面向量子计算的编码无关的优化问题表述

Federico Dominguez, [1, *] Josua Unger, [2] Matthias Traube, [1] Barry Mant, [2]
Christian Ertler, [1] 和 Wolfgang Lechner[1, 2, 3, †]
[1] *Parity Quantum Computing Germany GmbH, Ludwigstraße 8, 80539 Munich, Germany*
[2] *Parity Quantum Computing GmbH, A-6020 Innsbruck, Austria*
[3] *Institute for Theoretical Physics, University of Innsbruck, A-6020 Innsbruck, Austria*
（日期：2023年2月9日）

我们提出了一种面向量子计算的优化问题表述，该表述独立于编码和硬件。借助这一广义方法，我们提供了一个内容丰富的优化问题库及其各种衍生自旋编码。我们还确定了可作为构造工具包、用于构建这些自旋哈密顿量的常见构件。这为任意离散优化问题的哈密顿量全自动构造铺平了道路。问题表述所具有的这种自由度，是针对不同硬件平台定制最优自旋哈密顿量的关键一步。

arXiv:2302.03711v1 [quant-ph] 7 Feb 2023

目录

I. 引言 2
II. 定义与记号 3
III. 编码库 3
A. 二进制编码 4
B. 格雷编码 5
C. 独热编码 5
D. 畴壁编码 5
E. 一元编码 6
F. 分块编码 6
IV. 奇偶架构 7
V. 编码约束 8
VI. 与编码无关的表述 10
VII. 问题库 11
A. 划分问题 11
1. 聚类问题 13
2. 数划分问题 13
3. 图着色 13
4. 图划分 14
5. 团覆盖 15
B. 受约束子集问题 15
1. 团 15
2. 极大独立集 16
3. 集合打包 16
4. 顶点覆盖 16
5. 最小极大匹配 16
6. 集合覆盖 17
7. 背包问题 17
C. 排列问题 17
1. 哈密顿回路 18
2. 旅行商问题（TSP） 18
3. 机器调度 18
4. 护士排班问题 18
D. 实变量问题 19
1. 金融崩盘问题 20
2. 连续黑箱优化 20
3. 具有实参数的护士排班 21
E. 其他问题 21
1. 综合征译码问题 22
2. k-SAT 22
3. 矩阵乘法改进 22
VIII. 构件概要 23
A. 辅助函数 24
1. 元素计数 24
2. 阶跃函数 24
3. 最小化集合的最大元素 24
4. 变量比较 24
B. 约束 24
1. 所有（相连的）变量均不相同 25
2. 使用价值 α 25
3. 不等式 25
4. 约束保持驱动项 25
C. 问题特定表示 25
1. 模2线性规划 25
2. 布尔函数表示 26
D. 元优化 26
1. 辅助变量 26
2. 代价函数项的优先级排序 26
3. 问题转换 27
第九节：结论与展望 27
X. 致谢 28
参考文献 28

* f.dominguez@parityqc.com
† wolfgang@parityqc.com

I. 引言

离散优化问题几乎遍布人类活动的所有领域，其中许多问题已知是 NP难。此类离散优化问题的目标，是在离散变量集合 v_k 上，求实值函数 $f(v_0, \ldots\ldots, v_{N-1})$（即代价函数）的最小值。搜索空间受硬约束限制，这些约束通常以等式的形式表示，例如 $g(v_0, \ldots\ldots, v_{N-1}) = 0$，或以不等式的形式表示，例如 $h(v_0, \ldots\ldots, v_{N-1}) > 0$。

除了采用经典启发式方法 [1, 2] 和机器学习方法 [3] 来解决这些问题，人们也越来越关注应用量子计算。实现这一目标的一大类方法，是首先将代价函数 f 编码到哈密顿量 H 中，使得哈密顿量 H 的一个子集特征向量（尤其是基态）表示 f 定义域中的元素，而特征值则分别表示 f的取值:

$$H\,|v_0,\ldots,v_{N-1}\rangle = f(v_0,\ldots,v_{N-1})\,|v_0,\ldots,v_{N-1}\rangle\,. \quad (1)$$

在这种编码中， H 的基态就是该优化问题的解。得到哈密顿量形式的重构后，可以使用各种量子算法来寻找该哈密顿量的基态，包括绝热量子计算 [4] 和变分方法，例如量子-经典混合量子近似优化算法（QAOA）[5] ，或其推广形式，如量子交替算子试探形式 [6]。在硬件方面，这些算法可以运行于基于门的量子计算机、量子退火机或专用伊辛机 [7]上。

在当前文献中，几乎所有用于优化的哈密顿量都被表述为二次无约束二元优化（QUBO）问题 [8]。QUBO 的成功反映了硬件受到的严格限制：硬件无法实现高于二次的相互作用。此外，采用硬约束动态实现的量子算法 [9, 10]需要驱动项，而驱动项可能难以设计，也难以在量子计算机上实现。因此，硬约束通常通过能量惩罚项纳入 QUBO 哈密顿量。QUBO 的普及也提升了独热编码的受欢迎程度。独热编码是一种将（离散）变量映射到自旋算子特征值的特定方式 [11], ，这种编码能形成低阶相互作用的哈密顿量，因此尤其适用于 QUBO 问题。

然而，近年来已有研究提出了颇具吸引力的 QUBO 和独热编码替代方案。越来越多的平台正在探索高阶相互作用 [12–20], ，而奇偶架构 [21–23]（LHZ架构 [24] 的一种推广）则可将任意阶相互作用映射到仅需局域连接性的量子比特上。约束的动态实现也得到了进一步研究[6, 25–27], ，其中包括近似驱动项的设计[28] ，以及在奇偶架构内编译带约束的问题。

此外，实验结果表明，替代编码优于传统的独热方法 [30–33]。显然，还需要进一步探索哈密顿量的替代表述方式；但如果哈密顿量已用独热编码表示为 QUBO，切换到其他表述方式并非易事。因此，用于探索不同表述方式的自动化工具将大有裨益。

我们在此提供一个问题库，旨在推动超越 QUBO 和独热编码的哈密顿量表述。我们重新审视了文献中的常见问题，并采用与编码无关的表述方式重新表述，也就是说，这些问题可以轻松地使用任意自旋编码进行编码。我们还识别并单独列出问题可能包含的约束，与代价函数区分开来，以便也能轻松探索约束的动态实现。近期也有人提出将与编码无关的表述作为问题的中间表示阶段 [28] ，并通过一些应用场景加以示例。在此，我们将这一方法扩展到 20 多个问题，并总结了可采用的最常见编码。本问题库还涉及另外两个子目标:

- 元参数／选择：我们介绍并评述将数学表述中的优化问题映射为自旋哈密顿量时可以进行的最重要选择。这主要包括编码，它们会极大地影响对优化的计算成本和性能至关重要的关键特性，同时也包括自由元参数或辅助变量的使用。所有这些自由度最终都源于这样一个事实：最优解通常仅编码在基态中，而其他低能本征态则编码了对最优解的良好近似。此外，进行近似可能很方便，使得解不完全对应于基态，而是对应于另一个低能态，例如 [34]中所报告的情况。

- （部分）自动化：通常，每个问题都需要单独进行评估。由此得到的代价函数并不一定是唯一的，而且目前还没有已知的自动创建 H的简单方法。通过提供代价函数构件的集合以及用于选择参数的启发式方法，可以辅助代价函数和约束的创建。这使得能够以与编码无关的方式对问题进行一般性表述，并且可以在这一中间阶段完成部分参数选择和性能分析。

在实践中，许多优化问题并非纯粹的离散问题，而是涉及实值参数和变量。

因此，第VIID节还讨论了将实值问题编码为离散优化问题（离散化）这一中间步骤。

本文结构如下。第 II 节介绍全文使用的记号，第 III 节列出各种编码。第 VI 节则说明如何将本文中的优化问题转换为可由量子计算机求解的形式，并通过两个具体示例加以说明。第 VII 节收录了按多个类别划分的优化问题，第 VIII 节列出了构造这些问题时使用的构件。这些构件也有助于处理许多其他问题。最后，第 IX 节总结研究结果，并展望未来项目。

II. 定义和记号

本节给出一些基本定义，并确定全文将使用的记号。

实数的一个离散集是一个可数子集$U \subset \mathbb{R}$，且没有聚点。离散集将用大写拉丁字母表示，但字母 G 除外，该字母保留用于表示图。离散变量是取值范围为离散集 R 的变量。离散变量将用小写拉丁字母表示，通常为 v 或 w。离散变量 v 所取值的离散集 R 称为其取值范围。 R 的元素用小写希腊字母表示。

如果离散变量的取值范围为 $\{0,1\}$，我们称其为二进制编码或布尔变量。二进制变量用字母 x表示。类似地，取值范围为 $\{-1,1\}$ 的变量称为自旋变量，字母 s 专用于表示自旋变量。二元变量 x 与自旋变量 s之间存在可逆映射：

$$x \mapsto s := 2x - 1\,. \tag{2}$$

对于取值范围为 R 的变量 v，我们将取值指示函数定义为

$$\delta_v^\alpha = \begin{cases} 1 & \text{if } v = \alpha \\ 0 & \text{if } v \neq \alpha, \end{cases} \tag{3}$$

其中 $\alpha \in R$。

我们还将考虑连续变量的优化问题。如果变量 v 的取值范围由某个 $d \in \mathbb{Z}_{>0}$的 $\mathbb{R}^d$ 给出，则称其为连续变量。

优化问题 O 是一个三元组（V， f， C），其中

1. $V := \{v_i\}_{i=0,\cdots,N-1}$ is a finite set of variables.
2. f: = f（v_0， $\cdots$， v_{N-1}）是一个实值函数，称为目标函数或代价函数。
3. $C = \{C_i\}_{i=0,\ \cdots,\ l}$ 是一个由约束 C_i组成的有限集。约束 C 要么是一个方程

$$c(v_0, \cdots, v_{N-1}) = k \tag{4}$$

对于某个 $k \in \mathbb{R}$ 和一个实值函数$c(v_0$， $\cdots$， $v_{N-1})$，要么是不等式

$$c(v_0, \cdots, v_{N-1}) \leq k\,. \tag{5}$$

优化问题 $O = (V$， f， $C)$ 的目标是在 y_{ex} 处满足所有约束的条件下，求出 f 的一个极值 y_{ex} 。

离散优化问题通常可以用图或超图来表述。图 是一个二元组 $(V$， $E)$，其中 V 是顶点或节点的有限集，$E \subset \{\{v_i, v_j\} \,|\, v_i \neq v_j \in V\}$ 是边的集合。元素 $\{v_i, v_j\} \in E$ 称为连接顶点 v_i 和顶点 v_j 的一条边 。请注意，按此方式定义的图既不能有自环（即从同一顶点出发并终止于该顶点的边），也不能在同一对顶点之间存在多条边。给定一个图 $G = (V$， $E)$，其邻接矩阵是一个对称的二进制编码 $|V| \times |V|$ 矩阵 A，其元素为

$$A_{ij} = \begin{cases} 1, & \text{if } \{v_i, v_j\} \in E, \\ 0, & \text{else}\,. \end{cases} \tag{6}$$

超图是图的一种推广，其中允许边与两个以上的顶点相邻。也就是说，超图 是一个二元组 $H = (V$， $E)$，其中 V 是顶点的有限集，并且

$$E \subseteq \bigcup_{i=1}^{|V|} \Big\{ \{v_{j_1}, \cdots, v_{j_i}\} \,|\, v_{j_k} \in V \text{ and } v_{j_k} \neq v_{j_\ell} \quad \forall\, k, \ell = 1, \cdots i \Big\} \tag{7}$$

是超边的集合。

在整篇论文中，我们将「量子比特」一词专用于实际的物理量子比特。要将编码无关哈密顿量转化为量子程序，需要将二进制或自旋变量转换为作用于相应量子比特上的泡利 Z 矩阵。

III. 编码库

对于许多重要问题，代价函数和问题约束都可以用两个基本构件来表示：整数变量的值 v 和值指示器 δ_v^α，定义见方程（3）。用这些构件来表示时，哈密顿量会更加紧凑，而且可以识别出许多不同问题中反复出现的项。此外，在这一阶段尚未涉及量子算符：编码无关哈密顿量只是用离散变量表示的代价函数，这使更广泛的受众能够更轻松地接触量子优化。变量的编码方式和量子算法的选择可以留待后续阶段完成。

构件用伊辛算符表示的方式取决于我们选择的编码。一种

编码是一个函数，它将σ_z 算符的特征向量与离散变量 v 的特定值相关联：

$$|s_0,\ldots,s_{N-1}\rangle \to v, \quad s_i=\pm 1, \tag{8}$$

其中，自旋变量 s_i 是 $\sigma_z^{(i)}$算子的特征值。编码通常也以二进制变量 $x_i=0,1$ 表示，它们与伊辛变量 $s_i=-1,1$ 的关系如方程（2）所示。

图 1总结了各种编码。有些编码是稠密的，也就是说，每个量子态 $|s_0,\ldots,s_{N-1}\rangle$ 都对变量 v的某个取值进行编码。另一些编码则是稀疏的，因为只有部分可能的量子态是有效态。对于每个采用稀疏编码的变量，在哈密顿量中加入一个核心项即可生成有效子集。一般而言，稠密编码所需的量子比特较少，但稀疏编码的值指示器表达式更简单 δ_v^α，因此有利于避免高阶相互作用。这是因为 δ_v^α 只需检查较少数量的量子比特态，就能判断变量 v 是否取值为 α；而稠密编码则需要知道寄存器中每个量子比特的状态 [28, 33]。

A. 二进制编码

二进制编码使用二进制表示来编码整数变量。给定整数变量$v\in[1,2^D]$，我们使用 D 个二进制变量 $x_i\in\{0,1\}$ 来表示 v:

$$v=\sum_{i=0}^{D-1}2^i x_i+1. \tag{9}$$

值指示器 δ_v^α可以用通用表达式表示

$$\delta_v^\alpha=\prod_{i\neq\alpha}\frac{v-i}{\alpha-i}, \tag{10}$$

该表达式适用于所有编码。利用 α 的值由比特串（$x_{\alpha,\,0}$ ……，$x_{\alpha,\,D-1}$）表示这一点，可以用布尔变量 x_i 写出δ_v^α的表达式。值指示器 δ_v^α用于检查二进制变量 x_i 是否等于 $x_{\alpha,\,i}$，以判断变量 v 的值是否为 α。我们注意到

$$x_1\left(2x_2-1\right)-x_2+1=\begin{cases}1 & \text{if } x_1=x_2\\ 0 & \text{if } x_1\neq x_2\end{cases} \tag{11}$$

因此，我们写作

$$\begin{aligned}\delta_v^\alpha&=\prod_{i=0}^{D-1}\left[x_i\left(2x_{\alpha,i}-1\right)-x_{\alpha,i}+1\right]\\&=\prod_{i=0}^{D-1}s_i\left(x_{\alpha,i}-\frac{1}{2}\right)+\frac{1}{2},\end{aligned} \tag{12}$$

其中

$$s_i=2x_i-1 \tag{13}$$

是对应的伊辛变量。因此，δ_v^α 中相互作用项的最高阶数随 D 线性增长，且有 $\binom{D}{k}$ 个 k阶项。项的总数为 $\sum_k\binom{D}{k}=2^D$。由于 D 个二进制变量编码一个取值为 2^D 的变量，因此，二进制编码中值指示器所需的相互作用项数量随变量的大小线性增长。

如果 $v\in\{1,\ldots,K\}$ 具有 $2^D<K<2^{D+1}$，$D\in\mathbb{N}$，那么我们需要 $D+1$ 个二进制变量来表示 v，并且会有 $2^{D+1}-K$ 个无效量子态，它们不代表变量 v 的任何值。无效状态的集合为

$$R=\left\{|\mathbf{x}\rangle\,\middle|\,\sum_{i=0}^{D}2^i x_i+1>K\right\}. \tag{14}$$

为了排除 R中的量子态，我们强制要求 $v\leq K$，这可以通过在哈密顿量中添加一个核心项 H_{core} 来实现

$$H_{\text{core}}=\sum_{\alpha=K+1}^{2^{D+1}}\delta_v^\alpha, \tag{15}$$

或者施加求和约束

$$c_{\text{core}}=\sum_{\alpha=K+1}^{2^{D+1}}\delta_v^\alpha=0 \tag{16}$$

核心项会惩罚任何表示变量无效值的状态 v。由于核心项会引入额外的能量尺度，当 $K\neq 2^D,\ \ D\in\mathbb{N}$时，性能可能会下降。在某些情况下，例如背包问题，无效状态的惩罚项可以纳入代价函数，因此无需添加核心项或约束 [32]（另见第 VIII B3节）。

对于也可能取负值的变量（例如 $v\in\{-K,-K+1,...,K'-1,K'\}$），在经典计算中通常会使用额外的一位来编码数值的符号。不过，这对我们的目的而言可能并非最佳选择，因为一次自旋翻转就可能大幅改变数值，而我们并不假设具备完全容错能力。对负值有一种更合适的编码方式。我们来考虑一下二进制编码。例如，我们可以不采用常规编码，而是直接平移这些数值

$$v=\sum_{i=0}^{D-1}2^i x_i+1-K, \tag{17}$$

其中 $2^{D-1}<K+K'\leq 2^D$。值指示函数的表达式保持不变，只需调整价值 α 的编码。与使用符号位相比，另一项优势是，可以更高效地对非零点对称的取值范围（$K\neq K'$）进行编码。将变量平移$-K$ 的相同方法也可用于其他编码。

B. 格雷编码

在二进制表示中，单次自旋翻转可能导致 v的价值发生剧烈变化，例如 $|1000\rangle$ 表示$v=9$，而 $|0000\rangle$ 表示 $v=1$。为避免这种情况，格雷编码会重新排列二进制表示，使 v 的任意两个连续值之间总是只相差一次自旋翻转。若将整数变量的所有可能取值 $v\in[1,2^D]$ 按垂直顺序排列，则这种使用D 个布尔变量的编码可描述如下：对于第i个布尔变量（即从右往左数的第 i列），序列以 2^{i-1} 个零开始，随后交替出现 2^i 1和 2^i 0。例如，考虑

1:
0000 2:
0001 3:
0011 4:
0010 5:
0110 6:
0111...

