arXiv:2301.12274v1 [cs.DS] 28 Jan 2023

广义超图比率割的割匹配博弈

Nate Veldt 德克萨斯农工大学 计算机科学
与工程系 nveldt@tamu.edu

摘要

超图聚类是分析复杂数据集和系统的一项基础算法原语，这些数据集和系统具有多方交互的特征，例如群组电子邮件对话、共同购买的零售产品群组以及合著数据。本文提出了一种实用的 $O(\log n)$-近似算法，用于求解一大类超图比率割 聚类目标。其中包括涉及广义超图割函数的目标，这类函数允许用户根据每个簇中的节点数量，对割超边施加不同惩罚。我们的方法是图比率割割匹配框架的一种推广，只需在一种特殊的约化图中求解最大 s-t 流问题。与现有超图比率割算法相比，该方法速度显著更快，同时还能求解更一般的问题。通过对多种类型超图进行数值实验，我们表明，该方法能快速找到与最优解相差不大的比率割解。

1 引言

图是对社交网络及其他以成对交互为特征的数据集进行建模的一种常用方式，但人们逐渐认识到，许多复杂的交互系统具有多元关系，无法直接使用图边进行编码。例如，社交媒体上的用户会加入兴趣群组并参与群组活动，网购者会一次购买多种产品，问答论坛中的讨论通常涉及多名参与者，而电子邮件通信通常涉及多名接收者。这些高阶交互可以通过超图进行建模：超图是一组通过称为超边的多元关系相互关联的节点。分析群组交互数据的一种标准基本方法是执行聚类，即识别超图中相互关联的节点群组。这是对研究充分的图聚类问题的直接推广。超图聚类方法已被用于许多不同的数据分析任务，例如，根据共同评价的产品识别不同类型的亚马逊产品 [46], 根据用户浏览行为在 Trivago 上识别相关的度假租赁房源 [13],基于 Yelp 评论对餐厅进行聚类 [32], 以及根据群组讨论数据在在线问答论坛中查找相关帖子 [47]。

（超）图中的“良好”簇，是包含许多内部（超）边、但连接不同簇的（超）边很少的一组节点。形式化这一目标最标准的方法之一，是最小化比率割目标，该目标衡量簇的割 与某种簇大小度量之间的比率。在图的情形下，割只计算跨越簇边界的边数。标准超图割函数同样计算位于边界上的超边数。然而，与图的情形不同，超边划分到两个簇之间的方式不止一种。因此，人们提出了广义超图割函数的概念，它会为不同的超边拆分方式赋予不同的惩罚。下游超图聚类应用的表现可能会强烈依赖于所使用的广义割惩罚类型 [16, 30–32， 46–48]。

前人工作与现有局限。常见的比率割目标包括导通率、归一化割和扩展率，这些目标已针对图和超图分别得到定义（见第2节）。即使在图的情形下，这些问题也是 NP-hard，但可以利用谱方法求得近似解，而这类方法通常带有所谓的 *Cheeger* 不等式保证 [14], ，或者使用各种多商品流和扩张图嵌入方法 [5, 25, 29, 40]。一些用于近似求解图上的比率割的现有技术已扩展到超图，但在这种情况下仍面临许多新挑战。首先，图拉普拉斯算子的许多广义形式都是非线性的，这给计算或近似用于聚类的特征向量和特征值带来了额外挑战 [10, 31, 55]。其次，要为广义超图割函数提供保证也面临诸多挑战，因为这类函数的通用性可能远高于图割函数。

鉴于上述挑战，目前理论算法与实际技术在最小化超图比率割目标方面仍存在巨大差距。许多理论近似算法依赖代价高昂的凸松弛和复杂的舍入技术，而这些技术在实践中并未得到实现 [10, 23, 31, 35, 55]。其中包括内存开销巨大的线性规划和半定规划松弛，其约束数量为 $O(n^3)$ [23, 35] （其中 n 为节点数）；还包括其他半定规划松弛，其约束集非常庞大，涉及次模函数的 Lovász 扩展 [31, 55],；以及舍入技术，其实现细节依赖常数未明确指定的黑盒子程序 [35]。尽管从理论角度看，这些方法颇有意义，但距离实用还很遥远。谱技术 [7, 10, 21, 30, 31, 55] 的另一个缺点是，其类似 Cheeger 的近似保证在最坏情况下为 $O(n)$，即使在图的情形下也是如此。此外，随着最大超边大小增大，谱技术的近似保证往往也会越来越差 [7, 10, 30]。另一方面的一个局限是，许多现有的超图比率割算法仅针对最简单的超图割函数定义，而非广义割 [7, 10, 21, 23, 35, 57]。最后，尽管近期有不少技术适用于广义超图割函数，并且提供了实用实现，但它们无法为全局比率割目标提供理论近似保证：要么是因为它们属于启发式方法 [31] , 要么是因为它们专注于最小化局部化比率割目标，而这类目标会偏向大型超图中的特定区域 [16, 19, 32, 46, 47]。

本文工作：面向超图比率割的实用且通用的近似算法。本文提出了首个用于

表1：超图比率割最小化算法及其保证概要，其中 n 为节点数。不同的*Cheeger*型近似保证存在差异（例如，近似比如何随最大超边大小变化），但在最坏情况下，没有一种优于 $O(n)$。次模基数型割函数比最基本的全有或全无 函数通用得多，并且可以说是实践中最常用的次模超图割函数。此处列出的先前方法是针对特定的节点权重函数开发的，该函数对应于特定的比率割问题（例如扩展率或导通率）；而我们的方法则适用于任意节点权重。最后一列表示该方法首次提出时是否提供了实现。未提供实现的方法面临各种实际挑战，且主要是作为理论结果提出的。

方法	近似保证	超图割函数	主要技术	已实现
CLTZ [10]	Cheeger	全有或全无	SDP 松弛	No
Li-Milenkovic CE [30]	Cheeger	次模	团扩展 + 图谱	Yes
Li-Milenkovic IPM [31]	N/A	次模	逆幂法	Yes
Li-Milenkovic 半定规划 [31]	Cheeger	次模	SDP 松弛	No
IMTY 热方程 [21]	Cheeger	全有或全无	求解超图热方程	No
Louis-Makarychev [35]	$O(\sqrt{\log n})$	全有或全无	SDP 松弛	No
KKTY [23]	$O(\log n)$	全有或全无	线性规划松弛	No
本文	$O(\log n)$	**次模，基于基数的**	割匹配（重复最大流）	**Yes**

具有非平凡近似保证（即最坏情况下优于 $O(n)$），且同样适用于广义超图割的超图比率割。此外，与仅适用于标准超图割函数的近似算法相比，我们的方法具有显著更快的运行时间保证，并提供了实用的实现。具体而言，我们的算法具有最坏情况下的 $O(\log n)$近似保证，且不依赖于最大超边大小。它适用于任意基于基数的次模超图割函数。这是一种通用且灵活的超图割函数，可将许多常用的超图割函数作为特例涵盖在内 [48]。该方法只需在一种特殊类型的约化图中依次求解一系列最大 s-t 流问题，并可借助现成的最大流子程序高效求解。表1 从近似保证、适用的超图割函数类型，以及是否存在实用实现等方面，对比了我们的方法与此前针对超图比率割的最佳算法。

我们的方法将用于图比率割的割匹配 框架推广到超图 [25, 40],，该框架通过交替求解最大流和寻找二分割，将扩展器嵌入图中。尽管在图的情形下，最大流问题和扩展图嵌入已得到充分研究，但在超图中，这些概念更为微妙，也尚未得到充分理解，尤其是在广义超图割的情形下。我们通过提出一种在超图内部嵌入扩展图 的新方法，克服了这些技术挑战。该方法通过在约化有向图中求解最大流问题来实现；这个约化有向图对超图的广义割性质进行了建模。总结如下：

- 我们提出了一种将扩展图嵌入超图的新框架，以获得 NP 难超图比率割目标的下界（引理 3.4）。
- 我们提出了一种 $O(\log n)$-近似算法，适用于采用任意次模基数型割函数和任意正节点权重函数的超图比率割目标，并且只需依次求解一系列图上的最大流问题。
- 我们提供了额外的实用技术，以提高我们的方法在实践中的效率，并获得改进的下界和事后近似保证（定理 5.3）。
- 我们实现了该方法，并表明其可扩展性比以往的近似算法高出数个数量级，还能在来自众多应用领域的大型超图中检测出高质量的比率割。

2 预备知识与相关工作

令 $G = (V, E)$ 表示一个节点集为 V、边集为 E 的图。对于无向图，我们用括号表示边$e = \{u, v\} \in E$；边还可以关联一个非负边权重 $w_G(e) = w_G(u, v) \geq 0$。给定节点子集 $S \subseteq V$，令 $\bar{S} = V \backslash S$表示其补集。 S 的边界定义为

$$\partial_G(S) = \partial_G(\bar{S}) = \{e \in E : |e \cap S| = 1 \text{ and } |e \cap \bar{S}| = 1\}. \quad (1)$$

这是由一端点位于 S、另一端点位于 $\bar{S}$ 的边组成的集合。 G 中 S 的割 是跨越该边界的边的权重之和：

$$\mathbf{cut}_G(S) = \mathbf{cut}_G(\bar{S}) = \sum_{e \in \partial_G(S)} w_G(e). \quad (2)$$

给定正的节点权重函数 $\pi : V \to \mathbb{R}_{>0}$，对于集合 S，相对于 π 的比率割目标记为

$$\phi_G(S, \pi) = \frac{\mathbf{cut}_G(S)}{\min\{\pi(S), \pi(\bar{S})\}}, \quad (3)$$

其中 $\pi(S) = \sum_{u \in S} \pi(u)$。这涵盖了众所周知的特殊情况：导通率 （当 $\pi(u)$ 等于节点度 d_u 时）和扩展率 （当 $\pi(u) = 1$ 时）。我们将 $\phi_G(S, \pi)$ 称为 π-扩展率。图 G 的 π-扩展率为

$$\phi_{G,\pi} = \min_{S \subset V} \phi_G(S, \pi). \quad (4)$$

当上下文明确时，我们会从下标中省略 G 和 π。如果一个图的扩展率至少为某个常数，则称其为扩展器 图。形式化地，对于给定的节点权重函数

π，如果存在常数$c > 0$，使得对于每个 $S \subseteq V$， $\phi(S) > c$，则称 $G = (V, E)$为 π-扩展器。

有向图。我们的结果依赖于有向图中割的概念。如果 $G = (V, E)$ 是有向的，我们使用圆括号表示边方向，即 $(u, v) \in E$表示一条从节点 u指向节点 v的边。对于集合 $S \subseteq V$，有向割函数为

$$\mathbf{cut}_G(S) = \sum_{(u,v)\in E:\, u\in S, v\in \bar{S}} w_G(u, v).$$

有向图中的扩展率通过使用（3）中的有向割函数来定义。我们可以将无向图 G 视为有向图，方法是将每条无向边替换为两条有向边，在这种情况下，两种 π-扩展率的概念一致。

2.1 图流

流函数 f：有向图 $G = (V, E)$ 上的 $E \to \mathbb{R}_{\geq 0}$为每条边 $(i, j) \in E$ 分配一个非负流值 $f_{ij} \geq 0$。如果对于每个 $(i, j) \in E$ 都有 $f_{ij} \leq w_{G\,ij}$，则满足容量约束。一般而言，f 的拥塞是容量违反的最大值：

$$\mathbf{congestion}(f) = \max_{(i,j)\in E} \frac{f_{ij}}{w_G(i, j)}. \tag{5}$$

我们称 f在节点 $v \in V$处满足流约束，当流入 v的流量等于流出 v的流量：

$$\sum_{i:\,(i,v)\in E} f_{iv} = \sum_{j:\,(v,j)\in E} f_{vj}. \tag{6}$$

如果流入 v的流量大于流出 v的流量，那么 v就是一个盈余节点。如果流出 v的流量大于流入 v的流量，那么v就是一个亏空节点。给定两个流函数 $f(1)$和 $f(2)$，它们的和 $f' = f(1) + f(2)$是通过对每个$(i, j) \in E$定义 $f'_{ij} = f(1)_{ij} + f(2)_{ij}$而得到的流。该流满足拥塞$(f')$ ≤拥塞$(f(1))$+拥塞$(f(2))$。如果 $f(1)$和 $f(2)$都在节点 v处满足流约束，那么 f'也满足流约束。我们将考虑两种特殊类型的流。

定义 *2.1*（s-t流）。给定$\{s, t\} \subseteq V$，一个 s-t流定义在 G上，且在每个 $v \in V - \{s, t\}$上都满足流约束。我们称 f将流从 s输送到 t，其流值为

$$|f| = \sum_{i:\,(s,i)\in E} f_{si} - \sum_{j:\,(j,s)\in E} f_{js}. \tag{7}$$

定义 *2.2*（多商品流）。G中的多商品流问题由需求对集合 D（u，v）$\in V \times V$及其对应的权重 $w_{uv} \geq 0$定义。若流 f可以写成 $f = \sum_{(u,v)\in\mathcal{D}} f_{(uv)}$，其中 $f_{(uv)}$是一个 u-v流，将 w_{uv}单位的流量从 u路由至 v，则称其为 D 上的可行多商品流。

我们将使用以下标准的流分解引理。

引理 2.3.（*[1]*中的定理 *3.5* 或 *[53]*中的引理 *2.20*。）设 f是图 $G = (V, E)$上的流函数，该图有 n个节点和 m条边。流可分解为 $f = \sum_{i=1}^{\ell} f_i$，其中 $\ell \leq m + n$；对于每个 i，在 f_i中流量为正的边要么构成一条从亏空节点到盈余节点的简单有向路径，要么构成一个环。

