高效的张量化神经网络和张量网络算法的部分范数初始化

Alejandro Mata Ali*
i3B Ibermatica Fundazioa, Quantum Development Department, Paseo Mikeletegi 5, 20009 Donostia, Spain

Iñigo Perez Delgado† and Marina Ristol Roura‡
i3B Ibermatica Fundazioa, Parque Tecnológico de Bizkaia, Ibaizabal Bidea, Edif. 501-A, 48160 Derio, Spain

Aitor Moreno Fdez. de Leceta§
i3B Ibermatica Fundazioa, Unidad de Inteligencia Artificial, Avenida de los Huetos, Edificio Azucarera, 01010 Vitoria, Spain

(10Dated: 2025 年 7 月 10 日)

我们提出了两种算法，用于通过部分计算张量化神经网络和一般张量网络的 Frobenius 范数及线性逐元素范数来初始化层，具体取决于所涉及的张量网络类型。该方法的核心是在迭代过程中使用张量子网络的范数，从而使我们能够通过导致发散或零范数的有限值进行归一化。此外，该方法还从中间计算结果的复用中获益。我们也将其应用于矩阵乘积态/张量列（MPS/TT）和矩阵乘积算子/张量列车矩阵（MPO/TT-M）层，并观察到了其相对于节点数量、键维数及物理维度的扩展性。所有代码均已公开。

Keywords: 机器学习，张量网络，量子启发的其他方法

arxiv:2309.06577v4 中译本

I. 介绍

深度神经网络 [1] 在机器学习中被广泛用于在工业、研究及各种其他应用中取得良好效果。这种良好的性能使其得以应用于更复杂的场景，需要更多的参数。因此，各种架构已被利用以提升其性能。最大的例子是大型语言模型 (LLM) [2]，它们使用了极大量的参数，需要大型设备才能运行。内存需求成为了未来应用和人工智能当前发展路线可扩展性的一大限制。

随着量子计算被应用于各个领域，由于量子系统处理信息的指数级容量，对量子信息压缩方法 [3] 的兴趣增加了。这是导致 量子机器学习领域发展的一个原因。然而，当前量子硬件的限制使得无法实现大多数量子计算提供的所需算法和模型。

在这种情况下，研究人员探讨了可以利用量子系统特性的不同经典技术，即量子启发式。这使得一些量子计算的优势可以在不需要量子设备来实现操作的情况下得到应用。最著名的量子启发技术之一是张量网络 [4, 5]，它们是对张量代数计算的图形表示。张量网络具有通过诸如矩阵乘积态/张量列车（MPS/TT）[6] 或投影纠缠对态（PEPS）[7] 等表示方式来 “压缩” 张量信息的强大能力。这使得我们可以用较少的参数保留所表示张量的重要信息。这种压缩已被以各种方式应用于几种机器学习模型，例如将矩阵分解为张量网络 [8, 9]。它已被应用于神经网络 [10], 卷积神经网络 [11], 变压器 [12], 脉冲神经网络 [13] 或 LLM [14, 15]。张量网络也被用作主要模型，直接训练张量网络本身 [16, 17]（图 1）。

我们的分析重点将放在生成大型张量网络以建模层并直接训练的方法上，而不是使用已训练的稠密矩阵并对其进行压缩。例如，当我们尝试使用张量化的物理信息神经网络 [18] 或张量神经网络来解决大规模工业案例中的微分方程 [19]，如发动机的热方程或涡轮机中的流体问题。在这种情况下，经常会遇到初始化问题 [20]，表示的张量值会爆炸或消失。如果我们用某种分布来初始化张量网络中每个张量的元素，在将张量网络收缩以获取它所表示的张量时，其中一些最终元素对于计算机来说过大（无穷大）或过小（零）。这个问题也可能出现在某些量子机器学习算法中，其

* alejandro.mata.ali@gmail.com
† iperezde@ayesa.com
‡ mristol@ayesa.com
§ aitormoreno@lksnext.com

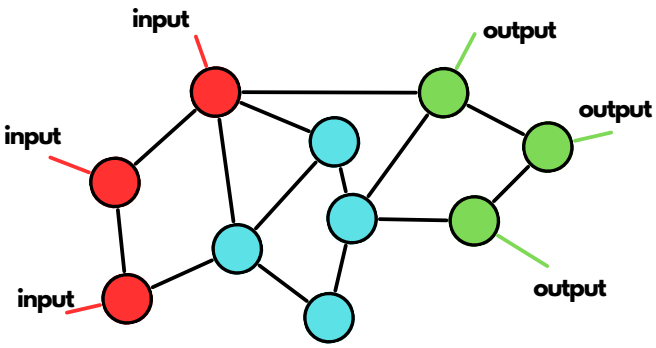

图 1. 任意张量网络层。

目标是压缩经典机器学习模型。

如果我们想消除这些问题，一个初步的建议可能是收缩张量网络并消除或减少这些元素。然而，在非常大的层中我们无法将所有最终的张量元素存储在内存中，因此用这种方式是不可能的。此外，我们可能不知道如何正确地重新调整那些发散的元素而不损害其他元素。一种方法是通过改变具有更好超参数的分布、改变均值和标准差或重新调整它们来重新初始化张量网络 [20]。然而，这些方法中的许多在所有情况下都难以高效地应用。

在这项工作中，我们提出了两种新的算法来解决在两种不同场景下的问题。第一种适用于一般值的张量网络，它包括迭代计算张量网络不同部分的弗罗贝尼乌斯范数直到满足某个条件，在此过程中我们将张量网络的所有参数除以用特定方式计算出的一个因子。这使我们能够逐步让层的 Frobenius 范数趋向于我们想要的数值，而无需反复重新初始化。我们称这种方法为 *Frobenius* 张量网络重整化（FTNR）。第二种方法适用于所有正值的张量网络，在此过程中与前一种方法类似但涉及计算简化形式的部分线性逐元素范数。我们称这种方法为线性张量网络重正化（LTNR）。

这些算法对于分层树形结构的层级特别有趣，尤其是在张量列车 (TT)、张量列矩阵 (TT-M) 和投影纠缠对态 (PEPS) 中。这也可以应用于其他张量网络方法中，例如组合优化，以确定超参数，并可以与其他初始化方法结合使用。

