Solving Vehicle Routing Problems using Quantum Computing

Paweł Gora

Faculty of Mathematics, Informatics and Mechanics
University of Warsaw

Quantum AI Foundation

Kraków Quantum Information Seminar
26.04.2022

Agenda

- ➔ **Definition of VRP and its variants**
- ➔ **Quantum annealing approaches to solve VRP**
  - ◆ **Explanation of quantum annealing**
  - ◆ **QUBO formulations / solvers**
  - ◆ **Experiments within the GLAD project**
- ➔ **QIntern / QResearch project**
  - ◆ **Comparison of QUBO formulations**
  - ◆ **VQE and other approaches**
- ➔ **Plans for the future research**

Vehicle Routing Problem and its variants

Input: we have a directed graph with non-negative costs assigned to edges.

Travelling salesman problem (TSP): we have N selected vertices in the graph (customers). What is the shortest possible route that visits all of them and returns to the origin vertex (depot)?

Vehicle Routing Problem (VRP): we have N selected vertices in the graph (customers). What is the optimal (with minimal cost) set of (up to M) routes which (in total) visit all the selected vertices and start and end in the depot?

Vehicle Routing Problem and its variants

Input: we have a directed graph with non-negative costs assigned to edges.

Travelling salesman problem (TSP): we have N selected vertices in the graph (customers). What is the shortest possible route that visits all of them and returns to the origin vertex (depot)?

Vehicle Routing Problem (VRP): we have N selected vertices in the graph (customers). What is the optimal (with minimal cost) set of (up to M) routes which (in total) visit all the selected vertices and start and end in the depot?

Capacitated Vehicle Routing Problem (CVRP): VRP with bounded capacities of vehicles.

Capacitated Vehicle Routing Problem with Time Windows (CVRPTW): VRP with bounded capacities of vehicles and with time windows.

(all these problems are NP-hard)

Solving combinatorial optimization problems

In **combinatorial optimization problems**, we search for the best of many possible combinations. Optimization problems include scheduling challenges, such as “Should I ship this package on this truck or the next one?” or “What is the most efficient route a traveling salesperson should take to visit different cities?”

Source: https://docs.dwavesys.com/docs/latest/c_gs_2.html

Solving combinatorial optimization problems

In **combinatorial optimization problems**, we search for the best of many possible combinations. Optimization problems include scheduling challenges, such as “Should I ship this package on this truck or the next one?” or “What is the most efficient route a traveling salesperson should take to visit different cities?”

Physics can help solve these sorts of problems because we can frame them as energy minimization problems. **A fundamental rule of physics is that everything tends to seek a minimum energy state.** Objects slide down hills; hot things cool down over time. This behavior is also true in the world of quantum physics. Quantum annealing simply uses quantum physics to find low-energy states of a problem and therefore the optimal or near-optimal combination of elements.

Simply: finding minimal “energy state” for a given optimization problem (encoded as entanglement of qubits).

Source: https://docs.dwavesys.com/docs/latest/c_gs_2.html

Adiabatic quantum computers

An **adiabatic process** - a process that does not involve the transfer of heat or matter into or out of a thermodynamic system. In an adiabatic process, energy is transferred to the surroundings only as work. (Source: https://en.wikipedia.org/wiki/Adiabatic_process)

Adiabatic quantum computers

An **adiabatic process** - a process that does not involve the transfer of heat or matter into or out of a thermodynamic system. In an adiabatic process, energy is transferred to the surroundings only as work. (Source: https://en.wikipedia.org/wiki/Adiabatic_process)

Adiabatic quantum computer:
"First, a (potentially complicated) Hamiltonian is found whose ground state describes the solution to the problem of interest. Next, a system with a simple Hamiltonian is prepared and initialized to the ground state. Finally, the **simple Hamiltonian is adiabatically evolved** to the desired complicated Hamiltonian. By the adiabatic theorem, the system remains in the ground state, so at the end the state of the system describes the solution to the problem." (Source: https://en.wikipedia.org/wiki/Quantum_annealing)

Adiabatic quantum computing has been shown to be **polynomially equivalent to conventional quantum computing in the circuit model**. ("Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation", D. Aharonov et al, https://arxiv.org/pdf/quant-ph/0405098.pdf)

Quantum annealing

Hamiltonian of the corresponding Ising model

$$H = (1 - s)H_i + sH_f$$

ground state easy to prepare

ground state is the solution of the problem

Adiabatic theorem:
if we change *s* **slow enough** the final state will be the solution of our problem

In practice: **s** can be changed by modifying a strength of the magnetic field.

Ising Model

The Ising model of ferromagnetism traditionally used in statistical mechanics. Variables are “spin up” (↑) and “spin down” (↓), states that correspond to +1 and −1 values (atomic “spins” or magnetic dipole moments). Relationships between the spins, represented by couplings, are correlations or anti-correlations. The objective function (Hamiltonian) expressed as an Ising model is as follows:

$$\mathrm{E}_{ising}(\boldsymbol{s}) = \sum_{i=1}^{N} h_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{i,j} s_i s_j$$

where the linear coefficients corresponding to qubit biases are h_i, and the quadratic coefficients corresponding to coupling strengths are $J_{i,j}$.

Finding the minimum of a nonplanar Ising formulation is NP-hard problem for classical computers.

Quantum Unconstrained Binary Optimization

Quadratic Unconstrained Binary Optimization (QUBO) problems are traditionally used in computer science. Variables are TRUE and FALSE, states that correspond to 1 and 0 values. A QUBO problem is defined using an upper-diagonal matrix Q, which is an N x N upper-triangular matrix of real weights, and x, a vector of binary variables, as minimizing the function:

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i<j} Q_{i,j} x_i x_j$$

where the diagonal terms $Q_{i,i}$ are the linear coefficients and the nonzero off-diagonal terms are the quadratic coefficients $Q_{i,j}$. This can be expressed more concisely as

$$\min_{x \in \{0,1\}^n} x^T Q x$$

Vehicle Routing Problem and its variants

Input: we have a directed graph with non-negative costs assigned to edges.

