题目 1 说明文档

问题 1

设变量 $x_i \in \{0,1\}(i=1,2,...,M)$ 代表在 X 类材料中是否选择了 x_i（1 代表选了，0 代表没选），$y_j \in \{0,1\}(j=1,2,...,N)$ 代表在 Y 类材料中是否选择了 y_j（1 代表选了，0 代表没选），则目标优化函数可以写作：

$$f(x_1,x_2,...x_M,y_1,y_2,...,y_N)=a\left(\sum_{i=1}^{M}\sum_{j=1}^{N}\alpha_{ij}x_iy_j-A\right)^2+b\left(\sum_{i=1}^{M}\sum_{j=1}^{N}\beta_{ij}x_iy_j-B\right)^2+c\left[\left(\sum_{i=1}^{M}x_i-m\right)^2+\left(\sum_{j=1}^{N}y_j-n\right)^2\right]$$

其中 $a>0,b>0,c>>0$ 为常系数，按需求给定。我们需要**求该函数的最小值点**。

解释一下函数各项的含义：

第一项 $\left(\sum_{i=1}^{M}\sum_{j=1}^{N}\alpha_{ij}x_iy_j-A\right)^2$ 为属性一得分与 A 的差值平方，其值越小则代表属性一得分越接近 A，该项的系数 a 代表该要求的权重。

第二项 $\left(\sum_{i=1}^{M}\sum_{j=1}^{N}\beta_{ij}x_iy_j-B\right)^2$ 为属性二得分与 B 的差值平方，其值越小则代表属性二得分越接近 B，该项的系数 b 代表该要求的权重。

第三项 $\left[\left(\sum_{i=1}^{M}x_i-m\right)^2+\left(\sum_{j=1}^{N}y_j-n\right)^2\right]$ 为惩罚函数。因为题目要求在 X 和 Y 两类材料中各挑选出 m 和 n 种（m,n 均给定），所以对于当前这种变量的取值，理应有约束 $\sum_{i=1}^{M}x_i=m$ 和 $\sum_{j=1}^{N}y_j=n$，这等价于 $\left(\sum_{i=1}^{M}x_i-m\right)^2+\left(\sum_{j=1}^{N}y_j-n\right)^2=0$。而要转有约束问题为无约束问题，这里我们采用惩罚函数法：若不满足约束条件，则 $\left(\sum_{i=1}^{M}x_i-m\right)^2+\left(\sum_{j=1}^{N}y_j-n\right)^2>0$，进而 $c\left[\left(\sum_{i=1}^{M}x_i-m\right)^2+\left(\sum_{j=1}^{N}y_j-n\right)^2\right]>>0$，$f>>0$，不可能取得最小值。

对 f 做整理得：

$$
\begin{aligned}
&f\left(x_1,x_2,...x_M,y_1,y_2,...,y_N\right)\\
&=a\left[\left(\sum_{i=1}^{M}\sum_{j=1}^{N}\alpha_{ij}x_iy_j\right)^2-2A\sum_{i=1}^{M}\sum_{j=1}^{N}\alpha_{ij}x_iy_j+A^2\right]\\
&+b\left[\left(\sum_{i=1}^{M}\sum_{j=1}^{N}\beta_{ij}x_iy_j\right)^2-2B\sum_{i=1}^{M}\sum_{j=1}^{N}\beta_{ij}x_iy_j+B^2\right]\\
&+c\left[\left(\sum_{i=1}^{M}x_i\right)^2+c\left(\sum_{j=1}^{N}y_j\right)^2-2m\sum_{i=1}^{M}x_i-2n\sum_{j=1}^{N}y_j+m^2+n^2\right]\\
&=a\left[\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{M}\sum_{l=1}^{N}\alpha_{ij}\alpha_{kl}x_iy_jx_ky_l-2A\sum_{i=1}^{M}\sum_{j=1}^{N}\alpha_{ij}x_iy_j+A^2\right]\\
&+b\left[\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{M}\sum_{l=1}^{N}\beta_{ij}\beta_{kl}x_iy_jx_ky_l-2B\sum_{i=1}^{M}\sum_{j=1}^{N}\beta_{ij}x_iy_j+B^2\right]\\
&+c\left[\sum_{i=1}^{M}\sum_{k=1}^{M}x_ix_k+\sum_{j=1}^{N}\sum_{l=1}^{N}y_jy_l-2m\sum_{i=1}^{M}x_i-2n\sum_{j=1}^{N}y_j+m^2+n^2\right]
\end{aligned}
$$

常数项对最值无影响，直接去掉，最终得：

$$
\begin{aligned}
&f\left(x_1,x_2,...x_M,y_1,y_2,...,y_N\right)\\
&=\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{M}\sum_{l=1}^{N}\left(a\alpha_{ij}\alpha_{kl}+b\beta_{ij}\beta_{kl}\right)x_iy_jx_ky_l-2\sum_{i=1}^{M}\sum_{j=1}^{N}\left(aA\alpha_{ij}+bB\beta_{ij}\right)x_iy_j\\
&+c\sum_{i=1}^{M}\sum_{k=1}^{M}x_ix_k+c\sum_{j=1}^{N}\sum_{l=1}^{N}y_jy_l-2cm\sum_{i=1}^{M}x_i-2cn\sum_{j=1}^{N}y_j
\end{aligned}
$$

可见这是一个**四次二值无约束优化问题**，属于 PUBO。

为了用 QAOA **算法**求解，我们先要将二值变量编码到量子比特上。采取如下的编码方式：

$$
\left|y_N...y_2y_1x_M...x_2x_1\right\rangle \quad\Leftrightarrow\quad \left|q_{M+N-1}...q_{M+1}q_Mq_{M-1}...q_1q_0\right\rangle
$$

即：

$$
\begin{aligned}
&\left|x_i\right\rangle\Leftrightarrow\left|q_{i-1}\right\rangle \quad \left(i=1,2,...,M\right)\\
&\left|y_j\right\rangle\Leftrightarrow\left|q_{j-1+M}\right\rangle \quad \left(j=1,2,...,N\right)
\end{aligned}
$$

这里一共用了 $M+N$ 个量子比特。

然后我们需要给出该系统的哈密顿量 H_P。

考察单变量函数的哈密顿量：

例如 $f = x_1$。用 $|0\rangle$ 代表 $x_1 = 0$，$|1\rangle$ 代表 $x_1 = 1$，则哈密顿量写做：

$$H_{x1} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \frac{I - \sigma_z^0}{2}$$

考察多变量函数的哈密顿量：

例如 $f = x_i x_k$，等价于 $f = x_i \wedge x_k$（逻辑与），哈密顿量为：

$$H_{xixk} = H_{xi} \otimes H_{xk} = \frac{I - \sigma_z^{i-1}}{2} \otimes \frac{I - \sigma_z^{k-1}}{2} = \frac{I - \sigma_z^{i-1} - \sigma_z^{k-1} + \sigma_z^{i-1}\sigma_z^{k-1}}{4}$$

再例如 $f = x_i y_j x_k y_l$，哈密顿量为：

$$H_{xiyjxkyl} = H_{xi} \otimes H_{yj} \otimes H_{xk} \otimes H_{yl} = \frac{I - \sigma_z^{i-1}}{2} \otimes \frac{I - \sigma_z^{j-1+M}}{2} \otimes \frac{I - \sigma_z^{k-1}}{2} \otimes \frac{I - \sigma_z^{l-1+M}}{2}$$

回到原问题，由哈密顿量的线性性质，可得 $f\left(x_1, x_2, ...x_M, y_1, y_2, ..., y_N\right)$ 的哈密顿量为：

$$\begin{aligned} H_P = &\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{M}\sum_{l=1}^{N}\left(a\alpha_{ij}\alpha_{kl} + b\beta_{ij}\beta_{kl}\right)H_{xiyjxkyl} - 2\sum_{i=1}^{M}\sum_{j=1}^{N}\left(aA\alpha_{ij} + bB\beta_{ij}\right)H_{xiyj} \\ &+ c\sum_{i=1}^{M}\sum_{k=1}^{M}H_{xixk} + c\sum_{j=1}^{N}\sum_{l=1}^{N}H_{yjyl} - 2cm\sum_{i=1}^{M}H_{xi} - 2cn\sum_{j=1}^{N}H_{yj} \end{aligned}$$

其中 $H_{xi}, H_{yj}, H_{xixk}, H_{yjyl}, H_{xiyj}, H_{xiyjxkyl}$ 分别表示对应维度哈密顿量的张量积，其中：

$$H_{xi} = \frac{I - \sigma_z^{i-1}}{2}，\quad H_{yj} = \frac{I - \sigma_z^{j-1+M}}{2}$$

借助 pyqpanda 的泡利算符类库，我们能够以 $O\left(M^2N^2\right)$ 的复杂度算出 H_P，从而完整得到原问题的哈密顿量。

问题 2 中的 H_P 形如：

```
Hp.to_hamiltonian(1)
```

```
[({}, -1070.4635658148732),
 ({0: 'Z'}, 42.67646590394355),
 ({0: 'Z', 1: 'Z'}, 53.11261613999507),
 ({0: 'Z', 1: 'Z', 5: 'Z'}, -0.9916866229225699),
 ({0: 'Z', 1: 'Z', 5: 'Z', 6: 'Z'}, 0.24187734225695529),
 ({0: 'Z', 1: 'Z', 5: 'Z', 7: 'Z'}, 0.11855243962801679),
 ({0: 'Z', 1: 'Z', 5: 'Z', 8: 'Z'}, 0.18833153393262014),
 ({0: 'Z', 1: 'Z', 5: 'Z', 9: 'Z'}, 0.3345690833469632),
 ({0: 'Z', 1: 'Z', 6: 'Z'}, -0.9138105836624622),
 ({0: 'Z', 1: 'Z', 6: 'Z', 7: 'Z'}, 0.10076970624319774),
 ({0: 'Z', 1: 'Z', 6: 'Z', 8: 'Z'}, 0.20004025966587552),
 ({0: 'Z', 1: 'Z', 6: 'Z', 9: 'Z'}, 0.24557539053759178),
 ({0: 'Z', 1: 'Z', 7: 'Z'}, -0.64063664089708),
 ({0: 'Z', 1: 'Z', 7: 'Z', 8: 'Z'}, 0.17125131611817226),
 ({0: 'Z', 1: 'Z', 7: 'Z', 9: 'Z'}, 0.17343862378357122),
 ({0: 'Z', 1: 'Z', 8: 'Z'}, -1.2832332991905258),
 ({0: 'Z', 1: 'Z', 8: 'Z', 9: 'Z'}, 0.5098234915912817),
 ({0: 'Z', 1: 'Z', 9: 'Z'}, -1.5674781804266744),
 ({0: 'Z', 2: 'Z'}, 53.20196477809637),
 ({0: 'Z', 2: 'Z', 5: 'Z'}, -1.050327942507225),
 ({0: 'Z', 2: 'Z', 5: 'Z', 6: 'Z'}, 0.10282907824792917)
```

可见，H_P 其实就是各种 I， σ_z^i， $\sigma_z^i\sigma_z^j$， $\sigma_z^i\sigma_z^j\sigma_z^k$， $\sigma_z^i\sigma_z^j\sigma_z^k\sigma_z^l$ 乘积项的线性组合。

下面进行 QAOA 算法的设计：

根据绝热定理设计 QAOA 线路，使得：

初态哈密顿量：

$$H_B = \sum_{i=0}^{M+N-1} \sigma_x^i$$

初态哈密顿量的基态：

$$|\varphi_0\rangle = |+\rangle_{yN} \ldots |+\rangle_{y2} |+\rangle_{y1} |+\rangle_{xM} \ldots |+\rangle_{x2} |+\rangle_{x1}$$

末态哈密顿量：

$$H_P$$

测量末态哈密顿量的基态大概率可以得到所求问题的解。

QAOA 线路是以 H_B 为生成元的酉变换跟以 H_P 为生成元的酉变换乘积的累积，即：

$$U\left(\vec{\beta},\vec{\gamma}\right)=\prod_{i=1}^{p}U\left(H_B,\beta_i\right)U\left(H_P,\gamma_i\right)$$

