基于量子近似优化算法重建最大简约树

(一) 摘要

在本文中，我们尝试将最大简约法重建树问题转化为一个二元无约束的优化问题(Quadratic Unconstrained Binary Optimization, QUBO)模型，进而使用量子近似优化算法(Quantum Approximate Optimization Algorithm, QAOA)进行求解。我们的目标是展示并评估一个量子算法解决系统发育树推断问题的实例，为将来系统发育树推断的量子算法设计和应用场景开拓作一个初步尝试。

(二) 最大简约法简介

最大简约法(maximum parsimony)是一种传统的重建系统发育树(phylogenetic tree)方式。在该方法中，每棵树都会根据所有叶(tip)的核苷酸类型以及树的拓扑结构产生简约得分(parsimonious score)，得分最低的树即为最大简约树(most parsimonious tree)。找到最大简约树是一个 NP 难的组合优化问题，但目前已有大量成熟的算法和软件支持这种方法。最大简约法的准确性不如最大似然法，但计算量与最大似然法相比要小得多，因此在当代系统发育树的研究中，最大简约法通常作为一种筛选基因树的方式，与最大似然法结合起来使用。在本文中，我们重建的对象是单基因树，即我们仅考虑每个类群同一位点上的碱基。多基因树和蛋白质树可以基于本模型进行拓展。

(三) 什么是树？

在数学上，连通的无环图(acyclic graph)称为树(tree)。因此，与图相关的部分概念可以迁移到树中，如顶点(vertex)/节点(node)， 边(edge)在树中称为分支(branch)。节点的度(degree of node)定义为与该节点相邻的分支数。树分为有根树和无根树，其根本区别在于，有根树存在一个所有类群的共同祖先，且该树的分支是有向的(directed)，它的方向反映了进化的方向。无根树则是无向的(undirected)，我们无法从该树中得到进化方向的信息。对于有根树，度还可以分为入度(in-degree)和出度(out-degree)，分别对应指向该节点的分支数和由该节点指向其他节点的分支数。因此，根的入度为 0。我们还可以知道树的末端（称为“叶节点”）的出度为 0。除叶节点以外的节点都成为内节点。若所有的内节点出度都为 2，那么该有根树又称为有根二叉树。

(四) 最大简约法重建树的过程

首先，定义两个节点在同一个位点中可能存在的最小碱基替换数量为特征长度(character length)或位点长度(site length)[1]。若两节点的碱基相同，那么节点间的分支特征长度为 0；若两节点的碱基不同，那么该分支的特征长度为 1。在一棵树中，所有分支特征长度的总和，称为简约评分(parsimonious score)、树长度(tree length)或树评分(tree score)。得分最低的树是对真树的最佳估计，称为最大简约树(maximum parsimony tree)或最简树(the most parsimonious tree)。

下面我们举例说明如何计算一棵树的简约得分。

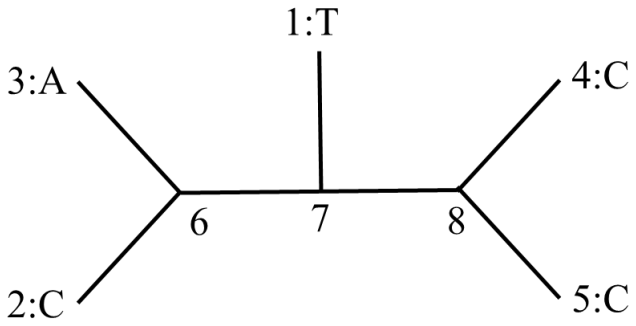

图 1 带有 5 个类群的无根单基因树 *T*，5 个类群分别对应节点 1~5，它们在同一位点上的碱基分别为“TCACC”。内节点 6~8 为它们的假想祖先。

6	7	8	(36)	(26)	(67)	(17)	(78)	(48)	(58)	*s*
A	T	C	0	1	1	0	1	0	0	3
C	T	C	1	0	1	0	1	0	0	3
A	A	C	0	1	0	1	1	0	0	3
C	A	C	1	0	1	1	1	0	0	4
A	C	C	0	1	1	1	0	0	0	3
C	C	C	1	0	0	1	0	0	0	2