其中 $D=4$。每行布尔变量左侧对应的价值为 v。例如，如果我们跟踪最右侧的布尔变量，会发现它对于第一个价值从 $2^{1-1}=1$ 零开始，对于第二和第三个价值为 2^1 一，对于第三和第四个价值为 2^1 零，依此类推。

值指示函数和核心项保持不变，唯一的区别在于，例如 α 在方程（12）的对应形式中也必须采用格雷编码。

这种编码对于量子算法的一项优势是，单自旋翻转不会导致代价函数发生巨大变化，因此可以选择较小的系数（参见第 VIII节的讨论）。

C. 独热编码

独热编码是一种稀疏编码，使用 N 二进制变量 x_α 来编码一个取 N 个值的变量 v。该编码由其变量指示函数定义：

$$\delta_v^\alpha = x_\alpha, \tag{18}$$

这意味着，当 $x_\alpha=1$时，$v=\alpha$。v 的值由下式给出

$$v=\sum_{\alpha=0}^{N-1}\alpha x_\alpha. \tag{19}$$

具有物理意义的量子态是其中单个量子比特处于状态 1 的状态，因此动力学必须被重新限制在由以下内容定义的子空间内

$$c_{\text{core}}=\sum_{\alpha=0}^{N-1}x_\alpha-1=0. \tag{20}$$

施加此求和约束的一种方法是将其编码为能量惩罚项，并在哈密顿量中加入一个核心项：

$$H_{\text{core}}=\left(1-\sum_{\alpha=0}^{N-1}x_\alpha\right)^2, \tag{21}$$

当且仅当只有一个 x_α 不为零时，其能量最小。

施加核心项的另一种方法可以表述如下。在布尔函数的构件表示（第VIIIC 2节）中，异或(XOR)$(x_0,\ x_1)$ 函数的表示为异或(XOR)$(x_0,\ x_1)$ $=x_0+x_1-2x_0x_1$。用自旋变量表示为 $(1-s_0s_1)/2$。同样，串接异或函数可得

$$\text{XOR}(x_0,\text{XOR}(x_1,...\text{XOR}(x_{N-2},x_{N-1})...)=\frac{1}{2}\left(1-(-1)^{N-1}\prod_{i=0}^{N-1}s_i\right). \tag{22}$$

这可以与一个额外的惩罚项结合使用，以消除简并并执行独热检查，即检查 x_i 中是否恰有一个等于 1。例如，考虑 $N=3$。对于恰有一个 x_i 为 1 的目标构型，XOR(x_0, XOR(x_1, x_2)) $=1$；但当三个值全为 1 时也是如此。事实上，一般而言，任意奇数个 1 都会得到 1。定义

$$H_{\text{core}}=-\text{XOR}(x_0,\text{XOR}(x_1,...\text{XOR}(x_{N-2},x_{N-1})...)+\alpha\sum_{i=0}^{N-2}x_i+1-\alpha \tag{23}$$

与 $0<\alpha<1$ 则是一个有效的独热检查，它需要 N 项（用自旋变量表示时）[1]。另一方面，方程 (20) 中的标准独热检查需要 $\mathcal{O}(N^2$项，且最高为二次项。

D. 畴壁编码

这种编码利用伊辛链中畴壁的位置来表示变量 v 的值。如果将 $N+1$ 自旋链的两个端点固定在相反的状态，

[1] 请注意，我们只需要求和中的 $N-1$，而不是 N 项，因为任何使异或链返回 1 且不是有效独热字符串的配置，其 onesinit 至少为 3。

链中就至少存在一个畴壁。由于铁磁伊辛链的能量只取决于畴壁的数量，而与它们的位置无关，因此端点固定为相反状态的 $N+1$ 自旋链有 N 个可能的基态，具体取决于唯一一个畴壁的位置。

变量的编码方式 $v=1,\ldots,N$ 采用畴壁编码需要核心哈密顿量 [30]:

$$H_{\text{core}}=-\left[-s_1+\sum_{\alpha=1}^{N-2}s_\alpha s_{\alpha+1}+s_{N-1}\right]. \tag{24}$$

由于链的固定端点无需自旋表示（$s_0=-1$ 和 $s_N=1$），$N-1$ 个 Ising 变量 $\{s_i\}_{i=1}^{N-1}$ 足以编码具有 N个值的变量。H_{core} 的最小能量为 $2-N$，因此核心项也可以用求和约束来编码:

$$c_{\text{core}}=-\left[-s_1+\sum_{\alpha=1}^{N-2}s_\alpha s_{\alpha+1}+s_{N-1}\right]=2-N. \tag{25}$$

变量指示函数可确认位置 α是否存在畴壁:

$$\delta_v^\alpha=\frac{1}{2}\left(s_\alpha-s_{\alpha-1}\right), \tag{26}$$

其中 $s_0\equiv-1$ 和 $s_N\equiv1$，变量 v 可写为

$$\begin{aligned} v&=\sum_{\alpha=1}^{N}\alpha\delta_v^\alpha \\ &=\frac{1}{2}\left[(1+N)-\sum_{i=1}^{N-1}s_i\right]. \end{aligned} \tag{27}$$

采用畴壁编码的量子退火实验表明，与独热编码相比，性能有显著提升 [31]。这部分是因为所需的搜索空间更小（$N-1$ 个量子比特，而 N 个量子比特对应一个具有 N 个值的变量），但也因为畴壁编码产生了更平滑的能量景观：在独热编码中，两个有效态之间的最小汉明距离为二，而在畴壁编码中，这一距离为一。这意味着独热编码中的每个有效量子态都是局部最小值，周围环绕着由方程（21）的核心能量产生的能垒。因此，畴壁编码问题中的动力学在退火过程中较晚冻结，因为从一个态转移到另一个态只需一次自旋翻转。

E. 一元编码

在这种情况下，v 的值由取值为 1 的二进制变量的数量来编码:

$$v=\sum_{i=0}^{N-1}x_i, \tag{28}$$

因此，编码一个取值为 N 的变量需要 $N-1$ 个二进制变量 x_i 。一元编码不需要核心项，因为每个量子态都是有效状态。然而，这种编码并不唯一，因为 v 的每个值都有多种表示形式。

一元编码（以及所有稠密编码）的一个缺点是，必须利用所有二进制变量的信息才能确定 v 的值。值指示器 δ_v^α 为

$$\delta_v^\alpha=\prod_{i\neq\alpha}\frac{v-i}{\alpha-i}, \tag{29}$$

其中涉及 2^N 个相互作用项。这种扩展方式完全不理想，因此一元编码可能只适用于不需要值指示器δ_v^α、只需要变量值 v 的变量。

使用数字退火器对背包问题进行性能比较后发现，一元编码的性能可能优于二进制编码和独热编码，而且所需的能量尺度更小 [32]。一元编码性能较高的原因仍在研究中，但人们认为冗余发挥了重要作用，因为它有助于退火器找到基态。对于畴壁编码 [35], ，两个有效态之间的最小汉明距离（即从一个有效态变换到另一个所需的自旋翻转次数）也可能解释一元编码为何性能更好。

F. 分块编码

也可以结合不同的方法，在稀疏编码与稠密编码之间取得平衡 [33]。分块编码基于 B 块，每个块由g 个二进制变量组成。与独热编码类似，分块编码的有效态是只有一个块包含非零二进制变量的状态。块 b中的二进制变量 $\{x_{b,i}\}_{i=0}^{g-1}$ 使用二进制编码、格雷编码或一元编码等稠密编码来定义块价值w_b。例如，如果 w_b 使用二进制编码，则有

$$w_b=\sum_{i=0}^{g-1}2^i x_{b,i}+1. \tag{30}$$

离散变量 v 由激活块 b及其对应的块价值 w_b定义，

$$v=\sum_{b=0}^{B-1}\sum_{\alpha=1}^{2^g-1}v\left(b,w_b=\alpha\right)\delta_{w_b}^\alpha, \tag{31}$$

其中，$v\left(b,w_b=\alpha\right)$ 是与量子态相关联的离散值，该量子态具有处于激活状态的块 b，且块值为 α；$\delta_{w_b}^\alpha$ 是一个值指示器，只需检查块 b 中二进制变量的值。对于每个块，都有 2^{g-1} 个可能的值（假设采用格雷编码或二进制编码

对该块进行编码），因为不允许全零状态（否则块 b 将处于非激活状态）。如果块值 w_b 使用一元编码进行编码，则有 g 个可能值。$\delta^{\alpha}_{w_b}$ 的表达式取决于编码方式，与相应编码章节中已介绍的表达式相同。

变量 v 的值指示器是相应的块值指示器。假设离散值 v_0编码在块 b 中，其块变量为 $w_b = \alpha$:

$$v_0 = v\left(b, w_b = \alpha\right). \tag{32}$$

那么，值指示器 $\delta_v^{v_0}$为

$$\delta_v^{v_0} = \delta^{\alpha}_{w_b}. \tag{33}$$

需要一个核心项，以确保只有单个块中的量子比特才能处于激发态。定义$t_b = \sum_i x_{i,\ b}$，核心项为

$$H_{\text{core}} = \sum_{b \neq b'} t_b t_{b'}, \tag{34}$$

或者，写成求和约束的形式，

$$c_{\text{core}} = \sum_{b \neq b'} t_b t_{b'} = 0. \tag{35}$$

H_{core} 的最小值为零。如果两个块 b 和b' 的二进制变量取值为一，那么 $t_b t_{b'} \neq 0$，且 H_{core} 对应的本征态不再是基态。

IV. 奇偶架构

含噪中等规模量子（NISQ）设备的硬件限制十分严格 [36] ，因此稀疏编码（尤其是独热编码）已成为问题建模的标准方法。这主要是因为在这些编码中，基本构件（取值和取值指示符）关于自旋变量的阶数为线性或二次。量子比特平台的低连通性要求哈密顿量采用 QUBO 形式，而将高阶相互作用转换为 QUBO [8]的成本很高。然而，选择不同的编码可以显著提升量子算法的性能 [30–33], 因为明智地选择编码能够缩小搜索空间，或生成更平滑的能量景观。

编码之间的这种差异，体现在改变变量的取值并使其变为另一个有效值所需的物理量子比特自旋翻转次数上 [35]。如果这一次数大于一次，那么被高成本惩罚的无效态会将局部极小值分隔开来，从而可能阻碍优化的性能。另一方面，这样的能量景观或许能够提供某种抵御错误的保护（另见参考文献 [23] 和参考文献 [37]）。此外，算法的其他基本方面，例如电路深度和能量尺度在 QUBO 之外也可以得到显著改进 [29, 38, 39]，这促使我们寻找区别于当前基于 QUBO 的哈密顿量的其他表述方式。奇偶架构是一种用于求解量子优化问题的范式 [21, 24] ，它不依赖于 QUBO 形式，因此为哈密顿量的构造提供了广泛的选择。该架构基于奇偶变换，将哈密顿量重新映射到一个二维网格中，仅需量子比特之间的局部连接。由于不存在长程相互作用，量子算法具有很高的并行化能力，同时也无需使用成本高昂且耗时的交换门，从而有助于克服量子计算面临的两个主要障碍：量子比特有限的相干时间，以及量子寄存器中量子比特之间较差的连接性。

Parity 变换会针对（原始）逻辑哈密顿量中的每个相互作用项创建一个奇偶量子比特:

$$J_{i,j,\ldots}\sigma_z^{(i)}\sigma_z^{(j)}\cdots \to J_{i,j,\ldots}\sigma_z^{(i,j,\ldots)}, \tag{36}$$

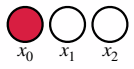
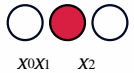
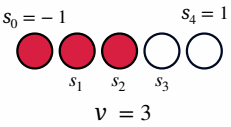
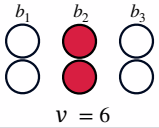
其中，相互作用强度 $J_{i,\ j,\ldots\ldots}$ 现在是奇偶量子比特 $\sigma_z^{(i,j,\ldots)}$ 的局域场。这便于处理高阶相互作用，并使问题建模摆脱 QUBO 方法。通过添加三体和四体约束，并将其放置在二维网格上，使约束中只涉及相邻量子比特，从而确保原始逻辑问题与奇偶变换后的问题等价。将逻辑问题映射到 Parity 芯片的规则网格上，例如可通过 Parity 编译器 [21]实现。虽然对问题进行 Parity 编译可能需要更多物理量子比特，但网格上相互作用的局域性使量子算法具有更高的并行化能力。因此，可以用更少的量子门 [39]实现常数深度算法 [22, 40] 。

下面的玩具示例（见图 2）展示了当原始哈密顿量包含高阶相互作用时，如何使用更少的量子比特求解经过奇偶变换的哈密顿量。给定逻辑哈密顿量

$$\begin{aligned} H =& \sigma_z^{(1)}\sigma_z^{(2)} + \sigma_z^{(2)}\sigma_z^{(4)} \\ &+ \sigma_z^{(1)}\sigma_z^{(5)} + \sigma_z^{(1)}\sigma_z^{(2)}\sigma_z^{(3)} + \sigma_z^{(3)}\sigma_z^{(4)}\sigma_z^{(5)}, \end{aligned} \tag{37}$$

相应的 QUBO 表述需要 $5+2=7$ 个量子比特，其中包括两个用于分解三体相互作用的辅助量子比特，且二体相互作用总数为 14。将 QUBO 问题嵌入量子硬件可能需要额外的量子比特和相互作用，具体取决于所选架构。相比之下，经过奇偶变换的哈密顿量仅由六个奇偶量子比特、局域场以及两个作用于相邻量子比特之间的四体相互作用构成。

对于一个优化问题，最佳的哈密顿量表述方式尚不明确。答案

编码	可视化示例	变量值	值指示器	核心项
二进制编码	 $v = 2$	$v = \sum_{i=0}^{\log_2(N)-1} 2^i x_i + 1$	$\delta_v^\alpha = \prod_{i=0}^{D-1} \left[x_i \left(2x_{\alpha,\,i} - 1 \right) - x_{\alpha,\,i} + 1 \right]$	无核心术语（稠密编码）[1]
独热	 $v = 1$	$v = \sum_{i=0}^{N-1} i x_i$	$\delta_v^\alpha = x_\alpha$	$\left(\sum_{d=0}^{N-1} x_d - 1 \right)^2$
一元编码	 $v = 2$	$v = \sum_{i=0}^{N-1} x_i$	$\delta_v^\alpha = \prod_{i \neq \alpha} \frac{v - i}{\alpha - i}$	无核心术语（稠密编码）
畴壁	 $v = 3$	$v = \frac{1+N}{2} - \sum_{i=1}^{N-1} \frac{s_i}{2}$	$\delta_v^\alpha = \frac{1}{2} \left(s_\alpha - s_{\alpha-1} \right)$	$\left(\sum_{\alpha=0}^{N-1} 秒_\alpha 秒_{\alpha+1} - N + 2 \right)^2$
块编码	 $v = 6$	$v = \sum_{b=0}^{B-1} \sum_{\alpha=1}^{2^g-1} v \ (b, w_b = \alpha) \ \delta_{w_b}^\alpha$	$\delta_v^{v_0} = \delta_{w_b}^\alpha$	$\left(\sum_{b \neq b'} \sum_{d,e} x_{d,\,b} x_{e,\,b'} \right)^2$

1：当 $\log_2 N \notin \mathbb{N}$ 时，可能需要惩罚项

图1. 常见离散变量编码的概述，分别采用二进制编码 x_i 或 Ising s_i 变量表示。每种编码都以特定方式表示变量的值 v 和值指示器 δ_v^α。如果寄存器中的每个量子态都表示 v的一个有效值，则称该编码为 *dense*。相反， *sparse encodings* 必须在哈密顿量中包含一个核心项，才能表示 v的值。与稠密编码相比，稀疏编码的值指示器表示更简单，但核心项的实现需要量子比特之间额外的相互作用项。哈密顿量的成本（按量子比特数量和相互作用项数量计）在很大程度上取决于所选编码，因此应针对每个具体问题进行评估。

可能在很大程度上取决于我们希望解决的具体使用场景，并且不仅要考虑所需量子比特的数量，还要考虑能量景观的平滑度，因为这会直接影响量子算法的性能 [41]。Parity 架构使我们能够探索超越标准 QUBO 的表述方式，而该库旨在促进对新哈密顿量表述方式的探索。

V. 编码约束

到目前为止，我们已经介绍了如何用二进制变量表示离散变量的可能编码。本节中，我们假设变量的编码已经选定，并说明如何实现与问题相关的硬约束。 $c(v_1, \ldots, v_N) = K$ 硬约束常见于优化问题中，它们会缩小搜索空间，使问题更难求解。我们考虑多项式约束，其形式为

$$c(v_1, \ldots, v_N) = \sum g_i v_i + \sum g_{i,j} v_i v_j + \sum g_{i,j,k} v_i v_j v_k + \ldots, \tag{38}$$

名约束

将离散变量 v_i 替换为任意编码后，该表达式仍为多项式。系数$g_i, g_{i,j}, \ldots$取决于问题及其约束。

一般来说，约束可以通过动态方式实现 [9, 10]（仅探索满足约束的量子态），也可以作为额外项加入 H_{core} 哈密顿量中，使得 H 的特征向量也是 H_{core} 的特征向量，而 H_{core} 的基态对应于 f 定义域中满足约束的元素。即使原问题没有约束，使用独热或畴壁等稀疏编码也会对量子变量施加硬约束。

