使用 ZX 演算加速量子电路模拟

Tristan Cam[1] 和 Simon Martiel[1]

[1]Atos Quantum Lab，法国莱克莱镇

2023 年 5 月 5 日

arXiv:2305.02669v1 [quant-ph] 4 May 2023

摘要

我们提出了一种简单高效的方法，用于降低模拟量子电路时张量网络的收缩代价。我们首先将电路解释为 ZX 图，然后利用化简规则和局部补变换规则使其稀疏化。我们发现，优化类图 ZX 图可以将现有最先进方法的收缩代价降低几个数量级。特别是，对于深度为 20 的 Sycamore 电路，我们展示了平均收缩代价可提升 1180 倍，在峰值性能下最高可提升 4200 倍。

关键词： 量子计算、计算复杂性、树宽、张量网络、经典模拟、ZX演算、图重写

1 引言

人们普遍认为，使用经典算法模拟量子电路本质上是一项困难的任务。然而，随着量子计算领域迅速发展，我们也越来越需要更深入地理解经典计算能够触及的范围与即使借助大型高性能计算基础设施仍无法企及的范围之间的界限。尽管小型电路（即少于 30 个量子比特）的模拟可通过直接线性代数模拟轻松实现 [1, 2], ，但模拟更大的量子电路则需要采用不那么简单的方法。例如，矩阵乘积态（MPS）形式可在浅电路条件下对超大量子系统进行有损模拟 [3, 4]。另一种方法是将门集限制为非通用门集，以实现多项式时间模拟。这种方法已成功应用于 Clifford 电路 [5]、非 Clifford 旋转次数受限的电路 [6] ，以及匹配门/高斯算子 [7]。总的来说，尽管这些方法都在各自适用的领域取得了成功，但基于张量网络的方法 [8–15] 最有希望突破谷歌 [16]提出的“量子霸权”边界，并帮助我们洞悉经典计算的真正极限。与其他方法不同，张量网络缩并框架提供了一种通用工具，可用于表示和模拟各种量子计算。

张量网络收缩。在这种情形下，量子电路被解释为多重图（网络），其中节点包含电路中各门的矩阵（张量），边则表示两个节点矩阵之间的“点积”。这种表示称为张量网络。给定一个张量网络，我们可以收缩图中的每条边，迭代地合并节点，直到只剩下一个节点。我们决定收缩张量网络的顺序至关重要：不同的收缩顺序可能导致模拟时间相差数个数量级。因此，收缩顺序确定这一步骤在基于张量网络的模拟方法中发挥着重要作用。虽然已知寻找最优收缩顺序是 *NP-hard*[17], Markov 和 Shi 定义了张量网络的收缩复杂度 [8] ，并证明收缩所需时间仅随图的一种基于树宽的特定度量呈指数增长。之后，O’Gorman 使用另一种度量进一步完善了这一复杂度 [18]。

ZX演算与张量网络。 ZX演算是一种图形语言，可表示离散量子过程，尤其是后选择量子电路 [19, 20]。这种语言中的项是张量网络，可通过一组规则进行重写，并确保满足两个性质：

- 可靠性，即保持底层线性算子不变
- 完备性，即如果两个图实现相同的算子，则其中一个可以被重写为另一个 [21]。

ZX 图构成了一类结构良好的张量网络，其中的张量在计算基（Z-蜘蛛）或对角基（X-蜘蛛）下是稀疏的。这种结构尤其允许进行张量分解 – ，即将一个张量拆分为两个或更多张量；在近似模拟中，这一操作用于截断矩阵乘积态（MPS） – ，而且“无需额外代价”，因为它恰好对应于一种可以随时应用的图示重写规则。

因此，自然会产生这样的问题：缩并 ZX 图是否比缩并常规张量网络更有益？如果确实如此，我们希望 выяс明是否可以利用 ZX演算中的重写规则，在像处理张量网络一样进行缩并之前，先优化 ZX 图。

我们的贡献是一种在收缩顺序确定步骤之前，利用 ZX演算简化和优化量子电路的方法。我们提出了一种朴素的启发式方法，使用图论操作[22——这些操作与 ZX 形式体系兼容——来重写 ZX 图，并改进一个基于树宽 [8, 25, 26]的“代理”评分。我们表明，对于所研究的最大规模电路（深度为 20 的 Sycamore 电路 [16]），与当前最先进的收缩顺序确定算法 [10] 相比，收缩代价平均可以降低三个数量级 – ，在峰值性能下则可降低 4200 倍 – 。

在第 2 节中，我们将回顾张量网络与量子电路模拟之间的联系，并详细介绍本文所用的求阶启发式方法。在第 3 节中，我们将展示如何通过局部操作应用 ZX演算来改变目标张量网络的结构。在第 4 节中，我们将阐述在缩并之前简化和优化张量网络的策略。最后，我们将在第 5 节中展示预处理的结果，并在第 6 节中讨论其性能和未来工作。

2 电路模拟与张量网络

2.1 用于电路模拟的张量网络缩并

在典型的（强）模拟场景中，给定某个量子电路 C 和某个计算基矢量 $|x\rangle$，目标是返回振幅 $\langle x|C|0\rangle$。计算最终振幅$\langle x|C|0\rangle$ 需要执行大量广义矩阵乘法（称为“张量点积”）。

张量网络缩并的核心思路是生成一个多重图，其中节点表示张量，边集表示所有需要执行的张量点积运算。在这种表示下，向量 $C|0\rangle$ 可以用一个图高效表示，其结构与 C的结构非常相似。构建好这个张量网络后，可以轻松地向网络中添加度为一的节点，其中包含 $1-$个量子比特线性形式 $\langle 0|$ 或 $\langle 1|$，从而得到一个表示标量 $\langle x|C|0\rangle$的网络。图 1 展示了一个电路及其底层张量网络结构的示例。为简化后续讨论，我们假设 $x = 0^n$。

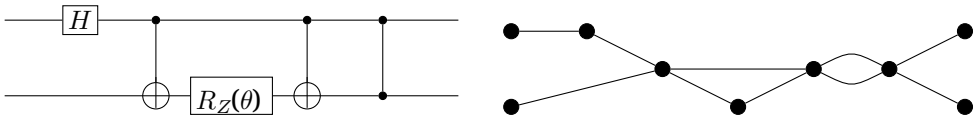


图 1：电路 C 及其对应的张量网络结构，考虑初始乘积态和最终投影算子。

给定这样的张量网络结构，可以通过依次缩并网络中的每条边来计算结果振幅，每次将（多重）边两端的节点替换为一个节点。然而，最终的缩并代价，即缩并网络中每条边所需浮点运算次数的总和，很大程度上取决于边的缩并顺序。

