arXiv:2304.01600v1 [cs.DS] 4 Apr 2023

CONGEST 模型中的最小费用流

Tijn de Vos*

摘要

我们研究具有 n 个节点、 m 条边、直径为 D，且整数费用和容量均有 poly n 上界的网络上的 CONGEST 模型。本文展示了如何在 $n^{1/2+o(1)}(\sqrt{n}+D)$ 轮次内求出最小费用流问题的精确解，改进了目前最先进的算法；该算法的运行时间为 $m^{3/7+o(1)}(\sqrt{n}D^{1/4}+D)$ [FGL+21]，但仅适用于单位容量图这一特殊情形。对于某些图，我们还能取得更好的结果。具体而言，对于平面图、扩展图、 $n^{o(1)}$-亏格图、$n^{o(1)}$-树宽图和排除小图图类，我们的算法只需 $n^{1/2+o(1)}D$ 轮次。我们通过结合近期关于 CONGEST 模型中拉普拉斯求解器的研究成果 [FGL+21、ALH+22] 与 Lee 和 Sidford 的线性规划求解器 [LS14]，的 CONGEST 实现，得到了这一结果；最后，我们证明可以将近似解舍入为精确解。我们的算法还能在 $n^{1/2+o(1)}(\sqrt{n}+D)\log^3(1/\epsilon)$ 轮次内，以加性误差 ϵ 求解某些推广最小费用流问题的线性规划。

1 引言

CONGEST 模型 [Pel00] 是研究最为广泛的分布式模型之一。它由一个包含 n 个节点的网络组成，这些节点通过同步轮次进行通信，其中每个节点都可以与其每个邻居交换大小为 $O(\log n)$ 的消息。最小费用流问题被认为是 CONGEST 模型中较难的问题之一。尽管目前最高的下界是 $\tilde{\Omega}(\sqrt{n}+D)$，这与最短路径、最小生成树、二分性、 s-t 连通性 [PR00, Elk06, SHK+12]等“更容易”的问题相同，但直到最近才提出了第一个分布式算法 [FGL+21]。对于近似版本，还存在一些更进一步且同样较新的结果 [GKK+18,BFKL21]。这些结果利用强大的拉普拉斯范式 来获得其结果。

拉普拉斯范式涵盖了一系列结合数值技术和组合技术的算法。加权图的 拉普拉斯矩阵 定义为 G $L(G) := \mathrm{Deg}(G) - A(G)$，其中 $\mathrm{Deg}(G)$ 是对角加权度矩阵：$\mathrm{Deg}(G)_{uu} := \sum_{(u,v)\in E} w(u,v)$，而 $\mathrm{Deg}(G)_{uv} := 0$ 对于 $u \neq v$，且 $A(G)$ 是邻接矩阵： $A(G)_{uv} := w(u,v)$。这一研究方向由 Spielman 和 Teng [ST04]，发起，他们证明了图的拉普拉斯矩阵中的线性方程可以在近线性时间内求解。此后，出现了更高效的顺序和并行拉普拉斯求解器，包括 [KOSZ13, KMP14, KMP11,CKM+14, KS16, PS14, KLP+16]。拉普拉斯范式取得了许多成功，其中包括但不限于流问题 [M10, She13, KLOS14, M16, Pen16, CMSV17, LS20a, LS20b, AMV20]，二分图匹配 [van den Brand et al. 2020]，以及（并行）最短路径 [Li20, ASZ20]。

最近，这些进展也已经进入分布式领域 [GKK+18、BFKL21、FGL+21、ALH+22、FdV22]。特别是，Forster、Goranci、Liu、Peng、Sun 和 Ye [FGL+21] 提供了一个拉普拉斯求解器，其运行需要 $n^{o(1)}(\sqrt{n}+D)$ 轮次，且接近最优：他们给出了一个 $\tilde{\Omega}(\sqrt{n}+D)$ 下界。此外，他们证明了其拉普拉斯求解器可导出（最小费用）最大流算法 [M16、CMSV17] 在 CONGEST 中的实现

*奥地利萨尔茨堡大学。本研究由奥地利科学基金（FWF）资助：P 32863-N。本项目获得欧洲研究理事会（ERC）通过欧盟地平线2020研究与创新计划（资助协议编号：947702）提供的资助。

模型。在本文中，我们显著降低了求解这些流问题精确变体的算法的轮复杂度。

1.1 我们的结果

我们的主要结果是一种能够解决最小费用流问题的算法，因此也能解决最大流问题。

定理 1.1. 存在一种算法，对于具有整数费用 $G = (V, E, w)$ 和容量 $q \in \mathbb{Z}_{\geq 0}^m$ $c \in \mathbb{Z}_{\geq 0}^m$ 且满足 $\|q\|_\infty, \|c\|_\infty \leq M$ 的有向图，能够在 *CONGEST* 模型中用 $\tilde{O}(\sqrt{n} T_{\text{Laplacian}}(G) \log^3 M)$ 轮次计算最小费用最大 s-t 流，其中 $T_{\text{Laplacian}}(G)$ 是在 G 上求解拉普拉斯系统所需的轮次数。

我们知道，对于一般图 [FGL+21]， $T_{\text{Laplacian}}(G) = n^{o(1)}(\sqrt{n} + D)$，其结果接近最优值。然而，对于某些图，我们可以得到更好的结果。这基于普遍最优值 算法的概念，该概念将输入图的拓扑结构纳入考量。相关细节可见第 1.2 节。特别地，对于平面图、扩展图、 $n^{o(1)}$-亏格图、 $n^{o(1)}$-树宽图以及排除小图，我们有 $T_{\text{Laplacian}}(G) = n^{o(1)} D$ 。

此外，我们指出，Cohen、Mądry、Sankowski 和 Vladu [CMSV17] 证明了，负权单源最短路径问题可以归约为最小费用流问题和非负权最短路径计算。对于后者，使用 [CM22] 在 $\tilde{O}(\sqrt{n} D^{1/4} + D)$ 轮次内完成计算，我们得到以下推论。

推论 1.2。 存在一种算法，对于给定的带整数权重 $w \in \mathbb{Z}^m$ 、满足 $\|w\|_\infty \leq M$ 的有向图 $G = (V, E, w)$ 以及源点 $s \in V$，能够在 *CONGEST* 模型中经过 s 从 $\tilde{O}(\sqrt{n} T_{\text{Laplacian}}(G) \log^3 M)$ 轮次计算从 s 出发的最短路径，其中 $T_{\text{Laplacian}}(G)$ 是在 G 上求解拉普拉斯系统所需的轮次数。

我们通过将该问题表述为一个线性规划、以高精度求解该线性规划并对结果进行舍入，得到定理 1.1。为此，我们针对 CONGEST 模型中的某些线性规划提出了一个线性规划求解器。

形式化地说，问题设置如下。令 $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$, $c \in \mathbb{R}^m$, $l_i \in \mathbb{R} \cup \{-\infty\}$，并且对所有 $i \in [m]$，有 $u_i \in \mathbb{R} \cup \{+\infty\}$，其中我们假设 $l_i \neq -\infty$ 或 $u_i \neq +\infty$。我们要解决的线性规划如下

$$\text{OPT} := \min_{\substack{x \in \mathbb{R}^m : A^T x = b \\ \forall i \in [m] : l_i \leq x_i \leq u_i}} c^T x.$$

我们假设线性规划 $\Omega^o := \{x \in \mathbb{R}^m : A^T x = b, \ l_i \leq x_i \leq u_i\}$ 的可行解集合非空。

定理 *1.3.* 设 $A \in R^{m \times n}$ 是一个秩为 $rank(A) = n$ 的约束矩阵，设 $b \in R^n$ 是需求向量，且 $c \in R^m$ 是成本向量。此外，设 $x_0 \in \Omega^o$ 是给定的初始点。假设一个 *CONGEST* 网络由 n 个节点组成，其中每个节点 i 都知道 A 中每个满足 $A_{ji} \neq 0$ 的完整第 j 行，并且在 $A_{ji} \neq 0$ 时知道 $(x_0)_j$ 。此外，假设对于每个 $y \in R^n$ 和正对角矩阵 $W \in R^{m \times m}$，我们都能在 $T_{\text{Laplacian}}(G)$ 轮次内将 $(A^T W A)^{-1} y$ 计算到精度 $poly(1/m)$。令 $U := max\{\|1/(u - x_0)\|_\infty, \|1/(x_0 - l)\|_\infty, \|u - l\|_\infty, \|c\|_\infty\}$。那么，高概率下，*CONGEST* 算法 *LPSolve* 会在 $\tilde{O}(\sqrt{n} log^3(U/\epsilon) T_{\text{Laplacian}}(G))$ 轮次内输出一个满足 $c^T x \leq OPT + \epsilon$ 的向量 $x \in \Omega^o$ 。