Travelling salesman problem (TSP): we have N selected vertices in the graph (customers). What is the shortest possible route that visits all of them and returns to the origin vertex (depot)?

Vehicle Routing Problem (VRP): we have N selected vertices in the graph (customers). What is the optimal (with minimal cost) set of (up to M) routes which (in total) visit all the selected vertices and start and end in the depot?

Vehicle Routing Problem and its variants

Input: we have a directed graph with non-negative costs assigned to edges.

Travelling salesman problem (TSP): we have N selected vertices in the graph (customers). What is the shortest possible route that visits all of them and returns to the origin vertex (depot)?

Vehicle Routing Problem (VRP): we have N selected vertices in the graph (customers). What is the optimal (with minimal cost) set of (up to M) routes which (in total) visit all the selected vertices and start and end in the depot?

Capacitated Vehicle Routing Problem (CVRP): VRP with bounded capacities of vehicles.

Capacitated Vehicle Routing Problem with Time Windows (CVRPTW): VRP with bounded capacities of vehicles and with time windows.

(all these problems are NP-hard)

Solving Vehicle Routing Problem - notation

T = {1, 2,... ,M} - identifiers of trucks/vehicles

V = {1, 2,... , N, N+1} - identifiers of vertices/nodes (N+1 - depot, 1,2,3,...,N - customers)

$C_{i,j}$ - cost of travel from node **i** to node **j** (for i,j $\in$ V), $C_{i,i} = 0$

$x_{i,j,k}$ = **1** if in a given setting the vehicle i visits the node j as k-th location on its route (**0** otherwise)

(**i** is from the set T, **j** from V, **k** is from {0,1,2,3,....,N+1})

Solving Vehicle Routing Problem - notation

T = {1, 2,... ,M} - identifiers of trucks/vehicles

V = {1, 2,... , N, N+1} - identifiers of vertices/nodes (N+1 - depot, 1,2,3,...,N - customers)

$C_{i,j}$ - cost of travel from node **i** to node **j** (for i,j ∈ V), $C_{i,i} = 0$

$x_{i,j,k}$ = **1** if in a given setting the vehicle i visits the node j as k-th location on its route (**0** otherwise)

(**i** is from the set T, **j** from V, **k** is from {0,1,2,3,....,N+1})

Observations:

1. For each vehicle i: $x_{i,N+1,0} = 1$, $x_{i,j,0} = 0$ for $j < N + 1$ (the depot is always the initial (0-th) location)
2. If $x_{i,N+1,L} = 1$ for some L, then for $W > L$: $x_{i,j,W} = 0$ for $j < N + 1$ and $x_{i,N+1,W} = 1$ (the depot is always the last location on each route and each car stays in the depot after reaching it)

Solving Vehicle Routing Problem - QUBO formulation

$$C = \sum_{m=1}^{M}\sum_{n=1}^{N} x_{m,n,1} C_{N+1,n} + \sum_{m=1}^{M}\sum_{n=1}^{N} x_{m,n,N} C_{n,N+1} + \\ + \sum_{m=1}^{M}\sum_{n=1}^{N-1}\sum_{i=1}^{N+1}\sum_{j=1}^{N+1} x_{m,i,n} x_{m,j,n+1} C_{i,j}$$

- The first component of the sum C is a sum of all costs of travels from the depot to the first visited node - the first section of each car's route.
- The second is a cost of the last section of a route (to depot) in a special case when a single car serves all N orders (only in such a case the component can be greater than 0).
- The last part is the cost of all other sections of routes.

Solving Vehicle Routing Problem - QUBO formulation

Let's consider a binary function:

$$A(y_1, y_2, \ldots, y_n) = (y_1 + y_2 + \ldots + y_n - 1)^2 - 1$$

where $y_i \in \{0,1\}$ for $i \in \{1,\ldots, n\}$. The minimum value of $A(y_1, y_2, \ldots, y_n)$ is equal to -1 and this value can be achieved only if exactly one of $y_1, y_2, \ldots, y_n$ is equal to 1.

Solving Vehicle Routing Problem - QUBO formulation

To assure that each delivery is served by exactly one vehicle and exactly once and that each vehicle is in exactly one place at a given time, the following term should be included in our QUBO formulation:

$$Q = \sum_{k=1}^{N} A(x_{1,k,1}, x_{2,k,1}, \ldots, x_{1,k,2}, \ldots, x_{M,k,N}) + \\ + \sum_{m=1}^{M} \sum_{n=1}^{N} A(x_{m,1,n}, x_{m,2,n}, \ldots, x_{m,N+1,n})$$

Solving Vehicle Routing Problem - QUBO formulation

Full QUBO Solver (FQS)

By definition of VRP, QUBO representation of this optimization problem is

$$QUBO_{VRP} = A_1 \cdot C + A_2 \cdot Q$$

for some constants A_1 and A_2, which should be properly set to ensure that the solution found by quantum annealer minimizes the cost C and satisfies the aforementioned constraints (Q).

Solving Vehicle Routing Problem

Average Partition Solver (APS)

We decrease the number of variables for each vehicle by assuming that every vehicle serves approximately the same number of orders - up to A+L deliveries, where A is the total number of orders divided by the number of vehicles (N/M) and L is a parameter (called “limit radius”), which controls the number of orders (in practice, we usually want our fleet to be evenly loaded).

The number of variables is lower which simplifies computations.

Solving Capacitated Vehicle Routing Problem

DBSCAN Solver (DBSS)

Hybrid algorithm which combines quantum approach with a classical algorithm (Recursive DBSCAN).

DBSS uses recursive DBSCAN as a clustering algorithm with limited size of clusters. Then, TSP is solved by FQS separately (we can just assume the number of vehicles equal to 1).

If the number of clusters is equal to (or lower than) the number of vehicles, the answer is known immediately. Otherwise, the solver runs recursively considering clusters as deliveries, so that each cluster contains orders which in the final result are served one after another without leaving the cluster.

We also concluded that by limiting the total sum of weights of deliveries in clusters, this algorithm can solve CVRP if all capacities of vehicles are equal.