在论文 [8], Markov 和 Shi 中，他们证明，对于任意图 $G = (V,\ E)$，G 的收缩代价等于 G的线图 G^*的树宽。此外，给定宽度为 d 的 G^* 的树分解，存在一种确定性算法，可在多项式时间内输出一个收缩顺序 π，使得 $cc(\pi) \leq d$ （见算法1）。

2.2 提取良好排序的启发式方法

我们的收缩顺序查找器采用 Gray 和 Kourtis[12]所描述的基于社区的方法：使用 Louvain 社区检测算法 [27], 找出张量网络的最佳社区划分，然后运行 QuickBB [26] 为每个独立社区的线图寻找良好的树分解，并使用 Markov 和 Shi 的算法将树分解转换为收缩顺序。由于社区规模相对较小（深度为 20 的 Sycamore 电路中，每个社区有 80 到 110 个节点），寻找最优树分解要容易得多，收缩顺序的质量也有所提升。执行实际收缩时，每个社区都会缩并为一个称为社区顶点的单一节点。我们将按照特定划分得到的这个部分缩并网络称为张量网络的元图 。因此，将各个社区的收缩顺序添加到主收缩顺序后，我们使用相同的方法寻找元图的收缩顺序。最终得到的顺序可以使所有社区几乎无需代价即可完成缩并（按浮点运算次数计），而大部分浮点运算都来自对所得元图的缩并。这就是为什么在指标切片 步骤中——这是一种通过将主要收缩任务分成许多子任务 [9] ，使张量网络收缩能够在内存限制内完成的标准方法——我们重点关注元图中间张量的索引（见算法1）。

算法1 基于社区的排序查找器

输入： 张量网络
输出： 收缩顺序

函数 TreeDecompositionToOrder($\mathcal{T}$) ▷ Markov 和 Shi 方法
  ▷ 输出 $\mathcal{T}$ 的顶点，即线图的顶点：原图中的边
  当 $\mathcal{T}$ 不是空图时 **执行**
    $\mathcal{T}$ 的 $l \leftarrow$ 叶节点
    如果 l 是 $\mathcal{T}$ 的最后一个顶点，**则**
      输出 B_l 中的顶点，顺序任意 ▷ B_l 是 l 中顶点的顶点袋
      从 $\mathcal{T}$ 中移除 l
    else
      $l' \leftarrow$ 的父项是 l
      如果 $B_l \subseteq B_{l'}$，**则**
        从 $\mathcal{T}$ 中移除 l，并重复此过程
      else
        $e \in B_l - B_{l'}$
        输出 e 并将其从 $\mathcal{T}$ 中移除
      结束 if
    结束 if
  结束循环
结束函数 ▷ 步骤数关于 $\mathcal{T}$ 的大小是多项式的

function FindOrder($network$)
  $G \leftarrow$ 来自输入 $network$ 的底层图
  $P \leftarrow$ LouvainCommunityPartition(G)
  $order \leftarrow$ 空列表 ▷ 待收缩的边列表
  对于每个社区 c 在 P **执行**
    $lg_c \leftarrow$LineGraph(c)
    $\mathcal{T}_c \leftarrow$QuickBB(lg_c) ▷ 树分解启发式算法
    $order$.insert(TreeDecompositionToOrder($\mathcal{T}_c$))
  结束每次迭代
  $mg \leftarrow$GraphFromCommunityVertices(P) ▷ 构建元图
  $lg \leftarrow$LineGraph(mg)
  $\mathcal{T} \leftarrow$QuickBB(lg)
  $order$.insert(TreeDecompositionToOrder($\mathcal{T}$))
  $slices \leftarrow$FindSlices(mg, $order$) ▷ 贪心地对最大缩并的索引进行切片
在 $network$ 上执行 $order$ 时发生
  返回 $order$, $slices$
结束函数

预收缩。 在大多数应用中，计算一个相当不错的收缩代价可能成本高昂。我们希望让张量网络操作更便于计算收缩顺序/代价，尤其是估计其线图的树宽。因此，我们引入了一种将张量网络压缩成更小且更密集的网络的方法，该网络封装了原始网络的结构，同时保留其线图的树宽。

定义1 （预收缩）。 设 G 为最大度为 3 的张量网络。对G 进行预收缩，即递归地将所有<escape>*leaves* </escape>（度为 1 的节点）与 G 合并到其邻居，直到 G中不再存在以叶节点为终点的<escape>分支 </escape>（仅剩环）。随后，收缩G 中所有不属于三角形的环边。

在这一预收缩步骤中，只需要进行少量张量积操作。由此可以获得原始张量网络的一个“压缩”版本，从而更容易确定收缩顺序，并进一步降低收缩代价。

请注意，在计算 ZX 图的树宽和其线图($tw(G^*)$)之前，也可以对其使用这种“模拟缩并”，从而获得至少快一个数量级的近似结果。在我们的基准测试中，我们将这种预收缩与 QuickBB 算法[26]结合使用，以近似 $tw(G^*)$（更多详情请参见 A节）。我们将此方法称为 $quick\ tw$。

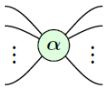
3 使用 ZX 演算简化张量网络

在他们的论文中，Duncan等人 [24] 简要介绍了 ZX演算：ZX演算是一种图示语言，与标准的量子电路符号相似。一个 ZX-图由以下部分组成：

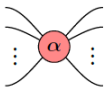
- 导线：所有从图左侧进入的导线都是输入，所有从右侧伸出的导线都是输出。

- 蜘蛛：蜘蛛是线性映射，可以有任意数量的输入或输出导线。它们分为两类：Z蜘蛛以绿色圆点表示，X蜘蛛以红色圆点表示。我们将用黄色方框表示哈达玛边。

用狄拉克符号明确写出时，这些线性映射为：

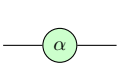
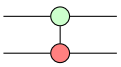
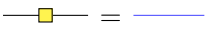


$:= |0...0\rangle\langle 0...0| + e^{i\alpha} |1...1\rangle\langle 1...1|$



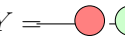
$:= |+...+\rangle\langle +...+| + e^{i\alpha} |-...-\rangle\langle -...-|$