表 1 该表格展示了树 *T* 每个分支的特征长度。该表格中的 1~3 列为内节点 6~8 的碱基取值[2]，4~10 列为各个分支的特征长度，用(ab)表示节点 a 和节点 b 之间的分支，最后一列 *s* 为特征长度的总和，记该树的简约

[1] 在后续的计算中，我们需要给不同的替换类型加上额外的权重来计算评分。原因是，在某些情况下，转换和颠换的频率是不同的。

[2] 该图表中并没有枚举所有 6~8 可能的取值，这是因为某些取值看起来是非常不可能的。例如节点 6 取 T 的情况，这会使分支 3-6 和分支 2-6 的特征长度都为 1，而这显然不是特征长度总和最小的情况。

得分为 min(s)，也就是 2。

图 1 展示一个无根单基因树 *T* 的示例。表 1 通过列举内节点部分碱基的组合方式，并计算了所有分支的特征长度，进而得出该树的简约得分。

从中可以看出，一般情况下，我们使用最大简约法重建需要分两步进行

i. 找到最小碱基替换数量的总和（计算出树的得分）

ii. 找到得分最低的树

实际上，上述过程可以进行合并。若我们能枚举出所有的树，枚举出所有内节点所有的碱基组合，并计算每种情况下，每棵树分支的特征长度的总和[3]，记为每种情况下的得分。那么当得分最小时，其对应的树必为最大简约树。

另一方面，由于在某些情况下，转换和颠换[4]的频率是不同的，为了更接近实际情况，我们需要加上对不同的替换类型在数量上增加额外的权重来计算评分。例如，转换的替换权重设置为 1，颠换的替换权重设置为 1.5。以图 1 的树 *T* 为例，若节点 7 为 A，那么节点 1 和节点 7 之间至少发生了一次颠换，因此该分支的特征长度为 1.5；若节点 7 为 C，那么节点 1 和节点 7 之间至少发生了一次转换，因此该分支的特征长度为 1。

(五) 编码无根树与最大简约法重建树的建模

对于最简树的重建，首先想到的是给每一个树的分支定义一个二值变量，当两个节点相连时，二值变量为 1，否则为 0；然后再给每个节点赋予一组二值变量来确定它的碱基是什么，从而计算出一个得分函数。然而，在建模中我们发现无法通过常规的约束来保证所连接的图具有树的拓扑结构，因此我们必须增加额外的辅助变量来对所遍历的变量空间进行约束[1]。

基于上述思路，我们给无根树确立一个参考节点。定义某节点的深度为该节点到参考节点之间路径(path)所经过的节点数，记参考节点的深度为 0。以图 1 的树 *T* 为例，若 7 为参考节点，则节点 1 的深度为 1，节点 3 的深度为 2。如果我们能够确保分支是由深度为(*i*-1)的节点连接至深度为 *i* 的节点，那么就能保证所连接的图是无环且连通的。进一步，如果对不同节点的入度和出度进行约束，则

[3] 它不一定是最小碱基替换数量的总和!!

[4] 转换定义嘌呤到嘌呤或者嘧啶到嘧啶的替换，颠换定义为嘌呤到嘧啶或者嘧啶到嘌呤的替换。某些实验表明，转换的频率比颠换的要高。

可以确保连接的拓扑结构是二叉树。

类群数	所需比特数
4	52
5	97
6	152
7	219
8	298

表 2 类群数与比特数的关系

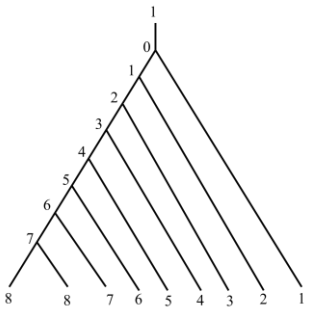

图 2 对于 10 个类群可能取得最大深度的无根树。图中标注了每个节点的深度。

对于 n 个类群的无根树，其内节点共有(n-2)个[5]。由图 2 可知，该树的最大深度为(n-2)。据此，我们定义一个包含 n 个叶子的集合 L，一个包含(n-2)个内节点的集合 I，定义参考节点 $u_0 \in I$，令 $V = L \cup I$。

