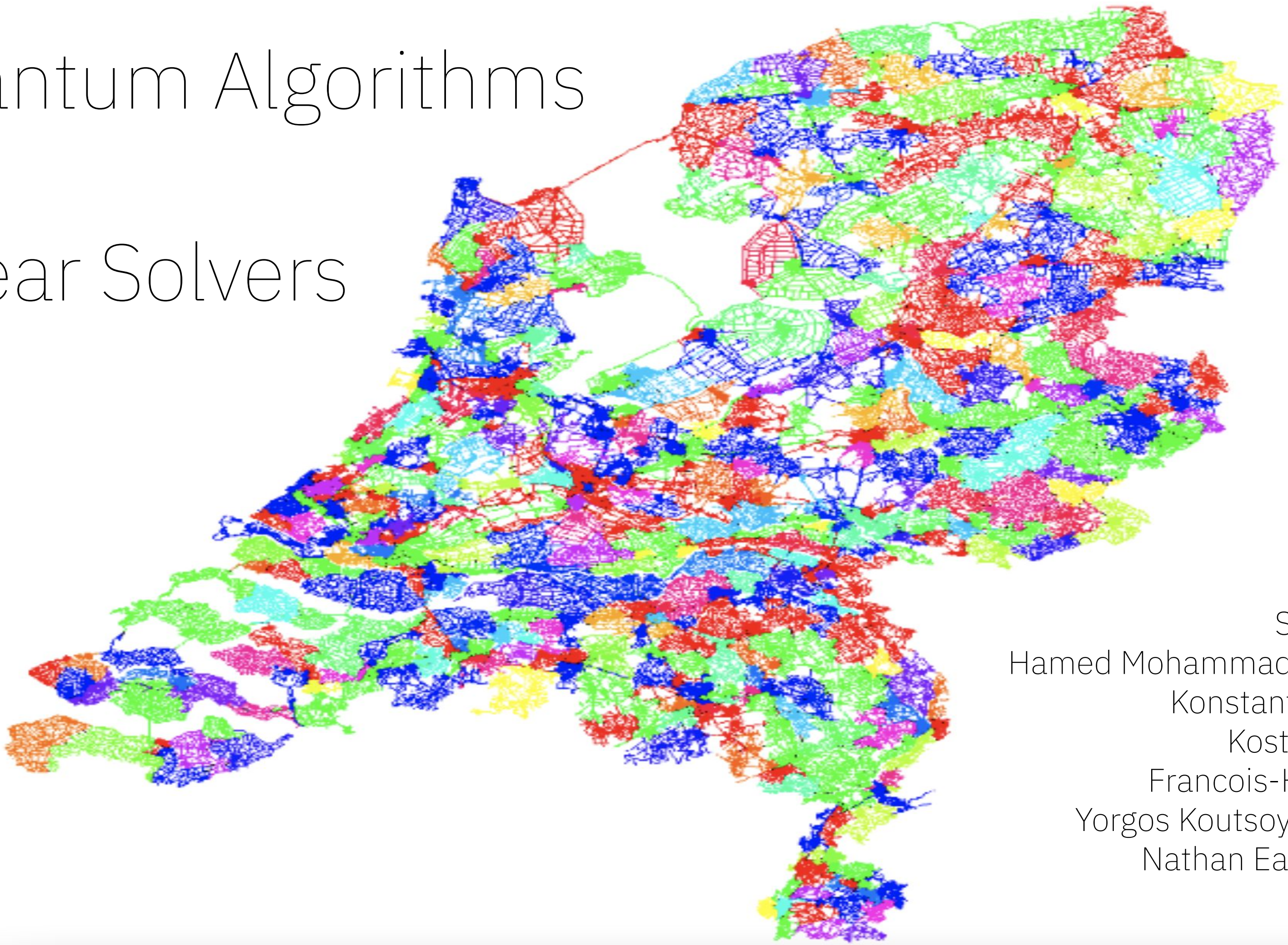
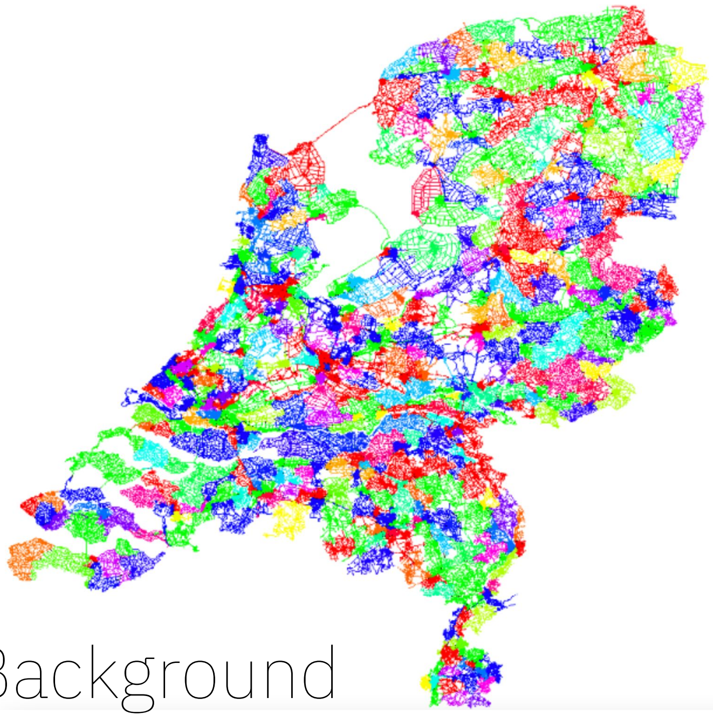
Quantum Algorithms for Linear Solvers



Sophia Kolak
Hamed Mohammadbagherpoor
Konstantis Daloukas
Kostas Kafousas
Francois-Henry Rouet
Yorgos Koutsoyannopoulos
Nathan Earnest-Noble
Bob Lucas