其中：

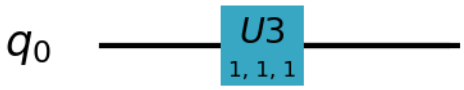
$$U\left(H_B,\beta_i\right)=\mathrm{e}^{-\mathrm{i}H_B\beta_i}=\prod_{k=0}^{M+N-1}\mathrm{RX}\left(k,2\beta_i\right)=\prod_{k=0}^{M+N-1}\mathrm{U}_3\left(k,2\beta_i,-\frac{\pi}{2},\frac{\pi}{2}\right)$$

$$U\left(H_P,\gamma_i\right)=\mathrm{e}^{-\mathrm{i}H_P\gamma_i}=\ldots$$

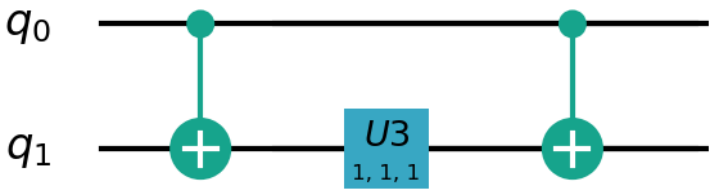
由于 H_P 是一堆泡利矩阵和单位矩阵的运算组合，所以可以预见生成的 $U\left(H_P,\gamma_i\right)$ 是一堆 RZ 门（可转化为 U_3 门）和 CNOT 门的组合。下面将对 $I,\sigma_z^i,\sigma_z^i\sigma_z^j,\sigma_z^i\sigma_z^j\sigma_z^k,\sigma_z^i\sigma_z^j\sigma_z^k\sigma_z^l$ 生成的酉变换一一进行考察，并给出对应的线路示例图：（忽略整体相位）

$$\mathrm{e}^{-\mathrm{i}\gamma_i I}=\mathrm{e}^{-\mathrm{i}\gamma_i}I=I$$

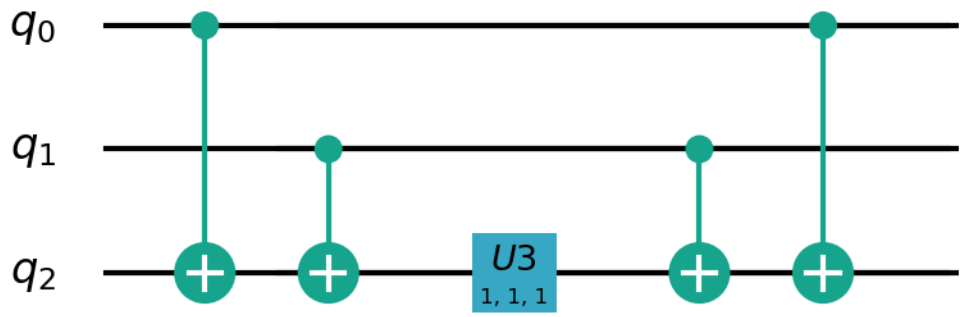
$$\mathrm{e}^{-\mathrm{i}\gamma_i\sigma_z^i}=\mathrm{RZ}\left(i,2\gamma_i\right)=\mathrm{U}_3\left(i,0,\gamma_i,\gamma_i\right)$$



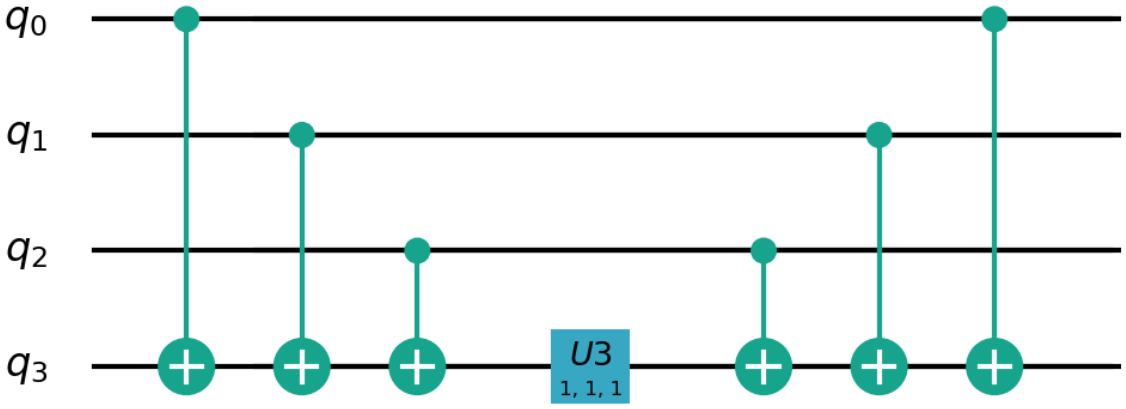
$$\mathrm{e}^{-\mathrm{i}\gamma_i\sigma_z^i\sigma_z^j}=\mathrm{CNOT}\left(i,j\right)\mathrm{RZ}\left(j,2\gamma_i\right)\mathrm{CNOT}\left(i,j\right)=\mathrm{CNOT}\left(i,j\right)\mathrm{U}_3\left(j,0,\gamma_i,\gamma_i\right)\mathrm{CNOT}\left(i,j\right)$$



$$\mathrm{e}^{-\mathrm{i}\gamma_i\sigma_z^i\sigma_z^j\sigma_z^k} = \mathrm{CNOT}(i,k)\mathrm{CNOT}(j,k)\mathrm{U}_3(k,0,\gamma_i,\gamma_i)\mathrm{CNOT}(j,k)\mathrm{CNOT}(i,k)$$



$$\mathrm{e}^{-\mathrm{i}\gamma_i\sigma_z^i\sigma_z^j\sigma_z^k\sigma_z^l} = \mathrm{CNOT}(i,l)\mathrm{CNOT}(j,l)\mathrm{CNOT}(k,l)\mathrm{U}_3(l,\gamma_i,\gamma_i)\mathrm{CNOT}(k,l)\mathrm{CNOT}(j,l)\mathrm{CNOT}(i,l)$$



然后就可以得出：

$$\begin{aligned}
&U(H_P,\gamma_i) = \mathrm{e}^{-\mathrm{i}\gamma_i H_P} \\
&= \exp\left\{-\mathrm{i}\gamma_i\sum\left[\lambda I + \lambda_i\sigma_z^i + \lambda_{ij}\sigma_z^i\sigma_z^j + \lambda_{ijk}\sigma_z^i\sigma_z^j\sigma_z^k + \lambda_{ijkl}\sigma_z^i\sigma_z^j\sigma_z^k\sigma_z^l\right]\right\} \\
&= \prod\exp\left(-\mathrm{i}\gamma_i\lambda_i\sigma_z^i\right)\cdot\prod\exp\left(-\mathrm{i}\gamma_i\lambda_{ij}\sigma_z^i\sigma_z^j\right)\cdot\prod\exp\left(-\mathrm{i}\gamma_i\lambda_{ijk}\sigma_z^i\sigma_z^j\sigma_z^k\right)\cdot\prod\exp\left(-\mathrm{i}\gamma_i\lambda_{ijkl}\sigma_z^i\sigma_z^j\sigma_z^k\sigma_z^l\right) \\
&= \prod\mathrm{U}_3(i,0,\gamma_i\lambda_i,\gamma_i\lambda_i) \\
&\quad\cdot\prod\mathrm{CNOT}(i,j)\mathrm{U}_3(j,0,\gamma_i\lambda_{ij},\gamma_i\lambda_{ij})\mathrm{CNOT}(i,j) \\
&\quad\cdot\prod\mathrm{CNOT}(i,k)\mathrm{CNOT}(j,k)\mathrm{U}_3(k,0,\gamma_i\lambda_{ijk},\gamma_i\lambda_{ijk})\mathrm{CNOT}(j,k)\mathrm{CNOT}(i,k) \\
&\quad\cdot\prod\mathrm{CNOT}(i,l)\mathrm{CNOT}(j,l)\mathrm{CNOT}(k,l)\mathrm{U}_3(l,\gamma_i\lambda_{ijkl},\gamma_i\lambda_{ijkl})\mathrm{CNOT}(k,l)\mathrm{CNOT}(j,l)\mathrm{CNOT}(i,l)
\end{aligned}$$

（其中泡利矩阵乘积项的系数$\lambda_{????}$已由代码`Hp.to_hamiltonian(1)`给出；γ_i不随求和符号和求积符号的指标而改变，因为γ_i中的i是$U\left(H_P,\gamma_i\right)$中的$i$而不是$\lambda_i$中的$i$）

可以预估，根据$U\left(\vec{\beta},\vec{\gamma}\right)$表达式，在不优化线路的情况下，完整实现$U\left(\vec{\beta},\vec{\gamma}\right)$要用到的量子逻辑门的数量为$O\left(M^2N^2\right)$量级。

综上，我们写出$U\left(\vec{\beta},\vec{\gamma}\right)$具体的量子逻辑门乘积形式，用于指导实际量子线路的搭建。而在实际情况下，我们会考虑各种门摆布的顺序问题，对相邻且相互抵消的CNOT门进行消除，相邻且可合并的U_3门进行合并，以尽量减少门的总数，优化拓扑结构——这在问题二中会详细进行。

最后给出**QAOA的工作流程：**

1. 制备初态$\left|\varphi_0\right\rangle=\left|+\right\rangle_{yN}...\left|+\right\rangle_{y2}\left|+\right\rangle_{y1}\left|+\right\rangle_{xM}...\left|+\right\rangle_{x2}\left|+\right\rangle_{x1}$（共$M+N$个量子比特）
2. 初始化参数$\left\{\vec{\beta},\vec{\gamma}\right\}$，用于确定上述的所有量子门
3. 根据参数生成量子线路，实现初态到末态的酉变换$\left|\varphi_1\right\rangle=U\left(\vec{\beta},\vec{\gamma}\right)\left|\varphi_0\right\rangle$
4. 测量末态量子状态，计算H_P基态能量的期望$E=\left\langle\varphi_1\right|H_P\left|\varphi_1\right\rangle$
5. 将当前参数及其对应的期望值传入经典优化器进行**最小值优化**得到一组新的参数
6. 重复执行3~5步，直到满足预先设定好的结束条件
7. 最终得到的最高概率的量子态$\left|\varphi_1\right\rangle=\left|y_N\right\rangle...\left|y_2\right\rangle\left|y_1\right\rangle\left|x_M\right\rangle...\left|x_2\right\rangle\left|x_1\right\rangle$就是问题的解

至此，我们达成了问题1的所有要求：使用了合适的编码手段，将问题转化为四次二值优化问题，给出了哈密顿量，并设计了对应的QAOA算法。

算法复杂度：迭代一次（3~5步）的复杂度为$O\left(pM^2N^2\right)$。

问题 2

根据题意和问题 1，取 $A=B=3,\quad a=b=1,\quad c=100$，$\alpha_{ij},\beta_{ij}$ 随机，$\beta_i=\gamma_i=1$（$i=p=1$）

```
np.random.seed(0)
a = 1
b = 1
c = 100
M = 5
N = 5
m = 3
n = 2
A = 3
B = 3
alpha = abs(np.random.randn(M + 1, N + 1))
beta = abs(np.random.randn(M + 1, N + 1))
betai = 1
gammai = 1
```

先求哈密顿量分量，再整合求哈密顿量：

```
def getH(*args):
    re = PauliOperator("", 1)
    for i in args:
        re *= PauliOperator({"": 0.5, "Z" + str(i): -0.5})
    return re
```

```
def getHp(a, b, c, M, N, m, n, A, B, alpha, beta):
    re = PauliOperator()
    for i in range(1, M + 1):
        for j in range(1, N + 1):
            re += (-2 * a * A * alpha[i, j] - 2 * b * B * beta[i, j]) * getH(
                i - 1, j - 1 + M
            )
            for k in range(1, M + 1):
                for l in range(1, N + 1):
                    re += (
                        a * alpha[i, j] * alpha[k, l] + b * beta[i, j] * beta[k, l]
                    ) * getH(i - 1, j - 1 + M, k - 1, l - 1 + M)
    for i in range(1, M + 1):
        re += (-2 * c * m) * getH(i - 1)
        for k in range(1, M + 1):
            re += c * getH(i - 1, k - 1)
    for j in range(N + 1):
        re += (-2 * c * n) * getH(j - 1 + M)
        for l in range(1, N + 1):
            re += c * getH(j - 1 + M, l - 1 + M)
    re.reduce_duplicates()
    return re
```

```
Hp = getHp(a, b, c, M, N, m, n, A, B, alpha, beta)
```

