arXiv:2303.05920v2 [cs.DS] 1 Dec 2023

更快的拟阵划分算法

Tatsuya Terao *

摘要

在拟阵划分问题中，给定定义在共同基集 V 上的 n 个元素上的 k 个拟阵 $\mathcal{M}_1 = (V, \mathcal{I}_1), \ldots, \mathcal{M}_k = (V, \mathcal{I}_k)$，我们需要找出基数尽可能大的可划分集合 $S \subseteq V$，记为 p。这里，如果存在 S 的一个划分 $(S_1, \ldots, S_k)$，使得对 $i = 1, \ldots, k$ 有 $S_i \in \mathcal{I}_i$，则称集合 $S \subseteq V$ 为可划分的。1986 年，Cunningham [SICOMP 1986] 提出了一种拟阵划分算法，该算法使用 $O(np^{3/2} + kn)$ 次独立性预言机查询，是此前已知的最佳算法。当 $k \leq n$ 时，这一查询复杂度为 $O(n^{5/2})$。

我们的主要结果是提出一种新的拟阵划分算法，该算法使用 $\tilde{O}(nr^{3/2})$ 次独立性预言机查询，其中 $r = \min\{k, p\}$。当 $k \leq n$ 时，该查询复杂度为 $\tilde{O}(n^{7/3})$，优于先前 Cunningham 算法的查询复杂度。为此，我们提出了一种新方法边回收增广，它借助以下新思路实现：Nguyˆen [2019]和 Chakrabarty-Lee-Sidford-Singla-Wong [在 FOCS 2019] 提出的二分搜索技术的高效˜运用，以及对独立性预言机查询复杂度的细致分析。我们的分析与拟阵交算法的分析有显著不同，因为涉及参数˜ k。我们还提出了一种拟阵划分算法，使用 $O((n+k)\sqrt{p})$ 次秩预言机查询。

1 引言

拟阵划分问题[1] 是组合优化中最基本的问题之一。该问题有时会与拟阵交问题一同被介绍为一个重要的拟阵问题；参见 [Sch03, 第 4 节1–42] 以及 [KV06, 第 13.5 节–6]。在这个问题中，给定定义在公共基集 V上的 k 个拟阵 $\mathcal{M}_1 = (V, \mathcal{I}_1), \ldots, \mathcal{M}_k = (V, \mathcal{I}_k)$，目标是找到基数尽可能大的可划分集合 $S \subseteq V$，记为 p。这里，如果存在一个划分 $(S_1, \ldots, S_k)$，使得 $S_i \in \mathcal{I}_i$对 $i = 1, \ldots, k$成立，我们就称集合 $S \subseteq V$为可划分的。该问题有许多应用，例如拟阵基打包、树和森林的打包与覆盖，以及香农开关博弈。此外还有更多应用；参见 [Sch03, 第 42]节。

为了设计适用于任意拟阵的算法，通常会考虑预言机模型：算法通过预言机访问拟阵。最标准且研究最充分的预言机是独立性预言机，它以集合 $S \subseteq V$ 作为输入，并输出 $S \in \mathcal{I}$ 是否成立。近期一些关于快速拟阵交算法的研究也考虑了一种功能更强的预言机，称为秩预言机，它以集合 $S \subseteq V$ 作为输入，并输出 S的最大基数独立子集的大小。在设计高效算法时，目标是尽量减少拟阵划分算法对这类预言机的访问次数。我们同时考虑独立性预言机模型和秩预言机模型，并针对这两种预言机模型给出最优的查询算法。

*京都大学。电子邮件：ttatsuya@kurims.kyoto-u.ac.jp[1]该 *matroid partitioning problem* 有时简称为 *matroid partition*。拟阵划分也称为*matroid union* 或 *matroid sum*。

拟阵划分问题与拟阵交问题密切相关。实际上，拟阵划分问题与拟阵交问题在多项式时间内等价；参见 [Edm70,Edm03]。

在拟阵交问题中，给定定义在共同基集 V 上的两个拟阵 $\mathcal{M}' = (V, \mathcal{I}')$ 和 $\mathcal{M}'' = (V, \mathcal{I}'')$，其中基集包含 n 个元素；目标是找出一个基数尽可能大的公共独立集 $S \in \mathcal{I}' \cap \mathcal{I}''$，其基数记为 r。

自 Edmonds [Edm68,Edm79,Edm03] 于20世纪60年代开创这项研究以来，人们一直在研究具有多项式查询复杂度的拟阵交问题算法 [AD71, Law75, Cun86, ˜LSW15, CLS+19, BvdBMN21, Bli21, BMNT23]。Nguyˆen [Ngu19] 和 Chakrabarty-Lee-Sidford- Singla-Wong [CLS+19] 独立提出了一种出色的新型二分搜索技术，可˜在交换图中查找边，并提出了首个使用 $O(nr)$ 次独立性预言机查询[2]的组合算法。Chakrabarty 等人 [CLS+19] 还提出了一种使用 $\tilde{O}(n^{1.5}/\epsilon^{1.5})$ 次独立性预言机查询的 $(1-\epsilon)$ 近似拟阵交算法。Blikstad-van den Brand-Mukhopadhyay-Nanongkai [BvdBMN21] 开发了一种快速算法来解决图可达性问题，并将其与此前的精确算法和近似算法相结合，打破了 $\tilde{O}(n^2)$ 次独立性预言机查询界。Blikstad [Bli21] 改进了近似拟阵交算法的独立性查询复杂度。由此得到了一种使用 $\tilde{O}(nr^{3/4})$ 次独立性预言机查询的随机化拟阵交算法；目前，对于 r 的完整取值范围，这是拟阵交问题的最佳算法。由此还得到了一种使用 $\tilde{O}(nr^{5/6})$ 次独立性预言机查询的确定性拟阵交算法；目前，对于 r 的完整取值范围，这是拟阵交问题的最佳确定性算法。

我们可以通过将拟阵划分问题归约到拟阵交问题来求解。一个著名的归约将拟阵划分归约为基集大小为 kn 的拟阵交问题。其中，所得拟阵交实例的一个输入拟阵是 k 个拟阵的直和。这会导致拟阵划分算法使用过多的独立性预言机查询。即使采用目前最优的拟阵交算法，朴素归约仍会导致拟阵划分算法使用 $\tilde{O}(k^2np^{3/4})$ 次独立性预言机查询。鉴于拟阵划分问题本身十分重要，且有多种应用，关注拟阵划分问题的查询复杂度是有意义的。

Edmonds 于 1968年[Edm68] 首次提出了拟阵划分问题的直接算法。文献中已研究了具有多项式查询复杂度的拟阵划分问题算法 [Knu73,GM75,Cun86, GW88,FM12,BMNT23,Qua23]。

Cunningham [Cun86] 设计了一种拟阵划分算法，需要进行 $O(np^{3/2} + kn)$ 次独立性预言机查询。该算法采用阻塞流 方法，与 Hopcroft-Karp 二分图匹配算法或 Dinic 最大流算法类似。当 $k \leq n$ 时，Cunningham 算法的独立性查询复杂度为 $O(n^{5/2})$。注意，$p \leq n$ 显然成立。近四十年来，这是拟阵划分问题的最佳算法。我们利用近期为快速拟阵交算法开发的技术，研究更快的拟阵划分算法。

我们的首个结果是以下定理，它结合了 Cunningham 的技术和 [Ngu19,CLS+19] 提出的二分搜索技术。

˜**定理 1.1** (详见定理 3.1)。 存在一种算法，只需进行 $O(kn\sqrt{p})$ 次独立性预言机查询，即可解决拟阵划分问题。

定理 1.1 中给出的算法的独立性查询复杂度优于 Cunningham 算法 [Cun86] 在 k 较小时的复杂度。然而，当 $k = \Theta(n)$时，

[2]该 $\tilde{O}$ 记号省略了关于 $\log n$ 的多项式因子。

定理 1.1 中给出的算法的独立性查询复杂度为 $\tilde{O}(n^{5/2})$，并不严格低于 Cunningham 算法的复杂度。

当 $k \leq n$ 时，我们有时会用单变量函数 n 来界定查询次数；这里回顾一下， $p \leq n$。这使得比较不同算法的查询复杂度变得容易。

我们的第二项结果是，获得一种算法，即使在 k 很大时，其性能也严格优于 Cunningham 算法。

定理 1.2 （详见定理 4.1）。 存在一种算法，使用 $\tilde{O}\left(k'^{1/3}np + kn\right)$ 次独立性预言机查询来解决拟阵划分问题，其中 $k' = min\{k, p\}$。

这是本文的主要贡献。定理 1.2 中算法的独立性查询复杂度在 $k = \omega\left(p^{3/4}\right)$ 时优于定理 1.1 中算法的独立性查询复杂度。当 $k \leq n$ 时，定理 1.2 中算法的独立性查询复杂度为 $\tilde{O}\left(n^{7/3}\right)$。这优于 Cunningham 在 [Cun86] 中提出的算法，也优于我们在定理 1.1 中给出的算法。需要强调的是，这是自 1986 年以来的首次改进。图 1 总结了适用于 k 各个取值范围的独立性查询复杂度新旧上界。如图所示，除紧界对应的取值范围外，对于 k 的所有取值范围，我们都得到了比 Cunningham 算法更快的拟阵划分算法。需要指出的是，除独立性预言机查询外，该算法的时间复杂度为 $O\left(k'^{2/3}np\right)$。

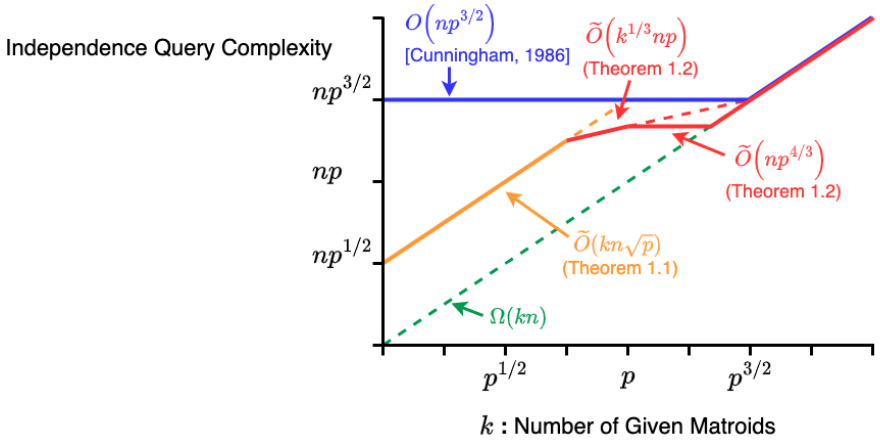


图 1：求解拟阵划分问题的独立性查询复杂度的当前最佳界。绿色虚线表示一个 Ω(kn) 下界；详情见第 5 节。

我们还考虑秩预言机模型下的查询复杂度。注意，秩预言机至少与独立性预言机一样强大。

定理 1.3 (详情见定理 3.7)。 存在一种算法，使用 $O((n+k)\sqrt{p})$ 次秩预言机查询即可解决拟阵划分问题。

定理 1.3 中给出的算法的秩查询复杂度为 $\tilde{O}(n^{3/2})$，其中 $k \leq n$。

1.1 技术概述

Cunningham 的拟阵划分算法。称为交换图的辅助图，在几乎所有拟阵交的组合算法中都发挥着重要作用。在拟阵

交算法中，我们从空集开始，反复沿交换图中的最短路径增广，从而增加独立集的大小。同样，Knuth [Knu73]和 Greene-Magnanti [GM75] 利用具有$O(np)$条边的辅助图给出了拟阵划分算法，我们称之为压缩交换图。

为提高运行时间，Cunningham [Cun86]为拟阵划分和交开发了阻塞流方法，该方法类似于 Hopcroft-Karp [HK73] 的二分图匹配算法。阻塞流方法应用于算法的每个阶段。在 Hopcroft-Karp 的二分图匹配算法中，我们找到一组极大的点不相交最短路径，并沿这些路径同时进行增广。与此不同，在拟阵划分算法中，增广无法并行进行，因为一次增广可能会改变压缩交换图。Cunningham 指出，我们可以找到多条长度相同的增广路径，并在一个阶段内完成所有增广。在 Cunningham 的拟阵划分算法中，每个阶段只需进行 $O(np)$ 次独立性预言机查询（每条边在一个阶段内只查询一次）。

Cunningham 指出，算法执行过程中最短增广路径的不同长度数量为 $O(\sqrt{p})$，因此阶段数为 $O(\sqrt{p})$。所以，Cunningham 的拟阵划分算法总共需要进行 $O(np^{3/2}+kn)$ 次独立性预言机查询（枚举所有进入汇点的边需要进行 $O(kn)$ 次独立性预言机查询）。我们注意到，当 $k \le n$ 时，这一查询复杂度为 $O(n^{5/2})$。