本工作的主要贡献是两种有效的算法，用于初始化张量化的人工智能模型或其他张量网络算法，而不会导致参数爆炸或消失的问题，特别是在大规模层中。

本文的结构如下。首先，在第 II 节我们将准确描述我们想要解决的主要问题，并介绍现有的处理该问题的算法的一些背景知识。然后，在第 III 和 IV 节，我们将分别介绍我们的新算法在一般情况和正态情况下的应用。此外，在第 V 节我们将进行若干实验以调查这些算法的局限性。最后，在第 VI 节我们将分析这些算法的其他可能应用场景。

我们创建了一个 Python 函数，可以在任意 PyTorch 层上运行，该函数在 GitHub 仓库的 Jupyter Notebook 中可用 https://github.com/DOKOS-TAYOS/高效初始化张量网络

II. 问题描述与背景

当我们有一个由 N 个节点组成的张量网络时，我们会发现表示该张量网络的张量元素是由一组值的和给出的，每个值都是不同节点中 N 个元素的乘积。如果我们考察如图 2.a 所示的 TT 层的情况，该层的元素被赋予为

$$T_{ijklm} = \sum_{nopq} T^0_{in} T^1_{njo} T^2_{ojp} T^3_{pjq} T^4_{qm}. \tag{1}$$

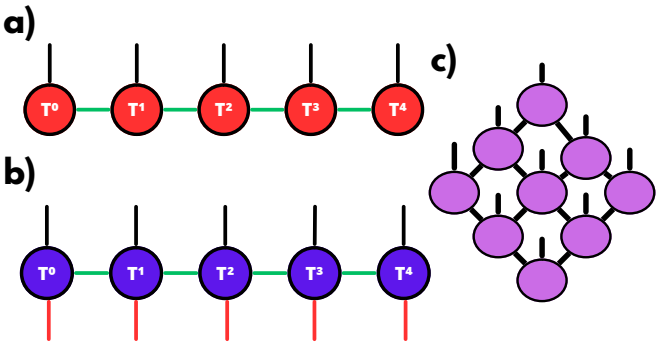

图 2. a) 具有 5 个索引的张量列层。b) 具有 10 个索引的张量列车矩阵层。c) 具有 9 个索引的 PEPS 层。

我们看到在张量中有 5 个索引时，需要乘以 5 个张量元素，但在一般情况下有 N 个索引时，则需

$$T_{i_0 i_1 \dots i_{N-1}} = \sum_{j_0 j_1 \cdots j_{N-2}} T^0_{i_0 j_0} T^1_{j_0 i_1 j_1} \dots T^{N-1}_{j_{N-2} i_{N-1}}, \tag{2}$$

乘以 N 个张量元素 T^i 以获得张量 T 的元素。

为了举例说明我们想要解决的问题，让我们思考一下这个问题。对于一般情况下，当结合维数为 b 时，在每个节点对之间收缩的索引维度，N 个节点，以及具有值 a 的节点常量元素，我们会有以下表示张量元素的结果：

$$T_{i_0 i_1 \ldots i_{N-1}} = b^{N-1} a^N. \tag{3}$$

我们可以看到，当有 $N = 20$ 个节点、元素值为 $a = 1.5$ 且结合维数为 $b = 10$ 时，最终的张量元素将会是 $3.3\ 10^{22}$。这是一个非常好的初始化的大元素。这是我们所说的张量值的 “爆炸”。另一方面，如果该元素的值是 $a = 0.01$，则张量元素将会是 10^{-20}。这是张量值的 “消失”。然而，如果我们把这些张量的值除以那个数，我们可能会得到一个对于我们的计算机来说太大或太小无法存储的 a^N，结果我们会得到所有重新归一化的元素都是 0 或无穷大。

这个问题因层中的节点数量而加剧，因为每个节点都是一个乘积。此外，我们不能简单地为具有许多物理索引的情况计算这些张量元素，即输出索引。这是因为需要保存在内存中的值的数量随着物理索引的数量呈指数级增长。还有可能的是，我们用不同的分布而不是类似的分布初始化张量，使得网络中的一些节点的元素比其他节点的小。然而，这种方法可能导致模型训练的问题。

张量网络架构通常在表达能力和可训练性之间存在权衡，这强烈受到张量核心初始值的影响。对于传统的密集网络，随机初始化 [21] 就足够了，利用了 中心极限定理。然而，由于张量收缩的多重线性和键维度的作用，TN 模型需要更复杂的策略，正如我们在上一节中所指出的。

第一种方法是在 [22] 中发展的，他们为概率模型的 TT 初始化做好准备，作为 “热启动”。该技术能够获得避免因果问题的模型。然而，它使用了 TT-交叉近似 [23]，因此不易于一般化。第二种方法是从 Haar 测度分布生成酉矩阵 [24]，因为它们保持范数不变。然而，对于一般结构而言，仅靠酉性是不够的，例如 PEPS。第三种方法是将一些单位矩阵与一个小的随机噪声项连接起来 [25]。然而，作者指出这是一种特定用例的技术。

III. 一般张量网络初始化协议 FTNR

我们首先处理一般实值张量网络的情况，在这种情况下，我们将每个节点初始化为类似的分布，使节点的元素相对于彼此处于相同的尺度。我们的协议基于使用部分弗罗贝尼乌斯范数来规范化所得张量的总弗罗贝尼乌斯范数。

矩阵的弗罗贝尼乌斯范数由方程

$$||A||_F = \sqrt{\sum_{ij} |a_{ij}|^2} = \sqrt{\mathrm{Tr}(A^\dagger A)}. \tag{4}$$

给出在张量网络中，这意味着将一层与其自身的副本进行缩并，使得每个物理指标都与它对应的副本相连。我们可以在图 3 中看到一些示例。

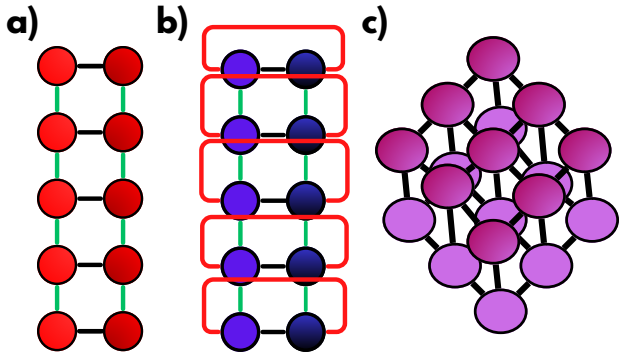

图 3. 弗罗贝尼乌斯范数的平方计算至：a) 张量列层。b) 张量列矩阵层。c) PEPS 层。

该张量网络的收缩等价于其表示矩阵的弗罗贝尼乌斯范数。此外，可以仅使用节点的元素进行计算，而无需计算所表示矩阵的元素。

Frobenius 范数是一种用于正则化模型 [26] 层的指标，并给出了矩阵元素量级的估计。在元素围绕 a_{00} 平滑分布的情况下，公式 (4) 中的范数对于矩阵 $n \times m$ 将是 $\sqrt{nm}\ |a_{00}|$ 的数量级。