接着，我们设计如下二值变量，当它们满足对应条件时为 1，否则为 0：

1) $x_{u,i}$：节点 u 的深度为 I，$u \in V/u_0, i \in \{1,2,\ldots,(n-2)\}$。

2) x_{uv}：节点 u 和节点 v 相连，$u,v \in V$。

3) $y_{u,p}$：内节点 u 的碱基为 p，$u \in I$。

可以看出，对于该模型所需比特数为 $(2n-3)\times(n-2)+(2n-2)^2+(n-2)\times 4$。表 2 展示了类群数与比特数的关系。进一步，我们构建如下约束：

1) **深度唯一性**：每个节点有且只有一个深度

$$\sum_{i=1}^{n-2} x_{u,i} = 1,\ \forall u \in V/u_0 \tag{5.1}$$

[5] 详见附录 A

2) **碱基唯一性**：每个内节点有且只有一个碱基：

$$\sum_{p} y_{u,p} = 1,\ \forall u \in I \tag{5.2}$$

3) **无环**：每个分支连接了深度为 i 和 i-1 的两个节点，基于我们的编码规则，我们将约束分成非参考节点之间的连接和参考节点与非参考节点之间的连接两部分[6]：

$$x_{uv}\left[2-\sum_{i=2}^{n-2}\left(x_{u,i-1}+x_{v,i}\right)\right]=0,\ \forall u,v \in V/u_0 \tag{5.3}$$

$$x_{u_0 v}\left(1-x_{v,1}\right)=0,\ \forall v \in V/u_0 \tag{5.4}$$

4) **连通性**：除参考节点外，其他任一节点 v 的入度为 1：

$$\sum_{u} x_{uv}=1,\ \forall v \in V/u_0 \tag{5.5}$$

5) 叶的出度为 0（确保集合 L 内都是叶节点）

$$\sum_{v \in V} x_{uv} = 0,\ \forall u \in L \tag{5.6}$$

6) 非参考节点的内节点的出度为 2，参考节点的出度为 3（确保集合 I 内都是内节点且约束该数为二叉树）

$$\sum_{v \in V} x_{uv} = 2,\ \forall u \in I/u_0 \tag{5.7}$$

$$\sum_{v \in V} x_{u_0 v} = 3 \tag{5.8}$$

接着，我们引入得分矩阵 $C_{p,q}$，其形式如表 3 所示。每个元素代表了不同的碱基替换类型在得分上的贡献。在重建过程中，该矩阵是一个先验的假设，可以根据具体研究对象的不同或已知实验结果进行调整。

		节点 2			
		A	T	C	G
节点 1	A	0	1.5	1.5	1
	T	1.5	0	1	1.5
	C	1.5	1	0	1.5
	G	1	1.5	1.5	0

表 3 得分矩阵 $C_{p,q}$。矩阵中每个元素表示节点 k 和节点 l 相连时，各个碱基组合的得分。例如节点 k 和节点 l 的碱基分别为 TC，那么它们相连时获得的得分为第二行第三列的数值，也就是 1。

接着，我们可以构建该模型的得分函数：

[6] 如果 n=3，那么该树只有一个内节点，该内节点必然为参考节点，此时该约束条件只有(5.4)。

$$\sum_{u\in I,v\in L}\sum_{p}C_{g(v),p}x_{uv}y_{u,p}+\sum_{u\in I,v\in I}\sum_{p,q}C_{p,q}x_{uv}y_{u,p}y_{v,q} \tag{5.9}$$

其中，$g(v)$ 为叶节点 v 的对应的碱基类型。

最终，我们可以把该问题转化为，在给定约束(5.1)(5.2)(5.3)(5.5)(5.6)(5.7)(5.8)下，求得分函数(5.9)的极小值。

(六) QUBO 模型和问题哈密顿量

根据文献[2]，我们可以使用上述模型转化为 QUBO 的形式，通过将惩罚以二次项的形式加入到得分函数中来构建目标函数，从而将问题转化为一个无约束条件下求解最小值的问题，即求解：