输出 H_P 泡利矩阵的组合形式：（由于一共有 256 项，显示过长，这里截断了）

```
L = Hp.to_hamiltonian(1)
```

```
len(L)
```

```
256
```

```
L
```

```
[({}, -1070.4635658148732),
 ({0: 'Z'}, 42.67646590394355),
 ({0: 'Z', 1: 'Z'}, 53.11261613999507),
 ({0: 'Z', 1: 'Z', 5: 'Z'}, -0.9916866229225699),
 ({0: 'Z', 1: 'Z', 5: 'Z', 6: 'Z'}, 0.24187734225695529),
 ({0: 'Z', 1: 'Z', 5: 'Z', 7: 'Z'}, 0.11855243962801679),
 ({0: 'Z', 1: 'Z', 5: 'Z', 8: 'Z'}, 0.18833153393262014),
 ({0: 'Z', 1: 'Z', 5: 'Z', 9: 'Z'}, 0.3345690833469632),
 ({0: 'Z', 1: 'Z', 6: 'Z'}, -0.9138105836624622),
 ({0: 'Z', 1: 'Z', 6: 'Z', 7: 'Z'}, 0.10076970624319774),
 ({0: 'Z', 1: 'Z', 6: 'Z', 8: 'Z'}, 0.20004025966587552),
 ({0: 'Z', 1: 'Z', 6: 'Z', 9: 'Z'}, 0.24557539053759178),
 ({0: 'Z', 1: 'Z', 7: 'Z'}, -0.64063664089708),
 ({0: 'Z', 1: 'Z', 7: 'Z', 8: 'Z'}, 0.17125131611817226),
 ({0: 'Z', 1: 'Z', 7: 'Z', 9: 'Z'}, 0.17343862378357122),
 ({0: 'Z', 1: 'Z', 8: 'Z'}, -1.2832332991905258),
 ({0: 'Z', 1: 'Z', 8: 'Z', 9: 'Z'}, 0.5098234915912817),
 ({0: 'Z', 1: 'Z', 9: 'Z'}, -1.5674781804266744)
```

如果不进行线路优化，将面临下面的庞大数量：

```
cnt = [0] * 5
for i in L:
    cnt[len(i[0])] += 1
print("哈密顿量的不同数量乘积项的数量：")
for i, v in enumerate(cnt):
    print(i, ":\t", v)
```

```
哈密顿量的不同数量乘积项的数量：
0 :      1
1 :      10
2 :      45
3 :      100
4 :      100
```

```
print("U3门数量：\t", 10 + 10 + 45 + 100 + 100)
print("CNOT门数量：\t", 45 * 2 + 100 * 4 + 100 * 6)
```

```
U3门数量：        265
CNOT门数量：      1090
```

【线路优化】

下面仅关注纯线路优化，不涉及到网络拓扑结构：

注意到：

$$
\begin{aligned}
&U\left(\vec{\beta},\vec{\gamma}\right)=U\left(H_B,\beta_i\right)U\left(H_P,\gamma_i\right)\\
&=\prod \mathrm{U}_3\left(i,2\beta_i,-\frac{\pi}{2},\frac{\pi}{2}\right)\\
&\cdot\prod \mathrm{U}_3\left(i,0,\gamma_i\lambda_i,\gamma_i\lambda_i\right)\\
&\cdot\prod \mathrm{CNOT}\left(i,j\right)\mathrm{U}_3\left(j,0,\gamma_i\lambda_{ij},\gamma_i\lambda_{ij}\right)\mathrm{CNOT}\left(i,j\right)\\
&\cdot\prod \mathrm{CNOT}\left(i,k\right)\mathrm{CNOT}\left(j,k\right)\mathrm{U}_3\left(k,0,\gamma_i\lambda_{ijk},\gamma_i\lambda_{ijk}\right)\mathrm{CNOT}\left(j,k\right)\mathrm{CNOT}\left(i,k\right)\\
&\cdot\prod \mathrm{CNOT}\left(i,l\right)\mathrm{CNOT}\left(j,l\right)\mathrm{CNOT}\left(k,l\right)\mathrm{U}_3\left(l,\gamma_i\lambda_{ijkl},\gamma_i\lambda_{ijkl}\right)\mathrm{CNOT}\left(k,l\right)\mathrm{CNOT}\left(j,l\right)\mathrm{CNOT}\left(i,l\right)
\end{aligned}
$$

除了$U\left(H_B,\beta_i\right)$的部分，其他连乘的项都是单位矩阵，所以它们可以互相交换。

考察CNOT门的部分：

记有序四元组$\left(t_i,t_j,t_k,t_l\right)$代表线路：先依次从比特$t_i,t_j,t_k$引出对比特$t_l$控制的CNOT门，然后对比特$t_l$进行$\mathrm{U}_3$门，最后再依次从比特$t_k,t_j,t_i$引出对比特$t_l$控制的CNOT门，即：

$$
\mathrm{e}^{-\mathrm{i}\gamma_i\sigma_z^i\sigma_z^j\sigma_z^k\sigma_z^l}=\mathrm{CNOT}\left(i,l\right)\mathrm{CNOT}\left(j,l\right)\mathrm{CNOT}\left(k,l\right)\mathrm{U}_3\left(l,\gamma_i\lambda_{ijkl},\gamma_i\lambda_{ijkl}\right)\mathrm{CNOT}\left(k,l\right)\mathrm{CNOT}\left(j,l\right)\mathrm{CNOT}\left(i,l\right)
$$

类似也有三元组$\left(t_i,t_j,t_k\right)$、二元组$\left(t_i,t_j\right)$。

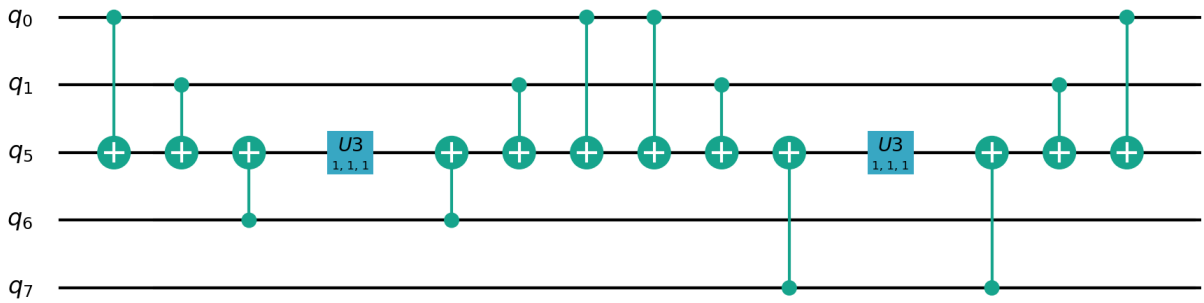
我们知道，$\mathrm{e}^{-\mathrm{i}\gamma_i\sigma_z^i\sigma_z^j\sigma_z^k\sigma_z^l}$中$i,j,k,l$本来就是完全对等的。所以对于一个特定的$i,j,k,l$，我们可以任意选择线路四元组$\left(t_i,t_j,t_k,t_l\right)$的顺序，比如$\left(l,k,j,i\right)$。三元组、二元组亦如此。

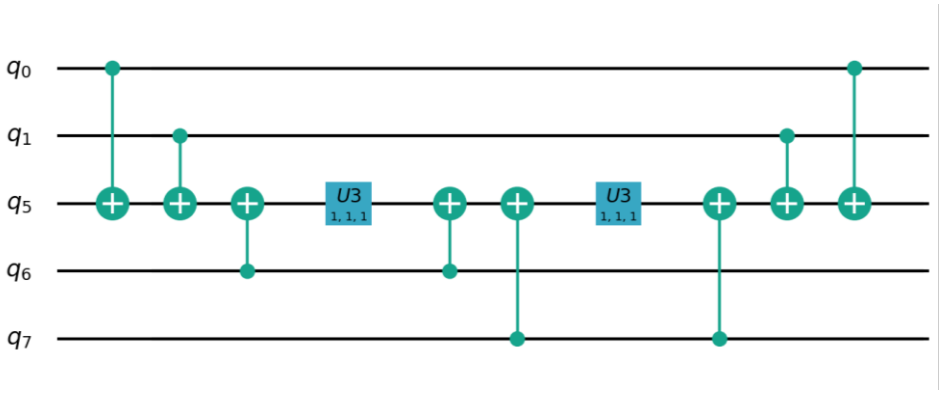
但是注意到以下情况：

如果我们把线路$\left(x_1,x_2,y_2,y_1\right)$和线路$\left(x_1,x_2,y_3,y_1\right)$接在一起，它们相邻部分的4个CNOT门会奇迹般地抵消掉了。甚至我们还可以在后面接$\left(x_1,x_2,y_4,y_1\right)$、$\left(x_1,x_2,y_5,y_1\right)$，都会依次消去4个CNOT门。而上面这些线路都是$U\left(\vec{\beta},\vec{\gamma}\right)$所需要的。这告诉我们：虽然四元组$\left(t_i,t_j,t_k,t_l\right)$中比特的次序对单个线路没有任何影响，但是选择合适的次序以及线路的接法可以大大减少CNOT门的数

量。这是一种有效的线路编译优化。

示意图：$(0,1,6,5)(0,1,7,5)$的抵消





知道了这些，我们先来看看 $e^{-i\gamma_i\sigma_z^i\sigma_z^j\sigma_z^k\sigma_z^l}$ 这些线路到底是些什么。问题 1 已经给出：

$$H_{xiyjxkyl}=H_{xi}\otimes H_{yj}\otimes H_{xk}\otimes H_{yl}=\frac{I-\sigma_z^{i-1}}{2}\otimes\frac{I-\sigma_z^{j-1+M}}{2}\otimes\frac{I-\sigma_z^{k-1}}{2}\otimes\frac{I-\sigma_z^{l-1+M}}{2}$$

$\sigma_z^i\sigma_z^j\sigma_z^k\sigma_z^l$ 完全来自于这里。可以推知所有有效的四元组线路是在 $q_0\sim q_4$（$x_1\sim x_5$）选择 2 比特和在 $q_5\sim q_9$（$y_1\sim y_5$）选择 2 比特组成。共 $C_5^2C_5^2=100$ 恰好符合之前的统计。