为了避免层中的元素过大或过小，从而在初始化时输出也过大或过小，我们将归一化这些元素，使张量的 Frobenius 范数是我们选择的一个数值，例如 $||A||_F = 1$。这防止了我们的最大值超过这个数值，同时利用值的局部分布来确保最小值不会太小。

尽管如此，对于矩阵 $n \times m$，我们将求和 nm 个值，因此我们应该调整范数使其与问题的规模 nm 成比例，从而不减少这些值的数量级。

为此，我们定义了所谓的张量网络的部分平方范数。

A. 张量网络的偏平方范数

对于本文的其余部分，我们假设可以一致地对张量网络的节点进行排序，使它们形成一个单一的增长网络。

定义 1（部分平方范数）
给定一个由 N 个节点组成的张量网络 $\mathcal{A}$，以及一个由 $\mathcal{A}$ 的前 n 个节点定义的张量网络 $\mathcal{A}_n$，在 $\mathcal{A}$ 的 n 个节点处的部分平方范数 ${}^{pF}||\mathcal{A}||_{n,N}$ 定义为 $\mathcal{A}_n$ 的 *Frobenius* 范数的平方。那就是，

$$ {}^{pF}||\mathcal{A}||_{n,N} = ||\mathcal{A}_n||_F^2. \tag{5}$$

为了了解这个部分平方范数是什么，我们用一个简单的例子来说明，即张量链层。我们考虑图 4 中的张量网络，其节点是有序的。

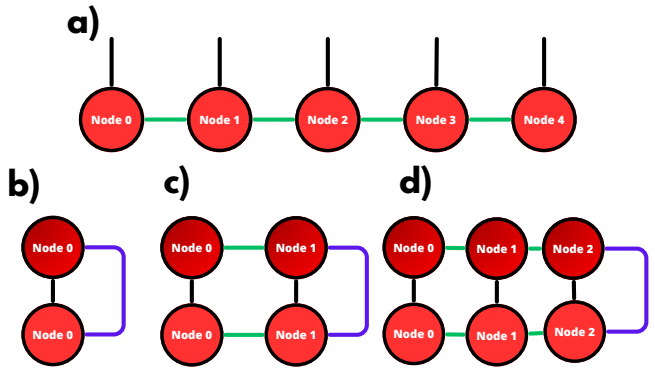

图 4. a) 具有 5 个节点的张量列层。b) 在 1 个节点处的部分平方范数。c) 在 2 个节点处的部分平方范数。d) 在 3 个节点处的部分平方范数。

如我们所见，在这种情况下，我们只需要进行计算总张量网络的总范数时相同的流程，但在第 n 步停止，并收缩链两端最终两个张量的绑定指标。

我们可以在以下图中看到，5 和 6，TT-矩阵层和 PEPS 层的部分平方范数会是如何。

此计算可以很容易地扩展到一般的张量网络，只要我们有一个一致的节点排序。

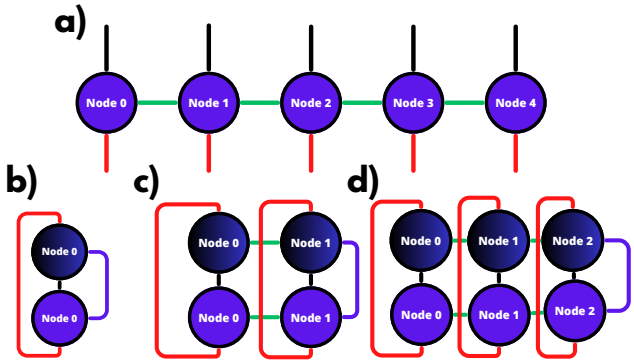

图 5. a) 具有 5 个节点的张量列矩阵层。b) 在 1 个节点处的部分平方范数。c) 在 2 个节点处的部分平方范数。d) 在 3 个节点处的部分平方范数。

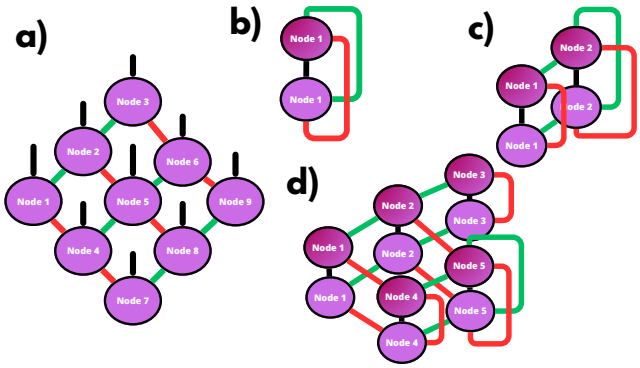

图 6. a) 包含 9 个节点的 PEPS 层。b) 在 1 个节点处的部分平方范数。c) 在 2 个节点处的部分平方范数。d) 在 5 个节点处的部分平方范数。

B. Frobenius 张量网络重正化初始化协议

如果我们有一个张量网络 $\mathcal{A}$，表示一个 $n_A \times m_A$ 矩阵，其 Frobenius 范数 $||\mathcal{A}||_F$ 无穷大（大于我们的计算机可以存储的范围）、零（小于我们计算机的精度），或者超出某个值域时，我们将希望对张量网络中的元素进行归一化处理，使得新张量网络 $\mathcal{B}$ 的范数 $||\mathcal{B}||_F$ 等于给定的一个数值 F。我们假设由这个张量网络表示的张量在其元素值上具有平滑分布，由于其张量网络节点的元素也具有平滑分布 [20]。我们还假设 $F = n_A m_A$，但我们看到另一个数值可以很容易地被选择。为了归一化具有 N 个节点的 $\mathcal{A}$ 张量的范数，我们只需将每个节点的元素除以 $||\mathcal{A}||_F^{1/N}$。