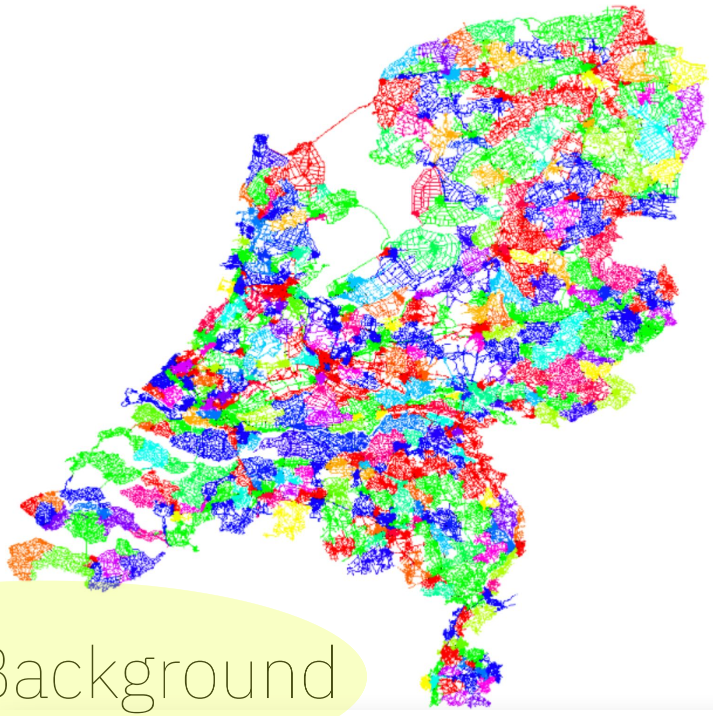
Guide



Part 1: Background

Part 2: Technical

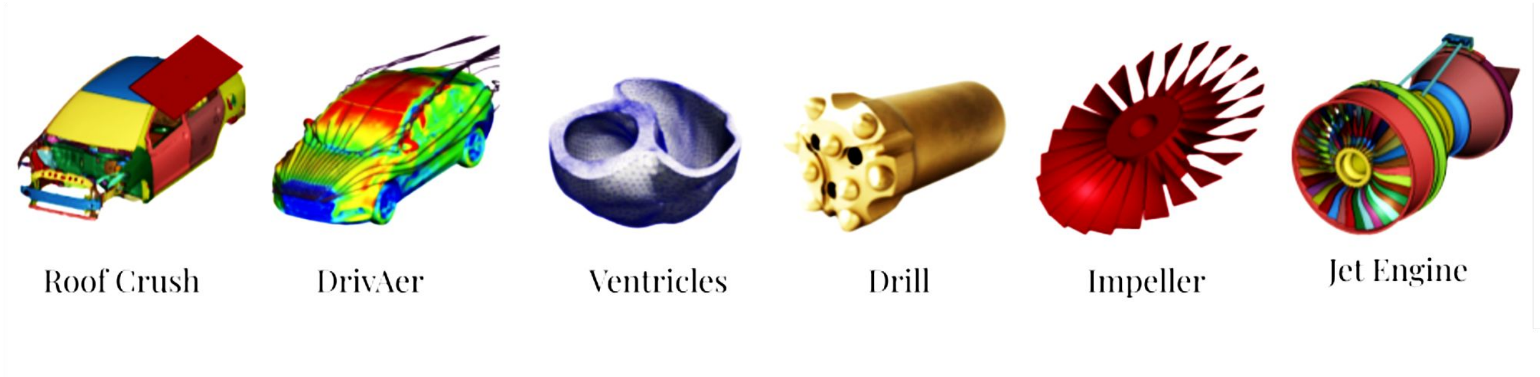
Guide

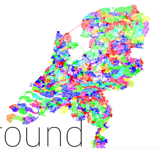


Part 1: Background

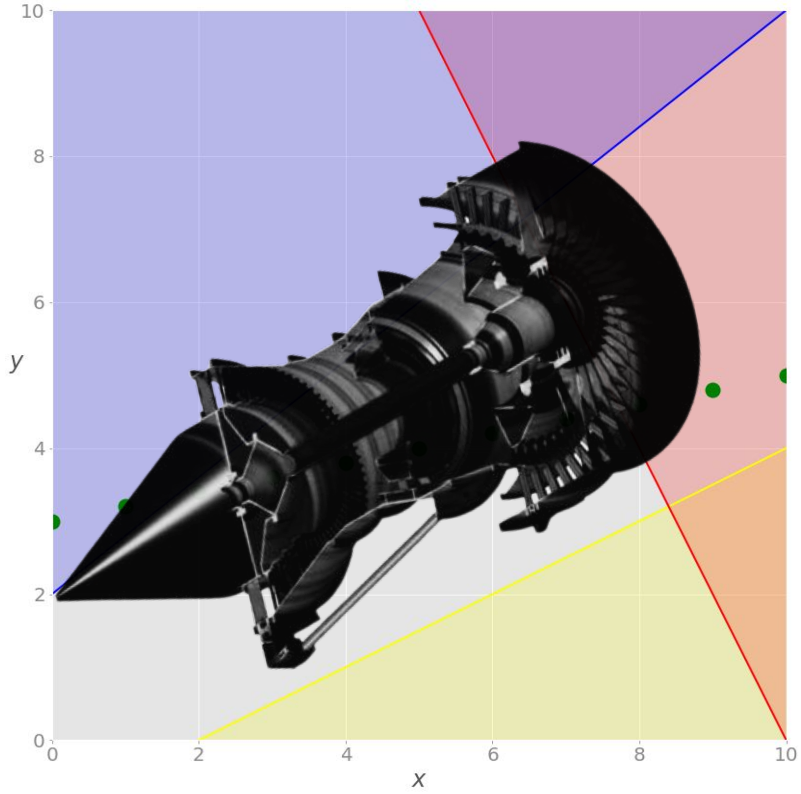
Part 2: Technical

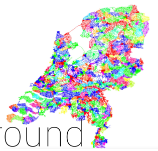
Ansys Simulates Objects





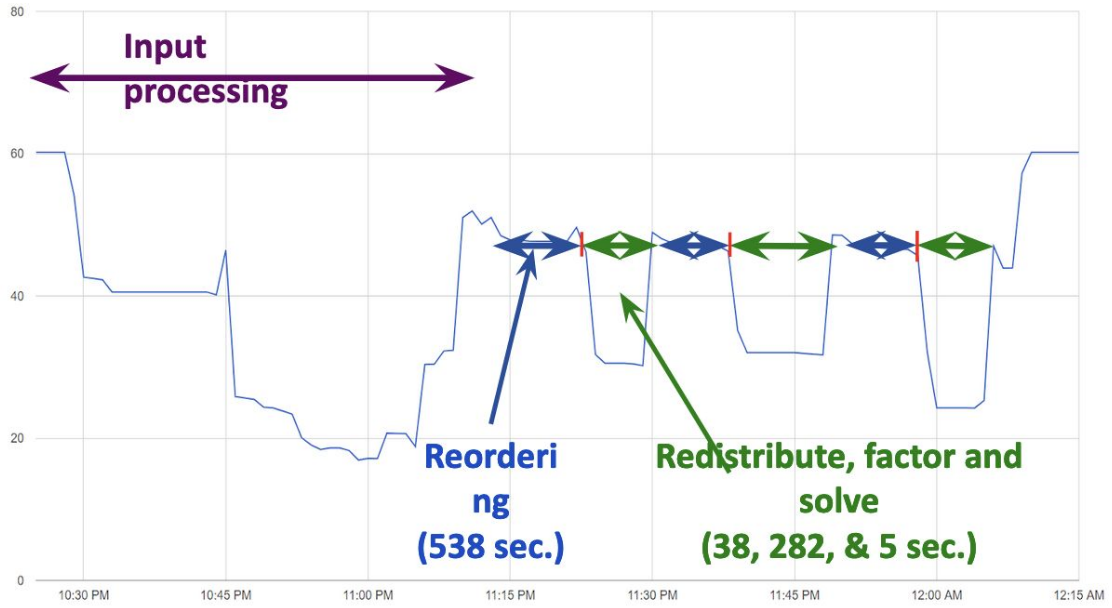
Simulating Objects Requires Solving Linear Equations