2.2 一般超图的割与扩展率

超图 $\mathrm{H} = (V, \mathrm{E})$ 是图的一种推广，其中 V表示节点集，E 是由超边组成的集合，其中每个 $e \in \mathrm{E}$ 都是 V中任意大小的节点子集。超边 $e \in \mathrm{E}$ 的阶是其包含的节点数，即 $|e|$。节点 v的节点度记为 d_v。$S \subseteq V$的边界是跨越二分划分 $\{S, \bar{S}\}$ 的超边集合：

$$\partial_{\mathcal{H}}(S) = \partial_{\mathcal{H}}(\bar{S}) = \{e \in \mathcal{E} : |e \cap S| > 0 \text{ and } |e \cap \bar{S}| > 0\}. \tag{8}$$

如果每条超边 $e \in \mathrm{E}$ 都具有非负标量权重 $w_{\mathcal{H}}(e)$，则标准的全有或全无超图割函数定义为

$$\mathbf{cut}_{\mathcal{H}}(S) = \textstyle\sum_{e\in\partial_{\mathcal{H}}(S)} w_{\mathcal{H}}(e). \tag{9}$$

在许多情况下，对以不同方式分离超边节点的情形赋予不同的割惩罚是很有用的。这促成了广义超图割函数这一概念 [16, 30–32, 46–48, 55, 58]。形式上，每条超边 $e \in \mathcal{E}$ 都与一个分裂函数$\mathbf{w}_e$：$A \subseteq e \to \mathbb{R}$ 相关联，该函数为分离超边节点的每一种方式分配一个惩罚。于是，广义超图割定义为

$$\mathbf{cut}_{\mathcal{H}}(S) = \sum_{e\in\partial_{\mathcal{H}}(S)} \mathbf{w}_e(S \cap e). \tag{10}$$

按照近期工作 [31, 46–48]，，我们关注满足以下性质的分裂函数，对所有 A，$B \subseteq e$均成立：

(nonnegative) $\mathbf{w}_e(A) \geq 0$
(uncut-ignoring) $\mathbf{w}_e(\emptyset) = 0$
(symmetric) $\mathbf{w}_e(A) = \mathbf{w}(e\backslash A)$
(submodular) $\mathbf{w}_e(A) + \mathbf{w}_e(B) \geq \mathbf{w}_e(A \cap B) + \mathbf{w}_e(A \cup B)$
(cardinality-based) $\mathbf{w}_e(A) = \mathbf{w}_e(B)$ if $|A| = |B|$.

当满足这些条件时，**割**$\mathcal{H}$ 是一个基于基数的次模超图割函数。为简化术语，我们将其简称为广义超图割函数。

广义超图割扩展率。现在，我们可以形式化地定义我们希望在超图 $\mathrm{H} = (V, \mathrm{E})$ 上求解的广义比率割问题。给定一个正的节点权重函数 π：$V \to \mathbb{R}_{>0}$ 以及一个广义超图割函数**割**$\mathcal{H}$，集合 $S \subseteq V$的超图 π-扩展率定义为。

$$\phi_{\mathcal{H}}(S, \pi) = \frac{\mathbf{cut}_{\mathcal{H}}(S)}{\min\{\pi(S), \pi(\bar{S})\}}. \tag{11}$$

问题 1。超图 π-扩展率问题是求解

$$\phi_{\mathcal{H},\pi} = \min_{S\subseteq V} \frac{\boldsymbol{cut}_{\mathcal{H}}(S)}{\min\{\pi(S), \pi(\bar{S})\}}$$

其中割$\mathcal{H}$ 是广义超图割函数，且 π 是正的节点权重函数。

我们的目标是为这一目标开发一种近似算法，使其适用于基于基数的超图次模割函数和任意节点权重函数。在运行时间分析中，我们假设当超边割惩罚和节点权重均被缩放至至少为 1 时，所有其他节点权重和割惩罚关于 $|V|$ 和 $|\mathrm{E}|$均为多项式有界的。

2.3 相关工作

扩展率（$\pi(v) = 1$）和导通率（$\pi(v) = d_v$）是研究最为广泛的两种图比率割目标。多年来，图扩展和导通率的最佳近似为 $O(\log n)$，其基础是求解一个多商品流问题 [29]，该问题也可视为一种线性规划松弛。当前最佳近似因子为 $O(\sqrt{\log n})$，其基础是一种半定规划松弛 [5]。Khandekar 等人 [25] 随后引入了割匹配框架，基于更快的最大 s-t 流计算，而非多商品流，提供了 $O(\log^2 n)$ 的近似。之后，人们利用同一框架设计出了改进的 $O(\log n)$ 近似算法 [38, 40]。

此前已有许多关于超图比率割目标 (11) 的变体 [30–32, 34, 48, 55, 57],，这些变体考虑了不同类型的割函数和节点权重函数 π。基于非线性超图拉普拉斯算子的特征分解，人们为这类目标提出了多种 Cheeger 型近似保证 [10, 31, 55],，但在最坏情况下，这些方法只能达到 $O(n)$ 的近似因子。其他方法依赖于超图的团扩展 [7, 30, 57],，其在最坏情况下的近似因子同样为 $O(n)$，且随着最大超边大小增加，计算规模也会变得很大。$O(\log n)$ 多商品流算法 [29] 和基于 SDP 的 $O(\sqrt{\log n})$ 扩展率近似 [5] 均已推广到超图情形 [23, 35]。然而，这些方法仅适用于标准的全有或全无超图割函数 (9)，并且只关注扩展度节点权重选择 ($\pi(v) = 1$) [35] 或导通率节点权重选择 ($\pi(v) = d_v$) [23]。最后，已有大量关于图 [3, 4, 26, 36, 41, 50, 51] 和超图 [16, 19, 32, 46, 47], 中局部偏置比率割目标最小化的研究，但这些研究并未为全局比率割目标提供保证。

超图割匹配的同期工作。在独立且同期开展的工作中，Ameranis 等人 [2] 开发了一种$O(\log n)$-近似算法，用于求解具有广义割函数的超图比率割问题；该算法同样通过扩展 Khandekar 等人 [25]的割匹配框架而得到。从高层次来看，他们获得 $O(\log n)$-近似的方法与我们的方法类似，因为其依赖于将超图归约为图，然后求解
最大流问题，以便在超图中嵌入一个多商品流问题。在更技术性的层面上，两种策略在所采用的图约简技术以及需要求解的最大流问题方面有所不同。Ameranis 等人 [2] 采用超图的二部图扩展，将每条超边替换为一个新顶点，并在原始顶点与超边顶点之间添加无向边。随后，他们在 无向 流图中求解带有顶点容量约束的最大流问题，以满足他们所定义的超图流概念。相比之下，我们在将超图归约为图时使用增广割保持器（见定义 3.1），然后反复求解不带有顶点容量约束的有向最大流问题，不带有 顶点容量约束。

从理论角度来看，Ameraniset al. 设计的算法 [2]适用于一种新定义的单调次模割函数；与一般次模割函数相比，这种函数的限制更多

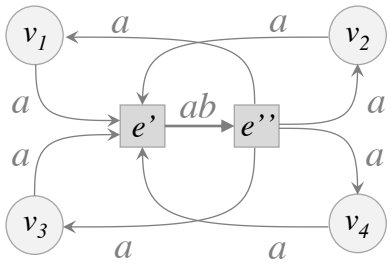


图 1：由权重 a 和 b 参数化的四节点超边 $e = \{v_1, v_2, v_3, v_4\}$ 的 CB 小工具。

但它推广了我们所考虑的基于基数的次模情形。尽管他们的方法在实际实现方面展现出潜力，作者却侧重于理论贡献，因此既未实现该方法，也未展示数值结果。相比之下，我们工作的主要贡献之一是为超图割匹配博弈提供实用实现，并与其他超图比率割算法进行详细比较。将这些策略结合起来应用于超图割匹配，以进一步提升理论成果和实现效果，是未来研究的一个有前景的方向。

3 超图扩展器嵌入

扩展图嵌入是为无向图中的扩展率建立下界的常用技术 [5, 25, 29, 42]。我们将这一基本方法加以推广，以通过将一个扩张图嵌入一种特殊类型的有向图（该图对广义超图割函数进行建模），来制定为超图比率割目标建立下界的策略。

3.1 超图割保持器

许多超图聚类方法依赖于将超图约简为图，然后应用现有的图技术 [7, 20, 30, 48, 54]。我们采用一种已有的精确定义的超图约简概念，以对广义超图割 [8, 48] 建模。

定义 *3.1*（增广割保持器）。设 $\mathrm{H} = (V, \mathrm{E})$ 是一个具有广义割函数的超图。若有向图$G(\mathrm{H}) = (\hat{V}, \hat{E})$ 满足 $\hat{V} = V \cup \mathrm{A}$（其中 A 是辅助节点集），且 $G(\mathrm{H})$ 按如下意义保持 H 中的割，则称其为 H 的增广割保持器：对于每个 $S \subseteq V$：

$$\mathbf{cut}_{\mathcal{H}}(S) = \min_{U \subseteq \mathcal{A}} \mathbf{cut}_{G(\mathcal{H})}(S \cup U). \tag{12}$$

割保持性质12）本质上表明，对于任意固定的 $S \subseteq V$，我们可以将$\mathcal{A}$ 中的节点划分到割的两侧，从而使 $G(\mathcal{H})$中的总体（有向）割惩罚最小化。构造增广割保持器 $G(\mathcal{H})$ 的方法不止一种，适用于基于基数的次模超图割函数 [27, 47, 48]。一般策略是将每条超 $e \in$ 边 E 替换为一个小型构件，其中包含有向加权边和辅助节点，以此对该超边的分裂函数进行建模。具体而言，我们使用基于对称基数型小工具（CB 小工具）的归约 [47, 48]（图1）。对于超边 $e \in \mathcal{E}$，若其包含 k 个节点，则一个 CB 小工具会引入两个新的

节点 $e_\prime$ 和 $e_{\prime\prime}$，并将它们加入增广节点集 A。对于每个 $v \in e$，该构件包含两条有向边（v，$e_\prime$）和（$e_{\prime\prime} v$），两者的权重均为 a；此外还包含一条有向边（$e_\prime e_{\prime\prime}$），其权重为 ab。其中，a 和 b 是该 CB 小工具的非负参数。

图 2(b) 展示了一个小型超图的增广割保持器示意图，其中每条超边都被替换为一个 CB- 小工具，并省略了边权重。每条超边也可以替换为由不同权重的 CB 小工具组成的组合，以便表示更复杂的超图割函数。通过为每个小工具 $\lfloor |e|/2 \rfloor$ CB 小工具仔细选择参数 (a, b)，可以使用 CB 小工具的组合表示任意基于基数的次模分裂函数 [48]。如果只需近似表示超图割函数，则可以改用增广稀疏化器 [8]，以更少的 CB 小工具近似该割函数。

3.2 有向图中的扩展图嵌入

设 $G = (V, E_G)$ 为一张有向图，每条边 $(u, v) \in E_G$ 的边权重为 $w_G(u, v)$；设 $H = (V, E_H)$ 为一张具有相同节点集合的无向图，其边 $\{i, j\} \in E_H$ 的权重为 $w_H(i, j) \geq 0$。对于每个集合 $S \subseteq V$，令 $\mathrm{D}(S) = \{(u, v) \in S \times \bar{S} : \{u, v\} \in E_H\}$ 表示在 H 中共享一条边的有向节点对集合。

定义 *3.2*（嵌入 H 于 G）。若对于每个二分 $\{S, \bar{S}\}$，图 H 可以以拥塞 γ 嵌入到 G 中，则：

- 对于每个 $(u, v) \in \mathrm{D}(S)$，都存在一个 u-v 流函数 $f_{(uv)}$，它通过 G 中的边，将 $w_H(u, v)$ 单位的流从 u 路由至 v。
- 流 $f = \sum_{(u,v) \in \mathcal{D}(S)} f_{(uv)}$ 可以通过 G 进行路由，且拥塞至多为 γ；也就是说，对于每个 $(i, j) \in E_G$，

$$\sum_{(u,v) \in \mathcal{D}(S)} f_{ij}^{(uv)} \leq \gamma w_G(i, j). \tag{13}$$

引理 3.3。若 H 以拥塞 γ 嵌入到 G 中，则对于任意节点权重函数 π, $\phi_{G,\pi} \geq \frac{1}{\gamma} \phi_{H,\pi_o}$

证明。设 $S \subseteq V$ π $\bar{S}$ 是满足 $(\,)$ $(\,)$ 的任意节点集。对于每一对 $u, v \in S$ $(\,)$ $\mathrm{D}(\,)$，都存在一个流 $f_{(\,)}^{uv}$，它沿着 E_G 中的边传输，流值为 $w_H(u, v)$；这些流可以同时在 E_G 中进行路由，且拥塞为 γ。所有这些流都必须经过 E_G 中的割边，因为每一对这样的节点 (u, v) 都跨越二分划分 $\{S, \bar{S}\}$。因此，我们有

$$\mathbf{cut}_H(S) = \sum_{(u,v) \in \mathcal{D}(S)} w_H(u, v) \leq \sum_{(u,v) \in \mathcal{D}(S)} \sum_{(i,j) \in \partial_G S} f_{ij}^{(uv)}$$
$$\leq \gamma \sum_{(i,j) \in \partial_G S} w_G(i, j) = \gamma \mathbf{cut}_G(S).$$

因此，$\phi_G(S) = \text{割}_G(S)\,\pi(S) \geq \frac{1}{\gamma} \text{割}_H(S)\,\pi(S) = \frac{1}{\gamma} \phi_H(S)$。

□

引理 3.3 的直接推论是：如果 H 是一个扩展器，那么 G 的扩展率为 $\Omega(\frac{1}{\gamma})$。定义 3.2 与近似图比率割时通常使用的其他嵌入概念略有不同。这是因为，将这些扩展器嵌入技术推广到超图场景时会产生一些细微差异，尤其是在考虑广义超图割函数时。具体而言，近似一个 无向 超图中的比率割，需要考虑一个有向图中的流，并且在考虑任意集合 $S \subseteq V$ 以及一对节点 $\{u, v\}$ 位于划分 $\{S, \bar{S}\}$两侧时，必须谨慎处理流路径的方向。定义 3.2 给出了扩展器嵌入的形式化概念；当我们在第 4 节处理有向流问题时，这一概念尤其方便。

