节点扩展下维护三连通分量

Simon D. Fink ！ 信息学与数学学院，帕绍大学，德国

Ignaz Rutter ！

信息学与数学学院，帕绍大学，德国 y

摘要

SPQR 树是图绘制中的核心组成部分，在计算机科学的许多其他领域也十分重要。自问世以来，它们始终与动态算法密切相关，这些算法用于在边插入（后来也包括删除）时维护平面性和三连通性等信息。本文聚焦于一种特殊的动态更新：将顶点扩展为任意双连通图，同时维护 SPQR 树及其他信息。这也使我们能够通过识别两个顶点各自关联的边，高效地合并两棵 SPQR 树。为此，我们依据一种公理化定义，将 SPQR 树提升为一种独立数据结构，使其能够独立于可能由其推导而来的图进行修改。通过修改这一结构，我们现在可以观察 SPQR 树所表示的图如何变化，而无需在表示图发生变化后再推理需要对 SPQR 树进行哪些更新。

采用高效的扩张与合并，使我们能够将 Bläsius 等人的同步平面性算法的运行时间从 $[8]$ (O)m^2 降至 $O(m\cdot\ \Delta)$，其中 Δ 为最大管道度。这也缩短了求解若干受约束平面性问题所需的时间，例如簇平面性问题的运行时间从 $O((n+d)^2)$ 降至 $O(n+d\cdot\ \Delta)$，其中 d 为簇边界与边之间的交叉总数，Δ 为单个簇边界上的最大边交叉数。

2012年 ACM主题分类 计算数学 → 图算法

关键词与短语 SPQR树、动态算法、簇平面性

资助 由德国研究基金会项目 RU-1903/3-1 资助。

arXiv:2301.03972v1 [cs.DS] 10 Jan 2023

1 引言

SPQR树是一种数据结构，用于表示图在分离对处的分解；分离对即移除后会使图断开的顶点对。通过这种分解得到的分量称为骨架。SPQR树是许多图可视化技术的核心组成部分，可用于平面性测试及其变体 [13, 19, 29, 31, 39]，也可用于计算嵌入和布局 [3, 7, 11, 20, 28, 42]；有关图绘制应用的综述，请参见 [37]。除图可视化之外，它们还应用于最小生成树 [6, 17], 、三角剖分 [5], 和交叉优化 [28, 42]等领域。它们在图论乃至计算机科学之外也有许多应用，例如用于创建集成电路 [14, 44],、业务流程建模 [40],、电气工程 [24],、理论物理 [41]和基因组学 [22]。

最初，SPQR 树由 Di Battista 和 Tamassia 提出，用于增量平面性测试 [16, 19]。因此，即使在最初的形式下，SPQR 树就已支持以添加边的形式进行动态更新。它们很快便被用于其他在线问题 [18, 17]。除了上述应用之外，这也促成了一系列后续论文，这些论文改进了增量数据结构的运行时间 [38, 39, 43]，并将其扩展为全动态结构，即允许在 $O(\sqrt{n})$ 时间内插入和删除顶点及边 [21, 27],，其中 n 是图中的顶点数。最近，Holm 和 Rotenberg 描述了一种全动态算法，可在每次操作 [31, 32] 耗时 $O(\log^3 n)$ 的情况下维护平面性和三连通性信息（动态 SPQR 树算法的简要发展史亦见该文）。

本文考虑一种增量设置，其中允许通过单次操作将一个顶点 v 扩展为任意双连通图 G_ν。采用 Holm 和 Rotenberg [31], 的方法，首先移除 v 及其关联边，然后增量插入 G_ν，所需时间为 $O((\deg(v) + |G_\nu|) \cdot \log^3 n)$。我们通过一种简单得多的算法将其改进为 $O(\deg(v) + |G_\nu|)$，因此该算法在实际应用中也更有可能提升性能。此外，我们的方法还支持高效地合并两棵 SPQR 树，具体如下。给定两个双连通图 G_1，G_2，它们分别包含顶点 v_1，v_2，并给定这两个顶点的关联边之间的一个双射，我们通过在 G_1 中用 $G_2 - v_2$ 替换 v_1，并依据给定的双射识别边，构造新图 G。给定 G_1 和G_2的 SPQR 树，我们证明可以在 $O(\deg(v_1))$ 时间内找到 G 的 SPQR 树。具体而言，我们提出一种数据结构，支持以下操作：$\texttt{InsertGraph}_{\text{SPQR}}$ 以与扩展子图大小成线性关系的时间扩展单个顶点；$\texttt{Merge}_{\text{SPQR}}$ 以与被替换顶点的度成线性关系的时间合并两棵 SPQR 树；`IsPlanar` 以常数时间判断当前表示图是否为平面图；`Rotation` 以常数时间给出三连通骨架中某个顶点的两种可能平面旋转次序之一。此外，我们的数据结构还可扩展为：为所有三连通骨架生成一致的平面嵌入，并检测任意两个顶点之间是否存在三条不同路径；所有操作的时间复杂度额外增加一个 $\alpha(n)$ 因子，其中 α 为反阿克曼函数。

我们方法的核心思想是：扩张顶点 v 所影响的 SPQR树 子树大小与 v的度呈线性关系，但其中可能包含任意大的骨架。在 SPQR树 的“非规范化”版本中，受影响的环形（‘S’）骨架可以轻松拆分为常数大小；而我们设计了一种定制的拆分操作，以限制三连通（‘R’）骨架的大小。这样一来，受影响结构的大小便受限于与 v 的度呈线性关系，并使我们能够高效地执行扩张。

除了对这种数据结构的描述外，本文的技术贡献是

问题	运行时间		
	在 [8] 之前	使用 [8]	通过本文
原子性 / 可嵌入性 同步平面性	$O(m^8)$ [26]	$O(m^2)$	$O(m\cdot\Delta)$
簇平面性	$O((n+d)^8)$ [26]	$O((n+d)^2)$	$O(n+d\cdot\Delta)$
连通同时平面嵌入	$O(n^{16})$ [26] 双连通：$O(n^2)$ [10]	$O(n^2)$	$O(n\cdot\Delta)$
部分 PQ约束 平面性	双连通：$O(m)$ [10]	$O(m^2)$	$O(m\cdot\Delta)$
行-列 独立 NodeTrix 平面性	双连通：$O(n^2)$ [35]	$O(n^2)$	$O(n\cdot\Delta)$
条带平面性	$O(n^8)$ [4, 26] 固定嵌入：poly [4]	$O(n^2)$	$O(n\cdot\Delta)$

表1 在《同步平面性》 [8] 发表之前，各类受约束平面性问题已知的最佳运行时间；按照 [8]中所述使用该算法时的运行时间；以及将其与本文的加速方法结合使用时的运行时间。以“双连通”开头的运行时间仅适用于某些具有某种形式双连通性的问题实例。变量 n 和 m 分别表示问题实例中的顶点数和边数。变量 d 表示簇平面性实例中边与簇边界的交叉数，而 Δ 表示相应同步平面性实例中的最大管道度。根据问题的不同，该值不超过与单个簇边界相交的边数上限或输入实例中的最大顶点度。

因此，我们可以从数据结构推导出表示图，而不是根据图计算数据结构。这样，我们便可以对数据结构进行或多或少任意的更改（同时遵守其一致性准则），并观察图如何变化，而不必推理图的哪些更改需要对数据结构进行哪些更新。

其次，我们将说明如何利用我们的数据结构，将 Bläsius 等人 [8]提出的求解同步平面性问题的算法运行时间从 $O(m^2)$改进到$O(m\cdot\Delta)$，其中 Δ 是最大管道度（即，具有同步约束、要求其旋转与另一顶点相同的顶点的最大度）。同步平面性可用于建模并求解种类繁多的受约束平面性问题，表1概述了从这一加速中受益的问题。其中包括著名的簇平面性问题，其复杂度在开放了30年后，Fulek 和 Tóth 提出了一个运行时间为 $O((n+d)^8)$的算法 2019 [26],，其中d 为簇边界与边之间的交叉总数。不久之后，Bläsius 等人 [8] 给出了一个运行时间为 $O((n+d)^2)$的解法。我们将其改进为 $O(n+d\cdot\Delta)$，其中Δ 是单个簇边界上的边交叉数上限。

