arXiv:2301.01836v1 [quant-ph] 4 Jan 2023

一种用于代表性选择的量子启发式二元优化算法

Anna G. Hughes,[1] Jack S. Baker,[1] 和 Santosh Kumar Radha[1],*[1]
Agnostiq Inc.，325 Front St W，多伦多，ON M5V 2Y1（日期：2023年1月6日）

量子计算的进步正推动跨学科的新兴应用，包括金融领域。在这些领域中，量子算法和量子启发式算法如今可用于进行市场预测、检测欺诈和优化投资组合。为扩展这一工具箱，我们提出了选择器算法：一种从较大的数据集中选出最具代表性子集的方法。所选子集中的数据点同时满足两项要求：与邻近数据点的距离尽可能近，与较远数据点的距离尽可能远；其中，距离的具体定义可由任意核或广义相似度函数给出。用于编码上述要求的成本函数自然构成一个二次无约束二元优化（QUBO）问题，非常适合采用量子优化算法求解，包括量子退火。选择器算法可应用于多个领域，尤其适用于金融领域，可用于从更大范围的资产中构建多元化投资组合。在对合成数据集进行实验后，我们展示了选择器算法在真实数据中的两种应用：（1）使用一组股票近似重构纳斯达克100指数；（2）分散加密货币投资组合。在对应用案例（2）的分析中，我们比较了 D-Wave Systems 提供的两台量子退火器的性能。

I. 引言

从较大的数据集合中选取具有代表性的样本，通常称为代表性选择；与研究整个数据集相比，这种做法具有多种优势（例如，[1–4]）。代表性选择能够缩小数据的规模并降低其复杂性，从而简化数据分析与处理，并减少数据存储的内存成本。数据建模的计算效率也能得到显著提升，例如分类器训练和模型应用。代表性选择已应用于众多领域，从计算机视觉 [5] 和语言处理 [6] 到蛋白质分析 [7]。人们提出了多种代表性选择流程，以减少某些特定监督学习分类器的训练数据量 [8, 9]。除了需要额外知识进行代表性选择的方法外，人们对通过无监督方法寻找具有代表性的样本也日益关注 [10]。

在这项工作中，我们实现了一种无监督的代表性选择算法。无标签数据集可能包含数量未知的类别，也可能其中的数据未形成簇，或仅根据某种相似性概念松散地聚为不同类别。该算法旨在从较大的数据集中找出一组代表性数据点。所选子集应仅包含彼此不相似、但与未选中的邻近数据点相似的数据点。具体而言，我们的选择器算法会在包含 n 个聚类数据点的样本中找出 k 个最不相似的数据点。具体做法是同时最大化所选数据与所有其他数据之间的距离同时最小化数据集内所选数据与其他相似聚类数据之间的距离。返回的 k 个数据点既能代表各个数据簇，又与其他组保持最大距离。这种方法可应用于多个学科；本文探讨其在金融领域的一种应用：分散化。

在构建稳健的投资组合时，分散化是一项至关重要的策略。为降低市场下行时相互依赖的组成部分表现不佳所带来的风险，投资于彼此相关性不强的资产至关重要。投资组合分散化策略多种多样，从简单的 $1/N$ 规则 [11] 到投资组合维度或贝叶斯方法等更复杂的方法 [12, 13]。许多分散化方法旨在通过对资产进行加权，刻意降低投资组合的整体方差。近年来，机器学习在量化金融领域日益普及，使研究人员能够提出新的分散化方法。本文将我们的选择器算法用作资产分散化方法。在这一框架下，当每项资产都与其他资产尽可能不同，并且能够代表其所属行业或其他表现相似的资产时，投资组合便实现了良好的分散化。我们的选择器算法的核心框架可归结为二次无约束二元优化（QUBO）问题，可通过一系列元启发式算法 [14–17] 利用经典计算和量子计算求解。由于分散化（以及其他选择性表示问题）所涉及的变量数量庞大，最终得到的 QUBO 目标函数包含许多二元变量；在量子环境中，这意味着需要大量量子比特。尽管门模型量子计算机的量子比特数量持续增加，但目前，量子启发式硬件（如量子退火机）已能满足这一需求（尽管其量子比特并非通用）。

* research@agnostiq.ai

随后，我们将选择器算法视为量子启发式算法。

II. 选择器算法

选择器算法旨在通过寻找 QUBO目标函数的低成本解，从较大的数据集中挑选出独特且具有代表性的数据点。该函数经过构造，使得低成本解能够最大化所选数据点与所有其他数据之间的某种距离度量，从而确保选出的数据点彼此独特，同时最小化所选数据点与相似聚类数据之间的距离。这确保了数据集中的每个所选 k 数据点都能代表与其邻近点聚类相似的数据，而每个k 数据点彼此不相似。

输入数据以矩阵 $\mathbf{Y}$ 的形式提供，其中每一行都是代表一个 i^{th} 数据点的向量

$$\mathbf{Y} = \begin{bmatrix} \vec{y}_1 \\ \vec{y}_2 \\ \vdots \\ \vec{y}_i \\ \vdots \\ \vec{y}_n \end{bmatrix} = \begin{bmatrix} y_1^{(1)} & y_1^{(2)} & y_1^{(3)} & \dots \\ y_2^{(1)} & y_2^{(2)} & y_3^{(3)} & \dots \\ & \vdots & & \\ y_i^{(1)} & y_i^{(2)} & y_i^{(3)} & \dots \\ & \vdots & & \\ y_n^{(1)} & y_n^{(2)} & y_n^{(3)} & \dots \end{bmatrix}. \tag{1}$$

其中 n 是数据集的大小。尽管可以使用任何度量来评估各数据点之间的距离，但除非另有说明，本文均采用欧几里得距离作为距离度量，

$$d_{ij}(\vec{y}_i, \vec{y}_j) = \sqrt{\sum_m \left(\vec{y}_i^{(m)} - \vec{y}_j^{(m)}\right)^2}. \tag{2}$$

其他距离度量包括标准化欧几里得距离、余弦距离、闵可夫斯基距离等。距离 d 在形式上是一个核函数，原则上也可以利用量子/经典神经网络进行计算和/或学习（例如， [18]）。因此，可以构建选择器算法的跨范式变体，其中 d 按照 [18], 中的方案在门模型设备上进行评估，而 QUBO 目标函数则使用量子退火器进行近似求解。不过，这超出了本文的讨论范围。

QUBO 目标函数定义为

$$\mathcal{C}(\vec{x}) = \frac{1}{2k}\vec{x}\mathbf{d}\vec{x}^T - \frac{1}{n}\vec{x}\mathbf{d}\vec{1}^T + A\left(\sum_{i=1}^{n} x_i - k\right)^2, \tag{3}$$