3.3 超图扩展图嵌入

结合现有的超图割保持器概念 [47, 48] 与定义 3.2，可为界定广义超图割的扩展率提供一种新策略。

引理 3.4。设 $\mathcal{H} = (V, \mathcal{E})$ 是一个具有广义 $\hat{}$ $\hat{}$超图割函数 割$_{\mathcal{H}}$ 的超图，并设 $G(\mathcal{H}) = (V = V \cup \mathcal{A}\ E)$ 是 $\mathcal{H}$ 的增广割保持器。如果图 $H = (V, E_H)$ 可以嵌入到 $G(\mathcal{H})$ 中，且拥塞为 γ，那么对于每个节点权重函数 π, $\phi_{\mathcal{H}\,\pi} \geq \frac{1}{\gamma} \phi_{H,\pi}$。

证明。设 $S \subseteq V$ π $\bar{S}$ 为满足 $(\,)$ $(\,)$ 的任意节点集合。定义 $\hat{}$，其中 A 的选取方式使得 割$_{\mathcal{H}}(S)$ = 割$_{G(\mathcal{H})}(\hat{}S)$。回忆一下，$\mathrm{D}(S)$表示由满足 $u \in S$和 $v \in V - S$的 $\{u, v\} \in E_H$ 构成的有序对 (u, v) 集合。由于 H以拥塞 γ嵌入到 $G(\mathrm{H})$ 中，对于每一对 $(u, v) \in \mathrm{D}(S)$，我们都从 u到 v 路由了 $w_H(u, v)$ 流。将所有穿过该割的流加总，得到：

$$\mathbf{cut}_H(S) = \sum_{(u,v) \in \mathcal{D}(S)} w_H(u, v) \leq \gamma \sum_{(i,j) \in \partial_{G(\mathcal{H})} \hat{S}} w_{G(\mathcal{H})}(i, j)$$
$$= \gamma \mathbf{cut}_{G(\mathcal{H})}(\hat{S}) = \gamma \mathbf{cut}_{\mathcal{H}}(S).$$

Thus, we have $\phi_{\mathcal{H}}(S) = \frac{\mathbf{cut}_{\mathcal{H}}(S)}{\pi(S)} \geq \frac{1}{\gamma} \frac{\mathbf{cut}_H(S)}{\pi(S)} = \frac{1}{\gamma} \phi_H(S)$. □

该引理假设节点权重函数 π 固定，因此它并未直接关联 H 中的导通率与H 中的导通率（因为导通率取决于节点度，而节点度在 H 与 H 之间未必保持不变）。不过，正如我们将看到的，这仍然为求出 H 中任意比率割目标（包括导通率）的下界提供了重要的一步。

4 超图流嵌入

为了应用引理 3.4，我们希望将一个扩展器以较小的拥塞嵌入 $G(\mathrm{H})$，然后找到一个集合 S，使其扩展率与该引理所蕴含的下界相差不大。为此，本节将展示如何把一种特殊类型的二分图嵌入 H，且其拥塞与一个扩展率较小的集合相关。图2以示意图概述了本节中的流嵌入技术。给定节点集 V的一个划分$\{R, \bar{R}\}$，我们将设计一个过程：它接收参数 $\alpha > 0$，并使用一次最大 s-t 流计算，从而（1）返回一个集合 $S \subseteq V$，使得$\phi \mathrm{H}\,(S, \pi) < \alpha$；或者（2）生成一个连接 R与 $\bar{R}$的二分图，该图可以以 $1/\alpha$ 的拥塞嵌入 $G(\mathrm{H})$。在第 5 节中，我们将展示如何组合这些二分图，从而将一个扩展器嵌入 $G(\mathrm{H})$。

4.1 辅助图中的最大流

固定 $\alpha > 0$ 并选取一个划分 $\{R, \bar{R}\}$ 满足 $\pi(R) \leq \pi(\bar{R})$，并设置$\eta = \pi(R)/\pi(\bar{R})$。我们将在辅助图 s-t 流问题上求解最大 $G(\mathcal{H}\ R,\ \alpha)$（见图2(c)），其构造如下：

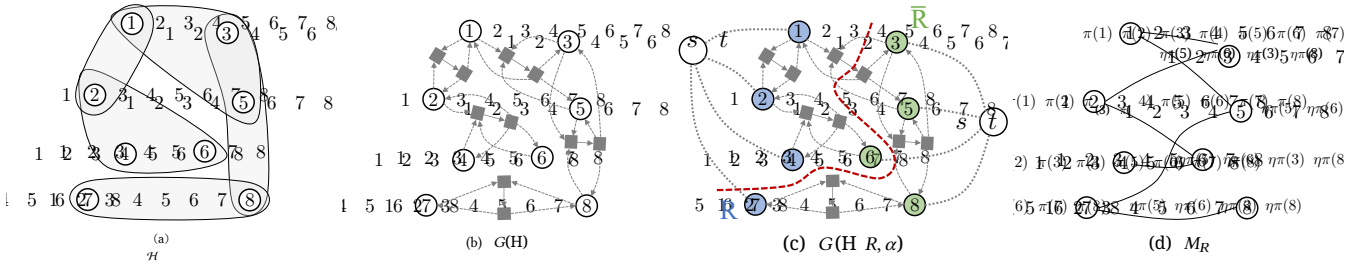


图2：（a）一个包含 8 个节点和 5 条超边的示例超图 H =(V, E)。（b）增广割保持器 G(H)（定义 3.1）利用带有增广节点集的有向图中的割性质，对 H 的广义超图割函数进行建模。（c）对于给定的划分 {R, R̄} 和参数 α，辅助图 G(H R, α) 包含源节点 s 和汇节点 t 以及额外的加权边。在该图中求解最大流问题，将产生一个小割（红色虚线）和一个满足 π- 扩展率小于 α 的节点集 S ⊆ V，或者产生一个 π-正则二分图 M_R（如（d）所示），该二分图可以嵌入 G(H)，且拥塞为 1/α。针对许多（经过仔细选择的）集合 R 重复求解最大流问题，可以生成许多 π-正则二分图。这些二分图的并集是一个扩张图，可以嵌入 G(H)，其拥塞上界由沿途找到的最佳集合 S* 的 π-扩展率决定。这将得到一个 O(log n) 近似。

- 为 H 构造一个 CB-构件割保持器 $G(\mathrm{H})$，并将所有边权重按 $\frac{1}{\alpha}$ 进行缩放。- 添加一个额外的源节点 s 和一个汇节点 t。
- 对于每个 $r \in R$，添加一条有向边（s，r），其权重为 $\pi(r)$。
- 对于每个 $v \in \bar{R}$，添加一条有向边（v，t），其权重为 $\eta\pi(v)$。

我们将利用最大 s-t 流问题的解，或者找到一个扩展率较小的集合 S（通过考虑其对偶的最小s-t 割问题），或者找到一个连接 R 和 $\bar{R}$ 的可嵌入二分图，其拥塞为 $\frac{1}{\alpha}$（通过考虑流分解）。这沿用了图扩展的割匹配博弈中采用的相同策略 [25]，，不过对于广义超图割函数，构造和证明要复杂得多。

寻找一个较小的 π-扩展率集合。 G 中的最小 s-t 割问题 $(\mathrm{H}\ R,\ \alpha)$ 等价于求解超图 H 上的以下优化问题：

$$\underset{S \subseteq V}{\text{minimize}}\ \frac{1}{\alpha}\mathbf{cut}_{\mathcal{H}}(S) + \pi(R \cap \bar{S}) + \eta\pi(\bar{R} \cap S). \tag{14}$$

若设 $S = \emptyset$，这对应于在 $G(\mathrm{H}\ R,\ \alpha)$ 中将节点 s 与所有其他节点分开，这是一个有效的 s-t割，其值为 $\pi(R)$。如果我们能在 $G(\mathrm{H}\ R,\ \alpha)$ 中找到一个值严格小于 $\pi(R)$ 的 s-t割，它将给出一个在 H 中扩展率较小的集合 S。

引理 4.1。 如果 $G(\mathcal{H}\ R, \alpha)$ 中最小的 s-t割值小于 $\pi(R)$，则使目标函数 (14) 最小化的集合 S 满足 $\phi_{\mathcal{H}}(S, \pi) < \alpha$。

证明。 $G(\mathrm{H}\ R,\ \alpha)$ 中最小的 s-t割集是某些节点集合 $\{s\} \cup S \cup \mathrm{A}_S$，其中 $S \subseteq V$是原超图 H 中的节点集合，而 A_S 是割保持器 $G(\mathrm{H})$ 中辅助节点的子集，其设计目的是使 $G(\mathrm{H})$ 中的割函数与 H 中的割函数相匹配。如果 $G(\mathrm{H}\ R,\ \alpha)$ 中最小的 s-t 割值小于 $\pi(R)$，这意味着对于集合 S：

$$\begin{aligned}
&\frac{1}{\alpha}\mathbf{cut}_{\mathcal{H}}(S) + \pi(R \cap \bar{S}) + \eta\pi(\bar{R} \cap S) < \pi(R) \\
\Longrightarrow\ &\mathbf{cut}_{\mathcal{H}}(S) + \alpha\pi(R \cap \bar{S}) + \alpha\eta\pi(\bar{R} \cap S) < \alpha\pi(R) \\
\Longrightarrow\ &\mathbf{cut}_{\mathcal{H}}(S) < \alpha\pi(R \cap S) - \alpha\eta\pi(\bar{R} \cap S) \\
\Longrightarrow\ &\frac{\mathbf{cut}_{\mathcal{H}}(S)}{\pi(R \cap S) - \eta\pi(\bar{R} \cap S)} < \alpha.
\end{aligned}$$

只要我们能够证明 $\min\{\pi(R \cap S) - \eta\pi(\bar{R} \cap S), \pi(S) - \pi(R \cap S) - \eta\pi(\bar{R} \cap S)\} \geq 0$，该结果就成立。首先注意到 $\pi(R \cap S) - \eta\pi(\bar{R} \cap S) \geq 0$。此外，我们有

$$\begin{aligned}
\pi(R \cap S) - \eta\pi(\bar{R} \cap \bar{S}) &= \pi(R \cap S) + \eta\pi(\bar{R} \cap \bar{S}) - \eta\pi(\bar{R}) \\
&= \eta\pi(\bar{R} \cap \bar{S}) - \pi(R \cap \bar{S}) \leq \pi(\bar{S}).
\end{aligned}$$

□

用于求解与（14）密切相关目标函数的最大流子程序，已经被用于最小化超图中的局部比率割目标函数 [46]。这将图中关于局部比率割的早期工作推广了 [4, 41, 50, 51]。本研究的不同之处在于，我们将使用最大流解来近似全局比率割目标函数。实现这一点的关键步骤之一，是说明当引理4.1 不适用时，如何将二分图嵌入 $G(\mathcal{H})$ 中。

嵌入二分图。如果 $G(\mathrm{H}\ R, \alpha)$ 中的最小 s-t割为 $\pi(R)$，则 $G(\mathrm{H}\ R, \alpha)$ 中的最大 s-t流解会饱和所有与 s或 t相连的边。于是，我们可以定义一个位于 R与 $\bar{R}$之间的二分图 M_R，并将其以拥塞 $\frac{1}{\alpha}$嵌入 $G(\mathrm{H})$ 中。令 f为 $G(\mathrm{H}\ R, \alpha)$ 上值为 $|f| = \pi(R)$ 的最大 s-t 流。令 $n = |V|$，定义矩阵 $\mathrm{M}_R \in \mathbb{R}^{n \times n}$，并将其初始化为全零矩阵。利用引理 2.3，我们可以将流f分解为 $f = \sum_{i=1}^{\ell} f_i$，其中对于每个 i，在 f_i中具有正流量的边构成一个环或一条简单的 s-t路径。就我们的目的而言，可以忽略这些环，专注于 s-t路径。一条 s-t流路径 f_i总是先将 $|f_i|$ 单位的流从 s发送到某个节点 $r \in R$，最终通过一条边 (v, t) 发送相同量的流，其中 $v \in \bar{R}$。对于每条此类流路径，我们执行更新 $\mathrm{M}_R(r, v) \leftarrow \mathrm{M}_R(r, v) + |f_i|$。遍历所有 ℓ 流路径后，我们将 M_R 定义为邻接矩阵为 M_R的二分图（见图 2(d)）。$G(\mathrm{H}\ R, \alpha)$ 的构造，以及 f饱和所有与 s和 t相连的边这一事实，意味着 M_R具有以下度性质：

- 对于每个 $r \in R$，M_R中 r的加权度为 $\pi(r)$。- 对于每个 $u \in \bar{R}$，M_R中 u的加权度为 $\eta\pi(u)$。

沿用之前的术语 [39]，，我们将满足这些性质的二分图称为 a π-正则二分图。为了将其与之前用于图扩展的割-匹配 博弈联系起来，请注意，如果 $|R| = |\bar{R}|$且对所有 $u \in V$ 都有 $\pi(u) = 1$，那么 M_R 将是一个分数的

匹配且 $\mathbf{M}_R$ 将是一个双随机矩阵。还值得注意的是，如果 $\pi(R) = \pi(\bar{R})$，那么 $u \in V$ 中每个节点的加权度都将等于 $\pi(u)$，从而建立 M_R 中的导通率与 H 的 π-扩展率之间的对应关系。在实践中，$\pi(R)$ 不会始终恰好等于 $\pi(\bar{R})$，但我们通常会选择大小大致均衡的划分。