本文结构如下。第2节概述本文所用的定义。第3节介绍骨架分解，并说明它与SPQR树的关系。第4节扩展这一数据结构，使其能够拆分

三连通分量。在第5节中，我们利用这一特性，确保用一个新图替换某个顶点时，SPQR树中受影响的部分很小。第6节详细介绍同步平面性和簇平面性的背景，并说明如何利用我们的结果缩短求解这些问题所需的时间。

2 预备知识

在本文的语境中，$G = (V, E)$是一个（通常为双连通且无环的）多重图，具有 n 个顶点 V 和 m 条（可能平行的）边 E。对于顶点 v，我们用 $N(v)$表示其开邻域（不包括 v 自身）。对于双射或匹配 ϕ，我们称 $\phi(x)$为元素 x的配对元素。我们用 $A \dot\cup B$ 表示两个不相交集合 A，B的并集。

分离 k集是指一个顶点 k 集合，移除该集合后会使连通分量的数量增加。分离 1 集称为割点，而分离 2 集称为分离对。如果一个连通图没有割点，则称其为双连通图。如果一个双连通图没有分离对，则称其为三连通图。极大双连通子图称为块。每个分离对都会将图分成若干桥，即无法通过移除或拆分该分离对中的顶点而断开的极大子图。键是仅由两个极点顶点及连接它们的多条平行边构成的图；多边形是一个简单环，而刚性图是任意一个简单的三连通图。轮是一个带有额外中心顶点的环，该中心顶点与所有其他顶点相连。

最后，本文的核心概念——扩张——正式定义如下。设G_α, G_β是两个图，其中 G_α 含有一个顶点 u，而 G_β含有 $|N(u)|$ 个标记顶点，并且存在一个双射 ϕ，将 u 的邻点映射到 G_β中的标记顶点。用$G_\alpha[u \to_\phi G_\beta]$ 表示由 G_α, G_β 的不交并经如下操作得到的图：将 u 的每个邻点 x 与 G_β 中对应的标记顶点 $\phi(x)$识别，并删除u；也就是说，所得图 G_α 是将顶点 u 扩张为 G_β后得到的图。

3 骨架分解

表示图 $G_\mathcal{S} = (V, E)$的骨架结构 $\mathcal{S} = (\mathcal{G}, \text{origV}, \text{origE}, \text{twinE})$由一组彼此不相交的骨架 图以及三个满射映射 twinE,origE 和 origV 组成，并满足以下条件：表示

- $\mathcal{G}$ 中的每个骨架 $G_\mu = (V_\mu, E_\mu^{\text{real}} \dot\cup E_\mu^{\text{virt}})$都是一个多重图，其中每条边要么属于$E_\mu^{\text{real}}$，因而称为实 边；要么属于 E_μ^{virt}，因而称为虚拟边。双射 twinE：$E^{\text{virt}} \to E^{\text{virt}}$ 将所有
- 虚边 $E^{\text{virt}} = \bigcup_\mu E_\mu^{\text{virt}}$配对，使得 twinE($e$) $\neq e$ 且 twinE^2 =id。满射 origV：$\bigcup_\mu V_\mu \to V$ 将所有骨架顶点映射到图顶点。双射 origE：$\bigcup_\mu E_\mu^{\text{real}} \to E$ 将所有实边映射
- 到图的边集 E。

注意，每个骨架的每个顶点和每条边都恰好属于三个映射之一的定义域。由于这些映射是满射，V 和 E 恰好分别是 origV 和 origE 的像。对于每个顶点 $v \in G_\mathcal{S}$，包含满足 origV(v') $= v$ 的分配顶点 v' 的骨架称为 v的分配骨架。此外，设 $T_\mathcal{S}$ 为一个图，其中每个节点 μ 对应于$\mathcal{G}$的一个骨架 G_μ。如果两个节点对应的骨架包含一对彼此匹配的虚边，则这两个节点在 $T_\mathcal{S}$ 中相邻。

如果骨架结构满足以下条件，我们就称其为骨架分解：**1（双连通）**每个骨架都是双连通的。**2（树）**图 $T_\mathcal{S}$ 是简单、无环、连通且无环的，即一棵树。**3（原始映射单射性）**对于每个骨架 G_μ，限制$\text{origV}|_{V_\mu}$ 是单射的。

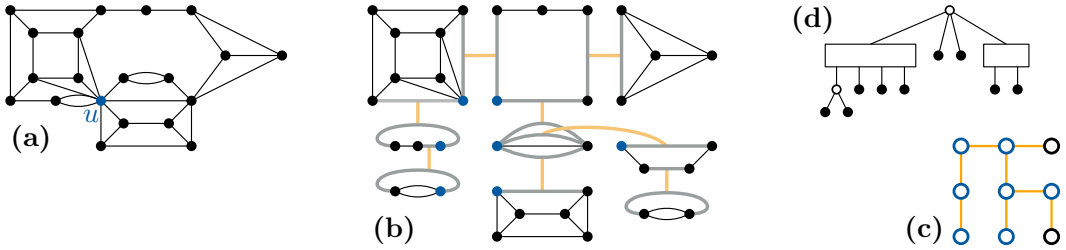

图1 骨架分解$\mathcal{S}$的不同视图。**(a)** 图 $G_{\mathcal{S}}$，其中顶点 u 以蓝色标记。**(b)** 的骨架。虚边以灰色绘制，其匹配的 twinE 以橙色显示。 u 的分配顶点以蓝色标记。**(c)** 树 $T_{\mathcal{S}}$。 u 的分配骨架以蓝色标记。**(d)** 顶点 u 的嵌入树，如第6.2节所述。P 节点显示为白色圆盘，Q节点显示为大矩形。嵌入树的叶节点对应于与 u关联的边。

4（原始边真实性）对于每条实边 uv，origE(uv) 的端点是 origV(u) 和 origV(v)。5（原始虚拟边一致性）设 uv 和 $u'v'$ 是两条虚边，且 $uv = \text{twinE}(u'v')$。对于它们各自的骨架 G_μ 和 G'_μ（其中 μ 和 μ' 在 $T_{\mathcal{S}}$ 中相邻），有 $\text{origV}(V_\mu) \cap \text{origV}(V_{\mu'}) = \text{origV}(\{u, v\}) = \text{origV}(\{u', v'\})$。

6（子图）任意顶点的分配骨架 $G_{\mathcal{S}}$ 构成 $T_{\mathcal{S}}$的一个连通子图。图1展示了$\mathcal{S}$、 $G_{\mathcal{S}}$和 $T_{\mathcal{S}}$的一个示例。我们将仅含一个骨架的骨架分解称为 G_μ 平凡的。请注意，在这种情况下， G_μ 与 $G_{\mathcal{S}}$同构，并且origE 和origV实际上是这两个图的边和顶点之间的双射。

为了对三连通分量分解进行建模，我们在骨架分解 S $=$ (G, origV, origE, twinE) 上定义操作`SplitSeparationPair` 及其逆操作 `JoinSeparationPair`。对于 `SplitSeparationPair`，设 u, v是骨架 G_μ 的一个分离对，且 (A, B) 是 u 和 v之间各桥的一个非平凡二分划分。[1]应用 `SplitSeparationPair`(S, (u, v), (A, B)) 后，得到如下骨架分解 S$'$ $=$ (G$'$, origV$'$, origE$'$, twinE$'$)。在 G$'$ 中，我们将 G_μ 替换为两个骨架 G_α, G_β，其中 G_α 由 $G_\mu[A]$ 添加一条位于 u 和 v之间的新虚边 e_α 得到。对于 G_β，同理，将其替换为 $G_\mu[B]$ 和 e_β。我们设 twinE$'$(e_α) $= e_\beta$，且 twinE$'$(e_β) $= e_\alpha$。请注意，origV 将 e_α 和 e_β 的端点映射到相同的顶点。所有其他骨架及其上定义的映射均保持不变。

对于 `JoinSeparationPair`，考虑虚边 e_α、 e_β，其 twinE(e_α) $= e_\beta$，并令 $G_\beta \neq G_\alpha$ 为它们各自的骨架。应用 `JoinSeparationPair`($\mathcal{S}$ e_α) 得到骨架分解 $\mathcal{S}' = (\mathcal{G}'$, origV$'$, origE$'$, twinE$'$)，如下所示。在 $\mathcal{G}'$ 中，我们将 G_α 与G_β 合并，通过识别映射到 $G_{\mathcal{S}}$同一顶点的 e_α 和 e_β 的端点，形成新的骨架 G_μ。此外，我们移除 e_α 和 e_β。所有其他骨架及其上定义的映射均保持不变。