$$\min(\theta) \tag{6.1}$$

其中

$$\begin{aligned}\theta=&\sum_{u\in I,v\in L}\sum_{p}C_{g(v),p}x_{uv}y_{u,p}+\sum_{u\in I,v\in I}\sum_{p,q}C_{p,q}x_{uv}y_{u,p}y_{v,q}\\&+A\sum_{u\in V/u_0}\left(1-\sum_{i=1}^{n-2}x_{u,i}\right)^2+A\sum_{u\in I}\left(1-\sum_{p}y_{u,p}\right)^2\\&+A\sum_{u,v\in V/u_0}x_{uv}\left[2-\sum_{i=2}^{n-2}\left(x_{u,i-1}+x_{v,i}\right)\right]+A\sum_{v\in V/u_0}x_{u_0v}\left(1-x_{v,1}\right)\\&+A\sum_{v\in V/u_0}\left(1-\sum_{u}x_{uv}\right)^2+A\sum_{u\in L}\sum_{v\in V}x_{uv}\\&+A\sum_{u\in I/u_0}\left(2-\sum_{v\in V}x_{uv}\right)^2+A\left(3-\sum_{v\in V}x_{u_0v}\right)^2\end{aligned} \tag{6.2}$$

A 是一个可调整的参数，一般为得分函数估计值的 75%-150%[2]。

(七) QAOA 算法简介

QAOA[3]是近期发展的一种变分量子本征求解器(Variational Quantum Eigensolver, VQE)算法[7]，它可以高效地处理二元无约束的优化问题。通过将无约束的目标函数 θ 转化为问题哈密顿量 H_p，使得哈密顿量的本征值对应于目标函数的每一个取值，进而通过类似于量子退火算法的思路，将哈密顿量演化到系统基态上，从而找到基态能量，它对应于目标函数的极小值。在本节中，我们只简单说

[7] 对于变分量子算法，其量子线路会包含多个可调的参数，这个参数会根据测量结果进行优化，最终使测量结果收敛到期望的结果。

明一下问题哈密顿量的构造方法以及接入量子线路的方式，更多关于 QAOA 机制的详细介绍可以参考[4]。

首先介绍一下如何构建模型的问题哈密顿量 H_p。我们知道对 Pauli 矩阵 σ_z 来说，它有两个本征值±1，分别对应自旋为上和自旋为下两个本征态，如果我们把这两个本征态记作 $|0\rangle$ 和 $|1\rangle$，并把目标函数 θ 中每一个二值变量 $x^{(k)}$ 用 $\left(1-\sigma_z^{(k)}\right)/2$ 代替[8]来构成 H_p，那么对于一个 N 比特模型，每个比特串($k_0k_1\ldots k_{N-1}$)都满足：

$$H_p\left|k_0k_1\ldots k_{N-1}\right\rangle=\theta\left(k_0k_1\ldots k_{N-1}\right)\left|k_0k_1\ldots k_{N-1}\right\rangle \tag{7.1}$$

其中 N 是下面我们使用一个简单的例子来解释(7.1)。假设我们的目标函数是 $\theta=k_1k_2$，取值如表 4 所示。进一步，我们可以得到问题哈密顿量为 $H_p=\frac{1-\sigma_z^{(1)}}{2}\otimes\frac{1-\sigma_z^{(2)}}{2}$，那么显然它有四个本征态 $|0\rangle\otimes|0\rangle$，$|0\rangle\otimes|1\rangle$，$|1\rangle\otimes|0\rangle$，$|1\rangle\otimes|1\rangle$，前三个的本征值为 0，最后一个的本征值为 1，恰好与目标函数的取值对应。

k_1	k_2	θ
0	0	0
0	1	0
1	0	0
1	1	1

表 4 目标函数的取值示例

构建了 H_p 之后，接下来我们对 QAOA 的算法流程进行简要介绍[3, 4]：

1) 制备初始态 $|+\rangle^{\otimes N}$

2) 引入两类含参数 β_n, γ_n 算符 $U_B\left(\beta_n\right)=\exp\left(i\beta_nH_B\right)$ 和 $U_P\left(\gamma_n\right)=\exp\left(i\gamma_nH_P\right)$，其中 $H_B=\sum_{k=0}^{N-1}\sigma_x^{(k)}$，进一步我们定义态 $\left|\vec{\beta},\vec{\gamma}\right\rangle=\left|\beta_1,\beta_2,\ldots\beta_n,\gamma_1,\gamma_2,\ldots,\gamma_n\right\rangle$ 可