Solving Capacitated Vehicle Routing Problem

Solution Partitioning Solver (SPS)

This algorithm divides TSP solution found by another algorithm (e.g., FQS) into consecutive intervals, which are the solution for CVRP.

Let $\mathbf{d_1}$, $\mathbf{d_2}$, …, $\mathbf{d_N}$ be the TSP solution for **N** orders, let $\mathbf{P_v}$ be a capacity of the vehicle **v**, let $\mathbf{w_{i,j}}$ be the sum of weights of orders $\mathbf{d_i}$, $\mathbf{d_i+1}$, $\mathbf{d_i+2}$, ... , $\mathbf{d_j}$ (in the order corresponding to TSP solution) and let $\mathbf{cost_{i,j}}$ be the total cost of serving only orders $\mathbf{d_i}$, $\mathbf{d_i+1}$, ... , $\mathbf{d_j}$. Also, let $\mathbf{dp_{i;S}}$ be the cost of the best solution for orders $\mathbf{d_1}$, $\mathbf{d_2}$, $\mathbf{d_3}$, …, $\mathbf{d_i}$ and for the set of vehicles **S**. Now, the dynamic programming formula for solving CVRP is given by:

$$dp_{i,S} = \min_{v \in S, 0 \leq j \leq i, w_{j+1,i} \leq P_v} \{dp_{j,S \setminus \{v\}} + cost_{j+1,i}\}$$

where $cost_{i,j} = 0$ and $w_{i,j} = 0$ for $i > j$.

Solving Capacitated Vehicle Routing Problem

Solution Partitioning Solver (SPS)

To speed up computations, we can apply a heuristic:

1. Instead of set S of vehicles, consider a sequence v_1, v_2, ..., v_M of vehicles and assume that we attach them to deliveries in such an order.
2. Now, our dynamic programming formula is given by:

$$dp_{i,\{v_1,\ldots,v_k\}} = \min_{0 \le j \le i, w_{j+1,i} \le P_{v_k}} \{dp_{j,\{v_1,\ldots,v_{k-1}\}} + cost_{j+1,i}\}$$

3. To count this dynamic effectively, we can observe that:

$$\forall_{j<i, 1 \le k \le M} (dp_{j-1,v_k} + cost_{j,i}) - (dp_{j-1,v_k} + cost_{j,i-1}) = C_{i-1,i} + C_{i,N+1} - C_{i-1,N+1}$$

We can now select some random permutations of vehicles and perform dynamic programming for each of them.

Experiments - setup

Goal: compare results of different algorithms (quantum and classical) on several datasets.

For quantum/hybrid algorithms, we ran experiments using **D-Wave's Leap platform** and 2 solvers: **qbsolv** (on QPU or CPU) and **hybrid solver** (on QPU and CPU at the same time).

In case of classical algorithms, we tested Simulated Annealing (SA), Bee Algorithm (BEE), Evolutionary Annealing (EA), DBSCAN with simulated annealing (DBSA).

Experiments - datasets

We prepared several datasets:

- Christofides1979 - a standard benchmark dataset for CVRP, well-known and frequently studied by the scientific community (14 tests with different number of vehicles, capacities and number of orders).
- A dataset built by us based on a realistic road network of Belgium, acquired from the OpenStreetMap service (51 tests).

Experiments - datasets

Christofides1979 - a standard benchmark dataset for CVRP, well-known and frequently studied by the scientific community (14 tests with different number of vehicles, capacities and number of orders).

Test name	Nr of vehicles	Capacity	Nr of orders
CMT11	7	200	120
CMT12	10	200	100
CMT13	11	200	120
CMT14	11	200	100
CMT3	8	200	100
CMT6	6	160	50
CMT7	11	140	75
CMT8	9	200	100
CMT9	14	200	150

Parameters of instances of Christofides1979 used in our experiments.

Experiments - datasets

A dataset built by us based on a realistic road network of Belgium, acquired from the OpenStreetMap service (51 tests): we considered different numbers of orders (from 1 to 200) and different locations of orders and depots.

Test	Number of orders
small-0	2
small-1	2
small-2	2
small-3	1
small-4	2
small-5	5
small-6	6
small-7	5
small-8	4
small-9	6
medium-0	20
medium-1	26
medium-2	27
medium-3	24

Examples of test cases from realistic road networks

Results of experiments

Test	vehicles	FQS CPU	FQS QPU	FQS Hybrid	APS CPU	APS Hybrid
small-0	1,2	11286	11286	11286	11286	11286
small-1	1	10643	10643	10643	10643	10643
	2	10643	10643	10643	12379	12379
	3	10643	-	10643	-	-
small-2	1	21311	21311	21311	21311	21311
	2	21311	-	21311	24508	24508
	3	22192	-	21311	-	-
small-3	1	18044	18044	18044	18044	18044
	2	20819	-	18033	22193	22193
	3	22843	-	18033	-	-
small-4	1	15424	15424	15424	15424	15424
	2	17364	-	15424	19472	19472
	3	17364	-	15424	-	-
small-5	1	10906	10906	10906	10906	10906
	2	11676	-	10906	13480	13480
	3	11754	-	10906	-	-
small-6	1	20859	20859	20859	20859	20859
	2	26735	-	20859	26735	26735
	3	27110	-	20859	-	-

Results for different quantum and hybrid algorithms on some small datasets

Results of experiments

Test	vehicles	FQS CPU	FQS QPU	FQS Hybrid	APS CPU	APS Hybrid	DBSS CPU
medium-0	1	20774	-	21775	20774	21775	24583
	2	36966	-	29879	25737	25217	27994
	3				28226	27237	34185
medium-1	1	29868	-	29423	29868	29423	27606
	2	50639	-	39485	30820	31129	31346
	3	-	-	-	33376	32018	32588
medium-2	1	37045	-	35208	37045	35208	29442
	2	55579	-	36511	33235	33163	32947
	3	-	-	-	36600	32569	34480
medium-3	1	30206	-	29422	30206	29422	31092
	2	51787	-	35774	31428	30273	33790
	3	-	-	-	35994	33627	33712
medium-4	1	21257	-	20762	21257	20762	21435
	2	34379	-	25470	22410	22722	22885
	3	-	-	-	23599	22176	25446
medium-5	1	23013	-	21642	23013	21462	21737
	2	36149	-	22041	22775	23076	23403
	3	-	-	-	24899	22386	24336
medium-6	1	23804	-	24664	23804	23804	23926
	2	35826	-	24490	24265	25178	25510
	3	-	-	-	27032	23364	25122