直观地说， U 是 LP 中出现的任意常数或变量大小的上界。对于图问题，这通常由 n 和 M的多项式所界定。

该定理的正式表述可能看起来有些复杂；本质上，它意味着我们可以求解约束矩阵能够用以下矩阵表示的线性规划：

邻接矩阵，其中每个节点都知道约束矩阵中与其关联边对应的元素。类似地，节点必须输出与其关联边对应的变量。这包括流问题，见附录 A。也包括近似分数最大匹配。不过，在这里，运行时间远不及 Ahmadi、Kuhn 和 Oshman [AKO18] 的 $\epsilon = \Omega(1/n^{1/4})$ 运行时间 $O(\log(nM)/\epsilon^2)$（至少对于 $\epsilon = \Omega(1/n^{1/4})$ 而言）。

1.2 相关工作

分布式流算法 我们的主要参考对象是 Forster、Goranci、Liu、Peng、Sun 和 Ye [FGL+21]。他们提供了此前最优的最小费用流求解器，其耗时为 $m^{3/7+o(1)}(\sqrt{n}D^{1/4} + D)$ 轮[1]。他们的方法采用了 Cohen、Mądry、Sankowski 和 Vladu [CMSV17], 提出的框架，该框架使用另一种内点法的 $\tilde{O}(m^{3/7})$ 次迭代来求解表示该问题的（不同的）线性规划。我们将迭代次数降至$\tilde{O}(n^{1/2})$。此外，该方法得到的是一个近似解，必须通过运行 $\tilde{O}(m^{3/7})$ 次最短路径计算将其转化为精确解。目前，最先进的最短路径计算算法需要 $\tilde{O}(\sqrt{n}D^{1/4} + D)$ 轮 [CM22], ，这已经比求解拉普拉斯矩阵系统的全局（近乎最优）轮复杂度（略微）更差。此外，这意味着他们的设置无法受益于近期（几乎）普适最优拉普拉斯求解器的进展。我们的方法将线性规划求解到更高的精度，从而通过内部舍入过程得到精确解，无需进一步进行最短路径计算。另一项改进是， [FGL+21] 只在具有单位容量的图中求解最小费用流，而我们可以处理任意容量。进一 步地， [FGL+21] 为具有任意容量的图提供了最大流算法，其耗时为 $\tilde{O}(m^{3/7}U^{1/7}(n^{o(1)}(\sqrt{n} + D) + \sqrt{n}D^{1/4}) + \sqrt{m})$ 轮。这是此前在 CONGEST 模型中关于最大流的最优结果。

对于近似版本，还存在一些仅适用于无向图的进一步结果。Ghaffari 等人 [GKK+18] 在加权无向图中，用 $n^{o(1)}(\sqrt{n} + D)/\epsilon^3$ 轮次给出了一个 $(1 + \epsilon)$-近似最大流。进一步地，Becker 等人 [BFKL21] 在无向图中，用 $\tilde{O}(n/\epsilon^2)$ 轮次给出了单位容量最小费用流的 $(1 + \epsilon)$-近似。

流问题的内点法 通过内点法求解流问题这一研究方向由 Daitch 和 Spielman [DS08], 开创，他们借助 Spielman 和 Teng [ST04] 的拉普拉斯求解器，设计出一个运行时间为 $\tilde{O}(m^{3/2})$ 的算法。最新进展是 Chen 等人 [CKL+22]提出的近线性时间算法。然而，他们的算法需要进行 $\Omega(m)$ 次迭代，这似乎使其难以在分布式环境中高效实现，因为任何直观的实现方式都至少需要每次迭代一轮。迭代次数最少的算法分别需要 $\Theta(m^{3/7})$ 次迭代 [M10、CMSV17], 或 $\Theta(\sqrt{n})$ 次迭代 [LS14]。在我们的工作中，我们展示了如何在 CONGEST 模型中高效实现后者。

分布式拉普拉斯求解器与快捷方式质量 Forster 等人 [FGL+21]提供了一个 CONGEST 模型算法，其复杂度为 $T_{\text{Laplacian}}(G) = n^{o(1)}(\sqrt{n} + D)$，并证明这是本质上最优的。对于任意图，我们知道 $T_{\text{Laplacian}}(G) = \Omega(D)$，然而事实证明，对于每个实例而言，$\sqrt{n}$-项并非必要。为精确定义这一点，我们定义图的快捷方式质量，该概念由 Ghaffari 和 Haeupler [GH16]引入。直观地说，快捷方式质量告诉我们，在给定节点的某个划分后，分别在每个部分上计算某个简单函数（例如，节点所持值的最小值）有多容易。由于分布式算法设计通常以此类函数为核心，因此快捷方式质量既可用于获得更好的上界，也可用于获得更好的下界。

[1]为简便起见，在讨论相关工作时，我们只考虑权重受 poly(m) 限制的图。

定义 1.4. 设 $G=(V,E)$ 是一个无向图，其节点集 V 被划分为 k 个不相交子集 $V=P_1\uplus P_2\uplus\cdots\uplus P_k$，使得每个诱导子图 $G[P_i]$ 都是连通的。一组 k 子图 $H_1,\cdots,H_k$ 称为 G 的一个捷径，其拥塞度为 c，扩张度为 d，如果

1. $G[P_i]\cup H_i$ 的（跳数）直径至多为 d；
2. 每条边至多被包含在 c 个图 H_i 中。

捷径的质量 定义为 $c+d$。 G的快捷方式质量 记为 $SQ(G)$，其定义为：将 V 划分为连通部分时，最坏情况划分的最小快捷方式质量。

Anagnostides 等人 [ALH+22] 根据~快捷方式质量，为拉普拉斯求解提供了高效算法。此外，他们给出了一个 Ω(*SQ*(*G*)) 下界。

定理 1.5 ([ALH+22])。 在 *CONGEST* 模型中，存在一个误差为 $\epsilon>0$ 的拉普拉斯求解器；给定一个图 G，其运行时间为 $n^{o(1)}\,\mathrm{poly}(SQ(G))\log(1/\epsilon)$ 轮次。对于次密度为 δ 且跳数直径为 D的图，运行时间为 $n^{o(1)}\delta D\log(1/\epsilon)$。

注意，对于次密度为 $n^{o(1)}$ 的图，该算法需要 $n^{o(1)}D\log(1/\epsilon)$ 轮次，与下界相比仅相差 $n^{o(1)}$ 因子。这包括平面图、 $n^{o(1)}$-亏格图、$n^{o(1)}$-树宽图以及排除小图图类。

进一步注意，特别是在图 G 具有 $SQ(G)=n^{o(1)}$ 的情况下，该算法需要$n^{o(1)}\log(1/\epsilon)$ 轮次。这包括扩展图、跳数约束扩展图，以及上文提到的具有受限直径 $D=n^{o(1)}$ 的类别。

2 概述与技术

2.1 线性规划求解器

对于我们的线性规划求解器，我们给出了 Lee 和 Sidford 的 [LS14] 线性规划求解器在 CONGEST 模型中的实现。关于正确性，请参阅 [LS14]。类似地，Forster 和 de Vos [FdV 22] 给出了该算法在广播拥塞团模型中的实现。在这一分布式模型中，每一轮中每个节点都可以向网络中的其他每个节点发送相同的 $O(\log n)$-比特消息。相比之下，在 CONGEST 模型中，节点可以发送不同的消息，但只能发送给其邻居。Forster 和 de Vos 基本上证明了该线性规划框架使用 $\sqrt{n}$次迭代，其中每次迭代包括：

(1) 垫 涉及某个矩阵的矩阵-向量乘法，其中矩阵条目对应边 ；

(2) 近似求解拉普拉斯系统；

(3) 计算杠杆分数；

(4) 投影到混合范数球上。

对于本节（以及定理 1.3 的表述），我们假设矩阵-向量乘法可以高效完成。实际上，这意味着我们必须能够用邻接矩阵来表示约束矩阵。换句话说，约束应当仅对应于边。

对于（2），我们可以使用 [FGL+21、ALH+22]，正如第 1.2 节所述。

关于 (3)；矩阵 M的杠杆分数定义为 $\sigma(M):=\mathrm{diag}(M(M^TM)^{-1}M^T)$，其中 diag(·) 返回对角线向量。我们指出（类似于 [LS14，FdV22]），利用 Johnson-Lindenstrauss 引理，我们可以通过一些局部采样以及少量的矩阵-向量乘法和拉普拉斯方程求解来计算出足够精确的近似值。详细信息请参见第 3 节。

剩下要证明的是，我们可以“投影到混合范数球上”。这里的目标是将向量 $a \in \mathbb{R}^m$ 投影到一个混合范数球上。具体而言，给定 $l \in \mathbb{R}^m$，我们考虑混合范数为 1 的球：$B := \{x : \|x\|_2 + \|l^{-1}x\|_\infty \le 1\}$。现在我们需要计算距离 a 最近的 $x \in B$，更正式地说，我们需要计算

$$\arg\max_{\|x\|_2+\|l^{-1}x\|_\infty \le 1} a^T x.$$

我们通过借鉴 [LS14, FdV22] 中的思路来实现这一点。详情见第 3 节。

2.2 最小费用流

设 $G = (V, E)$ 为一个有向图，其整数容量为 $c \in \mathbb{Z}^m_{\ge 0}$，整数费用为 $q \in \mathbb{Z}^m_{\ge 0}$，源点和汇节点分别为 s 和 t。最小费用（最大）流 问题要求在所有最大值流中，找到费用最小的 s-t流。更形式化地说，如果 $f_e \le c_e$ 对所有 $e \in E$ 和 $\sum_{e\in E: v\in e} f_e = 0$均成立，我们就称 $f \in \mathbb{R}^E_{\ge 0}$ 是一个 s-t 流。该流的值为 $\sum_{v\in V:(s,v)\in E} f_{(s,v)}$。最大流是值最大的流，而最小费用（最大）流则是在所有最大值流中费用最小的流 $\sum_{e\in E} f_e q_e$。