其中， $\vec{1}$ 是由 n个 **1** 组成的向量，$\mathbf{d}$ 是距离矩阵，其元素 d_{ij} 如方程中所示

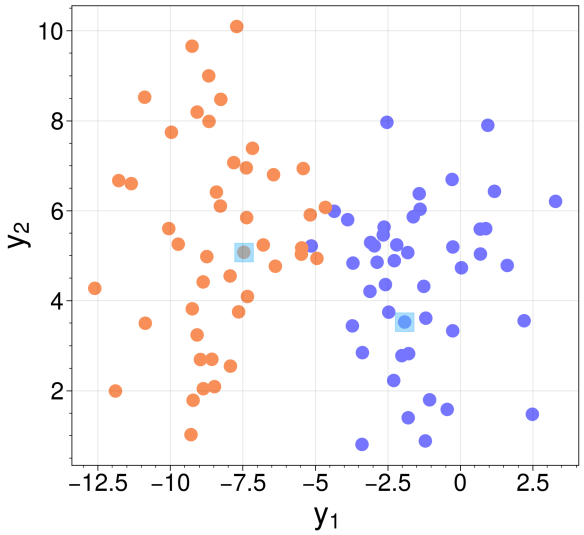


图 1. 选择器算法在聚类数据点上的性能示例。通过从二维平面上以两个点为中心的高斯分布中随机抽取数据点，生成这些簇。每个簇包含 90 个数据点，标准差为 2。选择器算法的任务是使用$qbsolv$ 求解器，从完整数据集中选出两个代表点。算法选出的点以蓝色方块标出，表明该算法成功地从每个簇中选出了具有代表性的点。

2（或其他用户定义的距离度量）， $\vec{x}$ 是$\{0,1\}^n$ 二元变量向量， A 是用于强制执行等式约束的惩罚缩放因子 $\sum x_i = k$。方程3 右侧的第一项表示所选数据点与簇内其他数据点之间的距离，确保选出的数据点能够代表其所属簇；第二项表示所选数据点与数据集中其他所有数据点之间的距离；最后一项则强制执行等式约束（即，如果选取的数据点多于 k 个，便会对成本函数施加惩罚）。这一优化问题可表示为 QUBO 形式，而此类问题通常是 NP困难。附录 A 对 QUBO模型进行了更详细的介绍，附录 $\vec{x}$ 则介绍了加权选择的形式化扩展（即，$\vec{x} \in \{0,1\}^n$可取多个离散值，而不局限于C）。采用这种形式后，该问题便可通过近似元启发式算法求解，包括第IVB节中使用的量子退火；我们将在附录 B 中对其进行定性讨论。

III. 合成数据实验

为展示选择器算法的基本功能，我们使用它从两个合成数据集

中选择数据点。一个数据集包含简单且明显聚类的数据点（按欧几里得距离衡量），另一个数据集则包含时间序列数据——即按时间顺序排列的数据点序列。我们展示了在这两种情况下，选择器算法都能做出合理且具有代表性的选择。在第一个应用中，我们使用选择器算法从一组聚类数据点中选出 2 个数据点。我们通过在二维平面上以两个点为中心的高斯分布中随机抽取数据点，生成两个簇（或称数据团）。每个数据团包含 90 个数据点，标准差为 2。所有数据点的坐标都作为选择器算法的输入，随后算法通过最小化方程中的成本函数，选出 k 具有代表性的数据点3，并使用 D-Wave 分解求解器 $qbsolv$ [19, 20]，该求解器使用经过修改的禁忌算法 [21]来最小化目标函数。在此示例中，我们生成了两个数据点簇，并要求选择器算法选出 $k = 2$具有代表性的数据点。数据团和选出的数据点如图1所示；正如预期，选择器算法成功地从每个簇中选出了一个数据点。由于采用欧几里得距离，选出的数据点与簇的大致中心非常接近。不过，并非所有度量方式都能呈现出这样的视觉效果；例如，詹森–香农散度等度量可能不存在视觉上可辨识的中心。

选择器算法同样可以选择任意高维度的数据点。我们通过两个示例对此进行演示：从三角函数曲线数组中选择数据点，以及从随机生成的随机微分方程（SDE）时间序列数组中进行选择。在这两种情况下，数据均为合成生成，但被聚类为两个不同的类别：在三角函数示例中为正弦或余弦，在 SDE 示例中为由耦合布朗过程生成的时间序列。在两种情况下，选择器算法都需要选择两个代表性数据点；在三角函数示例中，如果选择器算法选择了一条正弦曲线和一条余弦曲线，则认为解是成功的；在 SDE 示例中，如果选择器算法从不同过程中选择了代表点，则认为解是成功的。

三角函数时间序列使用以下方式生成：

$$\begin{aligned} y_i^{\text{sine}} &= sin(t_i) + n_i \\ y_i^{\text{cosine}} &= cos(t_i) + n_i, \end{aligned} \tag{4}$$

其中，t_i 的范围为 0 到 2π，n_i 是从均值为$\mu = 0$、标准差 σ 在 $\sigma = 0.1$ 到$\sigma = 5.5$之间的高斯分布中抽取的人工随机噪声。随着 σ 增大，由于引入的噪声主导了曲线的振幅，正弦曲线和余弦曲线之间的差异很小。我们生成 50 个正弦时间序列和 50 个余弦时间序列，使用 10 个不同的 σ 值，并让选择器算法在每种情况下选出 $k = 2$个代表性数据点。我们预计，当 σ 较低时，选择器算法会选出一个正弦数据点和一个余弦数据点；但随着 σ 增大，它应该会更难以区分这两个簇。

结果如图2所示。在前两个面板（a、b）中，正弦（绿色）和余弦（橙色）曲线的簇以时间为横轴绘制，其中 t 的范围从 0到 2π。左上面板中的曲线具有 $\sigma = 0.3$，而右上面板中的曲线具有 $\sigma = 3.0$。在每个图中，选择器算法选出的 2 条代表性曲线以黑色绘制。如预期，$\sigma = 0.3$ 时间序列曲线很容易区分，算法分别选择了一条。图（b）中，当标准差增加到 $\sigma = 3$时，正弦曲线和余弦曲线在视觉上已无法区分。

在 (c) 中，展示了相关距离矩阵各元素的密度图。每条向上延伸的水平线代表逐渐增大的噪声标准差，范围从 $\sigma = 0.1$ 到 $\sigma = 1.0$。正如预期，在较低的 σ下，两簇曲线明显分离，形成两个不同的峰；但随着 σ 增大，它们开始合并。这可以衡量数据集中各簇的可区分程度（或不可区分程度）如何随 σ变化。

在 (d) 中，绘出了所选解的准确率随 σ变化的情况。这里的准确率是指在总计 200 次试验中，包含一条正弦曲线和一条余弦曲线的算法解所占的百分比。正如预期，在较低的 σ下，选择器算法选出的解准确率为 100%。尽管在较高的σ下准确率有所下降，即使在 $\sigma > 3$ 处两组密度曲线（图 (c)）已无法区分，选择器算法仍能以高于随机猜测的准确率从每个簇中选出曲线。这表明，即使簇间的分离逐渐消失，选择器算法在选择代表性数据点时仍具有鲁棒性。