以下是一些将门表示为 ZX 图的示例：

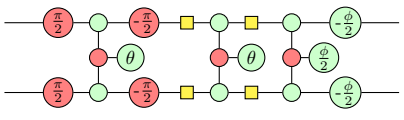
$R_\alpha =$  $CNOT =$  $H =$ 

为了模拟谷歌的 Sycamore 电路，我们需要额外的单量子比特门：

$$\sqrt{X} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}, \sqrt{Y} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \sqrt{W} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -\sqrt{i} \\ \sqrt{-i} & 1 \end{bmatrix}$$

$\sqrt{X} =$  $\sqrt{Y} =$  在 ZX 中表示：

以及由 θ 和 ϕ 参数化的双量子比特门：

$$\mathrm{fSim}(\theta,\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -i\sin(\theta) & 0 \\ 0 & -i\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & e^{-i\phi} \end{bmatrix} =$$ 

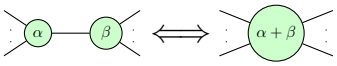
3.1 类图图示

考虑以下在 ZX演算中均成立的重写规则：

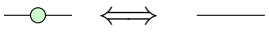
规则 1. 

通过创建哈达玛边，在一种蜘蛛类型与另一种蜘蛛类型之间切换。

规则 2. 两条 Hadamard边相互抵消。

规则 3. 

蜘蛛融合与解融合。

规则 4. 

消除平凡旋转。

使用这组规则，我们可以将任意 ZX 图转换为以下形式。

定义 2 （图状图）。 如果 ZX 图满足以下条件，则称其为类图：

1. 所有蜘蛛都是 Z蜘蛛。2. 所有边都是 Hadamard边。3. 不存在平行的 Hadamard边或自环。4. 每个输入或输出都连接到一个 Z-蜘蛛，并且每个 Z-蜘蛛至多连接一个输入或输出。

引理 3. 每个 ZX 图都可以转换为类图 ZX 图。

证明。 从任意一个 ZX 图开始，依据第一条规则，我们可以将所有红色蜘蛛转换为被 Hadamard 门包围的绿色蜘蛛。然后依据第二条规则移除多余的 Hadamard 门。接着，依据第三条规则，将相邻的蜘蛛融合，从而移除所有非阿达玛边。最后，我们在每一条仍然开放的导线上放置平凡的 Z-蜘蛛（参见 [24] 以获取更深入的演示）。 □

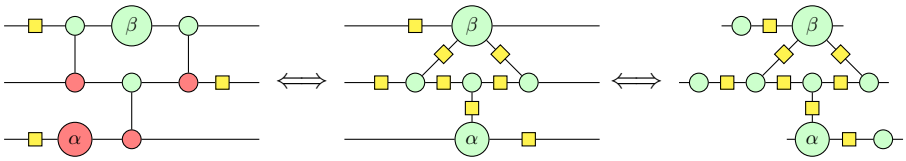


图 2：将 ZX 图重写为图状图。

由于我们关注的是计算形状为 $\langle 0|\, C\, |0\rangle$ 的量，因此所得图中不会有输入或输出。因此，在我们的设置中，定义 2 的性质 4 始终得到满足。我们将这样的类图图示称为<样式 id='2'>闭合。

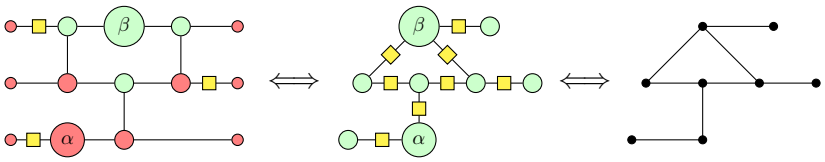


图 3：将闭合 ZX 图重写为闭合图状图。

这个生成的图本质上就是我们的张量网络（我们通常会把它称为图状图，不区分 ZX演算和图论）。现在我们可以开始简化过程：通过一些 ZX 重写来改写图状图，以使其变得稀疏。

3.2 重写类图图示

现在我们介绍将在化简策略中使用的不同重写规则 .

3.2.1 局部补变换和枢轴

局部补变换是由 Kotzig [22] 引入的一种图 minor（见图 4）：

定义 4（局部补变换）。设 G 为一个图，u 为 $G=$ 中的一个顶点（V，E）。根据 u对 G 进行的局部补变换 记作 $G\star u$，所得图与 G具有相同的顶点，但 u的所有邻点 v_1，v_2、..... 在 $G\star u$中相连，当且仅当它们在 G中不相连。其他所有边均保持不变。

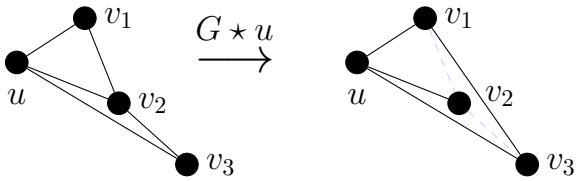


图 4：根据 u对图 G 进行局部补变换。v_1 和 v_3 在 $G\star u$中相连，因为存在边 $v_1v_3\notin E$，而 $v_1v_2\in E$ 和 $v_2v_3\in E$。

事实证明，我们可以对一类特定的类图图示执行这种图变换，这类图示称为图态。图态是所有蜘蛛角均为 0 且每个蜘蛛都有一个输出（即一条悬挂边）的类图图示。给定某个图态 G 及其一个蜘蛛 u，可以执行局部补变换 $G\star u$，并通过以下方式修正图态：

- 向 u的输出添加一个角度为 $-\pi/2$ 的红色蜘蛛
- 将角度为 $\pi/2$ 的绿色蜘蛛添加到 u 的邻居的输出中

图 5 描绘了图态的局部补变换。这是 ZX-演算中的标准定义，详尽证明请参见 [24] 。

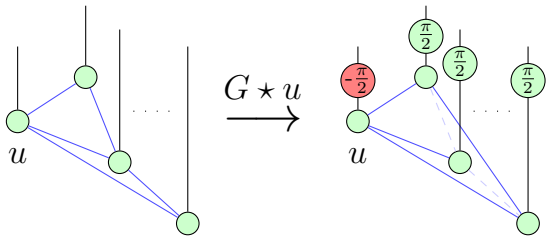


图 5：根据 u 对图状态 G 进行局部补变换的 ZX 图表示。

然而，我们希望对任意闭合图状图（即没有输出且角度非零的图态）应用局部补变换。我们为闭合图状图提出以下局部补变换过程（见图 6）：

- 首先通过拆分规则（规则 3）将角度从蜘蛛中提取出来。 ，
- 对得到的图态应用局部补变换

请注意，所得图形并非类图形式，因为其中包含一个额外的红色蜘蛛和非阿达玛边。还要注意，对类图形式应用多次此类变换，会在携带角度的度为 1 Z的蜘蛛与图态中的蜘蛛之间“生长”出度为 2 的蜘蛛链。不过，除去这些链，所得图形与其图态部分具有相同的结构。