Results for different quantum and hybrid algorithms on some medium datasets

Results of experiments

	vehicles	APS CPU	DBSS CPU
big-0	1	80084	71594
	2	97286	71051
big-1	1	157660	146828
	2	206782	149200
big-2	1	168646	154105
big-3	1	85873	62236
big-4	1	156411	129279

Comparison of results for Average Partition Solver and DBSCAN Solver on big test cases.

Results of experiments

	vehicles	capacity	SPS (CPU)	DBSS (CPU)
big-0	2	100	70928	73508
	2	85	72295	73189
	2	80	75150	Not valid
	3	100	71320	76717
	3	70	71251	78012
	3	55	Not valid	76807
	5	100	71740	Not valid
	5	50	78726	91066
	5	40	85976	Not valid
big-1	2	100	150608	158631
	2	80	150608	152946
	2	65	150804	156188
	3	100	151525	153673
	3	60	153190	152854
	3	45	164055	Not valid
	5	100	151930	168789
	5	40	156242	165271
	5	30	174519	176935

Comparison of DBSCAN Solver and Solution Partitioning Solver (SPS) run on CPU on big test cases with various capacities.

Results of experiments

	type	deliveries	SPS	Simul. Ann.	Bee	Evolution
clustered1-1	average	57	69850	66379	60876	48923
	best	57	69080	52119	56358	48152
clustered1-2	average	55	77173	74341	81438	54719
	best	55	75530	59947	68772	53490
group1-1	average	42	158919	156217	153495	137989
	best	42	155388	146526	142774	135593
group1-2	average	54	171732	145380	145325	137626
	best	54	165043	141065	140947	136307
range-6-1	average	47	71670	68003	67234	59937
	best	47	68459	62312	64404	59827
range-6-2	average	50	80490	84380	83915	73651
	best	50	79640	79574	85917	73051
range-8-12-1	average	50	142008	146553	142835	129069
	best	50	140170	136369	127372	126555
range-8-12-2	average	50	146798	137628	145332	129048
	best	50	143598	135493	136776	128803
range-8-12-3	average	46	105544	105051	98366	92792
	best	46	101577	99004	94423	91921
range-8-12-4	average	51	147993	143309	148900	128316
	best	51	145559	140088	128575	124405
range-8-12-5	average	50	146719	143516	145685	134162
	best	50	143993	139784	139796	133245
range-8-12-6	average	50	146984	148194	150121	136326
	best	50	141467	138781	139400	134692
range-5-1	average	50	81728	68900	69052	67896
	best	50	72527	67984	68022	67691
range-5-2	average	50	81759	69342	68564	67981
	best	50	76868	67958	67780	67716

Results of Solution Partitioning Solver compared with results for classical algorithms run on artificially generated test cases.

Results of experiments

Test name	SPS	SA	BEE	EA	DBSA
CMT11	25.54	23.62	36.18	16.52	19.94
CMT12	26.84	53.06	20.24	20.68	21.37
CMT13	25.97	86.72	34.66	35.05	19.44
CMT14	26.83	52.52	20.23	20.23	22.8
CMT3	25.13	48.3	28.38	28.82	-
CMT6	17.58	48.3	15.42	28.82	15.82
CMT7	29.42	41.4	27.89	31.68	23.18
CMT8	26.5	51.16	26.67	28.09	19.4
CMT9	34.14	76.34	44.25	42.81	-

Comparison of results achieved by Solution Partitioning Solver (SPS) and classical algorithms (SA - simulated annealing, BEE- Bee algorithm, EA - evolutionary annealing, DBSA - DBSCAN with simulated annealing) on a benchmark dataset Christofides79.

Conclusions

- It doesn’t make sense to run experiments on QPU for large test cases
- SPS (with qbsolv run on CPU) gives the best results among “quantum” algorithms
- We compared SPS with some classical metaheuristics for CVRP well-known in the scientific literature (e.g., simulated annealing (SA), bee algorithm (BEE), evolutionary annealing (EA), DBSCAN with simulated annealing (DBSA)) - it usually gives a bit worse (but sometimes better) results.
- Github repository: https://github.com/dwave-examples/D-Wave-VRP

Future plans

- Develop our algorithms further in order to solve CVRPTW (and other variants of VRP).
- Investigate for which scenarios the hybrid algorithms give the best results comparing to classical algorithms (for which scenarios we may expect some advantages?).
- Investigating algorithms to solve CVRPTW using “circuit-based” quantum computers.

Acknowledgement

- The presented research is based on the article **“New hybrid quantum annealing algorithms for solving Vehicle Routing Problem”** and was presented at the **ICCS 2020** conference.

- **Michał Borowski, Paweł Gora, Katarzyna Karnas, Mateusz Błajda, Krystian Król, Artur Matyjasek, Damian Burczyk, Miron Szewczyk, Michał Kutwin**, (2020) New Hybrid Quantum Annealing Algorithms for Solving Vehicle Routing Problem. In: Krzhizhanovskaya V. et al. (eds) Computational Science – ICCS 2020. ICCS 2020. Lecture Notes in Computer Science, vol 12142. Springer, Cham. https://doi.org/10.1007/978-3-030-50433-5_42

- https://link.springer.com/chapter/10.1007/978-3-030-50433-5_42

Acknowledgement

The presented research was carried out within the frame of the project “Green LAst-mile Delivery” (GLAD) realized at the University of Warsaw with the project partners: Colruyt Group, University of Cambridge and Technion. The project is supported by EIT Food, which is a Knowledge and Innovation Community (KIC) established by the European Institute for Innovation & Technology (EIT), an independent EU body set up in 2008 to drive innovation and entrepreneurship across Europe.