最后，我们测试选择器算法从二维随机微分方程的解中选取解的性能，该方程表示为

$$dx_i(t) = \mu_i x_i(t)dt + \sum_{j=1}^{2} \sigma_{ij} x_i(t) dW_j(t), \tag{5}$$

其中，μ_i 和σ_{ij} 分别为漂移向量和波动率矩阵，$W_j(t)$ 为标准布朗运动。对于模拟情形，我们从均匀分布 $[-1, 1]$ 中随机选取 μ_i 和 σ，并数值生成多个二维时间序列。

虽然三角函数时间序列展示了选择器算法从松散聚类或未聚类的数据中选出代表性数据点的能力，但在本例中，我们展示了选择器算法评估模拟金融数据的能力。与之前一样，选择器算法接收一个包含所有时间序列的数组，并被要求选出（事先已知的）$k = 2$ 代表性曲线。两组相关随机微分方程时间序列绘制于图3中，选择器算法选出的曲线以黑色标出。选择器算法成功地从每个时间序列簇中选出一条曲线，展示了其评估金融时间序列数据（如日收益率）的能力。

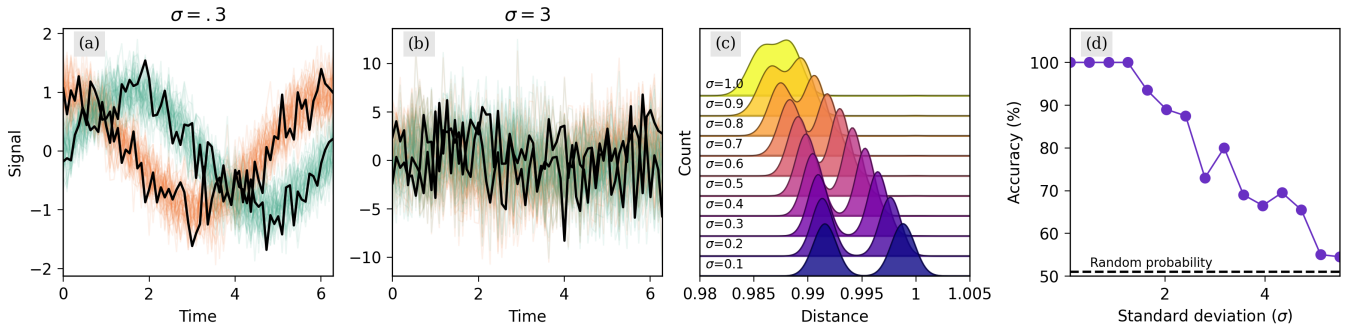


图 2. 根据方程生成的两簇各含 50 条正弦和余弦曲线4，作为选择器算法的输入，其中高斯噪声的标准差取不同值。*(a)*：引入高斯噪声 $\sigma = 0.3$ 后的 50 条正弦和余弦时间序列曲线分别以绿色和橙色绘制。算法选出的曲线以黑色绘制，可见算法从每个簇中各选出了一条曲线。*(b)*：将 50 条正弦和余弦时间序列曲线中的高斯噪声增大至 $\sigma = 3.0$。与之前相同，算法选出的曲线以黑色绘制，但由于噪声振幅过大，簇已无法区分。*(c)*：曲线密度显示为距离的函数，其中每条向上延伸的水平线表示噪声标准差逐渐增大。正如预期，在较低的 σ下，两簇曲线清晰地分离为两个不同的峰；但随着 σ 增大，它们开始合并。*(d)*：右下图绘出了所选解的准确率随 σ 变化的情况。此处的准确率以 200 次试验中算法解同时包含一条正弦曲线和一条余弦曲线的比例来衡量。在较低的 σ下，选择器算法选出的解准确率为 100%。尽管准确率在较高的 σ下有所下降，即使在 $\sigma > 3$ 时两组曲线已无法区分，选择器算法仍能以高于随机选择的准确率从每个簇中选出曲线。本例中的所有优化均使用$qbsolv$ 求解器完成。

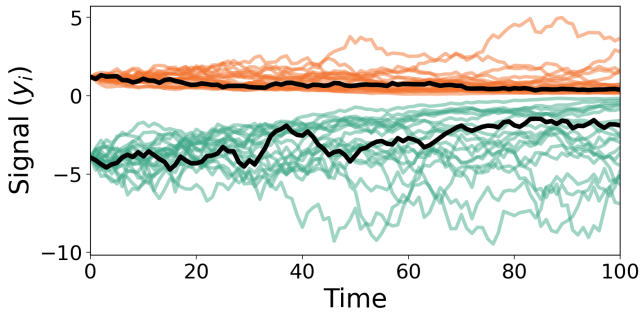


图3。两个簇，每个簇包含50条合成金融时间序列，按照 方程5 所示生成，分别以橙色（正）和青绿色（负）绘制。每个簇均使用从高斯分布中随机选取的相关性 $\mu = 0$ 和 $\sigma = 4$创建。对于每组曲线，其波动率和收益率均从高斯分布中随机选取，分布参数为 $\mu = -1$ 和 $\sigma = 1$。选择器算法的任务是从完整数据集中选取2个点。图中以黑色绘制的是选中的两条曲线，展示了该选择器算法能够可靠地从合成金融数据中选出具有代表性的时间序列。

IV. 用例：构建多元化投资组合

通过投资于不相关资产来分散投资组合，是降低特定市场下行或突发危机所带来风险的一种方法。由于选择器算法旨在挑选既独特又具有代表性的数据点，因此它对于从更广泛的资产列表中构建多元化投资组合尤其有用。量子退火器特别适合处理这一规模的 QUBO问题。也就是说，可以将大量变量映射到当前一代量子退火器上可用的大量（~ 1000）非通用量子比特。在第四节A中，我们将首先使用选择器算法作为投资组合多元化工具，旨在用较小的资产子集近似纳斯达克100指数的表现。对于这一任务，我们使用目前实际可用的经典大规模 QUBO 求解器。随后进入第四节B，在其中使用 D-Wave 的量子退火器对该算法进行实验。也就是说，我们将针对加密货币投资组合多元化这一任务，对两种量子设备的性能进行基准测试和比较。

A. 使用经典 QUBO 求解器重构纳斯达克100指数

选择器算法可以从市场指数中选取股票的代表性子集，从而构建多样化的投资组合。在这项针对真实金融数据的首次应用中，我们使用选择器算法，通过选取一个子集（S_k） k，从市场指数的全部 102 只股票中挑选股票，以近似重构纳斯达克100指数。我们使用 D-Wave 分解求解器$qbsolv$[19, 20]进行优化。

为了从纳斯达克100指数中选择股票，我们将每只股票的日收益率视为数组中的独立数据点。向量 $\vec{y}_i$ 由每只股票的日收益率组成

这些日收益率涵盖指数中每只股票一个交易年的表现，共253天，起始日期为2021-02-01 (YYYY-MM- DD)，结束日期为2022-02-01 (YYYY-MM-DD)。由此得到一个 102×253 矩阵 $\mathbf{Y}$，

$$\mathbf{Y} = \begin{bmatrix} \vec{y}_1 \\ \vec{y}_2 \\ \vec{y}_3 \\ \vdots \\ \vec{y}_n \end{bmatrix} = \begin{bmatrix} \overrightarrow{AAPL} \\ \overrightarrow{ABNB} \\ \overrightarrow{ADBE} \\ \vdots \\ \overrightarrow{ZS} \end{bmatrix} \tag{6}$$

