OPTIMIZATION : QUBO

uses QAOA (Quantum Approximate Optimization algorithm) and QA (Quantum Annealing) to solve a combinatorial optimization problem.

First step: Translate combinatorial optimization probs to QUBO (Quadratic unconstrained Binary Optimization)

Knapsack problem:

```
item_values = {"football": 8, "Laptop": 47,
               "camera": 10, "BOOKS": 5,
               "Guitar": 16}.
values_list = [8, 47, 10, 5, 16]
```

Has limited space. Assuming there's a limit to the weight it can hold. So, assigning weights:

```
item_weights = {"football": 3, "Laptop": 11,
                "camera": 14, "BOOKS": 19,
                "Guitar": 5}.
weights_list = [3, 11, 14, 19, 5]
```

max_weight = 26.

Combinations = 2^n ; n → no. of items.

To do: max. weight constraint & has the largest sum of values.

QUBO representation

$x_i : x \in \{0,1\}$

$x_i = x_i^2 \quad (0=0, 1=1)$

can be represented in quadratic form.

0 (if the item hasn't been chosen)
1 (if the item has been chosen)

$x = \{x_0$: football, x_1 : laptop, x_2 : camera, x_3 : Books, x_4 : Guitar$\}$

value

~~$\Rightarrow \max_x f(x) = 8x_0 + 47x_1 + 10x_2$~~

$\Rightarrow \max_x f(x) = \max_x (8x_0 + 47x_1 + 10x_2 + 5x_3 + 16x_4)$.

objective function.

usually solvers minimize functions so:

$$\min_x -(8x_0 + 47x_1 + 10x_2 + 5x_3 + 16x_4).$$

QUBO problem: Given a quadratic function, what's the set of binary variables $x_i \in \{0,1\}$ that gives the smallest output?

$$\min_x x^T Q x$$

square matrix of weights/costs.
(what happens if you pick each item alone as together)

Initial problem/objective representation. (raw Q).

$$\min_x \left(\sum_i Q_{ii} x_i + \sum_{j>i} Q_{ij} x_i x_j \right)$$

how much choosing item (i) costs

how much choosing i and j costs.

constraint here: $3x_0 + 11x_1 + 14x_2 + 19x_3 + 5x_4 \leq 26$

↓ including this as penalty term in the objective function. This penalty term should be zero if total weight of the items is less (or) equal to 26. Use slack variables (S).

↓ converts inequality constraints into equality constraints.

S → auxiliary variable.

$$3x_0 + 11x_1 + 14x_2 + 19x_3 + 5x_4 + S = 26 \quad \text{— (1)}$$

$0 \leq S \leq 26$

Ex: $\{x_0 : 1, x_1 : 1, x_2 : 1, x_3 : 1, x_4 : 1\}$

total-weight = 52. (not a valid sol)

$S = 26 - 52 = -26$

$\{x_0 : 1, x_1 : 0, x_2 : 0, x_3 : 1, x_4 : 0\}$.

$22 + S = 26 \rightarrow S = 4$ (valid).

Generalizing, Binary encoding ←

$$S = \sum_{k=0}^{N-1} 2^k s_k$$

$N = [\log_2(\text{max } S)]_H$

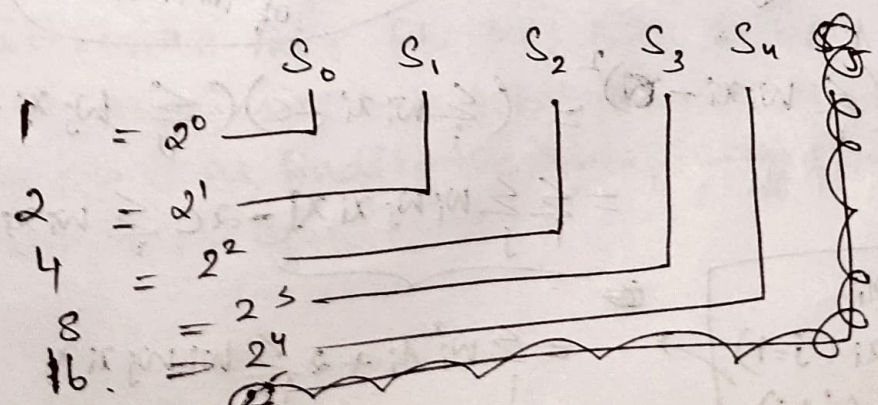
Here $N = [\log_2(26)] + 1 = 5$.