EIT Food is supported by the EIT
a body of the European Union

Route Activation Solver (RAS)

- Route Activation Solver is another such exact solver that allows us to encode information about the edges connecting two nodes and the order in which they are visited by the vehicles to formulate a Quadratically Constrained Quadratic Program (QCQP) model of the Vehicle Routing Problem.
- **Variables:** $x_{i,j}\ (binary), y_{i,k}\ (binary)\ and\ t_i\ (integer)$

$$x_{i,j} = \begin{cases} 1, & iff\ the\ edge\ (i,\ j)\ is\ traversed\ by\ a\ vehicle \\ 0, & otherwise \end{cases}$$

$$y_{i,k} = \begin{cases} 1, & iff\ the\ i^{th}\ node\ is\ visited\ by\ the\ k^{th}\ vehicle \\ 0, & otherwise \end{cases}$$

$$t_i \in \{1 \dots N\} \quad time\ taken\ for\ any\ vehicle\ to\ reach\ the\ i^{th}\ node$$

- **Objective Function:**

$$\min \sum_{i=0}^{N} \sum_{j=0, i \neq j}^{N} C_{i,j} x_{i,j}$$

- **Constraints:**

1. Each client node must have exactly one edge directed away from it and exactly one edge directed towards it.

$$\sum_{j=0, j \neq i}^{N} x_{i,j} = 1 \quad \forall\, i \qquad\qquad \sum_{i=0, i \neq j}^{N} x_{i,j} = 1 \quad \forall\, j$$

Source: “Heuristic QUBO Formulations for solving the Vehicle Routing Problem using Quantum Annealing”, Shantom Borah, Asish Kumar Mandoi, Avneesh Verma

Cluster-Route Heuristics

- Cluster First, Route Second

- These methods can be summed up as a combination of two phases:
- **Clustering Phase**: For the first phase, we split the client node set into k clusters. Clustering may be done classically, or using a quantum computer.
- **Routing Phase**: For the second phase, each cluster is assigned to a separate vehicle. And the corresponding routes are then found via solving the TSP. The TSP solution is done on a quantum computer via quantum annealing
- Examples: DBSCAN Solver, Max-Cut Partitioning Solver

Source: “Heuristic QUBO Formulations for solving the Vehicle Routing Problem using Quantum Annealing”, Shantom Borah, Asish Kumar Mandoi, Avneesh Verma

Max-Cut Partitioning Solver (MPS)

- Cluster via Max-Cut, Route via TSP

- **Phase 1**: Clustering via Max-k-Cut formulation.
- Generalization of the Max Cut problem to k clusters.

$$x_{i,j} = \begin{cases} 1, & \text{if node } i \text{ is assigned to the } j^{th} \text{ cluster} \\ 0, & \text{otherwise} \end{cases}$$

- **Objective Function**:

$$\min \sum_{i,j,k} C_{ij} x_{ik} x_{jk}$$

- **Constraints**:

$$\sum_{j} x_{ij} = 1 \ \forall \ i$$

Source: “Heuristic QUBO Formulations for solving the Vehicle Routing Problem using Quantum Annealing”, Shantom Borah, Asish Kumar Mandoi, Avneesh Verma

Max-Cut Partitioning Solver (MPS)

- Cluster via Max-Cut, Route via TSP

- **Phase 2**: Solving the Travelling Salesman Problem.

$$x_{i,j} = \begin{cases} 1, & \text{if the salesman is at node } i \text{ at the } j^{th} \text{ time-step} \\ 0, & \text{otherwise} \end{cases}$$

- **Objective Function**:

$$\min \sum_{n=1}^{N} C_{0,n} x_{n,1} + \sum_{n=1}^{N} C_{n,0} x_{n,N} + \sum_{n=1}^{N-1} \sum_{i=0}^{N} \sum_{j=0}^{N} C_{i,j} x_{i,n} x_{j,n+1}$$

- **Constraints**:

$$\sum_{j=1}^{N} x_{i,j} = 1 \; \forall \, i \qquad \sum_{i=0}^{N} x_{i,j} = 1 \; \forall \, j$$

Source: “Heuristic QUBO Formulations for solving the Vehicle Routing Problem using Quantum Annealing”, Shantom Borah, Asish Kumar Mandoi, Avneesh Verma

The GPS Solver

Saul Gonzalez-Bermejo, Guillermo Alonso-Linaje, and Parfait Atchade-Adelomou

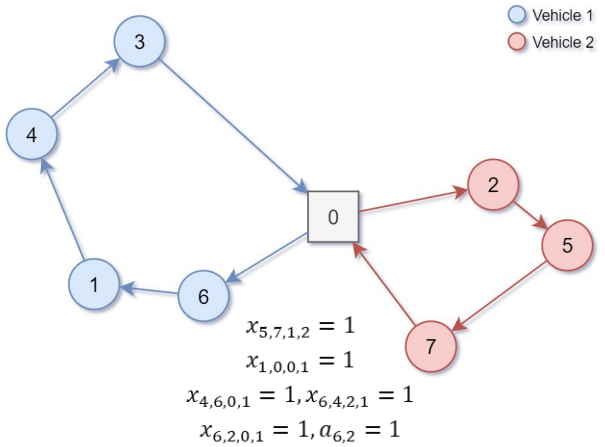
- It becomes important to finding a good QUBO formulation that minimizes the number of variables to be used especially in the current NISQ era. The GPS Solver was primarily motivated by this goal to find a suitable exact formulation of TSP. The authors of the paper [2] who proposed this formulation have extended it to a VRP formulation.
- Rather than keeping track of the exact time step at which a vehicle visits a node, the binary variables in this model contain information about the order of traversal of the nodes by a single vehicle as well as the order in which different vehicles traverse the nodes.
- **Variables:** $x_{i,j,r,k}\ (binary),\ \ a_{i,j}\ (binary)$

$$x_{i,j,0,k} = \begin{cases} 1, & \text{iff city i is reached before city j, and the edge (i, j) is not travelled} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{i,j,1,k} = \begin{cases} 1, & \text{iff vehicle k traverses the edge (i, j) and hence i is reached before j} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{i,j,2,k} = \begin{cases} 1, & \text{iff city j is reached before city i, and the edge (j, i) is not travelled} \\ 0, & \text{otherwise} \end{cases}$$

$$a_{i,j} = \begin{cases} 1, & \text{iff city i is reached before city j considering every vehicle} \\ 0, & \text{otherwise} \end{cases}$$