稀疏编码产生的约束通常通过惩罚项 H_{core} 加入哈密顿量，以惩罚任何位于所需子空间之外的状态：

$$H_{\text{core}} = A \left[c(v_1, \ldots, v_N) - K \right]^2, \tag{39}$$

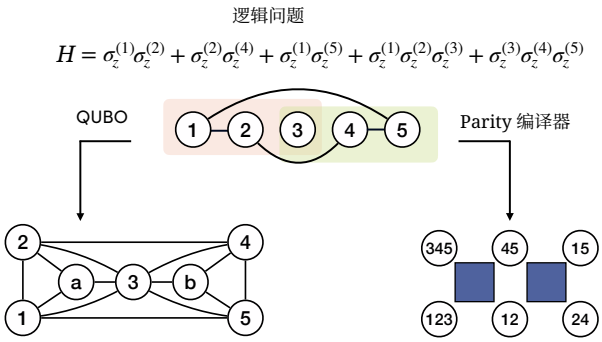


图2. 一个涉及高阶相互作用的示例玩具问题 H，展示了奇偶架构如何直接处理超出 QUBO 表述范围的哈密顿量。该问题由图顶部的超图表示。将其分解为 QUBO 形式时，需要 7 个量子比特和 14 个二体相互作用，此外还需根据嵌入方式增加量子比特开销。相比之下，奇偶编译器可以将该问题重新映射到一个二维网格中，仅需 6 个量子比特、局域场和四体相互作用，如蓝色方块所示。如果硬件是为奇偶架构设计的，则无需额外嵌入

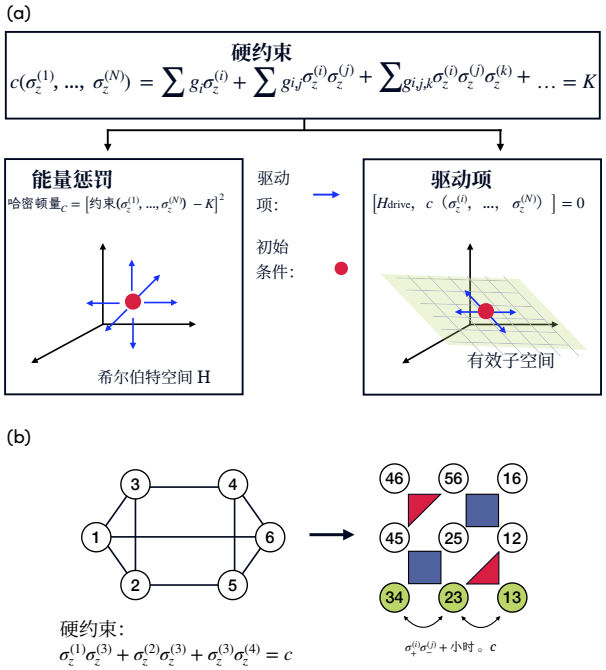


图 3. (a) 用于编码约束的决策树。约束可以作为能量惩罚项纳入问题的哈密顿量，也可以通过选择能够保持所需条件的驱动哈密顿量来动态实现。使用能量惩罚项时（左图），驱动项需要探索整个希尔伯特空间。相比之下，动态实现约束（右图）会将搜索空间缩减至满足硬约束的子空间，从而提升算法的性能。(b) 受约束逻辑问题（左图）及其奇偶表示。在奇偶架构中，多项式约束的每一项都由一个奇偶量子比特（绿色量子比特）表示，因此，多项式约束定义了一个子空间，其中相关量子比特的总磁化强度保持不变；可通过保持总磁化强度的交换驱动项 $\sigma_+\sigma_- + h.c.$ 对该子空间进行探索。图最初发表于参考文献 [29]。

or

$$H_{\text{core}} = A\left[c(v_1, \ldots, v_N) - K\right], \tag{40}$$

在 $c(x_1, \ldots, x_N) \geq K$ 满足这一特殊情况时。常数 A 必须足够大，以确保总哈密顿量的基态满足该约束，但实现较大的能量尺度会降低量子算法的效率 [42]，此外还会带来技术上的挑战。此外，哈密顿量中的额外项意味着需要更多计算资源，尤其是方程（39）中的平方项。

用于寻找哈密顿量基态的量子算法（如 QAOA 或量子退火）需要驱动项 $U_{\text{drive}} = \exp(-itH_{\text{drive}})$，将初始量子态扩展至整个希尔伯特空间。约束的动态实现采用一种驱动项，它只探索希尔伯特空间中满足约束的子空间。给定以伊辛算符 $\sigma_z^{(i)}$: 表示的编码约束，

$$\begin{aligned} c(\sigma_z^{(1)}, \ldots, \sigma_z^{(N)}) = &\sum g_i\sigma_z^{(i)} \\ &+ \sum g_{i,j}\sigma_z^{(i)}\sigma_z^{(j)} + \cdots = K, \end{aligned} \tag{41}$$

与$c(\sigma_z^{(1)}, \ldots, \sigma_z^{(N)})$对易的驱动哈密顿量 H_{drive} 会限制在有效子空间内，前提是初始量子态满足该约束：

$$\begin{aligned} c|\psi_0\rangle &= K|\psi_0\rangle \\ U_{\text{drive}}\, c|\psi_0\rangle &= U_{\text{drive}}\, K|\psi_0\rangle \\ c\left[U_{\text{drive}}|\psi_0\rangle\right] &= K\left[U_{\text{drive}}|\psi_0\rangle\right]. \end{aligned} \tag{42}$$

一般而言，保持约束的驱动项的构造取决于问题 [9, 10, 25, 28, 30]。人们提出了近似驱动项，它们允许一定程度的泄漏，并且可能更易于构造 [28]。在奇偶架构中，多项式约束的每一项都是一个奇偶量子比特 [21, 24]。这意味着，对于奇偶架构而言，多项式约束实际上就是所涉及量子比特之间磁化强度的守恒：

$$\begin{aligned} &\sum_i g_i\sigma_z^{(i)} + \sum_{i,j} g_{i,j}\sigma_z^{(i)}\sigma_z^{(j)} + \cdots = K \\ &\quad\rightarrow \sum_{\mathbf{u}\in c} g_{\mathbf{u}}\sigma_z^{(\mathbf{u})} = K, \end{aligned} \tag{43}$$

其中奇偶量子比特 $\sigma_z^{(\mathbf{u})}$表示逻辑量子比特的乘积 $\sigma^{(u_1)}_z \sigma^{(u_2)}_z \ldots \sigma_z^{(u_n)}$，我们对所有这些求和

出现在约束中的乘积 c。基于交换（或翻转-翻转）项构造的驱动哈密顿量 $\sigma_+^{(\mathbf{u})}\sigma_-^{(\mathbf{w})}+h.c.$对所有乘积对 $\mathbf{u},\mathbf{w} \in c$ 求和，可保持总磁化强度，并探索满足该约束的完整子空间 [29]。编码约束的决策树见图 3。

VI. 与编码无关的表述

本库中的问题由一个代价函 f 数和一组约束构成，这些约束定义了函数 f 必须在其中取得最小值的子空间。代价函数和约束都以离散变量表示，我们称之为与编码无关的表述。为了生成适用于量子优化算法的哈密顿量，必须使用第 III 节介绍的编码，将离散变量编码为伊辛算符。如果某些变量采用稀疏编码，则必须将这些变量的核心约束纳入问题的约束集合。

对问题进行编码后，必须决定将约束（问题约束或核心约束）实现为能量惩罚，还是在驱动项中动态实现（见第 V 节）。问题的哈密顿量由编码后的代价函数 f 以及作为能量惩罚实现的约束 c_i 构成：

$$H = Af + Bc_1 + Cc_2 + \ldots, \tag{44}$$

其中，能量尺度 $A < B < C$，.....的选取应确保 H 的基态同时也是表示各约束项的基态。这就保证了违反约束以进一步降低代价函数在能量上绝不会更有利。如果能量尺度差异过大，算法的性能可能会下降（此外还会带来实验上的挑战），因此参数A、B、C、.....不能任意设得很大。确定最优能量尺度是一项重要的开放问题。对于库中的部分问题，我们会提供能量尺度的估算值（另见第 VIII D2 节）。

本库收录的问题分为四类：子集、划分、排列、连续变量。这些类别根据离散变量的作用来定义，旨在组织问题库、方便查找问题，同时也为构建类似应用场景提供基础。第 VIIE 节另设一类，收录不属于上述类别、但也可能是量子算法重要应用场景的问题。第 VIII 节总结了贯穿整个问题库、反复使用的编码无关构件，这些构件或可用于构建新问题。

现在我们给出两个示例，从与编码无关的问题到伊辛哈密顿量及其约束。这一输出已经可以使用 Parity 编译器 [21] 进行编译，无需将问题化简为 QUBO 或进行额外的嵌入。

下面以聚类问题和旅行商问题为例，说明这种编码。

a. 聚类问题：聚类问题是一种划分问题，见第 VII A1 节。图 4 展示了如何从编码无关哈密顿量得到需要在量子计算机上实现的自旋哈密顿量。问题实例由必需的输入定义：我们要创建的簇数 K 、 N 个带权对象 w_i 以及对象之间的距离$d_{i,\ j}$ 。需要两种不同类型的离散变量：变量$v_i = 1, \ldots, K$ （$i = 1, \ldots, N$）表示对象 i 被分配到 K 个可能簇中的哪一个；变量$y_j = 1, \ldots,$ $W_{\max}$ 则跟踪簇 j 中的权重。

变量 v_i 和 y_j 在聚类问题的表述中承担不同的作用，因此采用不同的编码可能更有利。对于变量 v_i只需要值指示器 $\delta_{v_i}^{\alpha}$ ，因此建议使用具有简单 $\delta_{v_i}^{\alpha}$ 的稀疏编码。相比之下，问题表述只需要变量 y_j 的值，因此采用二进制编码或格雷编码等稠密编码可以减少所需的布尔变量数量。为了找到问题的自旋表示，需要选择一种编码，并决定是将核心项（如果存在）作为能量惩罚项纳入，还是通过保持约束的驱动项实现（参见第 V节）。

选定编码后，将离散变量代入代价函数和与问题相关的约束中，分别见式（49）和（47）。问题约束可以作为能量惩罚项编码到哈密顿量中，也可以作为求和约束，通过量子算法的驱动项动态实现。由此得到的哈密顿量和约束现在都以自旋算子表示，并且可以使用 Parity 编译器进行编译，从而得到仅包含局域相互作用的二维芯片设计。

b. 旅行商问题：我们介绍的第二个例子是旅行商问题（TSP），详见第VII C2节。该问题的输入是一个（通常为完全图）图 G ，图中的每条边都关联一个成本$w_{i,\ j}$ 。我们要寻找一条遍历所有节点 G 且不重复访问同一节点的路径，同时使路径上各边的总成本$\sum w_{i,\ j}$ 最小。

该问题需要 N个变量 $v_i = 1,$ ....., N，其中 N 是 G中的顶点数。每个变量 v_i表示图中的一个节点，并指明旅行过程中访问节点 i 的阶段。如果 $v_i = \alpha$ 且 $v_j = \alpha + 1$，则旅行商从节点 i 前往节点j ，因此代价函数会将 $w_{i,\ j}$ 计入旅行总成本，如方程（96）所示。此公式要求满足方程（93）中的约束 $v_i \neq v_j$，对于所有 $i \neq j$ 均成立，从而确保旅行商恰好访问每个节点一次。

在图5中，我们给出了一个

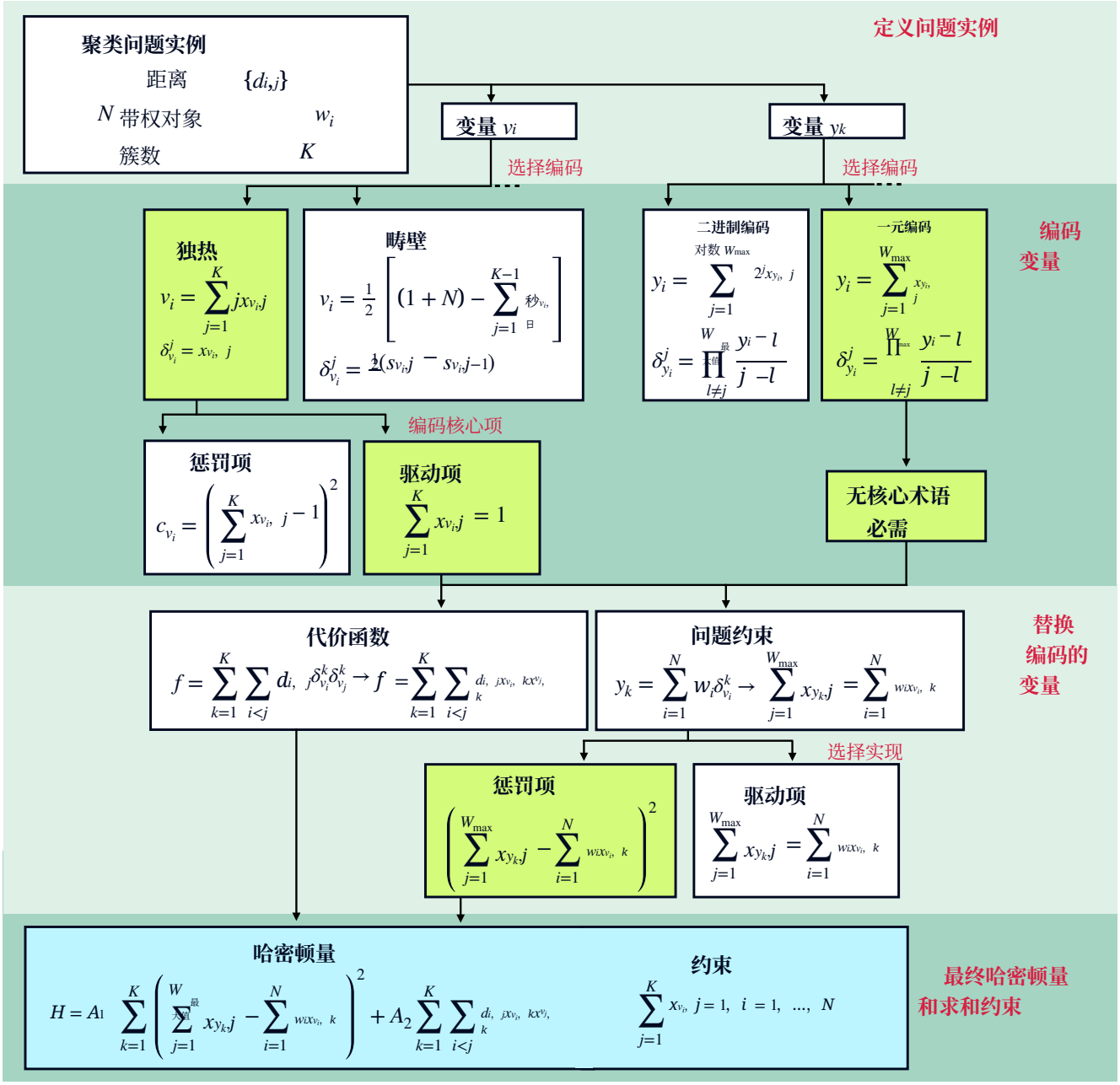


图 4. 聚类问题的决策树示例。整个流程分为四个不同的块（定义问题、编码变量、替换变量和最终哈密顿量），所做的决策以绿色突出显示。定义一个问题实例需要将离散变量 v_i 和 y_k 编码为量子比特算符。在此示例中，v_i 变量采用独热编码，而 y_k 变量采用二进制编码。根据所选编码，将离散变量 47）和（49）中的编码无关表达式里的变量 v_i、y_k 以及值指示变量 δ_v^α 代入，即可得到哈密顿量。与稀疏编码（此处为独热编码）相关的核心项以及问题约束，可以作为能量惩罚项添加到哈密顿量中，也可以作为求和约束，在量子算法的驱动项中动态实现。最终哈密顿量和求和约束可直接在 Parity 芯片上使用 Parity 编译器进行编码，无需将问题化简为 QUBO 表述。

该问题的决策树示例。代价函数和约束中包含值指示变量的乘积$\delta_{v_i}^{\alpha}\,\delta_{v_k}^{\beta}$，因此更适合采用畴壁等稀疏编码。