结合阻塞流方法与二分搜索子程序。为了开发定理 1.1 中给出的拟阵划分算法，使其使用 $O(kn\sqrt{p})$ 次独立性预言机查询，我们结合了 Cunningham [Cun86] 提出的阻塞流方法，以及 [Ngu19, CLS+19]提出的二分搜索过程。通过使用二分搜索过程，我们得到了一个使用 $O(kn)$ 次独立性预言机查询、并在压缩交换图中执行广度优先搜索的算法。我们还得到了一个使用 $O(kn)$ 次独立性预言机查询、并在单个阶段内执行所有增广的算法。由于 Cunningham 证明了阶段数为 $O(\sqrt{p})$，我们可以轻松得到一个使用 $O(kn\sqrt{p})$ 次独立性预言机查询的拟阵划分算法。从某种意义上说，我们的算法没有技术上的新意，因为它只是简单地结合了 Cunningham 的技术和 [Ngu19, CLS+19]提出的二分搜索技术。尽管如此，这项成果仍然很重要，因为我们改进了拟阵划分算法的独立性查询复杂度。

边回收增广。 在广度优先搜索中，我们需要检查所有顶点$v \in V$ 和所有下标 $i \in [k]$，判断压缩交换图中是否存在一条从顶点 v 到S_i 中的某个顶点的边。注意，当前可分割集合 S的划分为 (S_1、.....、 S_k)。因此，即使使用二分搜索过程，对压缩交换图进行广度优先搜索的独立性查询复杂度似乎也是 $\Omega(\min\{kn, np\})$；详情请参见第 3 节。目前尚不清楚我们能否开发出一种拟阵划分算法，使其执行广度优先搜索 $o(\sqrt{p})$ 次。因此，采用阻塞流方法的算法目前只能做到 $\Omega(\min\{k,p\}n\sqrt{p})$ 次独立性预言机查询。在 $k=\Theta(n)$ 且 $p=\Theta(n)$ 的情况下，即使使用出色的二分搜索过程，采用阻塞流方法的算法也只能做到 $\Omega(n^{5/2})$ 次独立性预言机查询。

为了打破这一 $O(n^{5/2})$-独立性预言机查询界，我们引入一种新方法**边回收增广** ，并开发了一种独立性查询复杂度关于 k 呈次线性的拟阵划分算法。随后，我们提出一种拟阵划分算法，当 $k \le n$ 时，该算法使用 $\tilde{O}(n^{7/3})$ 次独立性预言机查询。

我们新的方法 与阻塞流方法相同，算法的每个阶段都会应用边回收增广

。在边回收增广的一个阶段中，我们首先计算压缩交换图中的边集 E^*，这需要进行 $O(np)$ 次独立性预言机查询。然后，我们只需重复执行广度优先搜索，在压缩交换图中寻找一条最短路径。执行该广度优先搜索时会利用 E^* 的信息。预先计算 E^* 似乎代价过高，因为我们有出色的二分搜索工具来查找压缩交换图中的边。然而，在重复执行广度优先搜索期间，我们可以回收 E^* 中的一些边，这对分析我们提出的新方法至关重要。请注意，边回收增广会在每次增广前执行一次广度优先搜索，而阻塞流方法只会在每个阶段开始时执行一次广度优先搜索。

我们的关键观察是，除非通过增广更新了 S_i，否则所有进入 S_i 的边都不会改变。回想一下，当前可划分集合 S 的划分为（S_1、.....、S_k）。因此，即使经过若干次增广，我们仍可使用 E^* 找到进入顶点 $u \in S_i$ 的边，其中 S_i 未通过增广更新。这一观察是拟阵划分所特有的。在边回收增广方法的广度优先搜索中，我们仅在寻找进入 $u \in S_i$ 的边时使用二分搜索过程，其中 S_i 已通过增广更新。在一个阶段中，我们重复执行广度优先搜索，使二分搜索过程调用的总次数为 $O(np)$。

我们结合阻塞流方法算法与边回收增广方法算法。通过仔细分析独立性查询复杂度，我们得到一种拟阵划分算法，其使用 $\tilde{O}(k'^{1/3}np + kn)$ 次独立性预言机查询，其中 $k' = \min\{k, p\}$.

请注意，这种边回收增广方法与现有的快速拟阵交算法有显著差异。本文的关键技术贡献是提出了这种新方法。

1.2 相关工作

Blikstad-Mukhopadhyay-Nanongkai-Tu [BMNT23] 提出了一种称为动态~预言机的新预言机模型，并开发了一种拟阵划分算法，该算法使用 $O((n+r\sqrt{r})\cdot\text{poly}(k))$ 次动态秩查询。~~其中 $r = \max_i \max_{S_i \in \mathcal{I}_i} |S_i|$。他们还获得了一~~种算法，可在 $O(n\sqrt{r})$时间内解决 k重拟阵~并问题，并使用动态秩查询。这是拟阵划分问题的一个特殊情形，其中所有拟阵 $\mathcal{M}_1, \ldots, \mathcal{M}_k$ 都相同。Quanrud [Qua23] 开发了一种算法，可解决 k重拟阵并问题，并在 r 和 k的完整取值范围内使用 $\tilde{O}(n^{3/2})$ 次独立性预言机查询。Quanrud 还在更一般的设定下研究了 k重拟阵并问题，其中元素具有整数容量和实值容量。

对于某些特殊拟阵，已有更快的拟阵划分算法。对于线性拟阵，Cunningham [Cun86] 提出了一种运行时间为 $O(n^3 \log n)$的算法，可解决 $O(n)$矩阵上的拟阵划分问题，这些矩阵有 n 列，且行数至多为 n 。对于图拟阵，k-森林问题是拟阵划分问题的一个特例。该问题给定一个无向图和一个正整数 k，目标是求 k个森林的最大规模并集。Gabow-Westermann [GW88] 提出了一种运行时间为 $O(\min\{k^{3/2}\sqrt{nm}(m + n\log\ n)\ k^{1/2}m\sqrt{m + n\log\ n},\ kn^2 \log\ k,\ \widetilde{m^2_k \log k}\})$的算法来解决 k-森林问题~~，~~其中 n 和 ~~m~~分别表示顶点数和边数。Blikstad 等人 [BMNT23] 与 Quanrud [Qua23] 分别独立地提出了一种运行时间为 $\tilde{O}(m + (kn)^{3/2})$的算法来解决 k-森林问题。

Kawase-Kimura-Makino-Sumita [KKMS21] 研究了针对各种目标函数的拟阵划分问题。

对于加权拟阵交，Huang-Kakimura-Kamiyama [HKK19] 提出了一种技术，可将任意无权拟阵交算法转换为求解加权情形的算法，其近似因子为 $O(W)$。Huang等人还提出了一种$(1 - \epsilon)$近似

加权拟阵交算法，使用 $\tilde{O}(nr^{3/2}/\epsilon)$次独立性预言机查询。Chekuri-Quanrud [CQ16] 降低了独立性查询复杂度，并提出了一种$(1-\epsilon)$近似加权拟阵交算法，使用 $O(nr/\epsilon^2)$次独立性预言机查询；通过应用Blikstad [Bli21]近期提出的更快的近似无权拟阵交算法，还可以进一步改进该算法。Tu [Tu22] 提出了一种加权拟阵交算法，使用 $\tilde{O}(nr^{3/4}\log W)$次秩预言机查询；该算法还使用了 [Ngu19, CLS+19]提出的二分搜索过程。

对于秩为 $n/2$ 的拟阵，Harvey [Har08] 给出了拟阵交所需独立性预言机查询次数的下界 $(\log_2 3)n-o(n)$。Blikstad 等人 [BMNT23] 在其动态秩预言机模型和独立性预言机模型中，分别为拟阵交和划分问题给出了超线性的 $\Omega(n\log n)$ 查询下界。

1.3 论文结构

在第 2 节中，我们介绍拟阵划分与拟阵交的符号约定和已知结果。接着，在第 3 节中，我们介绍使用阻塞流方法的拟阵划分算法。然后，在第 4 节中，我们介绍新的边回收增广方法，以及针对较大 k 的更快拟阵划分算法。

2 预备知识

2.1 拟阵

符号约定。对于正整数 a，我们记为 $[a]=\{1,\dots,a\}$。对于有限集 X，令 $\#X$ 和 $|X|$ 表示 X 的基数，也称为 X 的大小。我们通常写作 $A+x:=A\cup\{x\}$ 和 $A-x:=A\setminus\{x\}$。在不会引起混淆时，我们也写作 $A+B:=A\cup B$ 和 $A-B:=A\setminus B$。

拟阵。对于有限集 V 和非空 $\mathcal{I}\subseteq 2^V$，若满足以下性质，则称二元组 $\mathcal{M}=(V,\mathcal{I})$ 为一个拟阵 。

（向下封闭性）若 $S\in\mathcal{I}$ 且 $S'\subseteq S$，则 $S'\in\mathcal{I}$。
（增广性质）若 $S,S'\in\mathcal{I}$ 且 $|S'|<|S|$，则存在 $x\in S\setminus S'$ 使得$S'+x\in\mathcal{I}$。

若集合 $S\subseteq V$ 满足 $S\in\mathcal{I}$，则称其为独立 集；否则称其为相关 集。

秩。对于拟阵 $\mathcal{M}=(V,\mathcal{I})$，我们将 $\mathcal{M}$ 的秩 定义为 $\mathrm{rank}(\mathcal{M})=\max\{|S|\mid S\in\mathcal{I}\}$。此外，对于任意 $S\subseteq V$，我们将 S 的秩 定义为 $\mathrm{rank}_{\mathcal{M}}(S)=\max\{|T|\mid T\subseteq S,T\in\mathcal{I}\}$。

拟阵交。给定两个拟阵 $\mathcal{M}'=(V,\mathcal{I}')$、$\mathcal{M}''=(V,\mathcal{I}'')$，我们将它们的交 定义为 $(V,\mathcal{I}'\cap\mathcal{I}'')$。拟阵交问题 要求我们找出最大的公共独立集，其基数记为 r。请注意，拟阵的交通常不是拟阵；而求解两个以上拟阵的最大公共独立集问题是 NP 困难。

拟阵划分（拟阵并）。给定 k 个拟阵 $\mathcal{M}_1=(V,\mathcal{I}_1),\dots,\mathcal{M}_k=(V,\mathcal{I}_k)$，$S\subseteq V$ 若存在 S 的一个划分 $(S_1,\dots,S_k)$，使得 $S_i\in\mathcal{I}_i$ 对 $i\in[k]$ 成立，则称 $S\subseteq V$ 为可划分的。拟阵划分问题要求我们找出最大的可划分集合，其基数记为 p。令 $\mathcal{I}$ 为 V 的可划分子集族。那么，$(V,\mathcal{I})$ 称为

k 个拟阵 $\mathcal{M}_1 \ldots,\ \mathcal{M}_k$ 的并或和。注意，纳什-威廉姆斯定理 [NW66] 指出，这些拟阵的并 $(V,\ \mathcal{I})$ 也是一个拟阵。

预言机。 在本文中，我们假设只能通过预言机访问拟阵 $\mathcal{M} = (V, \mathcal{I})$。给定一个子集 $S \subseteq V$，独立性预言机将输出 $S \in \mathcal{I}$ 是否成立。给定一个子集 $S \subseteq V$，秩预言机将输出秩$_{\mathcal{M}}(S)$。由于一次秩预言机查询就能确定给定子集是否独立，因此秩预言机比独立性预言机更强大。

二分搜索技术。 Chakrabarty-Lee-Sidford-Singla-Wong [CLS+19] 指出，可以在独立性预言机模型中使用二分搜索高效地实现以下过程。（Nguyễn [Ngu19]也独立地提出了这一方法。）给定一个拟阵 $\mathcal{M} = (V, \mathcal{I})$、一个独立集 $S \in \mathcal{I}$、一个元素 $v \in V \setminus S$以及 $B \subseteq S$，目标是找到一个与 v 可交换的元素 $u \in S$（即 $S + v - u \in \mathcal{I}$），或者得出不存在这样的元素。本文省略证明；证明见 [CLS+19 的第 3] 节。

引理 2.1 （通过二分搜索进行边搜索，[Ngu19,CLS+19]）。存在一个算法 `FindOutEdge`，给定一个拟阵 $\mathcal{M} = (V, \mathcal{I})$、一个独立集 $S \in \mathcal{I}$、一个元素 $v \in V \setminus S$ 和 $B \subseteq S$，能够找到一个元素 $u \in B$，使得 $S + v - u \in \mathcal{I}$；或者确定不存在这样的元素，并使用 $O(\log |B|)$ 次独立性查询。此外，如果不存在这样的元素，则过程 `FindOutEdge` 仅使用一次独立性查询。[3]

与引理 2.1 类似，Chakrabarty 等人 [CLS+19] 也提出了一种技术，可以在秩预言机模型中高效地找到一个可交换元素，或者判定不存在这样的元素；证明见 [CLS+19，第 3] 节。