以这种熟悉的形式，数据已可用于成本函数（方程3）进行评估，其中 $n = 102$ 是矩阵 $\mathbf{Y}$ 中向量的总数，k 是算法选出的目标股票数量，而$\mathbf{d}$ 的元素由相关距离给出，

$$d_{ab} = 1 - \frac{\sum(a_i - \bar{a}) \cdot \sum(b_i - \bar{b})}{\sqrt{\sum(a_i - \bar{a})^2 \sum(b_i - \bar{b})^2}}, \tag{7}$$

其中，$\bar{a}$ 和 b 分别是向量 $\vec{a}$ 和 b的均值。

所选股票可以任意接近于完整地重现市场指数；随着所选股票数量 k 增加，所选股票的组合数据会越来越接近对指数本身的精确重现。不过，我们的目标只是用较少的股票近似重现指数，因此选择大量股票未必有利，而股票数量是最终用户可调整的超参数。

在以下所有分析中，我们仅采用二元情形，即加权子空间仅限于二进制值，$\vec{w} \to \vec{x} \in \{0,1\}^n$（一般离散加权情形的讨论请参见附录C）。与前面的示例一样，这意味着具体的数据点要么被选中，要么不被选中，没有中间状态。这是一个 NP困难问题 [22]。为评估二元选择算法重现纳斯达克100指数的性能，我们通过对所有股票进行线性组合并对所得向量取平均值，构造一个代理纳斯达克100指数。请注意，虽然纳斯达克100指数按市值加权，但为简化起见，我们的纳斯达克100指数中的股票采用等权重。

我们首先使用选择器算法，从输入矩阵 $\mathbf{Y}$ 的收益率中选择两只股票。算法选出的股票是 福克斯公司（‘FOX’）和 新思科技（‘SNPS’）。我们使用这些股票创建 $S_{k=2} := S_2$ 这是一个数据点，由所选股票日收益率的线性组合平均值构成。左侧面板的图 S_2 显示了 S_2 和代理纳斯达克100指数的平均日收益率百分比直方图4。可以看出，所选S_2 股票的收益率特征在 $k = 2$时就已经很好地替代了纳斯达克100指数。为了进一步评估该算法选择的质量，我们计算了成本函数值以及代理纳斯达克100指数与某个数据点 S_2 之间的相关距离，并考虑两只股票的所有可能组合。图4的中间和右侧面板分别显示了成本函数值和相关距离值范围的直方图。 S_2 指数的成本函数值低于所有成本函数值中的 74% ，且相关距离小于所有值中的 89% 。

在图 5中，我们展示了代理纳斯达克100指数与 S_k 时间序列的平均收益率（左）和累计收益率（右），每一行都显示了 k逐步增加时的情况。正如预期，选取的股票数量（k）增加时， S_k 向量会越来越接近代理纳斯达克100指数。

最后，我们通过计算算法选定的指数 S_k与纳斯达克100指数之间的均方误差（MSE），比较二者。选取的股票数量 k 逐步增加，直至纳入指数中的全部102只股票。均方误差随 k变化的情况绘于图 6中，可以看出，只需约40只股票就能有效地再现纳斯达克100指数。

B. 使用量子退火机分散加密货币投资组合

从有限的指数中选取多样化的投资组合可能颇具挑战，尤其是当资产难以轻易划分为传统行业时，例如加密货币（crypto）。硬币之间意料之外的相关性可能使加密货币投资组合容易受到单一资产波动率的影响。因此，选择器算法特别适合用于构建多样化的加密货币投资组合。在这一应用场景中，我们使用选择器算法和量子退火机，从 Crescent 加密货币市场指数（CCMIX）中构建多样化的投资组合。我们比较了各设备的表现，考察每台设备满足问题约束的程度以及所得解的质量。

以下实验使用的两台量子退火设备是 D-Wave 2000 Q 和 D-Wave Advantage。D-Wave 2000Q 于2017年首次推出，是一台拥有2048个量子比特的量子退火设备。$2000+$ 量子比特以 Chimera 拓扑排列，配有6016个耦合器。2000Q 系统已用于解决从电子商务商品列表排序到密码学等问题 [23, 24]。D-Wave Advantage 量子处理单元（QPU）于2020年9月发布，是一套更新且更先进的系统。它包含 $5000+$量子比特，配有$35{,}000+$ 个耦合器——每个量子比特的耦合器数量从6个增加到15个。量子比特和耦合器数量的增加，使 D-Wave Advantage 量子处理器不仅能够解决比 D-Wave 2000Q 设备规模更大的问题，还能让具有更复杂连接性的问题更容易映射到设备的量子比特上，相较于2000Q 设备尤为如此。D-Wave Advantage 量子处理器已用于解决铁路调度和分子展开等问题 [25, 26]。

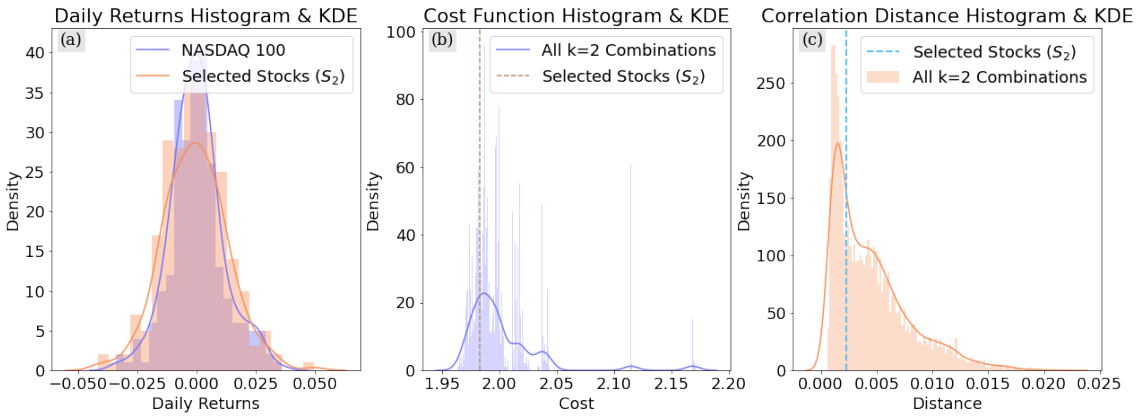


图 4. *(a)* 算法使用 *qbsolv* 求解器选出的 $k = 2$ 只股票（S_2）构成的综合指数与代理纳斯达克100指数的日收益率直方图及相应的核密度估计（KDE）图。*(b)* 纳斯达克100指数中所有两只股票组合的成本函数值直方图及相应的 KDE 图，虚线表示算法选出的 $k = 2$ 只股票对应的成本函数值。*(c)* 纳斯达克100指数中所有两只股票组合的相关距离直方图及相应的 KDE 图，虚线表示算法选出的 $k = 2$ 只股票对应的距离。

