arXiv:2305.02166v1 [cs.DS] 3 May 2023

近乎线性时间内的最小链覆盖*

Manuel C´aceres†

摘要

宽度为 k 的有向无环图（DAG） $G=(V,E)$ 的最小链覆盖（MCC）是由 k 条链（传递闭包中的路径）构成的集合G，其中每个顶点至少出现在覆盖中的一条链中。

目前最先进的最小链覆盖算法的运行时间为 $O(T_{MF}(|E|)+k|\tilde{V}|)$，$O(k^2|V|+|E|)$ [C´aceres 等，SODA 2022]，$\tilde{O}(|V|^{3/2}+|E|)$ [Kogan 和 Parter，ICALP 2022] 和$\tilde{O}(T_{MCF}(|E|)+\sqrt{k}|V|)$ [Kogan 和 Parter，SODA 2023]，其中 $T_{MF}(|E|)$ 和 $T_{MCF}(|E|)$ 分别是求解最大流（MF）和最小费用流（MCF）的运行时间。

在这项工作中，我们提出了一种运行时间为 $O(T_{MF}(|E|)+(|V|+|E|)\log k)$ 的算法。结合近期求解最大流（MF）的结果 [Li 等人，FOCS 2022]，我们的算法是首个运行时间近乎线性的算法。此外，我们的技术具有确定性；如果最大流问题有确定性的近线性时间算法，那么我们也能据此为最小链覆盖（MCC）推导出确定性的近线性时间算法。

我们方案的核心是采用一种经过修改的可合并字典版本¨Farach 和 Thorup，Algorithmica]，[Iacono 和 Ozkan，ICALP 2010] 数据结构，并通过 SIZE-SPLIT 操作加以增强，且能以摊销对数时间回答查询；这一数据结构本身也可能具有独立的研究价值。

1 引言

计算有向无环图 $G=(V,E)$ 的所有顶点的最小链集合是一个众所周知的可在多项式时间内求解的问题 [15, 18]，，在生物信息学 [34, 7, 13, 4, 8, 32] 等广泛研究领域中有许多应用。这里，我们将这种对象称为一个最小链覆盖（MCC），即包含 k条链 $\mathcal{C}=\{C_1,\ldots,C_k\}$ 的集合，这些链是 G 的传递闭包中的路径。MCC 的大小k 称为 G 的宽度，根据偏序集（posets）上的迪尔沃斯定理 [15]，它等于 G 中两两不可达顶点的最大数量（即反链）。

*本工作由芬兰科学院资助（资助编号：352821、328877）。我非常感谢 Alexandru I. Tomescu 和 Brendan Mumey 就表示 MPC 的最小流的流分解进行的初步讨论。

†芬兰赫尔辛基大学计算机科学系，manuel.caceresreyes@helsinki.fi。

最小链覆盖的历史。Fulkerson [18] 在20世纪50年代率先为偏序集（传递有向无环图）提出了多项式时间算法。他的算法将问题归约为在一个具有 $2|V|$ 个顶点和 $|E|$ 条边的二分图中寻找最大匹配，因此使用 Hopcroft-Karp 算法 [21][1] 可在 $O(|E|\sqrt{|V|})$ 时间内求解。基于这些思路，Chen 和 Chen [10, 11], 提出了运行时间为 $O(|V|^2 + k\sqrt{k}|V|)$ 和 $O(\sqrt{|V||E| + k\sqrt{k}|V|})$ 的算法, Felsner 等人 [17]则提出了运行时间为 $O(k|V|^2)$ 的算法。在同一篇文章中, Felsner 等人还展示了一种组合方法，可在近线性时间内计算情形 $k = 2$、3、4 下的最大反链（MA），后来该方法被推广至 $O(f(k)(|V|+|E|))$ [5]，其中 $f(k)$ 是一个指数函数。最先进的方法大幅改善了其运行时间对 k 的依赖。这些方法求解的是与之密切相关的最小路径覆盖（MPC）问题。MPC P 是覆盖 G 的顶点的最小规模路径集合，因此也是有效的链覆盖。此外，由于每个 MCC 都可以转换为 MPC（通过连接各链中的相邻顶点），MPC 的大小也等于宽度 k。

MCC 的最新成果。M¨akinen 等人 [29] 为 MPC 提供了一个算法 $\widetilde{}$，其运行时间为 $O(k(|V|+|E|)\log |V|) = O(k(|V|+|E|))$；而 C´aceres 等人 [6] 提出了首个 $O(k^2|V|+|E|)$ 参数化线性时间算法。这两种算法都基于到最小流 [30], 的经典归约，我们稍后将重新讨论该归约。在同一项工作中，C´aceres 等人 [6] 展示了如何在时间 $O(T_{MF}(|E|) + ||\mathbb{P}||)$ 内计算 MPC，以及如何在时间 $O(T_{MF}(|E|))$ 内计算最大反链，其中$T_{MF}(|E|)$是求解最大流（MF）的运行时间，而 $||\mathbb{P}||$ 是所报告 MPC 的总长度。因此，利用 Chen 等人 [9]关于 MF 的最新结果，我们可以在（几乎）最优的（输入+输出大小）时间内求解 MPC 和最大反链。然而，同样的方法并不适用于 MCC，因为 MCC 的总长度可以恰好为 $|V|$（例如，通过移除重复顶点），而 MPC 的总长度在最坏情况下可以为 $\Omega(k|V|)$，如图 1 所示。