Source: “Heuristic QUBO Formulations for solving the Vehicle Routing Problem using Quantum Annealing”, Shantom Borah, Asish Kumar Mandoi, Avneesh Verma

The GPS Solver

Saul Gonzalez-Bermejo, Guillermo Alonso-Linaje, and Parfait Atchade-Adelomou

- **Objective Function**:

$$\min \sum_{k=1}^{M} \sum_{i=0,\, i\neq j}^{N} \sum_{j=0}^{N} C_{i,j} x_{i,j,1,k}$$

- **Constraints**:

1. For each i, j, q, one and only one of the possibilities must be met for r.

$$\sum_{r=0}^{2} x_{i,j,r,k} = 1 \quad \forall\, i, j \neq i, k$$

2. Each vehicle must

a) leave the depot in the beginning and reach the depot in the end.

$$\sum_{j=1}^{N} x_{0,j,1,k} = 1 \quad \forall\, k \qquad \sum_{i=1}^{N} x_{i,0,1,k} = 1 \quad \forall\, k$$

b) arrive at each city exactly once and leave each city exactly once.

$$\sum_{k=1}^{M} \sum_{i=0, i\neq j}^{N} x_{i,j,1,k} = 1 \quad \forall\, j \in \{1 \dots N\} \qquad \sum_{k=1}^{M} \sum_{j=0, j\neq i}^{N} x_{i,j,1,k} = 1 \quad \forall\, i \in \{1 \dots N\}$$

Source: “Heuristic QUBO Formulations for solving the Vehicle Routing Problem using Quantum Annealing”, Shantom Borah, Asish Kumar Mandoi, Avneesh Verma

The GPS Solver

Saul Gonzalez-Bermejo, Guillermo Alonso-Linaje, and Parfait Atchade-Adelomou

3. The i^{th} city can either be reached earlier or later than the j^{th} city.

$$\sum_{k=1}^{M} x_{i,j,0,k} + x_{i,j,1,k} = a_{i,j} \cdot M \quad \forall\, i \in \{1 \dots N\}, j \in \{1 \dots N\}, i \neq j$$

$$a_{i,j} + a_{j,i} = 1 \quad \forall\, i, j \in \{1 \dots N\}, i \neq j$$

4. If a vehicle k arrives in the q^{th} city, then it must leave the q^{th} city.

$$x_{p,q,1,k} \cdot \left(1 - \sum_{r=0, r\neq q}^{N} x_{q,r,1,k}\right) = 0 \quad \forall\, p \in \{0 \dots N\}, q \in \{1 \dots N\}, p \neq q, k \in \{1 \dots M\}$$

5. A vehicle can either arrive at the i^{th} city before the j^{th} city or vice-versa.

$$x_{i,j,0,k} + x_{i,j,1,k} + x_{j,i,0,k} + x_{j,i,1,k} = 1 \quad \forall\, i, j \in \{1 \dots N\}, i \neq j, k \in \{1 \dots M\}$$

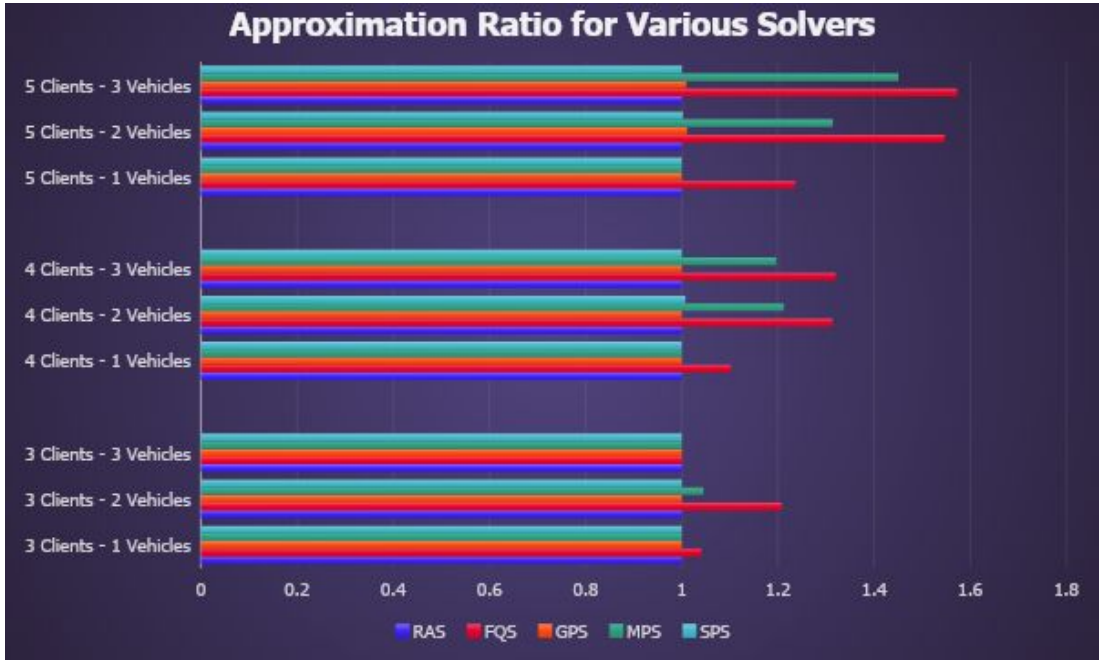
44

6. If the i^{th} city is reached before the j^{th} city, and the j^{th} city is reached before the k^{th} city, then the i^{th} city must be reached before the k^{th} city.