可以构造下面这种线路接法实现尽可能消除更多的 CNOT 门。

依次固定四元组末尾元素为 $y_1\sim y_5$，然后按顺序遍历 x_i,x_j,y_k，并且避免不重复，最终得到：

```
display(order4())
```

```
总长度： 100     原本的CNOT门数： 600     减少的CNOT门数： 288
(x1,x2,y2,y1)(x1,x2,y3,y1)(x1,x2,y4,y1)(x1,x2,y5,y1)(x1,x3,y2,y1)(x1,x3,y3,y1)(x1,x3,y4,y1)(x1,x3,y5,y1)(x1,x4,y2,y1)
(x1,x4,y3,y1)(x1,x4,y4,y1)(x1,x4,y5,y1)(x1,x5,y2,y1)(x1,x5,y3,y1)(x1,x5,y4,y1)(x1,x5,y5,y1)(x2,x3,y2,y1)(x2,x3,y3,y1)
(x2,x3,y4,y1)(x2,x3,y5,y1)(x2,x4,y2,y1)(x2,x4,y3,y1)(x2,x4,y4,y1)(x2,x4,y5,y1)(x2,x5,y2,y1)(x2,x5,y3,y1)(x2,x5,y4,y1)
(x2,x5,y5,y1)(x3,x4,y2,y1)(x3,x4,y3,y1)(x3,x4,y4,y1)(x3,x4,y5,y1)(x3,x5,y2,y1)(x3,x5,y3,y1)(x3,x5,y4,y1)(x3,x5,y5,y1)
(x4,x5,y2,y1)(x4,x5,y3,y1)(x4,x5,y4,y1)(x4,x5,y5,y1)(x1,x2,y3,y2)(x1,x2,y4,y2)(x1,x2,y5,y2)(x1,x3,y3,y2)(x1,x3,y4,y2)
(x1,x3,y5,y2)(x1,x4,y3,y2)(x1,x4,y4,y2)(x1,x4,y5,y2)(x1,x5,y3,y2)(x1,x5,y4,y2)(x1,x5,y5,y2)(x2,x3,y3,y2)(x2,x3,y4,y2)
(x2,x3,y5,y2)(x2,x4,y3,y2)(x2,x4,y4,y2)(x2,x4,y5,y2)(x2,x5,y3,y2)(x2,x5,y4,y2)(x2,x5,y5,y2)(x3,x4,y3,y2)(x3,x4,y4,y2)
(x3,x4,y5,y2)(x3,x5,y3,y2)(x3,x5,y4,y2)(x3,x5,y5,y2)(x4,x5,y3,y2)(x4,x5,y4,y2)(x4,x5,y5,y2)(x1,x2,y4,y3)(x1,x2,y5,y3)
(x1,x3,y4,y3)(x1,x3,y5,y3)(x1,x4,y4,y3)(x1,x4,y5,y3)(x1,x5,y4,y3)(x1,x5,y5,y3)(x2,x3,y4,y3)(x2,x3,y5,y3)(x2,x4,y4,y3)
(x2,x4,y5,y3)(x2,x5,y4,y3)(x2,x5,y5,y3)(x3,x4,y4,y3)(x3,x4,y5,y3)(x3,x5,y4,y3)(x3,x5,y5,y3)(x4,x5,y4,y3)(x4,x5,y5,y3)
(x1,x2,y5,y4)(x1,x3,y5,y4)(x1,x4,y5,y4)(x1,x5,y5,y4)(x2,x3,y5,y4)(x2,x4,y5,y4)(x2,x5,y5,y4)(x3,x4,y5,y4)(x3,x5,y5,y4)
(x4,x5,y5,y4)
```

共 $\mathrm{C}_5^2\times 4+\mathrm{C}_5^2\times 3+\mathrm{C}_5^2\times 2+\mathrm{C}_5^2\times 1=100$ 种，无重复，符合条件。从原本的 600 个 CNOT 门中消除了 288 个，将近一半！

我们再来看看 $\mathrm{e}^{-\mathrm{i}\gamma_i\sigma_z^i\sigma_z^j\sigma_z^k}$ 这些线路，它的来源依然是 $H_{xiyjxkyl}$。可以推知所有有效的三元组线路是在 $q_0\sim q_4$（$x_1\sim x_5$）选择 2 比特和在 $q_5\sim q_9$（$y_1\sim y_5$）选择 1 比特组成，或者在 $q_0\sim q_4$（$x_1\sim x_5$）选择 1 比特和在 $q_5\sim q_9$（$y_1\sim y_5$）选择 2 比特组成。共 $2\mathrm{C}_5^2\mathrm{C}_5^1=100$ 恰好符合之前的统计。

我们试图把这些线路插入到上面已经排好的四元组线路中实现消除。

固定末尾元素 $y_1\sim y_4$，前面两个元素在 x 中选，共计 40 个元组，我们把它分别插到已经排好的四元组中与它最大重叠的第一个元素之前，例如 (x_1,x_2,y_1) 插到 (x_1,x_2,y_2,y_1) 前，最终得：

```
display(order31())
```

```
总长度： 140     原本的CNOT门数： 760     减少的CNOT门数： 448
(x1,x2,y1)(x1,x2,y2,y1)(x1,x2,y3,y1)(x1,x2,y4,y1)(x1,x2,y5,y1)(x1,x3,y1)(x1,x3,y2,y1)(x1,x3,y3,y1)(x1,x3,y4,y1)(x1,x
3,y5,y1)(x1,x4,y1)(x1,x4,y2,y1)(x1,x4,y3,y1)(x1,x4,y4,y1)(x1,x4,y5,y1)(x1,x5,y1)(x1,x5,y2,y1)(x1,x5,y3,y1)(x1,x5,y4,y
1)(x1,x5,y5,y1)(x2,x3,y1)(x2,x3,y2,y1)(x2,x3,y3,y1)(x2,x3,y4,y1)(x2,x3,y5,y1)(x2,x4,y1)(x2,x4,y2,y1)(x2,x4,y3,y1)(x2,x
4,y4,y1)(x2,x4,y5,y1)(x2,x5,y1)(x2,x5,y2,y1)(x2,x5,y3,y1)(x2,x5,y4,y1)(x2,x5,y5,y1)(x3,x4,y1)(x3,x4,y2,y1)(x3,x4,y3,y
1)(x3,x4,y4,y1)(x3,x4,y5,y1)(x3,x5,y1)(x3,x5,y2,y1)(x3,x5,y3,y1)(x3,x5,y4,y1)(x3,x5,y5,y1)(x4,x5,y1)(x4,x5,y2,y1)(x4,x
5,y3,y1)(x4,x5,y4,y1)(x4,x5,y5,y1)(x1,x2,y2)(x1,x2,y3,y2)(x1,x2,y4,y2)(x1,x2,y5,y2)(x1,x3,y2)(x1,x3,y3,y2)(x1,x3,y4,y
2)(x1,x3,y5,y2)(x1,x4,y2)(x1,x4,y3,y2)(x1,x4,y4,y2)(x1,x4,y5,y2)(x1,x5,y2)(x1,x5,y3,y2)(x1,x5,y4,y2)(x1,x5,y5,y2)(x2,x
3,y2)(x2,x3,y3,y2)(x2,x3,y4,y2)(x2,x3,y5,y2)(x2,x4,y2)(x2,x4,y3,y2)(x2,x4,y4,y2)(x2,x4,y5,y2)(x2,x5,y2)(x2,x5,y3,y2)(x
2,x5,y4,y2)(x2,x5,y5,y2)(x3,x4,y2)(x3,x4,y3,y2)(x3,x4,y4,y2)(x3,x4,y5,y2)(x3,x5,y2)(x3,x5,y3,y2)(x3,x5,y4,y2)(x3,x5,y
5,y2)(x4,x5,y2)(x4,x5,y3,y2)(x4,x5,y4,y2)(x4,x5,y5,y2)(x1,x2,y3)(x1,x2,y4,y3)(x1,x2,y5,y3)(x1,x3,y3)(x1,x3,y4,y3)(x1,x
3,y5,y3)(x1,x4,y3)(x1,x4,y4,y3)(x1,x4,y5,y3)(x1,x5,y3)(x1,x5,y4,y3)(x1,x5,y5,y3)(x2,x3,y3)(x2,x3,y4,y3)(x2,x3,y5,y3)(x
2,x4,y3)(x2,x4,y4,y3)(x2,x4,y5,y3)(x2,x5,y3)(x2,x5,y4,y3)(x2,x5,y5,y3)(x3,x4,y3)(x3,x4,y4,y3)(x3,x4,y5,y3)(x3,x5,y3)(x
3,x5,y4,y3)(x3,x5,y5,y3)(x4,x5,y3)(x4,x5,y4,y3)(x4,x5,y5,y3)(x1,x2,y4)(x1,x2,y5,y4)(x1,x3,y4)(x1,x3,y5,y4)(x1,x4,y4)(x
1,x4,y5,y4)(x1,x5,y4)(x1,x5,y5,y4)(x2,x3,y4)(x2,x3,y5,y4)(x2,x4,y4)(x2,x4,y5,y4)(x2,x5,y4)(x2,x5,y5,y4)(x3,x4,y4)(x3,x
4,y5,y4)(x3,x5,y4)(x3,x5,y5,y4)(x4,x5,y4)(x4,x5,y5,y4)
```

剩下的：“末尾元素 y_5，前面两个元素在 x 中选”和“末尾元素 $x_1\sim x_5$，前面两个元素在 y 中

选”。把这第一种情况，调整为“末尾元素 $x_1 \sim x_5$，前面两个元素分别在 x, y 中选”，即 $\left(x_i, y_5, x_j\right)$，把这种元组留下，在后面和二元组重叠。对于第二种情况，为了最大重叠，我们可以这样构造：

```
display(order32())
```

```
总长度： 50     原本的CNOT门数： 200     减少的CNOT门数： 60
(y1,y2,x1)(y1,y3,x1)(y1,y4,x1)(y1,y5,x1)(y2,y3,x1)(y2,y4,x1)(y2,y5,x1)(y3,y4,x1)(y3,y5,x1)(y4,y5,x1)(y1,y2,x2)(y1,y3,x
2)(y1,y4,x2)(y1,y5,x2)(y2,y3,x2)(y2,y4,x2)(y2,y5,x2)(y3,y4,x2)(y3,y5,x2)(y4,y5,x2)(y1,y2,x3)(y1,y3,x3)(y1,y4,x3)(y1,y
5,x3)(y2,y3,x3)(y2,y4,x3)(y2,y5,x3)(y3,y4,x3)(y3,y5,x3)(y4,y5,x3)(y1,y2,x4)(y1,y3,x4)(y1,y4,x4)(y1,y5,x4)(y2,y3,x4)(y
2,y4,x4)(y2,y5,x4)(y3,y4,x4)(y3,y5,x4)(y4,y5,x4)(y1,y2,x5)(y1,y3,x5)(y1,y4,x5)(y1,y5,x5)(y2,y3,x5)(y2,y4,x5)(y2,y5,x5)
(y3,y4,x5)(y3,y5,x5)(y4,y5,x5)
```

最后看看 $\mathrm{e}^{-\mathrm{i}\gamma_i \sigma_z^i \sigma_z^j}$ 这些线路，它来源于 $H_{xixk}, H_{yjyl}, H_{xiyj}, H_{xiyjxkyl}$。可以推知所有有效的二元组线路是在所有比特中任意选择 2 比特组成。共 $\mathrm{C}_{10}^2=45$ 恰好符合之前的统计。

我们试图把这些线路插入到上面已经排好的四元组线路和三元组线路中实现消除。

$\left(x_i, x_j\right)$ 类型的元组与三元组第一种情况留下的 $\left(x_i, y_5, x_j\right)$ 形成最大重叠；$\left(y_i, x_j\right)$ 类型的元组与三元组第二种情况的 $\left(y_i, y_?, x_j\right)$ 形成最大重叠；$\left(y_i, y_j\right)$ 单列。整合得到：

```
display(order2())
```

```
总长度： 105     原本的CNOT门数： 330     减少的CNOT门数： 120
(y2,y1)(y3,y1)(y4,y1)(y5,y1)(y3,y2)(y4,y2)(y5,y2)(y4,y3)(y5,y3)(y5,y4)(x2,x1)(x2,y5,x1)(x3,x1)(x3,y5,x1)(x4,x1)(x4,y
5,x1)(x5,x1)(x5,y5,x1)(x3,x2)(x3,y5,x2)(x4,x2)(x4,y5,x2)(x5,x2)(x5,y5,x2)(x4,x3)(x4,y5,x3)(x5,x3)(x5,y5,x3)(x5,x4)(x
5,y5,x4)(y1,x1)(y1,y2,x1)(y1,y3,x1)(y1,y4,x1)(y1,y5,x1)(y2,x1)(y2,y3,x1)(y2,y4,x1)(y2,y5,x1)(y3,x1)(y3,y4,x1)(y3,y5,x
1)(y4,x1)(y4,y5,x1)(y1,x2)(y1,y2,x2)(y1,y3,x2)(y1,y4,x2)(y1,y5,x2)(y2,x2)(y2,y3,x2)(y2,y4,x2)(y2,y5,x2)(y3,x2)(y3,y4,x
2)(y3,y5,x2)(y4,x2)(y4,y5,x2)(y1,x3)(y1,y2,x3)(y1,y3,x3)(y1,y4,x3)(y1,y5,x3)(y2,x3)(y2,y3,x3)(y2,y4,x3)(y2,y5,x3)(y3,x
3)(y3,y4,x3)(y3,y5,x3)(y4,x3)(y4,y5,x3)(y1,x4)(y1,y2,x4)(y1,y3,x4)(y1,y4,x4)(y1,y5,x4)(y2,x4)(y2,y3,x4)(y2,y4,x4)(y2,y
5,x4)(y3,x4)(y3,y4,x4)(y3,y5,x4)(y4,x4)(y4,y5,x4)(y1,x5)(y1,y2,x5)(y1,y3,x5)(y1,y4,x5)(y1,y5,x5)(y2,x5)(y2,y3,x5)(y2,y
4,x5)(y2,y5,x5)(y3,x5)(y3,y4,x5)(y3,y5,x5)(y4,x5)(y4,y5,x5)(y5,x1)(y5,x2)(y5,x3)(y5,x4)(y5,x5)
```