VII. 问题库

A. 划分问题

划分问题的共同目标是寻找集合 U 的划分，使代价函数 f 最小化。一个

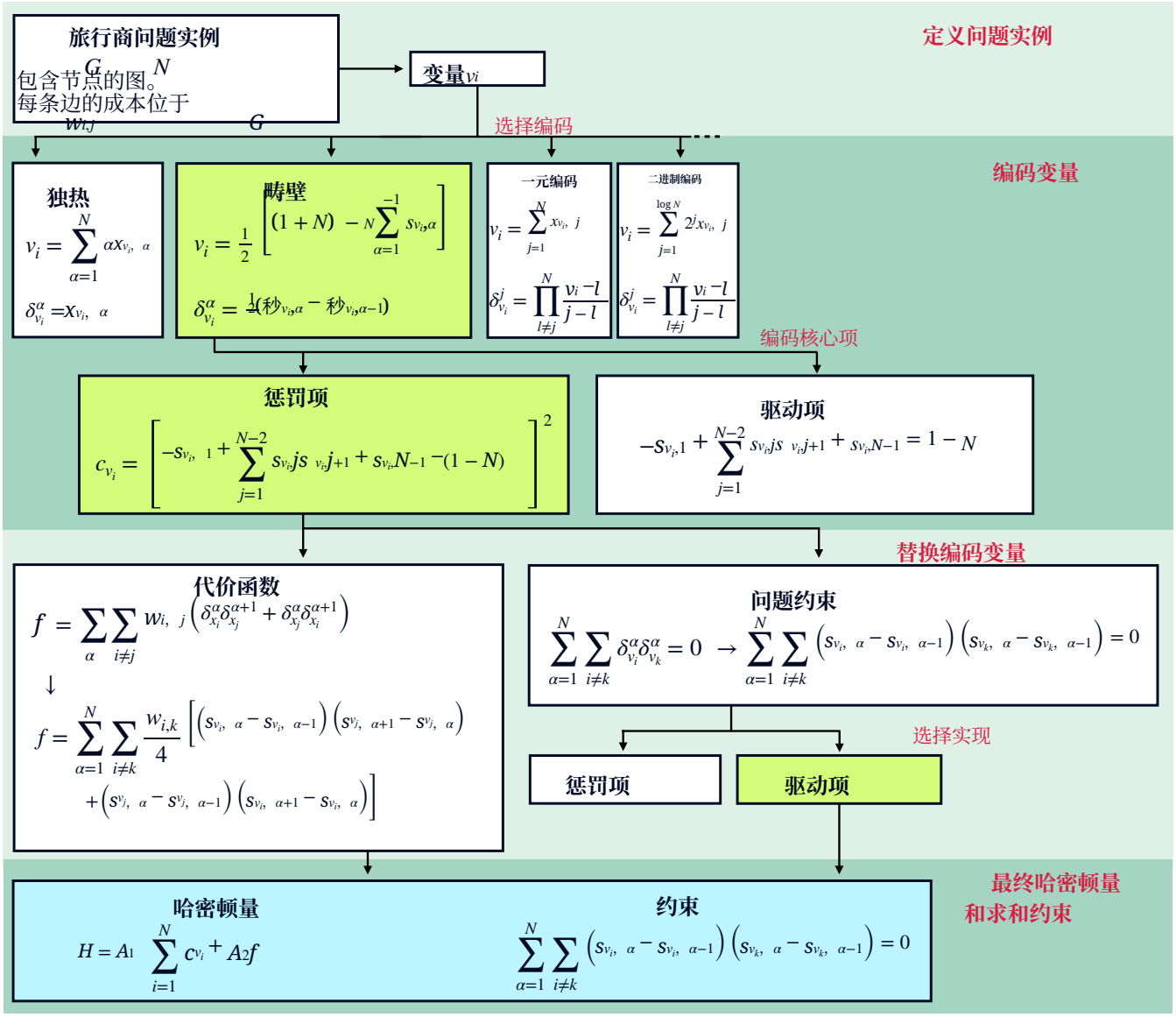


图5. 旅行商问题的决策树示例。通过将值指示变量替换为所选编码对应的公式（式（96）和（93）中的编码无关表达式），即可得到哈密顿量。决策树中所作的选择以绿色突出显示。对于此问题实例，我们假设图 G 是完全图（每对节点之间都有一条边），并选择畴壁编码。由于代价函数和问题约束包含值指示变量的乘积 $\delta_{v_i}^{\alpha}\delta_{v_k}^{\beta}$，使用稠密编码会产生大量相互作用项，因此这种稀疏编码很方便。稀疏编码需要一个核心项来定义希尔伯特空间的有效态。在此情况下，我们选择在哈密顿量中将此核心项编码为能量惩罚项，同时将问题约束视为求和约束。最终哈密顿量和求和约束可直接编码到 Parity 芯片上，使用 Parity 编译器即可，无需将问题化简为 QUBO 表述。

划分 $P$of U 是一个子集 $U_k \subset U$ 的集合 $P := \{U_k\}_{k\in K}$，满足 $U = \bigcup_{k=1}^{K} U_k$；如果 $k \neq k'$，则 $U_k \cap U_{k'} = 0$。划分问题要求为每个元素 $u_i \in U$ 设置一个离散变量 v_i。变量的价值表示该元素属于哪个子集，因此 v_i 可以取 K 个不同的值。分配给各子集的值是任意，因此在这些情况下，变量 v_i 的值通常并不重要，只需要值指示器$\delta_{v_i}^{\alpha}$，所以稀疏编码可能适用于这些变量。

1. *Clustering problem*

a. 描述 设 $U = \{u_i\}_{i=1}^N$ 为由权重 $w_i \in \{1, ..., w_{\max}\}$ 和距离 $d_{i,j}$ 表征的一组 N 元素。聚类问题旨在将集合 U 划分为 K 个非空子集 U_k，使每个子集内顶点之间的距离最小。划分受权重限制：对于每个子集 U_k，该子集中顶点的权重总和不得超过给定的最大权重 $W_{\max}$。

b. 变量 我们为 U 中的每个元素定义一个变量 $v_i = 1, \ldots, K$。我们还为每个子集 U_k 定义一个辅助变量 $y_j = 0, 1, \ldots, W_{\max}$，用于表示 U_k 中元素的总权重：

$$y_k = \sum_{\{i:u_i \in U_k\}} w_i. \tag{45}$$

c. 约束 权重限制是一个不等式约束：

$$y_k \le W_{\max}, \tag{46}$$

该约束可表示为：

$$y_k = \sum_i w_i \delta^k_{v_i}. \tag{47}$$

如果辅助变量的编码方式使其有此必要（例如，采用二进制编码且 $W_{\max}$ 不是 2 的幂），则必须按照第 VIII B3 节所述对该约束进行平移。

d. 代价函数 一个子集中各元素的距离之和为：

$$f_k = \sum_{\{i<j:u_i,u_j \in U_k\}} d_{i,j} = \sum_{i<j} d_{i,j} \delta^k_{v_i} \delta^k_{v_j}, \tag{48}$$

由此得到代价函数：

$$f = \sum_k f_k. \tag{49}$$

e. 参考文献 该问题作为带容量限制的车辆路径问题的一部分，在参考文献 [43] 中进行了研究。

2. *Number partitioning*

a. 描述 给定由 N 个实数 u_i 构成的集合 U，我们要寻找一个大小为 K 的划分，使每个子集中的数值之和尽可能均匀。

b. 变量 该问题需要 N 个变量 $v_i \in [1, K]$，每个元素 u_i 对应一个变量。v_i 的值表示其所属的子集 u_i。

c. 代价函数 部分和可以使用与变量 v_j 相关联的价值指示变量来表示：

$$p_j = \sum_{u_i \in U_j} u_i = \sum_{i=1}^N u_i \delta^j_{v_i}. \tag{50}$$

寻找最优划分有三种常见方法：最大化最小部分和、最小化最大部分和，或最小化最大划分与最小划分之间的差值。后一种方法可表示为

$$f = \sum_{i<j} (p_i - p_j)^2. \tag{51}$$

为了最小化最大部分和（最大化最小部分和的方法类似），我们引入一个辅助变量 l，其取值为 $1, ..., \sum_i u_i \equiv l_{\max}$。根据问题实例，$l$ 的取值范围还可以进一步缩小。代价函数中的第一项

$$f = l_{\max} \left(1 - \prod_i \Theta(l - p_i)\right) + l \tag{52}$$

从而确保 l 至少与最大值一样大 p_i（参见第 VIII A3 节），第二项则最小化 l。根据构件方程（150），Θ 阶跃函数可以用值指示函数表示；对此，我们需要引入辅助变量，或根据第 VIII D1 节的讨论直接表示值指示变量。

d. 特殊情况 对于 $K = 2$，使部分和之差最小的代价函数为

$$\begin{aligned} f &= (p_2 - p_1)^2 \\ &= \left[\sum_{i=1}^N \left(\delta^1_{v_i} - \delta^2_{v_i}\right) u_i\right]^2. \end{aligned} \tag{53}$$

$\delta^1_{v_i} - \delta^2_{v_i}$ 仅有两种可能的结果，即 ± 1，因此可以使用自旋变量 $s_i = \pm 1$ 对这些因子进行简单编码，从而得到

$$f = \sum_{i=1}^N s_i u_i. \tag{54}$$

e. 参考文献 $K = 2$ 的哈密顿量表述可见于 [11]。

3. *Graph coloring*

a. 描述 在这个问题中，图 $G = (V, E)$ 的节点被划分为 K 个不同的子集，每个子集代表一种不同的颜色。我们要寻找一种划分，使任意两个相邻节点的颜色不同；也可以寻找一种能使所用颜色数量最少的划分。

b. 变量 我们为图中的每个节点 $v_i = 1 \dots K$ $i = 1, ..., N$定义一个变量。

c. 约束 我们必须对两个相邻节点被涂成相同颜色的解施加惩罚。方程（60）中给出的图划分问题的代价函数可用于图着色的约束。在这种情况下，$[1 - \delta（v_i - v_j）] A_{i,\ j}$原本用于计算属于不同子集的相邻节点数量，现在我们使用 $\delta（v_i - v_j）\ A_{i,\ j}$ 以指示节点 i 和 j是否被涂成相同颜色。因此，约束为（见构建块 VIII B1）

$$c = \sum_{i<j} \delta(v_i - v_j) A_{i,j} = 0. \tag{55}$$

d. 代价函数 判定问题（“是否存在一种使用 K 种颜色的着色？”）可以通过将约束实现为哈密顿量 $H = c$ 来回答（注意 $c \geq 0$）。存在零能量的解意味着存在一种使用 K 种颜色的着色。

或者，我们可以寻找使用最少颜色数的着色。要检查某种颜色α 是否被使用，可以使用以下项（见构建块 VIII B2）：

$$u_\alpha = 1 - \prod_{i=1}^{N} \left(1 - \delta_{v_i}^{\alpha}\right), \tag{56}$$

当且仅当着色中未使用颜色 α 时，其值为 1；若至少有一个节点被涂成颜色α，则其值为 0。着色中使用的颜色数量即为该问题的代价函数：

$$f = \sum_{\alpha=1}^{K} u_\alpha. \tag{57}$$

该目标函数的实现成本可能很高，因为 u_α 包含 $\delta_{v_i}^{\alpha}$ 的 N 阶乘积，因此，若能较好地估算最小数量 $K_{\min}$，就可以避免使用不必要的大型 K。

e. 参考文献 该问题的独热编码版本可见参考文献 [11]。

4. *Graph partitioning*

a. 描述图划分是将图 $G = (V, E)$ 的节点分成 K 个不同的子集，使连接不同子集中节点的边数（割边数）最小化或最大化。

b. 变量我们为图中的每个节点使用一个离散变量 $v_k = 1, \dots, K$，用于指示该节点所属的划分。

c. 代价函数 连接两个节点的边 i和 j 在 $v_i \neq v_j$ 时被切断。使用符号（见构件 VIII A4）：

$$\delta(v_i - v_j) = \sum_{\alpha=1}^{K} \delta_{v_i}^{\alpha} \delta_{v_j}^{\alpha} = \begin{cases} 1 \text{ if } v_i = v_j \\ 0 \text{ if } v_i \neq v_j \end{cases}, \tag{58}$$

当节点 i 和 j 属于同一划分（$v_i = v_j$）时，其值为 1；否则（$v_i \neq v_j$）为 0。这样一来，以下项：

$$(1 - \delta(v_i - v_j))\, A_{i,j} \tag{59}$$

当连接节点 i 和j的边为割边时，其值等于 1，其中 $A_{i,\ j}$ 为图的邻接矩阵。对所有节点求和，即可得到用于最小化割边数的代价函数：

$$f = \sum_{i<j} [1 - \delta(v_i - v_j)]\, A_{i,j}, \tag{60}$$

或者 $-f$ 最大化割边数量。

d. 约束 一种常见的约束是要求各划分具有特定的大小。元素计数构件 VIII A1 定义为

$$t_\alpha = \sum_{i=1}^{|V|} \delta_{v_i}^{\alpha}. \tag{61}$$

如果希望划分 α 包含 L 个元素，则

$$t_\alpha = L \tag{62}$$

必须成立。如果希望两个划分 α 和 β 大小相同，则约束为

$$t_\alpha = t_\beta, \tag{63}$$

并且要求所有划分大小相同，则有

$$\sum_{a<b} (t_a - t_b)^2 = 0 \tag{64}$$

这只有在 $|V|/N$ 是自然数时才可能。

e. 参考文献 $K = 2$的哈密顿量可在 [11]中找到。

f. 超图划分 问题的公式可以扩展到超图。当一条超边包含来自至少两个不同子集的顶点时，该超边就被切割。给定一条由 G构成的超边 e，函数

$$\text{cut}(e) = \prod_{i,j \in e} \delta(v_i - v_j) \tag{65}$$

仅当 e 中包含的所有顶点都属于同一个划分时才等于 1，其他情况下均为零。对边 e 的顶点 i, j 所求的乘积只需包含这样的顶点对，使得每个顶点至少出现一次。通过求和惩罚项，实现最小化被切割超边的优化目标

$$\begin{aligned} f &= \sum_{e \in E} (1 - \text{cut}(e)) \\ &= \sum_{e \in E} \left(1 - \prod_{i,j \in e} \sum_{a=1}^{K} \delta_i^a \delta_j^a\right). \end{aligned} \tag{66}$$

此目标函数对所有可能的切割一视同仁，不考虑被切割的顶点数量，也不考虑一条边所属的划分数量。

5. *Clique cover*

a. 描述 给定一个图 $G=(V,\ E)$，我们要寻找为所有顶点着色所需的最少颜色数 K，使得颜色为 α 的顶点子集 W_α，连同限制在 W_α 中顶点之间的边所构成的边集 E_α，形成完全图。一个子问题是判断是否存在使用 K 种颜色的团覆盖。

b. 变量 对于每个顶点 $i=1,\ldots,N$，我们定义变量 $v_i=1,\ldots,K$，表示分配给该顶点的颜色。如果未给定颜色数，则必须先为 K 取一个初始猜测值或某个最小值。

c. 约束 在这个问题中，$G_\alpha=(W_\alpha,E_\alpha)$ 必须是一个完全图，因此 E_α 中必须包含最多数量的边。使用元素计数构建模块，我们计算颜色为α 的顶点数（参见构建模块VIIIA1）：

$$t_\alpha=\sum_{i=1}^{N}\delta_i^\alpha. \tag{67}$$

如果 G_α 是一个完全图，那么E_α 中的边数为 $t_\alpha(t_\alpha-1)/2$，因此约束为

$$c=\sum_{\alpha=1}^{K}\left(\frac{t_\alpha(t_\alpha-1)}{2}-\sum_{(i,j)\in E}\delta_i^\alpha\delta_j^\alpha\right)=0. \tag{68}$$

请注意，我们不必对该项平方，因为它不可能为负。

d. 代价函数 判定问题（“是否存在使用 K 种颜色的团覆盖？”）可以将约束 c 用作该问题的哈密顿量来回答。为了找到存在团覆盖时所需的最少颜色数 $K_{\min}$（即团覆盖数），我们添加一个用于最小化 K 的代价函数。与图着色问题一样，我们通过以下方式最小化颜色数

$$f=\sum_\alpha(1-u_\alpha), \tag{69}$$

其中

$$u_\alpha=\prod_{i=1}^{N}\left(1-\delta_{v_i}^\alpha\right) \tag{70}$$

表示颜色 α 是否被使用（参见构建模块VIII B2）。

e. 参考文献 该判定问题的独热编码哈密顿量可见参考文献 [11]。

B. 受约束子集问题

给定集合 U，我们要寻找一个非空子集 $U_0\subseteq U$，使其在满足一组约束 c_i 的同时，将代价函数 U 最小化。一般而言，这类问题需要为 U 中的每个元素设置一个二元变量 x_i，以指示元素 i 是否包含在子集 U_0 中。尽管二进制变量可以直接编码为单个量子比特，但构造约束时可能需要非二进制辅助变量，因此，对这类问题进行与编码方式无关的表述仍然有用。

1. *Cliques*

a. 描述 给定图 $G=(V,E)$ 上的一个团是顶点 $W\subseteq V$ 的子集，满足 W，并且 W 中顶点之间的边的子集 E_W 构成一个完全图，即 E_W 中包含可能的最大数量的边。目标是找到一个基数为 K 的团。此外，也可以询问该图的最大团是什么。

b. 变量 我们定义 $|V|$ 个二元变量 x_i，用于指示顶点 i 是否属于该团。

c. 约束该问题有两个约束，即团的基数为 K，且该团确实具有最多的边数。前者通过以下方式强制实现

$$c_1=\sum_{i=1}^{|V|}x_i=K \tag{71}$$

后者则通过以下方式实现

$$c_2=\sum_{(i,j)\in E}x_ix_j=\frac{K(K-1)}{2}. \tag{72}$$

若将约束作为能量惩罚项来实现，就必须确保第一个约束不被违反，以免降低第二个约束的惩罚。通过对单自旋翻转进行成本/收益分析（见第VIIID 2节），只要满足 $a_1\gtrapprox a_2\Delta$，就能避免这种情况；其中 Δ 是 G 的最大度，而$a_{1,\ 2}$ 是第一个和第二个约束的能量尺度。

d. 代价函数判定问题（是否存在大小为 K的团？）可以将约束作为该问题的哈密顿量来求解。若要找到图 G中的最大团，就必须将 K 编码为离散变量 $K=1,\ \ldots,\ K_{\max}$，其中 $K_{\max}=\Delta$ 是 G 的最大度，也是团可能达到的最大大小。此时的代价函数就是 K的值：

$$f=-K \tag{73}$$

e. 资源 将约束作为能量惩罚来实现时，判定问题（固定 K）的总成本函数为，

$$f=(c_1-K)^2+\left(c_2-\tfrac{K(K-1)}{2}\right)^2, \tag{74}$$

相互作用项的最高阶数为二，且项数随 $|E|+|V|^2$ 增长。如果将 K 编码为离散变量，所需资源取决于所选的编码方式。

f. 参考文献 该哈密顿量在参考文献 [11] 中采用独热编码进行表述。

2. *Maximal independent set*

a. 描述 给定一个超图 $G=(V,\ E)$，我们要寻找顶点的一个子集 $S\subset V$，使得 E 中不存在连接 S中任意两个顶点的边。寻找最大的 S 是一个NP难问题。

b. 变量 对于 V中的每个顶点，我们使用一个二元变量 x_i 。

c. 代价函数 为了最大化 S中顶点的数量，代价函数为

$$f=-\sum_{i=1}^{|V|}x_i. \tag{75}$$

d. 约束 给定 S中的两个元素，E 中不得有任何边或超边连接它们。约束

$$c=\sum_{i,j\in V}A_{i,j}x_ix_j \tag{76}$$

计算 S中相邻顶点的数量，其中 A 为邻接矩阵。令 $c=0$，则 S 中的顶点构成一个独立集。

e. 参考文献 关于图的问题可见参考文献 [11, 44]。

3. *Set packing*

a. 描述 给定一个集合 U 以及一族由 U 的子集 V_i 组成的集合 $S=\{V_i\}_{i=1}^N$，我们要寻找集合打包，即 S 的子集，使得所有子集两两不相交，$V_i\cap V_j=\emptyset$。寻找最大打包（子集 V_i 数量最多的打包）是一个称为集合打包的NP 难优化问题。