引理 2.2 （通过二分搜索进行边搜索，[CLS+19]）。存在一个算法 `FindInEdge`，给定一个拟阵 $\mathcal{M} = (V, \mathcal{I})$、一个独立集 $S \in \mathcal{I}$、一个元素 $u \in S$ 和一个子集 $A \subseteq V \setminus S$，能够找到一个元素 $v \in A$，使得 $S + v - u \in \mathcal{I}$；或者确定不存在这样的元素，并使用 $O(\log |A|)$ 次秩查询。

2.2 拟阵交的技术

这里介绍有关拟阵交的已知结果。

定义 2.3 (交换图)。 考虑一个公共独立集 $S \in \mathcal{I}' \cap \mathcal{I}''$。交换图 定义为有向图 $G(S) = (V \cup \{s, t\}, E)$，其中 s、$t \notin V$ 和 $E = E' \cup E'' \cup E_s \cup E_t$，其中

$$
\begin{aligned}
E' &= \{(u, v) \mid u \in S, v \in V \setminus S, S - u + v \in \mathcal{I}'\},\\
E'' &= \{(v, u) \mid u \in S, v \in V \setminus S, S - u + v \in \mathcal{I}''\},\\
E_s &= \{(s, v) \mid v \in V \setminus S, S + v \in \mathcal{I}'\}, \text{and}\\
E_t &= \{(v, t) \mid v \in V \setminus S, S + v \in \mathcal{I}''\}.
\end{aligned}
$$

引理 2.4 （最短增广路径 [Sch03, 定理 41。2]）。设 $s, v_1, v_2, \ldots, v_{l-1}, t$ 是交换图 $G(S)$ 中相对于公共独立集 $S \in \mathcal{I}' \cap \mathcal{I}''$的一条最短$(s, t)$路径。那么$S' = S + v_1 - v_2 + \cdots - v_{l-2} + v_{l-1} \in \mathcal{I}' \cap \mathcal{I}''$。

[3]当且仅当 $S + v - B \in \mathcal{I}$ 时，存在这样的元素。

在拟阵交算法中，我们从空集 S开始。然后反复在交换图 $G(S)$中寻找增广路径，并更新当前集合 S。如果交换图 $G(S)$中不存在 (s,t)-路径，那么 S 就是一个最大规模的公共独立集。如果交换图 $G(S)$中存在一条 (s,t)-路径，那么我们选取一条最短路径，从而得到一个包含 $|S|+1$ 个元素的公共独立集 $S' \in \mathcal{I}' \cap \mathcal{I}''$。

Cunningham拟阵交算法 [Cun86] 以及近期更快的拟阵交算法 [CLS+19、Ngu19、BvdBMN21、Bli21] 依赖以下两个引理。

引理2.5 ([Cun86]). 对于任意两个拟阵 $\mathcal{M}' = (V, \mathcal{I}')$ 和 $\mathcal{M}'' = (V, \mathcal{I}'')$，若相对于公共独立集 $S \in \mathcal{I}' \cap \mathcal{I}''$，交换图 $G(S)$ 中的最短增广路径长度至少为 d，则 $|S| \geq (1 - O_{(1)}(1/d)) \cdot r$，其中 r 是最大公共独立集的大小。

引理2.6 (单调性引理 [Cun86、Has08、Pri15、CLS+19]). 对于任意两个拟阵 $\mathcal{M}' = (V, \mathcal{I}')$ 和 $\mathcal{M}'' = (V, \mathcal{I}'')$，假设我们沿着 $G(S)$中的最短增广路径进行增广 $S \in \mathcal{I}' \cap \mathcal{I}''$，从而得到一个公共独立集 $S' \in \mathcal{I}' \cap \mathcal{I}''$。注意 $|S'| > |S|$。令 d 表示 $G(S)$ 中的距离，d' 表示 $G(S')$中的距离。那么对于所有 $v \in V$，

(i) 若 $d(s,v) < d(s,t)$，则 $d'(s,v) \geq d(s,v)$。若 $d(v,t) < d(s,t)$，则 $d'(v,t) \geq d(v,t)$。

(ii) *If* $d(s,v) \geq d(s,t)$, *then* $d'(s,v) \geq d(s,t)$. *If* $d(v,t) \geq d(s,t)$, *then* $d'(v,t) \geq d(s,t)$.

2.3 拟阵划分的压缩交换图

ˆ 拟阵划分问题可以通过拟阵交算法求解。令 $V = V \times [k]$，并定义

$$\hat{\mathcal{I}}' = \{\hat{I} \subseteq \hat{V} \mid \forall v \in V, \#\{i \in [k] \mid (v,i) \in \hat{I}\} \leq 1\},$$
$$\hat{\mathcal{I}}'' = \{\hat{I} \subseteq \hat{V} \mid \forall i \in [k], \{v \in V \mid (v,i) \in \hat{I}\} \in \mathcal{I}_i\}.$$

那么，$\hat{\mathcal{M}}' = (\hat{V}, \hat{\mathcal{I}}')$ 是一个分划拟阵。由于 $\hat{\mathcal{M}}'' = (\hat{V}, \hat{\mathcal{I}}'')$ 是对所有 $i \in [k]$ 的拟阵 $(V, \mathcal{I}_i)$ 的直和，因此它也是一个拟阵。那么，V 的可划分子集族可以表示为

$$\{S \subseteq V \mid \exists \pi \colon S \to [k], \{(v, \pi(v)) \mid v \in S\} \in \hat{\mathcal{I}}' \cap \hat{\mathcal{I}}''\}.$$

因此，我们可以通过计算 $\mathcal{I}'$ 和 $\mathcal{I}''$ 中最大规模的公共独立集，来解决拟阵划分问题。然而，使用这一归约将拟阵划分问题转化为拟阵交问题来求解时，可能会使用过多的独立性预言机查询。这是因为以下原因：在求解由拟阵划分问题归约得到的拟阵交问题时，该拟阵交问题的基集大小为 $O(kn)$，交换图中的边数为 $O(knp)$，这在很大程度上取决于 k。此外，由于我们考虑对所有 $i \in [k]$ 的拟阵 $\mathcal{M}_i = (V, \mathcal{I}_i)$ 的独立性预言机的总查询复杂度，拟阵 $\hat{\mathcal{M}}'' = (\hat{V}, \hat{\mathcal{I}}'')$ 的独立性查询复杂度也在很大程度上取决于 k。

然后，为了提升运行时间，Knuth [Knu73] 和 Greene-Magnanti [GM75] 提出了一种拟阵划分算法，该算法使用以下含有 $O(np)$ 条边的辅助图，我们称之为压缩交换图。

定义 2.7 (压缩交换图 [Knu73,GM75,Cun86])。 考虑 $S \subseteq V$ 的一个划分$(S_1, \ldots, S_k)$，满足对于所有 $i \in [k]$, $S_i \in \mathcal{I}_i$ 。压缩交换图 定义为有向图 $G(S_1, \ldots, S_k) = (V \cup \{s, t_1, \ldots, t_k\}, E)$，其中 $s, t_1, \ldots, t_k \notin V$ 和 $E = E' \cup E_s \cup E_t$，其中

$$\begin{aligned} E' =& \{(v,u) \mid \exists i \in [k], u \in S_i, S_i + v \notin \mathcal{I}_i, S_i + v - u \in \mathcal{I}_i\}, \\ E_s =& \{(s,v) \mid v \in V \setminus S\}, and \\ E_t =& \bigcup_{i=1}^{k} \{(v,t_i) \mid v \in V \setminus S_i, S_i + v \in \mathcal{I}_i\}. \end{aligned}$$

我们记作 $T = \{t_1, \ldots, t_k\}$。

在拟阵划分算法中，我们从空集 S 开始，并对所有 $i \in [k]$ 初始化 $S_i = \emptyset$ 。如果在压缩交换图 $G(S_1, \ldots, S_k)$ 中，没有任何顶点可从 s 到达 T，那么 S 是最大规模的可划分集合。如果在压缩交换图中存在一条从 s 到 T 的路径，那么我们选取一条最短路径 $s, v_1, \ldots, v_{l-1}, t_j$。于是，我们可以得到一个可划分集合 $S' = S + v_1$ 以及 S' 的一个划分 $(S'_1, \ldots, S'_k)$，使得对所有 $i \in [k]$，都有 $S'_i \in \mathcal{I}_i$。该算法的正确性由以下两个引理保证，本文将反复使用这两个引理。Cunningham [Cun86] 利用拟阵划分的压缩交换图与约化拟阵交的交换图之间的等价性证明了这些引理；直接证明（不使用拟阵交归约）见 [Sch03, Theorem 42.4] 。

引理 2.8。 给定 S 的一个划分$(S_1, \ldots, S_k)$，且对所有 $i \in [k]$ 都有 $S_i \in \mathcal{I}_i$，存在一个大小至少为 $|S| + 1$ 的可分割集合 S'，当且仅当在压缩交换图 $G(S_1, \ldots, S_k)$ 中，存在一个从 s 可达的顶点 $t_j \in T$。

引理 2.9 (最短增广路径)。 设 $s, v_1, v_2, \ldots, v_{l-1}, t_j$ 是压缩交换图 $G(S_1, \ldots, S_k)$ 中一条最短的(s, T)路径。那么 $S' = S + v_1$ 是一个可分割集合。

我们可以利用以下过程 Update（算法 1），根据 S 的一个划分 $(S_1, \ldots, S_k)$，以及压缩交换图中的一条增广路径，构造 S' 的一个划分 $(S'_1, \ldots, S'_k)$。

算法 1: 更新

输入: $S (\subseteq V)$ 的一个划分 $(S_1, \ldots, S_k)$，满足对所有 $i \in [k]$，$S_i \in \mathcal{I}_i$，以及一条增广路径 $s, v_1, \ldots, v_{l-1}, t_j$

输出: $S' (\subseteq V)$ 的一个划分 $(S'_1, \ldots, S'_k)$，满足对所有 $i \in [k]$，$S'_i \in \mathcal{I}_i$，并且 $S' = S + v_1$

对所有 $i \in [k]$，令 $S'_i \leftarrow S_i$

对所有 $v \in S$，将满足 $v \in S_{\pi(v)}$ 的指标记为 $\pi(v)$

对于 $i \in [l-2]$ **执行**

  $S'_{\pi(v_{i+1})} \leftarrow S'_{\pi(v_{i+1})} + v_i - v_{i+1}$

$S'_j \leftarrow S'_j + v_{l-1}$

返回 S' 的一个划分（$S'_1, \ldots, S'_k$）

Cunningham [Cun86] 指出，两个拟阵 $\hat{\mathcal{M}}' = (\hat{V}, \hat{\mathcal{I}}')$ 和 $\hat{\mathcal{M}}'' = (\hat{V}, \hat{\mathcal{I}}'')$ 的拟阵交的交换图，与 k 拟阵 $(V, \mathcal{I}_1), \ldots, (V, \mathcal{I}_k)$ 的拟阵划分的压缩交换图之间存在等价关系，并利用这一点证明引理 2.10 和 2.11，进而开发出一种采用阻塞流方法的高效拟阵划分算法。对于 S 的一个固定划分 $(S_1, \ldots, S_k)$ 和一个元素 $v \in S$，令 $\pi(v)$ 为满足 $v \in S_{\pi(v)}$ 的指标。我们还用 $\hat{S}$ 表示集合 $\{(v, \pi(v)) \in \hat{V} \mid v \in S\}$。

拟阵划分的压缩交换图中的一条路径 $s, v_1, v_2, \ldots, v_{l-1}, t_j$，对应于拟阵交的交换图中的一条路径 $s, (v_1, \pi(v_2)), (v_2, \pi(v_2)), (v_2, \pi(v_3)) \ldots, (v_{l-1}, \pi(v_{l-1})), (v_{l-1}, j)\ t$。于是，对于所有元素 $v \in S$，有 $d_{G(S_1,\ldots,S_k)}(s, v) = 1 + \frac{1}{2} d_{G(\hat{S})}(s, (v, \pi(v)))$，并且 $d_{G(S_1,\ldots,S_k)}(v, T) = \frac{1}{2} d_{G(\hat{S})}((v, \pi(v))\ t)$。我们还有 $d_{G(\bar{S}_1,\ldots,S_k)}(s, T) = 1 + \frac{1}{2} d_{G(\hat{S})}(s, t)$。

Cunningham [Cun86] 利用以下两个引理，通过阻塞流方法设计了一种高效的拟阵划分算法。这些引理可由交换图与压缩交换图之间的对应关系得出。注意，引理 2.10 和 2.11 分别对应引理 2.5 和 2.6。我们也在快速拟阵划分算法中使用这两个引理。

引理 2.10 ([Cun86])。给定 S 的一个划分 $(S_1, \ldots, S_k)$，满足对所有 $i \in [k]$ 均有 $S_i \in \mathcal{I}_i$。如果压缩交换图 $G(S_1, \ldots, S_k)$ 中的最短增广路径长度至少为 d，那么 $|S| \geq (1 - O_{(1)\, d}) \cdot p$，其中 p 表示最大可划分集合的大小。 ——

引理 2.11 (单调性引理 [Cun86,Has08,Pri15,CLS+19])。假设我们沿着 $G(S_1, \ldots, S_k)$ 中的一条最短增广路径，对 S 的一个划分 $(S_1, \ldots, S_k)$ 进行增广，从而得到 S' 的一个划分 $(S'_1, \ldots, S'_k)$。注意，$|S'| > |S|$。令 d 表示 $G(S_1, \ldots, S_k)$ 中的距离，令 d' 表示 $G(S'_1, \ldots, S'_k)$ 中的距离。那么对于所有 $v \in V$，

(i) *If* $d(s, v) < d(s, T)$, *then* $d'(s, v) \geq d(s, v)$. *If* $d(v, T) < d(s, T)$, *then* $d'(v, T) \geq d(v, T)$.