这两种操作的主要特点是：它们在 $T_{\mathcal{S}}$ 中拆分节点或收缩边的同时，不会影响由骨架分解表示的图；这一点可以通过检查各项条件来验证。

▶**引理1**。 在骨架分解 $S = (G$,origV,origE,twinE) 上应用 `SplitSeparationPair` 或 `JoinSeparationPair`，会得到一个骨架分解 $S' = (G'$, origV$'$,

[1] 请注意，一个桥可能仅由连接 u 和 v 的一条边构成，并且每个桥都包含顶点 u 和 v。

origE′, twinE′) 其表示图 $G_{\mathcal{S}'} = G_{\mathcal{S}}$保持不变。

证明。我们首先检查由 `SplitSeparationPair` 返回的骨架分解 S′ 是否仍满足所有条件。由于 (A, B) 是一个非平凡二分划分，每个集合都至少包含一条桥。再加上 e_α （以及 e_β），这条桥确保 G_α （以及 G_β）仍然是双连通的，从而满足条件 1（双连通）。该操作将 $T_{\mathcal{S}}$ 中的一个节点 μ 拆分为两个相邻节点 α、β，它们的邻点分别由 A、B 中的虚边精确定义。因此，条件 2（树）仍然满足。映射 origV′、origE′ 和 twinE′ 显然仍满足条件 3（原始映射单射性）和条件 4（原始边真实性）。我们恰好复制了相邻骨架 G_α 和 G_β 中的两个节点 u 和 v。由于 G_μ 满足条件 3 （原始映射单射性）， G_α 和 G_β 不会共享其他映射到 $G_{\mathcal{S}'}$ 中同一顶点的顶点。因此，条件 5（原始虚拟边一致性）仍然满足。

只有当由某个顶点 $z \in G_{\mathcal{S}'}$ 的分配骨架构成的 $T_{\mathcal{S}'}$ 子图不再连通时，才可能违反条件 6（子图）。只有当 G_α 和 G_β 中只有一个是 z 的分配骨架，而另一个还有一个更远的邻居，且该邻居也是 z 的分配骨架时，才会发生这种情况。不失一般性，假设 G_α 和 G_β 的邻居 G_ν 是 z 的分配骨架，而 G_β 本身不是。由于 G_ν 和 G_β 在 $T_{\mathcal{S}'}$ 中相邻，因此存在虚边 $xy = \text{twinE}'(x'y')$，其端点为 $xy \in G_\beta$ 和 $x'y' \in G_\nu$。输入实例中也存在相同的虚边，只是其中 $xy \in G_\mu$ 和 μ （而非 β）以及 ν 在 $T_{\mathcal{S}}$ 中相邻。由于输入实例满足条件 5（原始虚拟边一致性），因此有 $z \in \text{origV}(V_\nu) \cap \text{origV}(V_\mu) = \text{origV}(\{x, y\}) = \text{origV}(\{x', y'\})$。由于 $\text{origV}(\{x, y\}) = \text{origV}'(\{x, y\})$，这与 G_β 不是 z 的分配骨架相矛盾。

最后，映射 origE 保持不变，而 origV 唯一的变化是纳入两个映射到已有顶点的新顶点。由于输入和输出实例均满足条件 4（原始边真实性），这不会影响表示图 $G_{\mathcal{S}'}$。

现在考虑 `JoinSeparationPair` 返回的骨架分解 $\mathcal{S}'$。识别不同连通分量中的不同顶点不会影响它们的双连通性，因此条件 1（双连通）仍然成立。该操作实质上是在 $T_{\mathcal{S}}$ 中收缩并删除一条边，这不会影响 $T_{\mathcal{S}'}$ 作为满足条件 2（树）的树。注意，输入实例满足条件 2（树）也确保了 G_α 和 G_β 是两个不同的骨架。由于输入实例还满足条件 5（原始虚拟边一致性），相邻骨架 G_α 和 G_β 中各恰有两个顶点，其 origV 映射到 $G_{\mathcal{S}}$ 中的同一顶点。这两个顶点必定属于使这两个骨架相邻的 twinE 对，因此它们正是我们相互识别的两对顶点。由此，$\text{origV}|_{V_\mu}$ 仍然是单射，满足条件 3（原顶点单射）。由于我们没有修改任何实边或其他虚边，映射 origV′ 和 origE′ 显然仍满足条件 4（原始边真实性）。由于每个图顶点的分配骨架构成一个连通子图，将两个骨架合并不会改变它们与任何邻居的交集，因此条件 5（原始虚拟边一致性）仍然成立。最后，收缩一条树边不会导致条件 6（子图）中的任何子图变得不连通，因此该条件也仍然成立。同样，我们没有修改 origE，而条件 5（原始虚拟边一致性）确保 origV 将合并顶点的两对顶点映射到了 $G_{\mathcal{S}}$ 中的同一顶点。因此，表示图 $G_{\mathcal{S}'}$ 保持不变。◀

这为我们提供了另一种查找表示图的方法：穷尽地连接所有骨架，直到只剩下一个，从而得到唯一的平凡骨架分解：

▶**引理 2.** 对骨架分解 $\mathcal{S}$ =(G, origV,origE,twinE) 穷尽地应用 `JoinSeparationPair`，会得到一个平凡骨架分解 $\mathcal{S}' = (G', \text{origV}', \text{origE}', \text{twinE}')$，其中origE′ 和origV′ 定义了 G'_μ 与 $G_{\mathcal{S}'}$ 之间的同构。

证明。由于所有虚边都已匹配，且匹配的虚边总属于不同的骨架（条件 2（树）确保 T_S 无环），因此我们可以不断对一条虚边应用 JoinSeparationPair，直到虚边全部消失。由于 T_S 连通，这意味着我们总能得到一棵只有一个节点的树，也就是说，得到一个仅含单个骨架的实例。单次应用 JoinSeparationPair 会保持表示图不变，因此连续多次应用也会保持表示图不变。注意，origE′ 是一个双射，而满射 origV′ 在唯一剩余的骨架上也因条件 3（原顶点单射）而是单射，因此它整体上也是一个双射。结合条件 4（原始边真实性），这确保 G'_μ 中任意两个顶点 u 和 v 相邻，当且仅当 origV′(u) 和 origV′(v) 在 $G_{\mathcal{S}'}$ 中相邻。因此，origV′ 是一个保边双射，也就是一个同构。◀

关于骨架分解，尤其是操作 SplitSepa-rationPair now 的一个关键点是，它们对图在分离对处的分解进行建模。Di Battista 和 Tamassia [16] 将这种分解形式化为 *SPQR*树 [33, 36]，对于给定的图而言，该分解是唯一的；另见 [28, 30]。Angelini 等人 [1] 描述了一种在概念上等同于我们骨架分解的分解树。他们还提出了 SPQR树 的另一种定义，将其定义为满足更多性质的分解树。我们在此采用这一关于骨架分解的定义，但不要求三连通分量具有平面性，并允许虚边和实边同时出现在一个骨架中（即将叶节点 Q节点 合并到其父节点中）。

▶ **定义 3.** 骨架分解 $S = (G,\text{origV},\text{origE},\text{twinE})$ 是指：G 中的任一骨架要么是多边形、键，要么是三连通的（“刚性”）；并且 T_S 中相邻的两个骨架不会同时为多边形，也不会同时为键。这样的骨架分解是 G_S 的唯一 *SPQR*树。

分解树背后的经典思想与我们的骨架分解之间，主要区别在于，后者允许通过公理化方法对分离对进行分解。对于骨架分解，我们采用纯函数式的“机械”数据结构，而不依赖于或沿用既定的图结构。在我们的方法中，表示图是从数据结构（即 SPQR树）推导出来的，而不是根据图来计算数据结构。