这些链可以看作实现二维简单线性形式（或向量）的张量网络。每次对图状图执行局部补变换时，更新它们的值都很简单：

- 其值最初可以设置为 $\left[1,e^{i\theta}\right]$，其中 θ 是初始蜘蛛的角度

- 然后，对于某个蜘蛛上的每次局部补变换 u:
  - 通过以下方式更新 u 的线性形式 L_u：

$$L_u \leftarrow R_X(-\pi/2)L_u$$

  - 并对每个邻居 $v \in N(u)$，通过以下方式更新其线性形式 L_v：

$$L_v \leftarrow R_Z(\pi/2)L_v$$

由此得到的数据结构在某种程度上是混合的，因为它描述了一个图态，并附带了若干关联到图态中各节点的线性形式。因此，它本身并不是 ZX 图。

请注意，总体而言，这种记录操作比局部补变换本身开销更小：我们只需更新 $|N(u)|+1$ 个常数大小的线性形式，而更新图态则需要修改 $|N(u)|^2$ 个邻接关系。

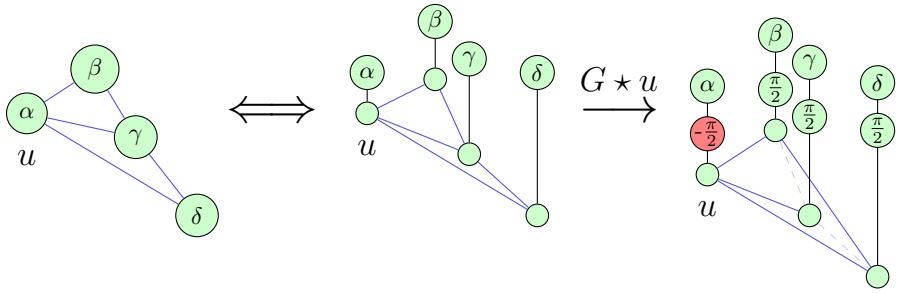
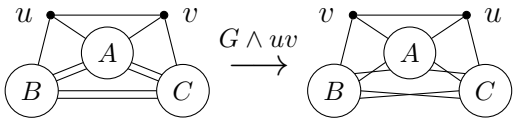


图 6：根据 u，对闭合的类图 G 进行局部补变换的 ZX 图，其中蜘蛛上的角度已被移出。

我们将使用以下基于局部补变换的局部变换：

定义 5（枢轴）。设 G 是一个图，且 u 和 v 是 G 中一对相连的顶点。沿边 uv 对 G 执行枢轴 操作，所得的图为 $G \wedge uv := G \star u \star v \star u$。



其中 $A = N_G(u) \cap N_G(v)$，$B = N_G(u)$ 以及 $C = N_G(v)$。

图 7：沿边 uv 对 G 执行枢轴操作：切换 u 和 v 的共同邻域、u 的独有邻域与 v 的独有邻域之间的边，并交换 u 和 v。

3.2.2 蜘蛛拆分

对于 G 中的顶点 u，在 u 上使用第 3.1 节介绍的解融合规则 会产生一个新节点 u'。然而，由于产生了非阿达玛边，它无法保持 ZX 图的图相似性。

定义 6（图状解融合规则）。设 G 是一个图状图，u 是 G 的一个蜘蛛。将 u 解融合为两个蜘蛛 u, u' 后，在导线 uu' 上插入一个新的平凡 X-蜘蛛 u''，然后将其切换为 Z-蜘蛛，从而生成哈达玛边。

注意，由于我们图状结构中的每个节点都附带一个线性形式，因此我们需要确定拆分后它会变成什么。在图8的示意图中，我们假设线性形式附着在带有角度的节点上（此处为 u）。在实践中，这意味着两个新节点 u'和 u''会获得一个平凡线性形式 $\sqrt{2}\cdot\langle+| = (1\ 1)$，而 u 原本的线性形式仍附着在 u 上。

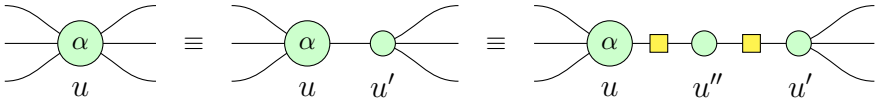


图8：图状解融合规则。

引理 7. 该拆分规则保持 ZX 图的图相似性。

证明。 使用恒等规则移除 u''，然后进行 Hadamard 消去和蜘蛛融合。

□

我们的策略使用图状解融合规则拆分图——即递归地对图中所有高 度节点进行解融合，直到生成的所有蜘蛛的度都小于或等于 3。不过，执行解融合时，我们需要先将 u 的边分成两组：一组仍连接到 u，另一组则连接到 u'。为构造这一划分，我们使用一种内部启发式方法，该方法基于计算围绕 u 的 G 的环基（详情请参阅 A节）。

4 简化策略

4.1 我们的策略

我们的化简策略是在图状图上执行随机枢轴操作，并根据所得图状图的收缩代价为其评分。我们在模拟退火环境中使用了这一策略 [KIR82] ，参数详见第 4.2 小节。当温度降至 0 时，该算法等同于贪心算法。我们在基准测试中测试了这两种运行模式。

因此，我们的方法由以下步骤组成：

- 将输入量子电路和投影算子 $\langle x|$ 转换为 ZX 图，
- 将 ZX 图转换为图状图，
- 使用 ZX 重写规则和枢轴优化类似图的图示，以降低 QuickBB 给出的树宽近似值，然后拆分所有高阶节点，
- 按照第 2.2 小节中的解释，预先将低度节点合并到其邻居中，
- 继续采用最先进的方法来计算收缩顺序，具体方法如第 2.2 小节所述，
- 收缩网络。