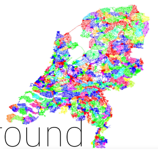


Background

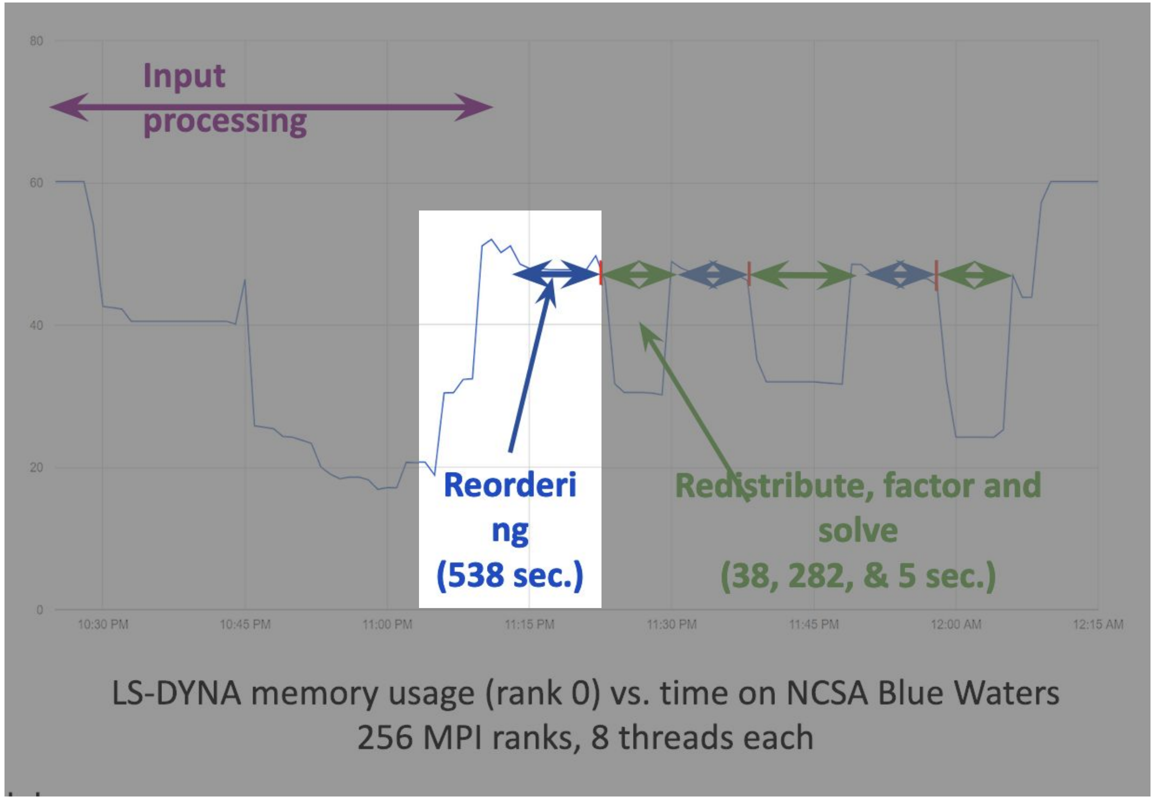
Linear Equation solving is Slow



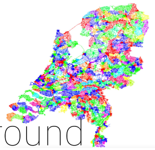
LS-DYNA memory usage (rank 0) vs. time on NCSA Blue Waters
256 MPI ranks, 8 threads each



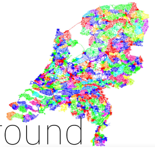
Reordering (**Graph Partitioning**) Dominates Runtime



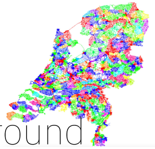
LS-DYNA memory usage (rank 0) vs. time on NCSA Blue Waters
256 MPI ranks, 8 threads each



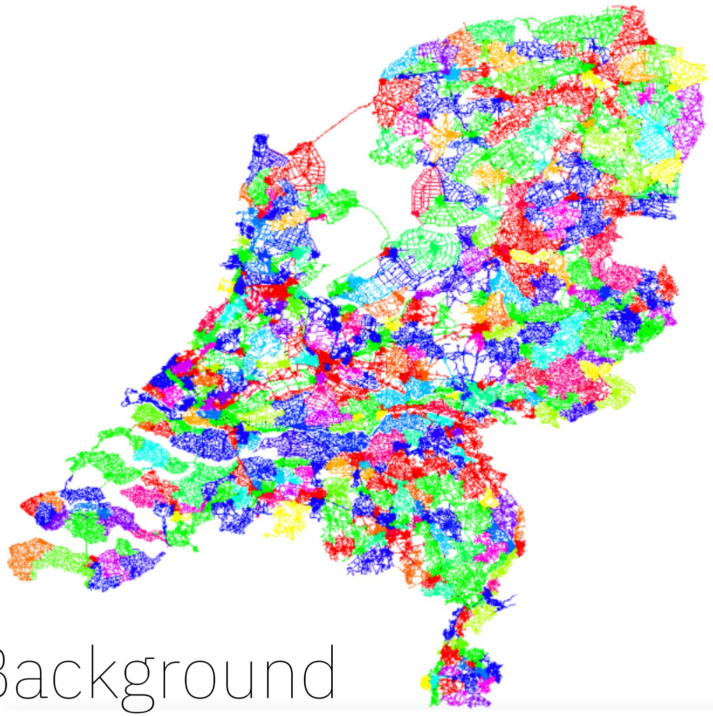
Graph Partitioning: the Bottleneck within the Bottleneck



We want to speed it up.