最小费用流问题有一个自然对应的线性规划。然而，当前最先进的线性规划求解器只能提供近似解。为了将其高效地转化为精确解，我们不采用教科书式线性规划表述，而采用一种与之密切相关的表述。在以高精度（加性误差 $\epsilon = O(1/\mathrm{poly}(m))$）求解该问题后，我们利用一个众所周知的事实：对于输入为整数的最小费用流问题，存在一个取整数值的最优解 [KT06], ，并在内部将近似的分数解舍入为一个最优整数解。

技术贡献在于证明这一特定的线性规划表述满足定理 1.3 的要求。证明见附录 A。

3 DistributedLP 求解器

本节中，我们给出一种在线性规划求解器给定拉普拉斯求解器时求解线性规划的算法。首先，我们重述该问题的形式化描述。对于所有 $i \in [m]$，令 $A \in \mathbb{R}^{m\times n}$、$b \in \mathbb{R}^n$、$c \in \mathbb{R}^m$、$l_i \in \mathbb{R} \cup \{-\infty\}$，以及 $u_i \in \mathbb{R} \cup \{+\infty\}$，其中我们假设 $l_i \ne -\infty$ 或 $u_i \ne +\infty$。我们试图求解的线性规划如下

$$\mathrm{OPT} := \min_{\substack{x\in\mathbb{R}^m : A^T x = b \\ \forall i\in[m]: l_i \le x_i \le u_i}} c^T x.$$

我们假设线性规划 $\Omega^o := \{x \in \mathbb{R}^m : A^T x = b,\ l_i \le x_i \le u_i\}$ 的可行解集合非空。

定理 1.3。设 $A \in \mathbb{R}^{m\times n}$ 是一个秩为 $(A) = n$ 的约束矩阵，$b \in \mathbb{R}^n$ 是一个需求向量，$c \in \mathbb{R}^m$ 是一个成本向量。此外，设 $x_0 \in \Omega^o$ 是给定的初始点。假设一个 *CONGEST* 网络由 n 个节点组成，其中每个节点 i 都知道满足 $A_{ji} \ne 0$ 的 j-*th* 行 A 的全部内容，并且当 $A_{ji} \ne 0$ 时知道 $(x_0)^j$。此外，假设对于每个 $y \in \mathbb{R}^n$ 和正对角矩阵 $W \in \mathbb{R}^{m\times m}$，我们都能在 $T_{\mathrm{Laplacian}}(G)$ 轮内将 $(A^T W A)^{-1} y$ 计算到精度 $\mathrm{poly}(1/m)$。令 $U := \max\{\|1/(u - x_0)\|_\infty, \|1/(x_0 - l)\|_\infty, \|u - l\|_\infty, \|c\|_\infty\}$。那么，以高概率，*CONGEST* 算法 *LPSolve* 在 $\tilde{O}(\sqrt{n}\log^3(U/\epsilon) T_{\mathrm{Laplacian}}(G))$ 轮内输出一个满足 $c^T x \le \mathrm{OPT} + \epsilon$ 的向量 $x \in \Omega^o$。

本节给出的算法是 Lee 和 Sidford 的 [LS14] 在 CONGEST 模型中的实现。正确性证明请参见他们的工作。对于我们修改的两个子程序——计算杠杆分数和投影到混合范数球上，我们给出了正确性分析。本节其余部分将介绍该算法并 Axaда

证明轮次数量的界。在两者中，我们都遵循 Forster 和 de Vos [FdV22], 的记号，他们为广播拥塞团模型提供了等价结果。

Lee 和 Sidford [LS19][2] 表明，只需以 poly(1/m) 的精度求解涉及 $A^T DA$ 的方程。我们在分析运行时间时利用这一事实，并通过假设这些方程可以精确求解来简化说明。同样，我们只需以 poly(ϵ/(mU)) 的精度执行邻接矩阵与对角矩阵的矩阵-向量乘法。此外，我们可以假设所有数值都以 poly(mU/ϵ) 为上界。由于 CONGEST 模型的带宽˜限制，这些乘法需要 O(log(U/ϵ)) 轮次。计算结束时，一条边两端相邻的节点都知道该边的值。

˜此外，在运行时间界中我们始终使用 D = O ($T_{\text{Laplacian}}(G)$)，这是成立的，因为
˜ ˜$T_{\text{Laplacian}}(G)$ =Ω($SQ(G)$) =Ω(D) [ALH+22]。

定义与设置 从较高层面来看，我们执行加权路径跟踪内点法。这意味着，在若干次迭代过程中，给定当前点 $x^{(i)} \in \Omega^o$，我们找到一个更接近最优解的点 $x^{(i+1)} \in \Omega^o$ 。为了确保点 $x^{(i)}$ 远离边界，我们需要控制 $l_j \le x_j^{(i)} \le u_j$ 对 $j \in [n]$的取值。这通过使用一个障碍函数 $\phi_i(x_i)$来实现；当 x_i 趋近边界（即趋近 l_j 或 u_j）时，该函数趋近 ∞ 。于是，路径如下所示：

$$x^{(i)} = \arg\min_{A^T x = b} \left(i \cdot c^T x + \sum_{j \in [m]} \phi_j(x_j) \right). \tag{1}$$

为了实现这一点，我们需要了解关于 ϕ 的更多性质，由此引出自协调障碍函数的定义。

定义 3.1。 凸的三次连续可微函数 $\phi: K \to \mathbb{R}^n$ 是开凸集 $K \subseteq \mathbb{R}^n$ 上的 ν-自协调障碍函数，若满足以下三个条件

1. $\lim_{i \to \infty} \phi(x_i) = \infty$ 对于所有满足 $x_i \in K$ 收敛到 K 的边界的序列 $(x_i)_{i \in \mathbb{N}}$ 均成立。
2. $|\phi'''(x)[h,h,h]| \le 2|\phi''(x)[h,h]|^{3/2}$ 对所有 $x \in K$ 和 $h \in \mathbb{R}^n$均成立。
3. $|\phi'(x)[h]| \le \sqrt{\nu}|\phi''(x)[h,h]|^{1/2}$ 对所有 $x \in K$ 和 $h \in \mathbb{R}^n$均成立。

在我们的情形中，我们按如下方式选择 ϕ 。

- 如果 l_i 是有限的且 $u_i = +\infty$，我们使用对数障碍： $\phi_i(x) := -\log(x - l_i)$。
- 如果 $l_i = -\infty$ 和 u_i 有限，我们使用对数障碍： $\phi_i(x) := -\log(u_i - x)$。
- 如果 l_i 和 u_i 有限，我们使用三角障碍： $\phi_i(x) := -\log \cos(a_i x + b_i)$，其中$a_i := \frac{\pi}{u_i - l_i}$和 $b_i := -\frac{\pi}{2}\frac{u_i + l_i}{u_i - l_i}$。

这是一个 1-协调障碍函数 LS。由于我们只需要局部了解约束，因此可以在 CONGEST 模型中内部计算该函数。将此函数用于方程 1，可得到一种 ($\sqrt{m}$log(1/ϵ)) 迭代方法 Ren。

我们可以将公式 1 推广为

$$x^{(i)} = \arg\min_{A^T x = b} \left(i \cdot c^T x + \sum_{j \in [m]} g_j(x)\phi_j(x_j) \right), \tag{2}$$

[2]在本节中，当技术细节只能在 arXiv 版本中找到时，我们将引用 arXiv 版本 [LS19] ，而非会议版本 [LS14], 。

对于某些权重函数 $g_j: \Omega^o \to \mathbb{R}^m_{\geq 0}$。Lee 和 Sidford [LS14] 证明，使用正则化的 *Lewis* 权重时，我们只需要 $O(\sqrt{n}\log(1/\epsilon))$ 次迭代。