4.2 优化参数

我们的收缩顺序查找器使用了 NetworkX [28] 对 Louvain 社区检测算法的实现，以及 Stephan Oepen [29]编写的 QuickBB 算法实现。考虑到社区和元图的节点数与边数较少，我们对它们贪心运行 QuickBB，并假定这足以确保找到的第一个树分解已经是令人满意的近似结果。因此，我们的排序查找器只需几秒钟便能返回收缩顺序及其对应的收缩代价（针对深度为 20 的 Sycamore 电路）。为进一步提升结果质量，我们并行运行多次收缩顺序确定试验，采用不同的社区划分，并仅返回代价最低的收缩顺序。

对于我们的预处理优化策略，我们实现了一种模拟退火算法。如果两个图状图示仅相差一次枢轴操作，我们就认为它们互为邻居。从原始闭合图状图示开始，经过约 100 个步骤后，我们会均匀随机地选取其中一个邻居，并根据全局温度和代价函数，决定是否将其作为下一次迭代的当前起点（见算法 2）。我们尝试了不同的“代理”代价函数来引导模拟退火。我们使用了基于树宽的启发式方法，例如最小填充启发式方法、QuickBB，以及类似于 $quick\ tw$ 的模拟缩并方法，作为

算法 2 类图优化的模拟退火

输入： 图状图、步骤数、代价函数
输出： 优化后的图状图

函数 SimulatedAnnealing($graphlike$, $nb\ steps$, CostFunction)
解$_{current} \leftarrow graphlike$
能量$_{current} \leftarrow$ CostFunction(解$_{current}$)
能量$_{optimal} \leftarrow$ 能量$_{current}$ ▷ 初始化
$step \leftarrow 0$
当 $step < nb\ steps$ **时**
▷ 两个类图若仅相差一个枢轴，则互为邻居
解$_{candidate} \leftarrow$RandomPivot(解$_{current}$)
能量$_{candidate} \leftarrow$ CostFunction(解$_{candidate}$)
如果能量$_{candidate} <$ 能量$_{optimal}$ **那么**
解决方案$_{optimal} \leftarrow$ 解决方案$_{candidate}$
能量$_{optimal} \leftarrow$ 能量$_{candidate}$
结束条件 if
$\tau \leftarrow$Temperature($step/nb_steps$)
$prob \leftarrow$Accept(能量$_{current}$, 能量$_{candidate}$, τ)
如果 $prob >$Random(0,1) **则** ▷ 在 0 和 1 之间均匀地选择一个随机数浮点数
<escape>solution$_{current} \leftarrow$ solution$_{candidate}$</escape>
能量$_{current} \leftarrow$ 能量$_{candidate}$
结束条件 if
结束循环
返回 解$_{optimal}$
▷ CostFunction的定义 , Temperature 以及 A节中的 Accept
结束函数

以及以浮点运算次数（FLOPs）为单位的收缩代价估计。我们发现，将 $quick\ tw$ 与 QuickBB 结合起来估计 $tw(G^*)$，能够获得最佳结果。

我们发现，这种寻找等价张量网络的朴素方法，相较于现有的最先进方法，能够显著降低缩并代价。

5 基准测试与结果

2019 年，Arute 等人 [16] 在他们拥有 53 个量子比特的 Sycamore 量子芯片上，用 200 秒完成了电路采样，并宣称实现了量子霸权；他们最初估计，这项任务需要 Summit——如今全球最强大的超级计算机——耗时约 10,000 年。此后，Huang 等人 [10]使用基于张量网络的模拟引擎 AC-QDP，将该采样任务在经典计算机上的运行时间缩短至约 20 天。

为了将我们的方法与Huang等人的方法 [10], 进行比较，我们针对深度 $\{12, 14, 16, 18, 20\}$ 生成了5个随机Sycamore电路，并使用AC-QDP执行采样任务。随后，我们将相同的电路转换为图状图示，并分别在运行AC-QDP前应用和不应用我们的预处理。对于每种方法的每个深度，我们对5个电路的结果取平均值。图9展示了我们的线图树宽约简策略的效率。标准图显示了AC-QDP将使用的初始张量网络的 $tw(G^*)$值，*ZX*未优化图显示了未使用额外重写规则和局部补策略的我们的方法，而*ZX*优化图则显示了预处理后的最终结果。

我们的预处理和序查找方法证明，在切片之前显著优于当前最先进的方法。图 10 展示了收缩与随机 Sycamore 电路相关联的张量网络所需的浮点运算次数。此处绘制的 AC-QDP 数据直接取自 Huang等人 [10]。模拟退火 曲线展示了我们的化简策略的结果，贪心算法 曲线展示了温度为 0 时模拟退火算法的表现。峰值性能 曲线展示了我们在每种深度下测试的所有电路中取得的最佳结果。