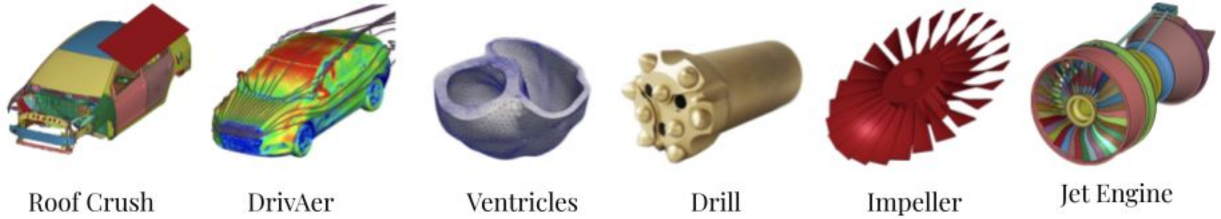
Guide

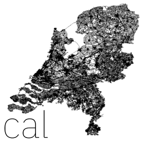


Part 1: Background

Part 2: Technical

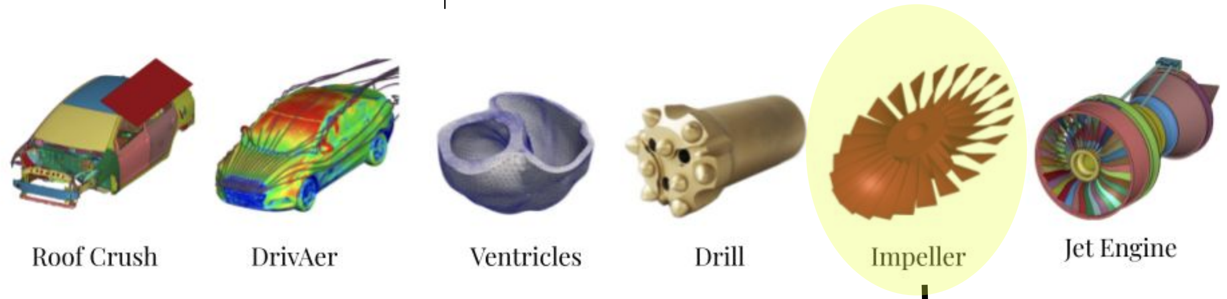
Coarsened Graphs

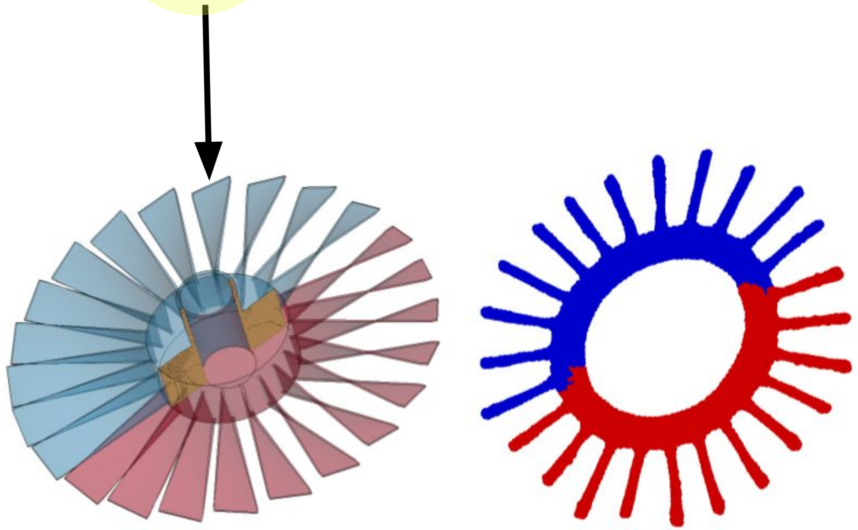


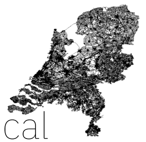


Technical

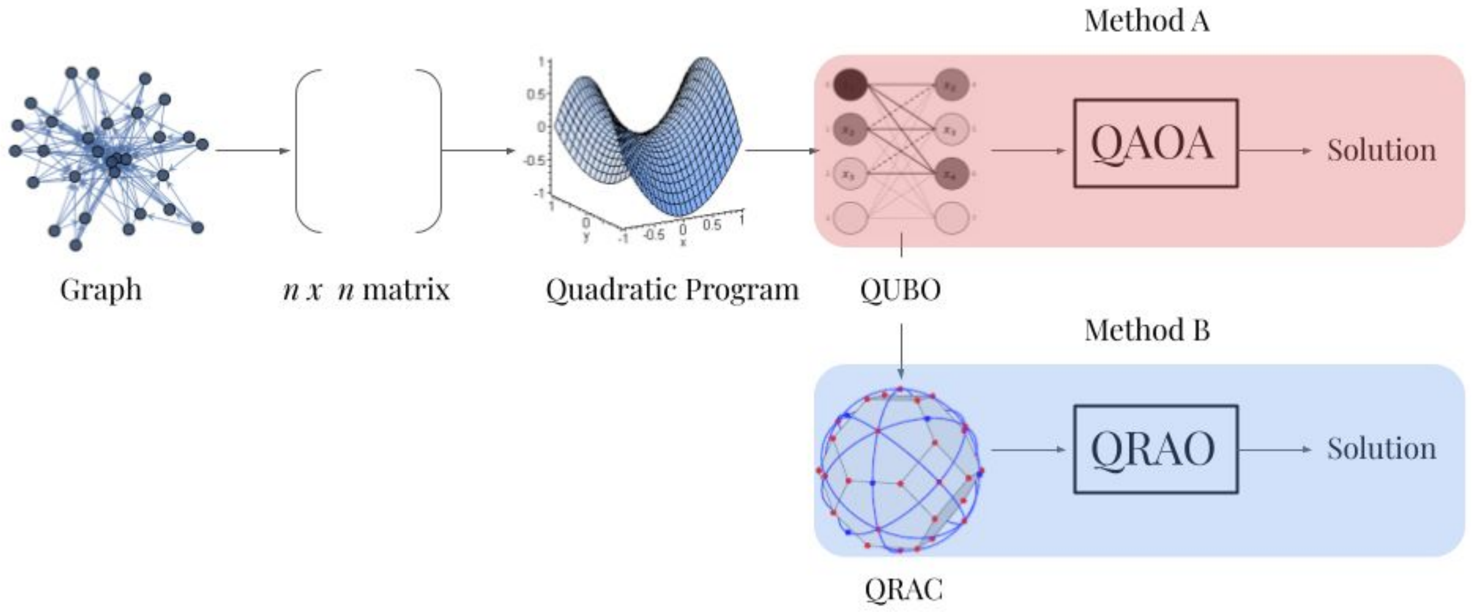
Coarsened Graphs

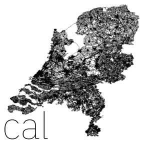




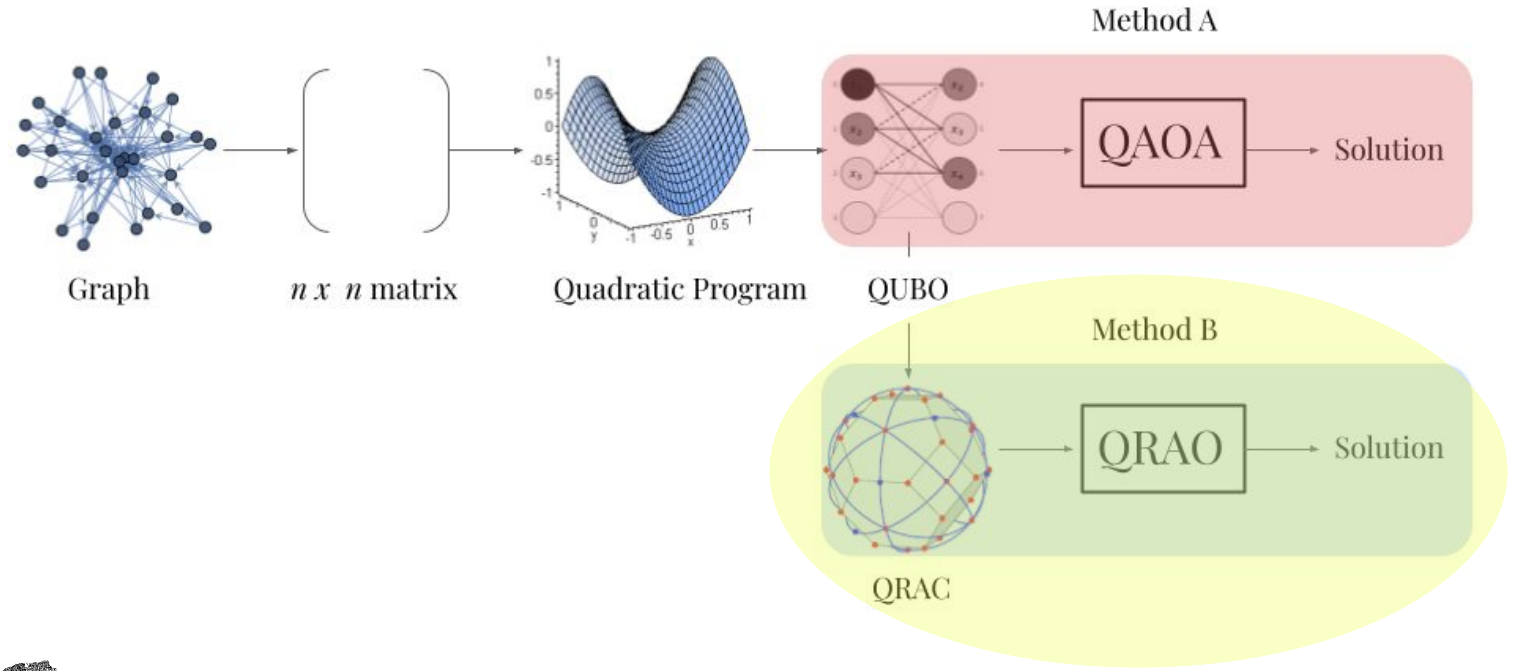

Technical

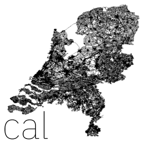
Two Methods for Quantum Graph Partitioning:





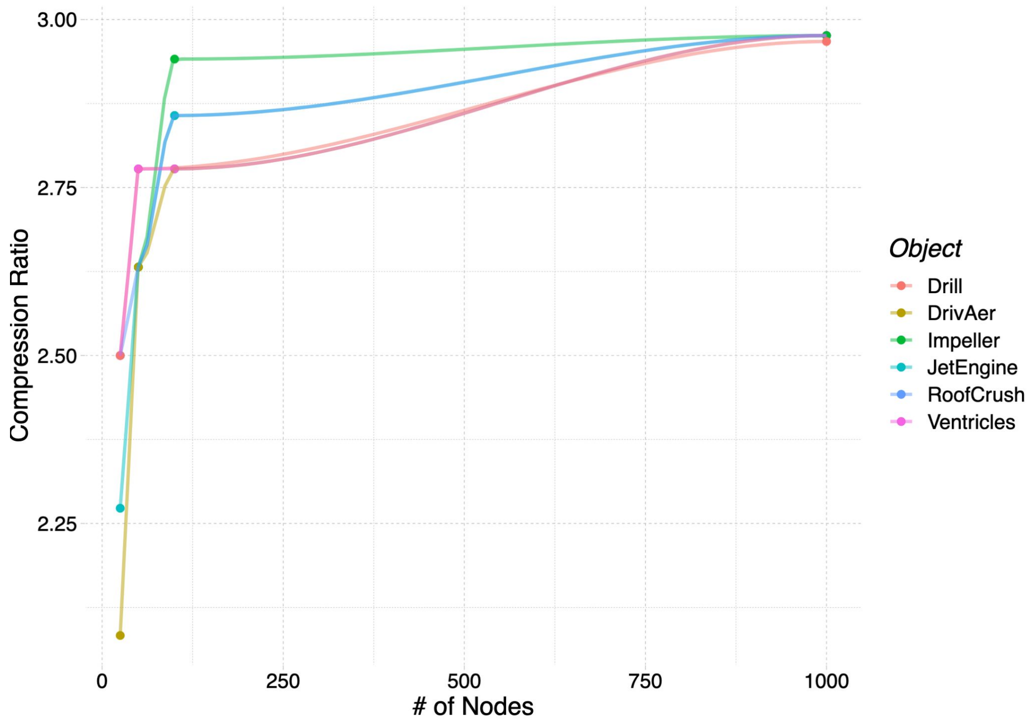
Two Methods for Quantum Graph Partitioning:

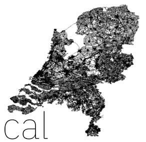




Technical

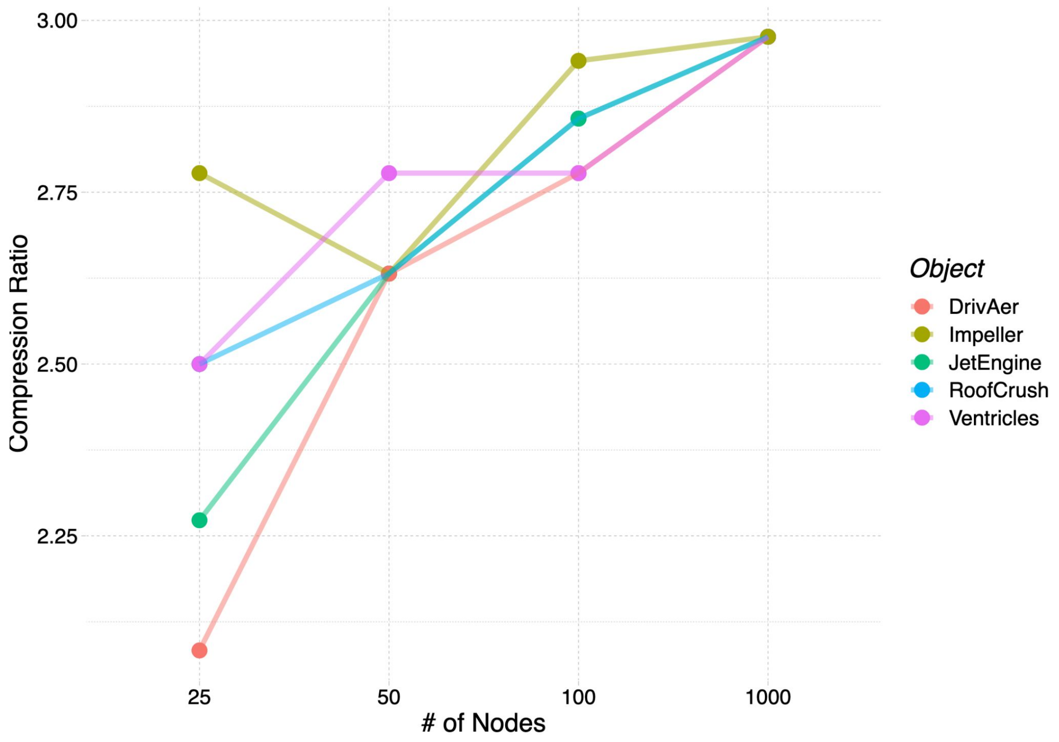
Compression Ratio Improved With Size

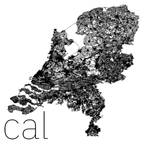




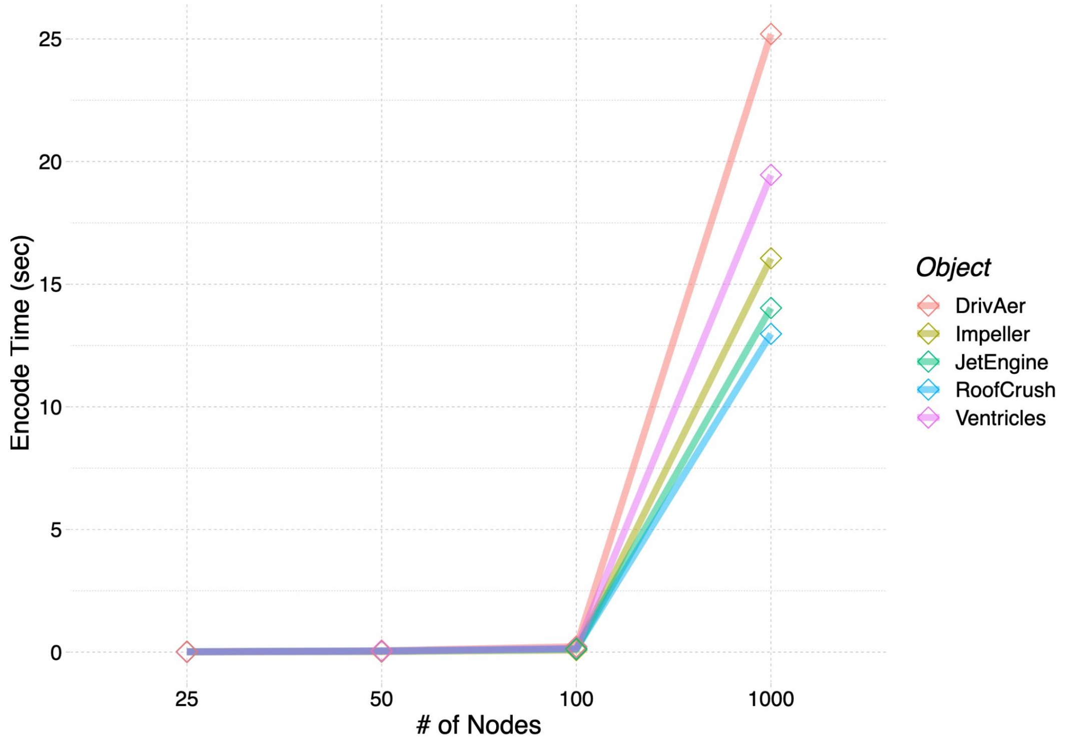
Technical

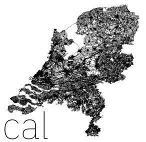
Compression Ratio Varied by Object



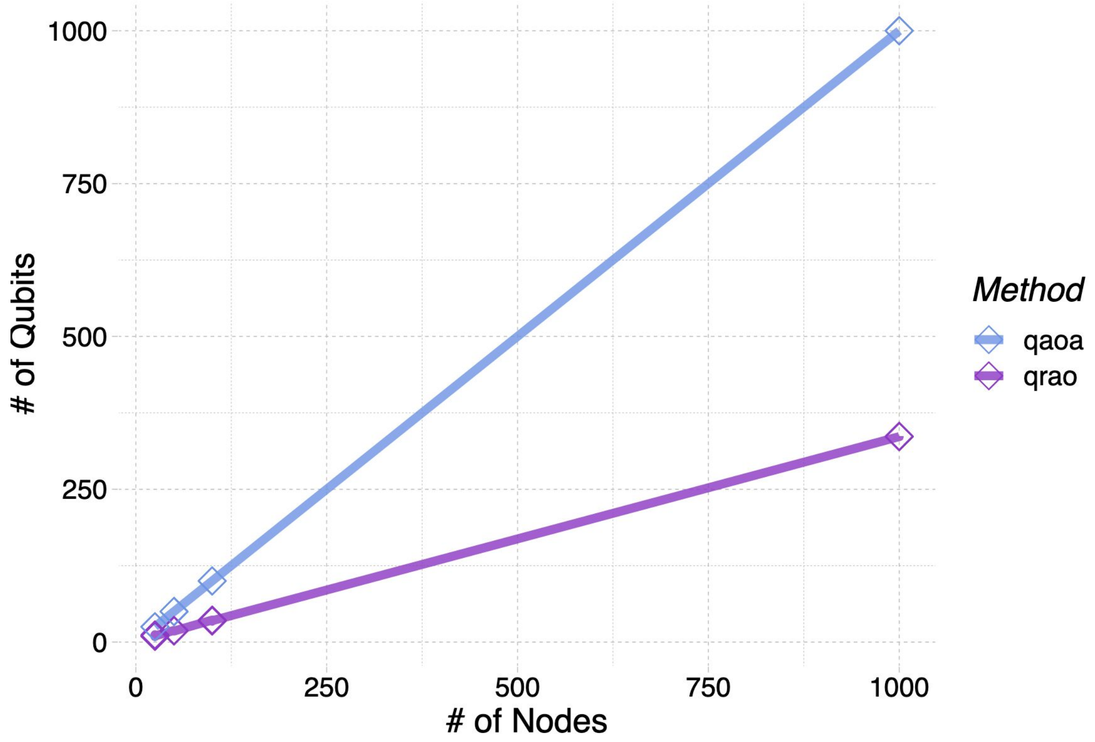


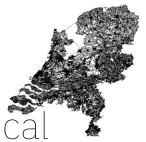
Encoding Time Scaled Exponentially





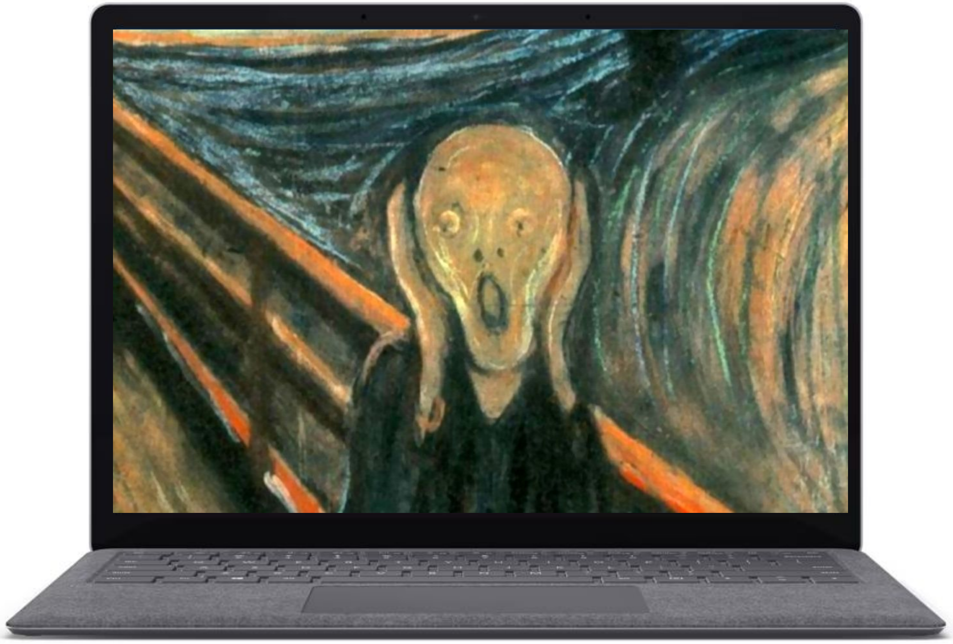
Still Seemed Preferable to QAOA





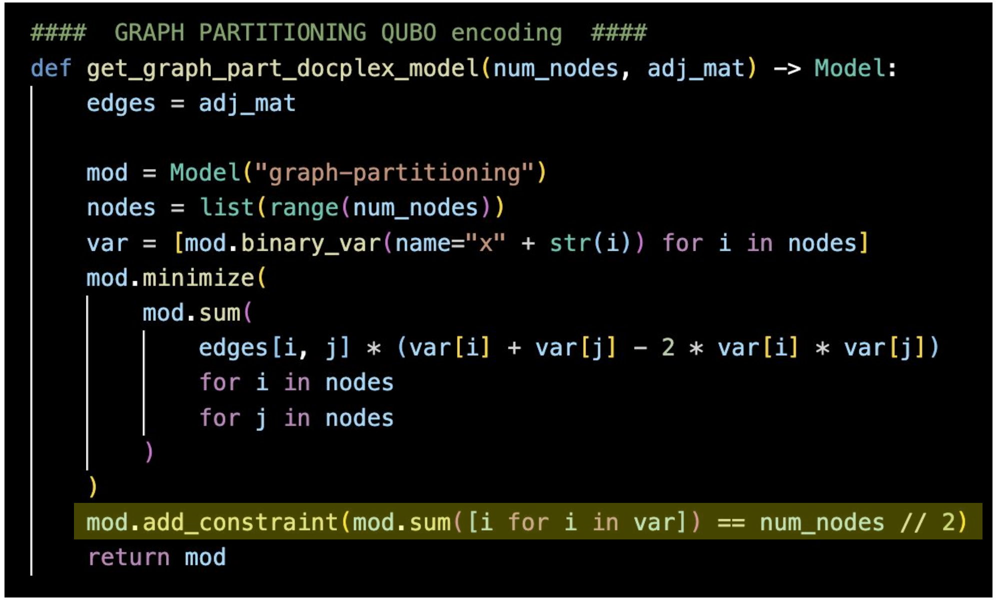
Technical

Trouble!