4 扩展骨架分解

请注意，大多数骨架，尤其是多边形和键，都可以轻松地分解成更小的部分。唯一的例外是三连通骨架，它们无法通过我们目前定义的操作进一步拆分。当要修改三连通骨架中的一个顶点，而这些骨架可能远大于该顶点的直接邻域时，就会出现问题。为了解决这一问题，我们定义了另一组操作，允许将任意三连通分量中的顶点分离出来，并用一个（“虚拟”）占位顶点替换它。随后，该占位顶点指向一个包含实际顶点的较小分量，见图 2。不允许修改与占位顶点关联的边，因此我们称这些边为“占用”边。

形式上，为了追踪以这种方式拆分的分量，扩展骨架分解 $\mathcal{S} = (\mathcal{G},\text{origV},\text{origE},\text{twinE},\text{twinV})$中所需的结构定义如下。现在，骨架的形式为 $G_\mu = (V_\mu \cdot \cup V_\mu^{\text{virt}},\ E_\mu^{\text{real}} \cup E_\mu^{\text{virt}} \cup E_\mu^{\text{occ}})$。双射 twinV : $V_{\text{virt}} \to V_{\text{virt}}$ 匹配所有虚拟顶点 $V_{\text{virt}} = \bigcup_\mu V_{\text{virt}\mu}$，使得 twinV($v$) $\neq v$，twinV^2 =id。与虚拟顶点关联的边包含在 E_μ^{occ} 中，因此被视为占用；参见图 2b。类似于由 twinE 匹配的虚边，任何一对由 twinV 匹配的虚拟顶点都会在 T_S 中的骨架之间产生一条边。条件 2（树）

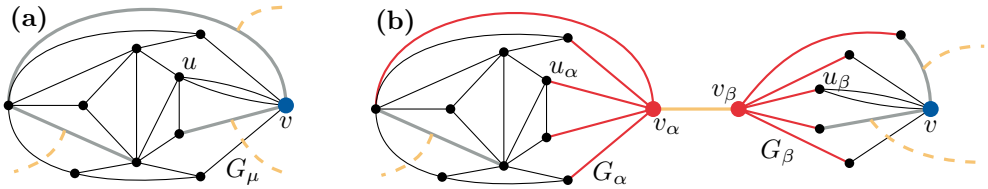

图 2 (a) 一个三连通骨架 G_μ，其中一个高亮顶点 v 与两条灰色虚边关联。(b) 应用 `IsolateVertex` 将 v 从骨架中分离出来后的结果。旧骨架 G_α 中的红色占用边构成一个以 v_α为中心的星形，而 G_β 中的红色占用边连接 v 的所有邻点，构成一个以 $v_\beta \neq v$为中心的星形。中心 v_α 和 v_β 是虚拟顶点，彼此匹配。v 的邻点 u 被拆分为顶点 u_α 和 u_β。

同样适用于由 twinV 产生的这些边，这尤其确保了 T_S 中不存在平行的 twinE 和 twinV 树边。类似地，条件 6（子图）中的连通子图也可以包含由 twinV 产生的树边。其他所有条件保持不变，但我们增加两个条件以确保 twinV 保持一致：

7（星形图）对于每个 v_α，若 v_β 满足 twinV(v_α) $= v_\beta$，则 deg(v_α) $=$ deg(v_β)。所有与 v_α 和 v_β 关联的边均为占用边，且端点各不相同（v_α 和 v_β除外）。反过来，每条占用边恰好与一个虚拟顶点相邻。8（orig-stars）设 v_α 和 v_β 仍为两个由 twinV 彼此匹配的虚拟顶点。对于它们各自的骨架 G_α 和 G_β（其中 α 和 β 在 T_S中相邻），有 origV(V_α) ∩origV(V_β) =origV($N(v_\alpha)$) =origV($N(v_\beta)$)。

注意，两个条件结合起来会在 v_α 的邻点与 v_β 之间产生一个双射 $\gamma_{v_\alpha v_\beta}$，因为当 origV 限制在单个骨架上时，它是单射（条件 3（orig- inj））， 且 deg(v_α) =deg(v_β)。操作 `SplitSeparationPair` 和 `JoinSeparationPair` 也可以应用于扩展骨架分解，从而得到一个不修改 twinV 的扩展骨架分解。为确保条件 7（星形图）和条件 8（orig-stars）不受这两个操作的影响，如果分离对中的某个顶点是虚拟顶点，则不能应用 `SplitSeparationPair`。

现在，操作 `IsolateVertex` 和 `Integrate` 分别使我们能够将顶点从三连通分量中分离出来，并将其重新整合。对于 `IsolateVertex`，设 v 是骨架 G_μ 的一个非虚顶点，使得 v 没有与之关联的占用边。在扩展骨架分解 S 上应用 `IsolateVertex`(S v)，会按如下方式得到扩展骨架分解 S′ =(G′, origV′, origE′, twinE′, twinV′)。将 v 的每个邻点拆分为两个不相邻顶点 u_α 和 u_β，其中 u_β 与连接 u 和 v 的所有边关联，而 u_α 保留 u 的所有其他边。我们设置 origV′(u_α) = origV′(u_β) = origV(u)。这样便创建了一个以 v 为中心的独立星形分量，我们将其移至骨架 G_β，同时将骨架 G_μ 重命名为 G_α。我们使用占用边将所有 u_α 连接到一个新的虚拟顶点 $v_\alpha \in V_\alpha^{\text{virt}}$，并使用占用边将所有 u_β 连接到一个新的虚拟顶点 $v_\beta \in V_\beta^{\text{virt}}$；见图 2。最后，我们设置 twinV′($v_\alpha$) $= v_\beta$、twinV′(v_β) $= v_\alpha$，并将 G_β 添加到 G′ 中。所有其他映射和骨架均保持不变。

对于 `Integrate`，考虑两个虚拟顶点 v_α，v_β，它们满足 twinV(v_α) $= v_\beta$，以及邻居 v_α 与 v_β 之间的双射 $\gamma_{v_\alpha v_\beta}$。应用 `Integrate`($\mathcal{S}$, ($v_\alpha$, v_β)) 得到扩展骨架分解 $\mathcal{S}' = (\mathcal{G}',$ origV′, origE′, twinE′, twinV′)，具体如下。我们通过根据 $\gamma_{v_\alpha v_\beta}$ 识别 v_α 和 v_β 的邻居，将两个骨架合并为一个骨架 G_μ（同时在 $\mathcal{G}'$ 中也将二者替换为该骨架）。此外，我们移除 v_α 和 v_β，以及与它们关联的占用边。其他所有映射和骨架均保持不变。

▶**引理 4**。 在扩展骨架分解 $S = (G,\text{origV},\text{origE},\text{twinE},\text{twinV})$ 上应用 `IsolateVertex` 或 `Integrate`，得到扩展骨架分解 $S' = (G',\text{origV}', \text{origE}', \text{twinE}', \text{twinV}')$，并满足 $G_{S'} = G_S$。

证明。 我们首先检查扩展骨架分解中所有条件是否仍然成立 $\mathcal{S}'$由 `IsolateVertex`返回。条件 1（双连通）仍然成立，因为 G_α的结构与 G_μ 相比保持不变，且骨架 G_β 是一个键。由于我们再次拆分 $T_\mathcal{S}$中的一个节点，条件 2（树）也仍然成立。由于 v_β 和 v_α的邻点映射到 $G_{\mathcal{S}'}$中的相同顶点，条件 3（原始映射单射性）、4（原始边真实性）和 5（原始虚拟边一致性）仍然成立。条件 7（星形图）和 8（orig-stars）根据构造成立。

最后，只有当由某个顶点 $z \in G_{\mathcal{S}'}$ 的分配骨架构成的 $T_{\mathcal{S}'}$ 子图不再连通时，条件 6（子图）才可能被违反。这只有在 G_α 和 G_β 中仅有一个是 z 的分配骨架，而另一个还有一个进一步的邻居 G_ν，且该邻居也是 z 的分配骨架时才会发生。注意，无论如何，ν 在 $T_\mathcal{S}$ 中都与 μ 相邻，而且 μ 必须是 z 的分配骨架，因此 $z \in$ 是 origV(G_ν) ∩ origV(G_μ)。根据 ν 的邻接关系，它要么是 origV(G_ν)∩origV(G_μ) = origV′(G_ν)∩origV(G_α)，要么是 origV(G_ν) ∩origV(G_μ) = origV′(G_ν) ∩origV(G_β)，因为该操作不会修改 ν，且 S 和 S′ 都满足条件 5（原始虚拟边一致性）和条件 8（orig-stars）。这立即与 $\{\alpha, \beta\}$ 的骨架相矛盾：它与 ν 相邻，却不是 z 的分配骨架。