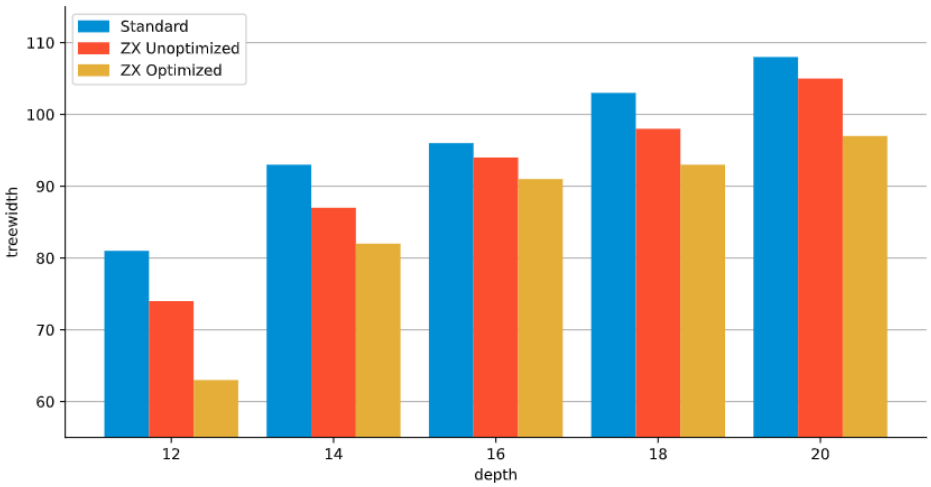


图 9： $tw(G^*)$ 的演变（最小填充启发式算法）。

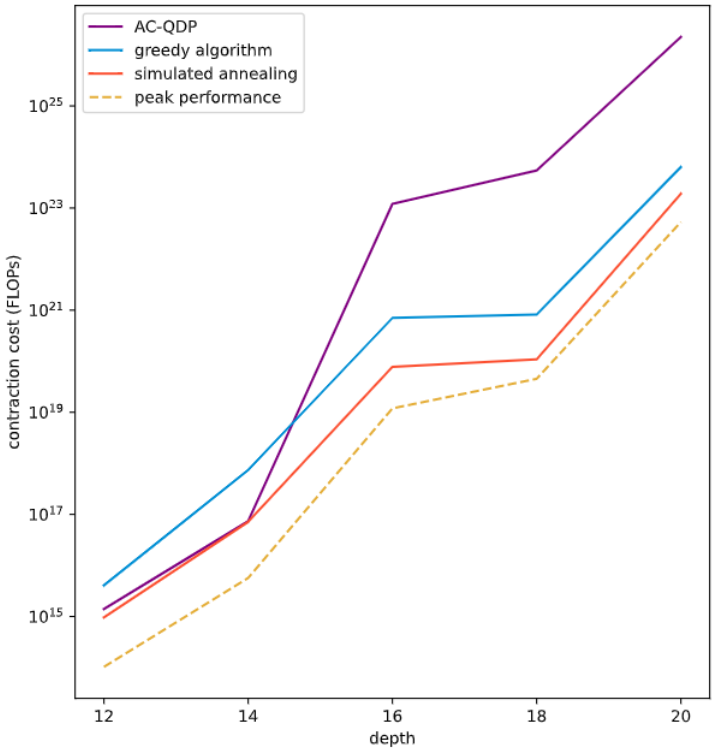


图 10：切片前的缩并代价。*AC-QDP* 数据根据其“单个样本的缩并代价和浮点运算次数效率”外推，再乘以其“# 子任务”数。

深度	12	14	16	18	20
AC-QDP	1.40×10^{15}	7.33×10^{16}	1.21×10^{23}	5.43×10^{23}	2.23×10^{26}
我们的方法（平均值）	9.61×10^{14}	7.15×10^{16}	7.75×10^{19}	1.09×10^{20}	1.89×10^{23}
Gain	1.45×	1.03×	1561.3×	5122.6×	1179.9×
我们的方法（峰值）	1.03×10^{14}	5.68×10^{15}	1.19×10^{19}	4.52×10^{19}	5.32×10^{22}
Gain	13.5×	12.9×	10168.1×	12013.3×	4200.6×

表 1：随机Sycamore电路在切片步骤之前进行缩并所需的平均和峰值收缩代价或浮点运算次数，增益 = AC-QDP / 我们的方法。

通过并行运行我们的排序查找器来提高效率，我们能够在几分钟内获得具有竞争力的结果：对于最大规模的深度为 20 的 Sycamore 电路，我们发现，与 AC-QDP 的方法相比，我们的方法平均可将浮点运算次数减少约 1180 倍，从而有望将原本需要 数年 的计算缩短至 数天。

6 讨论

现在，我们展示将优化后的张量网络与当前最先进方法进行实际缩并的结果。所有计算均使用相同的硬件和相同的随机Sycamore电路完成。为确保公平比较，展示 AC-QDP 性能的图表是使用我们自己的 GPU 计算得出的。

为了衡量我们方法的效率，我们在硬件（NVIDIA Tesla V100S）上运行 AC-QDP 之前，先将基于 ZX 的优化算法应用于随机 Sycamore 电路，作为预处理步骤。我们未能复现其深度为 20 的 Sycamore 模拟结果，因为我们抽样的任务在 24 小时内未能完成。