[8] 实际上，根据直积运算的性质，我们实际上需要把二值变量的乘积转化为直积的形式。

以由 $U_B(\beta_n)$和 $U_P(\gamma_n)$交替作用在初始态上得到：

$$\left|\beta_1,\beta_2,\ldots\beta_p,\gamma_1,\gamma_2,\ldots,\gamma_p\right\rangle=\prod_{n=1}^{p}U_P\left(\gamma_n\right)U_B\left(\beta_n\right)\left|+\right\rangle^{\otimes N} \tag{7.2}$$

3) 计算 $\left|\vec{\beta},\vec{\gamma}\right\rangle$ 的期望值 $M_p=\left\langle\vec{\beta},\vec{\gamma}\right|H_p\left|\vec{\beta},\vec{\gamma}\right\rangle$，然后通过经典优化器得到 $\min M_p$ 时对应的 $\left|\vec{\beta},\vec{\gamma}\right\rangle_{op}$，该态对应于问题哈密顿量的基态，因此也就是目标函数的近似最优解。

(八) 程序实现

目前，我们使用 *python* 的 *qiksit* 模块进行量子计算的模拟和测试。在此之前，我们使用 *Python* 的 *sympy* 模块将目标函数转化为了问题哈密顿量。进一步，我们构建了问题哈密顿量分别与经典最小本征值求解器模块和 QAOA 模块的接口并进行计算。另一方面，我们尝试基于 QuantumCircuit 模块构建出该模型对应的量子线路，并根据 QAOA 的算法流程设计了一套计算程序。程序中使用到的参考资料为[4-6]。对于三次项在量子线路中的实现方式在附录 B 中给出。

(九) 总结和展望

尽管我们构建了一套基于 QAOA 实现最简树重建的算法，然而目前无论是量子计算模拟器还是实际的量子计算机，都无法支持足够多的量子比特数。因此，我们需要通过一些方式来降低模型的比特数。从我们的建模中可以发现，为了约束我们解是一个二叉树的拓扑结构，我们使用了大量的比特数。因此，减少比特数的最佳方式就是在编码的过程中尽可能使我们的解是二叉树的拓扑结构。

目前，与系统发育树相关的量子算法已有相关的研究。Ellinas 等人基于量子行走发展了一套系统发育树推断的量子算法[7]。在该算法中，使用的比特数大约为 n，远远低于本文使用的比特数，也许在之后，我们可以进一步学习和了解相关的算法。Dinneen 等人在树包含问题中提出了 QUBO 形式的模型，并对其在 *D-Wave* 上的运行结果进行评估和分析[8]，在该工作最后提出了两个可拓展的问题，这两个问题也可以作为未来的研究方向。与此同时，目前还有基于模拟退火算法实现的最简树重建方法[9]，我们也可以基于此开发量子退火算法来实现最简树的重建。除了这些目前已有的量子算法，我们也可以在系统发育树问题中寻找新

的建模和算法[9]。

除了在算法上找到创新点，我们还可以尝试对其在生物问题上的意义进行挖掘。Zou 等人提出目前在系统发育基因组学的近展，通过使用不同基因片段建树，我们可以从基因树的冲突中分析出更多的进化信息[10]。然而，受限于目前比特数的限制，或许将该算法应用到实际的生物问题中还为时尚早，或许在算法的优化和评估上还有很多问题有待解决。