(ii) *If* $d(s, v) \geq d(s, T)$, *then* $d'(s, v) \geq d(s, T)$. *If* $d(v, T) \geq d(s, T)$, *then* $d'(v, T) \geq d(s, T)$.

稍后我们会看到，在独立性预言机模型下，我们使用引理 2.1 中给出的二分搜索技术，在压缩交换图中查找边。请注意，FindOutEdge($\mathcal{M}_i$, S_i, v, B) 过程为我们提供了一种高效的方法，用于在压缩交换图中查找从顶点 v 到顶点 $u \in B$（$\subseteq S_i$）的边。

我们还使用引理 2.2 中给出的二分搜索技术，在秩预言机模型下查找压缩交换图中的边。请注意，FindInEdge($\mathcal{M}_i$, S_i, u, A) 过程为我们提供了一种高效的方法，用于在压缩交换图中查找从顶点 $v \in A$ 到顶点 $u \in S_i$ 的边。

与引理 2.2 中的做法相同，给定一个拟阵 $\mathcal{M} = (V, \mathcal{I})$、一个独立集 $S \in \mathcal{I}$ 和一个子集 $A \subseteq V \setminus S$，我们可以使用 O 次秩查询（$\log |A|$）找到一个元素 $v \in A$，使得 $S + v \in \mathcal{I}$；或者确定不存在这样的元素。请注意，FindInEdge($\mathcal{M}_j$, S_j, $\emptyset$, A) 过程为我们提供了一种高效的方法，用于在压缩交换图中查找从 $v \in A$ 到顶点 $t_j \in T$ 的边。

3 阻塞流算法

本节将介绍我们在独立性预言机模型和秩预言机模型下的拟阵划分算法。这些算法只需结合 Cunningham [Cun86] 提出的阻塞流方法，以及 [Ngu19,CLS+19] 提出的二分搜索过程即可得到。在第 3.1 节和第 3.2 节中，我们将分别介绍独立性预言机模型和秩预言机模型下的拟阵划分算法。

3.1 使用独立性预言机的阻塞流算法

本小节介绍我们在独立性预言机模型中采用阻塞流方法的拟阵划分算法。我们证明以下定理，由此可推出定理 1.1。

定理 3.1。 存在一种算法，使用 $O\left((kn+p\text{对数 }p)\sqrt{p}\right)$ 次独立性预言机查询即可解决拟阵划分问题。

这一结果改进了 Cunning- ham [Cun86] when $k=o(p)$此前已知的拟阵划分算法。

为证明这一点，我们首先给出过程 GetDistanceIndependence（算法 2），它能够高效地求出压缩交换图中从 s 到每个顶点的距离。该算法只需使用过程 FindOutEdge 执行广度优先搜索。

引理 3.2 （使用独立性预言机的广度优先搜索）。 给定$S\ (\subseteq V)$ 的一个划分$(S_1,\ldots,S_k)$，使得对所有 $i\in[k]$均有 $S_i\in\mathcal{I}_i$，过程 `GetDistanceIndependence` （算法 2）输出 $d\in\mathbb{R}^{V\cup\{s\}\cup T}$，使得对于 $v\in V\cup\{s\}\cup T$， d_v 是压缩交换图 $G(S_1,\ldots,S_k)$中从 s 到 v 的距离。过程 `GetDistanceIndependence` 使用 $O\left(kn+p\log\ p\right)$ 次独立性预言机查询。

证明。 过程 GetDistanceIndependence 只是在压缩交换图 $G(S_1,\ldots,S_k)$ 中执行广度优先搜索。因此，过程 GetDistanceIndependence 能正确计算从 s到 $G(S_1,\ldots,S_k)$ 中各顶点的距离。注意，每个顶点 $v\in V$至多被加入 Q一次，且每个顶点 $v\in S$ 至多从 $B_{\pi(v)}$ 中移除一次。因此，第 11 行使用的独立性预言机查询次数为 $O(kn)$。不输出 $\emptyset$ 的 FindOutEdge 调用次数为 $O(p)$，输出 $\emptyset$ 的 FindOutEdge 调用次数为 $O(kn)$。因此，根据引理 2.1，第 14 行使用的独立性预言机查询次数为 $O\left(kn+p\text{log }p\right)$，证明完毕。

□

算法 2： GetDistanceIndependence

输入： 划分 $(S_1,\ldots,S_k)$ of $S(\subseteq V)$，使得 $S_i\in\mathcal{I}_i$对所有 $i\in[k]$

输出： $d\in\mathbb{R}^{V\cup\{s\}\cup T}$，使得对于 $v\in V\cup\{s\}\cup T$， d_v是从 s到 v的距离 $G\ (S_1,\ldots,S_k)$

```
1  d_s ← 0
2  对于所有 v ∈ V \ S，令 d_v ← 1
3  对于所有 v ∈ S，令 d_v ← ∞
4  对于所有 i ∈ [k] 令 d_{t_i} ← ∞
5  Q ← {v ∈ V \ S} // Q：队列
6  对于所有 i ∈ [k] 令 B_i ← S_i
7  当 Q ≠ ∅ 时执行
8      令 v 为最早添加到 Q 中的元素
9      Q ← Q − v
10     对于 i ∈ [k] 满足 d_{t_i} = ∞ 执行
11         如果 v ∉ S_i 且 S_i + v ∈ I_i 那么
12             d_{t_i} ← d_v + 1
13     对于 i ∈ [k] 具有 v ∉ S_i 执行
14         当 u = FindOutEdge(M_i, S_i, v, B_i) 满足 u ≠ ∅ 执行
15             Q ← Q + u
16             d_u ← d_v + 1
17             B_i ← B_i − u
18 返回 d
```

接下来，我们给出更快的拟阵划分算法所用的增广子程序。我们通过使用 [Ngu19, CLS+19] 提出的二分搜索过程，实现 Cunningham 的 [Cun86]拟阵划分阻塞流方法。该算法类似于拟阵

[CLS+19] 在秩预言机模型中的交集算法。其实现称为 BlockFlowIndependence（算法 3）。

算法 3： BlockFlowIndependence

输入： $S(\subseteq V)$ 的一个划分 $(S_1, \ldots, S_k)$，满足 $S_i \in \mathcal{I}_i$ 对所有 $i \in [k]$

输出： $S'(\subseteq V)$ 的一个划分 $(S'_1, \ldots, S'_k)$，满足 $S'_i \in \mathcal{I}_i$ 对所有 $i \in [k]$，$|S'| > |S|$，以及 $d_{G(S'_1,\ldots,S'_k)}(s,T) \geq d_{G(S_1,\ldots,S_k)}(s,T) + 1$，或者在没有这样的 $d_{G(S'_1,\ldots,S'_k}$ 时，S 的一个划分 $(S_1, \ldots, S_k)$ 这样的 S' 存在。

$d \leftarrow$ GetDistanceIndependence(S_1, ..., S_k)
若 $d_T = \infty$，**则返回** $(S_1, \ldots, S_k)$;
对所有 $v \in V$，令 $I_v \leftarrow [k]$
对所有 $i \in [d_T - 1]$，令 $L_i \leftarrow \{v \in V \mid d_v = i\}$
当 $L_i \neq \emptyset$ 对所有 $i \in [d_T - 1]$ **时**
    $l \leftarrow 0$，$a_{d_T} \leftarrow \emptyset$
    当 $l < d_T$ **时**
        $a_{l+1} \leftarrow \emptyset$
        如果 $l < d_T - 1$，**则**
            如果 $l = 0$，**则**
                任意选择 $v \in L_1$
                $a_1 \leftarrow v$
            else
                当 $I_{a_l} \neq \emptyset$ **时**
                    任意选择 $i \in I_{a_l}$
                    $a_{l+1} \leftarrow$ FindOutEdge($\mathcal{M}_i$, S_i, a_l, $L_{l+1} \cap S_i$)
                    如果 $a_{l+1} = \emptyset$ **则** $I_{a_l} \leftarrow I_{a_l} - i$;
                    否则中断;
        else
            对于 $i \in [k]$ **执行**
                如果 $a_l \notin S_i$ 且 $S_i + a_l \in \mathcal{I}_i$ **则**
                    $a_{l+1} \leftarrow t_i$
                    中断
        if $a_{l+1} = \emptyset$ **则**
            $L_l \leftarrow L_l - a_l$
            如果 $L_l = \emptyset$ **那么中断**;
            $l \leftarrow l - 1$
        else
            $l \leftarrow l + 1$
    如果 $a_{d_T} \neq \emptyset$ **那么**
        用 P 表示路径 s，a_1、.....、a_{d_T-1}，a_{d_T}
        $(S_1, \ldots, S_k) \leftarrow$ 更新$((S_1, \ldots, S_k), P)$
        对于所有 $i \in [d_T - 1]$，令 $L_i \leftarrow L_i - a_i$
返回 $(S_1, \ldots, S_k)$

在过程 BlockFlowIndependence 中，给定 S的一个划分 $(S_1, \ldots, S_k)$，我们首先使用 GetDistanceIndependence（算法 2）计算从 s 到压缩交换图 $G(S_1, \ldots, S_k)$ 中每个顶点的距离。利用这些距离，我们将 V 划分为集合$L_1, L_2, \ldots$，其中每个 L_i都包含满足以下条件的所有顶点 v：在压缩交换图 $G(S_1, \ldots, S_k)$ 中，从 s到 v的距离为 i。然后，我们在压缩交换图 $G(S_1, \ldots, S_k)$ 中搜索一条路径 $s, a_1, a_2, \ldots, a_{d_T-1}, a_{d_T}$，其中对所有 $i \in [d_T - 1]$，$a_i \in L_i$。注意，我们写作

$d_T = \min(d_{t_1}, \ldots\ldots, d_{t_k})$。如果我们找到这样一条路径，就增广 S的一个划分（$S_1, \ldots\ldots, S_k$），并对所有 $i \in [d_T - 1]$从 L_i中移除 a_i。然后我们再次搜索一条新路径，直到找不到长度为d_T 的（s, T）-路径。在搜索此类路径时，如果过程判定 L_i 中的某个顶点不在这样的路径上，就会将该顶点从 L_i中移除。

引理 *3.3*（使用独立性预言机的阻塞流）。给定 S（$\subseteq V$）的一个划分（$S_1, \ldots\ldots, S_k$），使得对所有 $i \in [k]$都有 $S_i \in \mathcal{I}_i$，过程 ***BlockFlowIndependence***（算法 *3*）输出 S'（$\subseteq V$）的一个划分（$S'_1, \ldots\ldots, S'_k$），使得对所有 $i \in [k]$、$|S'| > |S|$都有 $S'_i \in \mathcal{I}_i$，且$d_{G(S'_1,\ldots,S'_k)}(s,T) \geq d_{G(S_1,\ldots,S_k)}(s,T) + 1$；或者在不存在这样的 S'时，输出 S 的一个划分（$S_1, \ldots\ldots, S_k$）。过程 ***BlockFlowIndependence*** 使用 $O(kn + p \log\ p)$次独立性预言机查询。

Cunningham [Cun86] 证明了以下引理；证明见 [Cun86，第 5] 节。我们用此引理证明引理 3.3。

引理 3.4 ([Cun86])。对于最短增广路径 P上的所有顶点 $v \in V$，沿路径 P进行增广后，从s 到 v 的距离必定严格增加。

引理 *3.3* 的证明。我们将 D_i 记作满足以下条件的顶点 $v \in V$的集合：在压缩交换图 $G(S_1, \ldots, S_k)$ 中，从 s 到 v的距离为 i，其中 $(S_1, \ldots, S_k)$ 是 S的输入划分。假设我们沿长度为 d_T的最短增广路径，将 S的划分 $(S_1, \ldots, S_k)$ 增广，得到 S^* 的划分 $(S_1^*, \ldots, S_k^*)$。根据引理 2.11，对于压缩交换图 $G(S_1^*, \ldots, S_k^*)$ 中任意一条长度为 d_T的最短路径 $s, v_1, v_2 \ldots, v_{d_T-1}, t$，对所有 $i \in [d_T - 1]$，均有 $v_i \in D_i$。