由于我们不能将元素除以 0 或无穷大，因此我们使用以下逻辑。如果总范数是无穷大（零），则存在一个

$n < N$ 的值使得 ${}^{pF}||\mathcal{A}||_{n,N}$ 是有限且非零的，从而使得 ${}^{pF}||\mathcal{A}||_{n+1,N}$ 是无穷大（零）。这是因为每一步，我们将一个新的节点添加到部分平方范数中，并乘以一个新的值，因此在一定数量的节点之后会出现无穷大（零），而具有少一个节点的部分平方范数是一个有效的可除数值。

想法是逐步规范化范数，以便我们最终实现完全规范化。为了避免多次从头开始迭代过程，我们可以利用这样一个事实：如果我们规范化了 $\mathcal{A}_n$，构成它的所有子网络 $\mathcal{A}_m, \forall m < n$ 也将被规范化。因此，每次我们进行规范化时，不必从只有一个节点的网络开始，而是可以继续使用有 n 个节点的网络。

对于某些张量网络，如 MPS，可以高效地重用中间计算。在每个归一化步骤中，当我们计算部分范数时，首先收缩连接子网节点的所有键指数和所有物理指数。在收缩未出现的节点之间的键索引之前，我们将张量暂存。因此，在重正化之后，我们可以对单个张量元素以及这个暂存张量进行归一化，从而不必再次收缩整个张量网络。此外，我们可以在归一化的下一步中使用这个张量。

我们想要一个张量网络 $\mathcal{A}$，其弗罗贝尼乌斯范数为 F，包含 N 个节点，并且设定容差范围为 (aF, bF)，其中 $a \leq 1$,$b \geq 1$。弗罗贝尼乌斯张量网络重正化初始化协议如下：

1. 我们使用某种初始化方法来初始化节点张量。我们推荐使用均值既不太高也不太低且为正数，标准差为常数（不大于 0.5）的高斯分布进行随机初始化。

2. 我们计算 $||\mathcal{A}||_F$。如果它是有限且非零的，我们将每个节点中的每个元素除以 $\left(\frac{||\mathcal{A}||_F}{F}\right)^{1/N}$ 并返回 $\mathcal{A}$。否则，我们继续。

3. 我们计算 ${}^{pF}||\mathcal{A}||_{1,N}$。

   (a) 如果是无穷大，我们将节点 $\mathcal{A}$ 的每个元素除以 $(10(1+\xi))^{1/2N}$，其中 ξ 是一个介于 0 和 1 之间的随机数，并返回第 2 步。

   (b) 如果是零，我们将 $\mathcal{A}$ 的每个节点元素乘以 $(10(1+\xi))^{1/2N}$ 并返回到第 2 步。

   (c) 否则，我们将此值保存为 ${}^{pF}||\mathcal{A}||_{1,N}$ 并继续。

4. 对于 $n \in [2, N-1]$，我们计算 ${}^{pF}||\mathcal{A}||_{n,N}$。

   (a) 如果是无穷大或零，我们将 $\mathcal{A}$ 的每个节点元素除以 $({}^{pF}||\mathcal{A}||_{n-1,N})^{\frac{1}{2N}}$，并重复步骤 2 和 4（从这个 n 的值开始）。

   (b) 如果是有限的，但大于 b 或小于 a，我们将 $\mathcal{A}$ 的每个节点元素除以 $\left(\frac{{}^{pF}||\mathcal{A}||_{n,N}}{F}\right)^{\frac{1}{2N}}$，并重复步骤 2 和步骤 4（从 n 的这个值开始）。

   (c) 否则，我们继续。

5. 如果没有任何部分平方范数超出范围、无穷大或零，我们将 $\mathcal{A}$ 节点的每个元素除以 $\left(\frac{{}^{pF}||\mathcal{A}||_{N-1,N}}{F}\right)^{\frac{1}{2N}}$，并重复步骤 2 和 5。

我们重复该循环直到获得一个有效的 $\mathcal{A}$ 或达到停止条件，这涉及重复最大次数的迭代。如果我们达到了这一点，协议将失败，我们将有两个选择。第一个是改变节点的顺序以检查其他结构。第二个是在初始化协议中使用其他超参数重新初始化。

在部分范数因一个节点发散而使用随机因子的原因是，由于不知道应该除以的真实值，我们按数量级重新缩放。然而，为了避免可能的无限重缩放循环，我们增加了一个可变性因素，以便不会陷入停滞状态。

如果张量网络中某些张量的元素值比其他张量的小或大，则有可能不均匀地在 N 个节点上分配带有指数 $\frac{1}{2N}$ 的重正化，而是允许其他节点以不同的方式进行重正化，使这些值的比例相等。我们还可以尝试通过调整每个张量的重正化来尽可能多地保留整数值，或者执行量化。这两个过程可以在完成重正化过程之前、期间或之后进行，建议至少在最后进行。

IV. 正张量网络初始化协议 LTNR

本例涉及所有元素均为正的张量网络，这将使我们能够应用计算成本较低的技术。我们的协议基于使用部分线性逐元范数来规范化结果张量的总线性逐元范数。

定义 2 (线性逐元素范数)
矩阵 A 的线性逐分量范数，其元素为 a_{ij}，由

$$||A||_L = \sum_{ij} a_{ij} = \mathbb{1}^T A \mathbb{1}, \tag{6}$$

给出，其中 $\mathbb{1} = (1, 1, \dots)$ 是一个全一向量。

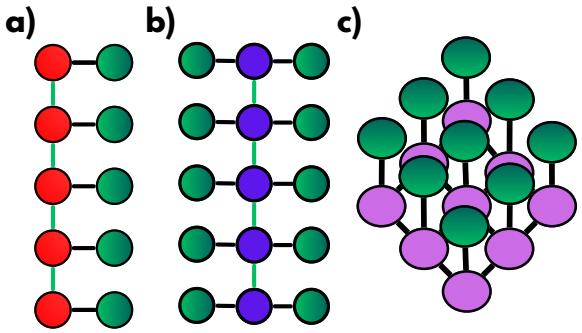

图 7. 线性逐元素范数计算到：a) 张量列层（红色节点）。b) 张量列车矩阵层（蓝色节点）。c) PEPS 层（紫色节点）。绿色节点是一向量。绿色节点是一向量。