(十) 参考文献

1. Lucas, A., *Ising formulations of many NP problems.* 2014. **2**.
2. Glover, F., et al., *Quantum bridge analytics I: a tutorial on formulating and using QUBO models.* Annals of Operations Research, 2022. **314**(1): p. 141-183.
3. Farhi, E., J. Goldstone, and S.J.a.p.a. Gutmann, *A quantum approximate optimization algorithm.* 2014.
4. 百度量子计算研究所. *量子近似优化算法*. Available from: https://qulearn.baidu.com/textbook/chapter5/%E9%87%8F%E5%AD%90%E8%BF%91%E4%BC%BC%E4%BC%98%E5%8C%96%E7%AE%97%E6%B3%95.html.
5. *Quantum Approximate Optimization Algorithm — Qiskit 0.38.0 documentation*. 2022 2022/10/07/; Available from: https://qiskit.org/documentation/tutorials/algorithms/05_qaoa.html?highlight=quantum%20approximate%20optimization%20algorithm.
6. Qiskit. *Solving combinatorial optimization problems using QAOA*. Data 100 at UC Berkeley 2022 2022/07/06/; Available from: https://qiskit.org/textbook/ch-applications/qaoa.html.
7. Ellinas, D. and P.D. Jarvis, *Quantum channel simulation of phylogenetic branching models.* Journal of Physics A: Mathematical and Theoretical, 2019. **52**(11): p. 115601.
8. Dinneen, M.J., P.S. Ghodla, and S. Linz, *A QUBO formulation for the Tree Containment problem.* Theoretical Computer Science, 2022.
9. Richer, J.-M., E. Rodriguez-Tello, and K.E. Vazquez-Ortiz. *Maximum Parsimony Phylogenetic Inference Using Simulated Annealing*. 2013. Berlin, Heidelberg: Springer Berlin Heidelberg.
10. Zou, X.-H. and S. G. E., *Conflicting gene trees and phylogenomics.* Journal of Systematics and Evolution, 2008. **46**(6): p. 795.
11. Yang, Z., *Computational Molecular Evolution*. 2006, Oxford University Press.
12. Zhang, S.-X., et al., *Differentiable Quantum Architecture Search.* arXiv, 2020.

附录 A. 二叉树的枚举

[9] 详见附录 C

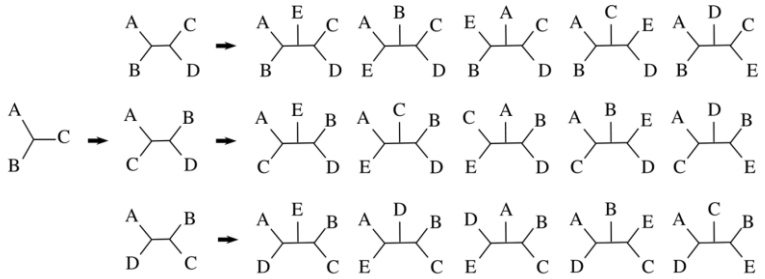

图 3 无根树的枚举，该图来自于[11].

无根二叉树的数量可以用逐步添加算法(stepwise addition algorithm)进行计算。

首先，对于叶的数量为 3 的无根树只有一种。该树共有 3 条分支。之后，我们加入新的叶，因为我们考虑的是二叉树，那么这个新外节点和与它相连的周分支会在原来的某一条分支上增加一个内节点并把原来的一条分支分成两条，使总分支数会增加 2，即共有$(3+2)$条分支。另一方面，由于原来有 3 条分支，因此一共有 3 种接入新外节点的方式，并且得到的树都互不等价。继续添加新的外节点，那么对每个外节数为 4 的树都会有$(3+2)$种接入方式，因此共有(3×5)种构成树的方式。通过这样的类推方式，我们可以得出，仅考虑无根树的拓扑结构时，n 个叶的无根树数目共有$(2n-5)!!$种，其中$n>3$。也就是说，给定 n 个类群，我们一共可以产生$(2n-5)!!$个互不等价的无根二叉树。

若是有根二叉树，那么我们可以让一个外节点数为$(n+1)$的树去掉某一个外节点就可以变为有根树，因此是$(2n-3)!!$种。

附录 B. 三次项的量子线路实现

由于：

$$
\begin{aligned}
e^{i\gamma Z_i\otimes Z_j\otimes Z_k} &= \cos\gamma\cdot\mathbb{I}+i\sin\gamma\cdot\left(Z_i\otimes Z_j\otimes Z_k\right)\\
&=\begin{pmatrix}
e^{i\gamma} & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & e^{-i\gamma} & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & e^{-i\gamma} & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & e^{i\gamma} & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & e^{-i\gamma} & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & e^{i\gamma} & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & e^{i\gamma} & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & e^{-i\gamma}
\end{pmatrix}\\
&=\left(|0\rangle\langle 0|\otimes|0\rangle\langle 0|+|1\rangle\langle 1|\otimes|1\rangle\langle 1|\right)\otimes\begin{pmatrix} e^{i\gamma} & 0\\ 0 & e^{-i\gamma}\end{pmatrix}\\
&\quad+\left(|0\rangle\langle 0|\otimes|1\rangle\langle 1|+|1\rangle\langle 1|\otimes|0\rangle\langle 0|\right)\otimes\begin{pmatrix} e^{-i\gamma} & 0\\ 0 & e^{i\gamma}\end{pmatrix}\\
&=\left(|0\rangle\langle 0|\otimes|0\rangle\langle 0|+|1\rangle\langle 1|\otimes|1\rangle\langle 1|\right)\otimes R_z(2\gamma)\\
&\quad+\left(|0\rangle\langle 0|\otimes|1\rangle\langle 1|+|1\rangle\langle 1|\otimes|0\rangle\langle 0|\right)\otimes R_z(-2\gamma)
\end{aligned}
\tag{B.1}
$$