最后，映射 origE 保持不变，而 origV 唯一的变化是加入一些映射到已有顶点的重复顶点。由于输入和输出实例都满足条件 4（原始边真实性），这不会影响表示图$G_{\mathcal{S}'}$。

现在考虑 `Integrate` 返回的扩展骨架分解 $\mathcal{S}'$。合并后的骨架是双连通的，因为实际上我们是用一个连通子图替换单个顶点，因此满足条件 1（双连通）。该操作实际上收缩并删除了$T_\mathcal{S}$中的一条边，这不会影响 $T_{\mathcal{S}'}$ 是树这一性质，因此满足条件 2（树）。请注意，输入实例满足条件 2（树），还确保 v_α 和 v_β 属于两个不同的骨架。由于输入实例满足条件 5（原始虚拟边一致性），在两个相邻骨架中，origV 映射到 $G_\mathcal{S}$ 的同一顶点的那些顶点，恰好是与匹配的 v_α 和 v_β相邻的顶点。因此，origV$|_{V_\alpha}$ 仍然是单射，满足条件 3（原顶点单射）。由于我们没有修改任何真实边或虚边，映射 origV′，origE′ 和 twinE′ 显然仍满足条件 4（原始边真实性）和条件 5（原始虚拟边一致性）。最后，收缩一条树边不会使条件 6（子图）中的任何子图变得不连通，因此该条件仍然满足。条件 7（星形图）和条件 8（orig-stars）也不受影响，因为我们只是从 twinV 中删除了一项。

同样，origE 未发生任何变化，而条件 8（原顶点星形）确保 origV 将每一对合并顶点映射到 $G_\mathcal{S}$ 的同一顶点。因此，表示图 $G_{\mathcal{S}'}$ 保持不变。◀

此外，由于 `Integrate` 是 `IsolateVertex` 的逆操作且没有前置条件，因此可以随时撤销 `IsolateVertex` 所做的任何更改，从而得到一个（非扩展）骨架分解，进而可能得到表示图的 SPQR树。

▶ 注 5. 对扩展骨架分解$\mathcal{S} = (\mathcal{G}, \text{origV},\text{origE}, \text{twinE}, \text{twinV})$穷尽地应用 `Integrate`，会得到一个扩展骨架分解$\mathcal{S}' = (\mathcal{G}',\text{origV}',\text{origE}',\text{twinE}',\text{twinV}')$，其中 twinV′ = ∅。因此，$\mathcal{S}'$ 等价于一个（非扩展）骨架分解$\mathcal{S}' = (\mathcal{G}',\text{origV}', \text{origE}', \text{twinE}')$。

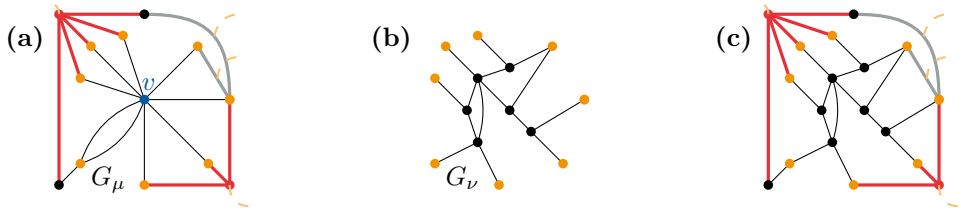

图3 将图4b的 SPQR树中的骨架顶点 v 扩展为图 G_ν。（**a**）图4b中的单个分配骨架 G_μ 及其单个分配顶点 v（属于 u）。v 的邻点以橙色标出。（**b**）插入的图 G_ν，其中标记顶点以橙色标出。注意，将所有标记顶点收缩为一个顶点后，该图是双连通的。（**c**）先应用 `InsertGraph`($\mathcal{S}$ u, G_ν, ϕ)，再对生成的虚拟顶点 v和 v'应用 `Integrate` 后所得的结果。

5 扩展骨架分解中的节点扩展

现在我们介绍第一个动态操作，它允许我们通过将单个顶点 u 扩展为任意连通图 G_ν，实际改变表示图。具体做法是通过双射 ϕ，将 G_ν 中的 $|N(u)|$ 个标记顶点与 u 的邻点对应起来，然后移除 u 及其关联边。我们使用上一节中的“占用星”来模拟这些顶点的识别，从而可以将实际插入推迟到应用 `Integrate`时进行。我们需要确保插入的图在连通性方面对周围图作出的“保证”与它所替代的顶点相同，也就是说，u 的所有邻点（即 G_ν中的所有标记顶点）都需要在 G_ν 中通过路径两两连通，且这些路径不能经过 u 的任何其他邻点（即任何其他标记顶点）。如果没有这一要求，单个顶点也可能被拆分成两个彼此不相邻的部分，这很容易使一个三连通分量分裂。因此，我们要求将所有标记顶点收缩为单个顶点后，G_ν 是双连通的。请注意，这也确保了可以通过收缩插入图的顶点来恢复旧图。为简单起见，我们要求表示图中的顶点 u 只有一个分配顶点$v \in G_\mu$，且 $\text{origV}^{-1}(u) = \{v\}$，这样我们只需更改骨架分解中的一个分配骨架$G_\mu$。我们稍后会清楚地说明，这一条件很容易满足。

形式化地，设 $u \in G_S$ 是一个仅有单个分配顶点 $v \in G_\mu$（因此也仅有单个分配骨架 G_μ）的顶点。设 G_ν 是一个任意的新图，其中包含 $|N(u)|$ 个标记顶点，并且标记顶点与 G_μ中 v 的邻点之间存在双射 ϕ。我们要求将所有标记顶点收缩为一个节点后，G_ν 是双连通的。操作 `InsertGraph`(S u, G_ν, ϕ) 生成扩展骨架分解 S′ = (G′, origV′, origE′, t winE′, twinV′)，具体如下，另见图3。我们将 G_ν 视为骨架，并将其添加到 G′中。对于 G_ν中的每个标记顶点 x，令 origV′(x) =origV($\phi(x)$)。对于 G_ν中的所有其他顶点和边，令 origV′ 和 origE′指向新顶点和新边，以便在 $G_{S'}$中形成 G_ν 的副本。将 G_ν 中的每个标记顶点通过占用边连接到一个新的虚拟顶点 $v' \in G_\nu$。我们还将 v 转换为虚拟顶点，并将其关联边转换为占用边，同时移除平行边。最后，令 twinV′(v) = v'，且 twinV′(v') = v。

▶ **引理 6。** 在扩展骨架分解 S = (G,origV,origE,twinE,twinV) 上应用 `InsertGraph`(S u, G_ν, ϕ)，得到扩展骨架分解 S' = (G',origV′, origE′, twinE′, twinV′)，其中 $G_{S'}$ 与 $G_S[u \to_\phi G_\nu]$同构。

证明。由于 G_μ 的结构保持不变，且根据前置条件，所得 G_ν 是双连通的，因此条件 1（双连通）仍然成立。对于 $T_\mathcal{S}$，我们将一个度为 1 的节点 ν 连接到一个已有节点 μ上，因此条件 2（树）也仍然成立。由于 G_ν 的所有顶点中，除 $N(v')$ 中的顶点外，其余顶点都由 origV′和 origV′($N(v')$) $=$分配了新的唯一副本，因此新G_ν满足条件 3（原顶点单射）。由于我们在更新 origV 的同时也更新了 origE，且 G_ν 不含虚边，因此条件 4（原始边真实性）和条件 5（原始虚拟边一致性）仍然成立。由于 ν 是 $T_\mathcal{S}$ 的叶节点，且 μ 是其唯一邻居，origV′($N(v')$) $\subset$ origV(V_μ)，并且 G_ν 是 $G_\nu \setminus N(v')$ 中所有顶点唯一对应的分配骨架，因此条件 6（子图）仍然成立。根据构造，条件 7（星形图）和条件 8（orig-stars）均成立。最后，根据构造，映射 origE′ 和 origV′ 已更新，能够在$G_{\mathcal{S}'}$中正确再现 G_ν 的结构。◀