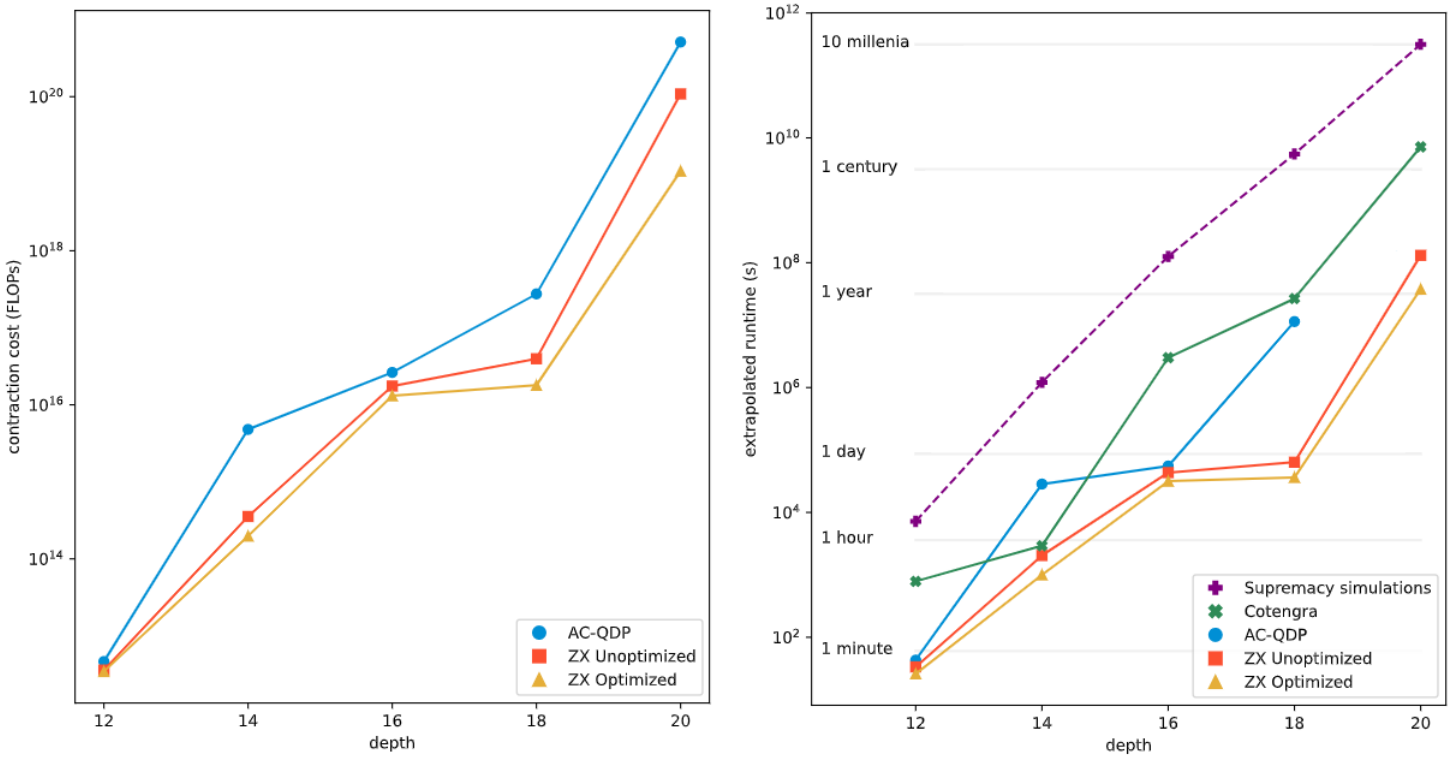


图 11：模拟随机Sycamore电路的收缩代价和外推运行时间。*Supremacy* 模拟 数据来自 [30], *Cotengra* 数据来自 [12], *AC-QDP*、*ZX Unopti- mized*、*ZX*优化 在我们的硬件上进行模拟。

深度	12	14	16	18	20
收缩代价	3.448×10^{12}	1.976×10^{14}	1.309×10^{16}	1.791×10^{16}	1.079×10^{19}
# 子任务	2^5	2^{10}	2^{16}	2^{16}	2^{23}
外推运行时间	$25.98s$	$998.71s$	$8.79h$	$10.09h$	$435.72d$

表 2：使用我们的预处理方法后，在运行 AC-QDP 之前，对 Sycamore 上随机电路采样模拟的外推基准测试结果。

与以往方法相比，我们的张量网络所需的切片少得多：对于最大的电路，在类图 ZX 图上仅用几分钟确定收缩顺序后，我们使用的子任务数量最多比 AC-QDP 少 32 倍。

这项工作展示了如何利用形式化方法简化张量网络。所得结果令人鼓舞，并且还有进一步改进的空间。具体而言，可以找到更好的“代理”

代价函数来指导基于枢轴的启发式算法，而更高效（可能并行化）的树分解启发式算法，则可通过 Markov 和 Shi 方法直接得到更好的收缩顺序。在理论方面，还需要进一步研究，为所有可通过局部补变换得到的图状图的线图树宽定义一个上界。事实上，我们可以考虑在与 G 通过局部补变换等价的图 G' 集合上，取 tw（G'^*）的最小值，从而进一步改进 Markov 和 Shi 推导出的上界。尽管求得这一全局最小值可能很困难，但它有助于进一步收紧张量网络收缩代价的复杂度上界。

致谢。 我们感谢 Atos Quantum Lab 的同事给予的支持和富有启发性的讨论，感谢 Vivien Vandaele 在 ZX演算方面提供的帮助，也感谢 Oc´eane Koska 和 Maxime Oliva 对本文早期版本的审阅。本文是 HQI 倡议（www.hqi.fr）的一部分，并获得法国国家研究署项目资助，项目编号为“ANR- 22-PNCQ-0002”。

参考文献

1. Guerreschi, G. G., Hogaboam, J., Baruffa, F. & Sawaya, N. P. 英特尔量子模拟器：一款可用于云端的高性能量子电路模拟器。《量子科学与技术》**5**, 034007 (2020). 2. Suzuki, Y. 等Qulacs：一款面向研究的快速、通用量子电路模拟器。《*Quantum*》**5**, 559 (2021). 3. Vidal, G. 对轻度纠缠量子计算进行高效经典模拟。《物理评论快报》**91**, 147902 (2003). 4. Schollw¨ock, U. 矩阵乘积态时代的密度矩阵重整化群。《物理学年鉴》**326**, 96–192 (2011). 5. Aaronson, S. & Gottesman, D. 稳定子电路的改进模拟。《物理评论*A*辑》**70**, 052328 (2004). 6. Bravyi, S. 等通过低秩稳定子分解模拟量子电路。《*Quantum*》**3**, 181. ISSN: 2521-327X (2019年9月). 7. Jozsa, R. & Miyake, A. 匹配门与量子电路的经典模拟。《英国皇家学会会刊*A*辑：数学、物理与工程科学》**464**,3089–3106 (2008). 8. Markov, I. L. & Shi, Y. 通过缩并张量网络模拟量子计算。《工业与应用数学学会计算杂志》**38**, 963–981 (2008). 9. Chen, J., Zhang, F., Huang, C., Newman, M. & Shi, Y. 中等规模量子电路的经典模拟。*arXiv*预印本 *arXiv:1805.01450* (2018). 10. Huang, C. 等量子霸权电路的经典模拟。*arXiv*预印本 *arXiv:2005.0678*(2020). 11. Lykov, D., Schutski, R., Galda, A., Vinokur, V. & Alexeev, Y. 采用随步骤变化的并行化策略的张量网络量子模拟器。*arXiv*预印本 *arXiv:2012.02430* (2020). 12. Gray, J. & Kourtis, S. 超优化张量网络收缩。《*Quantum*》**5**, 410 (2021). 13. Dudek, J. M., Duenas-Osorio, L. & Vardi, M. Y. 通过图分解高效缩并用于加权模型计数的大型张量网络。*arXiv* 预印本 *arXiv:1908.04381*(2019). 14. Liang, L. 等快速搜索张量网络的最优收缩序列。《*IEEE*信号处理精选主题期刊》**15**, 574–586 (2021). 15. Vincent, T. 等Jet：基于并行任务的张量网络收缩实现快速量子电路模拟。《*Quantum*》**6**, 709 (2022). 16. Arute, F. 等使用可编程超导处理器实现量子霸权。《自然》**574**, 505–510 (2019). 17. Biamonte, J. D., Morton, J. & Turner, J. 用于#SAT的张量网络缩并。《统计物理学杂志》**160**, 1389–1404 (2015).