现在我们分析过程 BlockFlowIndependence 中使用的独立性预言机查询次数 性。根据引理 3.2，第 1 行使用的独立性预言机查询次数为 $O(kn + p\log\ p)$。第 21 行使用的独立性预言机查询次数为 $O(k|D_{d_T-1}|)$。在第 16 行中，输出 $\emptyset$ 的 FindOutEdge 调用次数为 $O(k\sum_{i=1}^{d_T-2}|D_i|)$，而未输出 $\emptyset$ 的 FindOutEdge 调用次数为 $O(p)$。因此，根据引理 2.1，第 16 行使用的独立性预言机查询次数为 $O(kn + p\log\ p)$。

过程 BlockFlowIndependence 只是重复寻找一条从 s 到 T、经过集合 L_i 的最短路径，并增广可分割集合，同时从 L_i 中移除这些顶点，直到不存在这样的路径。因此，Update 中使用的所有路径都是最短增广路径。于是，根据引理 2.8，对于所有 $i \in [k]$，我们有 $S'_i \in \mathcal{I}_i$。此外，如果 $d_T \neq \infty$，则必定至少存在一条长度为 d_T 的增广路径。因此，我们有 $|S'| > |S|$。

接下来，我们证明 $d_{G(S'_1,\ldots,S'_k)}(s, T) \geq d_{G(S_1,\ldots,S_k)}(s, T) + 1$。为此，我们证明，从 L_i 中移除的任何元素都不可能再位于长度为 d_T 的任何最短增广路径上。当元素 $a_i \in L_i$ 没有通向 L_{i+1} 的出边时，从 $a_i \in L_i$ 到 T 的距离大于 $d_T - i$。于是，根据引理 2.11，这样的 a_i 不可能再位于长度为 d_T 的任何增广路径上。此外，根据引理 3.4，增广路径上的顶点 v 的距离在增广后必须严格增大。因此，引理 2.11 意味着，增广路径上的元素不可能再位于长度为 d_T 的任何增广路径上，由此完成证明。

□

现在，我们利用引理 2.10 证明定理 3.1。在我们的拟阵划分算法中，我们只需反复应用 BlockFlowIndependence，直到找不到任何 (s, T)-路径为止。

定理 *3.1* 的证明。在我们的算法中，我们从 $S = \emptyset$ 开始，并为所有 $i \in [k]$ 初始化 $S_i = \emptyset$。然后，我们反复应用 BlockFlowIndependence，对当前的 S 划分 $(S_1, \ldots, S_k)$ 进行增广，直到在压缩交换图 $G(S_1, \ldots, S_k)$ 中找不到任何 (s, T)-路径为止。

由于每次执行 BlockFlowIndependence 都会严格增加 $d_{G(S_1,\ldots,S_k)}(s, T)$，因此经过 $O(\sqrt{p})$ 次执行 BlockFlowIndependence 后，$d_{G(S_1,\ldots,S_k)}(s, T)$ 为 $\Omega(\sqrt{p})$。引理 2.10 表明，若 $d_{G(S_1,\ldots,S_k)}(s, T)$ 为 $\Omega(\sqrt{p})$，则 $|S| \geq p - O(\sqrt{p})$。那么，

在整个拟阵划分算法中，BlockFlowIndependence 的执行次数为 $O(\sqrt{p}) + O(\sqrt{p}) = O(\sqrt{p})$。引理 3.3 表明，一次 BlockFlowIndependence 执行需要 $O(kn + p\log\ p)$ 次独立性预言机查询，由此完成证明。

□

与秩预言机模型下 [CLS+19] 的拟阵交算法相同，我们很容易得到以下定理。

定理 3.5. 对于任意 $\epsilon > 0$，存在一种算法，使用 $O\left((kn + p\log\ p)\epsilon^{-1}\right)$ 次独立性预言机查询，并求得 k 个拟阵的最大可划分集合的 $(1-\epsilon)$ 近似解。

证明。 与定理 3.1 的证明类似，我们从 $S = \emptyset$ 开始，并为所有 $i \in [k]$ 初始化 $S_i = \emptyset$，然后反复应用 BlockFlowIndependence 来扩充 S 的当前划分 $(S_1, \ldots, S_k)$。唯一的区别是，我们只应用 BlockFlowIndependence ϵ^{-1} 次，这需要 $O\left((kn + p\log\ p)\epsilon^{-1}\right)$ 次独立性预言机查询。

每次执行 BlockFlowIndependence 都会严格增加 $d_{G(S_1,\ldots,S_k)}(s,T)$。因此，执行 BlockFlowIndependence ϵ^{-1} 次后，我们有 $d_{G(S_1,\ldots,S_k)}(s,T) = \Omega(\epsilon^{-1})$。引理 2.10 表明，如果 $d_{G(S_1,\ldots,S_k)}(s,T) = \Omega(\epsilon^{-1})$，那么 $|S| \geq p - O(p\epsilon)$，由此完成证明。 □

与我们在算法 3 中给出的实现不同，Cunningham [Cun86] 提出了过程 BlockFlowIndependence 的另一种实现，该实现不使用二分搜索技术。回想一下，给定 $S(\subseteq V)$ 的一个划分 $(S_1, \ldots, S_k)$，且满足对所有 $i \in [k]$ 都有 $S_i \in \mathcal{I}_i$，过程 BlockFlowIndependence 会输出 $S'(\subseteq V)$ 的一个划分 $(S'_1, \ldots, S'_k)$，使得对所有 $i \in [k]$、$|S'| > |S|$，都有 $S'_i \in \mathcal{I}_i$，并且 $d_{G(S'_1,\ldots,S'_k)}(s,T) \geq d_{G(S_1,\ldots,S_k)}(s,T) + 1$；若不存在这样的 S'，则输出 S 的一个划分 $(S_1, \ldots, S_k)$。

现在我们评估 Cunningham 的实现的查询复杂度。由于压缩后的交换图中有 $O(np)$ 条边不与 T 相邻，在一次 BlockFlowIndependence 执行中，这些边需要 $O(np)$ 次独立性预言机查询。算法执行期间，与 T 相邻的边会被高效维护。事实上，我们可以先用 $O(kn)$ 次独立性预言机查询计算出这些边，并在每次增广后通过 $O(n)$ 次查询更新它们。因此，我们得到以下结论。

注 3.6 (Cunningham 阻塞流算法 [Cun86，定理 5。1])。 有一种算法执行 d 次 `BlockFlowIndependence` 过程，并使用 $O(npd + kn)$次独立性预言机查询。

根据注 3.6，Cunningham 得到了一个拟阵划分算法，使用 $O(np^{3/2} + kn)$ 次独立性预言机查询；当 $k = \omega(p)$ 时，该算法优于定理 3.1 中给出的算法。回顾一下，在整个拟阵划分算法中，BlockFlowIndependence 的执行总次数为 $O(\sqrt{p})$。正如我们将在第 4 节中看到的，即使 k 很大，我们提出的算法仍严格优于 Cunningham 算法。

3.2 使用秩预言机的阻塞流算法

本小节中，我们将在秩预言机模型下，采用阻塞流方法介绍我们的拟阵划分算法。我们的目标是证明以下定理，该定理蕴含定理 1.3。

定理 3.7。 存在一种算法，使用 $O((n+k)\sqrt{p}\log\ n)$ 次秩预言机查询即可解决拟阵划分问题。

注意，在秩预言机模型下，我们可以通过调用一次二分搜索过程 FindInEdge，检查每个顶点 $u \in S$ 是否存在一条进入顶点u 的边。由此可得

在秩预言机模型下，解决拟阵划分问题所需的查询次数少于在独立性预言机模型下。

定理 3.7 中给出的算法的秩查询复杂度为 $\tilde{O}(n^{3/2})$，当 $k \leq n$ 时。如我们稍后将看到的，定理 4.1 中给出的拟阵划分算法的独立性查询复杂度为 $\tilde{O}(n^{7/3})$，这是目前独立性预言机模型下最佳的查询复杂度。

论证与定理 3.1 中给出的算法类似。然而，与独立性预言机模型中的算法不同，秩预言机模型中的算法通过反向遍历边，使用广度优先搜索和阻塞流方法执行增广过程。

我们首先给出 GetDistanceRank 过程（算法 4），它能高效地求出压缩交换图中每个顶点到 T 的距离。该算法只需使用 FindInEdge 过程执行广度优先搜索。

引理 3.8 （使用秩预言机的广度优先搜索）。给定 $S(\subseteq V)$ 的一个划分 $(S_1, \ldots, S_k)$，使得对所有 $i \in [k]$ 都有 $S_i \in \mathcal{I}_i$，过程 GetDistanceRank （算法 4）输出 $d \in \mathbb{R}^{V \cup \{s\} \cup T}$，且对于 $v \in V \cup \{s\} \cup T$，$d_v$ 是压缩交换图 $G(S_1, \ldots, S_k)$ 中从 v 到 T 的距离。过程 GetDistanceRank 使用 $O((n+k)\log n)$ 次秩查询。

算法 4： GetDistanceRank

输入： 一个划分 $(S_1, \ldots, S_k)$，满足 $S(\subseteq V)$，且 $S_i \in \mathcal{I}_i$ 对所有 $i \in [k]$

输出： $d \in \mathbb{R}^{V \cup \{s\} \cup T}$，使得对于 $v \in V \cup \{s\} \cup T$，$d_v$ 是从 v 到 T 的距离 $G\ (S_1, \ldots, S_k)$

对于 $v \in S$，记 $\pi(v)$ 为满足 $v \in S_{\pi(v)}$ 的指标
对于所有 $v \in V \cup \{s\}$，令 $d_v \leftarrow \infty$
$Q \leftarrow \emptyset$ // Q: 队列
$A \leftarrow V$
对 $i \in [k]$ **执行**
    $d_{t_i} \leftarrow 0$
    当 $u =$ FindInEdge$(M_i, S_i, \emptyset, A \setminus S_i)$ 满足 $u \neq \emptyset$ **时执行**
        $d_u \leftarrow 1$
        $Q \leftarrow Q + u$
        $A \leftarrow A - u$
当 $Q \neq \emptyset$ **执行**
    令 v 为最早添加到 Q 的元素
    $Q \leftarrow Q - v$
    如果 $v \notin S$ **则**
        if $d_s = \infty$ **然后**
            $d_s \leftarrow d_v + 1$
    else
        当 $u =$ FindInEdge$(\mathcal{M}_{\pi(v)}, S_{\pi(v)}, v, A \setminus S_{\pi(v)})$ 满足 $u \neq \emptyset$ **时**
            $d_u \leftarrow d_v + 1$
            $Q \leftarrow Q + u$
            $A \leftarrow A - u$
返回 d

证明。 GetDistanceRank 过程只是在压缩交换图 $G(S_1, \ldots, S_k)$ 中执行广度优先搜索。因此，GetDistanceRank 过程能够正确计算到 T 在 $G(S_1, \ldots, S_k)$ 中的距离。注意，每个顶点 $v \in V$ 至多被添加到 Q 一次，也至多从 A 中移除一次。因此，不输出 $\emptyset$ 的 FindInEdge 调用次数为 $O(n)$，而

输出 $\emptyset$ 的 FindInEdge 调用次数为 $O(n+k)$。因此，根据引理 2.2，第 7 行和第 18 行中使用的秩预言机查询次数为 $O((n+k)\log\ n)$，证明完毕。 □

接下来，类似于独立性预言机模型中的过程 BlockFlowIndependence，我们给出秩预言机模型下快速拟阵划分算法的增广子程序。其实现称为 BlockFlowRank（算法 5）。在过程 BlockFlowRank 中，我们用 I 表示满足汇点 $t_j \in T$ 未被移除的索引集合 j。

算法 5： BlockFlowRank

输入： $S(\subseteq V)$ 的一个划分 $(S_1, \ldots, S_k)$，使得对所有 $i \in [k]$，$S_i \in \mathcal{I}_i$