$$a_{i,j} \cdot a_{j,k} - a_{i,j} \cdot a_{i,k} - a_{j,k} \cdot a_{i,k} + a_{i,k} \cdot a_{i,k} = 0 \quad \forall\, i, j, k \in \{1 \dots N\}, i \neq j, j \neq k, k \neq i$$

Source: “Heuristic QUBO Formulations for solving the Vehicle Routing Problem using Quantum Annealing”, Shantom Borah, Asish Kumar Mandoi, Avneesh Verma

Numerical Experiments
Characterizing Solver Performance



Source: “Heuristic QUBO Formulations for solving the Vehicle Routing Problem using Quantum Annealing”, Shantom Borah, Asish Kumar Mandoi, Avneesh Verma

Numerical Experiments
Characterizing Solver Performance

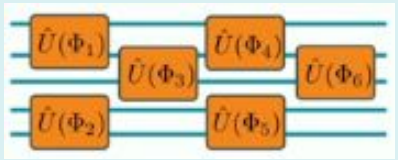
- ***Qubit Complexity***: *Number of logical qubits required as a function of the number of client nodes* ***N****.*

Solver Name	Best Case Complexity	Worst Case Complexity
FQS	$O(N^2)$	$O(N^3)$
RAS	$O(N^2)$	$O(N^2)$
MPS	$O(N^{3/2})$	$O(N^2)$
SPS	$O(N^2)$	$O(N^2)$
GPS	$O(N^2)$	$O(N^3)$

Source: “Heuristic QUBO Formulations for solving the Vehicle Routing Problem using Quantum Annealing”, Shantom Borah, Asish Kumar Mandoi, Avneesh Verma

VQA components

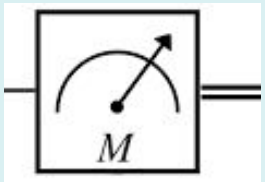
Ansatz



Parameterized quantum circuit

Measurement scheme

$Z, X, or\ Y$



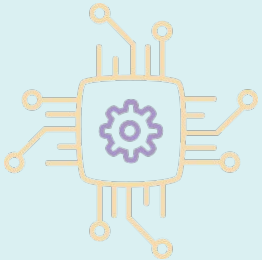
Specifies the operators to be measured

Objective function

$$C(\theta)$$

Computed from the measurement results

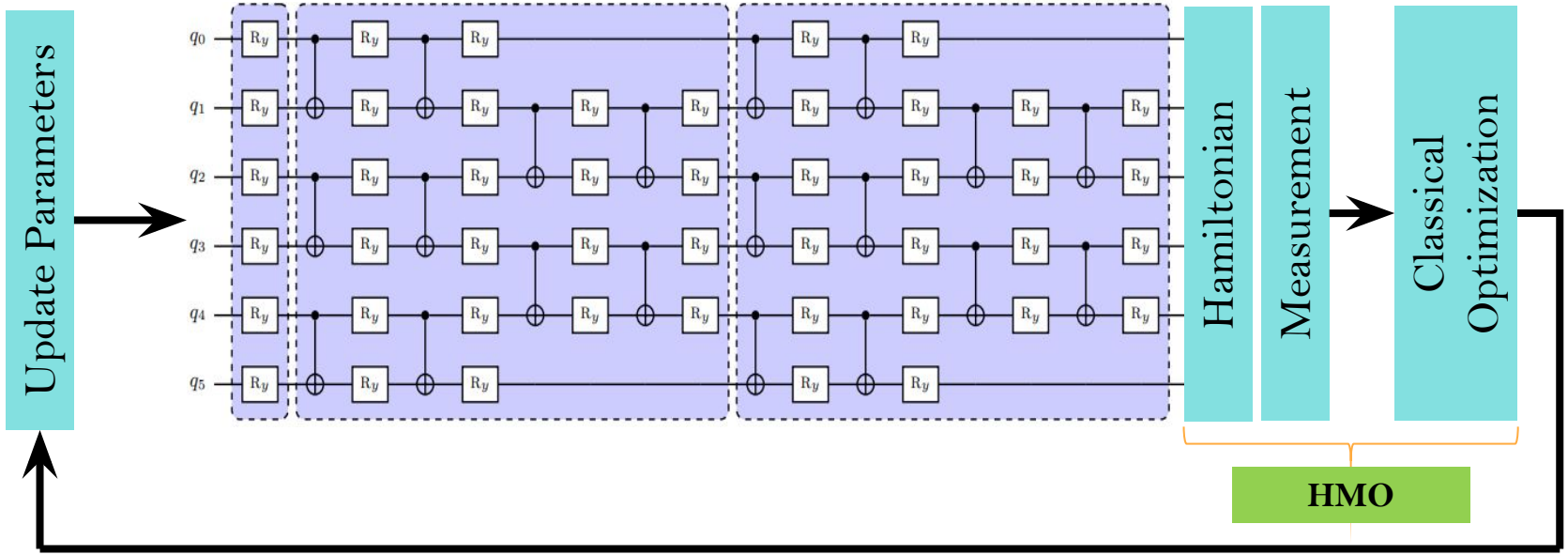
Classical optimizer



Adjust the **Ansatz** parameters to **minimize** the **objective function**

Source: “Solving Vehicle Routing Problem using Variational Quantum Eigensolver and its variants”, Walid El Maouaki, Atharva Vidwans