因此，我们知道这个算符的作用是：qubit 0 和 qubit1 相同时，qubit2 执行 $R_z(2\gamma)$；qubit 0 和 qubit1 不同时，qubit 2 执行 $R_z(-2\gamma)=XR_z(2\gamma)X$。因为 CNOT 门的作用是 $|a,b\rangle\xrightarrow{CNOT_{0\to 1}}|a,a\oplus b\rangle$，且 $CNOT^2=\mathbb{I}$，因此线路图可以表示为图 4。

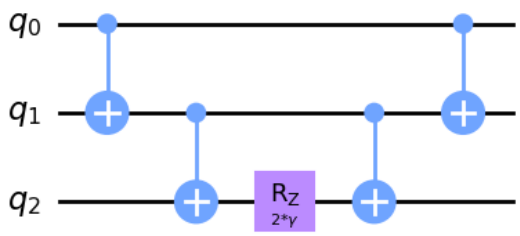

图 4 三次项对应的量子门

附录 C. 一个猜想

我们在下文给出一种通过最大似然法推断系统发育树的算法设计设想。

对于 VQE 算法，它所使用的是含参量子线路，经典优化器能够通过线路的测量结果对参数进行优化，使得最终求得量子线路对应的基态及其对应的线路参数。在 QAOA 算法中，目标函数的近似最优解对应于参数最优化的量子线路的

基态能量。除了本文介绍的最大简约法，最大似然法也是一种重要的重建系统发育树的方法。在给定叶节点的碱基类型后，我们选取一个核苷酸替换模型[10]，然后计算该树在所选核苷酸替换模型条件下，得到该基因分布的概率，该概率也就是该树的似然值，通过调整每一个分支长度，我们能找到该树的最大似然值，作为它的一个评分，最终我们枚举所有的树，选取评分最大的树即为我们所求的最优的系统发育树。启发于[7, 12]，根据最大似然法推断树的过程以及 VQE 算法的特点，我们可以设想一种量子线路的结构，该线路的初始态对应于每个叶节点的碱基；每一个比特的演化对应碱基的反向演化；定义一组含参量子门对应于每个进化事件，而参数对应于进化的分支长度。该对应关系如图 5 所示。对于最大似然法而言，对于每个系统发育树的最大似然值的求解是一个困难的计算过程。因此，该算法可能可以体现一定程度上的加速。实际上我们将该树的似然值转变为出现该量子线路以及其对应结果的似然值，那么根据[12]，我们或许能够找到一种方式来加速这个计算最大似然值得过程。

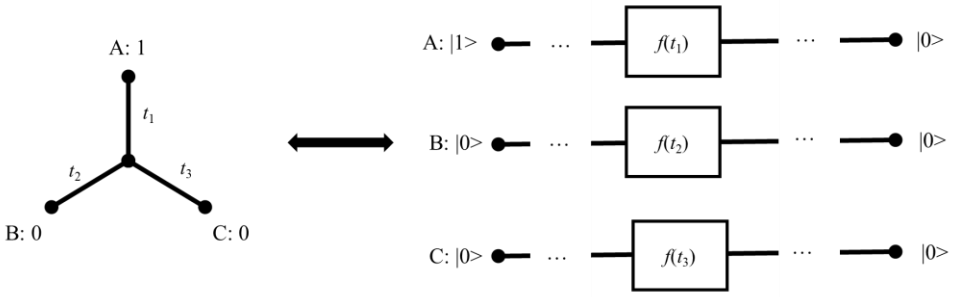

图 5 系统发育树和量子线路的对应的一个示例。为了叙述方便，碱基只考虑二态的标记。

[10] 该模型告诉我们各个碱基之间的替换速率，进而我们可以把不同节点间的替换概率表示为一个关于分支长度的函数。