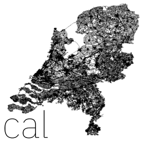


Technical

Balancing Constraint Removes Encoding Benefits

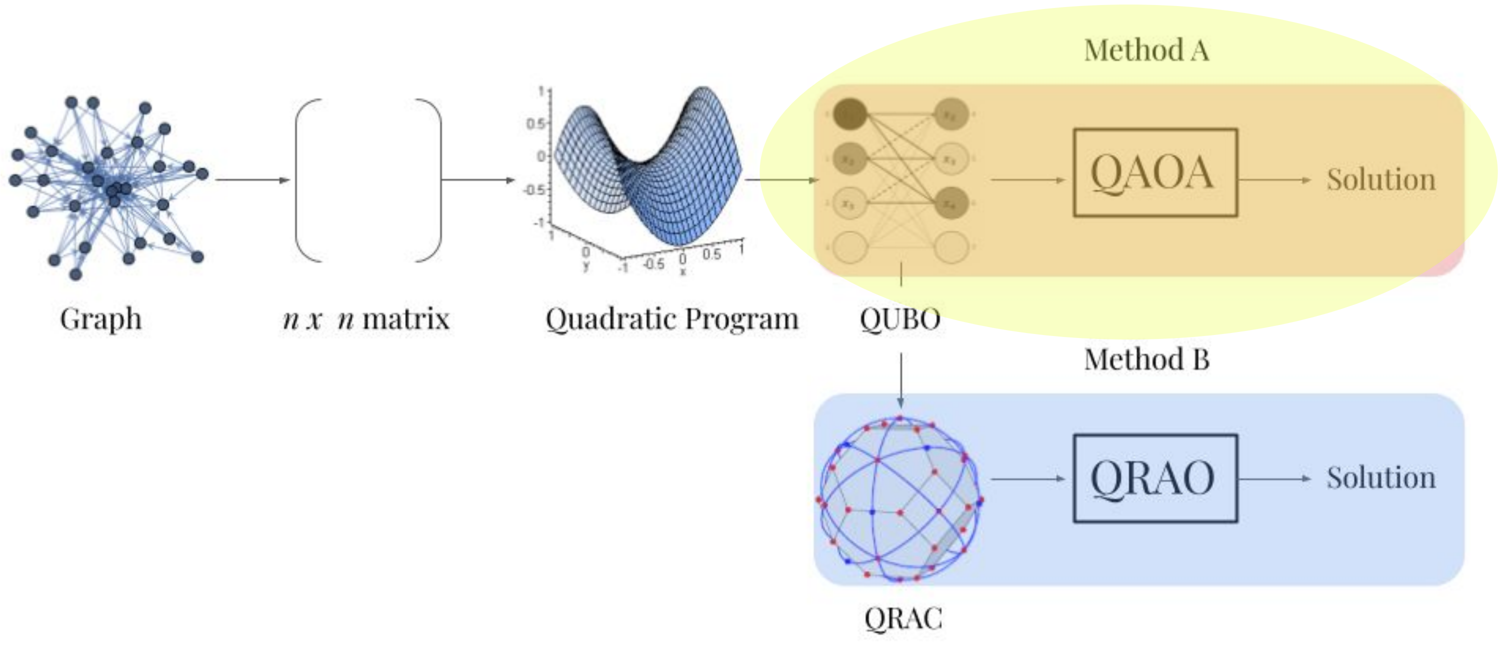


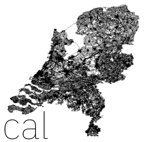
```
####  GRAPH PARTITIONING QUBO encoding  ####
def get_graph_part_docplex_model(num_nodes, adj_mat) -> Model:
    edges = adj_mat

    mod = Model("graph-partitioning")
    nodes = list(range(num_nodes))
    var = [mod.binary_var(name="x" + str(i)) for i in nodes]
    mod.minimize(
        mod.sum(
            edges[i, j] * (var[i] + var[j] - 2 * var[i] * var[j])
            for i in nodes
            for j in nodes
        )
    )
    mod.add_constraint(mod.sum([i for i in var]) == num_nodes // 2)
    return mod
```



Technical

Two Methods for Quantum Graph Partitioning:

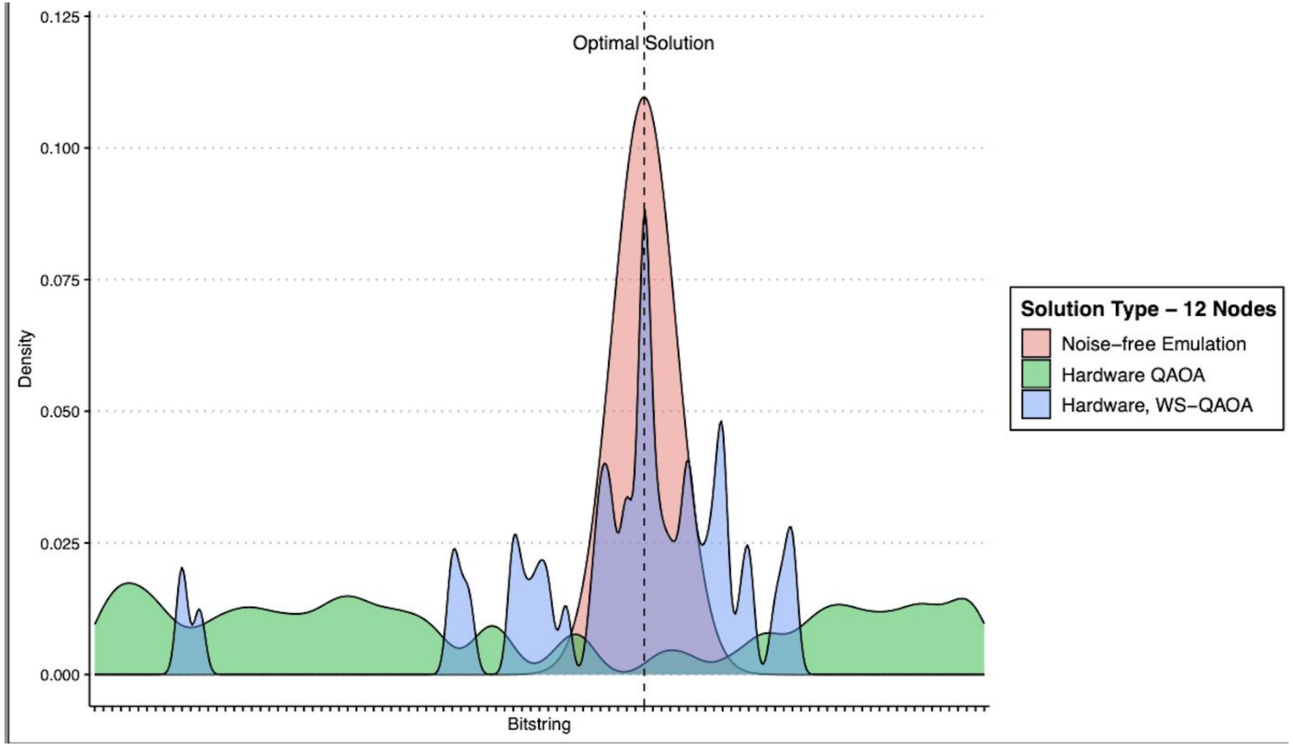


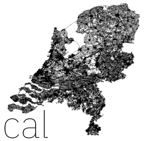


12 Nodes

Hardware Results :