其实还可以更优化，考察四元组线路与三元组、二元组的组合，也能消去一些 CNOT 门。但消去也不会太多，我们这里就不再过度考虑了，直接把上面的所有结果拼接：

```
display(order())
```

```
总长度： 245     原本的CNOT门数： 1090     减少的CNOT门数： 568
(y2,y1)(y3,y1)(y4,y1)(y5,y1)(y3,y2)(y4,y2)(y5,y2)(y4,y3)(y5,y3)(y5,y4)(x2,x1)(x2,y5,x1)(x3,x1)(x3,y5,x1)(x4,x1)(x4,y
5,x1)(x5,x1)(x5,y5,x1)(x3,x2)(x3,y5,x2)(x4,x2)(x4,y5,x2)(x5,x2)(x5,y5,x2)(x4,x3)(x4,y5,x3)(x5,x3)(x5,y5,x3)(x5,x4)(x
5,y5,x4)(y1,x1)(y1,y2,x1)(y1,y3,x1)(y1,y4,x1)(y1,y5,x1)(y2,x1)(y2,y3,x1)(y2,y4,x1)(y2,y5,x1)(y3,x1)(y3,y4,x1)(y3,y5,x
1)(y4,x1)(y4,y5,x1)(y1,x2)(y1,y2,x2)(y1,y3,x2)(y1,y4,x2)(y1,y5,x2)(y2,x2)(y2,y3,x2)(y2,y4,x2)(y2,y5,x2)(y3,x2)(y3,y4,x
2)(y3,y5,x2)(y4,x2)(y4,y5,x2)(y1,x3)(y1,y2,x3)(y1,y3,x3)(y1,y4,x3)(y1,y5,x3)(y2,x3)(y2,y3,x3)(y2,y4,x3)(y2,y5,x3)(y3,x
3)(y3,y4,x3)(y3,y5,x3)(y4,x3)(y4,y5,x3)(y1,x4)(y1,y2,x4)(y1,y3,x4)(y1,y4,x4)(y1,y5,x4)(y2,x4)(y2,y3,x4)(y2,y4,x4)(y2,y
5,x4)(y3,x4)(y3,y4,x4)(y3,y5,x4)(y4,x4)(y4,y5,x4)(y1,x5)(y1,y2,x5)(y1,y3,x5)(y1,y4,x5)(y1,y5,x5)(y2,x5)(y2,y3,x5)(y2,y
4,x5)(y2,y5,x5)(y3,x5)(y3,y4,x5)(y3,y5,x5)(y4,x5)(y4,y5,x5)(y5,x1)(y5,x2)(y5,x3)(y5,x4)(y5,x5)(x1,x2,y1)(x1,x2,y2,y1)
(x1,x2,y3,y1)(x1,x2,y4,y1)(x1,x2,y5,y1)(x1,x3,y1)(x1,x3,y2,y1)(x1,x3,y3,y1)(x1,x3,y4,y1)(x1,x3,y5,y1)(x1,x4,y1)(x1,x
4,y2,y1)(x1,x4,y3,y1)(x1,x4,y4,y1)(x1,x4,y5,y1)(x1,x5,y1)(x1,x5,y2,y1)(x1,x5,y3,y1)(x1,x5,y4,y1)(x1,x5,y5,y1)(x2,x3,y
1)(x2,x3,y2,y1)(x2,x3,y3,y1)(x2,x3,y4,y1)(x2,x3,y5,y1)(x2,x4,y1)(x2,x4,y2,y1)(x2,x4,y3,y1)(x2,x4,y4,y1)(x2,x4,y5,y1)(x
2,x5,y1)(x2,x5,y2,y1)(x2,x5,y3,y1)(x2,x5,y4,y1)(x2,x5,y5,y1)(x3,x4,y1)(x3,x4,y2,y1)(x3,x4,y3,y1)(x3,x4,y4,y1)(x3,x4,y
5,y1)(x3,x5,y1)(x3,x5,y2,y1)(x3,x5,y3,y1)(x3,x5,y4,y1)(x3,x5,y5,y1)(x4,x5,y1)(x4,x5,y2,y1)(x4,x5,y3,y1)(x4,x5,y4,y1)(x
4,x5,y5,y1)(x1,x2,y2)(x1,x2,y3,y2)(x1,x2,y4,y2)(x1,x2,y5,y2)(x1,x3,y2)(x1,x3,y3,y2)(x1,x3,y4,y2)(x1,x3,y5,y2)(x1,x4,y
2)(x1,x4,y3,y2)(x1,x4,y4,y2)(x1,x4,y5,y2)(x1,x5,y2)(x1,x5,y3,y2)(x1,x5,y4,y2)(x1,x5,y5,y2)(x2,x3,y2)(x2,x3,y3,y2)(x2,x
3,y4,y2)(x2,x3,y5,y2)(x2,x4,y2)(x2,x4,y3,y2)(x2,x4,y4,y2)(x2,x4,y5,y2)(x2,x5,y2)(x2,x5,y3,y2)(x2,x5,y4,y2)(x2,x5,y5,y
2)(x3,x4,y2)(x3,x4,y3,y2)(x3,x4,y4,y2)(x3,x4,y5,y2)(x3,x5,y2)(x3,x5,y3,y2)(x3,x5,y4,y2)(x3,x5,y5,y2)(x4,x5,y2)(x4,x5,y
3,y2)(x4,x5,y4,y2)(x4,x5,y5,y2)(x1,x2,y3)(x1,x2,y4,y3)(x1,x2,y5,y3)(x1,x3,y3)(x1,x3,y4,y3)(x1,x3,y5,y3)(x1,x4,y3)(x1,x
4,y4,y3)(x1,x4,y5,y3)(x1,x5,y3)(x1,x5,y4,y3)(x1,x5,y5,y3)(x2,x3,y3)(x2,x3,y4,y3)(x2,x3,y5,y3)(x2,x4,y3)(x2,x4,y4,y3)(x
2,x4,y5,y3)(x2,x5,y3)(x2,x5,y4,y3)(x2,x5,y5,y3)(x3,x4,y3)(x3,x4,y4,y3)(x3,x4,y5,y3)(x3,x5,y3)(x3,x5,y4,y3)(x3,x5,y5,y
3)(x4,x5,y3)(x4,x5,y4,y3)(x4,x5,y5,y3)(x1,x2,y4)(x1,x2,y5,y4)(x1,x3,y4)(x1,x3,y5,y4)(x1,x4,y4)(x1,x4,y5,y4)(x1,x5,y4)
(x1,x5,y5,y4)(x2,x3,y4)(x2,x3,y5,y4)(x2,x4,y4)(x2,x4,y5,y4)(x2,x5,y4)(x2,x5,y5,y4)(x3,x4,y4)(x3,x4,y5,y4)(x3,x5,y4)(x
3,x5,y5,y4)(x4,x5,y4)(x4,x5,y5,y4)
```