Kogan 和 Parter 最近通过减少 MPC 的总长度，突破了 $k|V|$ 障碍。他们通过使用可达性捷径并设计更复杂的最小费用流（MCF）规约，实现了这一改进。使用 Bran 等人 [35] 和 Chen 等人 [9],的 MCF 算法时，他们的算法运行时间分别为 $\widetilde{O}(|E| + |V|^{3/2})$ [26] 和 $\widetilde{O}(\sqrt{k}|V|+|E|^{1+o(1)})$ [27] 。

本文提出了一种最小链覆盖算法，相较于当前最先进的方法，其运行时间对 k 的依赖呈指数级改善。

定理 1. 给定一个 k-宽度有向无环图 $G = (V, E)$，我们可以在 $O(T_{MF}(|E|) + (|V|+|E|)\log k)$时间内计算出最小链覆盖，其中 $T_{MF}(|E|)$ 是求解最大流所需的时间。

因此，通过应用 Chen 等人的流算法， [9] 我们解决了这个问题

[1]近期的快速最大匹配算法并不能加速这种方法，因为必须先计算该有向无环图的传递闭包。

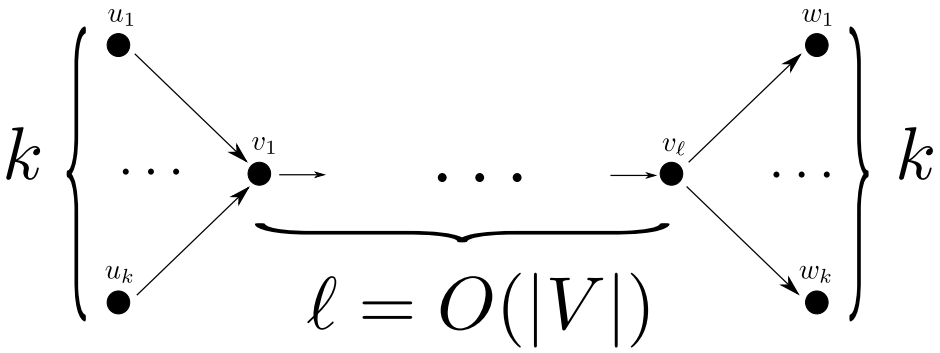


图 1：一个宽度为 k 的有向无环图示例，其中每个 MPC P 的总长度 ||P|| =Ω($k|V|$)。实际上，MPC 中的每条路径都必须从某个 u_i 出发，并沿着中间路径 $v_1, \ldots, v_\ell$ 前进，直到到达某个 w_j；否则就无法以最小规模覆盖图中的其余部分。此外，如果 $k = \ell = |V|/3$，那么 ||P|| = Ω($|V|^2$)。另一方面，始终存在一个总大小为 ||C || = $|V|$ 的 MCC（在本例中，只有一条链覆盖中间路径上的顶点）。

并首次实现了近线性时间。

推论 1. 给定一个宽度为 k 的有向无环图 $G = (V, E)$，我们可以以 $O(|E|^{1+o(1)})$ 的时间复杂度计算出一个最小链覆盖，且该算法以高概率成功。

此外，定理 1 中的解法将最大流求解器用作黑盒，除此之外则是确定性的。因此，如果能找到最大流的近线性时间求解器，我们便可提供一个确定性的最小链覆盖解法。我们解法的核心是采用经 SIZE-SPLIT 操作增强的可合并字典，以高效地将最大流求解器输出的流转换为最小链覆盖。可合并字典是一种数据结构，用于维护自然数 {1, . . . , k} 的动态划分 K（有意重复使用 k，因为我们的方法正是这样使用该数据结构的），初始划分为 K = {{1, . . . , k}}，并支持以下操作，对于 $K, K_1, K_2 \in$ K:

– 搜索(K, j)：若存在这样的元素，则返回最大值$_{i \in K, i \le j}\, i$ [2]。 – 合并(K_1, K_2)：在 K 中用 $K_1 \cup K_2$ 替换 K_1 和 K_2。 – 拆分(K, j)：在 K 中用 $K' = \{i \in K \mid i \le j\}$ 和 $K \setminus K'$ 替换 K。

注意，合并操作并不假设拆分操作生成的 max$_{i \in K_1}\, i <$min$_{i \in K_2}\, i$（集合互不重叠）。如果假设前一个条件（或类似的 max$_{i \in K_2}\, i <$min$_{i \in K_1}\, i$），该操作

[2]在相关文献中，这种查询也称为 *predecessor* 查询。

称为连接。可合并字典可应用于 Lempel-Ziv 解压缩 [16, 3, 31], 、可合并树 [19] 以及并查集拆分的推广 [28]。本文将说明如何对其进行修改，以获得快速的 MCC 算法。接下来，我们回顾文献中用于实现可合并字典的不同方法。

可合并字典。首个高效的可合并字典实现可由 Farach 和 Thorup 提出的分段合并策略推导得出。该策略旨在高效合并两棵自平衡二叉搜索树，并假设搜索、拆分和连接操作均可在对数时间内实现。作者展示了如何通过对两个集合进行最少次数的拆分，将其划分为互不重叠的子集，再将所得部分两两连接，从而实现合并操作。他们证明，经过 $|V|+|E|$ 次操作（这里再次有意滥用了符号约定）后，该策略的每次操作摊还时间为 $O(\log k \cdot \log(|V|+|E|))$[3]。后来，Iacono 和 Ozkan 提出了一种基于偏置跳表的可合并字典实现，每次操作的摊还时间为 $O(\log k)$。此后，Karczmarz 通过一种非常简单的方法也达到了相同的摊还运行时间：使用字典树表示集合。