证明拥塞上界。当 $|f| = \pi(R)$ 时，$G(\mathrm{H}\ R, \alpha)$ 中的最大 s-t 流可以视为一种经由 $G(\mathrm{H})$ 的 $\frac{1}{\alpha}$ 倍缩放副本进行路由的有向多商品流。然而，为了确认 M_R 可以以 $\frac{1}{\alpha}$ 的拥塞度嵌入 $G(\mathrm{H})$，我们必须解决一个微妙的问题。定义 3.2 要求，对于每个二分 $\{S, \bar{S}\}$，如果 $u \in S$ 且 $v \in \bar{S}$，就必须能够将 $\mathbf{M}_R(u, v)$ 单位的流从 u 路由至 v。然而，f 只会在有向图 $G(\mathrm{H}\ R, \alpha)$ 中将流从 R 路由至 $\bar{R}$。在无向图上的割匹配博弈中不会出现这个问题，因为每条无向边 $\{u, v\}$ 都可以视为一对有向边 (u, v) 和 (v, u)，因此很容易同时沿两个方向发送流。然而，在有向图 $G(\mathrm{H}\ R, \alpha)$ 中，对于给定的二分划分 $\{S, \bar{S}\}$，流 f 可能会将流从节点 $r \in R \cap \bar{S}$ 发送至节点 $u \in \bar{R} \cap S$，这并不能直接满足定义 3.2 中的要求。下面的引理证实了我们可以解决这个问题。其证明依赖于仔细考察 $G(\mathrm{H})$ 中的边结构，并说明如何在必要时使用隐式流反转过程，以满足定义 3.2。附录中给出了证明。

引理 4.2。如果 $|f| = \pi(R)$，图 M_R 可以嵌入 $G(\mathcal{H})$，其拥塞为 $\frac{1}{\alpha}$，具体含义参见定义 3.2。

4.2 流嵌入算法

我们结合上述两种方法，提出 HyperCutOrEmbed（算法 1），对于任意输入划分 $\{R, \bar{R}\}$，在 $\pi(R) \le \pi(\bar{R})$ 的条件下，同时获得一个良好的割和一个可嵌入二分图。我们假设超图 H 是连通的，并且对超图权重进行缩放，使得割任意超边时的最小惩罚为 1。这意味着最小 π-扩展率的下界为 $2/\pi(V)$；当存在集合 S，满足 $\pi(S) = \pi(V)/2$ 且 $\mathrm{cut}_{\mathrm{H}}(S) = 1$ 时，该下界可以达到。我们还假设对节点权重进行缩放，使得对于每个 $v \in V$，$\pi(v) \ge 1$。HyperCutOrEmbed 反复求解最大 s-t 流问题，以寻找二分图或具有有界 π-扩展率的割。该算法使用最大 s-t 流和最小 s-t 割的黑盒子程序，以及过程 FlowEmbed；该过程将流分解为一个 π-正则二分图，具体如第 4.1 节所述。

算法 1 HyperCutOrEmbed(H R, π)

输入： H $= (V, E)$，节点权重 π，二分 $\{R, \bar{R}\}$

输出： M_R，拥塞为 $1/\alpha$；S，满足 $\phi_{\mathcal{H}}(S, \pi) \le 2\alpha$

设置 $\alpha = 2/\pi(V)$，NoCutFound = 为 true

当 NoCutFound**时**

  $f =$ MaxST流$(G(\mathrm{H}\ R, \alpha))$.

  如果 $|f| = \pi(R)$ **则**

    $M_R \leftarrow$ FlowEmbed$(G(\mathrm{H})\ R, f)$; $\alpha \leftarrow 2\alpha$

  else

    $S =$ MinSTcut$(G(\mathrm{H}\ R, \alpha)\ f)$

    NoCutFound =false

返回 M_R、S、α

定理 4.3。 *HyperCutOrEmbed* 返回一个 π-正则二分图 M_R，该图可以嵌入到 $G(\mathrm{H})$ 中，拥塞为 $1/\alpha$，以及一个集合 S，满足 $\phi_{\mathcal{H}}(S, \pi) < 2\alpha$，其中 $\alpha \ge 2/\pi(V)$。该算法在 $O(\log |\mathcal{E}| + \log U + \log \pi(V))$ 次迭代后终止，其中 U 是最大的超边割惩罚。

证明。在每次迭代中，算法计算 s-t 流值为 $G(\mathcal{H}\ R, \alpha)$ 的最大流。根据引理 4.2，如果流值为 $\pi(R)$，则我们可以返回一个 π-正则二分图 M_R，该图可以以拥塞 $1/\alpha$ 嵌入。否则，引理 4.1 保证最小 s-t 割集的扩展率小于 α。由于不可能找到一个具有 π-扩展率小于最小值的集合，因此算法保证在第一次迭代中返回一个二分图。

值 $2/\pi(V)$。如果算法找到一个集合 S，其满足 $\phi_{\mathcal{H}}(S, \pi) < \alpha$，然后终止，这意味着它在前一次迭代中找到了一个二分图 M_R，该图可以以拥塞 $\alpha/2$ 嵌入。

对于每个 $\alpha > U|\mathrm{E}|$，在 $G(\mathrm{H}\ R, \alpha)$ 中寻找一个最大 s-t 流，保证会返回一个割集 S，满足 $\phi_{\mathcal{H}}(S, \pi) < \alpha$。这是因为**割**$_{\mathcal{H}}(R) \le U|\mathrm{E}|$，因此节点集 $\{s \cup R\}$ 是辅助图中的一个 s-t 割集，其割值为

$$\frac{\mathbf{cut}_{\mathcal{H}}(R)}{\alpha} \le \frac{U|\mathcal{E}|}{\alpha} < 1 \le \pi(R).$$

由于 HyperCutOrEmbed 从 $\alpha = 2/\pi(V)$ 开始，并在每次迭代时将 α 加倍，因此最多经过 $O(\log U|\mathrm{E}|\pi(V))$ 次迭代后，就会返回一个割集而非匹配。□

对于无权重的全有或全无割函数，将 HyperCutOrEmbed 应用于扩张率或导通率时，会在 $O(\log |V| + \log |\mathrm{E}|)$ 轮内终止。在更一般的情形下，迭代次数上界仅对超图规模和权重呈对数依赖。

5 超图比率割算法

HyperCutOrEmbed 找到一个节点集，使其 π-扩展率与一个可嵌入 $\mathcal{H}$ 中的图相关。直接应用引理3.4 并不能为 π-扩展率在 $\mathcal{H}$ 中提供有用的下界或近似算法，因为该二分图本身并不具有较大的 π-扩展率。本节将介绍如何将 Hy-perCutOrEmbed与构建扩张图的现有策略相结合，从而设计出一种针对超图 π-扩展率的近似算法。

5.1 扩张图构建子程序

无向图 $G = (V, E)$ 中用于扩展的标准割匹配过程可以描述为割玩家与匹配玩家之间的双人博弈。在第 i 次迭代开始时，割玩家生成一个二分割 $\{R_i, \bar{R}_i\}$，匹配玩家则生成 R_i 与 $\bar{R}_i$ 之间的分数完美匹配 M_i，并用双随机矩阵 $\mathbf{M}_i \in [0, 1]^{|V| \times |V|}$ 编码。经过 t 次迭代后，所有匹配的并集定义了一个图 $H_t = \bigcup_{j=1}^{t} M_j$，其邻接矩阵为 $\mathbf{A}_t = \sum_{j=1}^{t} \mathbf{M}_j$。割玩家的目标是选择二分割，使得在 H_t 成为扩展器之前所需的轮数最少；匹配玩家的目标则是选择匹配，使轮数尽可能多。

算法 2 用于构建扩展器的通用割匹配博弈

$M_0 = (V, \emptyset)$
对 $i = 1$ 至 t **执行**
  $R_i = \text{FindBisection}(M_0, M_1, M_2, \ldots\ldots, M_{i-1})$
  $M_i = \text{FractionalMatch}(\{R_i, \bar{R}_i\})$
  $H_i = \bigcup_{j=1}^{i} M_j$

算法 2 概述了这一过程，其中 查找二分割 和 FractionalMatch 分别是割玩家和匹配玩家的子程序。Khandekar 等人 [25] 为割玩家提供了一种策略：对于任意匹配玩家子程序 FractionalMatch，该策略都能使 H_t 以高概率达到至少 1/2 的扩展率，所需迭代次数为 $t = O(\log^2 n)$。这一结果被用于证明图扩展问题存在 $O(\log^2 n)$ 近似算法。

自 Khandekar 等人 [25], 以来，割匹配框架以许多不同的方式得到了应用、改进和推广 [33, 38–40, 42]。尽管以往关于割匹配博弈的工作主要集中于扩展器节点权重（$\mu(v) = 1$），但 Orecchia 等人 [39] 最近引入了一种设置：在每次迭代中，割玩家生成一个满足 $\pi(R_i) \le \pi(\bar{R}_i)$ 的集合 R_i，而匹配玩家则在$\{R_i, \bar{R}_i\}$上生成一个 π-正则二分图 M_i。将截至第 i次迭代生成的所有二分图取并集，可得到图 $H_i = (V, E_{H_i})$。割玩家希望在较小的 i下，使 H_i成为一个 π-扩展器。引理 5.1总结了适用于这一设置的割玩家策略，并为图比率割目标带来了$O(\log n)$-近似。

引理 5.1（定理 7，见 [39]）。存在一种割玩家策略，使得对于任意匹配玩家策略，H_t满足 $\phi_{H_t,\Omega} = \frac{\pi}{n}$ $t = O(\log^2 n)$ π_i（对数）以高概率成立，其中某个（对数）。在第 轮中，割玩家可以在时间 $O(|E_{H_i}| \cdot$ 多对数（$\pi(V)$））内计算 R_i。

该策略基于在图中使用热核随机游走 H_t。该方法最初由 Orecchia 等人用于扩展率权重（$\mu(v) = 1$），并在 Orecchia 的博士论文中进行了更深入的研究 [40] 。引理5.1 将其推广到一般节点权重，并在近期关于重叠图聚类的工作中进行了介绍 [39]。最近，一组有重叠的作者对具有一般节点权重的割匹配博弈给出了更多技术细节 [2]。我们将引理5.1 中的割玩家策略称为HeatKernelPartition。在第一次迭代中，H_0 为空，因此割玩家从任意平衡二分开始$\{R, \bar{R}\}$。

5.2 近似算法

我们引入的扩张图嵌入概念涉及将一个扩张图嵌入超图中$\mathcal{H}$。因此，我们可以使用现有的割玩家策略来构造一个 π-扩张图，并根据定义$\mathcal{H}$ 3.2 中的含义，使用算法HyperCutOrEmbed将其嵌入其中。我们针对广义超图割扩展的割匹配近似算法（算法3）通过将算法2 中的通用分数匹配子程序替换为 HyperCutOrEmbed，并在割玩家步骤中使用HeatKernel-Partition 而得到。

定理 5.2. 对于某个 $t = O(\text{对数}^2 n)$，算法 3是针对超图 π-扩展率的 $O(\text{对数}\, n)$-近似算法，其中 割$\mathcal{H}$ 是任意基于基数的超图次模割函数。

算法 3 $O(\log n)$ 超图 π-扩展率近似

输入： $\mathrm{H} = (V, \mathrm{E})$，广义割$\mathcal{H}$，二分 $\{R, \bar{R}\}$
输出： 具有较小 π-扩展率的集合 S
$M_0 = \emptyset$
对于 $i = 1$ 至 t，**执行**
  $R_i = \text{HeatKernelPartition}(M_0, M_1, M_2, \ldots, M_{i-1})$
  $M_i, S_i, \alpha_i = \text{HyperCutOrEmbed}(\mathrm{H}\ R_i)$
返回 $S^* = \text{argmin}_{i=1,2,\ldots t}\, \phi_{\mathrm{H}}(S_i, \pi)$

证明。在第 i 次迭代中，HyperCutOrEmbed 生成一个 π-正则二分图 M_i，该图可嵌入 $G(\mathrm{H})$ 中，其扩展率为 $\frac{1}{\alpha_i}$，并生成一个扩展率为 ϕ 的集合 S_i，$\mathrm{H}(S_i, \pi) \le 2\alpha_i$。定义 $i^* = \text{argmin}_i\, \alpha_i$，并注意到 $\phi_{\mathrm{H}}(S^*, \pi) \le \phi_{\mathrm{H}}(S_{i^*}, \pi)$。二分图 H_t 的并可嵌入 $G(\mathrm{H})$ 中，拥塞度为 $\sum_{i=1}^{t} \frac{1}{\alpha_i} \le \frac{t}{\alpha_{i^*}}$。根据引理 5.1，以高概率有 $t \le c_1 \log^2 n$ 和 $\phi_{H_t,\pi} \ge c_2 \log n$，其中 c_1 和 c_2 是正数常量。结合引理 3.4，我们得到

$$\frac{\alpha_i^*}{\log n} \le \frac{c_1 \alpha_i^*}{t} \log n \le \frac{c_1 \alpha_i^*}{c_2 t} \phi_{H_t,\pi} \le \frac{c_1}{c_2} \phi_{\mathcal{H},\pi}.$$

因此 $\phi_{\mathcal{H}\,\pi} = \Omega(\alpha_i^*/\log n)$，算法返回一个集合 S^*，其中 $\phi_{\mathcal{H}}(S^*, \pi) \le 2\alpha_i^*$，从而证明了 $O(\log n)$ 的近似保证。

□

5.3 运行时间分析

对于运行时间分析，我们假设超图是连通的，并且所有超边权重和节点权重都已缩放为整数。令 $n = |V|$、$m = |\mathrm{E}|$，且 $\mu = \sum_{e \in \mathcal{E}} |e|$。为了˜关注运行时间中的主要项，我们将使用 O 记号来隐藏 m 和 n的对数因子。在分析中，我们还假设最大边惩罚 U 和节点权重之和˜足够小，使得 $O(\log U)$ 和 $O(\log \pi(V))$ 都是 $O(1)$，因此 HyperCutOrEmbed 的最大迭代次数为 $\tilde{O}(1)$。选择与导通率对应的节点权重函数时，有 $\log \pi(V) = O(\log mn)$；选择与标准扩展对应的 π时，有 $\log \pi(V) = \log n$。