b. 变量 我们定义 N 二进制变量 x_i，用于指示子集 V_i 是否属于该打包。

c. 代价函数 通过元素计数构件，可以实现打包中子集数量最大化

$$f=-\sum_{i=1}^{N}x_i. \tag{77}$$

d. 约束 为了确保打包中的任意两个子集互不相交，我们对重叠的集合施加代价，具体如下

$$c=\sum_{i,j:V_i\cap V_j\neq\emptyset}x_ix_j=0. \tag{78}$$

e. 资源总成本函数 $H=H_A+H_B$的相互作用项最高为二阶（因此这是一个 QUBO问题），项数最多按 N^2 的规模增长。

f. 参考文献此问题可见参考文献 [11]。

4. *Vertex cover*

a. 描述 给定一个超图 $G=(V,E)$，我们要寻找最小子集 $C\subseteq V$，使得每条边都至少包含 C 中的一个顶点。

b. 变量 我们定义 $|V|$ 二进制变量 x_i，用于指示顶点 i 是否属于顶点覆盖 C。

c. 代价函数 通过元素计数构件，可以实现最小化 C 中顶点的数量

$$f=\sum_{i=1}^{|V|}x_i. \tag{79}$$

d. 约束通过

$$c=\sum_{e=(u_1,\ldots,u_k)\in E}\prod_{a=1}^{k}(1-x_{u_a})=0 \tag{80}$$

可以惩罚所有不包含属于 C 的顶点的边。将该约束编码为能量惩罚项后，哈密顿量为

$$H=Af+Bc. \tag{81}$$

通过设置 $B>A$，我们可以避免该约束与最小化 C 相互权衡。

e. 资源 相互作用项的最高阶数为超边 k 的最高秩，项数随 $|E|2^k+|V|$ 增长。

f. 参考文献 仅考虑图的特殊情况可见参考文献 [11]。

5. *Minimal maximal matching*

a. 描述 给定一个秩最大为 k 的超图 $G=(V,E)$，我们希望找到一个极小（即边数最少）的匹配 $C\subseteq E$。这里的极大是指，所有顶点未与 C 中的边相接的边都必须包含在该匹配中。

b. 变量 我们定义二进制变量 x_i，用于指示边是否属于匹配 C。

c. 代价函数 只需通过以下代价函数，即可最小化匹配中的边数

$$f_A=A\sum_{i=1}^{|E|}x_i. \tag{82}$$

d. 约束我们必须确保 C 确实是一个匹配，即任意两条共享同一顶点的边都不能属于 C。通过能量惩罚，可按以下方式实现:

$$f_B=B\sum_{v\in V}\sum_{(i,j)\in\partial v}x_ix_j=0, \tag{83}$$

其中， ∂v 是与顶点 v相连的边的集合。此外，该匹配应为极大的。对于每个顶点 u，我们定义一个变量 $y_u = \sum_{i\in\partial u} x_i$ ，只有当该顶点不属于 C中的任何一条边时，该变量才为零。如果满足第一个约束，该变量只能取 0 或 1。在这种情况下，可通过以下方式施加该约束：

$$f_C = C \sum_{e=(u_1,\dots,u_k)\in E} \prod_{a=1}^{k} (1 - y_{u_a}) = 0. \tag{84}$$

然而，必须确保 f_B 所实现的约束不会被违反而偏向 f_C ；如果对于某些 v、 $y_v > 1$ 以及 m 个相邻顶点 $y_u = 0$，就可能发生这种情况。于是，来自 v 的贡献为

$$f_v = By_v(y_v - 1)\frac{1}{2} + C(1 - y_v)m \tag{85}$$

由于 $m+y_v$ 受G的最大度 Δ 与超边 k的最大秩的乘积所界定，因此我们需要设置 $B > (\Delta k - 2)C$ ，以确保 $f_B + f_C$ 的基态不会违反第一个约束。最后，还必须防止 f_C 被违反而偏向 f_A ，这意味着 $C > A$。

e. 资源 相互作用项的最高阶为 k ，项数大致随$(|V| + |E|2^k)\Delta(\Delta - 1)$增长。

f. 参考文献该问题可见参考文献 [11]。

6. *Set cover*

a. 描述 给定一个集合 $U = \{u_\alpha\}_{\alpha=1}^{n}$ 及 N个子集 $V_i \subseteq U$，我们要寻找满足 $U = \bigcup V_i$的 V_i的最小数量。

b. 变量 我们为每个子集 V_i 定义一个二元变量 $x_i = 0, 1$，用于指示是否选择了该子集 V_i。我们还定义辅助变量 $y_\alpha = 1, 2, \dots, N$，用于指示包含元素 u_α 的活动子集（V_i 满足 $x_i = 1$）的数量。

c. 代价函数 代价函数很简单：

$$f = \sum_{i=1}^{N} x_i, \tag{86}$$

它统计所选子集 V_i 的数量。

d. 约束 约束 $U = \bigcup_{i:x_i=1} V_i$ 可表示为

$$y_\alpha > 0, \ \forall \alpha = 1, \dots, n, \tag{87}$$

这意味着每个元素 $u_\alpha \in U$ 至少包含一次。当以下不等式 y_α 限定为有效值（$y_\alpha = 1$、2、.....、 N、 $y_\alpha \neq 0$）（参见第 VIII B3节）时，这些不等式成立。 y_α的值应与 x_i的值一致，因此约束为

$$c_\alpha = y_\alpha - \sum_{i:u_\alpha\in V_i} x_i = 0. \tag{88}$$

e. 特殊情况：精确覆盖 如果我们希望 U 中的每个元素在顶点覆盖中恰好出现一次，那么对于所有 α ， y_α =1，且问题的约束简化为

$$c_\alpha = \sum_{i:\alpha\in V_i} x_i = 1. \tag{89}$$

f. 参考文献 独热编码的哈密顿量可见参考文献 [11]。

7. *Knapsack*

a. 描述 集合 U 包含 N 个对象，每个对象都有价值 d_i 和权重 w_i。我们要从中寻找一个子集 U ，使其价值 $\sum d_i$ 最大，同时所选对象的总权重不超过上限 W。

b. 变量 我们为 U 中的每个元素定义一个二元变量 $x_i = 0, 1$，用于指示是否选中了元素 i 。我们还定义一个辅助变量 y，用于表示所选对象的总权重：

$$y = \sum_{i:x_i=1} w_i. \tag{90}$$

如果权重是自然数 $w_i \in \mathbb{N}$ ，那么 y 也为自然数，并且可以大大简化这个辅助变量的编码。

c. 代价函数 代价函数为

$$f = -\sum_{i=1}^{N} d_i x_i, \tag{91}$$

它对所选元素的价值进行计数。

d. 约束 通过强制 $y < W$ y 取可能值之一$y = 1, \dots W - 1$（参见参考文献 VIII B 3）。 y的值必须与从 U中选出的项目相一致：

$$c = y - \sum_{i}^{N} w_i x_i = 0. \tag{92}$$

e. 参考文献 独热编码哈密顿量见参考文献 [11]。

C. 排列问题

在排列问题中，我们需要找到 N 个元素的一个排列，使代价函数最小，同时满足给定的一组约束。一般来说，我们会使用一个离散变量 $v_i \in [1, N]$ 来表示元素 i 在排列中的位置。

1. *Hamiltonian cycles*

a. 描述 对于图 $G = (V, E)$，我们要判断是否存在哈密顿回路，即一条通过现有边连接图中所有节点且不重复经过同一节点的闭合路径。

b. 变量 我们为图中的每个节点定义一个变量 $v_i = 1, \ldots, |V|$，表示该节点在排列中的位置。

c. 代价函数 此问题没有代价函数，因此所有满足约束的问题排列都是解。

d. 约束 此问题需要两个约束。第一个约束是所有排列问题固有的，它要求 $|V|$ 变量 $\{v_i\}$成为 $[1, \ldots, |V|]$的一个排列。这等价于要求 $v_i \neq v_j$ 若 $i \neq j$，可通过以下约束对其进行编码（见第VIIIB 1节）：

$$c_1 = \sum_{\alpha=1}^{N} \sum_{i \neq k} \delta_{v_i}^{\alpha} \delta_{v_k}^{\alpha} = 0. \tag{93}$$

第二个约束确保路径只经过图中的边。设 A 为图的邻接矩阵，使得 $A_{i,j} = 1$ 若节点 i 和 j 之间有边相连，否则为零。为惩罚无效解，我们使用以下约束

$$c_2 = \sum_{i,j} \sum_{\alpha=1}^{|V|} (1 - A_{i,j}) \left(\delta_{v_i}^{\alpha} \delta_{v_j}^{\alpha+1} + \delta_{v_i}^{\alpha+1} \delta_{v_j}^{\alpha} \right) = 0, \tag{94}$$

它用于统计解中未通过图中的边连接起来的相邻节点数量。$\alpha = |V| + 1$表示 $\alpha = 1$，因为我们寻找的是一条闭合路径。

e. 参考文献 独热编码哈密顿量可参见参考文献 [11]。

2. *Traveling salesperson problem (TSP)*

a. 描述 旅行商问题是哈密顿回路问题的简单扩展。在这种情况下，节点代表城市，边代表连接城市的可能道路，不过通常假定所有城市彼此相连（该图是完全图）。对于连接城市 i 和 j 的每条边，都有一个成本$w_{i,\ j}$。旅行商问题的解是使总成本$\sum w_{i,\ j}$最小的哈密顿回路。

b. 变量 我们为图中的每个节点定义一个变量 $v_i = 1, \ldots\ldots, |V|$，表示该节点在排列中的位置。

c. 代价函数 如果旅行者从位置 α 的城市 i 前往下一步中的城市 j，那么

$$\delta_{v_i}^{\alpha} \delta_{v_j}^{\alpha+1} = 1, \tag{95}$$

否则该表达式将为零。因此，旅行的总成本由以下函数表示：

$$f = \sum_{\alpha} \sum_{i<j} w_{i,j} \left(\delta_{v_i}^{\alpha} \delta_{v_j}^{\alpha+1} + \delta_{v_j}^{\alpha} \delta_{v_i}^{\alpha+1} \right). \tag{96}$$

d. 约束 这些约束与哈密顿回路问题中使用的约束相同（参见第VII C1 d段）。如果该图是完全图（所有城市彼此相连），那么约束 c_2 就不是必需的。

e. 参考文献 独热编码哈密顿量可见参考文献 [11]。

3. *Machine scheduling*

a. 描述 机器调度问题旨在寻求将若干作业分配到有限数量机器上的最佳方式。从本质上讲，这些问题研究的是作业列表的排列，其中作业在排列中的位置表示该作业在哪台机器上以及何时执行。文献中可以找到该问题的许多变体，包括以自旋哈密顿量形式表述的问题 [45–47]。这里我们考虑包含 M 台机器和 N个作业的问题，其中所有作业的完成时间都相同，因此时间可以划分为时长相等的时间槽 t。通过使用适当的约束，强制作业在同一台机器上的连续时间槽中运行，可以纳入持续时间为 nt（$n \in \mathbb{N}$）的作业。这样，通过将时间 t 设定得足够小，就可以解决作业持续时间不同的问题。

b. 变量 我们定义变量 $v_{m、\ t} = 0、\ \ldots N$，其中下标 $m = 1、\ \ldots M$表示机器，而 $t = 1、\ \ldots T$表示时间段。当 $v_{m、\ t} = 0$时，机器 m 在时间段 t处于空闲状态；如果 $v_{m、\ t} = j \neq 0$，则作业 j 在时间段t由机器 m完成。

c. 约束 根据我们要运行的使用场景，可能存在许多不同的约束。与所有排列问题一样，我们要求任意一对变量的值都不相同，$v_{m、\ t} \neq v_{m'、\ t'}$，否则某些作业就会被执行两次。我们还要求每项作业都必须完成，因此每项作业 j都必须恰好对应一个 $v_{m、\ t} = j$ 。此约束将在第 VIII B1节中说明，并且对每项作业 $j \neq 0$均成立：

$$c_1 = \sum_{j \neq 0} \left[\sum_{m,t} \delta_{v_{m,t}}^{j} - 1 \right]^2 = 0. \tag{97}$$

请注意，如果某个作业 j 未被分配（即不存在$m,\ t$ 使得 $v_{m,\ t} = j$），那么 $c_1 > 0$。此外，如果变量不止一个$v_{m,\ t} = j$，那么同样有 $c_1 > 0$。当且仅当每个作业都被分配到单个时间槽和单台机器时，该约束才会得到满足（$c_1 = 0$）。

假设只有在另一个作业 j 先完成后，才能启动作业 k 。此约束可以实

现为

$$c_{2,k>j} = \sum_{m,m',t\geq t'} \delta^{j}_{v_{m,t}} \delta^{k}_{v_{m',t'}} = 0, \tag{98}$$

这排除了任何作业 j 在作业 k之后完成的解。或者，也可以将其表示为

$$c'_{2,k>j} = \sum_{m,m',t<t'} \delta^{j}_{v_{m,t}} \delta^{k}_{v_{m',t'}} = 1. \tag{99}$$

这些约束可用于编码涉及不同操作的作业的问题 O_1, $\ldots O_q$这些操作必须按顺序执行。请注意，约束 c_2，c'_2 允许不同操作使用不同的机器。如果我们希望作业 k 紧接在工作 j之后完成，则可以在 c'_2中用 $t+1$ 替换 t'。

如果我们希望两个作业 j_1 和 j_2 在同一台机器的连续时间槽中运行，则该约束可以编码为

$$c_3 = \sum_{m,t} \delta^{j_1}_{v_{m,t}} \delta^{j_2}_{v_{m,t+1}} = 1, \tag{100}$$

或者将其作为哈密顿量中的奖励项，

$$c'_3 = -\sum_{m,t} \delta^{j_1}_{v_{m,t}} \delta^{j_2}_{v_{m,t+1}}, \tag{101}$$

这会降低任何满足以下条件的解的能量：作业 j_2在同一台机器 m上紧接在作业 j_1 之后执行。该约束允许编码作业时长不同的问题，因为 j_1 和 j_2 可以视为同一项作业的一部分，其时长为 $2t$。

d. 代价函数 对于此问题，可以选择不同的目标函数，例如最小化机器空闲时间或提前和延迟交付。一种常见的选择是最小化最大完工时间，即最后一个已排定作业所在的时间槽。为此，我们首先引入辅助函数 $\tau(v_{m,\ t})$，用于指示安排了某个作业的时间槽：

$$\tau\left(v_{m,t}\right) = \begin{cases} t & \text{if } v_{m,t} \neq 0 \\ 0 & \text{if } v_{m,t} = 0. \end{cases} \tag{102}$$

这些函数可以由 $v_{m,\ t}$ 生成：

$$\tau\left(v_{m,t}\right) = \left(1 - \delta^{0}_{v_{m,t}}\right). \tag{103}$$

请注意，τ 的最大值对应于问题的最大完工时间，而该时间需要最小化。为此，我们引入额外变量 $\tau_{\max}$，并对以下情况的构型施加惩罚：对于所有 m, t，$\tau_{\max} < \tau(v_{m,t})$：

$$f_{\tau_{\max}} = \prod_{m,t} \Theta\left[\tau(v_{m,t}) - \tau_{\max}\right], \tag{104}$$

with

$$\Theta(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0. \end{cases} \tag{105}$$

关于如何表示 $\Theta(x$，请参见第 VIII A2 节。通过最小化 $f_{\tau_{\max}}$，我们确保$\tau_{\max} \geq \tau(v_{m,t})$。因此，代价函数就是通过以下方式最小化 $\tau_{\max}$，即作业安排的最晚时间

$$f = \tau_{\max}. \tag{106}$$

此问题的另一种代价函数是

$$f' = \sum_{m,t} \tau(v_{m,t}), \tag{107}$$

该函数会促使所有作业尽可能早地安排。*e.* 参考文献 关于此问题及相关问题的 Quantum 表述方式，可参见参考文献 [45–48]

4. *Nurse Scheduling problem*

a. 描述 在此问题中，我们有 N 名护士和 D 个工作班次。必须在硬约束和软约束下，以最低工作量安排护士，例如在给定班次 t 中，护士 i 的工作量为 p_i 时，护士的最低工作量为 $n_{\min,\ t}$，并尽可能均衡班次数量。此外，任何护士连续工作的天数都不应超过 $d_{\max}$ 天。

b. 变量 我们定义 ND 二进制变量 $v_{i,\ t}$ 用于表示护士 i 是否被安排在班次 t中。

c. 代价函数 使总班次数量最少的代价函数为

$$f = \sum_{t=1}^{D} \sum_{i=1}^{N} v_{i,t} \tag{108}$$

d. 约束 班次的均衡由以下约束表示

$$c = \sum_{i<j} \left(\sum_{t} v_{i,t} - \sum_{t'} v_{j,t'} \right)^2. \tag{109}$$

为了使每个班次的工作量最小，我们引入辅助变量 y_t，并将其绑定到以下值

$$y_t = \sum_{i=1}^{N} p_i v_{i,t} \tag{110}$$

以及惩罚项

$$\left(y_t - \sum_{i=1}^{N} p_i v_{i,t} \right)^2. \tag{111}$$

现在，约束的形式为

$$c = \sum_{t=1}^{D} \sum_{\alpha<n_{\min,t}} \delta^{\alpha}_{y_t} = 0. \tag{112}$$

请注意，我们还可以使用代价函数，将最小化班次数量的代价函数与此约束结合起来。

$$f = \sum_{t=1}^{D} \left(\sum_{i=1}^{N} p_i v_{i,t} - n_{\min,t} \right)^2 . \tag{113}$$