为了给出正则化 Lewis 权重函数的形式化定义，我们先引入一些一般记号。

- 对于任意矩阵 $M \in \mathbb{R}^{n\times n}$，我们用 $\mathrm{diag}(M) \in \mathbb{R}^n$ 表示 M 的对角线，即 $\mathrm{diag}(M)_i := M_{ii}$。
- 对于任意向量 $x \in \mathbb{R}^n$，我们用大写 $X \in \mathbb{R}^{n\times n}$ 表示与 x 对应的对角矩阵，即 $X_{ii} := x_i$，且当 $i \neq j$ 时有 $X_{ij} := 0$。
- 对于 $x \in \Omega$，我们记作 $A_x := (\Phi''(x))^{-1/2} A$。
- 对于 $h: \mathbb{R}^n \to \mathbb{R}^m$ 和 $x \in \mathbb{R}^n$，我们用 $J_h(x) \in \mathbb{R}^{m\times n}$ 表示 h 在 x 处的雅可比矩阵，即 $[J_h(x)]_{ij} := \frac{\partial}{\partial x_j} h(x)_i$。
- 对于正的 $w \in \mathbb{R}^n_{>0}$，我们令 $\|\cdot\|_w$ 为由 $\|x\|_w^2 = \sum_{i\in[n]} w_i x_i^2$ 定义的范数，并令 $\|\cdot\|_{w+\infty}$ 为由 $\|x\|_{w+\infty} = \|x\|_\infty + C_{\text{范数}}\|x\|_w$ 定义的混合范数，其中 $C_{\text{范数}} > 0$ 是稍后定义的某个常数。
- 每当我们对向量应用标量运算时，这些运算都是按坐标进行的，例如，对于 x、$y \in \mathbb{R}^n$，我们有 $[x/y]_i := x_i/y_i$，以及 $[x^{-1}]_i := x_i^{-1}$。

定义 3.2。可微函数 $g: \Omega^o \to \mathbb{R}^m_{>0}$ 是一个 (c_1, c_s, c_k)-权重函数，如果以下界对所有 $x \in \Omega^o$ 和 $i \in [m]$ 均成立：

- 大小界：$\max\{1, \|g(x)\|_1\} \leq c_1$；
- 敏感度界：$e_i^T G(x)^{-1} A_x (A_x^T G(x)^{-1} A_x)^{-1} A_x^T G(x)^{-1} e_i \leq c_s$；
- 一致性界：$\|G(x)^{-1} J_g(x) (\Phi''(x))^{-1/2}\|_{g(x)+\infty} \leq 1 - c_k < 1$.

我们将其记为 $C_{\text{norm}} := 24\sqrt{c_s} c_k$。

沿用 Lee 和 Sidford [LS14], ，我们使用正则化 *Lewis* 权重。ℓ_p-Lewis 权重推广了一种称为杠杆分数的 ℓ_2 行重要性度量。它们是近似矩阵 ℓ_p-范数的关键工具。

定义 3.3. 对于 $M \in \mathbb{R}^{m\times n}$，其秩 $(M) = n$，我们令

$$\sigma(M) := \mathrm{diag}(M(M^T M)^{-1} M^T)$$

表示杠杆分数 M。对于所有 $p > 0$，我们将 ℓ_p-Lewis 权重 $w_p(M)$ 定义为 $1 \quad 1$ 满足 $w = \sigma(W^{\frac{1}{2}-\frac{1}{p}} M)$ 的唯一向量 $w \in R^m_{>0}$，其中 $w = \mathrm{diag}(W)$。对于 $p = 1 - \quad {}^1/_{\log(4m)}$ 和 $c_0 := \frac{n}{2m}$，我们将正则化 Lewis 权重定义为 $g(x) := w_p(M_x) + c_0$。

正则化 Lewis 权重函数 g 是一个具有 $c_1 \leq \frac{3}{2}n$、$c_s \leq 4$ 和 $c_k \leq 2\log(4m)$ [LS19] 的 (c_1, c_s, c_k)-权重函数。

如前所述，Lee 和 Sidford 证明，使用式（2）和此权重函数进行 $\tilde{O}(\sqrt{n}\log(1/\epsilon))$ 次迭代，可以得到我们 LP 的 ϵ 近似解。

计算杠杆分数 现有用于计算正则化 Lewis 权重的技术，会将计算杠杆分数 $\sigma(M) := \mathrm{diag}(M(M^TM)^{-1}M^T)$作为中间步骤。如下文所示，通过反复计算杠杆分数，我们可以近似得到 Lewis 权重。遗憾的是，目前还没有已知的能够精确计算杠杆分数的高效算法。然而，获得足够接近的近似值是可行的 [SS11,DMMW12, Mah11,LMP13, Woo14, CLM+15]。我们注意到 $\sigma(M)_i = \|M(M^TM)^{-1}M^Te_i\|_2^2$，并指出，根据 Johnson-Lindenstrauss 引理 [JL84] 该范数可以在投影到低维子空间时得到近似保持。特别地，Achlioptas [Ach03] 给出了一种显式的（随机化）构造。

定理 3.4 ([Ach03]). 设 $m > 0$ 为整数， $\eta, \beta > 0$ 为参数， $k = \Omega(\beta \log m/\eta^2)$为整数，且 $R \in \mathbb{R}^{k\times m}$ 为一个随机矩阵，其中 $R_{ij} = \pm 1/\sqrt{k}$，各自的概率为1/2。那么对于任意 $x \in \mathbb{R}^m$，我们有

$$\mathbb{P}[(1-\eta)\|x\|_2 \le \|Rx\|_2 \le (1+\eta)\|x\|_2] \ge 1 - m^{-\beta}.$$

现在我们准备给出一个算法来计算 $\sigma^{(\mathrm{apx})}$，使得 $(1-\eta)\sigma(M)_i \le \sigma_i^{(\mathrm{apx})} \le (1+\eta)\sigma(M)_i$，对于所有 $i \in [m]$ 均成立。

算法 1： ComputeLeverageScores(M, η)

1 设置 $k = \Theta(\log(m)/\eta^2)$ 。2 令 $R \in \mathbb{R}^{k\times m}$ 为一个矩阵，其中 $R_{ij} = \pm 1/\sqrt{k}$，每个的概率为 1/2。3 计算 $p^{(j)} = M(M^TM)^{-1}M^TR^{(j)}$，其中 $j \in [k]$。4 **返回** $\sum_{j=1}^k (p^{(j)})^2$。

引理 3.5。 对于任意 $\eta > 0$，以至少 $1 - 1/m^{O(1)}$ 的概率，*CONGEST* 模型算法 ComputeLeverageScores(M, η) 计算出 $\sigma^{\mathrm{apx}}(M)$，使得

$$(1-\eta)\sigma(M)_i \le \sigma^{\mathrm{apx}}(M)_i \le (1+\eta)\sigma(M)_i$$

~对所有 $i \in [m]$ 均成立。如果对于某个对角线 $W \in \mathbb{R}^{m\times m}$，有 $M = WA$，那么算法会在 $O((\log(U/\epsilon) + T_{\mathrm{Laplacian}}(G))/\eta^2)$ 轮次内终止。

证明。 ComputeLeverageScores(M, η) 返回 $\sigma^{\mathrm{apx}}(M)_i$。利用矩阵 $M(M^TM)^{-1}M^T$ 是对称矩阵这一性质，可得

$$\begin{aligned}\sigma^{\mathrm{apx}}(M)_i &:= \sum_{j=1}^k (M(M^TM)^{-1}M^TR^{(j)})_i^2 \\ &= \sum_{j=1}^k (RM(M^TM)^{-1}M^T)_{ji}^2 \\ &= \|RM(M^TM)^{-1}M^Te_i\|_2^2.\end{aligned}$$

由于我们还有 $\sigma(M)_i = \|M(M^TM)^{-1}M^Te_i\|_2^2$，由定理 3.4 可得

$$(1-\eta)\sigma(M)_i \le \sigma^{\mathrm{apx}}(M)_i \le (1+\eta)\sigma(M)_i,$$

以至少 $1 - 1/m^{O(1)}$ 的概率成立。利用联合界，我们可以得到对所有 $i \in [m]$ 同时成立的相同保证。

在 CONGEST 模型中，我们按如下方式构造所需的随机矩阵 R。对于每条边，ID 较大的节点抛掷 k 枚硬币，以确定 $\pm 1/\sqrt{k}$ 的值，并通过该边发送结果。这需要 $O(k/\log n) = O(1/\eta^2)$ 轮次。

对于第 3 行中的计算，我们注意到，可以将其视为 k 次

- 次矩阵-向量乘法 $M^T R_{(j)}$，随后
- 一个拉普拉斯系统求解（$M^T M$）$^{-1} M^T R_{(j)}$，如 $M^T M = A^T W^2 A$，接着进行
- 一次矩阵-向量乘法 $M(M^T M)^{-1} M^T R_{(j)}$。

~第一步和最后一步可在 O (log(U/ϵ)) 轮次内完成，而拉普拉斯求解可在 $T_{\text{Laplacian}}(G)$ 轮次内完成。最后，第 4 行可在内部完成。因此，总运行时间为 ~O((log(U/ϵ) $+ T_{\text{Laplacian}}(G)$)/$\eta^2$)。 □

计算权重函数 接下来，我们给出用于计算初始权重以及在路径查找算法中更新权重的算法。由于我们使用后者来完成前者，因此先介绍后者。

Algorithm 2: ComputeApxWeights($M, p, w^{(0)}, \eta$)

$L = \max\{4, \frac{8}{p}\}$, $r = \frac{p^2(4-p)}{2^{20}}$, and $\delta = \frac{(4-p)\eta}{256}$.
$T = \left\lceil 80\left(\frac{p}{2} + \frac{2}{p}\right)\log\left(\frac{pn}{32\eta}\right)\right\rceil$.
for $j = 1, \ldots, T-1$ **do**
  $\sigma^{(j)}$ =ComputeLeverageScores($W_{(j)}^{\frac{1}{2}-\frac{1}{p}} M, \delta/2$).
  for $i \in [m]$ **do**
    Let $w_i^{(j+1)}$ be the median of $(1-r)w_i^{(0)}$, $w_i^{(j)} - \frac{1}{L}\left(w_i^{(0)} - \frac{w_i^{(0)}}{w_i^{(j)}}\sigma_i^{(j)}\right)$, and $(1+r)w_i^{(0)}$.
return $w^{(T)}$.