为了加快算法 3 的运行时间，我们可以将现有的超图到图的稀疏化技术应用于归约 [47]。借助一个˜增强图 $G(\mathrm{H})$，我们可以在一个小常数 $\varepsilon > 0$ 下，以因子 $(1 + \varepsilon)$ 的精度对 H 的广义割函数进行建模；该图包含 $N = O(n + \sum_{e \in \mathcal{E}} \log |e|) = O(n + m)$ 个节点和 $M = O(\sum_{e \in \mathcal{E}} |e| \log |e|) = \tilde{O}(\mu)$ 条边。构造该图需要 $O(M)$ 的时间。对于 $\alpha > 0$，图 $G(\mathrm{H}\ R, \alpha)$ 同样包含 $O(N)$ 个节点和 $O(M)$ 条边。

HyperCutOrEmbed 中的流分解步骤可以使用动态树˜在 $O(M \log N) = O(\mu)$ 时间内完成 [43]；请注意，对于这一步，我们无需显式返回整个流分解，只需确定所嵌入二分图中每条有向路径的端点。引理 5.1 表明，割玩家策略所花费的总时间为 $\tilde{O}(|E_{H_t}|)$，我们可以将其上界限定为 $\tilde{O}(\mu)$。为了解释原因，请注意，我们添加到 FlowEmbed 构造的二分图中的边数，上界为不同有向流路径的数量；根据引理 2.3，该数量的上界为 $O(M) = O(\mu)$。合并所有 $O(\log^2 n)$

二分图表明 $\tilde{O}(|E_{H_t}|) = \tilde{O}(\mu)$。我们的算法的总体运行时间主要取决于求解最大 s-t流在 $G(\mathcal{H}\ R, \alpha)$所需的时间。若使用 van den Brand 等人的算法，总体运行时间为 $\tilde{O}(\mu+(n+m)^{3/2})$。Chen 等人的近期算法 [45] [11] 将运行时间降至 $\tilde{O}(\mu^{1+o(1)})$，相对于超图规模而言几乎是线性的 μ。

作为比较，现有的线性规划松弛 [23] 和半定规划松弛 [35], 仅适用于全有或全无超图割，它们都涉及 $\Omega(n^2+m)$ 个变量和 $\Omega(n^3+\sum_{e\in E}|e|^2)$ 个线性约束。对于许多 m、n 和 μ 的取值，仅仅写出这些松弛的运行时间就比运行我们的方法更长，而实际求解它们当然要慢得多。写成 $\min_{\mathbf{Ax=b}} \mathbf{c}^T\mathbf{x}$ 的形式时，该线性规划有 $\Omega(n^3+\sum_{e\in E}|e|^2)$ 个约束和变量。对于如此规模的问题，即使是近期在线性规划求解器方面取得的突破性理论成果 [15, 22]，其运行时间也远逊于 $\Omega(n^6+n^3\sum_e|e|^2+\mu^2)$。

5.4 实践改进

我们引入了多项实用改进，以简化算法的实现并提升近似保证。

实用的流算法。首先，我们的实现使用推送-重标记最大流算法 [12]，其理论运行时间较差（但仍然很快），并且配备了各种启发式方法，使其在实践中运行得非常快。对于流分解，我们采用一种标准的分解技术（见 [1] 中的定理 3.5），不需要使用动态树。这种方法在理论上同样较慢，但仍然很快，并且在实践中更易于使用。

更新为 HyperFlowEmbed。我们还使用了 HyperFlowEmbed 的一个改进版本，该版本返回一个二分图 M_R，它可以在拥塞为 $1/\alpha$的情况下进行嵌入，同时还返回一个集合 S，该集合的 π-扩展率等于 α，而不仅仅是返回一个满足 $\phi_{\mathcal{H}}(S,\pi) < 2\alpha$的集合 S。这使近似效果提高了 2 倍；虽然不会改变理论上的最坏情况 $O(\log n)$ 近似界，但在实践中可能会产生显著差异。为实现这一点，我们在第一次迭代中设置 $\alpha = \phi_{\mathcal{H}}(R,\pi)$，并在 G(H R, α) 上求解一个最大 s-t流问题，以搜索一个扩展率优于 $\phi_{\mathcal{H}}(R,\pi)$ 的集合 S，其扩展率为 π。在每次迭代中，我们将 α更新为上一轮迭代找到的改进集合的π-扩展率，直到不再发现改进为止。这种迭代细化方法是标准做法，相关的比率割改进算法 [28, 39, 46, 50]通常会在实践中采用它。尽管对 α 执行二分法可以获得更好的理论运行时间，但在实践中，迭代细化过程通常只需进行几次割改进就会收敛。因此，除了将近似保证提高 2 倍之外，这种方法在实践中通常也更快。

改进的下界与后验近似。最后，我们为算法在每次迭代中满足的近似保证建立了更精确的下界。这使我们能够获得改进的后验近似保证。对于节点权重函数 π，令 $\mathbf{D}_\pi$ 为对角矩阵，其中 $\mathbf{D}_\pi(i,\ i) = \pi(i)$，并定义 $\mathbf{L}_t = \mathbf{D}_\pi^{-1\ 2}/\mathbf{L}_t\mathbf{D}_\pi^{-1\ 2}/$。令 $\lambda_2(\mathbf{L}_t)$ 为该矩阵的第二小特征值。下面的定理给出了一个精确且易于计算的下界，用于衡量我们的近似算法所达到的近似因子在每次迭代中。为证明这一结果，我们使用更新后的 HyperFlowEmbed 版本，该版本返回一个集合 S，其中 $\phi_{\mathcal{H}}(S,\ \pi) = \alpha$。

定理 5.3. 经过 t 次迭代后，算法 3 将返回一个集合 S_{2t}^*，满足 $\phi_{\mathcal{H}}(S_t^*,\pi) \le \rho_t\phi_{\mathcal{H}\,\pi}$，其中 $\rho_t = \frac{2\gamma_t\phi_{\mathcal{H}}}{\lambda_2(\mathcal{L}_t)} \le \frac{2}{\lambda_2(\mathcal{L}_t)}$ 且 $\gamma_t = \sum_{i=1}^{t}\frac{1}{\alpha_i}$。

证明。将特征值表示为 Rayleigh 商，并利用谱图论中的标准技术，首先可得

$$\lambda_2(\mathcal{L}_t) = \min_{\substack{\mathbf{x}\in\mathbb{R}\\ \mathbf{x}^T\mathbf{D}_\pi\mathbf{e}=0}} \frac{\mathbf{x}^T\mathbf{L}_t\mathbf{x}}{\mathbf{x}^T\mathbf{D}_\pi\mathbf{x}} \le 2\cdot\min_{S\subseteq V}\phi_{H_t}(S,\pi) = 2\phi_{H_t,\pi}. \quad (15)$$

通过考虑任意集合 $S\subseteq V$，并定义向量 $\mathbf{x}$，可以证明 π-扩展率在 H_t 中的这个下界

$$\mathbf{x}(i) = \begin{cases} \frac{1}{\pi(S)} & \text{if } i\in S \\ -\frac{1}{\pi(\bar{S})} & \text{if } i\in\bar{S}. \end{cases}$$

注意，该向量满足 $\mathbf{x}^T\mathbf{D}\pi\mathbf{e} = 0$，并且

$$\frac{\mathbf{x}^T\mathbf{L}_t\mathbf{x}}{\mathbf{x}^T\mathbf{D}_\pi\mathbf{x}} = \frac{\mathbf{cut}_{H_t}(S)}{\pi(S)} + \frac{\mathbf{cut}_{H_t}(S)}{\pi(\bar{S})} \le 2\phi_{H_t}(S,\pi).$$

结合式（15）中的界，以及 H_t 可以以拥塞 γ_t 嵌入 H 这一事实，可得 $\phi_{\mathcal{H}\,\pi}$的下界：

$$\frac{1}{2\gamma_t}\lambda_2(\mathcal{L}_t) \le \frac{1}{\gamma_t}\phi_{H_t,\pi} \le \phi_{\mathcal{H},\pi}.$$

将 $\phi_{\mathcal{H}}(S_t^*,\pi)$ 除以此下界即可证明近似比。

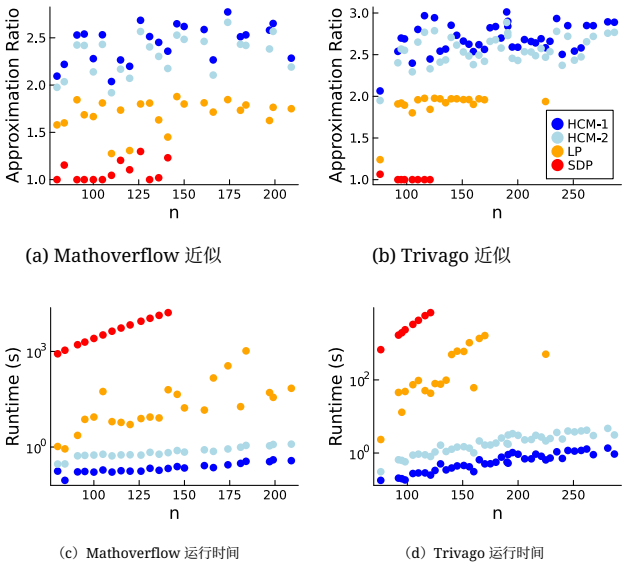
□

该定理为割玩家策略提出了另一种选择：通过对 $\{R,\bar{R}\}$ 中第二小特征向量的各项进行阈值处理来选择一个划分 $\mathcal{L}_t$。在该划分上嵌入一个 π-正则二分图会在后续迭代中增大 $\lambda_2(\mathcal{L}_t)$ 的值。在极端情况下，根据该特征向量贪心地选择二分划分可能会进展缓慢（参见 [38]第 35 页的讨论）。然而，这种方法通常能在实践中产生良好结果，我们可以使用定理 5.3 结合该策略计算 π-扩展率的具体下界。

6 实验

我们使用第 5.4 节中的所有实践改进，在 Julia 中实现了我们的超图切割匹配算法（HCM）。我们将其与其他超图比率切割算法进行比较，目标是在来自不同应用领域的超图上最小化全局比率割目标。所有实验均在配备 Apple M1 芯片和 16GB 内存的 MacBook Air 上运行。所有数据和代码均提供于 https://github.com/nveldt/HyperCutMatch。附录提供了有关实现和数据集的更多细节。

数据集。*Trivago* 对单个用户在 Trivago.com 浏览会话期间点击的一组度假租赁房源进行编码 [13]。*Mathoverflow* 对单个用户在 mathoverflow.com 回答的一组帖子进行编码 [46]。亚马逊 9 [52]对由同一用户评论的亚马逊零售商品进行编码（这些商品来自更大数据集中的 9 个最小商品类别 [37]）。*TripAdvisor* 对在 Tripadvisor.com 上由同一用户评论的一组住宿进行编码



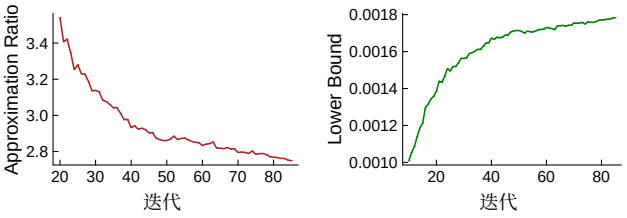
(a) Mathoverflow 近似　(b) Trivago 近似

(c) Mathoverflow 运行时间　(d) Trivago 运行时间

图 3：在一系列小型超图上，三种超图扩张近似算法的结果。先前的近似算法（线性规划和半定规划）无法扩展到更大的超图，并且即使对于这些小型超图也常常失败。HCM-1 使用 $10\log_2 n$ 轮割匹配，而 HCM-2 使用 $30\log_2 n$ 轮。

表 2：在四个超图上运行 HCM（运行 5.3 定理所得的近似因子）及运行时间，结果为 5 次试验的平均值（$_2 n$ 次迭代）（±标准差）。

	n	m	平均 $\lvert e\rvert$	近似值	运行时间（秒）
Amazon9	13138	31502	8.1	2.78 ±0.012	254.8 ±10.4
Mathoverfl	73851	5446	24.2	3.08 ±0.014	643.8 ±31.1
TripAdvisor	8929	130568	4.1	2.71 ±0.025	699.3 ±29.8
Trivago	172738	233202	3.1	2.78 ±0.038	2372.6 ±50.8



（a）近似　（b）下界

图 4：在 *Trivago* 超图上，HCM 在每一步计算得到的近似因子和下界。

相同用户 [49]。有关超图统计信息，请参见表 2。我们还考虑了来自 UCI 仓库的超图 [6]，这些超图已在以往的超图学习工作中被频繁使用 [18, 31, 56, 58]。