我们的 MCC 算法首先计算一个编码 MPC 的最小流 f^*，然后通过按拓扑序处理 G 并查询经 SIZE-SPLIT 操作增强的可合并字典，从 f^* 中提取 MCC。形式上，我们要求支持以下操作：对于 $K, K_1, K_2 \in \mathrm{K}$ $s \in \{1,\ldots,|K|-1\}$：

– SOME(K)：返回某个元素 $i \in K$。– MERGE(K_1, K_2)：在 K 中用 $K_1 \cup K_2$ 替换 K_1 和 K_2。– SIZE-SPLIT(K, s)：在 K 中用 $K' \subseteq K$, $|K'| = s$ 和 $K \setminus K'$ 替换 K。

在第 3 节中，我们将展示如何以 $O(\log k)$ 的摊还时间实现这些操作。接着，在第 4 节中，我们将展示如何使用前文介绍的数据结构来实现我们的 MCC 算法。除了前面提到的理论改进之外，我们还将突出介绍解决方案的简洁性，包括增强型可合并字典的方案，以及 MCC 算法。

2 符号约定和预备知识

图。对于顶点 $v \in V$，我们用 $N^-(v)$（$N^+(v)$）表示 v 的入（出）邻居集合，即 $N^-(v) = \{u \in V \mid (u,v) \in E\}$（$N^+(v) = \{w \in V \mid (v,w) \in E\}$）。$v_1 v_\ell$-路径是一个顶点序列 $P = v_1, \ldots, v_\ell$，满足对于 $i \in \{1,\ldots,\ell-1\}$，有 $(v_i, v_{i+1}) \in E$；在这种情况下，我们称 v_1 可达 v_ℓ。如果 $\ell \geq 2$，则称 P 是简单的；如果 $v_1 = v_\ell$，则称 P 是一个环。有向无环图（DAG）是不含简单环的图。在有向无环图中，我们可以在线性时间内计算 [23, 33],

[3] [16] 的作者没有对可合并字典作出形式化定义。

一个拓扑序 $v_1, \ldots, v_{|V|}$，其顶点满足：对于每个 $i < j$，$(v_j, v_i) \notin E$。本文假设 G 是一个有向无环图，并且由于我们的算法运行时间为 $\Omega(|V| + |E|)$，因此假设输入中给定了一个拓扑序。链是顶点序列 $C = v_1, \ldots, v_{\ell'}$，满足对于每个 $i \in \{1, \ldots, \ell' - 1\}$，$v_i$ 可达 v_{i+1}。我们用 $|C| = \ell'$ 表示链的长度。链覆盖 C 是由若干链构成的集合，其中每个顶点都出现在 C 中的某条链上。如果每个顶点恰好出现在 C 中的一条链上，我们称其为链分解；如果 C 中的每条链都是一条路径，则称其为路径覆盖，此时我们改用 P 表示它。我们用 $\|\mathsf{C}\|$ 表示 $\|\mathsf{C}\| = \sum_{C \in \mathcal{C}} |C|$ 这一链覆盖的总长度。反链是顶点的子集 $A \subseteq V$，满足对于 $u, v \in A, u \neq v$，u 不可达 v。链覆盖的最小规模称为 G 的宽度，记为 k。

流。给定需求函数 $d : E \to \mathbb{N}_0$ 和 $s, t \in V$，st-流是满足流守恒的函数 $f : E \to \mathbb{N}_0$，即对于每个 $v \in V \setminus \{s, t\}$，有 $inFlow_v := \sum_{u \in N^-(v)} f(u, v) = \sum_{w \in N^+(v)} f(v, w) := outFlow_v$，并满足需求，即对于每个 $e \in E$，有 $f(e) \geq d(e)$。 k（再次有意重复使用该记号）的流分解是由 f 的 st-路径组成的集合 $\mathsf{D} = P_1, \ldots\ldots, P_k$，使得对于每条边 $e \in E$，$f(e) = |\{P_i \in \mathsf{D} \mid e \in P_i\}|$。[4] f 的大小 $|f|$ 定义为进入 t 的净流量（根据流守恒，等价于离开 s 的净流量），即 $|f| = inFlow_t - outFlow_t$。最小流问题要求寻找大小最小的可行 st-流。寻找 MPC 可以归约为分解一个特定的最小流 [30], ，我们将在第 4 节重新讨论这一归约。用于最大流问题的相同技术也可应用于最小流问题的情形 [12, 2]。事实上，对于 MPC，可以直接应用容量至多为 $|V|$ [6, 的最大流算法]定理 2.2（完整版本）。

数据结构。 像 AVL 树或红黑树这样的自平衡二叉搜索树 [20] 是一种二叉搜索树，支持 SEARCH、拆分和连接操作，且每种操作的时间复杂度（最坏情况下）均为对数级。一个（二叉）字典树 [14] 是一棵二叉树，它通过将整数的二进制表示 存储为字典树中的根到叶 路径来表示一个整数集合。在我们的字典树中，所有根到叶路径的长度均为 $\lfloor \log k \rfloor + 1$。对于支持一组操作的数据结构，以及用于刻画该数据结构状态的势函数 ϕ，我们称某个操作的摊还 时间等于其（最坏情况）运行时间加上该操作引起的势能变化 $\Delta\phi$。如果我们执行一系列 $O(|V| + |E|)$ 个摊还时间为 $O(\log\ k)$ 的操作，那么总（最坏情况）运行时间为 $O((|V| + |E|) \log\ k)$。

3 使用 SIZE-SPLIT 的可合并字典

我们展示了如何实现增强型可合并字典，使其支持 SOME、合并和 SIZE-SPLIT 操作，且每项操作的摊还时间为 $O(\log k)$。为

[4]这是一个简化的流分解定义，足以满足我们的需要。

了实现这一结果，我们在现有的可合并字典解决方案上进行修改：该方案实现了搜索、合并和拆分操作，我们通过添加选择操作对其进行扩展。形式化地，对于 $K \in \mathrm{K}\ s \in 1,\ldots, |K|$，

– 选择(K，s)：返回 K 中第 s 小的元素。

借助选择操作，我们可以使用（普通的）可合并字典来实现 SOME 和 SIZE-SPLIT，具体如下：

– $\mathrm{SOME}(K) \leftarrow \mathrm{SEARCH}(K, k)$.