在张量网络中，这将意味着用一组全一向量的节点来收缩这一层。我们可以在图 7 中看到一些示例。这个张量网络的收缩相当于求它所表示矩阵的所有元素之和。就像弗罗贝尼乌斯范数的情况一样，它可以不用计算所表示矩阵的元素而直接通过仅使用节点的元素来计算。

在这种情况下，我们可以将该规则解释为类似于一般情况下的 Frobenius 范数，因为所有元素都是正的，我们将有

$$||A||_L = \sum_{ij} a_{ij} = \sum_{ij} |a_{ij}| = \sum_{ij} |\sqrt{a_{ij}}|^2 = ||A^{\circ\frac{1}{2}}||_F. \tag{7}$$

因此，对于正值矩阵，线性逐元范数等于矩阵 Hadamard 根的 Frobenius 范数。当元素在 a_{00} 周围分布平稳时，公式 (6) 中的范数将为 nma_{00} 阶，对于一个 $n \times m$ 矩阵。

如同一般情况，避免值过大或过小的方法是对矩阵进行归一化，但这次是相对于线性逐元素范数而言。对于矩阵 $n \times m$，我们将求和 nm 个值，因此应该调整范数使其与问题的大小 nm 成比例，从而不因之减小这些值的数量级。

为此，我们定义了我们将称之为张量网络的部分线性范数，类似于 Frobenius 范数方法的部分平方范数。

A. 张量网络的部分线性范数

定义 3 (部分线性范数)
给定一个由 N 个节点组成的张量网络 $\mathcal{A}$，以及由 $\mathcal{A}$ 的前 n 个节点定义的张量网络 $\mathcal{A}_n$，我们将 ${}^{pL}||\mathcal{A}||_{n,N}$ 定义为 $\mathcal{A}$ 中 n 个节点的部分线性范数，即 $\mathcal{A}_n$ 的线性逐元素范数。那就是，

$${}^{pL}||\mathcal{A}||_{n,N} = ||\mathcal{A}_n||_L. \tag{8}$$

为了了解这种部分线性范数是什么，我们将用一个简单的例子来说明，即张量链层。我们将考虑图 8 中的张量网络，其节点是有序的。

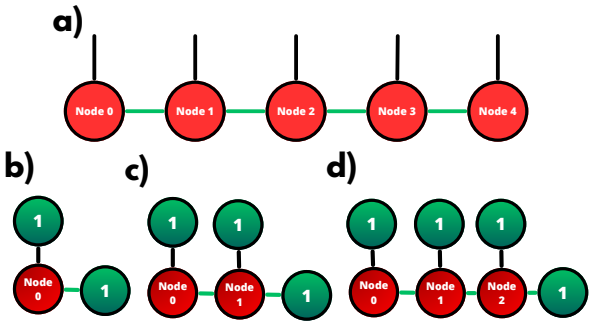

图 8. a) 具有 5 个节点的张量列层。b) 在 1 个节点处的部分线性范数。c) 在 2 个节点处的部分线性范数。d) 在 3 个节点处的部分线性范数。1 个节点是单位向量。

如我们所见，在这种情况下，我们只需要执行与计算总张量网络的总范数相同的步骤，但在第 n 步停止，并将链中最后一个张量的边索引与全 1 向量收缩。

我们可以在图 9 和 10 中看到 TT-Matrix 层和 PEPS 层的部分线性范数是如何形成的。这种计算可以很容易地扩展到一般的张量网络，只要我们有一个一致的节点排序。

B. 线性张量网络重正化初始化协议

如果我们有一个张量网络 $\mathcal{A}$，表示一个矩阵 $n_A \times m_A$，其线性逐元素范数 $||\mathcal{A}||_L$ 是无穷大、零或超出某个值范围时，我们将希望规范化我们的张量网络的元素，使得新的张量网络 $\mathcal{B}$ 的范数 $||\mathcal{B}||_L$ 等于一个特定数值。我们假设这个张量网络所表示的张量的元素具有平滑分布，因为它的张量网络节点 [20] 的元素也具

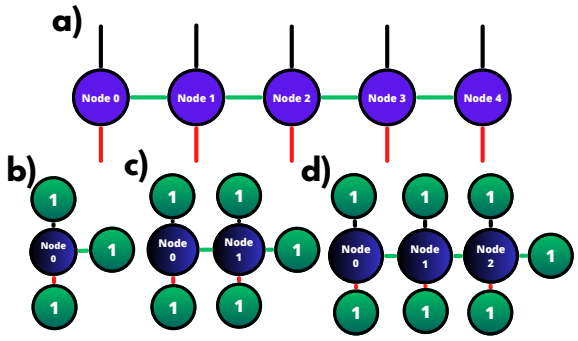

图 9. a) 具有 5 个节点的张量列车矩阵层。b) 在 1 个节点处的部分线性范数。c) 在 2 个节点处的部分线性范数。d) 在 3 个节点处的部分线性范数。1 个节点是单位向量。

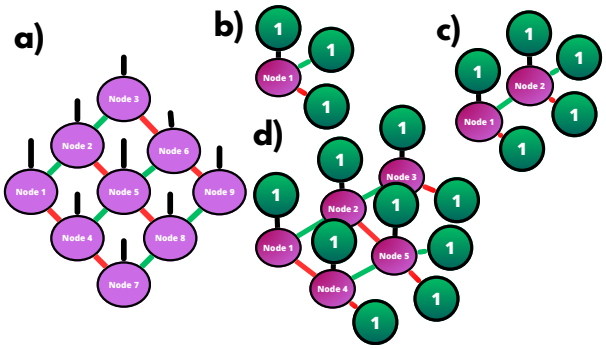

图 10. a) 包含 9 个节点的 PEPS 层。b) 在 1 个节点处的部分线性范数。c) 在 2 个节点处的部分线性范数。d) 在 5 个节点处的部分线性范数。这 1 个节点是单位向量。

有平滑分布。我们还将假设 $F = n_A m_A$，但我们会看到另一个数字可以很容易地被选择。为了归一化具有 N 个节点的 $\mathcal{A}$ 张量的范数，我们将只需要将其每个节点的元素除以 $||\mathcal{A}||_L^{1/N}$。