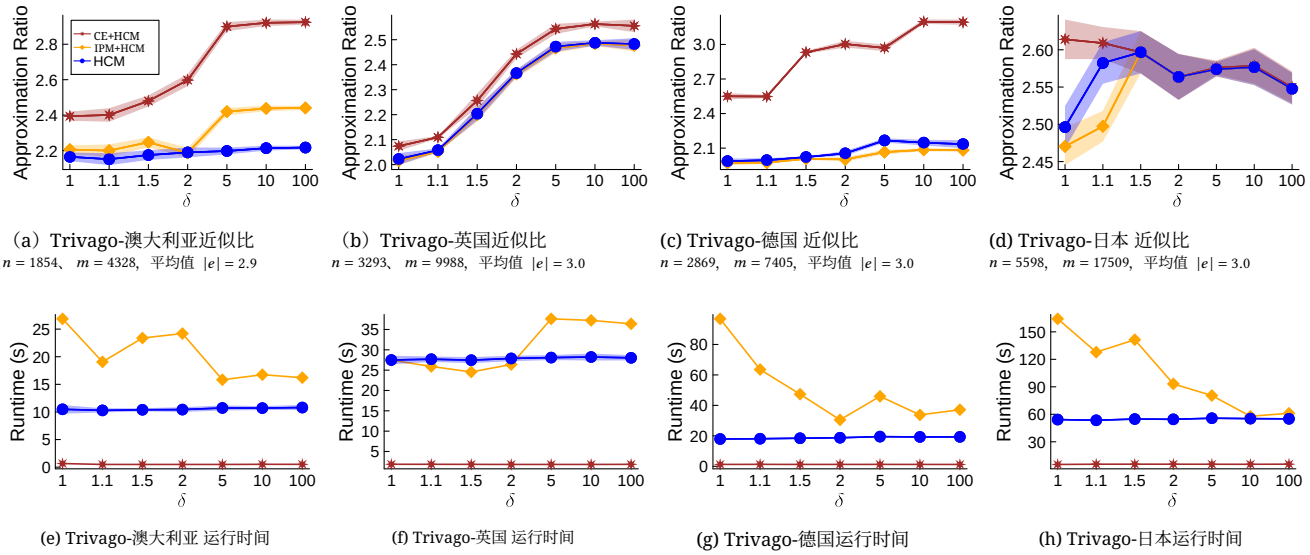
6.1 与凸松弛方法的比较

我们的方法比以往针对超图比率割的理论近似算法具有高出几个数量级的可扩展性而且适用于范围广得多的问题。我们将其与基于半定规划的 $O(\sqrt{\log n})$ 近似（SDP） [35] 以及基于对线性规划松弛进行舍入的 $O(\log n)$ 近似算法（LP） [23]进行比较。我们重点研究采用标准超图割函数的扩展目标（对所有 v，$\pi(v) = 1$），因为这些方法不适用于广义超图割。

我们选择了一系列规模较小、*Mathoverflow* 和 *Trivago* 超图作为基准测试，并逐步增加规模。这些超图分别对应于表 *Mathoverflow* 和 *Trivago* 超图的子超图 2，每个子超图均通过选择具有特定元数据标签的节点获得。我们选取了 23 个 *Mathoverflow* 超图，每个超图对应一组带有特定主题标签的帖子（节点）（例如 图论）。这些超图包含 $n = 80$ 至 $n = 209$ 个节点。超边数量 m 通常介于$n/3$ 和 $n/2$ 之间；对于这些超图，平均超边大小介于 3 到 7 之间。我们还考虑了 41 个 *Trivago* 超图（每个超图对应一个不同的城市标签，例如 *Austin, USA*），其中大多数超图的超边数量是节点数量的 1 至 3 倍，并且所有超图的平均超边大小都介于 2 和 4之间。

图 3 展示了线性规划、半定规划和我们的方法 HCM 的结果；这些方法都会计算最优扩展率的显式下界，可用于计算事后近似保证。我们使用两种不同的迭代次数运行 HCM，以展示运行时间与近似保证之间的权衡。当半定规划和线性规划收敛时，它们会为超图扩张给出非常好的下界，并可通过舍入得到非常好的事后近似保证。问题在于，即使对于非常小的实例，这些方法也难以扩展。我们能够使用线性规划处理所有 *Mathoverflow* 数据集，但该方法在除 41 个 > 30分钟）外的所有实例上都会超时*Trivago* 超图。取消时间限制后，该方法有时能找到不错的解，但代价是运行时间大幅增加（例如，一个包含 229 个节点的超图耗时超过 5 小时）；但在某些情况下，底层线性规划求解器会在尚未收敛时返回。半定规划方法的可扩展性更差：如果将时间限制设为 30 分钟，对于几乎所有这些小型超图，该方法都无法收敛。处理一个包含 121 个节点的 *Trivago* 超图时，该方法耗时超过一小时；处理一个包含 141 个节点的 *Mathoverflow* 超图则耗时 4.5 小时。鉴于可扩展性问题，我们尝试用该方法处理的各类超图中，这些是规模最大的实例。总体而言，尽管线性规划和半定规划在成功求解底层凸松弛时能给出非常好的下界，但它们并不实用。相比之下，HCM 能以极快的速度获得高质量解，通常只需几秒钟。

值得注意的是，图 3(a) 和 3(b) 中近似比的差异几乎完全源于每种方法计算出的下界不同。这些方法得到的扩展率几乎完全相同，并且在许多情况下找到相同的输出集合。这表明，找到扩展率较小的集合并不是主要挑战。相反，比较的重点在于每种方法的下界与最优解的接近程度。



（a）Trivago-澳大利亚近似比
$n = 1854$、 $m = 4328$，平均值 $|e| = 2.9$

（b）Trivago-英国近似比
$n = 3293$、 $m = 9988$，平均值 $|e| = 3.0$

(c) Trivago-德国 近似比
$n = 2869$， $m = 7405$，平均值 $|e| = 3.0$

(d) Trivago-日本 近似比
$n = 5598$， $m = 17509$，平均值 $|e| = 3.0$

(e) Trivago-澳大利亚 运行时间

(f) Trivago-英国 运行时间

(g) Trivago-德国运行时间

(h) Trivago-日本运行时间

图 5：顶部一行：将每种方法找到的最佳导通率集合与 HCM 计算出的导通率下界进行比较所得到的近似结果。底部一行：以秒为单位的运行时间。IPM 和 CE 的运行时间不包括计算 HCM 下界所需的时间。我们绘制了 HCM 运行 5 次的均值；阴影区域表示标准差。

6.2 更大超图上的扩展率

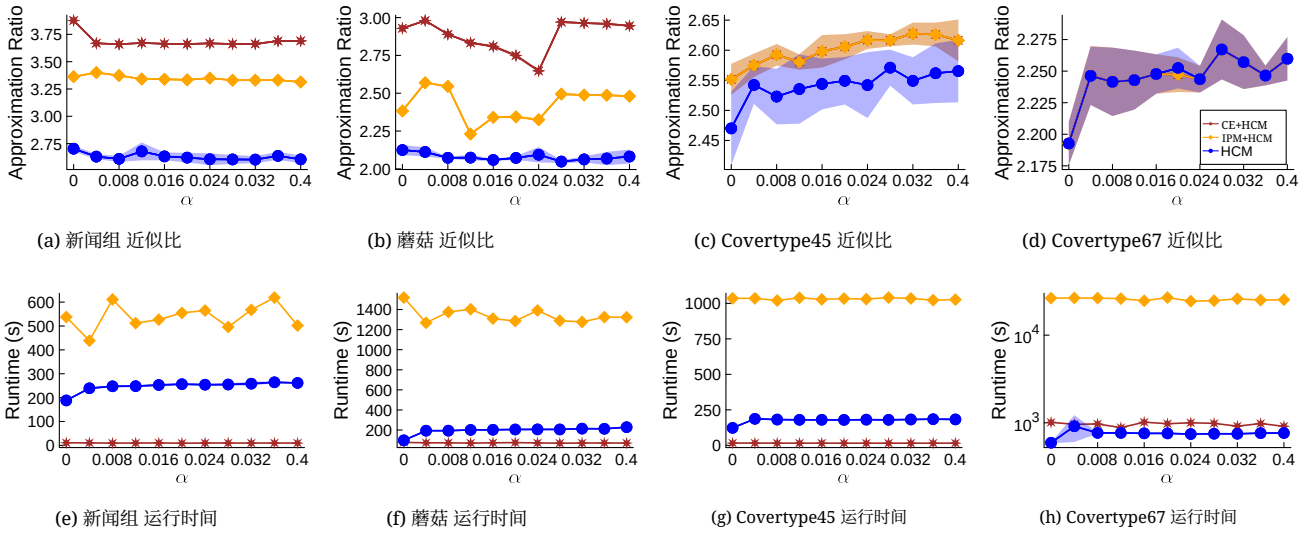
表2 列出了 HCM 在四个超图上获得的运行时间和近似结果；这些超图的规模比 LP 和 SDP 能够处理的任何超图大几个数量级。我们的方法能够快速获得与最优超图扩张解仅相差一个较小因子的解。图4 更详细地展示了对于 *Trivago* 超图，HCM 在每次迭代中维持的下界和近似因子如何得到改进。其他三个数据集的图形非常相似。第 t 次迭代时的下界（图 4(b)）使用定理 5.3计算，而第 t 次迭代时的近似因子（图 4(a)）则通过将前 t 次迭代中找到的最佳导通率集合除以下界得到。HCM 返回的最佳导通率集合通常在前几次迭代中就能找到。因此，运行算法进行多次迭代所实现的主要目标，是改进下界证书。图 4 显示，算法在 $5\log_2 n$ 次迭代之后仍在取得进展，这表明，通过运行算法更长时间，可以获得更强的下界证书和近似保证。

IPM 和 CE 是目前在具有广义割函数的超图中最小化比率割的最先进方法。IPM 是一种推广方法，源自先前仅适用于标准超图割 [18]的方法。CE 可视为先前团扩展技术 [7, 57], 的推广，而这些技术仅适用于限制更严格的超图割函数。除适用于广义超图割外，这两种方法也比线性规划和半定规划更实用，但其理论保证较弱。IPM 是一种启发式方法，没有近似保证；而 CE 的近似保证会随最大超边大小增加而大幅变差，最坏情况下即使对于图也可能达到 $O(n)$。在我们的实验中，我们使用了 Li 和 Milenkovic [31]先前提供的 IPM 的 C++ 实现。IPM 的性能很大程度上取决于将哪个向量作为热启动输入。我们尝试了几种选择，发现使用 CE 计算出的特征向量作为热启动，效果最佳。我们提供了自己全新的 CE Julia 实现，其速度显著快于原始的概念验证实现。附录中包含了更多实现细节。

6.3 Trivago 超图与广义割

HCM 的一个显著优势是，它能够计算超图 π扩展率的显式下界（通过定理 5.3），在实践中可用于事后检验近似保证，即使是那些本身不提供任何近似保证的其他超图比率割算法也适用。为展示这一功能的强大之处，我们比较了 HCM 与用于次模超图的逆幂法（IPM） [31]以及用于非齐次超图的团扩展方法（CE） [30]在 *Trivago* 超图上最小化比率割目标的表现。我们还展示了在某些情况下，组合多种算法如何能够带来最佳结果。

图5 展示了 HCM、IPM 和 CE 在四个 Trivago超图上的运行时间和事后近似保证。这些超图分别对应澳大利亚、英国、德国和日本的度假租赁房源。我们特别考虑每个超图的 2-核，因为这些 2-核具有更有趣且更多样的割结构，因此更适合作为比较广义超图比率割算法的案例。对于这些实验，我们最小化导通率目标函数（$\pi(v) = d_v$），并使用一种广义超图割函数，该函数在每条超边上应用 δ-线性分裂函数 $\mathbf{w}_e(S) = \min\{|S \cap e|, |\bar{S} \cap e|, \delta\}$。我们针对 δ的多种不同取值进行了实验。选择这种分裂函数，是因为先前研究表明， δ的选择会显著影响



(a) 新闻组 近似比 (b) 蘑菇 近似比 (c) Covertype45 近似比 (d) Covertype67 近似比

(e) 新闻组 运行时间 (f) 蘑菇 运行时间 (g) Covertype45 运行时间 (h) Covertype67 运行时间

图6：在四个基准超图上运行三种超图比率割算法时的近似因子和运行时间。IPM 和 CE 的运行时间不包括计算 HCM下界所需的时间。我们绘制了 HCM 运行 5 次的均值；阴影区域表示标准差。

输出集的大小，并影响下游聚类应用中的性能 [32, 46, 47]。这四个 *Trivago* 超图展示了通过改变 δ可以找到的多种不同割，因此为研究这些算法在广义分裂函数下寻找不同类型比率割的表现提供了良好的案例。

在寻找低导通率集合方面，HCM 与 IPM 的性能各有得失，但二者得到的结果非常相近。然而，HCM 的速度明显更快，并且还会计算下界，从而能够证明其解与最优解的接近程度。我们的 CE 实现比 HCM 更快，但在近似比方面表现较差。在图 5中，我们使用 HCM下界为 IPM 和 CE 获得后验近似。这些方法自身无法提供如此强的近似保证。因此，即使在 IPM 找到更优导通率集合的情况下，HCM 计算出的下界也能提供新的信息，用于证明近似保证。由于 IPM 的性能也依赖于将 CE 用作热启动，我们可以看到，通常是三种算法的组合能够带来最佳结果。

6.4 其他基准超图

我们在一组常被用作超图学习和聚类基准的超图上进行了类似的实验 [18, 31, 56, 57]。这些超图的统计数据见表3。图6 展示了使用以下广义分裂函数获得的运行时间和近似因子

$$\mathbf{w}_e(A) = \frac{1}{2} + \frac{1}{2}\min\left\{1, \frac{|S|}{\lceil \alpha|e| \rceil}, \frac{|e\backslash S|}{\lceil \alpha|e| \rceil}\right\}. \tag{16}$$

该分裂函数由 Li 和 Milenokovic [31] 提出并进行了研究，旨在更好地处理分类错误和异常值。它以一种使超边中的大多数节点位于

表3：图6中超图的统计数据

超图	n	m	平均 $\lvert e\rvert$	$\sum_e \lvert e\rvert$
新闻组	16242	100	654.5	65451
蘑菇	8124	112	1523.2	170604
Covertype45	12240	128	1147.5	146880
Covertype67	37877	136	3342.0	454514

割的同一侧的方式，对分裂超边施加较小的惩罚。在*Covertype67*上，三种方法得出的结果基本相同；但在其他三个超图上，就近似因子而言，HCM 相较于 IPM 和 CE 有显著改进。它的速度也始终快于 IPM，并且在 *Covertype67* 上甚至略快于 CE。

7 讨论

我们提出了首个用于最小化超图比率割的算法，该算法同时满足以下条件：(1) 具有为 $O(\log n)$ 的近似保证；(2) 适用于广义超图割函数；(3) 配备了实用的实现。在实践中，该算法在寻找比率割方面表现非常出色，所得结果与该算法生成的最优解下界相差一个较小的因子（约为 2-3）。一个尚待解决的问题是，探索如何针对不同的目标应用选择最佳的广义超图割函数。另一个研究方向是，为适用于一般子模分割函数（包括那些并非基于基数的函数）的超图比率割寻找改进的近似算法。

参考文献

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. 网络流. Prentice-Hall Inc., 1988.