引理 3.6。 设 $W \in R^{m\times m}$ 为某个对角矩阵， $w^{(0)} \in \mathbb{R}^m_{>0}$ 为一个向量，且 $\eta \in (0,1]$和 $p \in [1-1/\log(4m), 2]$ 为参数。令 $M = WA$。则 ComputeApxWeights($M, p, w^{(0)}, \eta$) 返回近似权重，其运行时间为

$$\tilde{O}\left(\frac{\log(1/\eta)}{\eta^2}(\log(U/\epsilon) + T_{\text{Laplacian(G)}})\right)$$

轮次。

证明。 该算法由 T = ~O ($(p + \frac{1}{p})$ log(p/η)) 次迭代组成。根据假设~， $p \in [1 - 1/\log(4m), 2],$，我们得到 T =O (log(1/η)。在每次迭代中，我们调用过程1 1_ComputeLeverageScores($W^{\frac{1}{2}-\frac{1}{p}}{}_{(j)} M, \delta/2$)，并计算一些中位数，后者可以在内部完成。调用 ComputeLeverageScores 需要 ~O ((log(U/ϵ) =) +$T_{\text{Laplacian}}(G)$)/($\delta/2$)^2) O((log(U/ϵ) $+ T_{\text{Laplacian}}(G)$)/$\eta^2$) 轮次，因此总运行时间如所述。 □

关于近似权重的性质和正确性，请参阅 [LS19]。使用以下算法，我们计算初始权重。我们通过迭代地将

全 1 向量更接近初始权重向量。

算法 3： ComputeInitialWeights($A, p_{\text{target}}, \eta$)

1 $p = 2$.
2 $w = 12c_{\text{k}}\mathbb{1}$。
3 **当** $p \neq p_{\text{target}}$ **时做】【**。
4 $h = \frac{\min\{2,p\}}{\sqrt{n}\text{对数}\frac{me^2}{n}} \cdot r$.
5 令 $p^{(\text{new})}$ 为 $p-h$、p_{target} 和 $p+h$ 的中位数。
6 w =ComputeApxWeights($A, p^{(\text{new})}, w^{p^{(\text{new})}/p}, \frac{p^2(4-p)2^{22}}{}$)
7 $p = p$（新）.
8 **返回** ComputeApxWeights($A, p_{\text{target}}, w, \eta$)。

引理 3.7。 设 $\eta \in (0,1]$ 和 $p_{\text{target}} \in [1-1/\log(4m), 2]$ 为参数，则 *CONGEST*˜模型算法 ComputeInitialWeights($A, p_{\text{target}}, \eta$) 在 $O((\sqrt{n}+\log(1/\eta)\,\eta^2(\log(U/\epsilon)+T_{\text{Laplacian(G)}}))$ 轮次内返回初始权重。

证明。 第 3 行的 while 循环经过 $O(\sqrt{n}(p_{\text{target}}+\frac{1}{p_{\text{target}}})\log(m/n))$ 次迭代后结束。利用 ˜有 $p_{\text{target}} \in [1-1/\log(4m), 2],$，可将其简化为 O ($\sqrt{n}$) 次迭代。每次迭代包括内部计算 h和若干个中位数，以及调用 ComputeApxWeights。该调用所需的精度为 $p^2(4-p)2^{22}$ ，对于我们的 p 范围，该值为 $\Omega(1)$。因此，while 循环总共需要 ˜O($\sqrt{n}$(log(U/ϵ) +$T_{\text{Laplacian}}(G)$)) 轮次。

然后在第 8 行，我们以精度 η 调用 ComputeApxWeights，该过程需要˜O(1og$(1/\eta)\,\eta^2$ (log(U/ϵ)+$T_{\text{Laplacian(G)}}$)) 轮次。综合起来，便得到所述的运行时间。 □

关于初始权重的性质和正确性，请参见 [LS19]。

算法 本节将给出求解线性规划的形式化算法，并通过一系列引理证明每个子程序的运行时间。

算法 4： LPSolve(x_0, ϵ)

输入： 一个初始点 x_0，满足 $A^T x_0 = b$。
1 w =ComputeInitialWeights($A, 1-1/\log(4m)$, $\frac{1}{2^{16}\log^3 m}$)+$\frac{n}{2m}$, $d = -w\phi'(x_0)$.
2 $t_1 = (2^{27}m^{3/2}U^2\log^4 m)^{-1}$, $t_2 = \frac{2m}{\eta}$, $\eta_1 = \frac{1}{2^{18}\log^3 m}$, and $\eta_2 = \frac{\epsilon}{8U^2}$.
3 $(x^{(\text{new})}, w^{(\text{new})})$ =PathFollowing($x_0, w, 1, t_1, \eta_1, d$).
4 $(x^{(\text{final})}, w^{(\text{final})})$ =PathFollowing($x^{(\text{new})}, w^{(\text{new})}, t_1, t_2, \eta_2, c$).
5 **return** $x^{(\text{final})}$.

计算出初始权重后，该算法会调用两次 PathFollowing：第一次将给定的初始点沿成本向量 c 移向一个中心起点，第二次则从该点继续沿路径前进。PathFollowing 算法如下。

算法 5: PathFollowing($x, w, t_{\text{start}}, t_{\text{end}}, \eta, c$)

1 $t = t_{\text{开始}}$, $R = \frac{1}{768c_{\text{k}}^2\log(36c_1c_{\text{s}}c_{\text{k}}m)}$ 并且 $\alpha = \frac{R}{1600\sqrt{n}\log^2 m}$.
2 **当** $t \neq t_{\text{end}}$ **时**
3 (x, w) =CenteringInexact(x, w, t, c)】【。
4 令 t 为 $(1-\alpha)t$, t_{end} 和 $(1+\alpha)t$ 的中位数。
5 **对** $i = 1, \ldots, 4c_{\text{k}}\log(\frac{1}{\eta})$ **do**
6 (x, w) =CenteringInexact(x, w, t_{end}, c)。
7 **返回** (x, w).

进展步骤由 CenteringInexact 完成，具体如下 .

算法 6： CenteringInexact(x, w, t, c)

1 $R = \frac{1}{768c_{\mathrm{k}}^2 \log(36c_1 c_{\mathrm{s}} c_{\mathrm{k}} m)}$, and $\eta = \frac{1}{2c_{\mathrm{k}}}$.

2 $\delta = \left\| P_{x,w}\left(\frac{tc+w\phi'(x)}{w\sqrt{\phi''(x)}}\right)\right\|_{w+\infty}$ `// where` $P_{x,w} := I - W^{-1}A_x(A_x^T W^{-1} A_x)^{-1} A_x^T$.

3 $x^{(\mathrm{new})} = x - \frac{1}{\sqrt{\phi''(x)}} P_{x,w}\left(\frac{tc-w\phi'(x)}{w\sqrt{\phi''(x)}}\right)$.

4 $z = \log\left(\texttt{ComputeApxWeights}(A_{x^{(\mathrm{new})}}, 1 - 1/\log(4m), w, e^R - 1)\right)$.

5 $u = \left(1 - \frac{6}{7c_{\mathrm{k}}}\right)\delta \cdot \texttt{ProjectMixedBall}(-\nabla\Phi\frac{\eta}{12R}(z - \log(w)), C_{\mathrm{norm}}\sqrt{w})$.

6 $w^{(\mathrm{new})} = \exp(\log(w) + u)$.

7 **return** $\left(x^{(\mathrm{new})}, w^{(\mathrm{new})}\right)$.