Need Warm Start For Good Convergence



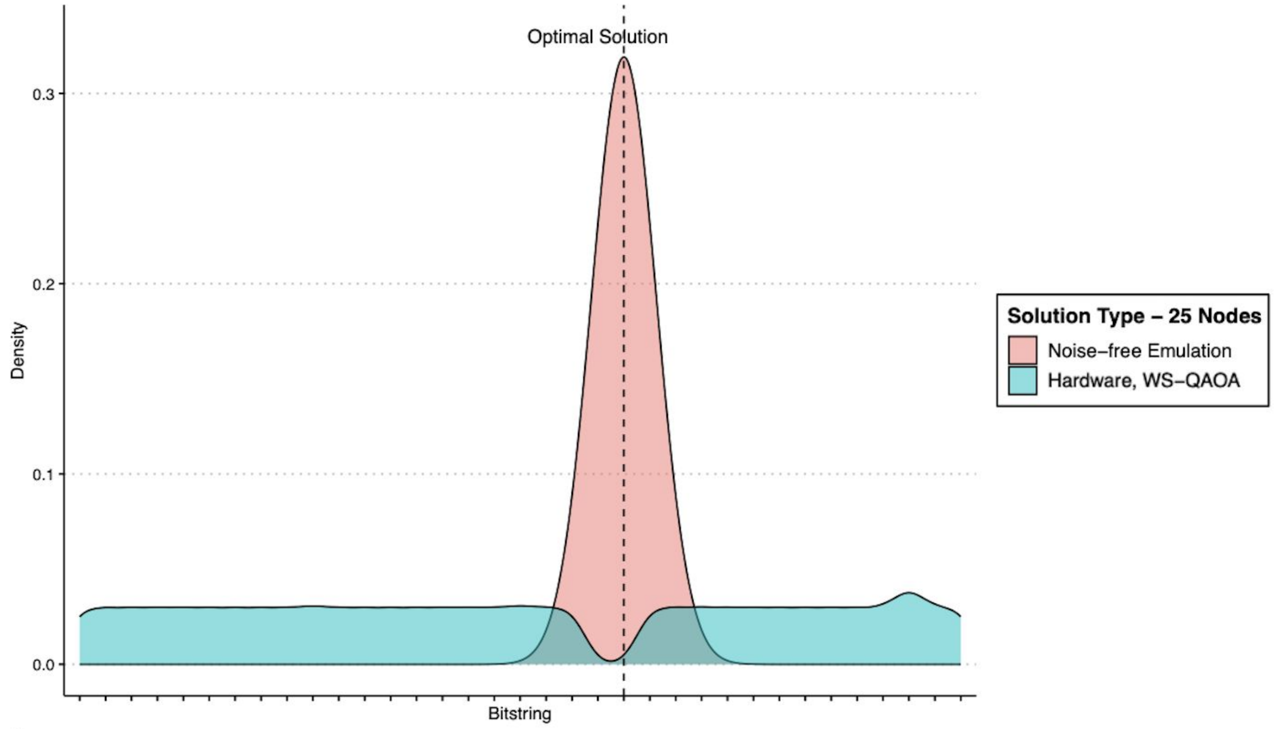


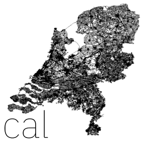
Technical

25 Nodes

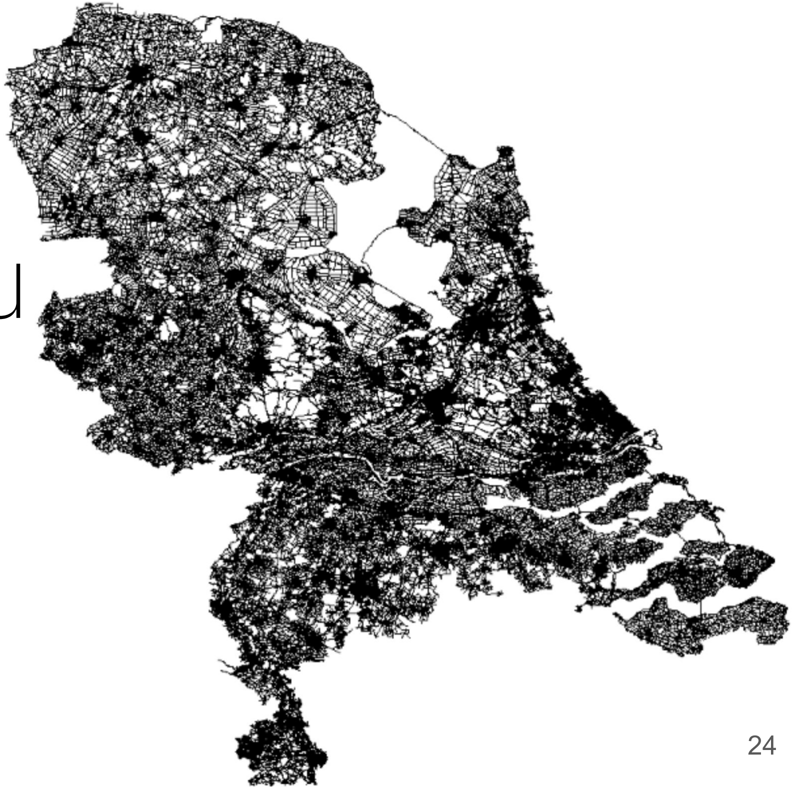
Hardware Results :

Need Warm Start For Good Convergence





Technical



Thank You

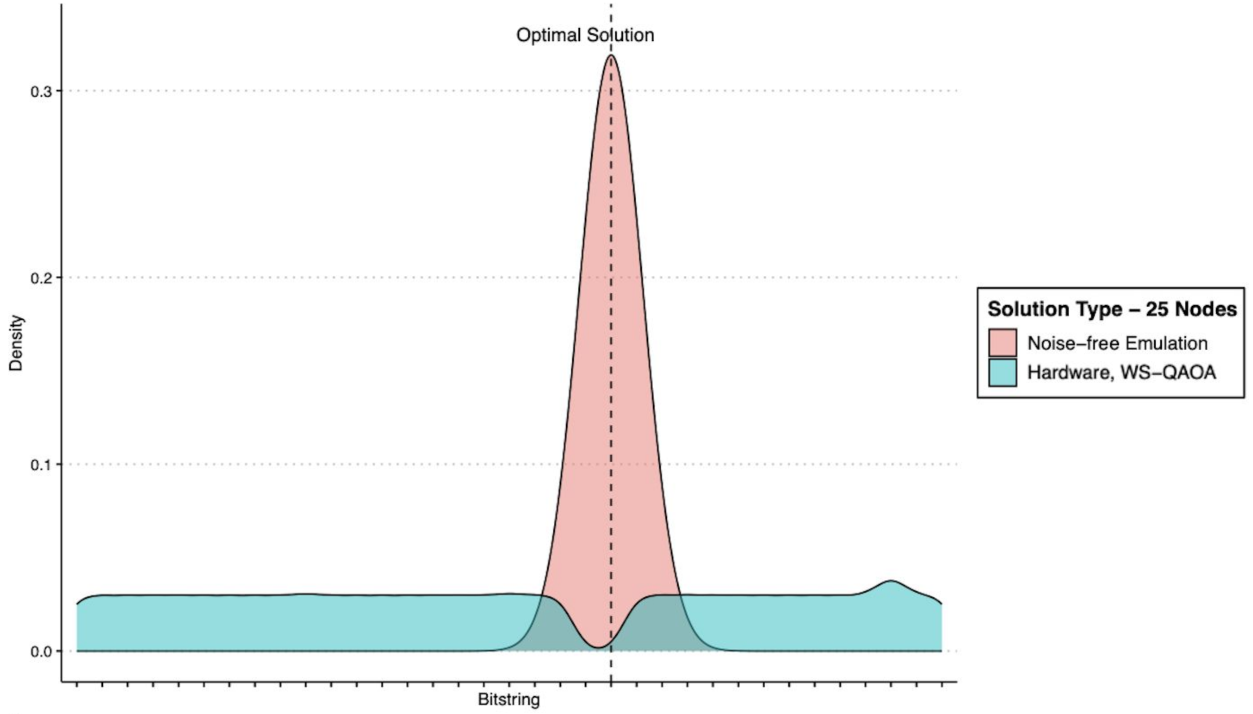
Questions?

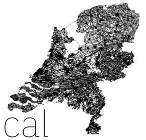
email: sophiakolak@cmu.edu
twitter: @KolakSophia
site: sophiakolak.github.io

25 Nodes

Hardware Results :

Need Warm Start For Good Convergence





Technical