最后，关于护士 i 最多应连续工作 $d_{\max}$ 个班次的约束可表示为

$$c(i) = \sum_{t_1 - t_2 > d_{\max}} \prod_{t'=t_1}^{t_2} v_{i,t'} = 0. \tag{114}$$

e. 参考文献 该问题可见参考文献 [49]。

D. 实变量问题

有些问题由一组实变量$\{v_i \in \mathbb{R}, i = 1, ..., d\}$，以及一个需要最小化的代价函数 $f(v_i)$ 定义。与前面讨论的离散问题一样，我们希望将这些问题表示为哈密顿量，使其基态代表最优解。这将分三步完成；首先，我们将连续变量编码为离散变量。其次，我们必须像之前一样，选择一种将这些离散变量编码为自旋变量的方法；最后，还需要一个解码步骤，将离散问题的解映射回连续定义域。接下来，我们将探讨两种可用于第一步的方法。

a. 标准离散化 给定一个向量 $v \in \mathbb{R}^d$，可以通过将坐标轴划分为长度为期望精度 q的区间来对其进行离散化，即将 $v = (v_i)$ $i = 1$, ..., d 的分量映射到 $v_i \to \tilde{v}_i \in \mathbb{N}$，使得 $(\tilde{v}_i - 1)q \le v_i < \tilde{v}_i q$。

根据问题的不同，为不同坐标轴设置不同的分辨率或采用非均匀离散化也可能很有用，例如对区间长度进行对数缩放。

对于离散变量 $\tilde{v}_i$，可以使用第 III 节中的编码。在最后一步中，找到中间量解 $\tilde{v}^*$ 后，必须将其映射回 $\mathbb{R}^d$，方法是对 v^* 的各个分量进行均匀采样，采样范围为对应于 $\tilde{v}^*$ 的超长方体。任何对离散编码有效的中间量解也都可以译码，因此，代价函数中除离散编码产生的项之外，不存在其他核心项。

b. 随机子空间编码 将向量 $v \in \mathbb{R}^d$ 编码为离散变量的另一种可能方法是随机子空间编码 [50]。首先随机选择一组坐标 $D_n := \{1, 2, \ldots, d_n\} \subset \{1, 2, \ldots, d\}$。对于每个 $i \in D_n$，使用狄利克雷过程在 $\mathbb{R}^d$ [51] 中沿 i 坐标方向选取一个区间 $[a_i, b_i]$ 。我们记

$$\begin{aligned} \pi_i : \mathbb{R}^d &\to \mathbb{R} \\ x = (x_1, \ldots, x_d) &\mapsto \pi_i(x) := x_i \end{aligned} \tag{115}$$

表示向 i 个坐标的投影。一个超矩形 $R \subset \mathbb{R}^d$ 定义为

$$R := \left\{ x \in \mathbb{R}^d \,\middle|\, \pi_i(x) \in [a_i, b_i], \forall\, i \in D_n \right\}. \tag{116}$$

对于选定坐标构成的固定集合 D_n，运行狄利克雷过程 m 次，以获得 m 个超矩形$\{R_1, \ldots, R_m\}$。对于 $k = 1$, ..., m，我们定义二进制投影映射

$$\begin{aligned} z_k : \mathbb{R}^d &\to \mathbb{Z}_2 \\ x \mapsto z_k(x) &:= \begin{cases} 1, & \text{if } x \in R_k \\ 0, & \text{else.} \end{cases} \end{aligned} \tag{117}$$

然后，随机子空间编码定义为一个映射

$$\begin{aligned} z : \mathbb{R}^d &\to \mathbb{Z}_2^m \\ x \mapsto z(x) &:= (z_1(x), \ldots, z_m(x)) . \end{aligned} \tag{118}$$

根据超矩形集合 $\{R_1, \cdots, R_n\}$，随机子空间编码可以是一种稀疏编码。令$M := \{1, \ldots, m\}$ 和 $K \subset M$。我们定义集合

$$U(K) := \bigcap_{k \in K} R_k - \bigcup_{i \in M - K} R_i \neq \emptyset . \tag{119}$$

对于 $z \in \mathbb{Z}_2^m$，令 $K(z) = \{k \in \{1, \ldots, m\} \,|\, z_k = 1\}$。二进制向量 z 属于随机子空间编码的像，当且仅当

$$U(K(z)) \neq \emptyset \tag{120}$$

成立。此后，只需使用第III 节讨论的离散编码，将 $z_i(x)$ 的分量映射为 $z(x)$ 的自旋变量。由于可能采用稀疏编码，必须在代价函数中添加一个核心项。令 $N = \{L \subset M \,|\, U(L) = \emptyset\}$，则核心项为

$$c = \sum_{L \in N} \prod_{\ell \in L} \delta^1_{z_\ell} \prod_{p \in M - L} \delta^0_{z_p} . \tag{121}$$

尽管存在这一缺点，与标准离散化的超立方体相比，随机子空间编码可能更可取，因为它简单易用，而且只需相对较少的超矩形便能达到高分辨率。

1. *Financial crash problem*

a. 描述 我们将根据参考文献 [52, 53] 中的一个简单模型，计算n 机构的市场价值 v_i, $i = 1$, ..., n 的金融均衡。在该模型中， m 资产的价格标记为 p_k, $k = 1$, ..., m。此外，我们定义所有权矩阵 D，其中 D_{ij} 表示 i持有资产 j 的比例；交叉持股 C，其中 C_{ij} 表示机构 i 持有机构 j 的比例（˜自持股除外）；以及自有所有权矩阵 C。

该模型假设，在没有崩盘的情况下，股权价值 V（满足 $v = CV$）在均衡时满足

$$V = Dp + CV \to v = \tilde{C}(1-C)^{-1}Dp. \tag{122}$$

随后，将崩盘建模为某机构所持资产价格的突然变化，即通过

$$v = \tilde{C}(1-C)^{-1}(Dp - b(v,p)), \tag{123}$$

其中 $b_i(v,\ p) = \beta_i(p)(1-\Theta(v_i - v_i^c))$，这使问题变得高度非线性。

b. 变量 对市场价值进行平移很有用，由此得到一个变量 $v - v^c = v' \in \mathbb{R}^n$。由于崩盘函数 $b_i(v,\ p)$ 明确依赖于 v' 的各个分量，因此使用随机子空间编码（或仅对各个分量分别编码）并不方便，所以我们采用标准离散化，即 v' 的每个分量都取离散值 0，...，K，从而以所需分辨率 r 达到截断值 rK。

c. 代价函数 为了确保系统处于金融均衡状态，我们只需将方程（123）平方

$$f = \left(v' + v^c - \tilde{C}(1-C)^{-1}(Dp - b(v,p))\right)^2, \tag{124}$$

对于 $b(v,\ p)$ 中的 θ 函数，我们采用

$$\Theta(v_i') = \sum_{\alpha \geq 0} \delta^{\alpha}_{v_i'}, \tag{125}$$

如第 VIII A4 节所述。

2. *Continuous black box optimization*

a. 描述 给定一个函数 $f\colon [0,1]^d \to \mathbb{R}$，该函数可以在有限次单独的点上求值，但没有可用的闭式表达式，我们希望找到其全局最小值。在经典优化中，通用策略是利用机器学习，根据一些样本求值结果学习一个闭式的解析采集函数 $g(x)$，优化该函数以生成下一个用于对 f 求值的点，并在资源允许的情况下重复这一过程。

b. 变量 变量的数量和取值范围取决于连续到离散编码。对于标准离散化，我们有 d 个变量 v_a，其取值属于 $\{0, ..., \lceil 1/q \rceil\}$，其中 q 是精度。

对于随机子空间编码，我们有 Δ 个变量 v_a，其取值属于 $\{0, ..., d_s\}$，其中 Δ 是矩形的最大重叠度，而 d_s 是矩形的数量。

c. 代价函数 与经典策略类似，我们首先使用试探形式拟合/学习一个采集函数。该试探形式可以表示为

$$y(v,w) = \sum_{r=1}^{k} \sum_{i_1,...,i_r=0,...,\Delta} w_{i_1,...,i_r} v_{i_1} ... v_{i_r}, \tag{126}$$

其中，k 是试探形式的最高阶数，变量 v_i 取决于连续到离散编码。用指示函数表示，它们可写为

$$v_i = \sum_{\alpha} \alpha \delta^{\alpha}_{v_i}. \tag{127}$$

或者，也可以将试探形式写成仅以指示函数为自变量的形式，而不是以变量为自变量。

$$y'(v,w) = \sum_{r=1}^{k} \sum_{i_1,...,i_r=0,...,\Delta} w'_{i_1,...,i_r} \delta^{\alpha_{i_1}}_{v_{i_1}} ... \delta^{\alpha_{i_r}}_{v_{i_r}}. \tag{128}$$

项数大致与之前相同，但这样做的优点是，代价函数中的能量尺度可以低得多。缺点是，通常要使优化优于随机采样，需要假设函数 f 以某种方式表现良好（例如是解析的）。方程 128 中的试探形式可能无法像第一种形式那样利用这一假设。

令 $w^* \equiv \{w^*_{i_1,...,i_r}\}$ 表示试探形式的拟合参数。那么，代价函数就是 $y(v,\ w^*)$ 或 $y'(v,\ w^*)$。

d. 约束 在这个问题中，可能出现的唯一约束是核心项。对于标准离散化，不需要这样的项；对于随机子空间编码，我们加入方程（121）。

e. 参考文献 此问题见参考文献 [54]。

3. *Nurse scheduling with real parameters*

下面我们将重新审视护士排班问题（见第VIIC 4节），其中护士的最低工作量$n_{\min,\ t}$和护士功率 p_i i 均为实值参数。代价函数和约束的一般结构将与第VII C4节中的相同；当我们选择将代价函数与每个班次具有最低工作量的约束结合起来，如方程（113）所示时，无需进行任何修改。不过，使用方程（110）中定义的辅助变量 y_t 来实现约束时，有几种可选方案，而通过考察这些方案所得出的结论可能对其他问题有所帮助。一种可能性是使用方程（110）及实值参数 p_i，这样y_t 将只取有限多个值，我们可以用某种方式为这些值标记。然而，问题在于，这些值的数量会随着变量 N数量的增加而呈指数增长。另一种方法是将 y_t 的值离散化，即把 $y_{\max} \equiv \sum_{i=1}^{N} p_i$ 划分为离散区间。但这同样不是最优方案，因为此时将辅助变量绑定到其取值的约束（例如方程（111）中的惩罚项）很可能无法得到精确满足，而剩余的惩罚会偏向于$v_{i,\ t}$ 的构型，在这些构型中该惩罚较低，但用于优化的代价函数可能并未达到最小值。

因此，自然的选择是直接将参数 $n_{\min}$、t、p_i 离散化，并重新缩放，使其取整数值。也就是说，我们将 $p_i \to \tilde{p}_i \in \{1,...,K\}$替换为相应表达式，其中根据 $Kr = \max_i p_i$ 和精度 r 设定 K，并且有 $\tilde{p}_i r \le p_i < (\tilde{p}_i+1)r$。

E. 其他问题

在这一最后类别中，我们纳入了不属于前述分类、但构成量子优化重要应用场景的问题。

1. *Syndrome decoding problem*

a. 描述 对于一个 $[n,k]$ 经典线性码（即将 k 逻辑比特编码为 n 物理比特的码），校验矩阵 H 指出物理比特状态 y是否为码字，或是否具有非零错误综合 $\eta = yH^T$。给定这样的综合，我们希望对其进行译码，即找出具有该综合的最可能错误，这等价于求解

$$\underset{e\in\{0,1\}^n, eH^T=\eta}{\operatorname{argmin}} \quad wt(e), \tag{129}$$

其中 wt 表示汉明重量，所有运算均为模 2。

b. 代价函数 该问题有两种不同的表述方式：基于校验矩阵的方法和基于生成矩阵的方法。在基于生成矩阵的方法中，我们注意到该码的生成矩阵 G 满足 $GH^T = 0$，因此任意逻辑字 u 都可通过 $e = uG+v$得到 $eH^T = \eta$ 的一个解，其中 v 是满足 $vH^T = 0$ 的任意状态，并且可以高效地找到。最小化 $uG + v$ 的权重即可得到代价函数

$$f_G = \sum_{j=1}^{n}(1-2v_j)\left(1-2\sum_{l=1}^{k}\delta^1_{u_l}G_{jl}\right). \tag{130}$$

注意，对 l 的求和是模 2 运算，而方程的其余部分是在 $\mathbb{N}$ 上进行的。我们将其改写为

$$f_G = \sum_{j=1}^{n}(1-2v_j)\prod_{l=1}^{k}s_l^{G_{jl}} \tag{131}$$

根据第 VIIIC 1节。

在基于校验的方法中，我们直接用代价函数最小化相对于 $eH^T = \eta$ 的偏差

$$f_1 = \sum_{j=1}^{n}(1-2\eta_j)\prod_{l=1}^{k}s_l^{H_{jl}} \tag{132}$$

但我们还必须通过以下项对权重更高的错误施加惩罚

$$f_2 = \sum_{j=1}^{n}\delta^1_{e_j} \tag{133}$$

因此我们有

$$f_H \equiv c_1 f_1 + c_2 f_2 \tag{134}$$

其中参数 c_1/c_2 为正。

c. 变量 基于校验的表述中的变量是误差 n 位 e_i。在基于生成元的表述中，定义了逻辑字 e 的 k 位 u_i。如果采用方程（u 131）中的重新表述，则这些变量会被替换为 自旋变量 k。注意，状态 s_l假定由高效的经典计算给出。

d. 约束 不存在硬约束；任意逻辑字 u 和任意物理状态 e 都是有效的。

e. 资源 通过分析两种方法所需的资源，可以发现若干有趣的权衡。基于校验的方法所采用的代价函数最多包含 $(n-k)+n$ 项（对于含有 $n-k$行的非退化校验矩阵，其中有 $(n-k)$ 项来自 f_1，另有 n 项来自 f_2）；而在f_G 中，最多有 n 项（即 G的行数）。基于生成元的表述中，这些项所涉及的变量的最高阶数由 G 的行数限定，为 k。一般而言，对于 f_H，这一阶数最高可达 n（即 H的列数）；但对于一类重要的线性码（低密度奇偶校验码），其阶数则受奇偶校验的常数权重限制。这个权重可以相当低，但这里同样存在权衡：校验的权重越高，编码率就越高（也就是说，对于优质的低密度奇偶校验码，它们会提高常数编码率 $n/k \sim \alpha$）。

f. 参考文献 此问题最初发表于参考文献 [55]。

2. *k-SAT*

a. 描述 一个 k-SAT 问题实例由一个布尔公式组成

$$f(x_1,...,x_n) = \sigma_1(x_{1_1},...,x_{1_k})... \wedge \sigma_m(x_{m_1},...,x_{m_k}) \tag{135}$$

该公式采用合取范式（合取范式），即 σ_i 是由 k 个文字构成的析取子句：

$$\sigma_i(x_{i_1},...,x_{i_k}) = l_{i,1} \vee \cdots \vee l_{i,k}, \tag{136}$$

其中，文字 $l_{i,\ k}$ 是一个变量 $x_{i,\ k}$ 或其否定 $\neg x_{i,\ k}$。我们希望确定是否存在一种变量赋值，使该公式成立。

b. 变量有两种策略可以用哈密顿量表述来表示 ak-SAT 问题。首先，可以使用基于违反的子句的哈密顿量代价函数。在这种情况下，变量就是赋值$x \in \{0,1\}^n$。第二种方法则是根据 CNF 中的 k-SAT 实例构造一个图，具体如下。每个子句σ_j 都将构成一个由 k 个变量x_{i_1}、.....、x_{i_k}组成的完全连接图。连接来自不同子句的两个顶点

，当且仅当它们互为否定时，即$y_{i_l} = \neg y_{j_p}$，并求解该图的最大独立集（MIS）问题 [44]。当且仅当该集合的基数为 m 时，该 SAT 实例才可满足。这种表述中的变量是 $m \times k$ 布尔变量 x_i：若顶点属于独立集，则取值为 1，否则取值为 0。

c. 代价函数 对于以合取范式表示的问题，可以将基于子句违约的代价函数写为

$$f_C = \sum_j (1 - \sigma_j(x)) \tag{137}$$

其中（见第 VIIIC 2节）

$$\begin{aligned} \sigma_j &= l_{j_1} \vee l_{j_2} \ldots \vee l_{j_k} \\ &= 1 - \prod_{n=1}^{k} (1 - l_{j_n}). \end{aligned} \tag{138}$$

MIS问题的代价函数可以由一个鼓励独立集具有更大基数的项构造：

$$f_B = -b \sum_{j=1}^{m \times k} x_j. \tag{139}$$

d. 约束 在最大独立集表述中，必须确保最大独立集的成员之间不存在连接：

$$c = \sum_{(i,j) \in E} x_i x_j = 0. \tag{140}$$

如果将该约束实现为能量惩罚$f_A = ac$，则它应具有更高的优先级。从 VIIID 2节中的自旋翻转（参见 f_A）的最小代价为 $a(m-1)$，而在 f_B 中，自旋翻转可能带来的最大收益为 b。因此 $a/b \gtrsim 1/(m-1)$。

e. 资源 代价函数 f_C 最多由$\mathcal{O}(2^k mk)$项组成，这些项在自旋变量中的最高阶数为 k 。在替代方法中，阶数仅为二次，因此自然可以采用 QUBO 形式。然而，f_{MIS} 中的项数可能高达 $\mathcal{O}(m^2 k^2)$，因此随着子句数量增加，其扩展性更差。

f. 参考文献此问题见参考文献 [44]。

3. *Improvement of Matrix Multiplication*

a. 描述 矩阵乘法是信息学中最常用的数学运算之一。为简明地介绍这一过程，我们将讨论实数上的矩阵乘法。

给定一个实数 $n \times m$ 矩阵 A 和一个实数 $m \times p$ 矩阵B，通常的矩阵乘法算法需要 npm 次乘法操作。不过，也有资源消耗更少的算法，例如参考文献 [56] 中的算法。Strassen 证明，存在一种将两个2×2 矩阵相乘的算法，只需进行 7 次乘法，但需要执行更多的加法操作；而加法操作的计算成本较低。