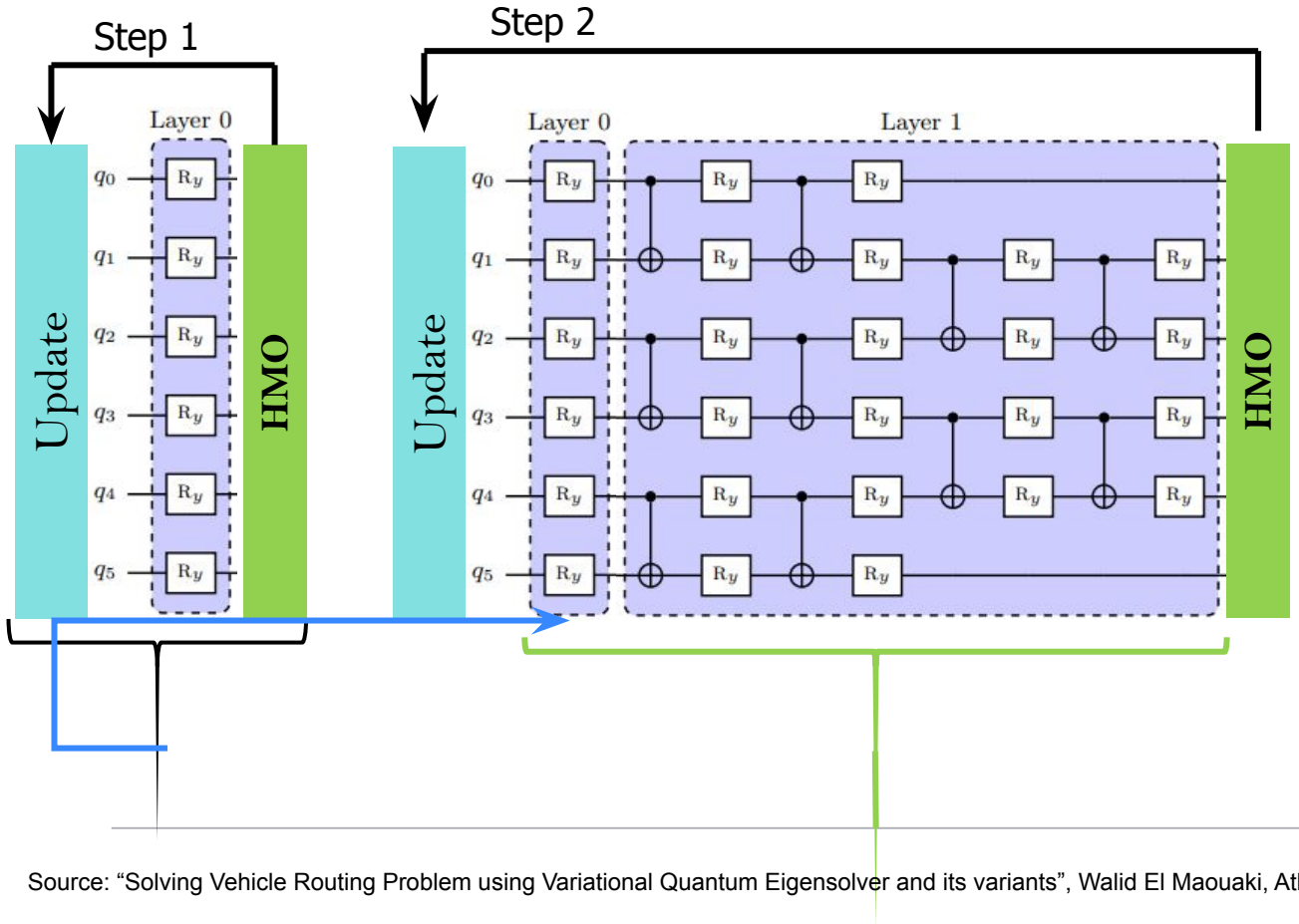
Variational Quantum Eigensolver (VQE)



- the cost function is defined as the expectation value of the Hamiltonian computed in the trial state

Source: “Solving Vehicle Routing Problem using Variational Quantum Eigensolver and its variants”, Walid El Maouaki, Atharva Vidwans

Layer-VQE



- We increment the ansatz before reaching convergence.
- Initializing with small random numbers may be useful to avoid local minima and speed up the optimization in general.
- We initialize the added layer such that it evaluates to identity to ensure that the quality of the solution does not decrease at each step.

https://arxiv.org/abs/2102.05566

Source: “Solving Vehicle Routing Problem using Variational Quantum Eigensolver and its variants”, Walid El Maouaki, Atharva Vidwans

Why choose L-VQE over VQE for solving VRP?

- Unlike VQE, which has an ansatz fixed upfront, L-VQE starts from a simple and shallow hardware efficient ansatz with a small number of parameterized gates and then adds layers to the ansatz systematically.
- This strategy allows us to make the ansatz more expressive and reduces the optimization overhead.
- Furthermore, VQE is likely to stuck in local minima, as opposed to LVQE which higher tendency to find the optimal solution for combinatorial optimization problems.

Source: “Solving Vehicle Routing Problem using Variational Quantum Eigensolver and its variants”, Walid El Maouaki, Atharva Vidwans

L-VQE vs VQE

- Initial experiments show that L-VQE can outperform VQE
- We are also conducting experiments with F-VQE and combined approach (LF-VQE)

Source: “Solving Vehicle Routing Problem using Variational Quantum Eigensolver and its variants”, Walid El Maouaki, Atharva Vidwans

Other approaches

- “Unconstrained Binary Models of the Travelling Salesman Problem Variants for Quantum Optimization”, Özlem Salehi, Adam Glos, Jarosław Miszczak
- “Solving the Traveling Salesperson Problem using TensorFlow Quantum”, Justyna Zawalska
- “QROSS: QUBO Relaxation Parameter Optimisation via Learning Solver Surrogates”, Tian Huang, Siong Thye Goh, Sabrish Gopalakrishnan, Tao Luo, Qianxiao Li, Hoong Chuin Lau
- …

Source: “Solving Vehicle Routing Problem using Variational Quantum Eigensolver and its variants”, Walid El Maouaki, Atharva Vidwans

Future work

- Scalability - graph clustering / coarsening
- CVRPTW
- Preparation of a book (Packt)

Source: “Solving Vehicle Routing Problem using Variational Quantum Eigensolver and its variants”, Walid El Maouaki, Atharva Vidwans

Thank you for your attention!

- Questions?
  - p.gora@mimuw.edu.pl
  - warsaw.quantum@gmail.com
- www:
  - http://www.mimuw.edu.pl/~pawelg
  - http://www.qaif.org

"Logic can get you from A to B, imagination will take you everywhere"

A. Einstein

*"The sky is **NOT** the limit"*