不过，单独来看，此操作用处不大，因为图顶点很少只有一个分配骨架。此外，我们的目标是动态维护 SPQR 树，而单独执行此操作在大多数情况下不会得到 SPQR 树。为此，我们引入完整过程 $\texttt{InsertGraph}_{\texttt{SPQR}}(\mathcal{S}, u, G_\nu, \phi)$，该过程可应用于任意图顶点 u，并且给定一个 SPQR 树 $\mathcal{S}$ 后，可得出 $G_\mathcal{S}[u \to_\phi G_\nu]$的 SPQR 树。该过程包括三个准备步骤、插入 G_ν，以及另外两个清理步骤：

1. 对 u 中顶点数大于三的每个多边形分配骨架，我们都应用 `SplitSeparationPair`，并将 u 的分配顶点的邻点作为分离对。2. 对于 u的每个刚性分配骨架，我们通过应用 `IsolateVertex`(S v)，将其中的分配顶点 v（属于 u）移至其自身的骨架。3. 对 u 中在 $T_\mathcal{S}$ 内相邻的任意一对分配骨架，我们穷尽地应用 `JoinSeparationPair`。由于条件 6（子图），这会得到一个单独的分量 G_μ，它是 u 唯一的分配骨架，其中包含 u的唯一分配顶点 v。此外，G_μ 的大小与 deg(u) 呈线性关系。4. 我们应用 `InsertGraph`，将 G_ν 作为骨架插入；随后对插入所引入的虚拟顶点 $\{v, v'\}$ 应用 `Integrate`，从而将 G_ν 整合到 G_μ 中。5. 对于 G_μ 中不涉及虚拟顶点的所有分离对，我们应用 `SplitSeparationPair`。这些分离对可以在线性时间内找到，例如，暂时复制所有虚拟顶点及其关联边，然后计算 SPQR 树。[2]6. 最后，我们穷尽地应用 `Integrate`，并且对任意两个相邻的多边形以及任意两个相邻的键应用 `JoinSeparationPair`，以得到更新后图的 SPQR 树。

该过程正确性的基本思路是：在步骤 5 中，按照新插入分量的 SPQR树对其进行拆分，所得的双连通分量各自要么是多边形，要么是键，要么是“几乎”三连通。后者（且只有后者）可能仍包含虚拟顶点，并且它们所有尚未在步骤 5 中拆分的其余分离对都包含其中一个虚拟顶点。此外，由于仍可能存在两个均为多边形或均为键的相邻骨架，因此该实例还不是SPQR树。步骤 6 解决了这两个问题：显然，`JoinSeparationPair` 应用可以修复相邻骨架。为了说明 `Integrate` 应用会移除虚拟顶点，使剩余分量成为三连通，我们需要以下引理。

[2] 注意，在定理 10 的证明中替代虚拟顶点的轮也确保了这一点。

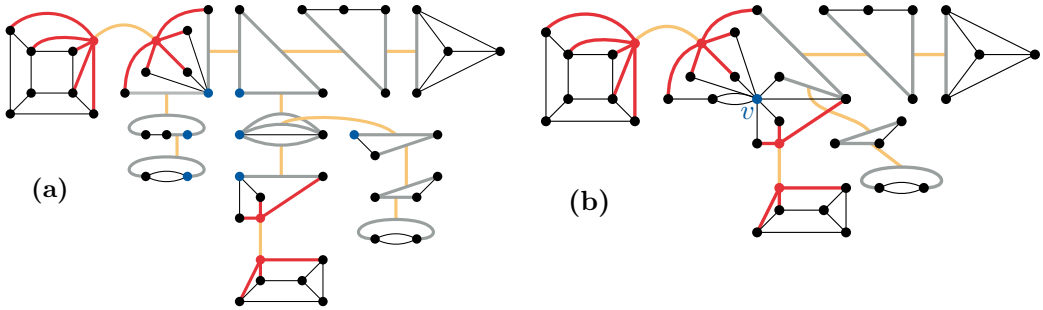

图 4 将 $\texttt{InsertGraph}_{\texttt{SPQR}}$ 的预处理步骤应用于图 1b 的 SPQR树。**(a)** 第 2 步之后的状态，此时 u 的所有分配骨架均已拆分。**(b)** 第 3 步之后的状态，此时 u 的所有分配骨架均已合并为一个。

▶**引理 7.** 设 G_α 是一个三连通骨架，其中包含一个虚拟顶点 v_α，该顶点与双连通骨架 G_β 的一个虚拟顶点 v_β 匹配。此外，设 $P \subseteq \binom{V(G_\beta)}{2}$ 为 G_β 中所有分离对的集合。应用 $\texttt{Integrate}(S, (v_\alpha, v_\beta))$ 会产生一个双连通骨架 G_μ，其分离对为 $P' = \{\{u, v\} \in P \mid v_\beta \notin \{u, v\}\}$。

证明。根据顶点分别来自 G_α、G_β 或两者，将 G_μ 的顶点划分为集合 A、B 和 N。因此，集合 N 包含 v_α 的邻点，这些邻点与 v_β 的邻点相对应。现在我们将通过导出矛盾来证明，除 P' 中的分离对外，G_μ 不含任何分离对。因此，考虑不属于 P' 的分离对 u、$v \in G_\mu$。首先，考虑 u、$v \in A \cup N$ 的情形。注意，在这种情况下，移除 u、v 后，B 仍然连通。因此，我们可以将 B 的所有顶点收缩为一个顶点，重新得到 G_α，并看到 u、v 是 G_α 中的分离对。这与 G_α 是三连通的前提矛盾。现在考虑 u、$v \in B \cup N$ 的情形。与上述情况类似，我们发现 u、v 是 G_β 中不包含 v_β 的分离对，这与 $\{u, v\} \notin P'$ 矛盾。最后，考虑剩下的情形，不失一般性地设 $u \in A$、$v \in B$。由于 $\{u, v\}$ 是分离对，u 有两个邻点 x、y，它们分别位于 $G_\mu - \{u, v\}$ 的不同连通分量中，因此也分别位于与 $G_\alpha - \{u, v_\alpha\}$ 同构的 $(G_\mu - \{u, v\}) - B$ 的不同分量中。这再次与 G_α 是三连通的前提矛盾。◀

▶ **定理 8.** 将 $\texttt{InsertGraph}_{\texttt{SPQR}}(S, u, G_\nu, \phi)$ 应用于 *SPQR*树 S，可在 $O(|G_\nu|)$ 时间内得到一个 *SPQR* 树 S'，且 $G_{S'}$ 与 $G_S[u \to_\phi G_\nu]$ 同构。

证明。 由于所有已执行的操作都保持扩展骨架分解有效，因此最终的扩展骨架分解 $\mathcal{S}'$ 也有效。注意，预处理步骤 1–3 的唯一目的是确保 InsertGraph 的前提条件得到满足，并且受影响的分量不会过大。请注意，步骤 2 中拆分的所有刚性分量在结构上均未改变：边只是改变了类型，但图，尤其是其三连通性，保持不变。步骤 4 执行实际插入，并根据引理 6 得到所需的表示图。因此，接下来只需证明清理步骤会将得到的扩展骨架分解转化为 SPQR树。在步骤 6 中穷尽地应用 Integrate 可确保扩展骨架分解等价于一个非扩展骨架分解（注 5）。回顾一下，若所有骨架均为多边形、键或三连通骨架，且任意两个相邻骨架都不会同为多边形或同为键（定义 3），则非扩展骨架分解就是一棵 SPQR树。步骤 6 确保了后半部分条件成立，因为用 JoinSeparationPair 连接两个多边形（或两个键）会得到一个更大的多边形（或键）。在执行

步骤 6 之前，所有并非 u 的分配骨架的骨架都未被修改，因此其结构已经符合要求，即它们均为多边形、键或三连通骨架。此外，不含虚拟顶点的 u 分配骨架也具有合适的结构，因为它们是根据步骤 5 中的 SPQR树进行拆分的。接下来需要证明其余骨架，也就是步骤 6 中应用 `Integrate` 后得到的骨架，均为三连通骨架。注意，在这些骨架中，步骤 5 确保每个分离对至少包含一个虚拟顶点；否则，计算出的 SPQR树就会进一步拆分该骨架。还要注意，对于每个这样的虚拟顶点，其匹配伙伴顶点都属于一个在结构上未被修改、且在步骤 2 中拆分的三连通骨架。引理 7 表明，若两侧之一为三连通，则应用 `Integrate` 不会引入新的分离对，同时会移除两个虚拟顶点。因此，我们可以穷尽地应用 `Integrate`，从而移除所有虚拟顶点，也就移除了所有分离对，得到三连通分量。这说明 SPQR树的判据均已满足；并且，由于 `InsertGraph` 将表示图中的 u 扩展为 G_ν，现在我们得到了 $G_{\mathcal{S}}[u \to_\phi G_\nu]$的唯一 SPQR树。