[2] Konstantinos Ameranis, Antares Chen, Lorenzo Orecchia, and Erasmo Tani. 通过广义割匹配博弈实现子模超图划分的高效基于流的近似算法。arXiv预印本 arXiv:2301 08920, 2023年 . .[3] Reid Andersen, Fan Chung, and Kevin Lang. 使用PageRank向量进行局部图划分。载于第47届IEEE年度计算机基础研讨会论文集，FOCS ’06, 2006年。[4] Reid Andersen and Kevin J. Lang. 一种改进图划分的算法。载于2008年ACM-SIAM离散算法年度研讨会论文集，SODA ’08, 第651–660页。工业与应用数学学会，2008。[5] Sanjeev Arora, Satish Rao, and Umesh Vazirani. 扩展流、几何嵌入与图划分。ACM期刊，56 37, 2009 2 1( ): – .[6] Arthur Asuncion and David Newman. UCI机器学习仓库。https: //archive.ics.uci.edu/ml/index.php, 2007。[7] Austin R. Benson, David F. Gleich, and Jure Leskovec. 复杂网络的高阶组织结构。《科学》，353(6295):163–166, 2016。[8] Austin R Benson, Jon Kleinberg, and Nate Veldt. 用于广义超图割的增强稀疏化器及其在可分解子模函数最小化中的应用。arXiv预印本 arXiv:2007.08075, 2020。[9] Jean Bourgain. 将有限度量空间Lipschitz嵌入Hilbert空间。《以色列数学杂志》，52(1):46–52, 1985。[10] T-H Hubert Chan, Anand Louis, Zhihao Gavin Tang, and Chenzi Zhang. 超图拉普拉斯算子的谱性质与近似算法。ACM期刊，65(3):1–48, 2018。[11] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. 近线性时间内的最大流和最小费用流。载于2022年IEEE第63届年度计算机基础研讨会，FOCS ’22, 第612–623页，2022年。[12] Boris V Cherkassky and Andrew V Goldberg. 关于最大流问题推送重标记方法的实现。《算法学》，19(4):390–410, 1997。[13] Philip S. Chodrow, Nate Veldt, and Austin R. Benson. 生成式超图聚类：从块模型到模块度。《科学进展》，7 1303, 2021 28( ):eabh .[14] Fan R. K. Chung. 《谱图论》，第92卷。美国数学学会，1997。[15] Michael B Cohen, Yin Tat Lee, and Zhao Song. 在当前矩阵乘法时间内求解线性规划。ACM期刊，68(1):1–39, 2021。[16] Kimon Fountoulakis, Pan Li, and Shenghao Yang. 局部超流扩散。载于《神经信息处理系统进展》，第34卷 21, 2021 of NeurIPS ’ 。[17] Michael Grant and Stephen Boyd. CVX：用于规范凸规划的Matlab软件，2.1版。http://cvxr.com/cvx, 2014年3月。[18] Matthias Hein, Simon Setzer, Leonardo Jost, and Syama Sundar Rangapuram. 超图上的全变差——重新审视超图学习。载于《神经信息处理系统进展》，第26卷 NeurIPS ’13, 2013。[19] Rania Ibrahim and David F Gleich. 使用容量释放扩散进行局部超图聚类。《PLOS ONE》，15(12):e0243485, 2020。[20] Edmund Ihler, Dorothea Wagner, and Frank Wagner. 用具有相同最小割性质的图对超图建模。《信息处理快报》，45(4):171 – 175, 1993。[21] Masahiro Ikeda, Atsushi Miyauchi, Yuuki Takai, and Yuichi Yoshida. 通过热方程寻找超图中的Cheeger切割。《理论计算机科学》，930:1–23, 2022年。[22] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. 求解一般线性规划的更快算法。载于第53届ACM SIGACT计算理论研讨会论文集，STOC ’21, 第823–832页，纽约，NY，美国，2021年。美国计算机协会。[23] Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. 关于超图谱稀疏化的紧确界。arXiv预印本 arXiv:2011.06530, 2020。[24] Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. 关于超图谱稀疏化的紧确界。载于第53届ACM SIGACT计算理论研讨会论文集，STOC ’21, 第598–611页，2021年。[25] Rohit Khandekar, Satish Rao, and Umesh Vazirani. 使用单商品流进行图划分。ACM期刊，56(4):1–15, 2009。[26] Kyle Kloster and David F Gleich. 基于热核的社区检测。载于第20届ACM SIGKDD知识发现与数据挖掘国际会议论文集，KDD ’14, 第1386–1395页，2014。[27] Pushmeet Kohli, L’ubor Ladický, and Philip H. S. Torr. 用于确保标签一致性的鲁棒高阶势。《国际计算机视觉杂志》，82(3):302–324, 2009。[28] Kevin Lang and Satish Rao. 一种基于流的方法，用于改善图割的扩展率或导通率。载于整数规划与组合优化会议，IPCO ’04, 第325–337页，2004年。[29] Tom Leighton and Satish Rao. 多商品最大流最小割定理及其在设计近似算法中的应用。ACM期刊，46(6):787–832, 1999. [30] Pan Li 和 Olgica Milenkovic。具有应用的非齐次超图聚类。载于《神经信息处理系统进展》第30卷，NeurIPS ’17，第2308–2318页，2017年。[31] Pan Li 和 Olgica Milenkovic。次模超图：p-拉普拉斯算子、Cheeger不等式与谱聚类。载于第35届国际机器学习会议论文集，ICML ’18，第3014–3023页，2018年。[32] Meng Liu、Nate Veldt、Haoyu Song、Pan Li 和 David F Gleich。用于聚类和半监督学习的强局部超图扩散。载于2021年Web会议论文集，WWW ’21，第2092–2103页，2021年。[33] Anand Louis。有向图上的割匹配博弈。arXiv预印本 arXiv:1010.1047, 2010年。[34] Anand Louis。超图马尔可夫算子、特征值与近似算法。载于第47届ACM计算理论年度研讨会论文集，STOC ’15，第713–722页，2015年。[35] Anand Louis 和 Yury Makarychev。超图小集合扩展和小集合顶点扩展的近似算法。《计算理论》，12(1):1–25，2016年。[36] Michael W Mahoney、Lorenzo Orecchia 和 Nisheeth K Vishnoi。图的局部谱方法：用于改进图划分和局部探索数据图。《机器学习研究杂志》，13 2365，2012 1 2339( ): – 。[37] Jianmo Ni、Jiacheng Li 和 Julian McAuley。利用远程标注评论和细粒度方面为推荐提供依据。载于2019年自然语言处理实证方法会议暨第9届国际自然语言处理联合会议论文集，EMNLP-IJCNLP ’19，第188–197页，2019年。[38] Lorenzo Orecchia。使用谱方法和半定规划技术进行图划分的快速近似算法。博士论文，加州大学伯克利分校，2011年。[39] Lorenzo Orecchia、Konstantinos Ameranis、Charalampos Tsourakakis 和 Kunal Talwar。用于混合及重叠图聚类的实用近线性时间近似算法。载于国际机器学习会议，ICML ’22，第17071–17093页，2022年。[40] Lorenzo Orecchia、Leonard J Schulman、Umesh V Vazirani 和 Nisheeth K Vishnoi。通过单商品流划分图。载于第40届ACM计算理论年度研讨会论文集，STOC ’08，第461–470页，2008年。[41] Lorenzo Orecchia 和 Zeyuan Allen Zhu。用于局部图聚类的基于流的算法。载于第25届ACM-SIAM离散算法年度研讨会论文集，SODA ’14，第1267–1286页。SIAM，2014年。[42] Jonah Sherman。突破多商品流瓶颈，实现对 $O(\sqrt{\log n})$的稀疏割近似。载于2009年第50届IEEE计算机科学基础年度研讨会，FOCS ’09，第363–372页，2009年。[43] Daniel D Sleator 和 Robert Endre Tarjan。动态树的数据结构。《计算机与系统科学杂志》，26(3):362–391，1983年。[44] Luca Trevisan。关于图划分、扩展图和谱方法的讲义。加州大学伯克利分校，https://people. eecs. berkeley. —— edu/luca/books/expanders-2016. pdf，2017年。[45] Jan van den Brand、Yin Tat Lee、Yang P. Liu、Thatchaphol Saranurak、Aaron Sidford、Zhao Song 和 Di Wang。针对稠密实例，在近线性时间内求解最小费用流、马尔可夫决策过程和 ℓ_1-回归。载于第53届ACM SIGACT计算理论研讨会论文集，STOC 2021，第859–869页，2021年。[46] Nate Veldt、Austin R. Benson 和 Jon Kleinberg。最小化超图中的局部化比率割目标。载于第26届ACM SIGKDD知识发现与数据挖掘国际会议论文集，KDD ’20，第1708–1718页，美国纽约州纽约市，2020年。美国计算机协会。[47] Nate Veldt、Austin R Benson 和 Jon Kleinberg。基于基数组成部分的近似可分解次模函数最小化。载于《神经信息处理系统进展》第34卷，NeurIPS ’21，2021年。[48] Nate Veldt、Austin R. Benson 和 Jon Kleinberg。具有一般分裂函数的超图割。《SIAM评论》，64(3):650–685，2022年。[49] Nate Veldt、Austin R Benson 和 Jon Kleinberg。高阶同质性的组合刻画与不可能性。《科学进展》，9 3200，2023 1( ):eabq 。[50] Nate Veldt、David Gleich 和 Michael Mahoney。一种简单且强局部的基于流的割改进方法。载于第33届国际机器学习会议论文集，ICML ’16，第1938–1947页，2016年6月20–22日。[51] Nate Veldt、Christine Klymko 和 David F. Gleich。具有更佳种子集包含性的基于流的局部图聚类。载于2019年SIAM数据挖掘国际会议论文集，SDM ’19，2019年。[52] Nate Veldt、Anthony Wirth 和 David F Gleich。超图和二分图中的参数化相关聚类。载于第26届ACM SIGKDD知识发现与数据挖掘国际会议论文集，KDD ’20，第1868–1876页，2020年。[53] David P Williamson。网络流算法。剑桥大学出版社，2019年。

[54] Hao Yin、Austin R. Benson、Jure Leskovec 和 David F. Gleich。局部高阶图聚类。载于第*23*届*ACM SIGKDD*知识发现与数据挖掘国际会议论文集，KDD '17，564，2017 555页 – 。[55] Yuichi Yoshida。次模变换的Cheeger不等式。载于*2019*年*ACM-SIAM*离散算法年度研讨会论文集，SODA '19，第2582–2601页，2019年。[56] Chenzi Zhang、Shuguang Hu、Zhihao Gavin Tang 和 TH Hubert Chan。重新审视超图学习：置信区间与次梯度法。载于国际机器学习会议，ICML '17，4034，2017 4026页 – 。[57] Dengyong Zhou、Jiayuan Huang 和 Bernhard Schölkopf。利用超图进行学习：聚类、分类与嵌入。载于神经信息处理系统进展，NeurIPS '06，第1601–1608页，2006年。[58] Yu Zhu 和 Santiago Segarra。具有边依赖顶点权重的超图割。《应用网络科学》，7(1):1–20，2022年。

引理 4.2 的证明

证明。令 $f_R = \sum_{(u,\ v)\in R\times\bar{R}} f_{(uv)}$ $_R$ 为从 $G(\mathrm{H}\ R,\ \alpha)$ 中的最大 s-t流 f得到的 $G(\mathrm{H})$ 中的有向多商品流。换言之，对于每一对 $(u,\ v)$，其中 $u \in R$且 $v \in \bar{R}$，$f_{(uv)R}$是一个 u-v流函数，将 $|f_{(uv)}{}_R\ | = \mathbf{M}_R(u,\ v)$ 流从 u发送到 v。每个 $u \in R$都是该多商品流函数的亏缺节点，每个 $v \in \bar{R}$都是盈余节点。

不失一般性，我们可以假设 f是无环的，这意味着 f_R 也无环。令 $S \subseteq V$为任意集合；我们的目标是修改 f_R，使其成为流函数$f_S = \sum_{(u,\ v)\in S\times\bar{S}} f_{(uv)S}$，以便在 $u \in S$、$v \in \bar{S}$ 且 $\{u,\ v\}$ 是 M_R 中的一条边时，将$\mathbf{M}_R(u,\ v)$ 流从 u发送到 v。特别地，这意味着我们需要找到一种方法，为任意一对 $(u,\ v) \in (S \cap \bar{R}) \times (\bar{S} \cap R)$ 反转流向，因为对于这类点对，f_R 发送流的方向是从 v到 u，而不是从 u到 v。

我们首先考虑如何反转单条有向流路径上的流。任取 M_R 中的一条边 $\{u, v\}$，满足 $u \in S \cap \bar{R}$ 和 $v \in \bar{S} \cap R$。原始多商品流函数 f_R 包含一个流函数 $f_{(vu)\ R}$，该函数将 $\mathbf{M}_R(u, v)$ 单位的流从 v 发送到 u。根据引理 2.3，该流可以分解为 $f_{(vu)\ R} = \sum_{j=1}^{r} f_{(vu)\ j}$，其中每个 f_j 都是从 v 到 u 的有向流路径。根据 $G(\mathrm{H})$ 的构造，这条流路径将依次经过一系列 CB 小工具中的节点和边，这些 CB 小工具对应于超图 $\mathrm{H} = (V, \mathrm{E})$ 中的一系列超边 $\{e_1, e_2, \ldots, e_k\} \subseteq \mathrm{E}$。对于该序列中的第 i 条超边 e_i 及其对应的第 i 个 CB 小工具，设 e'_i 是该超边对应小工具中的第一个辅助节点，设 e''_i 是第二个辅助节点。回顾可知，其中存在一条边 (e'_i, e''_i)；对于每个节点 $x \in e_i$，还存在一条有向边 (x, e'_i) 和一条有向边 (e''_i, x)（见图 1 和图 7）。因此，一条从 $v = v_0$ 到 $u = v_{k+1}$ 的简单有向流路径 $f_{(vu)\ j}$ 具有如下形式：

$$\begin{aligned} v = v_0 \to (e'_1 \to e''_1) \to v_1 \to (e'_2 \to e''_2) \to v_2 \to \cdots \\ \cdots \to v_{k-1} \to (e'_k \to e''_k) \to v_k = u, \end{aligned} \tag{17}$$