我们将在第 4 节介绍子程序 ProjectMixedBall。我们将按相反的顺序证明这三个算法的运行时间。

Lemma 3.8. *CONGEST* 模型算法 `CenteringInexact`(x, w, t, c) 在

$$\tilde{O}(\log^2(U/\epsilon)T_{\mathrm{Laplacian}}(G))$$

轮次内终止。

证明。 计算 $P_{x,w}\left(\frac{tc+w\phi'(x)}{w\sqrt{\phi''(x)}}\right)$ 需要 $\tilde{O}(T_{\mathrm{Laplacian}}(G))$ 轮次，利用内部计算来与对角矩阵相乘并进行拉普拉斯求解。为了计算 δ 并使其为每个节点所知，我们使用 $O(D\log(U/\epsilon))$ 轮次。

根据引理 3.6，需要 $O(\log(U/\epsilon) \cdot T_{\mathrm{Laplacian}}(G))$ 轮次。最后，我们此处需要 ProjectMixedBall，该算法需要

$$\tilde{O}(D\log^2(U/\epsilon)) = \tilde{O}(\log^2(U/\epsilon)T_{\mathrm{Laplacian}}(G))$$

根据引理 4.1，需要轮次。 □

我们用这一结果证明 PathFollowing 的运行时间。

引理 3.9。 令 $t_{\mathrm{start}}, t_{\mathrm{end}} \geq 1$ 和 $\eta \in (0,1]$ 为参数。*CONGEST* 模型算法`PathFollowing`($x, w, t_{\mathrm{start}}, t_{\mathrm{end}}, \eta, c$) 在

$$\tilde{O}(\sqrt{n}(|\log(t_{\mathrm{end}}/t_{\mathrm{start}})| + \log(1/\eta))\log^2(U/\epsilon)T_{\mathrm{Laplacian}}(G))$$

轮次。

证明。 首先注意，第 2 行的 while 循环需要 $O(\sqrt{n}(|\log(t_{\mathrm{end}}/t_{\mathrm{start}})| + \log(1/\eta)))$ 次迭代 [LS19]。每次迭代都包含一次对 CenteringInexact 的调用和内部计算。然后，第 5 行的 for 循环需要 $O(c_k\log(1/\eta))$ 次迭代，每次迭代都包含一次对 CenteringInexact 的调用。显然，这部分的运行时间由 while 循环的运行时间主导。

根据引理 3.8，需要 $O(\log^2(U/\epsilon)\tilde{T}_{\mathrm{Laplacian}}(G))$ 轮次，因此可得所述运行时间。 □

最后，我们给出完整算法的运行时间。

引理 3.10。 给定 $\epsilon > 0$，*CONGEST* 模型算法 `LPSolve`(x_0, ϵ) 在 $\tilde{O}(\sqrt{n}\log^3(U/\epsilon)T_{\mathrm{Laplacian}}(G))$ 轮次内终止。

证明。 除了一些内部计算外，该算法由三个不同部分组成：计算初始权重，以及使用不同参数调用两次 PathFollowing。

调用 ComputeInitialWeights 所需时间为

$$\tilde{O}(\sqrt{n}(\log(U/\epsilon) + T_{\text{Laplacian(G)}}))$$

轮次，因为我们以精度 2^{16} ^{1} $\log^3 m$ 调用它。

执行 PathFollowing($x_0, w, 1, t_1, \eta_1, d$) 需要

$$\tilde{O}(\sqrt{n}\log(U)\log^2(U/\epsilon)T_{\text{Laplacian}}(G))$$

轮次，根据引理 3.9 并代入 t_1 和 η_1。

执行 PathFollowing($x^{(\text{new})}, w^{(\text{new})}, t_1, t_2, \eta_2, c$) 需要

$$\tilde{O}(\sqrt{n}\log^3(U/\epsilon)T_{\text{Laplacian}}(G))$$

轮次，根据引理 3.9 并代入 t_1, t_2 和 η_2。

最后一个运行时间占主导，并得出所述结果。 □

4 在混合范数球上投影

本节介绍一种用于在混合范数球上投影的 CONGEST 算法。该问题定义如下：给定 $a, l \in \mathbf{R}^m$，求

$$\arg\max_{\|x\|_2+\|l^{-1}x\|_\infty \le 1} a^T x.$$

在原始工作中，Lee 和 Sidford [LS19] 首先对 m 个值进行排序，并预先计算 m 个函数在 a 和 l 上的值。在 CONGEST 模型中，这些操作开销很大。我们借鉴了 Forster 和 de Vos [FdV22]，的思路；他们在广播拥塞团模型中解决了同样的问题。大致思路是仅进行隐式排序，并通过二分搜索，将需要计算的函数数量减少到可控范围。我们在算法 7 中给出了伪代码，并在引理 4.1 的证明中提供了更多细节。伪代码中的二分搜索相当复杂，还包含一些令人望而生畏的方程。理解二者的最佳方式或许是仔细阅读证明。

请注意，该问题与 CONGEST 模型中的图结构关系不大；正如预期的那样，除了为通信建立一棵最短路径树外，该算法实际上并未利用图结构。

算法 7： ProjectMixedBall(a, l)

1 确定 $\{|a_i|/l_i : i \in [m]\}$ 的最小值、最大值和步长，并将这些可能值构成的空间记为 S。2 对于 $s \in S$，令 i 为与 s 最接近的值 $|a_i|/l_i$ 的索引。3 针对 g_i 对 S 执行二分搜索：4 计算 $j \in \{i-1, i\}$ 对应的 $\sum_{k\in[j]} |a_k||l_k|$、$\sum_{k\in[j]} a_k^2$ 和 $\sum_{k\in[j]} l_k^2$。5 在内部计算

$$g_i := \max_{t:i_t=i} t \sum_{k\in[i]} |a_k||l_k| + \sqrt{(1-t)^2 - t^2 \sum_{k\in[i]} l_k^2}\sqrt{\|a\|_2^2 - \sum_{k\in[i]} a_k^2}.$$

6 令 t 为最大 g_i 对应的索引。

7 $x_j^i := \begin{cases} \frac{t}{1-t}\operatorname{sign}(a_j)l_j & \text{if } j \in [i] \\ \sqrt{\frac{1-\left(\frac{t}{1-t}\right)^2 \sum_{k\in[i]} l_k^2}{\|a\|_2^2 - \sum_{k\in[i]} a_k^2}} a_j & \text{otherwise.} \end{cases}$

8 **return** x

引理 4.1。 假设向量 $a, l \in \mathbb{R}^m$ 分布在网络中，满足：*1*）对于每个 $i \in [m]$， a_i 和 l_i 恰好由一个节点知晓；*2*）一个节点知道 a_i 当且仅当它知道 l_i。此外，假设 $\|a\|_\infty, \|l\|_\infty \le O(\mathrm{poly}(m)U)$。那么算法 `ProjectMixedBall`$(a, l)$ 可以找到

$$\arg\max_{\|x\|_2+\|l^{-1}x\|_\infty \le 1} a^T x$$

在 $\tilde{O}(D\log^2(U/\epsilon))$ 轮次内，在 ***CONGEST*** 模型中达到 $O(1/(\mathrm{poly}(mU/\epsilon))$ 精度。

证明。我们将问题改写为对某个凹函数求最大值，该函数存在唯一的最大值，可通过在定义域上进行二分搜索找到。我们首先对 ℓ_2-范数进行参数化：

$$\begin{aligned}
\max_{\|x\|_2+\|l^{-1}x\|_\infty \le 1} a^T x &= \max_{0\le t\le 1}\left[\max_{\|x\|_2\le 1-t,\ -tl_i\le x_i\le tl_i} a^T x\right] \\
&= \max_{0\le t\le 1}(1-t)\left[\max_{\|x\|_2\le 1,\ -\frac{t}{1-t}l_i\le x_i\le \frac{t}{1-t}l_i} a^T x\right] \\
&= \max_{0\le t\le 1} g(t),
\end{aligned}$$

其中我们将 $g(t)$ 定义为

$$g(t) := (1-t)\left[\max_{\|x\|_2\le 1,\ -\frac{t}{1-t}l_i\le x_i\le \frac{t}{1-t}l_i} a^T x\right].$$

我们在概念上按单调递减顺序对 a 和 l 的值进行排序，即我们仅在证明中为这种记法对它们进行排序，算法并不会对这些值进行排序。接下来，我们将 i_t 记作满足以下条件的第一个坐标 $i \in [m]$，使得

$$\frac{1-\left(\frac{t}{1-t}\right)^2 \sum_{k\in[i_t]} l_k^2}{\|a\|_2^2 - \sum_{k\in[i_t]} a_k^2} \le \frac{\left(\frac{t}{1-t}\right)^2 l_i^2}{a_i^2}.$$

现在可以证明（例如参见 [LS19]），在 $g(t)$ 中取得最大值的向量为$x^{i_t} \in \mathbb{R}^m$，其定义为

$$x_j^{i_t} := \begin{cases} \frac{t}{1-t}\,\mathrm{sign}(a_j) l_j & \text{if } j \in [i_t] \\ \sqrt{\frac{1-\left(\frac{t}{1-t}\right)^2 \sum_{k\in[i_t]} l_k^2}{\|a\|_2^2 - \sum_{k\in[i_t]} a_k^2}}\, a_j & \text{otherwise.} \end{cases}$$

我们将其代入 $g(t)$ 的定义中：

$$g(t) = t \sum_{k\in[i_t]} |a_k||l_k| + \sqrt{(1-t)^2 - t^2 \sum_{k\in[i_t]} l_k^2}\sqrt{\|a\|_2^2 - \sum_{k\in[i_t]} a_k^2}.$$

注意， $g(t)$ 是一个凹函数（其二阶导数非正），因此它有唯一的最大值。我们通过在定义域内搜索来求得这个最大值。为此，我们将 g 改写为关于索引 i_t 的表达式：

$$\begin{aligned}
\max_{0\le t\le 1} g(t) &= \max_{0\le t\le 1} \max_{i\in[m]} g_i(t) \\
&= \max_{i\in[m]} \max_{t: i_t = i} g_i(t),
\end{aligned}$$