– $\text{SIZE-SPLIT}(K, s) \leftarrow \mathrm{SPLIT}(K, \mathrm{SELECT}(K, s))$.

对于 SOME，只需在已知上界的情况下执行一次搜索即可（回想一下，k 是所考虑全集中的最大元素）；而对于 SIZE-SPLIT，我们可以先选择相应的枢轴，然后利用该枢轴按其值拆分集合，从而得到所需的拆分规模。

通过这一归约，只需以对数摊还代价实现选择操作，即可获得增强型可合并字典（SIZE-SPLIT 可视为一次选择调用，随后进行一次独立的拆分调用）。此外，如果选择操作的实现不会修改数据结构（因而也不会修改势能 ϕ），那么其摊还时间就等于其（最坏情况下的）运行时间（因为 ϕ 不会发生变化）。我们证明，在 Farach 和 Thorup 的分段合并策略 [16]，以及 Karczmarz 的字典树实现 [24]，中，情况确实如此；后者达到了所需的运行时间。

基于分段合并的可合并字典将每个集合表示为自平衡二叉搜索树。因此，通过在树的每个节点存储子树大小，可以在 $O(\log k)$ 时间内实现选择操作；这些子树大小可以在与树的常规操作相同的（最坏情况）运行时间内维护（即在树发生变化时更新）（参见例如 [25]）。

类似地，Karczmarz [24] 的实现将每个集合表示为其元素构成的字典树。如第 2 节所述，字典树将其元素存储为二进制表示，并将其编码为字典树中长度相等的根到叶路径。例如，如果 $k = 6$ 3 的对应二进制表示为 011，那么它就表示为一条根到叶路径：先沿着左子节点，再沿着右子节点，最后沿着右子节点。与二叉搜索树类似，可以扩展字典树中的节点，使其存储各自子树中的叶节点数量。如果对字典树进行这种扩展，那么操作 SELECT 可以在 $O(\log k)$ 的（最坏情况）时间内实现，与其在二叉搜索树中的实现类似，因为字典树中的叶节点遵循与其所表示的元素相同的顺序：对于字典树节点上的查询 SELECT(s)，只需查看左子节点下方的叶节点（元素）数量 l；如果 $l \geq s$，则继续访问左子节点并回答 SELECT(s)，否则继续访问右子节点并回答 SELECT($s - l$)。

Karczmarz [24] 指出，可以通过找到与 P 和 j之间最长公共前缀相对应的根到节点路径，来执行 SPLIT(K，j)

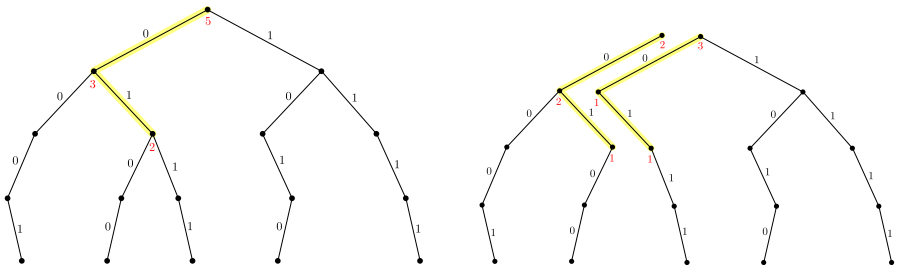


图 2：在增强型可合并字典的字典树表示上调用 SIZE-SPLIT(K = {1,4,7,10,15},2) 的结果。集合中各数字的二进制表示被表示为字典树的根到叶路径。部分叶节点的计数器以红色标注在各自对应的节点下方。这些计数器用于回答 $4 \leftarrow$ SELECT(K,2)。随后执行 SPLIT(K,4)。表示 4 (0100) 和 7 (0111) 的二进制表示之间最长前缀 (01) 的路径 P 以黄色突出显示。右图展示了最终结果。请注意，只有 P 中节点的计数器发生了变化。

j和 K的二进制表示中， K里大于j 的最小值。随后，通过移除 P中各节点的右子节点来拆分字典树（从而形成 K'），再将这些节点作为 P 副本的右子节点连接起来（从而创建 $K \setminus K'$）。注意，可以轻松扩展此过程，以维护每个节点下的叶节点数量：只有 P 中的节点（及其副本）会将其值减去所失去子节点的叶节点数量。图 2 展示了 SPLIT 操作的一个示例。MERGE(K_1， K_2)通过一个简单的递归算法实现，该算法在每一步保留一个（共有的）节点，然后合并相应的左右子树（如果其中一棵子树为空，则只保留另一棵子树；详情请参阅原始文献 [24]）。在这种情况下，可以在递归调用返回时维护叶节点数量：只需将两个子节点的叶节点数量相加，重新计算叶节点总数即可。每次计算都是常数时间操作，因此不会改变 MERGE 的渐近运行时间。由此得到以下引理。