选择器算法的输入数据由 CCMIX 中每种硬币在七个月期间的日收益率组成，时间从2021-04-01（YYYY-MM-DD）开始，到2021-11-11（YYYY-MM-DD）结束。根据实验不同，选择器算法会从数组中选择指定数量的硬币，并使用指定的量子退火器来最小化成本函数。

1. 使用默认参数的约束满足

在第一轮实验中，我们考察每种设备选择满足方程 3 中惩罚项所施加等式约束的解的能力。对于某些经典方法，满足该约束可能轻而易举；但量子退火器受到的限制更多，尤其是在约束以惩罚项形式施加时 [27]。本节实验采用退火器的默认参数，尤其使退火时间为 $t = 20\,\mu s$。我们还采用了 $A = 2$的惩罚缩放因子。在每种设备上进行 300 次试验，选择器算法的任务是从 CCMIX 中的 18 种硬币里选出 $k = 3$ 种加密货币。所有实验中，每次运行的采样次数固定为 1000。

结果如图 7所示。每台设备满足 $k = 3$约束的能力存在显著差异。D-Wave 2000Q 设备在 300 次试验中仅有 49 次选择了三种加密货币，成功率仅为 16%。相比之下，D-Wave Advantage 量子处理器在 300 次运行中有 256 次选择了三种硬币，成功率超过 85%。显然，即使使用默认参数，较新的 D-Wave Advantage 量子处理器在性能上也有显著提升。

2. 不同退火时间下解的一致性

在后续实验中，我们研究了惩罚缩放因子 A （见公式 3）和退火时间对约束满足的影响。首先，我们在 11 次运行中研究了惩罚缩放因子的调节，对每个值进行 50 次重复试验，取值范围为 A，其数值从 $A = 0$ 到 $A = 10$ ，以固定增量变化，同时将退火时间固定为 $20\,\mu s$。在这些实验中，我们发现惩罚缩放因子对解的质量（即成本的高低）或问题约束的满足情况几乎没有影响。因此，在本节其余部分，我们将继续使用默认值 $A = 2$。最后，我们在每台设备上分别进行了 19 次试验，退火时间从默认值 20 μs 到990 μs不等，每个退火时间进行 50 次重复试验。

如图 8所示，退火时间的取值对满足 $k = 3$约束的解所占比例影响微乎其微。对于 Advantage 量子处理器而言，这一比例趋于稳定，因为它倾向于找到满足该约束的解。相反，延长退火时间有助于 2000Q 量子处理器找到更多满足软约束的解。即使退火时间较长，2000Q 设备选出的解大多仍不满足问题约束，而 Advantage 量子处理器在每种退火时间下满足该约束的比例均为 $> 85\%$ 。

退火时间对成本函数总体值（$k = 3$）可能产生的影响更难以 discern。对于中等退火时间 600 μs和较长退火时间 900 μs，2000Q 量子处理器上的成本函数值以及各次试验之间的标准差都降至较低水平。对于 Advantage 设备，则几乎无法观察到这种变化。

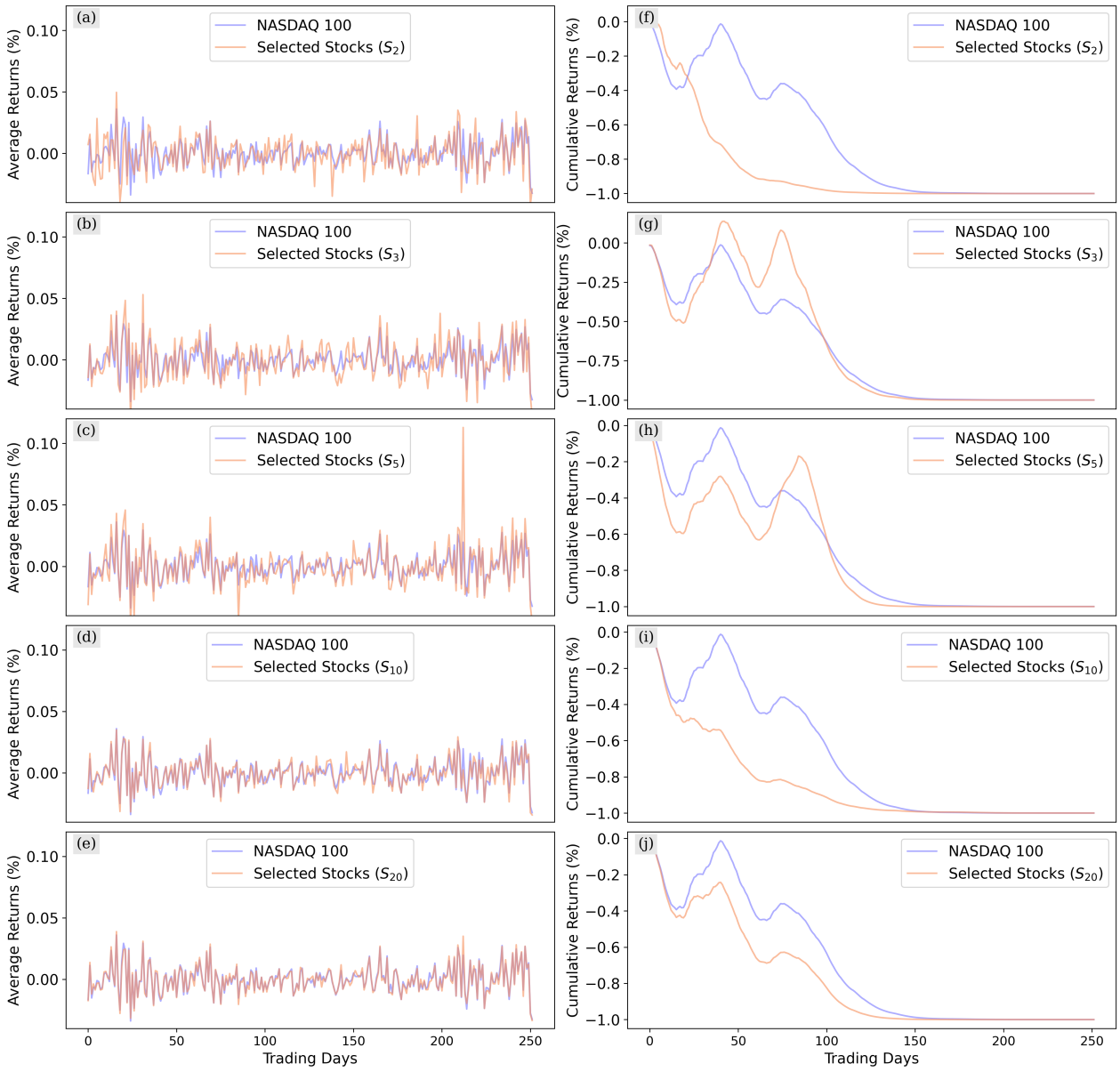


图 5. *(a)-(e)*：代理纳斯达克100指数（紫色）的所有日收益率，以及仅根据选择器算法所选股票生成的指数的所有日收益率。代理纳斯达克100指数通过对2021-02-01至2022-02-01（YYYY-MM-DD）期间纳斯达克100指数中的所有股票赋予相同权重而构建；选择器算法指数则仅对算法选出的股票赋予相同权重。所选股票的数量 k 会变化，从第一行的 $k = 2$ 开始，逐渐增加至最后一行的$k = 20$ 。随着 k 增加，选择器生成的指数会逐渐接近代理纳斯达克100指数，这符合预期。*(f)-(j)*：代理纳斯达克100指数和选择器算法生成指数的累计收益，其中每一行都表示所选股票数量 k逐渐增加。随着所选股票数量增加，选择器算法生成的指数会更准确地复现代理纳斯达克100指数的特征。