减少了超过一半的 CNOT 门，已经很满足了。

直接用 pyqpanda 画出这种排布的线路：

```
prog.get_qgate_num()
```

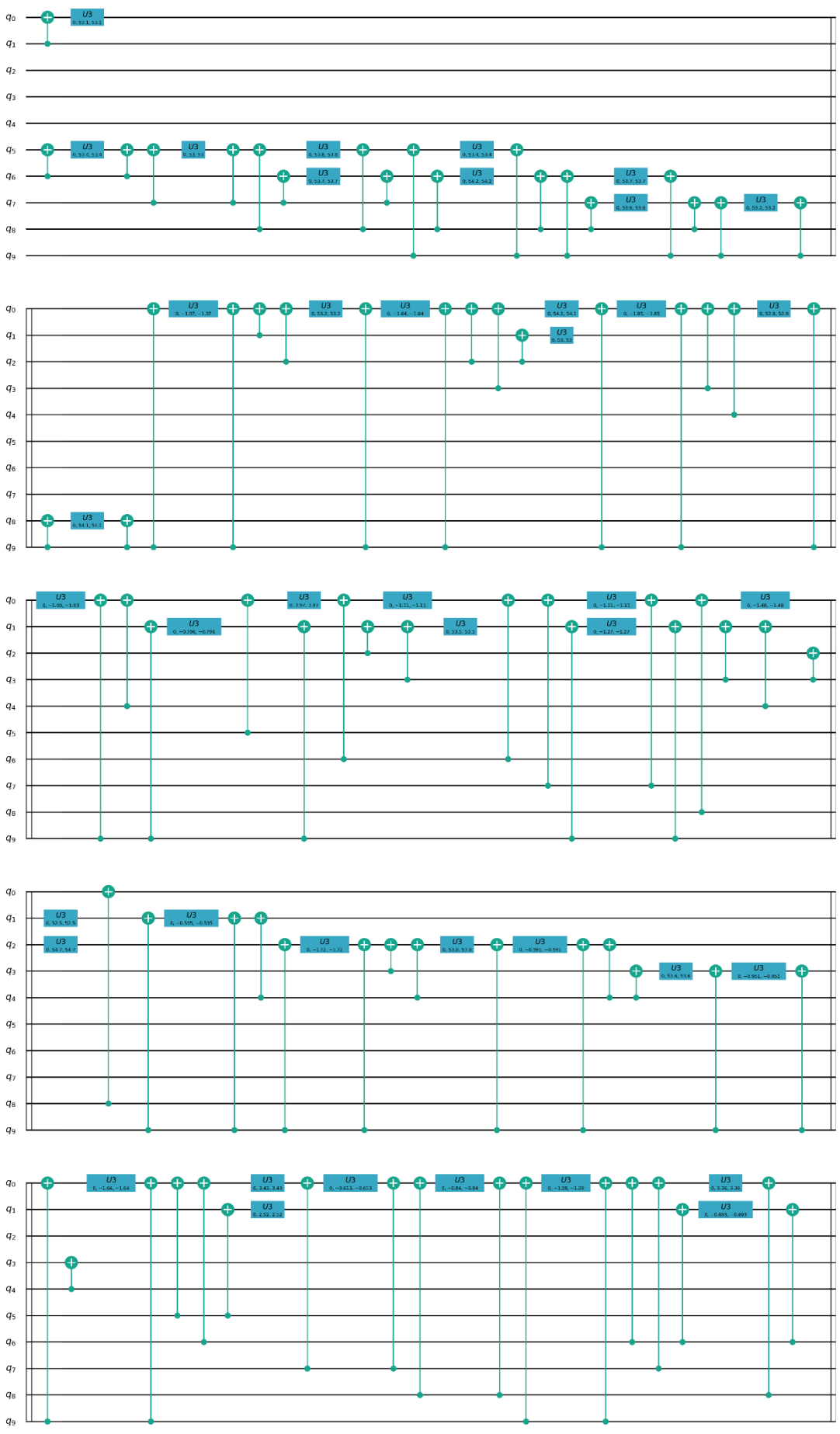
```
787
```

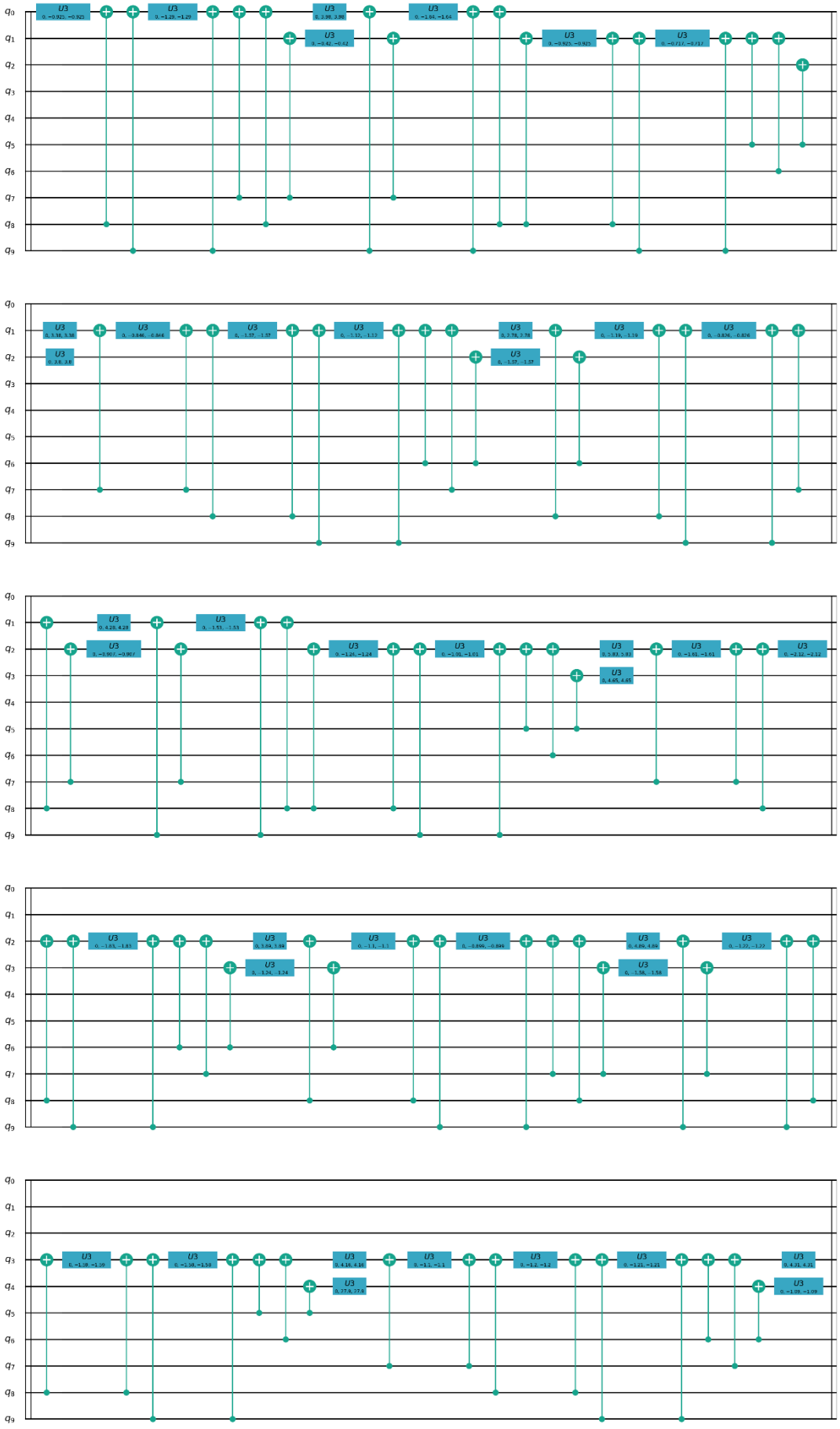
```
265 + 1090 - 568
```

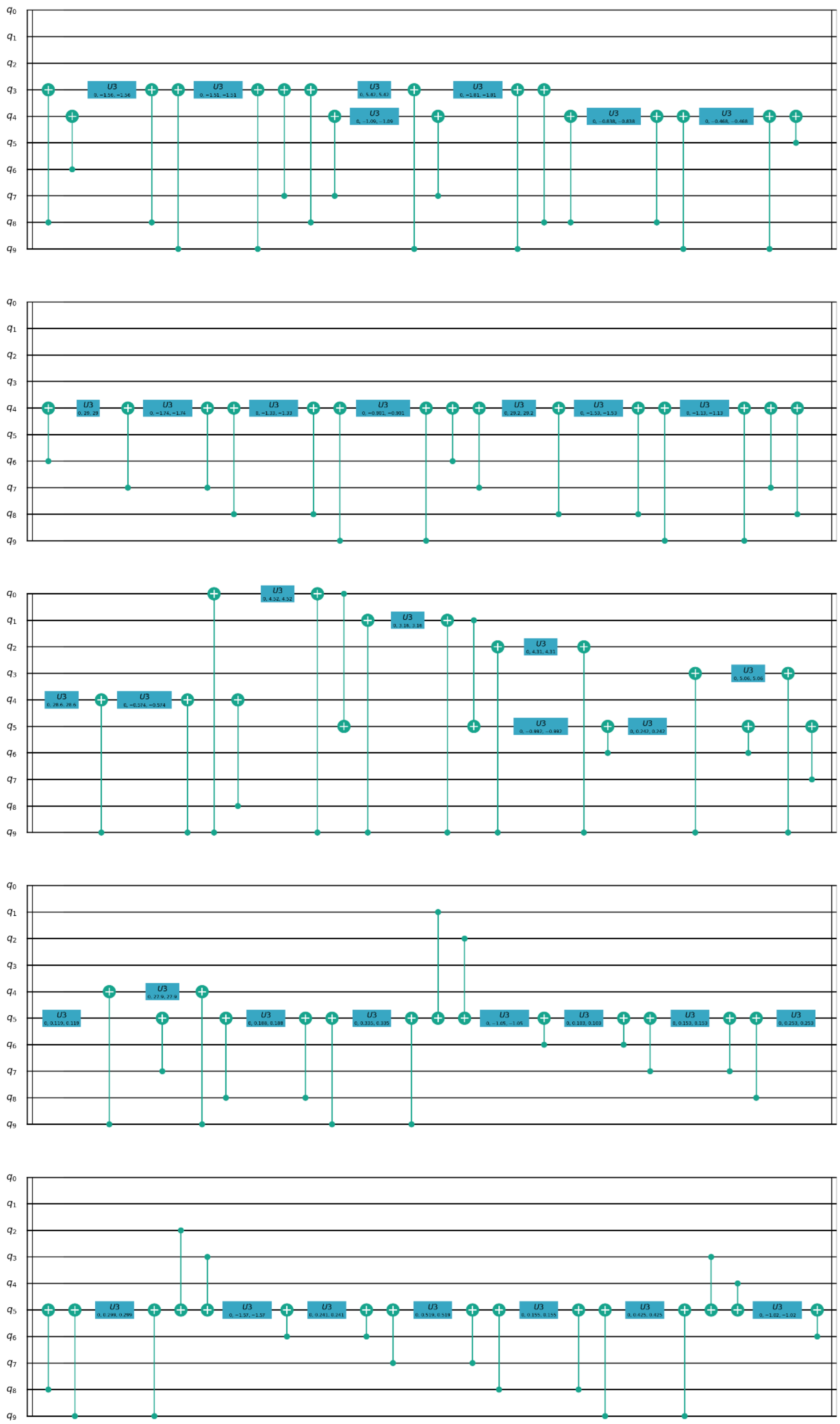
```
787
```

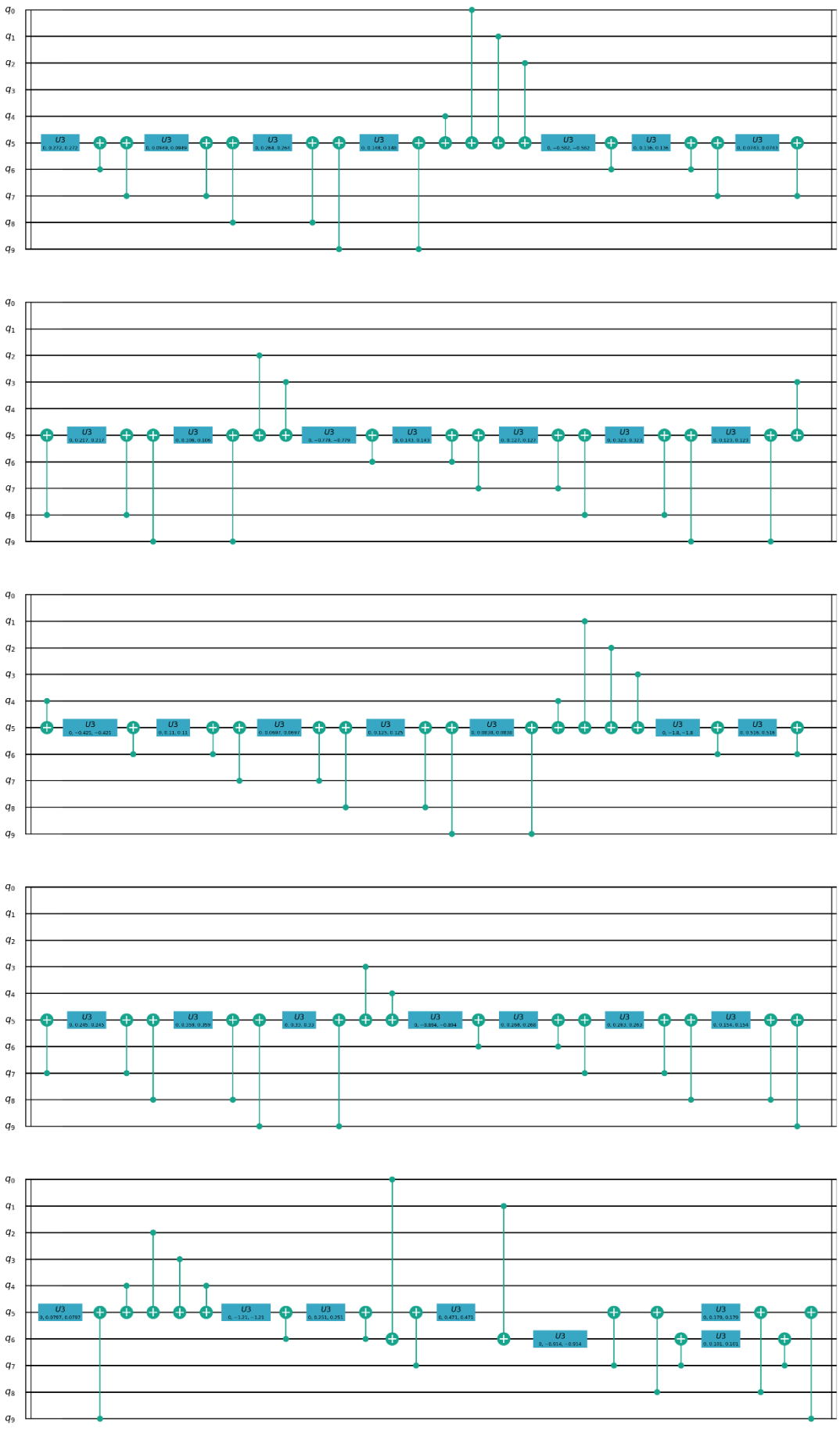
总共需要 787 个门，其中有 522 个 CNOT 门。

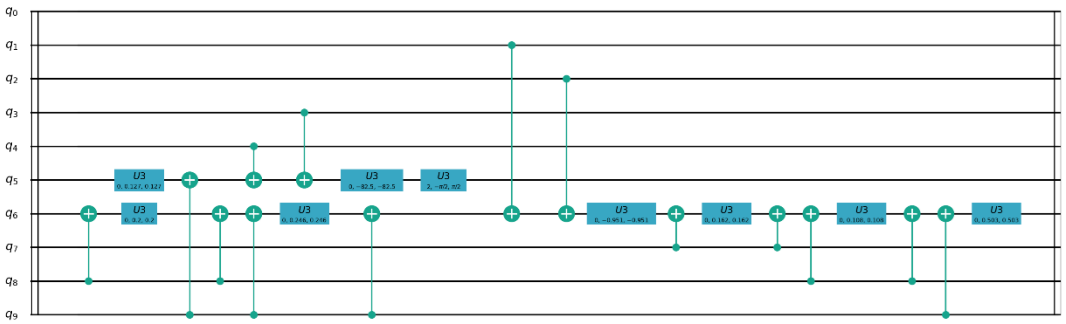
线路较长，展示如下：

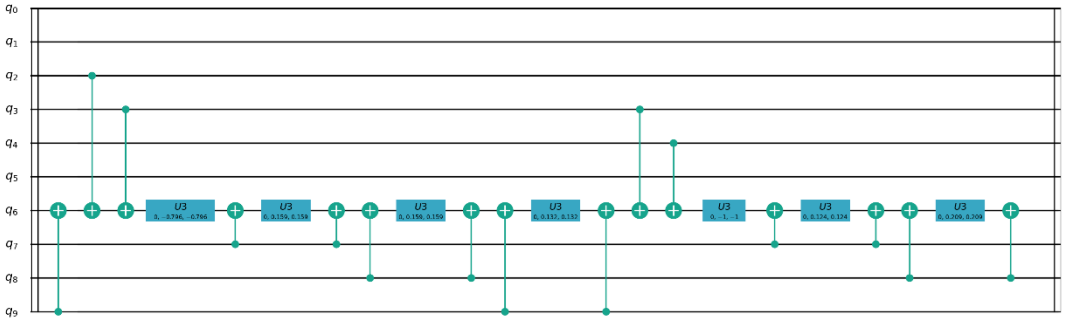


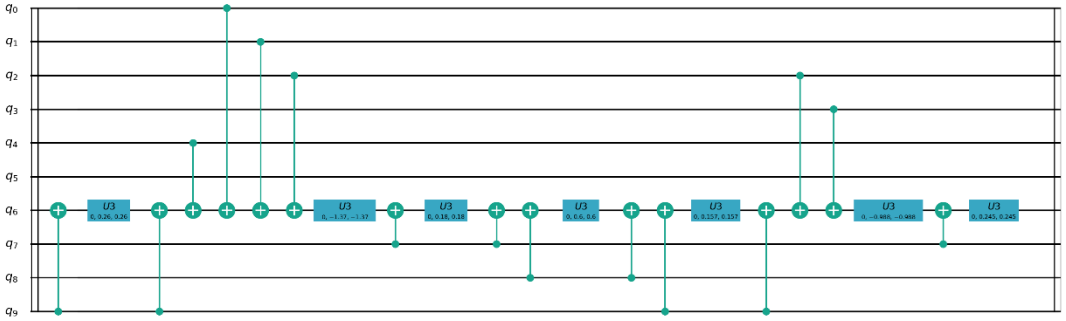


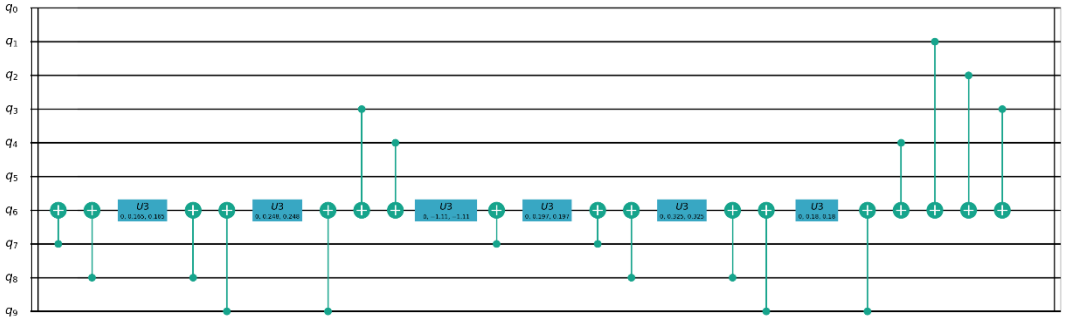

q_0 q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8 q_9
U3 0, −1.56, −1.56
U3 0, −1.51, −1.51
U3 0, 5.42, 5.42
U3 0, −1.09, −1.09
U3 0, −1.81, −1.81
U3 0, −0.838, −0.838
U3 0, −0.468, −0.468
q_0 q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8 q_9
U3 0, 29, 29
U3 0, −1.74, −1.74
U3 0, −1.33, −1.33
U3 0, −0.901, −0.901
U3 0, 29.2, 29.2
U3 0, −1.53, −1.53
U3 0, −1.13, −1.13
q_0 q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8 q_9
U3 0, 28.6, 28.6
U3 0, −0.574, −0.574
U3 0, 4.52, 4.52
U3 0, 3.16, 3.16
U3 0, 4.31, 4.31
U3 0, −0.992, −0.992
U3 0, 0.242, 0.242
U3 0, 5.06, 5.06
q_0 q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8 q_9
U3 0, 0.119, 0.119
U3 0, 27.9, 27.9
U3 0, 0.188, 0.188
U3 0, 0.335, 0.335
U3 0, −1.05, −1.05