其中

$$g_i(t) := t \sum_{k\in[i]} |a_k||l_k| + \sqrt{(1-t)^2 - t^2 \sum_{k\in[i]} l_k^2}\sqrt{\|a\|_2^2 - \sum_{k\in[i]} a_k^2}.$$

现在固定索引 i，并假设一个节点知道 $\sum_{k\in[j]} |a_k||l_k|$、$\sum_{k\in[j]} a_k^2$ 和 $\sum_{k\in[j]} l_k^2$，对于 $j \in \{i-1, i\}$。那么我们可以在内部计算 $g_i := \max_{t:i_t=i} g_i(t)$，因为我们可以在内部找到满足 $i_t = i$ 的 t 的范围，因为我们有 $i_t \geq i_s$ if $t \leq s$，因此满足 $i_t = j$ 的 t 集合是一个区间。

接下来，我们描述如何计算这些和 $\sum_{k\in[j]}$。我们通过从持有值 a_j, l_j 的节点出发，构建一棵直径为 D 的最短路径树来实现。沿着树，我们分别汇总所有 $k \leq j$ 对应的 $|a_k||l_k|$, a_k^2 或 l_k^2 的值。结果可以广播给所有节点，且无需额外费用。注意，如果索引并非显式已知，持有 a_j 和 l_j 的节点可以先广播 $|a_j|/l_j$，然后其他节点仅在满足 $|a_i|/l_i \leq |a_j|/l_j$ 时才将其值 $|a_i||l_i|$（以及其他相应的值）加入汇总。由于这些值需要以精度 $\tilde{}$poly(mU/ϵ) 维护，发送一条消息最多需要 O(log(U/ϵ)) 轮次，因此整个过程 $\tilde{}$ 最多需要 O (Dlog(U/ϵ)) 轮次。

直观而言，此时我们只需对 $i \in [m]$ 进行简单的二分搜索即可完成；然而，问题在于我们只是从概念上对索引进行了排序，因此节点并不知道自己持有的值对应哪些索引。于是，我们改为对 $|a_i|/l_i$ 的可能取值进行二分搜索。同样地，利用从任意领导者出发的通信树，我们可以找出 $|a_i|/l_i$ 各值的全局最小值、全局最大值和步长（分母 l_i 的最小公倍数）。由于搜索空间中并非所有值都会出现，因此对于二分搜索中的给定值，我们取与之最接近的、实际出现的值。这样一来，总搜索空间大小为 O（poly(mU/ϵ)），因此我们需要 O(log(U/ϵ) 次迭代，每次耗费 O (Dlog(U/ϵ)) 轮次。

□

参考文献

[Ach03] Dimitris Achlioptas。适用于数据库的随机投影：使用二元硬币的 Johnson-Lindenstrauss。《计算机与系统科学杂志》，66(4):671–687，2003年。在 PODS 2001 上首次公布。[AKO18] Mohamad Ahmadi、Fabian Kuhn 和 Rotem Oshman。CONGEST 模型中的分布式近似最大匹配。载于第*32*届国际分布式计算研讨会，*DISC 2018*，第121卷*LIPIcs*，第6:1–6:17页。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2018年。[ALH+22] Ioannis Anagnostides、Christoph Lenzen、Bernhard Haeupler、Goran Zuzic 和 Themis Gouleakis。通过低拥塞快捷路径实现近乎普适最优的分布式拉普拉斯求解器。载于第*36*届国际分布式计算研讨会，*DISC 2022*，第246卷*LIPIcs*，第6:1–6:20页。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2022年。在 PODC 2022 上首次公布。[AMV20] Kyriakos Axiotis、Aleksander Mądry 和 Adrian Vladu。单位容量图中用于加速最小费用流的环流控制。载于第*61*届*IEEE*计算机科学基础年会，*FOCS 2020*，第93–104页。IEEE，2020年。[ASZ20] Alexandr Andoni、Clifford Stein 和 Peilin Zhong。通过低跳数模拟器实现并行近似无向最短路径。载于第*52*届*ACM SIGACT*计算理论年会，*STOC 2020*，第322–335页。ACM，2020年。[BFKL21] Ruben Becker、Sebastian Forster、Andreas Karrenbauer 和 Christoph Lenzen。分布式和

流式模型中的近乎最优近似最短路径与转运问题。*SIAM*计算杂志，50(3):815–856，2021年。在 DISC 2017 上首次公布。[CKL+22] Li Chen、Rasmus Kyng、Yang P. Liu、Richard Peng、Maximilian Probst Gutenberg 和 Sushant Sachdeva。近线性时间内的最大流和最小费用流。$CoRR$，abs/2203.00671，2022年。[CKM+14] Michael B. Cohen、Rasmus Kyng、Gary L. Miller、Jakub W. Pachocki、Richard Peng、Anup B. Rao 和 Shen Chen Xu。在近$m$log1/2n 时间内求解 SDD 线性系统。载于计算理论研讨会论文集，*STOC 2014*，第343–352页。ACM，2014年。[CLM+15] Michael B. Cohen、Yin Tat Lee、Cameron Musco、Christopher Musco、Richard Peng 和 Aaron Sidford。矩阵近似的均匀采样。载于理论计算机科学创新大会（*ITCS 2015*）论文集，第181–190页，2015年。[CM22] Shiri Chechik 和 Doron Mukhtar。具有改进界的 CONGEST 模型单源最短路径。《分布式计算》，35(4):357–374，2022年。在 PODC 2020 上首次公布。[CMSV17] Michael B. Cohen、Aleksander Mądry、Piotr Sankowski 和 Adrian Vladu。在$\tilde{o}(m^{10/7}\log w)$时间内求解负权最短路径和单位容量最小费用流（扩展摘要）。载于第二十八届*ACM-SIAM*离散算法研讨会，*SODA 2017*，第752–771页。SIAM，2017年。[DMMW12]Petros Drineas、Malik Magdon-Ismail、Michael W. Mahoney 和 David P. Woodruff。矩阵相干性和统计杠杆值的快速近似。《机器学习研究杂志》，13(1):3475–3506，2012年。在 ICML 2012 上首次公布。[DS08] Samuel I. Daitch 和 Daniel A. Spielman。通过内点算法更快地近似有损广义流。载于第*40*届*ACM*计算理论年会（*STOC 2008*）论文集，第451–460页，2008年。[Elk06] Michael Elkin。分布式最小生成树问题中时间-近似权衡的无条件下界。*SIAM*计算杂志，36(2):433–456，2006年。[FdV22] Sebastian Forster 和 Tijn de Vos。广播拥塞团中的拉普拉斯范式。载于*ACM*分布式计算原理研讨会论文集，*PODC 2022*，第335–344页。ACM，2022年。[FGL+21] Sebastian Forster、Gramoz Goranci、Yang P. Liu、Richard Peng、Xiaorui Sun 和 Mingquan Ye。次图稀疏化器与分布式拉普拉斯范式。载于第*62*届*IEEE*计算机科学基础年会，*FOCS 2021*，第989–999页。IEEE，2021年。[GH16] Mohsen Ghaffari 和 Bernhard Haeupler。平面网络的分布式算法 II：低拥塞快捷路径、最小生成树和最小割。载于 Robert Krauthgamer 编，第二十七届*ACM-SIAM*离散算法研讨会，*SODA 2016*，第202–219页。SIAM，2016年。

[GKK+18] Mohsen Ghaffari、Andreas Karrenbauer、Fabian Kuhn、Christoph Lenzen 和 Boaz Patt-Shamir。近乎最优的分布式最大流。《SIAM计算杂志》，47(6):2078–2117，2018年。曾发表于 PODC 2015。[Gre96] Keith D. Gremban。稀疏、对称、对角占优线性系统的组合预条件子。博士论文，卡内基梅隆大学，匹兹堡，1996年。[JL84] William B Johnson 和 Joram Lindenstrauss。到希尔伯特空间的利普希茨映射扩展。《当代数学》，26:189–206，1984年。[KLOS14] Jonathan A. Kelner、Yin Tat Lee、Lorenzo Orecchia 和 Aaron Sidford。用于无向图近似最大流的近乎线性时间算法及其多商品流推广。载于《第二十五届ACM-SIAM离散算法研讨会论文集》，SODA 2014，第217–226页。SIAM，2014年。[KLP+16] Rasmus Kyng、Yin Tat Lee、Richard Peng、Sushant Sachdeva 和 Daniel A. Spielman。用于联络拉普拉斯算子的稀疏化乔列斯基分解与多重网格求解器。载于《第48届ACM SIGACT计算理论研讨会论文集》，STOC 2016，第842–850页。ACM，2016年。[KMP11] Ioannis Koutis、Gary L. Miller 和 Richard Peng。用于 SDD 线性系统的近乎 m log n 时间求解器。载于《IEEE第52届计算机科学基础年会论文集》，FOCS 2011，第590–598页。IEEE 计算机学会，2011年。[KMP14] Ioannis Koutis、Gary L. Miller 和 Richard Peng。逼近求解 SDD 线性系统的最优性。《SIAM计算杂志》，43(1):337–354，2014年。曾发表于 FOCS 2010。[KOSZ13] Jonathan A. Kelner、Lorenzo Orecchia、Aaron Sidford 和 Zeyuan Allen Zhu。一种简单的组合算法，可在近乎线性时间内求解 SDD 系统。载于《第45届ACM计算理论年会论文集》（STOC 2013），第911–920页，2013年。[KS16] Rasmus Kyng 和 Sushant Sachdeva。拉普拉斯矩阵的近似高斯消元——快速、稀疏且简单。载于《IEEE第57届计算机科学基础年会论文集》，FOCS 2016，第573–582页。IEEE 计算机学会，2016年。[KT06] Jon M. Kleinberg 和 Éva Tardos。《算法设计》。Addison-Wesley，2006年。[Li20] Jason Li。用于近似最短路径的更快并行算法。载于《第52届ACM SIGACT计算理论年会论文集》，STOC 2020，第308–321页。ACM，2020年。[LMP13] Mu Li、Gary L. Miller 和 Richard Peng。迭代行采样。载于《第54届IEEE计算机科学基础年会论文集》（FOCS 2013），第127–136页，2013年。[LS14] Yin Tat Lee 和 Aaron Sidford。线性规划的寻路方法：在 $\tilde{O}(\sqrt{秩})$ 次迭代内求解线性规划，以及用于最大流的更快算法。载于《IEEE第55届计算机科学基础年会论文集》（FOCS 2014），第424–433页。IEEE 计算机学会，2014年。[LS19] Yin Tat Lee 和 Aaron Sidford。通过 $\tilde{O}(\sqrt{秩})$ 次线性系统求解来求解线性规划。CoRR，abs/1910.08033，2019年。