与前一个算法相同的理由导致了类似的算法，但将部分平方范数更改为部分线性范数。想法是逐步规范化范数，以便最终实现完全规范化。如同一般情况，在每次规范化之后，无需从单个节点重新计算，可以从当前子网络继续。

我们希望得到一个张量网络 $\mathcal{A}$，其具有线性逐元素范数 F，包含 N 个节点，并设定容差范围为 (aF, bF)，其中包含 $a \leq 1, b \geq 1$。线性张量网络重正化初始化协议如下：

1. 我们使用某种初始化方法来初始化节点张量。我们推荐使用具有常数标准差（不大于 0.5）和一个既不太高也不太低且为正的常数值均值的高斯分布进行随机初始化。

2. 我们计算 $||\mathcal{A}||_L$。如果它是有限且非零的，我们将每个节点中的每个元素除以 $\left(\frac{||\mathcal{A}||_L}{F}\right)^{1/N}$ 并返回 $\mathcal{A}$。否则，我们继续。

3. 我们计算 ${}^{pL}||\mathcal{A}||_{1,N}$。

   (a) 如果是无限的，我们将 $\mathcal{A}$ 节点中的每个元素除以 $(10(1+\xi))^{1/N}$，其中 ξ 是一个介于 0 和 1 之间的随机数，并返回步骤 2。

   (b) 如果是零，我们将 $\mathcal{A}$ 的每个节点元素乘以 $(10(1+\xi))^{1/N}$ 并返回步骤 2。

   (c) 否则，我们将此值保存为 ${}^{pL}||\mathcal{A}||_{1,N}$ 并继续。

4. 对于 $n \in [2, N-1]$，我们计算 ${}^{pL}||\mathcal{A}||_{n,N}$。

   (a) 如果是无穷大或零，我们将 $\mathcal{A}$ 的每个节点元素除以 $({}^{pL}||\mathcal{A}||_{n-1,N})^{\frac{1}{N}}$，并重复步骤 2 和 4（从 n 的这个值开始）。

   (b) 如果是有限的，但大于 b 或小于 a，我们将 $\mathcal{A}$ 的每个节点元素除以 $\left(\frac{{}^{pL}||\mathcal{A}||_{n,N}}{F}\right)^{\frac{1}{N}}$，并重复步骤 2 和步骤 4（从此值的 n 开始）。

   (c) 否则，我们继续。

5. 如果没有部分线性范数超出范围、无穷大或零，我们将 $\mathcal{A}$ 节点的每个元素除以 $\left(\frac{{}^{pL}||\mathcal{A}||_{N-1,N}}{F}\right)^{\frac{1}{N}}$，并重复步骤 2 和 5。

如同一般情况，我们重复该循环直到达到停止条件，即达到了最大迭代次数。如果我们到达这一点，协议将失败，我们将有与之前相同的两个选项。

V. 实验

在本节中，我们将对这两种算法进行若干实验。我们将检查归一化张量网络所需的步骤数量的扩展性，每次因为部分范数中发现无穷大或零而需要部分归一时算作一步。我们测试了具有 N 个节点、均匀物理维度 p 和键维数 b 的 TT 层和 TT 矩阵层。我们使用 1 作为均值的值，0.5 作为标准差的值。我们选择 $F = p^N$。对于 TT 层，这是我们拥有的元素数量，而对于 TT 矩阵层，则是其平方根，一个我们为了收敛目的所采用的数量。我们的容差范围是 $(10^{-3}F, 10^3F)$。

我们首先测试了使用 Frobenius 范数的归一化性能。首先，我们检查步数与张量网络节点数量 N 的关系，从 2 到 34，对于图 11 中不同值的 p 和 $b = 12$。然后，我们检查步骤数量是否与图 12 中相同值的 $N = 25$ 和 $b = 10$ 对应的 p 进行对比。最后，我们将步骤数量与图 13 中具有相同值的 b、$N = 25$ 和 $p = 15$ 进行对比。

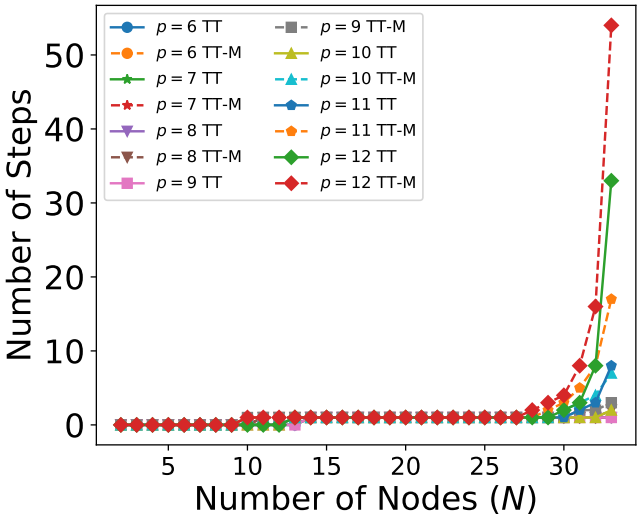

图 11. 步骤数与 N 的关系，对于 TT 层和 TT-矩阵层的 Frobenius 方法中固定 $b = 10$ 时，p 从 6 到 12 的变化。

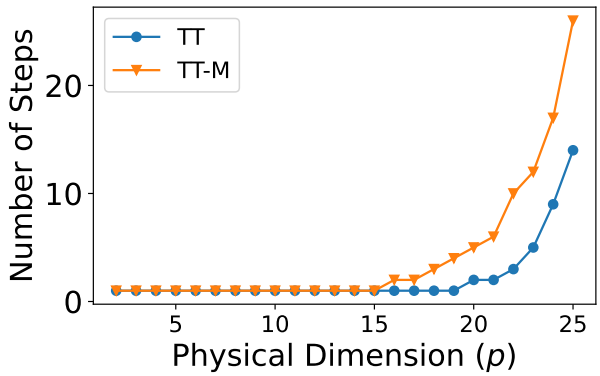

图 12. 步骤数与 p 的关系，TT 层和 TT-矩阵层的 Frobenius 方法中固定了 $N = 25$ 和 $b = 10$。

我们可以观察到，在所有情况下，对于 $N < 10$ 不需要任何步骤，而在 $N = 27$ 的区域内只需要一步。对于较大的节点数量，步数随 N 呈指数级增加，且对于更大的 p 增长得更快。相对于 p，我们看到对于 $p < 15$ 只需一步，而对于较大的值，则有另一个指数级的增长。相对于 b，没有观察到显著的依赖性。算法应该需要更多的步骤来处理更大的情况，但我们没有足够的内存来进行计算。对于所有的检查，TT-矩阵都需要较多的步数。