3. 比较解的质量

在满足约束 $k = 3$ 的条件下，我们考察各设备所选解的质量。为了了解量子处理单元所选的解与所有可能的解方面，我们首先计算每种硬币组合的成本函数值。CCMIX指数包含18种加密货币，因此共有 $2^{18} = 262144$ 种可能的组合，从选择0枚硬币到选择全部18枚硬币。

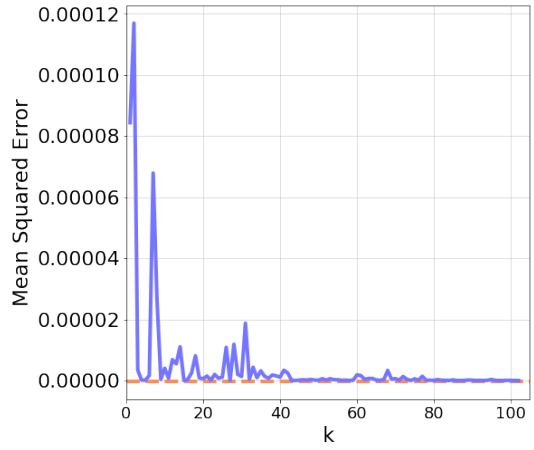


图 6. 代理纳斯达克100指数与选择器算法生成的指数之间的均方误差（MSE）随所选股票数量 k 的变化情况。代理纳斯达克100指数由2021-02-01至2022-02-01（YYYY-MM-DD）期间纳斯达克100指数中的所有股票等权重构成。相比之下，选择器算法生成的指数仅对算法选出的股票等权重构成。当所选股票数量从1只代表性股票增加到指数中的全部102只股票时，均方误差会急剧下降至零。

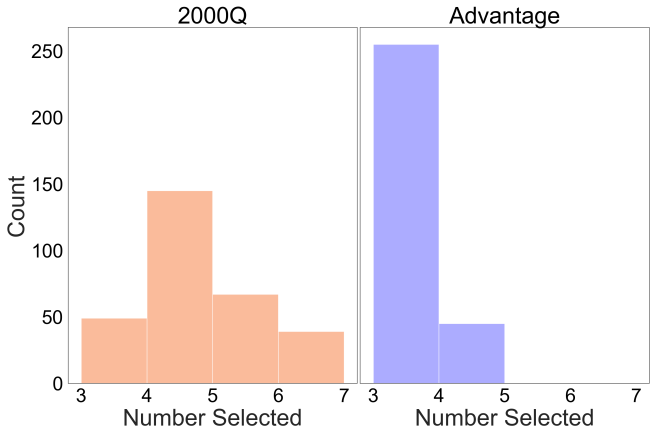


图 7. 选择器算法的任务是从包含18种加密货币的 CCMIX 中选出三种。优化分别使用 D-Wave 2000Q 设备（左）和 D-Wave Advantage设备（右）进行，每个量子处理单元进行了300次试验。D-Wave 2000Q 设备选出的解仅在16%的试验中满足问题约束，而 D-Wave Advantage设备选出的解在 > 85 %的试验中满足问题约束。

成本函数值的计算假设了$k = 3$ 约束，因此成本函数会对包含少于或多于三枚硬币的所有组合进行惩罚。$A = 2$ 在这些实验中。考虑所有组合后，成本函数的平均值为81。

为衡量量子设备相对于随机化解的性能，我们计算了各设备在所有试验中所选解的成本函数平均值。

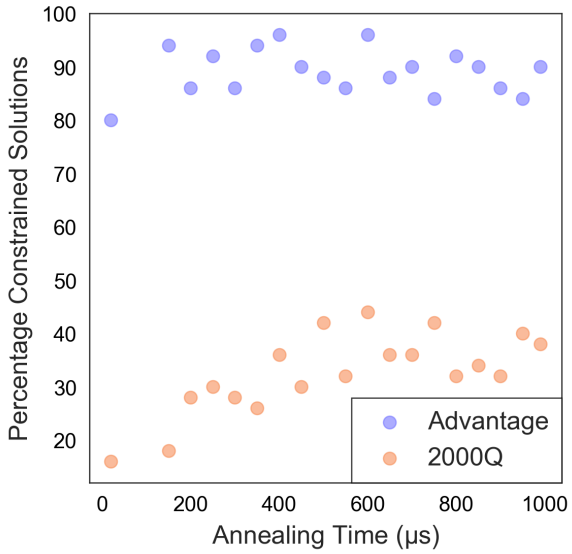


图 8. 我们研究了各设备选择满足问题约束的解的能力如何随退火时间变化，其中退火时间从 $20\mu s$ 增加到 $990\mu s$。图中绘出了 D-Wave 2000Q（米色）和 D-Wave Advantage（紫色）在各退火时间下（每个时间进行 50 次试验）满足问题约束的解所占百分比。退火时间的变化对 Advantage 量子处理器所选解中满足约束的比例影响可以忽略不计，而较长的退火时间似乎能让 2000Q 量子处理器更好地满足问题约束。不过，即使退火时间较长，2000Q 设备所选的大多数解仍不满足问题约束；而 Advantage 量子处理器所选解中，有 $> 85\%$ 在每个退火时间下都满足约束。

在各设备的最优退火时间下（2000Q 为 $550\mu s$， $900\mu s$ Advantage 为）。对于 2000Q 设备，成本函数的平均值为 4.02，而 Advantage 量子处理器的成本函数平均值为 0.32。结果如图9所示，图中绘制了成本函数值密度的直方图，竖线分别标示总平均成本值以及 2000Q 和 Advantage 量子处理器的成本值。

两种设备的平均成本函数值都显著低于考虑所有可能解时的平均值。2000Q 设备的成本函数平均值处于所有可能值中最低的 4%，而 Advantage 量子处理器的成本函数平均值则处于所有解中最低的 0.03%。总体而言，两种设备都明显展现出选择成本函数值较低的解的能力。不过，比较两种设备可以发现，Advantage 量子处理器在选择符合问题约束的解以及成本函数值更低的解方面，能力显著提升。