注意，我们使用的所有操作，其运行时间均与应用这些操作的顶点的度数呈线性关系。对于输入到 `SplitSeparationPair` 的桥的二划分，只需通过每个桥中与分离对关联的边来描述该桥，而无需显式枚举桥中的所有内部顶点。因此，在步骤 1 和 2 中应用 `SplitSepara-tionPair` 和 `IsolateVertex` 时，每条与 u 关联的边至多被访问一次，所需时间为 $O(\deg(u))$。此外，它们生成的骨架大小与 u 中各自分配顶点的度数呈线性关系。由于 u 的分配骨架子树大小至多为 $\deg(u)$，步骤 3 的 `JoinSeparationPair` 应用所需时间也至多为 $O(\deg(u))$。由此还可得，u 最终生成的单个分配骨架大小为 $O(\deg(u))$。步骤 4 中应用 `InsertGraph` 和 `Integrate` 的时间与已识别邻居的数量呈线性关系，即为 $O(\deg(u))$。在步骤 5 中生成插入图的 SPQR 树（其中所有虚拟顶点都被轮替换），所需时间与插入图 $[30, 33]$，的大小呈线性关系，即为 $O(|G_\nu|)$。根据该 SPQR树识别出的所有分离对应用 `SplitSeparationPair`，总共也可在 $O(|G_\nu|)$ 时间内完成。注意，在步骤 4 之前已存在的骨架与步骤 4 和 5 中新建或修改的骨架之间，边数至多为 $\deg(u)$；而这些是现在可能连接两个多边形或两个键的唯一边。由于这些树边的一个端点位于 u 的单个分配骨架中，步骤 6 中应用 `Integrate` 和 `JoinSeparationPair` 的总时间为 $O(\deg(u))$。此外，它们会移除所有相邻多边形对和所有相邻键对。这表明，除步骤 5 需要 $O(|G_\nu|)$ 时间外，所有步骤均需 $O(\deg(u))$ 时间。由于插入图中每个 u 邻居至少对应一个顶点，总运行时间为 $O(|G_\nu|)$。◀

▶ **推论9**。 设 S_1,S_2 是两个 *SPQR*树，分别带有顶点 $u_1 \in G_{\mathcal{S}_1}$、 $u_2 \in G_{\mathcal{S}_2}$，并设 ϕ 是关联于 u_1 的边与关联于 u_2 的边之间的一个双射。操作 $\text{Merge}_{\text{SPQR}}$（$S_1$,$S_2$、$u_1$、$u_2$、$\phi$） 生成图 $G_{\mathcal{S}_1}[u_1 \to_\phi G_{\mathcal{S}_2} - u_2]$ 的 *SPQR*树，即将两个图的关联于 u_1、u_2 的边按照 ϕ 进行识别，并移除 u_1、u_2 后所得的并图，所需时间为 O（$\deg(u_1)$ $= O$（$\deg(u_2)$ 。

证明。 操作 $\text{Merge}_{\text{SPQR}}$ 与更一般的 $\text{InsertGraph}_{\text{SPQR}}$类似，不过运行时间更优，因为我们已经知道待插入图的 SPQR树。我们应用预处理步骤 1–3，确保 u_1 和 u_2 分别只有一个分配顶点 v_1 和 v_2。为了正确处理平行边，我们细分所有与 u_1关联的边，u_2（因此，与 v_1 关联的相应实边也会被细分，v_2）并

然后识别由 ϕ 匹配的每对边的细分顶点。删除顶点v_1 和 v_2 并抑制细分顶点（即移除这些顶点，并识别每对关联边），即可得到大小为 $O(\deg(u_1)) = O(\deg(u_2))$ 的骨架 G_μ。最后，我们对 G_μ 应用清理步骤 5 和 6，得到最终的 SPQR树。同样，由于 u 的分配骨架中每个虚拟顶点的配对顶点都属于一个三连通骨架，因此在步骤 6 中反复应用 `Integrate` 会得到三连通骨架。如前所述，预处理步骤和清理步骤的运行时间与受影响顶点的度呈线性关系，因此本例的总运行时间为 $O(\deg(u_1)) = O(\deg(u_2))$。◀

5.1 维护平面性和平面顶点旋转次序

请注意，使用另一个平面图通过 $\texttt{Insert-Graph}_{\texttt{SPQR}}$ 扩展平面图的一个顶点（或根据推论9合并两个平面图的 SPQR 树），实际上可能会得到一个非平面图。例如，这是因为两个图的刚性分量可能要求被替换顶点的邻点采用不兼容的次序。本节旨在高效检测这种情况，即平面图变为非平面图的情况。要检查一般图是否为平面图，只需检查其 SPQR 树中的刚性分量是否具有平面性；每个刚性分量恰好有两种平面嵌入，其中一种是另一种的反向嵌入 [19]。因此，如果图通过应用 $\texttt{InsertGraph}_{\texttt{SPQR}}$ 变为非平面图，这一点会从被替换顶点的三连通分配骨架中显现出来。为了能够在实例变为非平面时立即报告，我们需要为每个三连通骨架中的每个顶点维护一种旋转次序，即所有关联边的循环次序。请注意，我们不跟踪次序的方向，也就是说，只存储忽略反向的次序。稍后将讨论如何以少量额外开销维护精确次序。

▶**定理 10**。 *SPQR* 树支持以下操作：$\texttt{InsertGraph}_{\texttt{SPQR}}(S, u, G_\nu, \phi)$：在 $O(|G_\nu|)$ 时间内扩张单个顶点 u，$\texttt{Merge}_{\texttt{SPQR}}(S_1, S_2, u_1, u_2, \phi)$：在 $O(\deg(u_1))$ 时间内合并两棵 *SPQR* 树，`IsPlanar`：在 $O(1)$ 时间内查询表示图是否为平面图，以及$\texttt{Rotation}(u)$：在 $O(1)$ 时间内，查询平面三连通骨架中顶点 u 的两种可能旋转次序之一。

证明。 请注意，布尔标志 `IsPlanar` 以及 `Rotation` 信息可以在线性时间内计算得出；在创建新的 SPQR 树时如此，而扩展顶点或合并两棵 SPQR 树都不会使非平面图变为平面图。我们对定理 8 和推论9中描述的操作 $\texttt{InsertGraph}_{\texttt{SPQR}}$ 和 $\texttt{Merge}_{\texttt{SPQR}}$ 作出以下修改，以维护这项新信息。在步骤 2 中拆分三连通分量后，我们现在引入额外结构，确保嵌入在拆分的两侧都得到维护。拆分出的顶点 v 周围生成的占用边（以及其副本v'周围的占用边）会被细分，并按照 $\texttt{Rotation}(v)$进行循环连接。因此，我们不再生成“星形图”，而是生成占用“轮”，以编码三连通分量嵌入中的边的次序。在步骤 5 中生成修改后子图的 SPQR 树时，该子图现在包含占用轮而非仅有星形图；我们还会为其所有三连通骨架生成平面嵌入。如果至少有一个骨架无法找到平面嵌入，我们就将 `IsPlanar` 设为 false，以报告所得实例是非平面图。否则，在执行完 SPQR 树所指示的所有拆分后，我们通过为所有新的刚性分量生成嵌入来设置 `Rotation`。请注意，对于所有含有虚拟顶点的骨架，生成的嵌入都将与相邻三连通分量的嵌入兼容，也就是说，由于插入了轮，每个虚拟顶点的旋转都将与其匹配伙伴顶点的旋转一致。最后，在步骤 6 应用 `Integrate` 之前，我们将每个

占用轮收缩为单个顶点，以重新得到占用星形图。创建和收缩轮所增加的开销至多与扩展顶点的度成线性关系，而为刚性分量生成嵌入所需的时间与刚性分量的大小成线性关系。因此，这不会影响这两种操作的渐近运行时间。◀