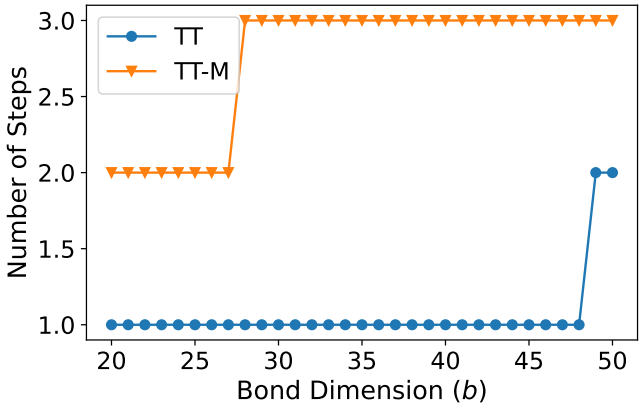

图 13. 步骤数与 b 的关系，TT 层和 TT-矩阵层中固定 $N = 25$ 和 $p = 15$ 的 Frobenius 方法。

现在，我们测试线性算法，配置与 Frobenius 实验相同。在图 14 中，我们可以观察到类似的趋势，如同 Frobenius 情况。所有 $N < 13$ 实例无需步骤，并且对于 $N < 30$ 仅需一步即可归一化矩阵。在接下来的区域，需要指数数量的步骤，但不像 Frobenius 情况那么多。只有 $p = 12$TT-矩阵实例需要更多的步骤。在图 15 中，我们可以观察到与 Frobenius 案例相同的行为，但所需的步骤较少。最后，在图 16 中，步数和 b 之间没有明显的依赖关系，可能是由于之前提到的相同原因。

如我们所见，两种算法在各种可能的实例规模中都表现出色。此外，线性算法表现更好，这可能是由于张量的逐元素范数平滑缩放所致，该范数呈线性缩放，而 Frobenius 范数在平方根内部呈二次方缩放。

VI. 其他应用

到目前为止，我们已经看到了张量化神经网络的应用，但这种方法可以用于更多领域。每个需要收缩张量网络的算法或应用，且构成它的张量的非零元素数量级相同的情况下，我们都可以从中受益。这在我们不关心最终张量元素的绝对规模，而是想要观察它们之间的相对尺度时是有帮助的。

一个示例是模拟虚时间演化过程，我们希望看到

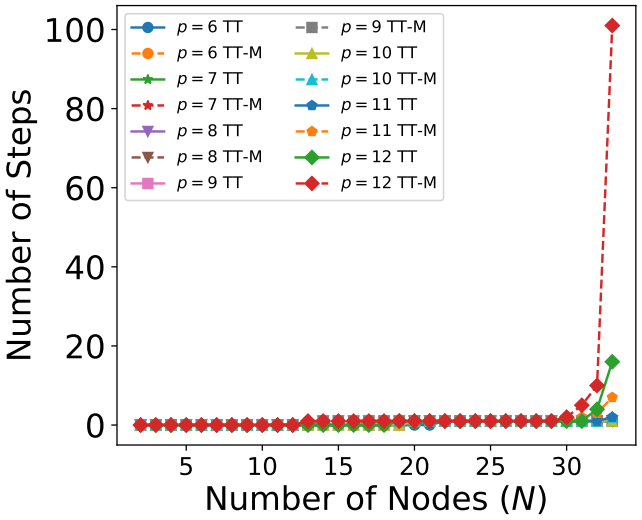

图 14. 步骤数与 N 的关系，对于线性方法中固定 $b = 10$ 的 TT 层和 TT-矩阵层，p 从 6 到 12。

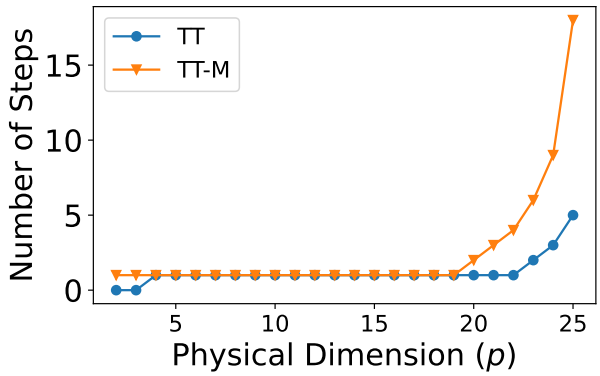

图 15. 步骤数量与 p 的关系，TT 层和 TT-矩阵层的线性方法中固定了 $N = 25$ 和 $b = 10$。

的是能量最小的状态，而不是它所具有的能量。然而，如果我们执行该方法，保存乘以张量网络元素的比例因子，并将结果张量网络的值乘以此因子，则可以恢复此能量。这可能很有趣，因为不同的因子可以按其数量级分开相乘，从而避免溢出。

VII. 结论

我们开发了两种方法，利用张量化神经网络的部分 Frobenius 范数和线性逐元素范数计算成功初始化这些层，具体取决于涉及的张量网络类型。我们揭示了

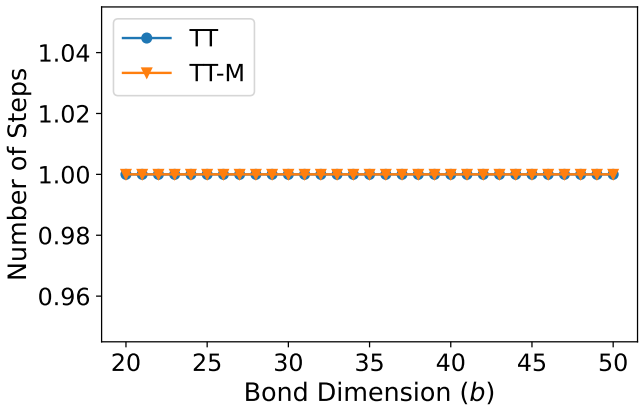

图 16. 步骤数量与 b 的关系，TT 层和 TT-矩阵层的线性方法中固定了 $N = 25$ 和 $p = 15$。

一种方式，可以利用各种类型的张量网络中的中间计算来优化这一过程。我们也将其应用于不同的层，并观察了其相对于节点数量、键维和物理维度的比例变化。

这些方法的主要限制是需要张量元素的值遵循平滑分布，这在某些情况下可能会有所局限。为了扩展这类算法的应用范围，必须克服这一限制，从而使得新的张量网络算法成为可能。

一个可能的未来研究方向可能是探讨如何减少需要执行的步骤数量。另一个可以是研究复杂性随着每种不同类型现有层大小增加的比例关系。我们也可以将其应用于第 VI 节中提到的方法，例如在组合优化 [27] 中确定适当的衰减因子，并将其适应到量子机器学习层。