→ QUBO solvers can only work with binary (0,1) variables not regular integers (like 0 to 26 : 27 possibilities).

So using binary encoding. ($N = 5$) → 5 binary variables.

⇒ s_0 : 1's place ; s_1 : 2's place ; s_2 : 4's place ; s_3 : 8's place ; s_4 : 16's place

(2^0) (2^1)



Ex: If $S_4, S_3, S_2, S_1, S_0 = 1, 0, 0, 1, 0$.

$S = 16 + 0 + 0 + 2 + 0 = 18$,

$$\therefore \boxed{S = 2^0 S_0 + 2^1 S_1 + 2^2 S_2 + 2^3 S_3 + 2^4 S_4} \quad \text{(2)}$$

To combine (1) & (2):

$S_0 = x_5$; $S_1 = x_6$; $S_2 = x_7$; $S_3 = x_8$; $S_4 = x_9$

$$\boxed{S = 1x_5 + 2x_6 + 4x_7 + 8x_8 + 16x_9}$$

Ex: To represent $S = 4$: $\{x_5 : 0, x_6 : 0, x_7 : 1, x_8 : 0, x_9 : 0\}$

→ done using penalty (quadratic in nature) term.

penalty coefficient

$$\boxed{p(x,s) = \lambda(3x_0 + 11x_1 + 14x_2 + 19x_3 + 5x_4 + x_5 + 2x_6 + 4x_7 + 8x_8 + 16x_9 - 26)^2}$$

Rewriting objective function:

$$\min_{x,s} f(x) + p(x,s) = \min_{x,s} -(8x_0 + 47x_1 + 10x_2 + 5x_3 + 16x_4) + \lambda(3x_0 + 11x_1 + 14x_2 + 19x_3 + 5x_4 + x_5 + 2x_6 + 4x_7 + 8x_8 + 16x_9 - 26)^2$$

Generally,

$$\min_{x,s} \left(f(x) + p(x,s) \right) = -\sum_i v_i x_i + \lambda\left(\sum_i w_i x_i - W\right)^2$$

(v_i, w_i → values & weights of i^{th} item.)

here, $W = C$

$$\left(\sum_i w_i x_i - C\right)^2 = \left(\sum_i w_i x_i - C\right)\left(\sum_j w_j x_j - C\right)$$

$$= \sum_i \sum_j w_i w_j x_i x_j - 2C \sum_i w_i x_i + C^2$$

$x_i^2 = x_i$

$x_i x_j = x_i \; (j = i)$

$x_i x_j \; (j \neq i)$

$$= \sum_i w_i^2 x_i + 2 \sum_{i<j} w_i w_j x_i x_j$$

$$\therefore \left(\sum_i w_i x_i - c\right)^2 = 2\sum_{i<j} w_i w_j x_i x_j + \sum_i w_i^2 x_i - 2c\sum_i w_i x_i + c^2$$

$$= 2\sum_{i<j} w_i w_j x_i x_j + \sum_i w_i (w_i - 2c) x_i + c^2$$

$$\min_{q,s}\left(f(x) + p(x,s)\right) = -\sum_i v_i x_i + \lambda\left(2\sum_{i<j} w_i w_j x_i x_j + \sum_i w_i (w_i - 2c) x_i + c^2\right)$$

From here we construct the augmented Q matrix as :-

(cow)

$$Q_{ii} = -v_i + \lambda w_i (w_i - 2W)$$

$$Q_{ij} = 2\lambda w_i w_j$$

target sum = 26

original value / weight

weights in the constraint.

Here, λW^2 is only an offset value that doesn't affect the optimization result & can be added after the optimization to represent the right cost function.

(This is just the representation of the problem in Quantum computers / Annealers understandable form). This is then solved by either QAOA, VQE (or) QA. The QUBO matrix is directly encoded into the problem Hamiltonian that QAs like VQE or QAOA seek to minimize. So, now minimizing $f(x)$ is the same as finding the lowest energy state (ground state) of H.

