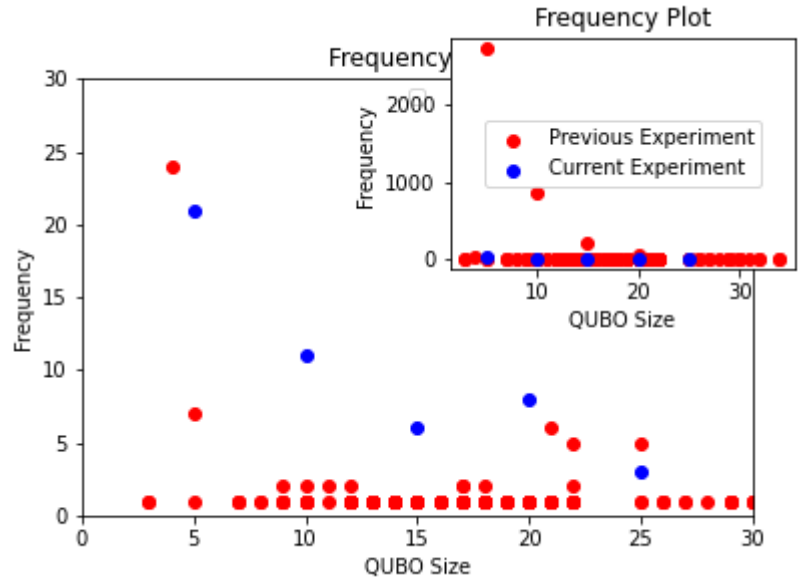

Processing Time
Processing Time
Graph Size
2000Q,p=0.6
Gurobi,p=0.6
2000Q,p=0.7
Gurobi,p=0.7
2000Q,p=0.9
Gurobi,p=0.9


Processing Time
Processing Time
Graph Size
Adv_System,p=0.6
Cplex,p=0.6
Adv_System,p=0.7
Cplex,p=0.7
Adv_System,p=0.9
Cplex,p=0.9


Processing Time
Processing Time
Graph Size
Adv_System,p=0.6
Gurobi,p=0.6
Adv_System,p=0.7
Gurobi,p=0.7
Adv_System,p=0.9
Gurobi,p=0.9

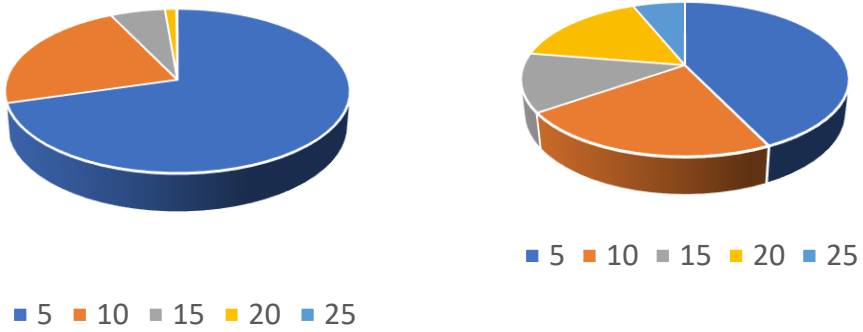
- With the increase of number of reads, the success probability increases, we can notice that the effectiveness plot has been shifted rightwards. The watershed starts around QUBO size 60 for 5 reads while for reads 100, the watershed starts around above QUBO size 100.
- Common sense will tell that with the increase of number of reads, the processing time increases.





- In the Brute Force Method -5 Graph Experiment : The watershed starts around QUBO size = 15

- In the Probabilistic Method -5 Graph Experiment : there is no watershed



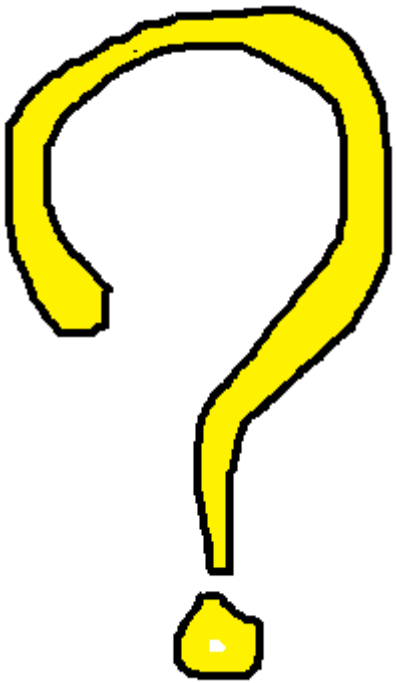
Future Work

- Investigation into the experiment by Redos_Renyi graph generation algorithm as the QUBO size after trimming falls neatly into the set {5,10,15,20,25} for graph size=5 for example – needs more rigorously mathematical explanation
- Early Prototype
  - Network Resilience – responsive to the fast changing dynamic

Experiment Configurations

- $C_{max} = 5$,d=10
- Flow rate val_i is generated following uniform distribution
- Number of samples per run is 10 by default
- For GUROBI and CPLEX, timers are deployed before and after the solver call, the solver processing is the lap between them
- For QPU, qpu_sampling_time within the ‘timing’ info
- Anneal time is by default 20μs
- Fixed_variable technique is used to slim the QUBO size

THE END



Email: cencen.cj2021@protonmail.com