输出： $S'(\subseteq V)$ 的一个划分 $(S'_1, \ldots, S'_k)$，使得对所有 $i \in [k]$，$|S'| > |S|$，$S'_i \in \mathcal{I}_i$ 以及 $d_{G(S'_1,\ldots,S'_k)}(s,T) \geq d_{G(S_1,\ldots,S_k)}(s,T) + 1$，或者在不存在此类 $d_{G(S'_1,\ldots,S'_k}$ 时，对 S 的一个划分 $(S_1, \ldots, S_k)$ 此类 S' 存在。

```
1  d ← GetDistanceRank(S_1, ....., S_k)
2  若 d_s = ∞，则返回 (S_1, ..., S_k);
3  I ← [k]
4  对于所有 i ∈ [d_s − 1]，令 L_i ← {v ∈ V | d_v = i}
5  对于 v ∈ S，用 π(v) 表示满足 v ∈ S_{π(v)} 的索引
6  当 L_i ≠ ∅ 对所有 i ∈ [d_s − 1] 和 I ≠ ∅ 执行
7      l ← 0，a_0 ← ∅，a_{d_s−1} ← ∅
8      当 l < d_s − 1 执行
9          a_{l+1} ← ∅
10         如果 l = 0，则
11             当 I ≠ ∅ 时
12                 任意选取 j ∈ I
13                 a_1 ← FindInEdge(M_j, S_j, ∅, L_1 \ S_j)
14                 如果 a_1 ≠ ∅，则
15                     a_0 ← t_j
16                     跳出
17                 else
18                     I ← I − j
19             如果 a_1 = ∅ 则中断;
20         else
21             a_{l+1} ← FindInEdge(M_{π(a_l)} S_{π(a_l)} a_l, L_{l+1} \ S_{π(a_l)})
22         if a_{l+1} = ∅ 则
23             L_l ← L_l − a_l
24             如果 L_l = ∅ 则中断;
25             l ← l − 1
26         else
27             l ← l + 1
28     如果 a_{d_s−1} ≠ ∅ 则
29         将 P记作路径 s, a_{d_s−1}, ..., a_1, a_0
30         (S_1, ..., S_k) ← 更新((S_1, ..., S_k) P)
31         对于所有 i ∈ [d_s − 1] 令 L_i ← L_i − a_i
32 返回 (S_1, ..., S_k)
```

引理 3.9 (使用秩预言机的阻塞流**).** 给定一个划分 $(S_1, \ldots, S_k)$，其满足 $S_i \in \mathcal{I}_i$ 对所有 $i \in [k]$，过程 `BlockFlowRank` （算法 5）会输出一个划分 $(S'_1, \ldots, S'_k)$，其满足 $S'_i \in \mathcal{I}_i$ 对所有 $i \in [k]$，$|S'| > |S|$，以及 $d_{G(S'_1,\ldots,S'_k)}(s,T) \geq$

$d_{G(S_1,\ldots,S_k)}(s,T)+1$，或者在不存在这样的 S' 时，对 S 的划分 $(S_1,\ldots,S_k)$。过程 `BlockFlowRank` 使用 $O((n+k)\log n)$ 次秩预言机查询。

证明。我们将 D_i 记为顶点集合 $v \in V$，其中从 v 到汇点 T 的距离在压缩交换图 $G(S_1, \ldots, S_k)$ 中为 i；划分 $(S_1, \ldots, S_k)$ 是 S的输入划分。我们首先分析过程 BlockFlowRank 中使用的秩预言机查询次数。根据引理 3.8，第 1 行使用的秩预言机查询次数为 $O((n+k)\log n)$。在第 13 行，输出 $\emptyset$ 的 FindInEdge 调用次数为 $O(k)$，未输出 $\emptyset$ 的 FindInEdge 调用次数为 $O(|D_1|)$。此外，在第 21 行，输出 $\emptyset$ 的 FindInEdge 调用次数为 $O(p)$，未输出 $\emptyset$ 的 FindInEdge 调用次数为 $O(\sum_{i=2}^{d_s-1}|D_i|)$。因此，根据引理 2.2，第 13 行和第 21 行使用的秩预言机查询次数为 $O((n+k)\log n)$。

根据引理 3.9 的证明中相同的论证，我们有 $S'_i \in \mathcal{I}_i$，对所有 $i \in [k]$ 和 $|S'| > |S|$ 均成立。接下来，我们证明 $d_{G(S'_1,\ldots,S'_k)}(s,T) \geq d_{G(S_1,\ldots,S_k)}(s,T) + 1$。为此，我们证明，从 L_i 中移除的任何元素都不可能再出现在长度为 d_T 的任何最短增广路径上。当元素 $a_i \in L_i$ 没有来自 L_{i+1} 的入边时，从 s 到 a_i 的距离大于 $d_T - i$。于是，根据引理 2.11，这样的 a_i 不可能再出现在长度为 d_T 的任何增广路径上。将元素 j 从 I 中移除的顶点 $t_j \in T$ 也不可能再出现在长度为 d_T 的增广路径上。根据引理 3.4，增广之后，对于增广路径上的每个顶点 v，从 s 到 v 的距离都必须严格增加。因此，引理 2.11 意味着，增广路径上的元素不可能再出现在长度为 d_T 的增广路径上，从而完成证明。

□

现在我们给出定理 3.7 的证明。

定理 *3.7* 的证明。与定理 3.1 的证明相同，我们从 $S = \emptyset$ 开始，并为所有 $i \in [k]$ 初始化 $S_i = \emptyset$。然后，我们反复应用 BlockFlowRank 来扩充当前对 S 的划分 $(S_1, \ldots, S_k)$，直到在压缩交换图$G(S_1, \ldots, S_k)$ 中找不到 (s,T)-路径为止。

根据与定理 3.1 相同的论证，整个拟阵划分算法中 BlockFlowRank 的执行总次数为 $O(\sqrt{p})$。根据引理 3.9，证明完毕。 □

根据与定理 3.5 相同的论证，我们得到以下定理。

定理 3.10。对于任意 $\epsilon > 0$，存在一种算法，使用 $O((n+k)\epsilon^{-1}\log n)$ 次秩预言机查询，并找到 k 个拟阵的最大可划分集合的$(1-\epsilon)$ 近似解。

证明。与定理 3.5 的证明相同，我们从 $S = \emptyset$ 开始，并为所有 $i \in [k]$ 初始化 $S_i = \emptyset$，然后仅应用 BlockFlowRank ϵ^{-1} 次。根据引理 3.9，整个算法使用 $O((n+k)\epsilon^{-1}\log n)$ 次秩预言机查询。 □

4 适用于大 k 的更快算法

本节中，当 k 较大时，我们提出一种使用 $o(\min\{k,p\}n\sqrt{p})$ 次独立性预言机查询的算法。在小节 3.1 中，我们介绍了该算法（算法 2）：它在压缩交换图上运行 $\tilde{}$ 次广度优先搜索，并使用 $O(kn)$ 次独立性预言机查询。请回想，压缩交换图有 $O(np)$ 条边，因此我们可以使用 $O(np)$ 次独立性预言机查询来构造压缩交换图。由此可得一种算法，它使用 $O(\min\{kn, np\})$ 次独立性预言机查询，并在压缩交换图上运行广度优先搜索。在评估第 3 节中给出的采用阻塞流 方法的拟阵划分算法的独立性查询复杂度时，一个关键观察是，算法执行期间最短增广路径的不同长度数量为 $O(\sqrt{p})$。

目前尚不清楚我们能否获得一种拟阵划分算法，使其运行广度优先搜索 $o(\sqrt{p})$ 次。因此，阻塞流方法目前受困于 $\tilde{\Omega}(\min\{k,p\}n\sqrt{p})$ 次独立性预言机查询。

为突破这一障碍，并改进 Cunningham [Cun86] 和定理 3.1 给出的、使用 $O(\tilde{\min}\{k,p\}n\sqrt{p})$ 次独立性预言机查询的算法，我们提出了一种称为边回收增广的新方法，它能够以更少的独立性预言机查询总次数执行广度优先搜索。我们通过两项新思路实现这一方法：高效利用二分搜索过程 FindOutEdge，并借助引理 2.10 仔细分析独立性预言机查询的次数。结合阻塞流方法的算法与边回收增广方法的算法，我们得到以下定理，该定理蕴含定理 1.2。

定理 4.1。 存在一种算法，使用 $O(k'^{1/3}np\log p + kn)$ 次独立性预言机查询来解决拟阵划分问题，其中 $k' = \min\{k,p\}$。

该定理意味着，当 $k = \omega(p^{3/4})$ 时，我们可以得到一种使用 $o(\min\{k,p\}n\sqrt{p})$ 次独立性预言机查询的拟阵划分算法。需要注意的是，除独立性预言机查询外，该算法还需要 $O(k'^{2/3}np)$ 的时间复杂度。

在第 4.1 节中，我们介绍新的方法——边回收增广；在第 4.2 节中，我们介绍针对较大 k 的更快拟阵划分算法，并给出定理 4.1 的证明。

4.1 边回收增广

为了为算法选择合适的参数，我们必须确定p的值。然而，在运行算法之前，我们并不知道最大可划分集合的大小 p 。因此，我们不使用 p的精确值，而是使用其一个$\frac{1}{2}$-近似解 $\bar{p}$ （即 $\bar{p} \le p \le 2\bar{p}$），该近似解可通过 O次（kn）独立性预言机查询计算得到。众所周知，可以通过以下简单的贪心算法找到拟阵交问题的$\frac{1}{2}$-近似解；参见 [KV06, 命题 13.26]。我们从空集开始。对于基集中的每个元素，我们检查将其加入集合是否会得到一个公共独立集。如果是，就将该元素加入集合。最后，我们得到一个极大公共独立集。利用第 2.3 小节中给出的从拟阵划分到两个拟阵的交的归约 $\hat{\mathcal{M}}' = (\hat{V}, \hat{\mathcal{I}}')$ 和 $\hat{\mathcal{M}}'' = (\hat{V}, \hat{\mathcal{I}}'')$，我们将该算法转化为以下用于拟阵划分问题的$\frac{1}{2}$-近似算法（算法 6）。

算法 6： $\frac{1}{2}$-拟阵划分近似算法

对于所有 $i \in [k]$，令 $S_i \leftarrow \emptyset$
对于 $i \leftarrow 1$ **到** k，**执行**
  对于 $v \in V \setminus \left(\bigcup_{j=1}^{i-1} S_j\right)$，**执行**
    如果 $S_i + v \in \mathcal{I}_i$，**那么**
      $S_i \leftarrow S_i + v$
返回 $\bar{p} = |\bigcup_{i=1}^{k} S_i|$

现在我们介绍一种新的方法 边回收增广。我们的新方法边回收增广应用于算法的每个阶段。边回收增广的一个阶段如 EdgeRecyclingAugmentation（算法 8）所述。

在 EdgeRecyclingAugmentation 中，我们首先计算压缩交换图 $G(S_1,\dots,S_k)$ 中的 $E^*(\subseteq V \times S)$ 条边，这需要 $O(np)$ 次独立性预言机查询。请注意，压缩交换图可能会因增广而改变，也就是说，增广可能会在压缩交换图中添加或删除若干条边，因此，执行一条增广路径可能会

破坏边集 E^*。然而，我们注意到，增广之后可以回收边集 E^* 的一部分，这一点是拟阵划分所特有的。

在 EdgeRecyclingAugmentation 中，我们只需重复执行广度优先搜索，然后增广可分割集合。与第 3.1 节中的 GetDistanceIndependence（算法 2）不同，我们的 BFS 会回收预先计算的边集 E^*。在一个阶段中，我们维护一个集合 J，其中包含所有满足 S_i 被增广更新的索引 i。我们的关键观察是，除非增广路径包含顶点 $S_i \cup \{t_i\}$，否则增广不会改变压缩交换图中进入 S_i 内某个顶点的任何边。与对所有索引 $i \in [k]$ 使用二分搜索过程 FindOutEdge 的 GetDistanceIndependence 不同，我们新的 BFS 过程仅对索引 $i \in J$ 使用 FindOutEdge。我们可以使用 E^*，以 $i \notin J$ 搜索进入 S_i 中某个顶点的边。然后，基于上述思路的 BFS 可以实现为 EdgeRecyclingBFS（算法 7）。

我们还对整个拟阵划分算法中的独立性预言机查询次数进行了新的重要分析。在 EdgeRecyclingAugmentation 中，我们重复运行广度优先搜索 EdgeRecyclingBFS，使二分搜索过程的总调用次数为$O(np)$。因此，EdgeRecyclingBFS 在一次 EdgeRecyclingAugmentation 调用中使用的独立性预言机查询次数，几乎等于 E^* 的预计算所使用的查询次数。因此，˜一次 EdgeRecyclingAugmentation 调用使用 $O(np)$ 次独立性预言机查询。EdgeRecyclingAugmentation 中 EdgeRecyclingBFS 的调用次数取决于无法回收的边数。因此，为了确定 EdgeRecyclingBFS 的调用次数，我们在 EdgeRecyclingAugmentation 的实现中（算法 8）使用值 sum。在整个拟阵划分算法中，我们反复应用 EdgeRecyclingAugmentation。由此，我们可以得到一个使用 $\tilde{O}(np^{3/2}+kn)$ 次独立性预言机查询的拟阵划分算法。此外，通过将其与阻塞流方法结合，整个拟阵划分算法中 EdgeRecyclingAugmentation 的总调用次数可达到 $O(\min\{k^{1/3}, p^{1/3}\})$。由此可得到一个使用 $\tilde{O}(\min\{k^{1/3}, p^{1/3}\}np + kn)$ 次独立性预言机查询的拟阵划分算法。这一分析与现有的更快的拟阵交算法有显著不同。