引理 1. 存在一种数据结构，用于维护 $\mathcal{K}$ 对 $\{1, \ldots, k\}$ 的动态划分$\mathcal{K} = \{\{1, \ldots, k\}\}$ ，并支持 *SOME* 、合并和 *SIZE-SPLIT* 操作；对于长度为 $n = \Omega(k)$ 的操作序列[5] ，总共可在 $O(n \log k)$ 时间内完成。

证明。 我们首先注意到，SOME、合并和 SIZE-SPLIT 操作可以分别以与 Karczmarz [24], 的数据结构中的搜索、合并和拆分操作相同的渐近（最坏情况）运行时间实现。事实上，正如前文所述，SOME 可通过一次调用

[5]这一要求源于可合并字典实际上从一个空集合族开始；不过，可以创建单元素集合，并在总计 $O(k \log k)$ 时间内将它们合并。

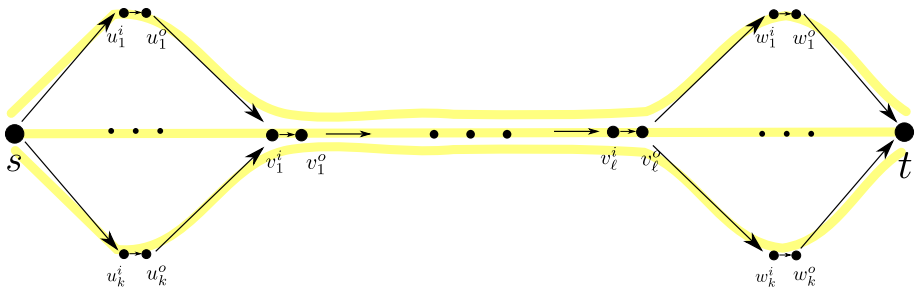


图 3：图 1 中图的流归约（未显示需求）及其上的一个最小流。图中仅显示流量为正的边。流以流分解的形式隐式呈现，其中每条路径都以黄色标出；该流分解对应于原始有向无环图的一个 MPC。

搜索实现；合并的实现方式与 [24] 中相同，只需处理叶子数量计数器的更新，这不会影响渐近运行时间；SIZE-SPLIT 则通过一次调用选择（耗时 $O(\log k)$），再调用一次拆分（同样耗时 $O(\log k)$）实现。对于摊还分析，我们沿用 Karczmarz [24],使用的势函数，即所有字典树中的节点总数。SOME 和合并操作的势能变化与 [24] 中相同，因此摊还运行时间为 $O(\log k)$。最后，由于选择不会改变节点总数（也不会改变任何字典树），SIZE-SPLIT 的势能变化与拆分相同，即与 [24]中一样为 $O(\log k)$。利用该数据结构各项操作的摊还运行时间即可得到此引理。

□

4 用于 MCC 的近线性时间算法

我们展示如何利用引理 1 得到一个快速的 MCC 算法。我们的算法计算一个最小流 f^*，该流编码了 G 一个 MPC；随后利用引理 1 中的数据结构，从 f^* 中高效提取一个 MCC。接下来，我们按照第 2.3 节（完整版）]的符号约定，描述从 MPC 到 MF 的经典 [30] 归约 [6, 。

给定有向无环图 G，我们构造其流归约图 G = (V, E)：通过添加一个全局源点 s、一个全局汇点 t，并将每个顶点$v \in V$ 拆分为两个由一条边连接的副本，从而得到该图。此外，第一个副本 v^i 与 v 的入邻居相连，第二个副本 v^o 与 v 的出邻居相连。形式上，$\mathrm{V} = \{s,t\} \cup \{v^i \mid v \in V\} \cup \{v^o \mid v \in V\}$，$\mathrm{E} = \{(s,v^i) \mid v \in V\} \cup \{(v^o,t) \mid v \in V\} \cup \{(v^i,v^o) \mid v \in V\} \cup \{(u^o,v^i) \mid (u,v) \in E\}$。注意，$|\mathrm{E}| = O(|V| + |E|)$，并且 G 也是一个有向无环图。我们还为边 $d : \mathrm{E} :\to \mathbb{N}_0$ 定义需求：若边的形式为 (v^i, v^o)，则需求为 1；否则为 0。

算法 1： 我们的 MCC 算法的未优化伪代码。该算法的朴素实现可获得 $O(\|\mathbb{P}\|)$ 的运行时间如本文所述。

输入： 一个有向无环图 $G = (V, E)$。

输出： G的最小链分解 $\mathcal{C} = C_1, \ldots, C_k$。

```
(G, d) ← 构造第 4 节中详细介绍的 G 流归约
f*, k = |f*| ← 利用引理 2 求得 (G, d) 的最小流
将 C_i 初始化为空列表，用于 i ∈ {1, ..., k}
I_s ← {1, ..., k}
对于 v ∈ V 按拓扑序 执行
    I_v ← 从 I_s 中取出 f*(s, v^i) 个元素。
    对于 u^o ∈ N^-(v^i)，执行
        I_uv ← 从 I_u 中取出 f*(u^o, v^i) 个元素。
        I_v ← I_v ∪ I_uv
    i ← 从 I 中选择一个元素 v
    C_i.append(v)
返回 C_1, ..., C_k
```

直观地说，这些需求要求每个顶点（属于 G）都至少有一单位的流经过，这直接对应于路径覆盖条件，即每个顶点至少由一条路径覆盖。事实上，G d 的任意可行 st-流 f 的流分解（回顾第 2 节）都对应于 G 的一个大小为 $|f|$ 的路径覆盖。此外，G d 的最小流 f^* 的任意分解都对应于 G 的一个 MPC，因此详见 $k = |f^*|$ [6, 第 2.3 节（完整版）]。图 3 通过一个示例阐释了这些想法。