Intro to QAOA (Quantum Approximate Optimization Algorithm)

↓

For combinatorial optimization probs. on NISQ devices.

Time evolution operator } $U(H,t) = e^{-iHt/\hbar}$ (scalar)

⇒ Any unitary 'U' can be written in the form of e^{irH} (K → hermitian operator, scalar)

But Hamiltonians usually has many parts.

$H = H_1 + H_2 + H_3 + \cdots$

When these parts don't commute (order matters) directly exponentiating the sum is hard to implement as a single quantum gate.

Trotter-Suzuki decomposition:

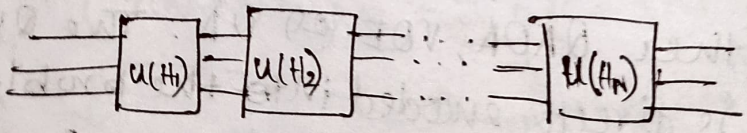
$$e^{A+B} \underset{N}{=} \left(e^{A/n} e^{B/n}\right)^n$$

$\therefore$ ~~$U(H,t,n) = \prod H = H_1 + H_2 + \cdots$~~

$$H = \sum_k H_k$$

$$\therefore U(H,t,n) = \prod_{j=1}^{n} \prod_{k} e^{-iH_k t/n}$$

(U is better as 'n' is ↑).



QAOA hamiltonian → Cost hamiltonian (H_C)

→ Mixer hamiltonian (H_M)

(i) cost hamiltonian, H_c = Encodes the optimization problem (QUBO matrix).

(ii) Mixer Hamiltonian H_M : Explores the solution space (usually sums of pauli-X operators acting on qubits).

Circuit layering means applying the unitary time evolution (operators) corresponding to these Hamiltonians alternately, multiple times.

Each layer looks like : $\underbrace{e^{-i\alpha_k H_M} e^{-i\gamma_k H_c}}_{k=1,2,\ldots p}$

Steps involved:

(i) Define a cost Hamiltonian H_c such that its ground state encodes the solution to the optimization problem.

(ii) Define a mixer Hamiltonian H_M.

(iii) construct the cost and mixer layers: $e^{-i\gamma H_c}$ and $e^{-i\alpha H_M}$.

(iv) choose a parameter $n \geq 1$ & build the circuit.

$$U(\gamma,\alpha) = e^{-i\alpha_n H_M} e^{-i\gamma_n H_c} \cdots e^{-i\alpha_1 H_M} e^{-i\gamma_1 H_c},$$

(v) Prepare an initial state, apply $U(\gamma,\alpha)$ & classical techniques to optimize the parameters.

(vi) After the circuit has been optimized, measurements of the output state reveal approximate solutions to the optimization prob.

Ising Hamiltonian

$$U(H_c, \gamma_i) = e^{-i\gamma_i \left(\sum_{i<j} J_{ij} z_i z_j + \sum_i h_i z_i\right)}$$

→ diagonal matrix. $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

↓ encodes the objective function, where each term corresponds to interactions or biases b/w bits.

↓ hence phase-flips comes into considerations.

↓ eigenstates of the Pauli-Z operator ($|0\rangle$ & $|1\rangle$).

→ Introduces phase flips.