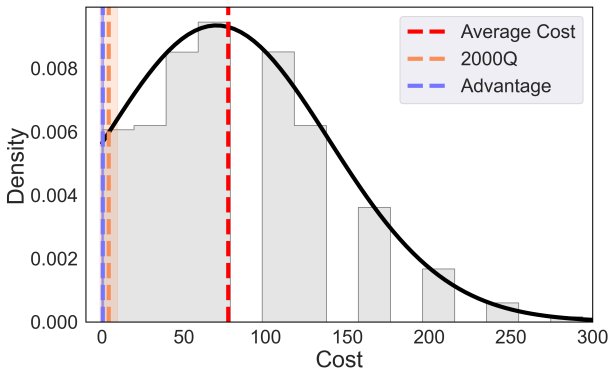


图 9. 以直方图绘出了所有硬币组合的成本函数值，并用黑色曲线拟合其分布。考虑所有可能的组合后，成本函数的平均值以红色垂直虚线表示。D-Wave 2000Q 和 Advantage QPU 所选解的平均成本函数值分别以蓝色和橙色水平虚线表示。

五、结论

在本研究中，我们设计了一种用于数据代表性选择的选择器算法，该算法依赖于求解 QUBO 问题。该选择器算法通过最大化所选数据点与数据集内所有其他数据点之间的距离，并最小化所选数据点与聚类相似的数据点之间的距离来选择数据点。鉴于生成的 QUBO 问题可能包含大量变量，选择器算法的优化阶段尤其适用于量子退火器等大规模 QUBO 求解器。

在对合成数据集进行实验后，在我们的第一个实际应用案例中，我们使用选择器算法，通过指数中的一部分股票近似纳斯达克100指数。我们通过对全部102只股票进行线性组合并取平均值，构建了代理纳斯达克100指数。选择器算法从指数中选出 k 只股票，将它们的表现进行组合并取平均，然后与代理纳斯达克100指数进行比较。随着所选股票数量 k增加，算法构建的指数与代理纳斯达克100指数越相似。我们将 $k = 2$ 点 S_2与所有可能的两只股票组合进行了比较。我们发现，算法选出的点 S_2的成本函数值处于所有 $k = 2$解中最低的26%。

在第二个应用中，我们使用选择算法，从 Crescent 加密货币市场指数中构建了一个加密货币资产多元化投资组合。我们使用两台量子退火器进行优化：D-Wave 2000Q 和 D-Wave Advantage。我们通过评估成本函数，并将其与指数中所有可能币种组合的成本函数值进行比较，来比较两台设备的 $k = 3$ 选择结果。我们发现，2000Q 设备所选解的平均成本函数值处于所有成本函数值的最低 4%，而 D-Wave Advantage设备的平均成本函数值则处于最低 0.03%。D-Wave Advantage设备也更稳定地选出符合问题约束的解： $> 85\%$ 的解符合约束，而 2000Q 的解中只有 16% 符合约束。总体而言，与较早的 2000Q 量子处理器相比，较新的 Advantage 量子处理器表现出明显提升，能够为组合优化问题提供有意义的解。

[1] B. J. Frey 和 D. Dueck，《科学》315，972（2007）。[2] Y. Yue 和 T. Joachims，载于第 25 届国际机器学习会议论文集（ICML '08）（ACM Press，2008）。[3] A. Rodriguez 和 A. Laio，《科学》344，1492（2014）。[4] A. Prasad、S. Jegelka 和 D. Batra，arXiv 电子预印本，arXiv:1411.1752（2014），arXiv:1411.1752 [cs.LG]。[5] E. Elhamifar、G. Sapiro 和 R. Vidal，载于 2012 年 IEEE 计算机视觉与模式识别会议（IEEE，2012），第 1600–1607 页。[6] R. Zhao 和 K. Mao，IEEE《模糊系统汇刊》26，794（2017）。[7] U. Hobohm、M. Scharf、R. Schneider 和 C. Sander，《蛋白质科学》1，409（1992）。[8] S. García、J. R. Cano 和 F. Herrera，《模式识别》41，2693（2008）。[9] E. Pękalska、R. P. Duin 和 P. Paclík，《模式识别》39，189（2006）。[10]S. Garcia、J. Derrac、J. Cano 和 F. Herrera，《IEEE模式分析与机器智能汇刊》34，417（2012年）。[11] V. Demiguel、L. Garlappi 和 R. Uppal，《金融研究评论》22（2009年）。[12] E. Brochu、M. W. Hoffman 和 N. de Freitas，CoRR abs/1009.5419（2010年），1009.5419。[13] M. Barkhagen、B. Fleming、S. Garcia Quiles、J. Gondzio、J. Kalcsics、J. Kroeske、S. Sabanis 和 A. Staal，arXiv电子印本，arXiv:1906.00920（2019年），arXiv:1906.00920 [q- fin.PM]。[14] F. Glover，《计算机与运筹研究》13，533（1986年），整数规划的应用。[15] F. Glover，《ORSA计算杂志》1，190（1989年）。[16] J. Beasley，伦敦，英格兰（1999年）。[17] I. Dunning、S. Gupta 和 J. Silberholz，《INFORMS计算期刊》30，608（2018年）。[18] S. K. Radha 和 C. Jao，arXiv电子印本，arXiv:2201.02310（2022年），arXiv:2201.02310 [quant-

ph]。[19] M. Booth、J. Berwald、U. Chukwu、J. Dawson、R. Dridi、D. Le、M. Wainger 和 S. P. Reinhardt，arXiv电子印本，arXiv:2005.11294（2020年），arXiv:2005.11294 [quant-ph]。[20] qbsolv¶（2022年）。[21] F. Glover，《INFORMS计算期刊》2，4（1990年）。[22] F. Barahona，《物理学杂志A：数学与一般》15，3241（1982年）。[23] N. Nishimura、K. Tanahashi、K. Suganuma、M. J. Miyama 和 M. Ohzeki，arXiv电子印本，arXiv:1903.12478（2019年），arXiv:1903.12478 [quant-ph]。[24] R. Mengoni、D. Ottaviani 和 P. Iorio，arXiv 电子印本，arXiv:2005.02268 （ 2020 年），arXiv:2005.02268 [quant-ph]。[25] K. Domino、M. Koniorczyk、K. Krawiec、K. Jałowiecki、S. Deffner 和 B. Gardas，arXiv电子印本，arXiv:2112.03674（2021年），arXiv:2112.03674 [quant-ph]。[26] K. Mato、R. Mengoni、D. Ottaviani 和 G. Palermo，arXiv电子印本，arXiv:2107.13607（2021年），arXiv:2107.13607 [quant-ph]。[27] I. Hen 和 F. M. Spedalieri，《物理评论应用》5，034007 2016 1508（），arXiv: 。04212 [quant-ph]。[28] F. Glover、G. Kochenberger 和 Y. Du，qubo模型的构建与使用教程（2019年），arXiv:1811.11538 [cs.DS]。[29] M. Zaman、K. Tanahashi 和 S. Tanaka，PyQUbo：用于将组合优化问题映射为QUBO形式的Python库（2021年），arXiv:2103.01708 [quant-ph]。