由于在 MPC 中，每个顶点最多只能属于 $|V|$ 条路径，因此该问题可以归约为最大流 [2, 定理 3.9.1]。我们在下面的引理中总结这些结果。

引理 2（改编自 [6, 定理 2.2（完整版本）]）。我们可以在时间 $O(T_{MF}(|E|)$ 内计算出图 $f^* \mathcal{G}$ d 的一个流，使得 f^* 的每个流分解都对应于 G的一个最大反链，其中 $T_{MF}(|E|)$ 是求解最大流所需的时间。

在这种情况下，分解算法很简单：从 s 开始，沿着正流边构成的路径 P 前进，直到到达 t，然后更新 f^*，将 P 的边上的流量减一。重复此过程，直到流量耗尽。分解过程中得到的 st-路径可以轻松转换为 G 的 MPC $\mathcal{P}$（修剪 s 和 t，并在每条路径上将 v^i, v^o 替换为 v，例如参见 [27]）。

如果实现得当（例如，参见 [26, 引理 1.11（完整版）]），前一个算法的运行时间为 $O(\|\mathbb{P}\|)$，并且会输出一个有效的 MCC $\mathcal{P}$（回想一下，每个 MPC 都是一个 MCC）。然而，如图 1 所示，$\|\mathbb{P}\|$ 在最坏情况下可能达到 $\Omega(k|V|)$。我们的算法通过直接从 f^*中提取最小链分解（MCD，回顾第 2 节）来突破这一障碍。

算法 2：我们的 MCC 算法运行时间为 $O(T_{MF}(|E|) + (|V|+|E|) \log k)$，其中 $T_{MF}(|E|)$ 是求解最大流所需的时间。该算法使用了第 3 节中的增强型可合并字典。输入：有向无环图 $G = (V, E)$。输出：G 的最小链分解 $C = C_1, \ldots, C_k$。 1 $(G, d) \leftarrow$ 构建第 4 节中详细说明的 G 的流归约 2 $f^*, k = |f^*| \leftarrow$ 使用引理 2 得到 (G, d) 的最小流 3 将 C_i 初始化为空列表，用于 $i \in \{1, \ldots, k\}$ 4 初始化可合并字典，以维护 $\{1, \ldots, k\}$ 5 $I_s \leftarrow \{1, \ldots, k\}$ 6 按拓扑序遍历 $v \in V$ do 7 $I_v, I_s \leftarrow$ SIZE-SPLIT(I_s, $f^*(s, v^i)$) 8 for $u^o \in N^-(v^i)$ do 9 $I_{uv}, I_u \leftarrow$ SIZE-SPLIT($I_u, f^*(u^o, v^i)$) 10 $I_v \leftarrow$ 合并(I_v, I_{uv}) 11 $i \leftarrow$ SOME(I_v) 12 C_i.append(v) 13 返回 $C_1, \ldots, C_k$

因此其总长度恰好为 $|V|$。

提取 MCD $C = C_1$、...、C_k 的主要思路是：对于每个顶点 $v \in V$，计算集合 $I_v \subseteq \{1, ..., k\}$，其中包含满足 v 会属于流分解 $D = P_1$、...、P_k 中的路径 $P_v = \{P_i \mid i \in I_v\}$ 的索引。注意，根据 G d 的构造，$I_s = I_t = \{1, ..., k\}$。为了高效地计算这些集合，我们按照 G 的拓扑序处理顶点，例如 s、v_1^i、v_1^o、...、$v_{|V|}^i$、$v_{|V|}^o$、t（回顾一下，输入中给定了 v_1、...、$v_{|V|}$ 的 G 的拓扑序）。处理顶点 v 时，我们按如下方式计算 I_v：对于每个 $u \in N^-(v)$，从 I_u 中恰好取出 $f^*(u, v)$ 个元素，并将这些元素记为 I_{uv}。然后，我们计算 I_v，即并集 $\cup_{u \in N^-(v)} I_{uv}$。最后，任取一个元素 $i \in I_v$，并将 v 追加到 C_i。算法 1 给出了该算法的伪代码。

请注意，由于顶点是按拓扑序处理的，因此它们会按正确的顺序加入各自的链中。

此外，由于只有在两个顶点之间存在边时，索引才会从一个顶点移到另一个顶点，因此相邻顶点之间总是存在一条位于 G 中的路径，列表 C_i 也就对应着恰当的链。最后，将每个顶点只加入这样一条链中，确保最终结果确实是一个 MCD（每个顶点恰好属于一条链）。

该算法的一个重要方面是，恰好有 $f^*(u^i, v^i)$（$f^*(s, v^i)$）个元素从 I_u（I_s）中取出。得益于 f^* 的流守恒，这一步总是可行的。

引理 3。给定一个以 k 为宽度的有向无环图 $G = (V, E)$ 作为输入，算法 *1* 计算出

G 的最小链分解 $\mathcal{C} = C_1$、……、C_k。

证明。 根据引理 2， f^* 的每个流分解都对应于G的一个 MPC。我们证明每个 C_i 都是 V的一条链；由于每个顶点恰好被加入一个 C_i，因此我们得出 C = C_1、……、C_k 是一个最小链分解。归纳地，若 v 在链 C_i中于 v' 之后加入，则 v' 在 G中可达 v 。事实上，$i \in I_v$，尤其是，对于某个 $u \in N^-(v)$，有 $i \in I_u$；并且根据归纳假设， v' 在 G 中可达 u，因此也可达 v。 □