致谢

本论文的研究工作得到了 Q4Real 项目（面向真实行业的量子计算）的资助，该项目属于 HAZITEK 2022 计划，编号为 ZE-2022/00033。

[1] M. H. M. Noor and A. O. Ige, A survey on deep learning and state-of-the-art applications (2024), arXiv:2403.17561.

[2] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, Llama: Open and efficient foundation language models (2023), arXiv:2302.13971 [cs.CL].

[3] V. Belis, P. Odagiu, M. Grossi, F. Reiter, G. Dissertori, and S. Vallecorsa, Guided quantum compression for high dimensional data classification, Machine Learning: Science and Technology **5**, 035010 (2024).

[4] J. Biamonte and V. Bergholm, Tensor networks in a nutshell (2017), arXiv:1708.00006.

[5] R. Orús, A practical introduction to tensor networks: Matrix product states and projected entangled pair states, Annals of Physics **349**, 117 – 158 (2014).

[6] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, Matrix product state representations (2007), arXiv:quant-ph/0608197 [quant-ph].

[7] V. M. F. Verstraete and J. Cirac, Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems, Advances in Physics **57**, 143 (2008), https://doi.org/10.1080/14789940801912366.

[8] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, Tensorizing neural networks (2015), arXiv:1509.06569.

[9] Z.-F. Gao, S. Cheng, R.-Q. He, Z. Y. Xie, H.-H. Zhao, Z.-Y. Lu, and T. Xiang, Compressing deep neural networks by matrix product operators, Phys. Rev. Res. **2**, 023300 (2020).

[10] Y. Qing, K. Li, P.-F. Zhou, and S.-J. Ran, Compressing neural networks using tensor networks with exponentially fewer variational parameters, Intelligent Computing **4**, 10.34133/icomputing.0123 (2025).

[11] S. Singh, S. S. Jahromi, and R. Orus, Tensor network compressibility of convolutional models (2024), arXiv:2403.14379 [cs.CV].

[12] H. Li, J. Zhao, H. Huo, S. Fang, J. Chen, L. Yao, and Y. Hua, T3srs: Tensor train transformer for compressing sequential recommender systems, Expert Systems with Applications **238**, 122260 (2024).

[13] D. Lee, R. Yin, Y. Kim, A. Moitra, Y. Li, and P. Panda, Tt-snn: Tensor train decomposition for efficient spiking neural network training (2024), arXiv:2401.08001 [cs.NE].

[14] B. Aizpurua, S. S. Jahromi, S. Singh, and R. Orus, Quantum large language models via tensor network disentanglers (2024), arXiv:2410.17397 [quant-ph].

[15] A. Tomut, S. S. Jahromi, A. Sarkar, U. Kurt, S. Singh, F. Ishtiaq, C. Muñoz, P. S. Bajaj, A. Elborady, G. del Bimbo, M. Alizadeh, D. Montero, P. Martin-Ramiro, M. Ibrahim, O. T. Alaoui, J. Malcolm, S. Mugel, and R. Orus, Compactifai: Extreme compression of large language models using quantum-inspired tensor networks (2024), arXiv:2401.14109 [cs.CL].

[16] J. Qi, C.-H. H. Yang, P.-Y. Chen, and J. Tejedor, Exploiting low-rank tensor-train deep neural networks based on riemannian gradient descent with illustrations of speech processing (2022), arXiv:2203.06031.

[17] P. Blagoveschensky and A. H. Phan, Deep convolutional tensor network (2020), arXiv:2005.14506.

[18] S. K. Vemuri, T. Büchner, J. Niebling, and J. Denzler, Functional tensor decompositions for physics-informed neural networks (2024), arXiv:2408.13101 [cs.LG].

[19] R. Patel, C.-W. Hsing, S. Sahin, S. S. Jahromi, S. Palmer, S. Sharma, C. Michel, V. Porte, M. Abid, S. Aubert, P. Castellani, C.-G. Lee, S. Mugel, and R. Orus, Quantum-inspired tensor neural networks for partial differential equations (2022), arXiv:2208.02235.

[20] F. Barratt, J. Dborin, and L. Wright, Improvements to gradient descent methods for quantum tensor network machine learning (2022), arXiv:2203.03366.

[21] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, Vol. 9, edited by Y. W. Teh and M. Titterington (PMLR, Chia Laguna Resort, Sardinia, Italy, 2010) pp. 249–256.

[22] X. Tang, Y. Khoo, and L. Ying, Initialization and training of matrix product state probabilistic models (2025), arXiv:2505.06419 [math.NA].

[23] I. Oseledets and E. Tyrtyshnikov, Tt-cross approximation for multidimensional arrays, Linear Algebra and its Applications **432**, 70 (2010).

[24] F. Mezzadri, How to generate random matrices from the classical compact groups (2007), arXiv:math-ph/0609050 [math-ph].

[25] E. Puljak, S. Sanchez-Ramirez, S. Masot-Llima, J. Vallès-Muns, A. Garcia-Saez, and M. Pierini, tn4ml: Tensor network training and customization for machine learning (2025), arXiv:2502.13090 [cs.LG].

[26] J. Wang, C. Roberts, G. Vidal, and S. Leichenauer, Anomaly detection with tensor networks (2020), arXiv:2006.02516.

[27] T. Hao, X. Huang, C. Jia, and C. Peng, A quantum-inspired tensor network algorithm for constrained combinatorial optimization problems, Frontiers in Physics **10**, 10.3389/fphy.2022.906590 (2022).