附录 A：二次无约束二元优化

二次无约束二元优化（QUBO）是指对含有二元变量、变量间耦合阶数至多为二次的目标函数进行优化。这种方法功能强大且应用广泛，因为许多组合问题都可以写成 QUBO 形式（见公式 A1），并使用通用 QUBO 求解器求解。这与其他更传统的方法形成对比，后者针对具体问题量身定制。一般而言，要近似求解 QUBO 问题（精确求解该问题属于 NP困难），我们必须从所有可能的解中选取一个近似最优解。

QUBO 的基本成本函数或目标函数 $\mathcal{C}(\vec{x})$ 为：

$$\mathcal{C}(\vec{x}) = \vec{x}^T \mathbf{Q} \vec{x}, \tag{A1}$$

其中 $\vec{x} = \{0,1\}^n$，且 $\mathbf{Q}$ 是一个 $n \times n$ 常数矩阵。QUBO 的目标是在 $\vec{x}$ 中找到二进制值的排列，使$\mathcal{C}(\vec{x})$的值最小（或最大）。除了向量 $\vec{x}$ 中的值必须为二进制值这一要求外，这个简单的 QUBO 模型没有任何正式约束。不过，所谓的软约束可以通过二次惩罚项编码到 QUBO 问题中（例如，参见公式 3右侧的最后一项）。软约束不会对问题施加严格限制，而是通过惩罚项阻止优化器选择会导致惩罚的解。

QUBO 问题近期备受关注，因为它们很容易映射到量子退火器和门模型量子计算机上。

附录 B：量子退火

在冶金学中，退火是通过有目的地对材料加热，使其变得更具延展性。退火过程首先将金属加热到再结晶温度以上，使金属中的原子快速移动。随后缓慢降温，原子便会沉降到具有所需性质的新晶体构型中。模拟退火是一种机器学习技术，其名称源自这一冶金过程。正如冶金退火使原子快速探索新的位置，模拟退火也会快速采样特定成本函数的解空间，并根据需要采用新的构型。采样速率逐渐降低，使退火算法最终收敛到一个优化后的解。

在模拟退火中，解从某个初始构型 C 开始，初始“温度”最高为 T_{max}。温度是一个超参数，它决定退火器以多大程度的随机性对成本函数进行采样。随后，状态 C 会转变为新的构型 N，进而可以计算成本函数值的变化。根据当前温度和成本函数值的变化，新构型 N 要么被接受为新状态，要么被拒绝。一个“好”的新构型 N，其成本函数值会低于先前构型的成本函数值，这表明状态已向最小值靠近。然而，即使是成本函数值变化为正的“差”构型，只要温度足够高，也可能被接受。与真实的冶金退火一样，温度随后会逐渐降低。这意味着，在初始阶段，除非成本函数值显著降低，否则较少有新构型会被接受。这种方法使模拟退火器能够广泛采样代价景观，并理想地收敛到全局最小值，而非任何局部最小值。

量子退火采用类似的技术，但利用量子涨落的强度作为采样率超参数，在解空间中探索量子态。量子退火成本函数是一个哈密顿量，用于表示系统在每个量子态下的总能量。最小化成本函数意味着找到能量最低的量子态。量子退火还具有量子隧穿这一额外优势，使算法能够在不同的低能量量子态之间隧穿，从而无需对这些状态之间的空间进行采样，并防止算法陷入局部最小值。随着采样率逐渐降低，量子隧穿半径也会随之减小。最初，算法会

获得一个较大的半径，以探索许多低能量量子态。随着系统演化且采样率降低，隧穿半径也会逐渐减小，算法最终稳定在一个状态上。量子退火提供的是近似解，不一定是绝对最优解。

附录 C：带权重的选择器算法

在正文中，我们探讨了从 n 个数据点中选择或不选择 k 个数据点的情形。由于对目标函数的加权贡献 w_i 被限制为 $w_i = x_i \in \{0,1\}$，因此这个问题本质上属于 QUBO。现在，我们将展示如何将这一形式扩展到更一般的情形：从 k 个总数据点中选出指定数量的点，并为每个选中的点分配权重，其精度由整数离散化数 n_D 控制。

第 i^{th} 个数据点的权重可以用 n_D 个二元变量展开，

$$w_i = w_{\min} + (w_{\max} - w_{\min}) \sum_{j=1}^{n_D} \frac{2^{j-1}}{2^{n_D}-1} X_{ij}, \qquad \text{(C1)}$$

分别对应最小/最大权重 $w_{\min/\max}$。在选择器算法的语境下，自然的选择是 $w_{min} = 0$ 和 $w_{max} = 1$。$X_{ij} \in \{0,1\}$ 是矩形 $n \times n_D$ 位矩阵的元素

$$\mathbf{X} = \begin{bmatrix} X_{1,1} & \dots & X_{1,n_D} \\ \vdots & \ddots & \\ X_{n,1} & & X_{n,n_D} \end{bmatrix}. \qquad \text{(C2)}$$

增加方程展开中使用的项数 C1 （即增加 n_D）会提高 w_i 的分辨率；在量子环境中，这需要付出增加量子比特需求 N 的代价，如 $N = n \times n_D$所示。当 $n_D = 1$时，得到原始的“已选择”或“未选择”情形 $w_i = X_{1,\,1} := x_i$。

现在，我们将新变量 χ_i 定义为权重与二元变量 x_i 的乘积：$\chi_i = w_i x_i$。现在，在公式 3 中，对于右侧的前两项，我们进行变量替换 $\vec{x} \to \vec{\chi}$，并在 $\vec{\chi}$ 各元素之和上添加一个新的惩罚项，从而得到目标函数

$$\mathcal{C}(\vec{\chi},\vec{x}) = \frac{1}{2k}\vec{\chi}\mathbf{d}\vec{\chi}^T - \frac{1}{n}\vec{\chi}\mathbf{d}\vec{1}^T + A\left(\sum_{i=1}^{n} x_i - k\right)^2 + B\left(\sum_{i=1}^{n}\chi_i - W\right)^2 \qquad \text{(C3)}$$

其中 W 是对 w_i 的预算约束：$\sum_i w_i = W$，而 B（例如 A）是惩罚缩放因子。如前所述，方程 C3 在二元变量中是四次的，但根据 [28], 中的讨论，它始终可以转换为二次形式，因此可归约为一个 QUBO 问题。虽然可以正式推导出这种转换，但我们使用 `PyQUBO` [29] 软件包，以智能方式将问题编译为 QUBO 形式。随后可以使用元启发式优化算法（包括量子退火）求解编译后的问题。

还应注意，通过对变量作一些特定选择，方程 C3 可以归约为正文中的方程 3。令 $w_{min} = 0$，$w_{max} = 1$ 且$n_D = 1$，每个 χ_i 都可化为 $\chi_i = x_i X_{1,\,1} = x_i^2 = x_i$。现在令 $W = k$，即可将两个二次惩罚项合并为一个，并得到方程 3 中的“已选择”或“未选择”形式。