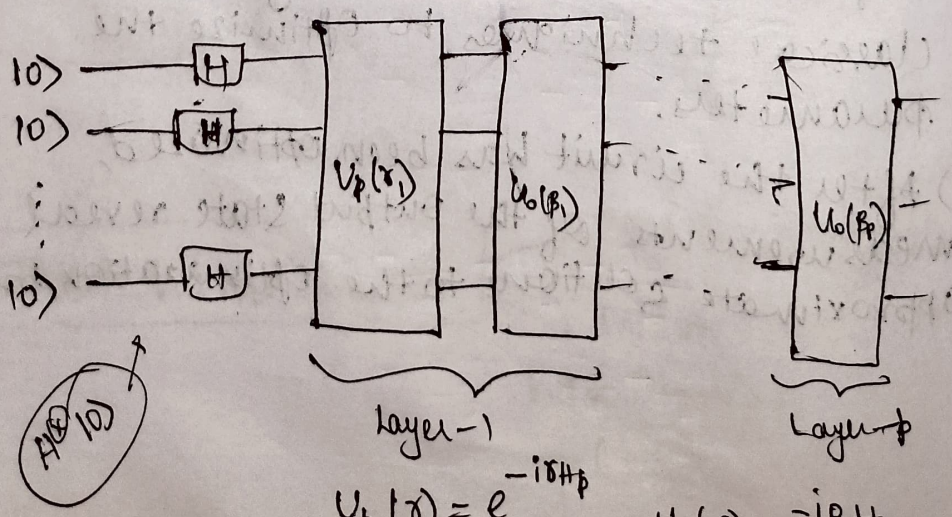
$$U(B, \beta_i) = e^{i\beta_i X} \quad ; \quad \boxed{X = \sum_{i=1}^{n} \sigma_i^x}$$

because it flips bits allowing the algorithm to transition b/w different bitstrings & explore the solution space broadly.

→ To encode QUBO into a quantum operator, each classical bit x_i is mapped to a qubit observable in the computational basis (eigenstates of Pauli-x operator):

$$\boxed{x_i = \frac{1 - z_i}{2}}$$

The main task is now about choosing QAOA parameters: (β_i & γ_i).



$$U_p(\gamma) = e^{-i\gamma H_p} \quad ; \quad U_b(\beta) = e^{-i\beta H_b}$$

EXAMPLE TO UNDERSTAND WHOLE PROCESS

(A small knapsack example).

↳ 3-item knapsack.

item_values $v = [3, 4, 5]$

item weights, $w = [2, 3, 4]$

capacity $W = 5$

* classical brute force: $2^3 = 8$ combinations

constructing Augmented QUBO matrix:

$\lambda = 6$

$$Q_{ii} = -v_i + \lambda w_i (w_i - 2W)$$

$$Q_{ij} = 2\lambda w_i w_j \quad i<j$$

$Q_{00} = -3 + 6(2)(2-2(5)$
$= -3 + 12(-8)$
$= -99$

$$Q = \begin{bmatrix} -99 & 72 & 96 \\ 0 & -130 & 144 \\ 0 & 0 & -169 \end{bmatrix}$$

(00, 01, 02, 10, 11, 12)

→ upper triangular matrix.

Now, $C(x) = x^T Q x$ → infeasible strings automatically receive large positive penalities.

Encoding: $x_i = \frac{1 - z_i}{2}$ —— (1)

$x_i x_j = \left(\frac{1 - z_i}{2}\right)\left(\frac{1 - z_j}{2}\right)$

$$x_i x_j = \frac{1}{4}\left[1 - z_j - z_i + z_i z_j\right] \quad \text{—— (2)}$$

$$C(x) = \sum_i Q_{ii} x_i + \sum_{i<j} (Q_{ij} + Q_{ji}) x_i x_j$$

$$C(x) = -99x_0 - 130x_1 - 149x_2 + 72x_0x_1 + 96x_0x_2 + 144x_1x_2$$

$H_c =$

$$H_c = \alpha_0 I + \sum_i \alpha_i z_i + \sum_{i<j} \alpha_{ij} z_i z_j \quad \text{— (I)}$$

From ① & ②:

* Linear term ($a_i x_i$) contributes:
  - ↳ constant: $a_i/2$
  - ↳ single z_i : $-a_i/2$.
* Quadratic term $b_{ij} x_i x_j$ contributes:
  - ↳ constant : $b_{ij}/4$
  - ↳ single z_i : $-b_{ij}/4$
  - ↳ single z_j : $-b_{ij}/4$
  - ↳ 2 body $z_i z_j$: $+b_{ij}/4$