其中，$v_t \in V$对每个 $t \in \{0、1、\ldots\ldots、k\}$，并且沿每条边发送相同数量的流。根据每个 CB- gadget 的构造，存在一条方向相反的路径，其边权重集合完全相同：

$$\begin{aligned} u = v_k \to (e'_k \to e''_k) \to v_{k-1} \to \cdots \\ \cdots \to v_2 \to (e'_2 \to e''_2) \to v_1 \to (e'_1 \to e''_1) \to v_0 = v. \end{aligned} \tag{18}$$

回顾一下，f_R是无环的。这意味着没有流沿形如 (v_i, e'_i) 的边发送，其中 $i \in \{1, 2, \ldots, k\}$，因为 f_R包含一条流路径 $f_{(vu)\ j}$，并且边 (e'_i, e''_i) 和 (e''_i, v_i) 上的流值为正。因此，我们可以将式（17）中的流路径 $f_{(vu)\ j}$替换为定义在式（18）所示边上的流路径 $\hat{}\, f_{(vu)\ j}$，其流值与 $f_{(vu)\ j}$相同，但方向相反。图7展示了单条流路径的这一流反转过程。

我们通过将这一流反转过程同时应用于 f_R 中所有方向错误的流路径，构造一个新的多商品流函数 f_S。形式化地，若 p_{vu} 表示 f_R 中从 v 到 u 的简单有向流路径的索引集合（由引理 2.3 中的流分解得到），则$f_R = \sum_{(v,u)\in R\times\bar{R}} \sum_{j\in p_{vu}} f_{(vu)\,j}$，并定义：

$$f_S = f_R + \sum_{(v,u)\in(\bar{S}\cap R)\times(S\cap\bar{R})} \sum_{j\in p_{vu}} \hat{f}_j^{(vu)} - f_j^{(vu)}.$$

换言之，对于每个 $(u, v) \in (S \cap \bar{R}) \times (\bar{S} \cap R)$ 以及 f_R 中从 v 到 u 的每条有向流路径 $f_{(vu)\,j}$，将 $f_{(vu)\,j}$ 替换为 $\hat{}\, f_{(vu)\,j}$。

还需证明 f_S 的拥塞至多为 $1/\alpha$。我们通过考察不同类型的边 (v, e')、$(e'' v)$ 和 $(e'e'')$ 来证明这一点，其中 $v \in V$，而 $\{e'e''\}$ 是某个超边 $e \in \mathrm{E}$ 的 CB 小工具中的两个辅助节点。首先注意到，这一流反转过程不会改变边 $(e'e'')$ 上的流，因此辅助节点之间的边上的拥塞保持不变。另外两条边 (v, e') 或 $(e'' v)$ 具有相同的权重，并且由于 f_R 无环，f_R 中至多有一条边上的流量为正。如果 (v, e') 在 f_R 中具有正流量，则 $(e'' v)$ 在 f_R 中没有流量。如果流反转过程改变了任何内容，它会将通过 (v, e') 的部分流量转移到 $(e'' v)$。在这种情况下，边 (v, e') 上的拥塞不会变得更糟，因为我们只移除了流量。与此同时，边 $(e'' v)$ 起初没有流量，随后接收了来自 (v, e') 的部分流量。经过流转移过程后，边 $(e''v)$ 上的拥塞不会超过 $1/\alpha$，因为它与边 (v, e') 具有相同的权重，而 (v, e') 上的拥塞至多为 $1/\alpha$。在 $(e'' v)$ 在 f_R 中具有正流量、但 (v, e') 没有流量的情况下，我们可以给出类似的论证。因此，这一流反转过程不会使拥塞变得更糟。

B 实验与实现

本附录部分提供了有关算法实现和实验结果的更多详细信息。

B.1 基于凸松弛的算法

在第 6.1 节中，我们将我们的方法与之前基于 SDP 的方法进行比较 [35] 以及之前基于 LP 的方法进行比较 [23]。在这些实验中，我们重点研究全有或全无超图割惩罚和标准扩展目标（$\pi(v) = 1$ 对于每个 $v \in V$）。我们考虑的所有超图均为无权超图。

线性规划方法的实现细节。Kapralov 等人 [24]主要研究超图的谱稀疏化器；关于超图最稀疏割的 $O(\log n)$近似算法的细节

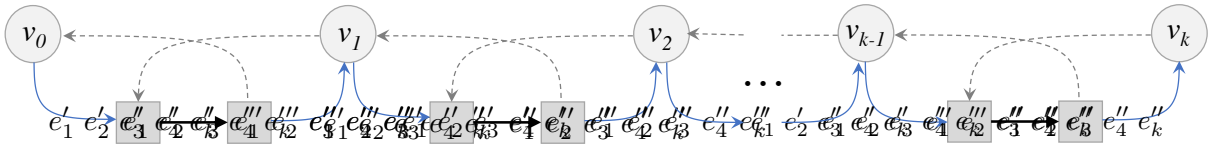


图7：约化图 G（H）**中任意一条连接两个节点的有向路径，都会交替经过** V**中的节点以及来自不同 CB 小工具的辅助节点对** $\{e_{vi}, e_{wi}\}$**。如果原始流函数将流从** v_0 **发送到** v_k**，把蓝色实线边上的流替换为灰色虚线边上的流，就会反转流的方向。可以证明，以这种方式反转流路径不会增加多商品流的拥塞。**

仅收录于论文的完整 arXiv 版本中 [23]。该算法针对导通率目标函数（$\pi(v) = d_v$对每个 $v \in V$），但很容易改为针对扩展率（$\pi(v) = 1$对每个 $v \in V$），以便更直接地与 SDP 方法进行比较 [35]。此改动需要在超图$\mathcal{H} = (V, \mathcal{E})$上求解以下线性规划松弛：

$$\begin{array}{lll} \min & \sum_{e\in\mathcal{E}} z_e & \\ \text{subject to} & \sum_{u,v\in V} x_{uv} = n & \\ & z_e \geq x_{uv} & \forall e \in \mathcal{E}; u, v \in e \\ & x_{uv} \leq x_{uv} + x_{vw} & \forall u, v, w \in V \\ & x_{uv} = x_{vu} \geq 0 & \forall u, v \in V \end{array} \tag{19}$$

可以验证，该线性规划的解是最稀疏割目标值的下界：

$$\min_{S\subseteq V} \frac{\mathbf{cut}_{\mathcal{H}}(S)}{|S|} + \frac{\mathbf{cut}_{\mathcal{H}}(S)}{|\bar{S}|}.$$

将线性规划的最优解值除以 2，即可得到超图扩张率的下界：

$$\min_{S\subseteq V; |S|\leq |V|/2} \frac{\mathbf{cut}_{\mathcal{H}}(S)}{|S|}.$$

请记住，这仅适用于 **割**$\mathcal{H}$ 是全有或全无超图割惩罚的情况。通过求解线性规划得到的变量 $\{x_{uv}\}$ 定义了超图节点上的伪度量。利用 Bourgain [9] 的嵌入结果，可以将其舍入为一个割，且失真为 $O(\log n)$。我们使用 Julia 实现该算法，并使用 Gurobi 优化软件求解线性规划松弛。Trevisan 讲义第 11 章 [44] 清晰而简明地概述了 Bourgain 嵌入定理；我们的舍入方法实现遵循了该章的介绍。

SDP 方法的实现细节。我们使用 Mosek 优化软件求解超图扩张的 SDP 松弛 [35]，并以 Matlab 中的 CVX [17] 作为前端。为提高效率，CVX 会在求解对偶问题前自动将原始程序转换为对偶半定规划。这一过程中最耗时的部分无疑是转换为对偶问题。我们还尝试以 Julia 作为前端调用 Mosek SDP 求解器，不先将问题转换为对偶形式，但这样做的效果明显较差。我们还尝试在 Matlab 中使用 SDPT3 求解器，但效果也较差，而且对于许多 Mosek 能够求解的问题，它都无法找到解。

Louis 和 Makarychev 的舍入过程 [35] 针对 SDP 松弛，十分复杂，且并非为实际实现而设计。该算法的细节分散在论文中的多个技术引理和定理中，而且还依赖于先前论文中的黑箱结果，而这些结果又依赖于舍入过程所需但未明确说明的常数。因此，我们采用了一种简单的舍入方案，在实践中仍能取得良好效果。求解 SDP 松弛后，会为每个节点 $u \in V$生成一个向量 $\mathbf{x}_u$，据此我们可以计算每对节点 $(u, v) \in V \times V$的距离变量 $d(u, v) = \|\mathbf{x}_u - \mathbf{x}_v\|_2^2$和相似度分数 $s(u, v) = \mathbf{x}_u^T \mathbf{x}_v$。接着，对于每个节点 $v \in V$，我们根据函数 d和 s，检查距离最近 或最相似 于 v的节点集合的扩展率。更准确地说，对于每个$v \in V$，我们检查形如以下形式的集合的扩展率

$$B(v, r) = \{u : d(u, v) < r\} \text{ and } M(v, r) = \{u : s(u, v) > r\},$$

对于每个产生不同集合的 $r \in \mathbb{R}^+$ 取值。尽管这比理论舍入方案简单得多，但它经常得到与 SDP 下界相匹配的扩展值，或者至少与该下界相差不到 1.3 倍。

B.2 广义超图割实验

在第4.5节中，我们将我们的方法与 IPM 和 CE 进行比较，这两种方法同样能够处理比率目标中的广义超图割。这两种方法都采用以下节点 $v \in V$的超图度定义：

$$d_v = \sum_{e:\, v\in e} \mathbf{w}_e(\{v\}). \tag{20}$$

对于 δ-线性分裂函数（其中 $\delta \geq 1$），该超边度定义等价于计算节点所参与的超16条边的数量，但对于（20）中的分裂函数则并非如此。我们的方法能够适应任意节点权重，因此也使用（20）以便进行比较。

我们在第 6.4 节中考虑的基准超图均来自 UCI机器学习仓库（https://archive.ics.uci.edu/ml/datasets.php）。不同论文中报告的这些超图的节点数量、超边数量以及超边大小总和存在一些细微差异 [18, 31, 56, 58],，这可能是由于将原始数据集转换为超图时所采用的方式略有不同。我们无法找到这些超图的处理版本，因此根据原始 UCI数据集重新构建了它们。我们的超图已收录在 GitHub仓库中。

IPM 和 *CE* 的实现细节。 我们使用先前的 IPM C++ 实现 [31]。我们使用自己的 CE 实现分别计算热启动向量；这些计算不计入报告的 IPM 运行时间。

非齐次超图的团扩展方法（CE）通过将每条超边替换为一个加权团来实现，并使团中的割惩罚与原超边分裂函数所定义的广义割惩罚之间的差异最小化 [30]。若 $\mathbf{w}_e$ 表示 $e \in \mathcal{E}$ 的超边分裂函数，而 $\hat{}\mathbf{w}_e$ 表示投影团中的割惩罚，则目标是选择团中的边权重，使得

$$\mathbf{w}_e(A) \leq \hat{\mathbf{w}}_e(A) \leq C\mathbf{w}_e(A) \text{ for all } A \subseteq e, \tag{21}$$

对于最小可能值 C。随后对约化图应用谱聚类。该方法针对广义超图割的首个实现，是为特定的非基于基数的割惩罚设计的（https://github.com/ lipan00123/InHclustering，附带 [30]）。后来的一个 Matlab 实现，是为特定的基于基数的次模分裂函数设计的（https://github.com/lipan00123/IPM-for-submodular-hypergraphs/clique_exp.m [31]）。

我们提供了一个用 Julia 实现的 CE，该实现适用于任意基于基数的次模分割函数。对于基于基数的函数，超边的团扩展中的所有边都被赋予相同的权重 p，因此 $\hat{}\mathbf{w}_e(A) = p|A||e\setminus A|$。通过检查 $|A|$ 从 1 到 $|e|$的所有可能取值，可以轻松找到使最小值满足 (p 21) 的最优取值的取值 C。将超图投影为图后，可以在约化图中应用谱聚类的几种细微变体。在计算归一化拉普拉斯矩阵时，可以根据约化图中的度进行归一化，也可以根据超图的度进行归一化。我们的代码支持这两种方式。图 5 和 6 展示了根据图的度进行归一化的结果，这与 Li 和 Milenkovic 的原始理论更为一致 [30]。根据超图的度进行归一化时，结果非常相似。在计算归一化拉普拉斯矩阵的第二小特征向量 $\mathbf{v}_2$ 后，谱聚类会检查形如 $\{u \in V : \mathbf{v}_2(u) > r\}$的扫描割，其中所有取值 r都会产生不同的割集。CE 的理论保证基于在约化图中选取使导通率最小的扫描割。沿用 Li 和 Milenkovic 先前的实现 [31]，我们选择原始超图中导通率最小的扫描割。在图中检查导通率的速度更快，因为在遍历所有扫描割时，更新图的割函数更加容易；但在原始超图中检查导通率在实践中会带来更好的结果。对于 Trivago 实验，我们用 Julia 实现的 CE 比之前的 Matlab 实现快 40-50 倍。我们没有检查 Matlab 实现在第 6.4 节超图上的运行时间。