[LS20a] Yang P. Liu 和 Aaron Sidford。通过更快的散度最大化实现更快的最大流。CoRR，abs/2003.08929，2020年。[LS20b] Yang P. Liu 和 Aaron Sidford。通过更快的能量最大化实现更快的最大流。载于：第52届 ACM SIGACT 计算理论研讨会论文集（STOC 2020），第803–814页。ACM，2020年。[Mah11] Michael W. Mahoney。矩阵与数据的随机化算法。《机器学习前沿与趋势》，3(2):123–224，2011年。[Mą13] Aleksander Mądry。通过电流探索中心路径：从流到匹配，再回到流。载于：IEEE第54届计算机科学基础年会论文集（FOCS 2013），第253–262页。IEEE 计算机学会，2013年。[Mą16] Aleksander Mądry。利用增广电流计算最大流。载于：IEEE第57届计算机科学基础年会论文集（FOCS 2016），第593–602页。IEEE 计算机学会，2016年。[Pel00] David Peleg。分布式计算：局部性敏感方法。SIAM，2000年。[Pen16] Richard Peng。在 O(mpolylog(n)) 时间内近似求解无向最大流。载于：第27届 ACM-SIAM 离散算法年会论文集（SODA 2016），第1862–1867页。SIAM，2016年。[PR00] David Peleg 和 Vitaly Rubinovich。分布式最小权生成树构造时间复杂度的近乎紧确下界。SIAM计算杂志，30(5):1427–1442，2000年。曾于 FOCS 1999 上公布。[PS14] Richard Peng 和 Daniel A. Spielman。用于 SDD 线性系统的高效并行求解器。载于：计算理论研讨会论文集（STOC 2014），第333–342页。ACM，2014年。[Ren88] James Renegar。一种基于牛顿法的线性规划多项式时间算法。《数学规划》，40(1-3):59–93，1988年。[She13] Jonah Sherman。近似线性时间内求解近似最大流。载于：IEEE第54届计算机科学基础年会论文集（FOCS 2013），第263–269页。IEEE 计算机学会，2013年。[SHK+12] Atish Das Sarma、Stephan Holzer、Liah Kor、Amos Korman、Danupon Nanongkai、Gopal Pandurangan、David Peleg 和 Roger Wattenhofer。分布式验证与分布式近似的难度。SIAM计算杂志，41(5):1235–1265，2012年。曾于 STOC 2011 上公布。[SS11] Daniel A. Spielman 和 Nikhil Srivastava。通过有效电阻实现图稀疏化。SIAM 计算杂志，40(6):1913–1926，2011年。曾于 2008年计算理论研讨会（STOC）上公布。[ST04] Daniel A. Spielman 和 Shang-Hua Teng。用于图划分、图稀疏化和求解线性系统的近线性时间算法。载于：第36届 ACM 计算理论研讨会年会论文集（STOC 2004），第81–90页。ACM，2004年。[vdBLN+20] Jan van den Brand、Yin Tat Lee、Danupon Nanongkai、Richard Peng、Thatchaphol Saranurak、Aaron Sidford、Zhao Song 和 Di Wang。中等稠密图上的近线性时间二分图匹配。载于：第61届 IEEE 计算机科学基础年会论文集（FOCS 2020），第919–930页。IEEE，2020年。

[Woo14] David P. Woodruff. 草图作为数值线性代数的工具。*Foundationsand Trends® in Theoretical Computer Science*，10(1–2):1–157，2014年。

一个最小费用流

在本节中，我们通过将定理 1.3 应用于一个合适的线性规划，并相应地将结果舍入为精确解，来证明定理 1.1。Daitch 和 Spielman [DS08], 最早提出了这种特定的最小费用流线性规划表述，Lee 和 Sidford [LS19], 以及 Forster 和 de Vos [FdV22]也采用了这种表述。与定理 1.1 的表述不同，本节中我们用 $|V|$ 和 $|E|$ 表示节点集和边集的大小。按照第 3 节的做法，我们用 n 和 m 表示线性规划的维度。我们用M 表示最大的边容量和费用。

令 $B \in \mathbf{R}^{(|V|-1)\times|E|}$ 为边-节点关联矩阵，其中省略了源点 s 对应的行。线性规划的变量包括 $x \in \mathbf{R}^{|E|}$、y、$z \in \mathbf{R}^{|V|}$和 $F \in \mathbf{R}$。该线性规划定义如下。

$$\begin{aligned} \min\ & \tilde{q}^T x + \lambda(1^T y + 1^T z) - 2n\tilde{M}F \\ subject\ to\ & Bx + y - z = Fe_t, \\ & 0 \le x_i \le c_i, \\ & 0 \le y_i \le 4|V|M, \\ & 0 \le z_i \le 4|V|M, \\ & 0 \le F \le 2|V|M, \end{aligned}$$

其中 $\tilde{M} := 8|E|^2M^3$、$\lambda := 28160|E|^82M^9$ 和 $\tilde{q} = c + r$，其中对于每条边 r_e 是从 $\{\frac{1}{4|E|^2M^2}$、$\frac{2}{4|E|^2M^2}$、…、$\frac{2|E|M}{4|E|^2M^2}\}$ 中均匀随机选取的一个数。Daitch 和 Spielman [DS08] 证明，该问题以至少 1/2 的概率具有唯一解，该解同时也是原问题的有效解。应用这一归约后，我们（概念上）将所有内容缩放$4|E|^2M^2$ ，以确保成本向量再次为整数。

我们将变量设定如下，以获得一个初始内点：$F = |V|M, x = \frac{c}{2}, y = 2|V|M\mathbb{1} - (B\frac{c}{2})^- + Fe_t, z = 2|V|M\mathbb{1} + (B\frac{c}{2})^+$，其中我们分别用 a^+和 a^-表示由下式定义的向量

$$(a^+)_i := \begin{cases} a_i & \text{if } a_i \ge 0; \\ 0 & \text{else.} \end{cases} \qquad \text{and} \qquad (a^-)_i := \begin{cases} a_i & \text{if } a_i \le 0 \\ 0 & \text{else.} \end{cases}$$

。

接下来，我们说明如何将该线性规划的一个 ϵ-近似解 x 转换为最小费用流问题的精确解。引入额外变量 y 和 z后，流量可能至多超出 $1^T y + 1^T z \le \epsilon$。为纠正这一点，我们设定 $\tilde{x} = (1-\epsilon)x$。我们设定$\epsilon := \frac{1}{320|E|^4M^5}$，此时相对于唯一解的误差至多为 1/6 [LS19], ，因此只需将每条边上的流量舍入到最接近的整数即可。显然，这两个步骤都可以在 CONGEST 模型中本地完成。

为求解上述线性规划，我们使用定理 1.3 和 $A = [B\ I\ -I\ -e_t]^T$。实际上，这并未使用整个网络，而只使用了 $n = |V|-1$ 个节点，因为源点无需参与计算。由于节点关联矩阵 B 的知识已按要求分布，A 的知识也已按要求分布。最后一步是证明我们可以在 A^TWA 中于 $T_{\text{Laplacian}}$内求解方程。这由 [Gre96] 推出，并在 [KOSZ13、FdV22],中明确说明：他们证明了 A^TWA 是对称对角占优的，因此可以通过求解两个拉普拉斯方程来求解 A^TWA 中的方程。我们得到 $U/\epsilon = M\,\text{poly}(|V|)$，因此 $\log^3(U/\epsilon) = \tilde{O}(\log^3 M)$。