U3 0, 0.103, 0.103
U3 0, 0.153, 0.153
U3 0, 0.253, 0.253
q_0 q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8 q_9
U3 0, 0.299, 0.299
U3 0, −1.57, −1.57
U3 0, 0.241, 0.241
U3 0, 0.519, 0.519
U3 0, 0.155, 0.155
U3 0, 0.425, 0.425
U3 0, −1.02, −1.02

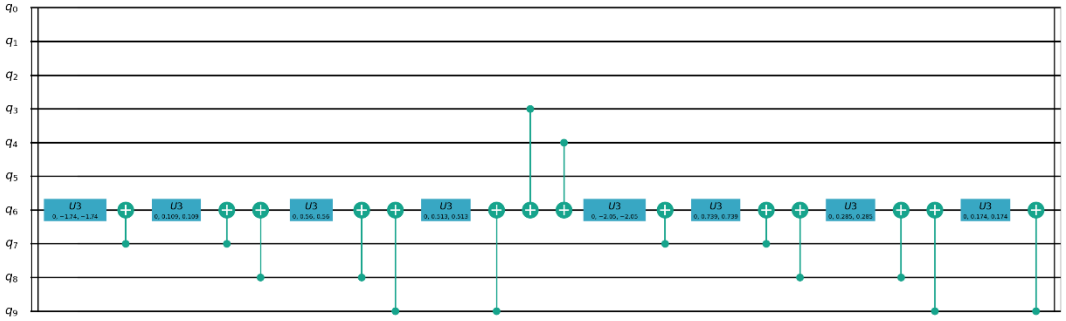




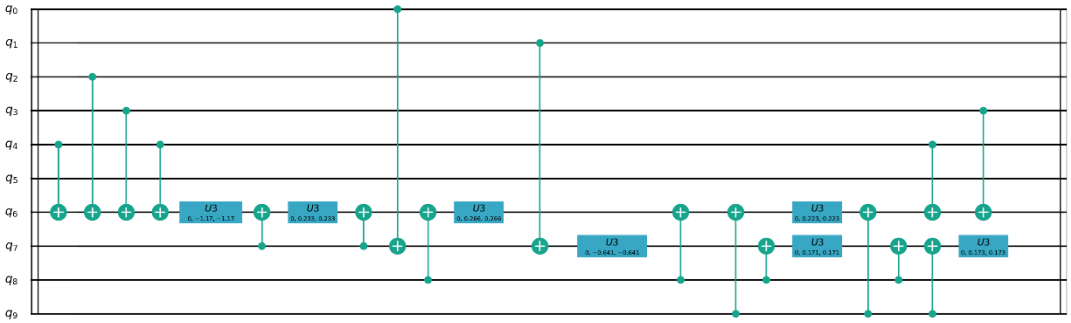


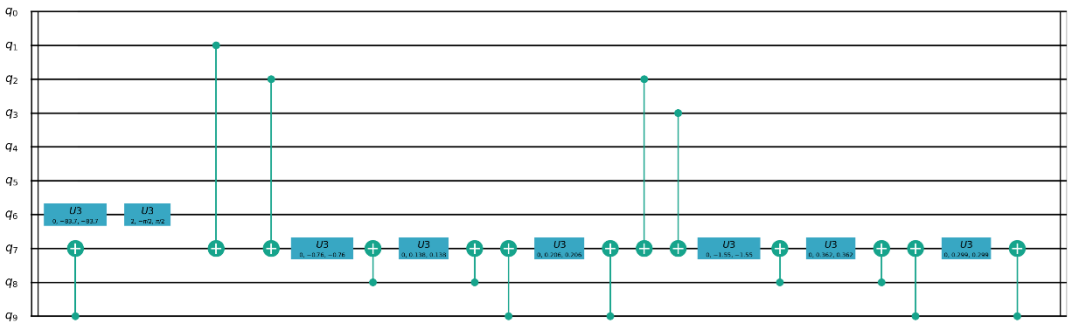


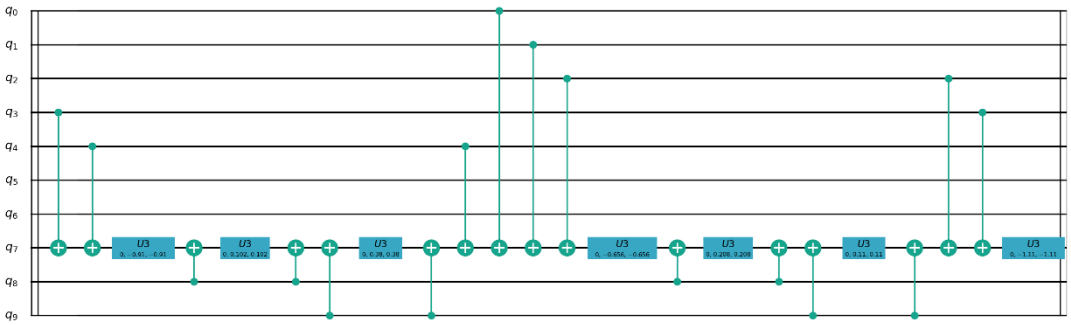


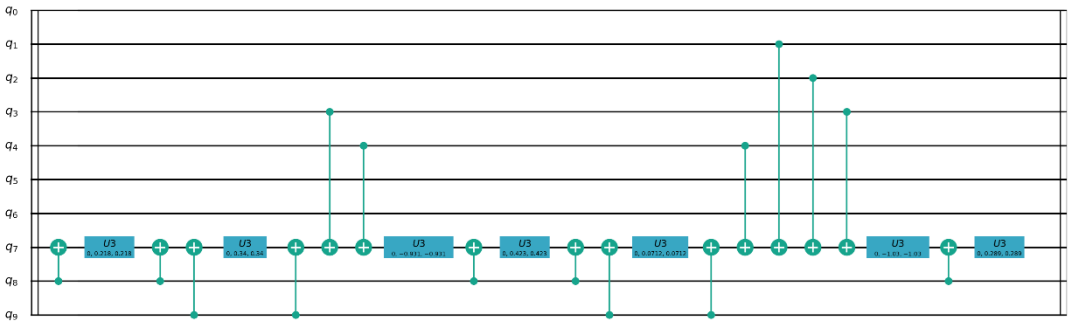


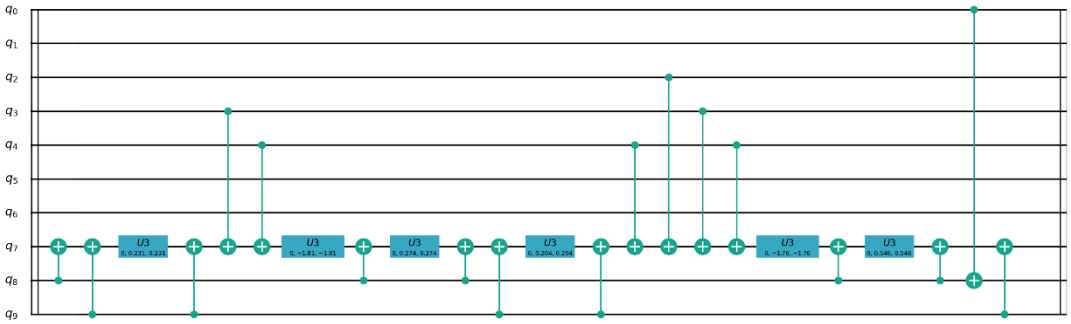
q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
q_9
U3 0, 0.127, 0.127
U3 0, 0.2, 0.2
U3 0, 0.246, 0.246
U3 0, −82.5, −82.5
U3 2, −π/2, π/2
U3 0, −0.951, −0.951
U3 0, 0.162, 0.162
U3 0, 0.108, 0.108
U3 0, 0.503, 0.503
U3 0, −0.796, −0.796
U3 0, 0.159, 0.159
U3 0, 0.159, 0.159
U3 0, 0.132, 0.132
U3 0, −1, −1
U3 0, 0.124, 0.124
U3 0, 0.209, 0.209
U3 0, 0.26, 0.26
U3 0, −1.37, −1.37
U3 0, 0.18, 0.18
U3 0, 0.6, 0.6
U3 0, 0.157, 0.157
U3 0, −0.988, −0.988
U3 0, 0.245, 0.245
U3 0, 0.165, 0.165
U3 0, 0.248, 0.248
U3 0, −1.11, −1.11
U3 0, 0.197, 0.197
U3 0, 0.325, 0.325
U3 0, 0.18, 0.18
U3 0, −1.74, −1.74
U3 0, 0.109, 0.109
U3 0, 0.56, 0.56
U3 0, 0.513, 0.513
U3 0, −2.05, −2.05
U3 0, 0.739, 0.739
U3 0, 0.285, 0.285
U3 0, 0.174, 0.174