对于 $i \in [k]$，令 $F_i (\subseteq V)$ 表示与 $t_i \in T$ 相邻的顶点集合。我们首先使用 $O(kn)$ 次独立性预言机查询，为所有 $i \in [k]$ 计算集合 F_i。注意，经过一次增广后，只需使用 $O(n)$ 次独立性预言机查询即可更新这些集合。

在接下来的两个引理中，我们证明过程 EdgeRecyclingAugmentation（算法 8）的正确性及其独立性查询复杂度。

引理 4.2。给定 $S(\subseteq V)$ 的一个划分$(S_1, \ldots, S_k)$，且对于所有 $i \in [k]$ 都有 $S_i \in \mathcal{I}_i$，过程 `EdgeRecyclingAugmentation`（算法 8）输出 $S'(\subseteq V)$的一个划分$(S'_1, \ldots, S'_k)$，使得对于所有 $i \in [k]$ 都有 $S'_i \in \mathcal{I}_i$，并且 $|S'| \geq |S|$。

证明。为了证明 EdgeRecyclingAugmentation 的正确性，我们证明在 while 循环每次迭代开始时，以下不变量成立。

(i) 对于所有 $i \in [k] \setminus J$ 以及所有 $(v, u) \in V \times S_i$，当且仅当 $S_i + v - u \in \mathcal{I}_i$ 和 $S_i + v \notin \mathcal{I}_i$ 时，我们有 $(v, u) \in E^*$。

(ii) 对于 所有 $i \in [k]$ 以及所有 $v \in V$，当且仅当 $S_i + v \in \mathcal{I}_i$ 和 $v \notin S_i$ 时，我们有 $v \in F_i$。

(iii) 对于所有 $i \in [k]$，我们有 $S_i \in \mathcal{I}_i$。

在执行 EdgeRecyclingAugmentation 之前，不变量成立。现在，假设在 while 循环某次迭代开始时，不变量 (i)–(iii) 均成立。令 $(S_1^{\text{old}}, \ldots, S_k^{\text{old}})$ 为执行第 13 行之前对 S^{old} 的划分，令 $(S_1^{\text{new}}, \ldots, S_k^{\text{new}})$ 为执行第 13 行之后对 S^{new} 的划分。对于所有 $i \in [k] \setminus J$，都有 $S_i^{\text{old}} = S_i^{\text{new}}$。因此，不变量 (i) 仍然成立。对于所有 $i \in [k] \setminus \{j\}$，都有 $|S_i^{\text{old}}| =$

$|S_i^{\text{new}}|$。众所周知，当 $|S_i^{\text{old}}| = |S_i^{\text{new}}|$ 时， S_i^{new} 与 S_i^{old} 的张成空间相同；参见 [Cun86, Lemma 4.4]。因此，不变量 (ii) 仍然成立。过程 EdgeRecyclingBFS 只需通过广度优先搜索，找到一棵以 s 为根的广度优先搜索树。因此，若不变量 (i)–(iii) 成立，则过程 EdgeRecyclingBFS 会正确计算出一棵以 s 为根的广度优先搜索树。于是，EdgeRecyclingBFS 在第 5 行输出的路径 P 是一条最短增广路径。因此，根据引理 2.9，不变量 (iii) 仍然成立。 □

引理 4.3. 过程 `EdgeRecyclingAugmentation` （算法 *8*）使用 $O(np\log p)$ 次独立性预言机查询。

证明。第 3 行使用的独立性预言机查询次数为 $O(np)$。此外，第 14 行使用的独立性预言机查询次数为 $O(np)$，因为 while 循环的迭代次数有上界 p。接下来只需证明，在整个 EdgeRecyclingAugmentation 过程中，FindOutEdge 的调用次数为 $O(np)$。

在过程 EdgeRecyclingBFS($(S_1, \dots, S_k)$ $E^*, J, \bigcup_{i=1}^{k} F_i$) 中，每个顶点 $v \in V$最多被加入 Q 一次，每个顶点 $v \in S$ 最多从 $B_{\pi(v)}$ 中移除一次，其中 $\pi(v)$ 是满足 $v \in S_{\pi(v)}$的索引。这意味着，不输出 $\emptyset$的 FindOutEdge 调用次数至多为 p，输出 $\emptyset$ 的 FindOutEdge 调用次数至多为 $n \cdot |J|$。因此，过程 EdgeRecyclingBFS 中 FindOutEdge 的调用次数为 $O(p + n \cdot |J|)$。

假设在过程 EdgeRecyclingAugmentatio 中，对 $J = J_1, J_2, \dots, J_c$ 调用了 EdgeRecyclingBFS n. 显然， $c \leq 2\bar{p} = O(p)$。此外，根据 EdgeRecyclingAugmentation 中 while 循环的条件，$\sum_{i=1}^{c} |J_i| = O(p)$。因此，整个 EdgeRecyclingAugmentation 过程中 FindOutEdge 的调用次数为

$$O\left(\sum_{i=1}^{c} (p + n \cdot |J_i|)\right),$$

即 $O(np)$。由引理 2.1 可知，整个 EdgeRecyclingAugmentation 过程中 FindOutEdge 所进行的独立性预言机查询次数为 $O(np\log\ p)$，证明完毕。

□

算法 7： EdgeRecyclingBFS

输入： 一个 $S(\subseteq V)$ 的划分 $(S_1, \dots, S_k)$，满足 $S_i \in \mathcal{I}_i$ 对所有 $i \in [k]$ 均成立；一个集合 $E \subseteq V \times S$，一个集合 $J \subseteq [k]$，一个集合 $F = \{v \in V \mid \exists i \in [k], v \notin S_i, S_i + v \in \mathcal{I}_i\}$。

输出： 若存在，则返回 $G(S_1, \dots, S_k)$ 中一条增广 (s, T)-路径。

```
1  Q ← {v ∈ V \ S} //  Q： 队列
2  B_i ← S_i 对于所有 i ∈ [k]
3  当 Q ≠ ∅ 时执行
4  |  令 v 为最早添加到 Q 的元素
5  |  Q ← Q − v
6  |  如果 v ∈ F 则
7  |  |  返回 广度优先搜索树中的最短增广路径
8  |  对于 i ∈ J 执行
9  |  |  当 u = FindOutEdge(M_i, S_i, v, B_i) 满足 u ≠ ∅ 时执行
10 |  |  |  Q ← Q + u
11 |  |  |  B_i ← B_i − u
12 |  对 i ∈ [k] \ J 执行
13 |  |  对于 u ∈ B_i 满足 (v, u) ∈ E 执行
14 |  |  |  Q ← Q + u
15 |  |  |  B_i ← B_i − u
16 返回 不存在路径
```

算法 8： EdgeRecyclingAugmentation

输入： 一个 $S(\subseteq V)$ 的划分 $(S_1, \ldots, S_k)$，满足对所有 $i \in [k]$，集合 $S_i \in \mathcal{I}_i$ $F_i = \{v \in V \setminus S_i \mid S_i + v \in \mathcal{I}_i\}$ 对所有 $i \in [k]$

输出： 一个 $S'(\subseteq V)$ 的划分 $(S'_1, \ldots, S'_k)$，满足对所有 $i \in [k]$ 和 $|S'| \geq |S|$ $S'_i \in \mathcal{I}_i$。

$sum \leftarrow 0$

$J \leftarrow \emptyset$

$E^* \leftarrow \{(v, u) \in V \times S \mid \exists i \in [k], u \in S_i, S_i + v \notin \mathcal{I}_i, S_i + v - u \in \mathcal{I}_i\}$

当 $sum < 2\bar{p}$ **时**

  $P \leftarrow$ EdgeRecyclingBFS($(S_1, \ldots, S_k)$ E^*, J, $\bigcup_{i=1}^{k} F_i$)

  如果 $P =$ 不存在路径**则**

    跳出

  对于 $v \in S$，用 $\pi(v)$ 表示满足 $v \in S_{\pi(v)}$ 的索引

  用 $V(P)$ 表示路径 P 中的 $= \{s, v_1, \ldots, v_{l-1}, t_j\}$ 个顶点

  对于 $i \leftarrow 2$ **到** $l-1$，**执行**

    $J \leftarrow J + \pi(v_i)$

  $J \leftarrow J + j$

  $(S_1, \ldots, S_k) \leftarrow$ 更新$((S_1, \ldots, S_k) P)$

  $F_j \leftarrow \{v \in V \mid v \notin S_j$ 和 $S_j + v \in \mathcal{I}_j\}$

  $sum \leftarrow sum + |J|$

返回 $(S_1, \ldots, S_k)$

此时，我们可以得到一个拟阵划分算法，其独立性预言机查询次数为 $O(np^{3/2} \log p + kn)$。在该算法中，我们首先计算所有 $F_i = \{v \in V \setminus S_i \mid S_i + v \in \mathcal{I}_i\}$ $i \in [k]$。接下来，我们反复应用 EdgeRecyclingAugmentation，对 S 的当前划分（S_1、……、S_k）进行增广，直到在压缩交换图 $G(S_1$、……、$S_k)$ 中找不到（s，T）-路径。正如我们将在引理 4.5 中证明的，该算法的独立性预言机查询次数为 $O(np^{3/2} \log p + kn)$。在下一小节中，我们将结合阻塞流方法算法和边回收增广方法算法对此进行改进。

4.2 通过结合阻塞流和边回收增广，加快大 k 情形的算法

我们已经介绍了两种在独立性预言机模型中求解拟阵划分问题的算法。我们将采用阻塞流方法的算法与采用边回收增广方法的算法结合起来。当压缩交换图中从 s 到 T 的距离较小时，我们采用阻塞流方法。另一方面，当压缩交换图中从 s 到 T 的距离较大时，我们采用边回收增广方法。其实现过程见算法 9。由此，我们得到一种拟阵划分算法，当 $k = \omega(p^{3/4})$ 时，所需的独立性预言机查询次数为 $o(\min\{k, p\}n\sqrt{p})$。这优于 Cunningham 算法和定理 3.1 中给出的算法，因为这两种算法都只采用阻塞流方法。

Alg算法 9：更快的拟阵划分算法　　g大规模情形下的算法 ge k

1通过运行12-拟阵划分近似算法（算法 6），为 p求得一个12-近似解 $\bar{p}$，并确定 d的值。2 对于所有 $i \in [k]$，令 $S_i \leftarrow \emptyset$3 重复应用 BlockFlowIndependence（算法 3），以增广当前的 S划分（S_1，……，S_k），直至压缩交换图 $G(S_1$，……，$S_k)$中从 s到 T的距离至少为 d。（如果注 3.6 中提到的实现优于算法 3，则采用该实现。）4 对于所有 $i \in [k]$，令 $F_i \leftarrow \{v \in V \setminus S_i \mid S_i + v \in \mathcal{I}_i\}$5 重复应用 EdgeRecyclingAugmentation（算法 8），以增广当前的 S划分（S_1，……，S_k），并用 $j \in [k]$更新 F_j，直至在压缩交换图 $G(S_1$，……，$S_k)$ 中找不到任何 $(s$，$T)$-路径。

该算法以整数 d为参数，我们将在最后确定其取值。为了分析算法 9 的˜独立性查询复杂度，我们首先证明第 3 行使用 $O(\min\{knd, pnd + kn\})$ 次独立性预言机查询，而第 5 行使用 $\tilde{O}\left(\frac{p^{3/2}n}{d^{1/2}}\right)$ 次独立性预言机查询。

引理 4.4. 算法 *9* 的第 *3* 行使用 $O(\min\{knd \log p, pnd + kn\})$ 次独立性预言机查询。

证明。引理 3.3 表明，执行 BlockFlowIndependence 后，压缩交换图中从 s 到 T 的距离至少增加 1。因此，调用 BlockFlowIndependence 的次数至多为 d。由于引理 3.3 表明，每次调用 BlockFlowIndependence 所需的独立性预言机查询次数为 $O(kn\log\ p)$，因此，算法 9 的第 3 行可以实现为一个使用 $O(knd\log\ p)$ 次独立性预言机查询的算法。此外，根据注 3.6，算法 9 的第 3 行也可以实现为一个使用 $O(pnd + kn)$ 次独立性预言机查询的算法，证明完毕。□

引理 4.5。 算法 *9* 的第 *5* 行使用 $O\left(\frac{p^{3/2}n}{d^{1/2}} \log\ p\right)$ 次独立性预言机查询。

证明。令 m 表示算法 9 第 5 行中调用 EdgeRecyclingAugmentation 的次数。根据引理 4.3，我们只需证明 $m = O\left(\sqrt{\frac{p}{d}}\right)$。对于 $i \in [m]$，令 c_i表示第 i次调用 EdgeRecyclingAugmentation 时找到的增广路径数量。对于 $i \in [m-1] \cup \{0\}$，m我们记作 $s_i = \sum_{j=i+1} c_j$。

我们先证明以下两个命题。

命题 4.6。 对于所有 $i \in [m-1]$，都有 $\sqrt{s_i} = O(c_i)$。

证明。令 $i \in [m-1]$。我们用 l_i 表示在第 i次调用 EdgeRecyclingAugmentation 时，最后一次 EdgeRecyclingBFS 找到的增广路径的长度。引理 2.10 意味着$s_i = O\left(\frac{p}{l_i}\right)$。注意，根据引理 2.11，随着可分割集合大小的增加，最短增广路径的长度不会减小。

在第 i次调用 EdgeRecyclingAugmentation 时，J 的大小之和至多为 $c_i^2 \cdot l_i$，因为 J的大小至多为 $c_i \cdot l_i$。此外，根据 EdgeRecyclingAugmentation 中 while 循环的条件，J 的大小之和至少为 $2\bar{p}(\geq p)$。因此，我们得到 $c_i^2 \cdot l_i \geq p$，进而有 $c_i^2 \geq \frac{p}{l_i}$。由于 $s_i = O\left(\frac{p}{l_i}\right)$，我们有 $\sqrt{s_i} = O(c_i)$，这就是所需的结论。□

By 命题 4.6，存在一个正常数 C，使得对于所有 $i \in [m-1]$，都有 $c_i \geq C\sqrt{s_i}$.