在一篇近期论文 [57] 中，研究人员使用机器学习方法找到了新的有效矩阵乘法算法。他们将寻找更有效算法（即使用更少的乘法操作）表述为一个优化问题。我们将展示一个与之等价的伊辛哈密顿量优化问题。

矩阵 A 和 B 的乘法可以用一个 $nm \times mp \times np$ 阶张量 $\mathcal{M}$ 来表示。通常的矩阵乘法算法对应于将 $\mathcal{M}$ 分解为以下形式的 nmp 项之和：

$$\mathcal{M} = \sum_{\alpha=0}^{nmp-1} \mathbf{u}^{(\alpha)} \otimes \mathbf{v}^{(\alpha)} \otimes \mathbf{w}^{(\alpha)}, \tag{141}$$

其中，对于任意 $\alpha = 0$、$\cdots$、nmp 和 $\mathbf{v}^{(\alpha)}$，$\mathbf{u}^{(\alpha)}$ 是一个含有二进制条目的 nm 维向量；$\mathbf{w}^{(\alpha)}$ 是二进制编码的 mp 维和 np 维向量。如何使用张量方程 (141) 乘两个矩阵，参考文献 [57, 的算法 1] 中有所描述。将矩阵 A、B 和 C 分别映射为相应长度的向量，例如，一个 $n \times m$ 矩阵 A 将对应一个 nm 维向量，其第 k 个条目为分量 A_{rs}，其中 $k = rm + s$。然后先计算一个中间量 nmp 维向量 $\mathbf{m} = (m_0, \cdots, m_{nmp-1})$，其计算方式为

$$m_i = \left(\sum_{j=0}^{nm-1} u_j^{(i)} A_j \right) \left(\sum_{k=0}^{mp-1} v_k^{(i)} B_i \right) \tag{142}$$

对于所有 $i = 0, \ldots, nmp-1$。在第二步中，通过以下方式确定最终矩阵 $C = AB$ 的分量 $\{C_i\}_{i=0,\ldots,np-1}$

$$C_k = \sum_{r=0}^{nmp-1} w_k^{(r)} m_k, \tag{143}$$

对于所有 $k = 0, \ldots, np-1$。提高矩阵乘法效率相当于寻找 $\mathcal{M}$ 的和分解，使其项数更少（秩更低），也就是说，我们希望找到向量 $\mathbf{a}^{(\beta)}$、$\mathbf{b}^{(\beta)}$、$\mathbf{c}^{(\beta)}$，其中 $\beta = 0, \ldots, B-1$ 和 $B < nmp$，使得

$$\mathcal{M} = \sum_{\beta=0}^{B-1} \mathbf{a}^{(\beta)} \otimes \mathbf{b}^{(\beta)} \otimes \mathbf{c}^{(\beta)} \tag{144}$$

成立。采用这个更短的张量分解执行上述算法，相当于一种涉及更少标量乘法的矩阵乘法算法。因此，新算法效率更高。一般来说，我们可以允许这些向量

$$\left\{ \mathbf{a}^{(\beta)}, \mathbf{b}^{(\beta)}, \mathbf{c}^{(\beta)} \right\}_{\beta=1,\cdots,B-1}$$

具有实数条目。为简化处理，我们将条目限制在有限集 $K := \{-k, -k+1, \ldots, k-1, k\}$其中 $k \in \mathbb{N}$，而参考文献 [57] 报告了关于$k = 2$的有希望的结果。不过，张量 $\mathcal{M}$ 的条目是二进制编码。允许分解向量取更多值，只会扩大可能解的搜索空间。

b. 变量问题的变量由$\mathcal{M}$的和分解中各向量的元素给出。也就是说，存在由变量组成的向量

$$\left\{\mathbf{a}^{(\alpha)}\right\}_{\alpha=1,\ldots,nmp-1}$$

，其中$\mathbf{a}^{(\alpha)} = (a_0^{(\alpha)}, \ldots, a_{nm-1}^{(\alpha)})$ 的每个元素 $a_i^{(\alpha)}$ 都是取值范围为 K 的离散变量。类似地，我们有变量向量 $\left\{\mathbf{b}^{(\alpha)}_{\alpha=0,\ldots,nmp-1}\right\}$，其中 $\mathbf{b}^{(\alpha)} = (b_0^{(\alpha)}, \ldots, b_{mp}^{(\alpha)})$，$b_i^{(\alpha)}$ 的取值范围为K ；还有变量向量 $\{\mathbf{c}^{(\alpha)}\}_{\alpha=0,\ldots,nmp-1}$ ，其中$\mathbf{c}^{(\alpha)} = (c_0^{(\alpha)}, \ldots, c_{np-1}^{(\alpha)})$，且 $c_i^{(\alpha)}$ 的取值范围同样为 K。

c. 代价函数 问题的代价函数为

$$f_{\mathcal{M}} := \sum_{\alpha=0}^{nmp-1} \left(1 - \prod_{i,j,k} \delta^0_{a_i^{(\alpha)}} \delta^0_{b_j^{(\alpha)}} \delta^0_{c_k^{(\alpha)}}\right). \tag{145}$$

请注意，对于固定的 α，这一项

$$\left(1 - \prod_{i,j,k} \delta^0_{a_i^{(\alpha)}} \delta^0_{b_j^{(\alpha)}} \delta^0_{c_k^{(\alpha)}}\right) \tag{146}$$

当且仅当以下三个向量 $\mathbf{a}^{(\alpha)}$、$\mathbf{b}^{(\alpha)}$ 和$\mathbf{c}^{(\alpha)}$都为零时才会消失。因此，这个代价函数会惩罚和分解中的较高秩（141），从而使基态表示一种更有效的乘法算法。

d. 约束 我们必须施加约束，使得到的张量分解确实与矩阵乘法张量 $\mathcal{M}$ 相一致。因此，约束为

$$\mathcal{M}_{ijk} = \sum_{\alpha=0}^{nmp-1} \mathbf{a}_i^{(\alpha)} \mathbf{b}_j^{(\alpha)} \mathbf{c}_k^{(\alpha)} \tag{147}$$

对于任意 $i = 0, \ldots, nm-1$，$j = 0, \ldots, mp-1$ 和$k = 0, \ldots, np-1$。

请注意，约束的数量以及代价函数中的项数都会随着待相乘矩阵的维数迅速增加。

VIII. 构件总结

这里总结了本库问题中反复用作构件的部分代价函数和技术。

A. 辅助函数

最简单的一类构件是关于多个变量的辅助标量函数，可通过惩罚项直接用于代价函数或约束中。

1. *Element counting*

最常见的构件之一是函数 t_a，它只对具有给定值 a 的变量 v_i 的数量进行计数。根据值指示函数，我们有

$$t_a = \sum_i \delta^a_{v_i}. \tag{148}$$

引入 t_a 作为额外变量可能会很有用。在这种情况下，必须通过约束将其绑定到所需的值

$$c = \left(t_a - \sum_i \delta^a_{v_i}\right)^2. \tag{149}$$

2. *Step function*

阶跃函数 $\Theta(v-w)$ 可以构造为

$$\Theta(v-w) = \sum_{\beta \geq \alpha}^{K} \delta^\beta_v \delta^\alpha_w = \begin{cases} 1 & \text{if } v \geq w \\ 0 & \text{if } v < w \end{cases}, \tag{150}$$

其中 K 是 v, w 可以取的最大值。阶跃函数可用于惩罚 $v \geq w$ 的构型。

3. *Minimizing the maximum element of a set*

阶跃函数特别适用于最小化集合 $\{v_i\}$ 的最大值。给定一个辅助变量 l，我们保证对于所有 i，$l \geq v_i$，并施加惩罚项

$$f = 1 - \prod_{i=1}^{n} \Theta(l - v_i) = \begin{cases} 1 & \text{if } l \geq v_i \ \forall i \\ 0 & \text{if } \exists\, i: \ v_i > l, \end{cases} \tag{151}$$

当 l 小于任意 v_i时，会增加能量。可以通过在问题的代价函数中加入 l的值，来最小化 $\{v_i\}$ 的最大值。在这种情况下，我们还必须将方程（151）中的项乘以 l 可能取到的最大值，以避免在惩罚项上作出取舍。

4. Compare variables

给定两个变量 v, $w \in [1, K]$, 以下项表示 v 和 w 是否相等:

$$\delta(v-w) = \sum_{\alpha=1}^{K} \delta_v^\alpha \delta_w^\alpha = \begin{cases} 1 \text{ if } v = w \\ 0 \text{ if } v \neq w. \end{cases} \tag{152}$$

如果我们想检查 $v > w$是否成立，则使用阶跃函数方程(150)。

B. 约束

这里介绍一类特殊的变量函数，其基态满足有用的约束。这些函数自然可作为通过惩罚项施加约束的构件。

1. All (connected) variables are different

若有一个变量集合 $\{v_i \in [1, K]\}_{i=1}^N$，且当邻接矩阵 A 的元素 A_{ij} 为 1 时，两个变量 v_i, v_j 相连，则以下项在所有相连的 $i \neq j$都满足 $v_i \neq v_j$ 时取最小值:

$$\begin{aligned} c &= \sum_{i \neq j} \delta(v_i - v_j) A_{ij} \\ &= \sum_{i \neq j} \sum_{\alpha=1}^{K} \delta_{v_i}^\alpha \delta_{v_j}^\alpha A_{ij} \end{aligned} \tag{153}$$

c的最小值为零，当且仅当不存在满足 $v_i = v_j$的相连变量对 i, j 时，才可能达到该最小值。对于全连接，可以使用这一构件来约束所有变量彼此不同。若 $K = N$, 那么对所有 $i \neq j$ 均满足条件 $v_i \neq v_j$，就等价于要求 K 个可能值中的每一个都由某个变量取到。

2. Value α is used

给定一个由 N 变量 v_i组成的集合，我们想知道是否至少有一个变量取值为 α。这可以通过以下项实现

$$u_\alpha = \prod_{i=1}^{N} \left(1 - \delta_{v_i}^\alpha\right) = \begin{cases} 0 & \text{if } \exists\, v_i = \alpha \\ 1 & \text{if } \nexists\, v_i = \alpha. \end{cases} \tag{154}$$

3. Inequalities

a. 单变量不等式 如果离散变量 v_i 可取 $1, ..., K$ 中的值，并且受到一个不等式 $v_i \leq a'$ 的约束，则可以通过对所有不满足该不等式的取值施加能量惩罚来实现

$$c = \sum_{K > a > a'} \delta_{v_i}^a. \tag{155}$$

也可以使用加权惩罚项，例如:

$$c = \sum_{K \geq a > a'} a \delta_{v_i}^a, \tag{156}$$

如果该不等式并非硬性约束，且更严重的违反应受到更重的惩罚，这种做法可能会很有用。不过，其缺点是通常会在系统中引入更高的能量尺度，尤其是在 K 较大的情况下，这可能会减小相对能隙。

b. 不等式约束 如果问题受到以下不等式约束的限制:

$$c(v_1, \ldots, v_N) < K, \tag{157}$$

可以方便地定义一个辅助变量 $y < K$，并施加以下约束:

$$(y - c(v_1, \ldots, v_N))^2 = 0, \tag{158}$$

因此， c 必定等于 y中的某个值，而 y 唯一可能的取值是满足该不等式的那些值。

如果辅助变量 y 采用二进制编码或格雷编码表示，那么当$2^n < K < 2^{n+1}$ 对于某个 $n \in \mathbb{N}$ 时，必须修改方程(158)。

$$\left(y - c(v_1, \ldots, v_N) - 2^{n+1} + K\right)^2 = 0, \tag{159}$$

这样可确保当 $y = 0$、.....、$2^{n+1} - 1$ 时， $c(v_1, \ldots, v_N) < K$。

4. Constraint preserving driver

使用量子退火或 QAOA 等受退火启发的算法时，除了在代价函数中添加惩罚不满足约束的状态的惩罚项之外，还可以让算法从满足约束的状态开始，并使用不会使系统离开满足约束的子空间的驱动哈密顿量[6, 38]。如第V节所述，我们要求驱动哈密顿量与生成约束的算符对易，并且能够遍历整个有效搜索空间。任意多项式约束 c 例如都可以通过奇偶映射来处理。它将约束转化为形式 $\sum_{\mathbf{u} \in c} g_{\mathbf{u}} \sigma_z^{(\mathbf{u})}$。从满足约束的状态开始，并采用由 σ_+、 σ_- 算符构造的、依赖于约束的翻转 - 交换项作为映射后的自旋上的驱动项，便可自动满足这些约束。

C. 问题特定表示

针对一些应用广泛的问题，我们展示了将其映射为代价函数的实用技术。

1. Modulo 2 linear programming

对于线性方程集合

$$xA = y, \tag{160}$$

给定 $A \in \mathbb{F}_2^{l\times n}$，$y \in \mathbb{F}_2^n$，我们希望求解 $x \in \mathbb{F}_2^l$。代价函数

$$f = \sum_{i=1}^{n}(1-2y_i)\left(1-\sum_{j=1}^{l}2x_jA_{ji}\right) \tag{161}$$

使 xA 与 y 之间的汉明距离最小化，因此基态代表一个解。如果我们在固定 i的情况下考虑第二个因子，会发现它统计了在 x （模 2）中、 A_{ji} 在相应索引处不为零的位置上 1 的数量。当与

$$H_i = \prod_{j=1}^{l}\sigma_{z,j}^{A_{ji}} \tag{162}$$

作用于 $|x\rangle$ 时，我们得到相同的结果，因此代价函数

$$f = \sum_{i=1}^{n}(1-2y_i)\prod_{j=1}^{l}s_j^{A_{ji}}, \tag{163}$$

（带有自旋变量 ） s_j 的基态就是方程（160）的解。

2. Representation of boolean functions

给定一个布尔函数 $f\colon \{0,1\}^n \to \{0,1\}$，我们希望将其表示为关于 n 个布尔变量的代价函数。一般来说这很困难 [58], ，但对于局部布尔函数（及其组合），存在简单的表达式。特别是，我们有例如，

$$\neg x_1 = 1 - x_1 \tag{164}$$

$$x_1 \wedge x_2 \wedge \ldots \wedge x_n = \prod_{i=1}^{n} x_i \tag{165}$$

$$x_1 \vee x_2 \vee \ldots \vee x_n = 1 - \prod_{i=1}^{n}(1-x_i) \tag{166}$$

$$(x_1 \to x_2) = 1 - x_1 + x_1x_2 \tag{167}$$

$$\mathrm{XOR}(x_1,x_2) \equiv x_1 + x_2 \bmod 2 = x_1 + x_2 - 2x_1x_2. \tag{168}$$

总是可以使用 Tseytin 变换 [59] ，将一个 k-SAT 实例转换为 3-SAT；更一般地，也可以将任意布尔公式转换为合取范式的 3-SAT，且公式规模仅增加线性开销。

请注意，为了对优化问题进行编码，可以采用不同的表达式；这些表达式只需与此处所示的表达式（相差一个常数偏移量）一致即可，以表示所讨论问题的基态/解。例如，如果我们对 $x_1 \wedge \ldots \wedge x_n$ 的满足赋值感兴趣，可以用两种不同的方式来构造代价函数，且两者都以 $x_1 = \ldots = x_n = 1$ 为唯一基态；具体而言，$f = -\prod_{i=1}^{n} x_i$ 和 $f = \sum_{i=1}^{n}(1-x_i)$。相应的两个自旋哈密顿量将具有截然不同的性质：用自旋变量表示时，第一个包含 2^n 个最高阶为 n 阶的相互作用项，而第二个则包含 n 个线性项。此外，在第二种方案中，越接近满足赋值的构型（也就是说， x_i 中取值为 1 的变量越多），其成本越低；第一种方案则并非如此。请注意，对于 $x_1 \vee \ldots \vee x_n$，没有类似的技巧，因为我们希望恰好惩罚一种构型。

D. 元优化

最后，我们介绍构造代价函数时可采用的若干选项。其中包括构件系数等元参数，也包括处理优化问题时反复使用的方法和技巧。

1. Auxiliary variables

通常，将一组变量 v_i 的信息合并到一个辅助变量 y 中很有用，这个辅助变量随后可用于代价函数的其他部分。例如，元素计数构件，或一组变量的极大值；我们展示了如何将辅助变量的值约束为该变量集合的这一函数值。如果函数 f 可以用变量值/指示函数的代数形式表示（即有限多项式），自然的做法是添加约束项

$$c\,(y - f(v_i))^2 \tag{169}$$

使用适当的系数 c。为避免引入约束带来的弊端，在某些情况下，可以采用另一种更可取的途径。这种替代方案只需使用变量指示函数$\delta_y^{\alpha}{}_{(v_i,\ \delta_{v_i}^{\beta})}$

并根据所选编码，用自旋/二进制变量表示变量指示函数（甚至不必对 y和 v_i使用相同的编码）。例如，考虑机器调度问题 VIIC 3中的辅助变量 $\tau_{m,t}$，我们需要表示 $\delta^{\alpha}_{\tau_{m,t}}$。为简单起见，我们对$v_{m,t}$ 和 $\tau_{m,t}$选择独热编码，从而得到

$$\delta^{\alpha}_{\tau_{m,t}} = x_{\tau_{m,t},\alpha} = \begin{cases} (1 - x_{v_{m,t},0}) & \text{if } \alpha = t \neq 0 \\ x_{v_{m,t},0} & \text{if } t = 0 \\ 0 & \text{else,} \end{cases} \tag{170}$$