In (I):

$$\alpha_0 = \sum_{ij} \frac{a_i}{2} + \frac{b_{ij}}{4}$$

$$= \frac{-99-130-149}{2} + \frac{72+96+144}{4}$$

$$\boxed{\alpha_0 = -111}$$

Single z coeffs: (α_i):

$$\alpha_0 = -\frac{a_0}{2} - \frac{b_{01}}{4} - \frac{b_{02}}{4}$$

$$= -\frac{(-99)}{2} - \frac{72}{4} - \frac{96}{4}$$

$$= 7.5$$

$$\alpha_1 = -\frac{a_1}{2} - \frac{b_{10}}{4} - \frac{b_{12}}{4}$$

$$= -\frac{(-130)}{2} - \frac{72}{4} - \frac{144}{4}$$

$$= 11$$

$$\alpha_2 = -\frac{\alpha_2}{2} - \frac{\alpha_{20}}{4} - \frac{\alpha_{21}}{4} = -\frac{(-149)}{2} - \frac{96}{4} - \frac{144}{4}$$

$$\boxed{\alpha_2 = 14.5}$$

2-body z coefficients:

$$\alpha_{01} = \frac{b_{01}}{4} = \frac{72}{4} = 18 \; ; \quad \alpha_{02} = \frac{96}{4} = 24$$

$$\alpha_{12} = \frac{144}{4} = 36$$

$$\therefore H_C = -111 + 7.5 z_0 + 11 z_1 + 14.5 z_2 + 18 z_0 z_1 + 24 z_0 z_2 + 36 z_1 z_2$$

what does z_0, z_1, $z_0 z_1$ mean when they're all pauli matrices?

$z_0 z_2 \Rightarrow z \otimes I \otimes z$ (1st, 2nd, 3rd) → only on 1st & 3rd qubit.

Now, preparing a uniform superposition over all bitstrings:

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$$

→ To explore full solution space.

$|0\rangle^{\otimes n} \rightarrow |s\rangle$ → Every candidate has equal amplitudes.

Full p-layer QAOA block:

$$[U_B(\beta_p) U_C(\gamma_p)] \cdots [U_B(\beta_1) U_C(\gamma_1)] |s\rangle$$

$p \rightarrow$ depth (no. of alternating layers)

$\{\gamma_k, \beta_k\}$ are the variational parameters to be optimized.

↳ optimizer iteratively updates these parameters to minimize the expectation:

$$\langle \psi(\gamma, \beta) | H_C | \psi(\gamma, \beta) \rangle$$

↳ Finally all the qubits in the computational basis many times. Most frequently observed bitstrings correspond to low-energy (ideally optimal) solutions of QUBO.

How is circuit layering done for cost unitary & Mixer unitary?

$$R_z(\theta) = e^{-i\theta Z/2}$$

(i) $U_c(\gamma) = e^{-i\gamma H_c}$

↳ Single-qubit: $e^{-i\gamma c_i Z_i} \rightarrow R_z(2\gamma c_i)$, θ_i

↳ Two-qubit: $e^{-i\gamma q_{ij} Z_i Z_j} \rightarrow$ RZZ rotation. ϕ_{ij}

$RZZ(2\gamma c_{ij})$

An example of how gate angles are calculated:

$\theta_i = 2\gamma c_i$: $\gamma = 1.780236$

qubit 0: $\theta_0 \approx 2 \times 1.780236 \times 7.5$ → check H_c hamiltonian

≈ 26.7035 rad

qubit 1: $\theta_1 \approx 39.6652$ rad — 11

qubit 2: $\theta_2 \approx 51.6268$ rad. — 14.5

$\phi_{ij} \approx 2\gamma c_{ij}$

(0,1): $\phi_{01} \approx 64.0885$ — 18

(0,2): $\phi_{02} \approx 85.4513$ — 24

(1,2): $\phi_{12} \approx 128.1770$ — 36

An example Numeric check for one computational basis state

Ex bitstring: 110 (→ qubit 0, qubit 1, → qubit 2.)

computing eigenvalues of each Pauli-Z term in Hamiltonian.

$Z_i|0\rangle = +1|0\rangle$ & $Z_i|1\rangle = -1|1\rangle$.

For 2-qubit ZZ terms:

$Z_0Z_1 = (-1)*(-1) = +1$

$Z_0Z_2 = (-1)*(+1) = -1$

$Z_1Z_2 = (-1)*(+1) = -1$

1	1	2
1	1	0

Eigenvalue of 110:

$$E_{110} = C_0 + C_1Z_0 + C_2Z_1 + C_3Z_2 + C_{01}Z_0Z_1 + C_{02}Z_0Z_2 + C_{12}Z_1Z_2$$

$$= -111 + 7.5(-1) + 11(-1) + \dots$$

$$= -157$$

Let's see for other bitstrings as well!