命题 4.7. 对于所有 $i \in [m-1]$，都有 $\frac{C}{\sqrt{1+C}} \leq \int_{s_i}^{s_{i-1}} \frac{dx}{\sqrt{x}}$。

证明. 设 $i \in [m-1]$。由于 $c_i \geq C\sqrt{s_i}$，我们得到

$$\begin{aligned}\int_{s_i}^{s_{i-1}} \frac{dx}{\sqrt{x}} = \int_{s_i}^{s_i+c_i} \frac{dx}{\sqrt{x}} \geq \int_{s_i}^{s_i+C\sqrt{s_i}} \frac{dx}{\sqrt{x}} \geq \int_{s_i}^{s_i+C\sqrt{s_i}} \frac{dx}{\sqrt{s_i+C\sqrt{s_i}}} \\ = \frac{C\sqrt{s_i}}{\sqrt{s_i+C\sqrt{s_i}}} = \frac{C}{\sqrt{1+C\frac{1}{\sqrt{s_i}}}} \geq \frac{C}{\sqrt{1+C}},\end{aligned}$$

由此完成证明。 □

By Claim 4.7, $m-1 = \sum_{i=1}^{m-1} 1 \leq \frac{\sqrt{1+C}}{C} \sum_{i=1}^{m-1} \int_{s_i}^{s_{i-1}} \frac{dx}{\sqrt{x}} = O\left(\int_{s_{m-1}}^{s_0} \frac{dx}{\sqrt{x}}\right) = O\left(\sqrt{s_0}\right).$

由于引理 2.10 表明 $s_0 = O\left(\binom{n}{d}\right)$，算法 9 第 5 行中调用 EdgeRecyclingAugmentation 的次数为 $O\left(\sqrt{\binom{p}{d}}\right)$。根据引理 4.3，证明完成。 □

在算法 9 中，我们设置参数 d，以平衡第 3 行和第 5 行所用独立性预言机查询的次数。因此，我们得到以下证明。

定理 *4.1* 的证明. 令 $k' = \min\{k, p\}$。根据引理 4.4，我们注意到，算法 9 第 3 行所用独立性预言机查询的次数上界为 $O(k'nd\log\ p + kn)$。这里，我们设置 $d = \frac{p}{k'^{2/3}}$ 并运行算法 9。于是，根据引理 4.4 和 4.5，第 3 行和第 5 行所用独立性预言机查询的次数为 $O(k'^{1/3}np\log\ p + kn)$。此外，第 1 行和第 4 行所用独立性预言机查询的次数为 $O(kn)$，由此完成证明。 □

注意，算法 9 除独立性预言机查询外，还需要 $O\left(np \cdot \binom{p}{d}\right) = O(k'^{2/3}np)$ 的时间复杂度。这是因为我们在 EdgeRecyclingBFS 中使用了大小为 np 的边集 E^*，而算法 9 中 EdgeRecyclingBFS 的总调用次数为 $O\left(\binom{p}{d}\right)$。

5 总结

仅仅结合 Cunningham 算法 [Cun86] 和 [Ngu19,CLS+19] 提出的二分搜索技术，并不能打破拟阵划分问题的 $O(n^{5/2})$ 独立性查询界。然而，我们引入了一种新方法 边回收增广，打破了这一界限，并得到了一个仅需 $\tilde{O}(n^{7/3})$ 次独立性预言机查询的算法。这一结果将是深入理解拟阵划分问题的重要进展。

我们的主要观察是，压缩交换图中的某些边在增广后会保持不变，因此我们无需再次查询即可找到它们。由此得到的拟阵划分算法，其独立性查询复杂度关于 k 是次线性的。这个想法相当简单，我们相信，新的边回收增广方法将有助于未来算法的设计。

近期，Blikstad-Mukhopadhyay-Nanongkai-Tu [BvdBMN21] 打破了拟阵交的 $\tilde{O}(n^2)$- 独立性查询界。因此，自然会有人问，我们能否对拟阵划分算法作出

类似的改进。然而，这种改进是不可能的。正如 ICALP 2023 的一位匿名审稿人所指出的，很容易证明，拟阵划分问题需要 $\Omega(kn)$ 次独立性预言机查询；当$k = \Theta(n)$时，这一查询数为 $\Omega(n^2)$。[4] 因此，这两个问题之间存在明显差异。

我们还研究秩预言机模型下的拟阵划分算法，并提出一种算法，在 $k \leq n$ 时使用 $\tilde{O}(n^{3/2})$ 次秩预言机查询。Blikstad 等人 [BvdBMN21]提出疑问：拟阵交问题在独立性预言机模型和秩预言机模型下的紧确界是否相同。拟阵划分问题中也自然会出现同类问题。与拟阵交问题不同，我们认为，就拟阵划分问题的查询复杂度而言，独立性预言机与秩预言机之间存在差异。

致谢

作者感谢 Yusuke Kobayashi 在手稿方面给予的大力支持和宝贵意见。作者还感谢 ICALP 2023 的三位匿名审稿人提出的宝贵意见。本研究部分获得京都大学与丰田汽车公司联合项目“面向移动社会的先进数学科学”的支持。

参考文献

[AD71] Martin Aigner 和 Thomas A Dowling。组合几何中的匹配理论。《美国数学会汇刊》，158(1):231–245，1971年。[Bli21] Joakim Blikstad。拟阵交的突破 $O(nr)$。载于第*48*届自动机、语言与编程国际研讨会论文集（*ICALP 2022*），第198卷，页码31:1–31:17，2021年。[BMNT 23] Joakim Blikstad、Sagnik Mukhopadhyay、Danupon Nanongkai 和 Ta-Wei Tu。通过动态预言机拟阵实现快速算法。载于第*55*届*ACM*计算理论年会论文集（*STOC 2023*），页码1229–1242，2023年。[BvdBMN21]Joakim Blikstad、Jan van den Brand、Sagnik Mukhopadhyay 和 Danupon Nanongkai。打破拟阵交的二次时间壁垒。载于第*53*届*ACM SIGACT*计算理论研讨会论文集（*STOC 2021*），页码421–432，2021年。[CLS+19] Deeparnab Chakrabarty、Yin Tat Lee、Aaron Sidford、Sahil Singla 和 Sam Chiu-wai Wong。更快的拟阵交算法。载于第*60*届计算机科学基础研讨会论文集（*FOCS 2019*），页码1146–1168。IEEE，2019年。[CQ16] Chandra Chekuri 和 Kent Quanrud。最大权拟阵交的快速近似算法。载于第*27*届*ACM-SIAM*离散算法研讨会论文集（*SODA 2016*），页码445–457。SIAM，2016年。

[4]设 M_1、.....、M_k 是定义在同一基集 V上的秩为 1 的拟阵，其中基集包含 n 个元素。现在，我们构造一个二分图 $G = (L \cup R,\ E)$，其具有 $|L| = n$、$|R| = k$，其中 $(v,\ i) \in E$ 当且仅当 $\{v\}$ 在 M_i中独立。这里，可分割集合的最大规模等于 G中匹配的最大规模。由于对大小至少为 2 的集合进行独立性查询没有意义，因此可以将其视为对该图 G 进行边查询访问。众所周知，求最大匹配的规模需要 $\Omega(kn)$ 次边查询；详情请参见 [Yao88] 。

[Cun86] William H Cunningham。拟阵划分与交算法的改进界。SIAM计算杂志，15(4):948–957，1986 年。[Edm68] Jack Edmonds。拟阵划分。决策科学数学，11:335，1968年。[Edm70] Jack Edmonds。次模函数、拟阵与某些多面体。收录于《组合结构及其应用》，R. Guy、H. Hanani、N. Sauer 和 J. Schonheim 编。纽约，页码69–87，1970年。[Edm79] Jack Edmonds。拟阵交。载于《离散数学年刊》第4卷，页码39–49。爱思唯尔，1979年。[Edm03] Jack Edmonds。次模函数、拟阵与某些多面体。载于《组合优化——尤里卡，你成功了！：献给Jack Edmonds的论文》，Jack Edmonds第五届国际研讨会，奥苏瓦，法国，2001年3月5日至9日，修订论文，页码11–26。施普林格，2003年。[FM12] Andr´as Frank 和 Zolt´an Mikl´os。拟阵与次模流的简易推送-重标记算法。《日本工业与应用数学杂志》，29:419–439，2012年。[GM75] Curtis Greene 和 Thomas L Magnanti。一些抽象枢轴算法。SIAM应用数学杂志，29(3):530–539，1975年。[GW88] Harold Gabow 和 Herbert Westermann。森林、框架与博弈：拟阵和的算法及其应用。载于第20届ACM计算理论年度研讨会论文集（STOC 1988），页码407–421，1988年。[Har08] Nicholas JA Harvey。拟阵交、指针追逐与 S_n的Young半正规表示。载于第19届ACM-SIAM离散算法年度研讨会论文集（SODA 2008），页码542–549，2008年。[Has08] Julian Haselmayr。《拟阵交的理论与算法》。硕士论文，奥格斯堡大学，2008年。[HK73] John E Hopcroft 和 Richard M Karp。二分图最大匹配算法。SIAM计算杂志，2(4):225–231，1973年。[HKK19] Chien-Chung Huang、Naonori Kakimura 和 Naoyuki Kamiyama。加权拟阵交的精确算法与近似算法。《数学规划》，177(1-2):85–112，2019年。曾在SODA 2016上公布。[KKMS21] Yasushi Kawase、Kei Kimura、Kazuhisa Makino 和 Hanna Sumita。最优拟阵划分问题。Algorithmica，83:1653–1676，2021年。曾在ISAAC 2017上公布。[Knu73] Donald Ervin Knuth。拟阵划分。斯坦福大学计算机科学系，1973年。[KV06] Bernhard H Korte 和 Jens Vygen。组合优化。施普林格，第三版，2006年。[Law75] Eugene L Lawler。拟阵交算法。《数学规划》，9(1):31–56，1975年。

[LSW15] Yin Tat Lee、Aaron Sidford 和 Sam Chiu-wai Wong。更快的割平面法及其对组合优化和凸优化的影响。载于第*56*届计算机科学基础年会（*FOCS 2015*）论文集，第1049–1065页。IEEE，2015年。[Ngu19] Huy L Nguyen。关于拟阵交 Cunningham 算法的一则注记。*CoRR*，abs/1904.04129，2019年。[NW66] Crispin St John Alvah Nash-Williams。拟阵在图论中的一项应用。载于图论——国际研讨会——´图论´国际研´讨会，罗马，第263–265页，1966年。[Pri15] Christopher Price。次模函数最小化及相关问题的组合算法。硕士论文，滑铁卢大学，2015年。[Qua23] Kent Quanrud。通过推送-重标记实现拟阵打包与覆盖的更快精确算法和近似算法。*CoRR*，abs/2303.01478，2023年。即将发表于 SODA 2024。[Sch03] Alexander Schrijver。组合优化：多面体与效率，第24卷。施普林格，2003年。[Tu22] Ta-Wei Tu。在秩预言机下的次二次加权拟阵交。载于第*33*届国际算法与计算研讨会（*ISAAC 2022*）论文集，第248卷，第63:1–63:14页，2022年。[Yao88] Andrew Chi-Chih Yao。单调二分图性质具有规避性。*SIAM*计算杂志，17(3):517–520，1988年。