其中，$x_{v_{m,t},\beta}$ 是 $v_{m,t}$独热编码的二进制变量。

2. *Prioritization of cost function terms*

如果代价函数由多个项构成，

$$f = a_1 f_1 + a_2 f_2, \tag{171}$$

通常需要让某一项优先于另一项，例如，第一项编码了硬约束时。一种确保 f_1 不会与 f_2 相互“权衡”的策略是，在 f_1 中计算 Δ_1 的最小代价Δ_2 ，并计算 f_2 中自旋翻转一次所带来的最大收益。分别对这两项独立进行计算，并假设状态接近最优状态，效率可能更高。然后可以根据 $c_1\Delta_1 \gtrsim c_2\Delta_2$设置系数。一般而言，这取决于编码，原因有二：首先，对于某些编码，必须在代价函数中添加核心项，并在确定优先级时将其纳入考虑（这些项通常具有最高优先级）。此外，在某些编码中，单次自旋翻转可能导致变量的价值变化超过 1。[2] 尽管如此，在编码无关的表述中，针对代价函数各项组成的项对，分别考察变量值变化 1 时引入的上述“成本”和“收益”，并据此确定它们的相对系数，仍可作为一种高效的启发式方法。这样，在选定编码后，只需确定核心项的系数。

3. *Problem conversion*

许多与离散优化问题相关的判定问题都是NP完全的，这意味着总能将它们映射到任意其他NP完全问题，而所需的经典运行时间相对于系统规模仅有多项式开销。因此，对于一个尚无代价函数表述的新问题，可以通过经典方式将其映射到另一个已有此类表述的问题。不过，由于预计量子算法对一般NP难问题只能带来多项式优势，因此最好仔细分析这种开销。也可能存在权衡，例如第 k VII E 2节中的-SAT示例：在原始表述中，代价函数项的阶数为 k ；而在最大独立集表述中，我们自然得到一个QUBO问题，其项数随子句数增长得更快。

[2] 例如，在二进制编码中，当变量取值为 1，……，K 时，自旋翻转最多可导致变量偏移 $K/2$ 。这正是采用格雷编码等改进方法的主要动机。

IX. 结论与展望

本文汇集并详述了多种优化问题，这些问题都可以表述为自旋哈密顿量，从而能够在量子计算机上求解。我们首先以与编码无关的方式介绍这些优化问题。只有在为问题的变量选择编码之后，才能得到可输入量子退火或QAOA等算法的自旋哈密顿量。用自旋哈密顿量表述优化问题本身就包含若干选择，其中之一便是编码。归根结底，这种自由度源于：问题的解只编码在哈密顿量的基态中，而激发态并未固定。

这一点可以通过第 IIIC 节中独热核心项的表示示例来说明，该节给出了两种满足独热条件的方法。如果我们只能将问题转换为 QUBO 形式，那么相互作用阶数较低的方法更优；但采用奇偶映射和相应架构时，可以更好地处理高阶相互作用项，因此项数更少的哈密顿量可能更可取。

由此我们强调，为优化问题求取自旋哈密顿量涉及多种选择。我们认为，这并非负担，而是一项优势。通过作出不同选择，可以提升量子算法的性能。这是本文的第二项主要贡献：我们识别出优化问题建模中常见且反复出现的构件。这些构件包括编码、代价函数的组成部分，以及施加约束的不同方法，共同构成了优化问题的构造工具包。这为自动化预处理算法铺平了道路，使其能够从构造工具包中挑选最佳方案，从而便于寻找性能最优的量子算法。

为量子计算机的特定硬件平台寻找最优自旋哈密顿量，本身就是一个重要问题。不同硬件平台的最优自旋哈密顿量很可能各不相同，因此需要一种能够针对任意给定平台定制问题的流程。我们不依赖特定硬件的方案，是朝这一方向迈出的关键第一步。

在第VI 节中，我们还通过两个示例展示了如何使用构件，以及在构造自旋哈密顿量时如何进行选择。这些示例可作为模板，以系统化的方式求解定制优化问题。

该计划迈向自动化的下一步，是针对选定问题开展一些明确的测试。具体而言，我们计划对变量的不同编码进行基准测试。最终，测试和基准测试结果将纳入一条流程管线，以自动搜索能够提升量子算法性能的哈密顿量表述方式。

X. 致谢

作者感谢 Kaonan Micadei 博士富有成效的讨论。本研究得到奥地利科学基金（FWF）的支持，资助来源为 START资助，项目编号 Y1067-N27，以及 SFB BeyondC 项目，编号 F7108-N38，QuantERA II计划协议编号101017733项下的资助、联邦经济事务与气候行动部通过QuaST项目提供的资助，以及联邦教育与研究部根据德国联邦议院决议提供的资助。

[1]Boris Melnikov，《离散优化问题——一些新的启发式方法》，载于 *Eighth International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05)* 2005 73 82 （IEEE，）第–页。[2] Marco Dorigo和Gianni Di Caro，《蚁群优化：一种新的元启发式方法》，载于 *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)* 2 1999 1470 1477 ，第卷（IEEE，）第–页。[3] Nina Mazyavkina、Sergey Sviridov、Sergei Ivanov和Evgeny Burnaev，《组合优化的强化学习：综述》，Comput. Oper. Res. 134，105400（2021）。[4]Edward Farhi、Jeffrey Goldstone、Sam Gutmann和Michael Sipser，《通过绝热演化实现量子计算》，arXiv:quant-ph/0001106（2000），10.48550/ARXIV.QUANT-PH/0001106。[5]Edward Farhi、Jeffrey Goldstone和Sam Gutmann，《量子近似优化算法》，arXiv:1411.4028（2014），10.48550/ARXIV.1411.4028。[6] Stuart Hadfield、Zhihui Wang、Bryan O'Gorman、Eleanor G Rieffel、Davide Venturelli和Rupak Biswas，《从量子近似优化算法到量子交替算子拟设》，Algorithms 12，34（2019）。[7] Naeimeh Mohseni、Peter L McMahon和Tim Byrnes，《伊辛机作为组合优化问题的硬件求解器》，Nat. Rev. Phys. 4，363–379（2022）。[8]Gary Kochenberger、Jin-Kao Hao、Fred Glover、Mark Lewis、Zhipeng L¨u、Haibo Wang和Yang Wang，《无约束二元二次规划问题：综述》，J. Comb. Optim. 28，58–81（2014）。[9] Itay Hen和Marcelo S. Sarandy，《量子退火中约束优化的驱动哈密顿量》，Phys. Rev. A 93，062312（2016）。[10] Itay Hen和Federico M. Spedalieri，《用于约束优化的量子退火》，Phys. Rev. Applied 5，034007（2016）。[11] Andrew Lucas，《许多NP问题的伊辛形式》，Front. Phys. 2，5（2014）。[12] Samuel A. Wilkinson和Michael J. Hartmann，《用于量子模拟与计算的超导量子多体电路》，Appl. Phys. Lett. 116，230501（2020）[13] Clemens Dlaska、Kilian Ender、Glen Bigan Mbeng、An-dreas Kruckenhauser、Wolfgang Lechner和Rick van Bijnen，《通过四体里德堡门实现量子优化》，Phys. Rev. Lett. 128，120503（2022）。[14] Niklas J. Glaser、Federico Roy和Stefan Fil-ipp，《由共享可调耦合器介导的超导量子比特受控受控相位门》，arXiv:2206.12392（2022），10.48550/ARXIV.2206.12392。[15] N. Chancellor、S. Zohren和P. A. Warburton，《超导系统中多体相互作用的电路设计》……量子退火系统，并应用于可扩展架构”，Npj Quantum Inf. 3, 21 (2017). [16] M. Sch¨ondorf 和 F.K. Wilhelm，“虚跃迁在四个耦合的人造原子中诱导的非成对相互作用”，Phys. Rev. Appl. 12, 064026 (2019). [17] Tim Menke、Florian H¨ase、Simon Gustavsson、Andrew J. Kerman、William D. Oliver 和 Al´an Aspuru-Guzik，“ 超导电路的自动化设计及其在 4- 局域耦合器中的应用”，Npj Quantum Inf. 7, 49 (2021). [18] Tim Menke、William P. Banner、Thomas R. Bergam- aschi、Agustin Di Paolo、Antti Veps¨al¨ainen、Steven J. Weber、Roni Winik、Alexander Melville、Bethany M. Niedzielski、Danna Rosenberg、Kyle Serniak、Mollie E. Schwartz、Jonilyn L. Yoder、Al´an Aspuru-Guzik、Simon Gustavsson、Jeffrey A. Grover、Cyrus F. Hirjibehedin、Andrew J. Kerman 和 William D. Oliver，“超导量子比特间可调三体相互作用的演示”，Phys. Rev. Lett. 129, 220501 (2022).[19] Yao Lu、Shuaining Zhang、Kuan Zhang、Wentao Chen、Yangchao Shen、Jialiang Zhang、Jing-Ning Zhang 和 Kihwan Kim，“任意离子量子比特上的全局纠缠门”，Nature 572, 363–367 (2019). [20] G Pelegr´ı、A J Daley 和 J D Pritchard，“通过双光子绝热快速通过实现高保真多量子比特里德堡门”，Quantum Sci. Technol. 7, 045020 (2022). [21] Kilian Ender、Roeland ter Hoeven、Benjamin E. Niehoff、Maike Drieb-Sch¨on 和 Wolfgang Lechner，“奇偶量子优化：编译器”，arXiv:2105.06233 (2021), 10.48550/arXiv.2105.06233.[22] Wolfgang Lechner，“使用可并行化量子门进行量子近似优化”，IEEE Trans. Quantum Eng. 1, 1–6 (2020).[23]Michael Fellner、Anette Messinger、Kilian Ender 和 Wolfgang Lechner，“通用奇偶量子计算”，Phys. Rev. Lett. 129, 180503 (2022).[24] Wolfgang Lechner、Philipp Hauke 和 Peter Zoller，“基于局域相互作用实现全连接的量子退火架构”，Sci. Adv. 1, e1500838 (2015). [25] Stuart Hadfield、Zhihui Wang、Eleanor G. Rieffel、Bryan O'Gorman、Davide Venturelli 和 Rupak Biswas，“带硬约束和软约束的量子近似优化”，*Proceedings of the Second International Workshop on Post Moores Era Supercomputing*，PMES' 17, 15–21 (2017).[26] Yingyue Zhu、Zewen Zhang、Bhuvanesh Sundar、Alaina M. Green、C. Huerta Alderete、Nhung H. Nguyen、Kaden R. A. Hazzard 和 Norbert M. Linke，“囚禁离子量子计算机上的多轮 QAOA 和高级混合算子”，arXiv:2201.12335 (2022), 10.48550/ARXIV.2201.12335.[27] Franz Georg Fuchs、Kjetil Olsen Lye、Halvor Møll Nilsen、Alexander Johannes Stasik 和 Giorgio Sartor，“约束

保持型量子近似优化算法混合算子”，Algorithms 15, 202 (2022).[28] Nicolas PD Sawaya、Albert T Schmitz 和 Stuart Had- field，“离散优化量子算法中的编码权衡与设计工具包：着色、路径规划、调度及其他问题”，arXiv:2203.14432 (2022), 10.48550/arXiv:2203.14432.[29] Maike Drieb-Sch¨on、Younes Javanmard、Kilian En- der 和 Wolfgang Lechner，“奇偶量子优化：编码约束”，arXiv:2105.06235 (2021), 10.48550/ARXIV.2105.06235.[30] Nicholas Chancellor，“用于量子退火和 QAOA 的离散变量畴壁编码”，Quantum Sci. Technol. 4, 045004 (2019).[31] Jie Chen、Tobias Stollenwerk 和 Nicholas Chancellor，“畴壁编码在量子退火中的性能”，IEEE Trans. Quantum Eng. 2, 1–14 (2021).[32] Kensuke Tamura、Tatsuhiko Shirai、Hosho Katsura、Shu Tanaka 和 Nozomu Togawa，“伊辛机中典型二进制整数编码的性能比较”，IEEE Access 9, 81032–81039 (2021). [33] Nicolas P. D. Sawaya、Tim Menke、Thi Ha Kyaw、Sonika Johri、Al´an Aspuru-Guzik 和 Gian Giacomo Guer- reschi，“面向光子、振动和自旋哈密顿量的 d 能级系统资源高效数字量子模拟”，Npj Quantum Inf. 6, 49 (2020).[34] Alejandro Montanez-Barrera、Alberto Maldonado-Romo、Dennis Willsch 和 Kristel Michielsen，“非平衡惩罚：一种为量子优化算法编码组合优化问题不等式约束的新方法”，arXiv:2211.13914 (2022), 10.48550/arXiv.2211.13914.[35]Jesse Berwald、Nicholas Chancellor 和 Raouf Dridi，“从理论和实验角度理解畴壁编码”，arXiv:2108.12004 (2021), 10.48550/ARXIV.2108.12004.[36] John Preskill，“NISQ 时代及之后的量子计算”，Quantum 2, 79 (2018).[37] Fernando Pastawski 和 John Preskill，“编码量子退火的纠错”，Phys. Rev. A 93, 052325 2016( ).[38] Kilian Ender、Anette Messinger、Michael Fellner、Clemens Dlaska 和 Wolfgang Lechner，“模块化奇偶量子近似优化”，PRX Quantum 3 (2022), 10.1103/prxquantum.3.030304.[39]Michael Fellner、Kilian Ender、Roeland ter Hoeven 和 Wolfgang Lechner，“奇偶量子优化：基准测试”，arXiv:2105.06240 (2021). [40] Josua Unger、Anette Messinger、Benjamin E. Niehoff、Michael Fellner 和 Wolfgang Lechner，“用于量子优化的奇偶约束低深度电路实现”，arXiv:2211.11287 (2022), 10.48550/ARXIV.2211.11287.[41] James King、Sheir Yarkoni、Jack Raymond、Isil Ozfidan、Andrew D King、Mayssam Mohammadi Nevisi、Jeremy P Hilton 和 Catherine C McGeoch，“局部崎岖性与全局挫折下的量子退火”，J. Phys. Soc. Jpn. 88, 061007 (2019).[42] Martin Lanthaler 和 Wolfgang Lechner，“ 优化问题奇偶校验表述中的最少约束 ”，New J. Phys. 23, 083039 (2021).[43] Sebastian Feld、Christoph Roch、Thomas Gabor、Chris- tian Seidel、Florian Neukart、Isabella Galter、Wolfgang Mauerer 和 Claudia Linnhoff-Popien，“使用量子退火机解决带容量限制的车辆路径问题的混合求解方法”，Front. ICT 6 (2019).[44]Vicky Choi，“针对 NP 完全最大权独立集、精确覆盖和 3-SAT 问题的绝热量子算法”，arXiv:1004.2226 (2010), 10.48550/ARXIV.1004.2226.[45] Krzysztof Kurowski、Jan Weglarz、Marek Subocz、Rafa-l R´o˙zycki 和 Grzegorz Walig´ora，“用于解决作业车间调度问题的混合量子退火启发式方法”，载于 *Computational Science – ICCS 2020*，由 Valeria V. Krzhizhanovskaya、G´abor Z´avodszky、Michael H. Lees、Jack J. Dongarra、Peter M. A. Sloot、S´ergio Brissos 和 Jo˜ao Teixeira 编（施普林格国际出版公司，2020），第 502–515 页。[46] Davide Venturelli、Dominic J. J. Marchand 和 Galo Rojo，“作业车间调度的量子退火实现”，arXiv:1506.08479 (2015), 10.48550/ARXIV.1506.08479.[47] Amaro、David、Rosenkranz、Matthias、Fitzpatrick、Nathan、Hirano、Koji 和 Fiorentini、Mattia，“变分量子算法用于作业车间调度问题的案例研究”，EPJ Quantum Technol. 9, 5 (2022).[48]Costantino Carugno、Maurizio Ferrari Dacrema 和 Paolo Cremonesi，“在 D-Wave 量子退火机上评估作业车间调度问题”，Scientific Reports 12, 6539 (2022).[49] Kazuki Ikeda、Yuma Nakamura 和 Travis S Humble，“量子退火在护士排班问题中的应用”，Sci. Rep. 9, 12837 (2019).[50]DA Rachkovskii、SV Slipchenko、EM Kussul 和 TN Baidyk，“随机子空间编码方案数值编码的性质”，Cybern. Syst. Anal. 41, 509–520 (2005).[51]Luc Devroye、Peter Epstein 和 J¨org-R¨udiger Sack，“关于随机区间和超矩形的生成”，J. Comput. Graph. Stat. 2, 291–307 (1993).[52] Matthew Elliott、Benjamin Golub 和 Matthew O Jack- son，“金融网络与传染效应”，Am. Econ. Rev. 104, 3115–53 (2014).[53]Rom´an Or´us、Samuel Mugel 和 Enrique Lizaso，“利用量子计算预测金融崩盘”，Phys. Rev. A 99, 060301 (2019).[54] Syun Izawa、Koki Kitai、Shu Tanaka、Ryo Tamura 和 Koji Tsuda，“结合量子退火与随机子空间编码的连续黑箱优化”，arXiv:2104.14778 (2021), 10.48550/ARXIV.2104.14778.[55] Ching-Yi Lai、Kao-Yueh Kuo 和 Bo-Jyun Liao，“通过量子近似优化进行综合译码”，arXiv:2207.05942 (2022), 10.48550/ARXIV.2207.05942.[56] Volker Strassen，“高斯消元并非最优”，Numer. Math. 13, 354–356 (1969).[57]Alhussein Fawzi、Matej Balog、Aja Huang、Thomas Hubert、Bernardino Romera-Paredes、Mohammadamin Barekatain、Alexander Novikov、Francisco J R Ruiz、Julian Schrittwieser、Grzegorz Swirszcz、 *et al.*，“通过强化学习发现更快的矩阵乘法算法”，Nature 610, 47–53 (2022).[58] Stuart Hadfield，“布尔函数和实函数在量子计算哈密顿量中的表示”，ACM Trans. Quant. Comput. 2, 1–21 (2021).[59] Grigori S Tseitin，“命题演算中推导的复杂性”，载于 *Automation of reasoning*(Springer

，1983），第 466–483 页。