Bitstring	Z_0	Z_1	Z_2	Z_0Z_1	Z_0Z_2	Z_1Z_2	Energy
000	+1	+1	+1	+1	+1	+1	0
001	+1	+1	-1	+1	-1	-1	-150
010	+1	-1	+1	-1	+1	-1	-130
011	+1	-1	-1	-1	-1	+1	-135
100	-1	+1	+1	-1	-1	+1	-99
101	-1	+1	-1	-1	+1	-1	-152
110	-1	-1	+1	+1	-1	-1	-157
111	-1	-1	-1	+1	+1	+1	-75

lowest energy

QAOA maximizes this prob. of 110.

Classical optimizer adjust (γ, β) to maximize the lowest-energy bitstrings.

Now, let's walk through the whole Optimization part: (An example).

(2 qubit)

$$\langle \psi(\gamma,\beta) | H_c | \psi(\gamma,\beta) \rangle$$

QUBO: $Q = \begin{bmatrix} -1 & 2 \\ 0 & -1 \end{bmatrix}$

$$H_c = -0.5 + 0.Z_0 + 0.Z_1 + 0.5\, Z_0 Z_1$$

$$\boxed{H_c = -0.5 + 0.5\, Z_0 Z_1}$$

Let's pick $\gamma = 0.2$ & $\beta = 0.1$

Initial state: $|+\rangle^{\otimes 2} = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$

Applying cost unitary $U_c(\gamma) = e^{-i\gamma H_c}$ → diagonal in computational basis. Adds only phases.

Eigenvalues of H_c for each bitstring:

Bitstring	$Z_0 Z_1$	H_c eigenvalue
00	$(+1)(+1) = +1$	$-0.5 + 0.5 * 1 = 0$
01	$(+1)(-1) = -1$	$-0.5 + 0.5 * (-1) = -1$
10	$(-1)(+1) = -1$	-1
11	$(-1)(-1) = 1$	0

Phase added by $U_c(\gamma)$: $e^{-i\gamma E}$

$\boxed{\gamma = 0.2}$

Bitstring	E	Phase $e^{-i\gamma E}$
00	0	$e^{-i(0.2 \times 0)} = 1$
01	-1	$e^{-i\,0.2(-1)} = e^{i(0.2)}$
10	-1	$e^{i(0.2)}$
11	0	1

$$|\psi_2\rangle = \frac{1}{2}\left(|100\rangle + e^{i0.2}|101\rangle + e^{i0.2}|10\rangle + |111\rangle\right)$$

↓ New state after cost layer.

Applying Mixer Unitary $U_B(\beta) = e^{-i\beta(X_0+X_1)}$:

For each qubit: $R_X(2\beta) = R_X(0.2)$ rotation.

↳ This mixes amplitude, redistributing probability across bitstrings.

↳ Approximate action for small angle: ($\beta = 0.1$).

$$R_X(\theta) \approx \begin{bmatrix} \cos\theta/2 & -i\sin\theta/2 \\ -i\sin\theta/2 & \cos\theta/2 \end{bmatrix}.$$

Each qubit rotates by $0.1 \rightarrow \theta/2 \approx 0.1$ rad.

$\cos(0.1) \approx 0.995$; $\sin(0.1) \approx 0.0998$.

The new amplitudes:

Bitstrings	Amplitude
00	0.495
01	0.496
10	0.496
11	0.495

+ small imaginary

$\therefore P(00) \approx 0.245$; $P(01) = P(10) \approx 0.246$

$P(11) = 0.245$ (Prob. almost uniform)

NOW, $\langle H_c \rangle = \sum PE$

Bitstring	E	P	$\langle H \rangle$
00	0	0.245	0
01	−1	0.246	−0.246
10	−1	0.246	−0.246
11	0	0.245	0

$\langle H_c \rangle = -0.492$

Now the classical optimizer tweaks γ, β to reduce $\langle H_C \rangle$.