此外，由于执行的操作只有拆分和合并，因此在算法执行过程中的任意时刻，集合 I_v和 I_{uv}构成{1、……、k}的一个划分。

将集合 I_v和 I_{uv}简单地实现为链表后，我们可以在常数时间内执行合并和选取元素操作，但每次拆分需要$O(f^*(u^o,\ v^i))$时间（以及 $O(f^*(s,\ v^i))$时间），因此总共需要 $O(|\mathrm{P}|)$时间。不过，我们可以将集合 I_v和 I_{uv}实现为第 3 节介绍的增强型可合并字典，从而缩短总运行时间。算法 2 给出了最终结果。

定理 1. 给定一个宽度为 k 的有向无环图 $G = (V, E)$，我们可以在 $O(T_{MF}(|E|) + (|V| + |E|) \log k)$ 时间内计算出最小链覆盖，其中 $T_{MF}(|E|)$ 是求解最大流所需的时间。

证明. 算法的正确性可由前面的讨论和引理 3 得出，因为算法 2 是算法 1 的一种实现。构建流归约需要 $O(|V| + |E|)$ 时间，而根据引理 2，求得最小流 f^*需要$O(T_{MF}(|E|))$ 时间。其余运行时间来自对可合并字典的操作调用。SOME 被调用 $O(|V|)$ 次，而 SIZE-SPLIT 和 MERGE 各调用一次，对应于流归约中的每条边，即调用 $O(|\mathrm{E}|) = O(|V| + |E|)$ 次。应用引理 1 可得，$O(|V| + |E|)$ 次操作调用的总时间为 $O((|V| + |E|) \log k)$。

□

我们以一个工具总结本文的结果，该工具可以高效地从流 f 中提取一组顶点不相交链 C，这组链编码了一组路径 P，而该流将 P 编码为流分解。如果该问题可以建模为最大流/最小费用流问题，那么 Chen 等人 [9] 的结果使我们能够在近线性时间内解决这类问题。举个简单的例子，我们可以在近线性时间内解决 ℓ-覆盖问题（找出一组覆盖最多顶点的 ℓ 顶点不相交链）。

推论 2. 设 $G = (V, E)$ 为一个有向无环图， $f : E \to \mathbb{N}_0$ 为一组 $|f|$ 路径 $\mathcal{P}$ 的流编码，其流分解由权重为1 的路径组成。在 $O((|V| + |E|) \log |f|)$ 时间内，我们可以计算出一组 $|f|$ 顶点不相交链 $\mathcal{C}$，也可以通过从$\mathcal{P}$中移除重复顶点来得到这组链。

参考文献

[1] Amitabha Bagchi、Adam L Buchsbaum 和 Michael T Goodrich。偏置跳表。*Algorithmica*，42：31–48，2005年。

[2] Jørgen Bang-Jensen 和 Gregory Z Gutin。有向图：理论、算法与应用。Springer Science & Business Media，2008年。[3] Philip Bille、Mikko Berggren Ettienne、Travis Gagie、Inge Li Gørtz 和 Nicola Prezza。解压 Lempel-Ziv 压缩文本。载于第 *30* 届数据压缩会议（*DCC 2020*）论文集，第 143–152 页。IEEE，2020年。[4] Manuel C´aceres、Massimo Cairo、Andreas Grigorjew、Shahbaz Khan、Brendan Mumey、Romeo Rizzi、Alexandru I Tomescu 和 Lucia Williams。宽度有助于也妨碍流拆分。载于第 *30* 届欧洲算法研讨会（*ESA 2022*）论文集，第 31:1–31:14 页，2022年。[5]Manuel C´aceres、Massimo Cairo、Brendan Mumey、Romeo Rizzi 和 Alexandru I Tomescu。用于计算有向无环图宽度的线性时间参数化算法。载于第 *47* 届图论概念国际研讨会（*WG 2021*）论文集，第 257–269 页。Springer，2021年。[6]Manuel C´aceres、Massimo Cairo、Brendan Mumey、Romeo Rizzi 和 Alexandru I Tomescu。通过稀疏化、收缩和拼接，以参数化线性时间求解最小路径覆盖。载于第 *33* 届 *ACM-SIAM* 离散算法研讨会（*SODA 2022*）论文集，第 359–376 页。SIAM，2022年。[7]Manuel C´aceres、Brendan Mumey、Edin Husi´c、Romeo Rizzi、Massimo Cairo、Kristoffer Sahlin 和 Alexandru I Tomescu。通过有向无环图所有路径覆盖中均出现的路径实现多重组装的安全性。*IEEE/ACM* 计算生物学与生物信息学汇刊，19(6)：3673–3684，2021年。[8] Ghanshyam Chandra 和 Chirag Jain。使用对间隙敏感的共线链进行序列到图的比对。载于第 *27* 届计算分子生物学研究国际年会（*RECOMB 2023*）论文集，第 58–73 页。Springer，2023年。[9] Li Chen、Rasmus Kyng、Yang P Liu、Richard Peng、Maximilian Probst Gutenberg 和 Sushant Sachdeva。近线性时间内的最大流与最小费用流。载于第 *63* 届 *IEEE* 计算机科学基础年会（*FOCS 2022*）论文集，第 612–623 页。IEEE，2022年。[10] Yangjun Chen 和 Yibin Chen。一种高效的图可达性查询应答算法。载于第 *24* 届国际数据工程会议（*ICDE 2008*）论文集，第 893–902 页。IEEE，2008年。[11] Yangjun Chen 和 Yibin Chen。关于图分解。载于第 *4* 届 *IEEE* 大数据与云计算国际会议（*BDCloud 2014*）论文集，第 777–784 页。IEEE，2014年。