18. O'Gorman,B. 张量网络收缩的参数化。*arXiv*预印本 *arXiv:1906.000*(2019). 19. Coecke, B. & Duncan, R. 量子可观测量的图形演算。预印本(2007). 20. Vilmart, R. 纯量子比特量子力学 *ZX*演算的近最优公理化2018. arXiv: 1812.09114 [quant-ph]. 21. Jeandel, E., Perdrix, S. & Vilmart, R. ZX演算的完备性。《计算机科学逻辑方法》**16**, 11–1 (2020). 22. Kotzig, A. 有限四价图中的欧拉线及其变换。《图论》，219–230 (1968). 23. Perdrix.量子计算的形式模型：资源、抽象机与基于测量的计算博士论文（格勒诺布尔国立综合理工学院（INPG），2006），111. 24. Duncan, R., Kissinger, A., Perdrix, S. & Van De Wetering, J. 用ZX演算对量子电路进行图论化简。《*Quantum*》**4**, 279 (2020). 25. Dumitrescu, E. F. 等将树宽作为张量网络模拟的实用组成部分进行基准测试。《*PLOS ONE*》**13**, e0207827 (2018). 26. Gogate, V. & Dechter, R. 一种求解树宽的完备随时算法。*arXiv*预印本 *arXiv:1207.4109* (2012). 27. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. 大型网络中的社群快速展开。《统计力学杂志：理论与实验》**2008**, P10008 (2008). 28. Hagberg, A. A., Schult, D. A. & Swart, P. J. 使用 *NetworkX*探索网络结构、动态特性与功能载于第七届*Python*科学会议论文集（编者：Varoquaux, G., Vaught, T. & Millman, J.）（美国加利福尼亚州帕萨迪纳，2008），11–15. 29. Oepen, S. 等 *mtool*2019. https://github.com/cfmrp/mtool. 30. Arute, F.,Arya, K., Babbush, R., 等《使用可编程超导处理器实现量子霸权》的补充信息，《自然：国际周刊科学杂志》**574**, 505–505 (2020).

A 补充定义

用于类图优化的模拟退火函数

函数 TEMPERATURE(prog) ▷ 模拟退火的温度函数
▷ prog $= step/nb\ steps$，模拟退火的进度
▷ 返回经过归一化（从 1 到 0）的递减指数函数进度
返回 $\dfrac{e^{-\text{prog}} - 1/e}{1 - 1/e}$
结束函数

函数 ACCEPT(成本, 新成本, $_\tau$) ▷ 接受新解的概率
▷ 成本，新成本 $=$ 能量$_{current}$，能量$_{candidate}$
如果新的成本 <成本 **那么**
返回 1 ▷ 贪心地接受更优解
else
返回 $e^{-log(log(\text{new cost})-log(\text{cost})+1)/\tau}$
结束 if
结束函数

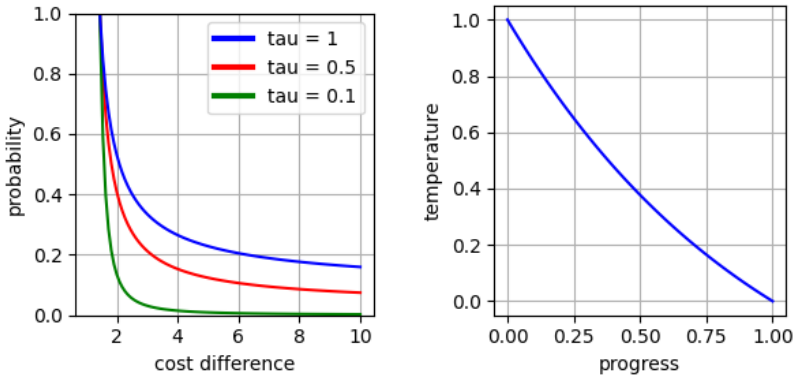


图 12: ACCEPT 和 TEMPERATURE 模拟退火函数

图状解融合规则的实现

函数 MatchingUnfusion（G, $node$） ▷ 使用环基算法找出最佳的节点对，以便在解融合高 degree 节点时予以保留

$cycles_{node}$ = 包含 $node$ 的 CycleBasis(G) 中的环集合
H =新图
N = $node$的邻居集合
适用于 u 在 N **执行**
for $v \neq u$ 在 N **do**
in_common = 中同时包含 u 和 v的 $cycles_{node}$ 循环集合
向 H 添加权重为 length($in\ common$) 的加权边（u, v）
直到结束
结束循环
返回 MaxWeightMatching(H)
结束函数

函数 Unfusion($graph-like$, $node$) ▷ 将度数较高（大于 4）的节点转换为类似三叉树的结构，同时保留图相似性

G = 输入 $graph-like$的底层图
$matching$ = MatchingUnfusion(G,$node$)
当 $matching$ 不为空时 **执行**
$edge(u, v)$ = $matching$.pop()
从 G 中移除边 ($node$, u)
从 G 中移除边 ($node$, v)

向 G 添加一个名称全新的节点 w ▷ 解融合后的节点
将边（u, w）添加到 G
将边（v, w）添加到 G

将具有新名称 m 的节点添加到 G ▷ 已添加以保留图相似性
将边（w, m）添加到 G
向 G 添加边（m, $node$）
结束 while 循环
结束函数

函数 Splitting(图相似) ▷ 将图相似结构的度降至 3，并保持图相似性

G = 来自输入 $graph-like$的底层图
$high_degree$ = 中度数大于 3 的节点集合
当 $high\ degree$ 不为空时 **执行**
$node$ = $high\ degree$.pop()
Unfusion(类图，$node$)
重新计算 $high\ degree$
结束 while 循环
结束函数

预收缩方法和 quick tw 启发式

函数 PRECONTRACT($network$)

▷ 将网络压缩成更小、密度更高的网络，同时保留其结构以及原始网络线图的树宽

$G =$ 来自输入 $network$ 的底层图

▷ 首先递归地将所有叶节点合并到其邻居

$leaves =$ G 中所有度为 1 的节点集合

当 $leaves$ 不为空时 **执行**

$leaf = leaves$.pop()

$parent =$ $leaf$ 的父节点

将 $leaf$ 收缩为 $parent$（在 G 中）

如果 $parent$ 在 G 中的度为 1 **则**

将 $parent$添加到 $leaves$

结束 if

结束 while

▷ 然后收缩非三角形边

$candidates =$ 所有两端均为度为 2 的节点的边集合

当 $candidates$ 不为空时 **执行**

$edge = candidates$.pop() ▷ $edge = (u,\ v)$

将 v 收缩为 u，位于 G中

对于 u 的每个邻居 w **执行**

如果边（$v,\ w$）在 $candidates$ **中，则** ▷ 更新边的名称

从 $candidates$ 中移除边 $(v,\ w)$

将边 (u, w) 添加到 $candidates$

结束 if

直到结束

结束 while

结束函数

函数 QUICK TW($network$) ▷ 用于模拟退火的代价函数

$G =$ 来自输入 $network$ 的底层图

PRECONTRACT(G)

$lg =$ G 的线图

返回 QUICKBB(lg)

结束函数