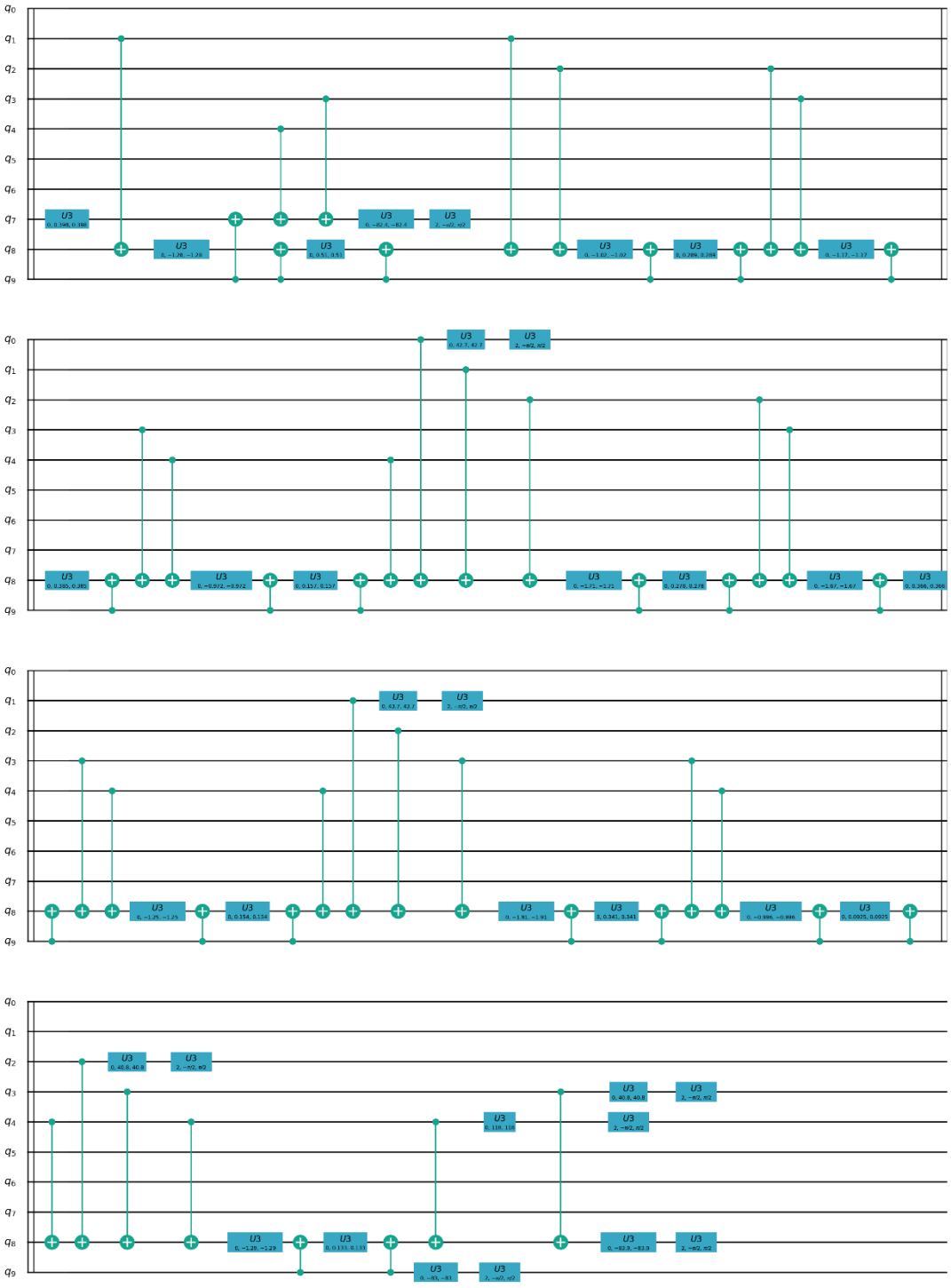








q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
q_9
U3
0, −1.17, −1.17
U3
0, 0.233, 0.233
U3
0, 0.266, 0.266
U3
0, −0.641, −0.641
U3
0, 0.223, 0.223
U3
0, 0.171, 0.171
U3
0, 0.173, 0.173
q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
q_9
U3
0, −83.7, −83.7
U3
2, −π/2, π/2
U3
0, −0.76, −0.76
U3
0, 0.138, 0.138
U3
0, 0.206, 0.206
U3
0, −1.55, −1.55
U3
0, 0.362, 0.362
U3
0, 0.299, 0.299
q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
q_9
U3
0, −0.91, −0.91
U3
0, 0.102, 0.102
U3
0, 0.38, 0.38
U3
0, −0.656, −0.656
U3
0, 0.208, 0.208
U3
0, 0.11, 0.11
U3
0, −1.11, −1.11
q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
q_9
U3
0, 0.218, 0.218
U3
0, 0.34, 0.34
U3
0, −0.931, −0.931
U3
0, 0.423, 0.423
U3
0, 0.0712, 0.0712
U3
0, −1.03, −1.03
U3
0, 0.289, 0.289
q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
q_9
U3
0, 0.231, 0.231
U3
0, −1.81, −1.81
U3
0, 0.274, 0.274
U3
0, 0.204, 0.204
U3
0, −1.76, −1.76
U3
0, 0.546, 0.546



至此，我们得出了完整的纯线路模型，解决了 Cost 层的问题。然而，QAOA 算法在真实量子芯片运行时，一般会遇到拓扑结构问题，需要进行线路编译优化，还需要解决 Mixer 层的问题。

下面将进行网络拓扑优化，重点解决量子比特间交流受限的问题。

【拓扑优化】

由于在物理层面上存在量子比特排布的限制，只能允许两个相邻的量子比特之间进行计算。

在本问题中，我们只使用到了10个量子比特，数量很少。但是由于我们选择优化的多项式为4次，量子比特间运算就较为复杂。量子比特少的代价就是它们之间的运算复杂，但量子比特少的好处是：占用芯片的区域少，需要交换（SWAP门）的次数也少。

当然还有一种方式是：增加变量（也就是增加量子比特），把四次多项式转化为二次多项式，从而减少量子比特间的运算复杂度，促使门的数量在纯线路层面就更少。量子比特多的好处是能够大大利用芯片不受串扰影响时的高并行CNOT门运算。但是，潜在的问题是：芯片占用的区域增加了，要让任意两个量子比特进行运算交换的次数也增多了。这里只是简单设想了一下，至于究竟是哪种方法最后得出的深度最小，我们并没有做深入研究。

在我们的设计中，尽量使要用到的量子比特靠在一起，比如：

$$\begin{array}{ccc} x_1 & y_1 & x_2 \\ y_2 & x_3 & y_3 \\ x_4 & y_4 & x_5 \\ y_5 & & \end{array}$$

在这种情况下进行量子比特门操作：例如，当要进行元组(x_1, x_2, y_2, y_1)线路运算时（不考虑线路优化消去CNOT门）：$\mathrm{CNOT}(x_1, y_1)$、$\mathrm{CNOT}(x_2, y_1)$可以直接运算，而$\mathrm{CNOT}(y_2, y_1)$需$\mathrm{SWAP}(y_2, x_3)$再$\mathrm{CNOT}(y_2, y_1)$再$\mathrm{SWAP}(y_2, x_3)$。

由于量子比特数量少，它们集中得很密集，我们也就不考虑芯片运算并行性了。（如果要深入研究下去，则可以考虑。但对于在比赛中的我们来说，这有些过于麻烦了，我们这里只是希望能得到一个相对较好的结果，而不是为了最优解陷入复杂的讨论而耗费大量时间）**所以我们让每一层只进行一个CNOT门运算！（无关比特直接用BARRIER隔开）**

我们的目标是找到一个量子比特的排布使得加入SWAP门（等价于3个CNOT门）后总门数最小。对于每一种排布方式，我们用pyqpanda的量子比特映射Sabre算法（尝试一定次数取门最少的时候）通过寻找可以插入SWAP门的位置，来对线路进行优化，使得原本不可以在量子计算机上运行的量子门，通过动态的改变量子比特的邻接位来使得所有的量子门都能够进行计算。

例如，对于上面给出的排布，可以得到：

```
print(map)
print(chain_graph)
```

```
[7, 9, 14, 19, 21, 8, 13, 15, 20, 25]
[[0. 0. 0. 0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1. 1. 0.]
 [1. 1. 1. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 1. 1. 0. 0. 0. 0. 0. 0.]
 [0. 1. 1. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 1. 1. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]]
```

```
result = None
min_num = 9999
for _ in range(1):
    sabre_result = transform_to_base_qgate(
        sabre_mapping(prog, machine, 20, 10, chain_graph),
        machine,
        ["U3"],
        ["CNOT"],
    )
    if sabre_result.get_qgate_num() < min_num:
        result = QProg(sabre_result)
        min_num = sabre_result.get_qgate_num()

result.get_qgate_num()
```

```
1267
```

最终得到的线路总共有1267 个门，这意味着我们的量子线路总层数大概不超过1300 层，独立CNOT 层数大概不超过1000 层。

例如，对于上面给出的排布，可以得到：

```
from get_layers import *
depth, layers = count(ans)
print('cx layer counts: ', depth)
```

```
cx layer counts:  1014
```

独立CNOT 层数大约1000 层。

我们后面的工作就是改变量子比特在芯片上排布，以求得层数最小值。

最后我们在答案文件中给出分布为：（x, y 交错排列）

$$\begin{array}{ccc} x_1 & y_1 & x_2 \\ y_2 & x_3 & y_3 \\ x_4 & y_4 & x_5 \\ & y_5 & \end{array}$$

```
!python answer.py
```

```
cx layer counts:  960
```

独立CNOT 层数为960层。

至此，我们完成了对本问题的解答。

最后来谈谈本文所述算法的通用性问题。

本文旨在直接利用 QAOA 求解 PUBO。通过含惩罚函数优化函数的建立、哈密顿量的推导、酉变换的分解，最后得到的量子线路是 $O\left(M^2N^2\right)$ 数量的CNOT 门和 U_3 门组合，属于多项式复杂度的近似算法。

本文提供了在给定情况下解决此问题的思路。如果要把它运用到其他的情况下，则需要对步骤细节进行相应的调整：（至少我们本题给出的代码就需要从头到尾修改对应内容）

1. 优化函数：一般的 PUBO 问题，写出来的优化函数的次数不一定是本题中的4 次，可能更高次。另外，变量的个数也可能发生变化（这决定了量子比特的个数）。
2. 哈密顿量：决定于优化函数。虽然形式上都是一些泡利矩阵相乘相加，但是乘方的次数和矩阵的维数可能会发生变化。
3. 酉变换的分解：更高次更多变量的优化问题意味着更多的CNOT 门，但它们都会如同本题那样形成一种对称的分布。
4. 线路优化（Cost 层）：更多的CNOT 门生成更多的元组，我们需要思考怎样排布这些元组达成最大重叠，实现纯线路层面的优化。
5. 拓扑优化（Mixer 层）：需要根据量子比特的个数在芯片上设置量子比特的排布，通过尝试量子比特映射算法取最少门数的情况。如果有需要，可以进一步尝试引入不受串扰影响时的并行CNOT 门运算，以最小化独立CNOT 层数。