[12] Eleonor Ciurea 和 Laura Ciupala。最小流的串行与并行算法。*Journal of Applied Mathematics and Computing*，15(1):53–75，2004。[13] Nicola Cotumaccio 和 Nicola Prezza。有限自动机的索引与压缩。载于*Proceedings of the 32nd ACM-SIAM Symposium on DiscreteAlgorithms (SODA 2021)*，第 2585–2599 页。SIAM，2021。[14] Rene De La Briandais。使用可变长度键进行文件搜索。载于*Paperspresented at the the March3-5, 1959, western joint computer conference*，第 295–298 页，1959。[15] Robert P Dilworth。偏序集的一个分解定理。*Annals of Mathematics*，51(1):161–166，1950。[16] Martin Farach 和 Mikkel Thorup。Lempel—Ziv 压缩字符串中的字符串匹配。*Algorithmica*，20(4):388–404，1998。[17] Stefan Felsner、Vijay Raghavan 和 Jeremy Spinrad。小宽度偏序与小 Dilworth 数图的识别算法。*Order*，20(4):351–364，2003。[18] Delbert R Fulkerson。关于偏序集的 Dilworth 分解定理的注记。*Proceedings of the American Mathematical Society*，7(4):701–702，1956。[19] Loukas Georgiadis、Haim Kaplan、Nira Shafrir、Robert E Tarjan 和 Renato F Werneck。可合并树的数据结构。*ACM Transactionson Algorithms*，7(2):1–30，2011。[20] Leo J Guibas 和 Robert Sedgewick。平衡树的双色框架。载于*Proceedings of the 19th Annual Symposium on Foundations ofComputer Science (FOCS 1978)*，第 8–21 页。IEEE，1978。[21] John E Hopcroft 和 Richard M Karp。用于二分图最大匹配的 $n^{5/2}$ 算法。*SIAM Journal on Computing*，2(4):225–231，1973。.. .[22] John Iacono 和 Ozg¨ur Ozkan。可合并字典。载于*Proceedings ofthe 37th International Colloquium onAutomata, Languages, and Program-ming (ICALP 2010)*，第 164–175 页。Springer，2010。[23] Arthur B Kahn。大型网络的拓扑排序。*Communications ofthe ACM*，5(11):558–562，1962。[24]Adam Karczmarz。一个简单的可合并字典。载于*Proceedings ofthe 15th Scandinavian Symposiumand Workshops on Algorithm Theory(SWAT 2016)*，第 7:1–7:13 页，2016。[25] Donald E Knuth。*The art of computer programming: Volume 3: Sortingand Searching*。Addison-Wesley Professional，1998。

[26]Shimon Kogan 和 Merav Parter。击败用于 $n^{1/3}$-有向捷径的矩阵乘法。载于*Proceedings ofthe 49thInterna-tional Colloquium on Automata, Languages, andProgramming (ICALP2022)*。Schloss Dagstuhl-Leibniz-Zentrum f¨ur Informatik，2022。完整版本见 https:\www.weizmann.ac.il/math/parter/sites/math.parter/files/uploads/main-lipics-full-version_3.pdf。[27] Shimon Kogan 和 Merav Parter。用于缩短直径和最小链覆盖的更快且统一的算法。载于*Proceedings ofthe 34th Annual ACM-SIAM Symposiumon Discrete Algorithms (SODA2023)*，第 212–239 页。SIAM，2023。[28] Katherine Jane Lai。并集-拆分-查找问题的复杂性。麻省理工学院硕士学位论文，2008。[29] Veli M¨akinen、Alexandru I Tomescu、Anna Kuosmanen、Topi Paavilainen、Travis Gagie 和 Rayan Chikhi。小宽度有向无环图上的稀疏动态规划。*ACM Transactions on Algorithms*，15(2):1–21，2019。[30] Simeon C Ntafos 和 S Louis Hakimi。有向图中的路径覆盖问题及其在程序测试中的应用。*IEEE Transactions on Software En-gineering*，5(5):520–529，1979。[31] Simon J Puglisi 和 Massimiliano Rossi。小空间中的 Lempel-Ziv 解压缩。载于*Proceedings of the 29th Data Compression Conference(DCC 2019)*，第 221–230 页。IEEE，2019。[32] Nicola Rizzo、Manuel Caceres 和 Veli M¨akinen。图中极大精确匹配的链式连接。*arXiv preprint arXiv:2302.01748*，2023年。[33] Robert E Tarjan。边不相交生成树与深度优先搜索。*ActaInformatica*，6(2):171–185，1976。[34]Cole Trapnell、Brian A Williams、Geo Pertea、Ali Mortazavi、Gordon Kwan、Marijke J Van Baren、Steven L Salzberg、Barbara J Wold 和 Lior Pachter。RNA-Seq 转录本组装与定量揭示了细胞分化过程中的未注释转录本和异构体转换。*Nature Biotechnology*，28(5):511，2010。[35] Jan van den Brand、Yin Tat Lee、Yang P Liu、Thatchaphol Saranurak、Aaron Sidford、Zhao Song 和 Di Wang。稠密实例的近线性时间最小费用流、MDP 及 ℓ_1-回归。载于*Proceedings of the53rd Annual ACM SIGACT Symposiumon Theory of Computing (STOC2021)*，第 859–869 页，2021。