▶推论 *11*。定理 *10* 中的数据结构可以进行调整，从而为刚性分量中的每个顶点提供带有匹配方向的精确旋转次序。此外，它还可以支持查询两个顶点 v_1、v_2 是否通过 *3Paths*（v_1、v_2）条不同的顶点不相交路径相连，查询时间为 $O((deg(v_1)+deg(v_2))\cdot\alpha(n))$。这些调整使 $\texttt{InsertGraph}_{\texttt{SPQR}}$ 的运行时间变为 $O(deg(u)\cdot\alpha(n)+|G_\nu|)$，使 $\texttt{Merge}_{\texttt{SPQR}}$ 的运行时间变为 $O(deg(u_1)\cdot\alpha(n))$，并使 *Rotation*（$u$）的运行时间变为 $O(\alpha(n))$。

证明。可以使用并查集跟踪每个顶点所属的刚性分量，并在两个刚性分量通过 Integrate 合并时，同步翻转一个刚性分量内所有顶点的旋转次序，以此维护 Rotation 的精确旋转信息。我们为三连通分量中的每个顶点创建一个并查集集合，并对同一刚性分量中的所有顶点应用 Union。除了指向并查集结构中代表元的指针外，我们还存储一个布尔标志，用于指示当前顶点的旋转信息相对于其直接代表元的旋转是否反转。为了查找一个 Rotation 是否需要翻转，我们使用异或沿着通往顶点实际代表元的路径累积所有标志。由于 Rotation（u）依赖于 Find 操作，其摊销运行时间为 $O(\alpha(n))$。使用 Integrate 合并两个刚性分量时，我们还对它们各自的代表元执行一次 Union（首先需要对它们执行 Find），使 Integrate（S,（v_α，v_β））在 $O(\deg(v_\alpha)+\alpha(n))$ 时间内运行。我们还会比较被替换顶点的 Rotation；如果二者不匹配，则翻转与最终未成为代表元的顶点一起存储的标志。总之，这使 $\texttt{InsertGraph}_{\texttt{SPQR}}$ 的运行时间变为 $O(\deg(u)\cdot\alpha(n)+|G_\nu|)$，因为最多可能有 $\deg(u)$ 个分裂的刚性分量。此外，$\texttt{Merge}_{\texttt{SPQR}}$ 现在在 $O(\deg(u_1)\cdot\alpha(n))$ 时间内运行。

维护骨架顶点所属刚性组件的信息，也可用于回答任意两个顶点是否由三条不相交路径连接的问题。当且仅当它们属于同一个刚性组件、作为同一个键的极点出现，或在多边形中由虚边连接时，答案才是肯定的。我们可以通过枚举两个顶点的所有分配骨架来检查这一点，所需时间与它们的度成线性关系。由于查找每个骨架可能需要一次 Find 调用，因此总运行时间为 $O((\deg(v_1)+\deg(v_2))\cdot\alpha(n))$。◀

6 同步平面性的应用

本节将介绍簇平面性和同步平面性问题的发展历史及更多细节，并概述 Bläsius 等人提出的、用于解决这两个问题的算法。此外，我们还将说明，本文工作以及此前关于动态 SPQR 树的研究如何应用于这两个问题。

6.1 背景与讨论

Lengauer [34] 于1989年首次以另一种名称讨论了簇平面性，因此 Feng 等人后来于 [23] 1995年独立地重新发现了这一问题。两者都为每个簇所诱导的子图均连通的情形提出了多项式时间算法。相比之下，对于簇不连通的一般问题是否存在一种

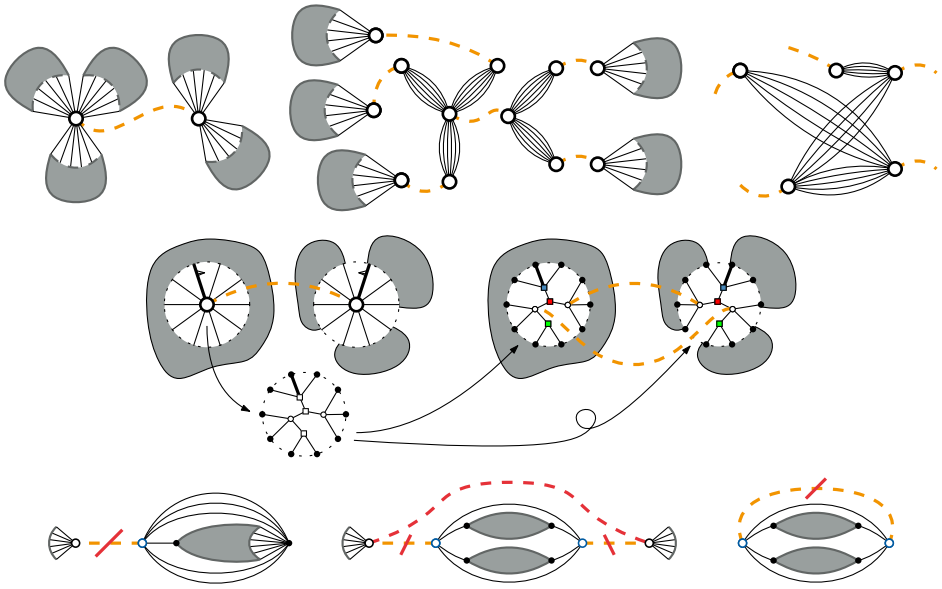

图5 Bläsius 等人用于解决同步平面性的三种操作示意图。匹配的顶点以较大的圆盘表示，匹配关系用橙色虚线表示。**上：** 两个彼此匹配的割点（左）、将其关联块封装后的结果（中）以及连接两个割点后得到的二分图（右）。**中：** 一个具有非平凡嵌入树的匹配非割点（左），该嵌入树被传播，用以替换该顶点及其配对元素（右）。**下：** 具有平凡嵌入树（蓝色）的匹配顶点的三种不同情形，以及如何移除或替换它们的管道（红色）。

高效解法，这个问题悬而未决长达30年。在此期间，人们为许多特殊情形找到了多项式时间算法 [2, 15, 25, 29]，之后 Fulek 和 Tóth [26] 于2019年提出了一种 $O((n+d)^8)$解法。不久之后，Bläsius 等人 [8] 提出了一种运行时间为$O((n+d)^2)$的解法，该解法还揭示了解决簇平面性所需的主要概念。该解法通过线性时间归约到同步平面性问题来实现；Bläsius 等人为该问题提出了一种二次时间算法。我们改进了后者算法的运行时间。由于同步平面性可以作为簇平面性之外若干其他受约束平面性问题的建模工具 [8], ，这也缩短了所有可通过线性时间归约到同步平面性来解决的受约束平面性问题的求解时间；见表1。

在簇平面性问题中，嵌入必须遵循一个层叠族的簇 [9,34], ；也就是说，每个顶点都属于某个（按层级嵌套的）簇，而边只有在连接簇内顶点与簇外顶点时，才可以穿过簇边界。在同步平面性问题中，给定图中部分顶点上的一个匹配，我们要寻找一种嵌入，使匹配顶点的旋转次序按照给定的双射 [8]对齐。匹配两个顶点所施加的同步约束也称为管道。从前一个问题到后一个问题的归约采用了簇平面性的 CD树表示 [9], ，其中每个簇都表示为一个独立的骨架，相邻簇则合并为单个“虚拟顶点”。现在，一个簇经由虚拟顶点“离开”的边的次序，需要与这些边经由相应虚拟顶点“进入”相邻簇的次序对齐（另见 [8,图 6]）。

16 在节点扩展下维护三连通分量

求解同步平面性的算法通过每一步移除一条任意管道来运行，并根据匹配顶点周围的图选择三种操作之一，见图5。

`EncapsulateAndJoin` 如果管道的两个顶点都是割点，则分别复制它们所属的分量，并将每个关联块收缩为单个顶点，从而得到中心相匹配的星形图；星形图的中心通过多条平行边与射线顶点相连。原来的割点被拆分，使每个关联块都获得自己的副本，并将这些副本与代表相应收缩块的顶点同步。现在可以通过“连接”两个星形图来移除割点，也就是按照匹配关系所附带的双射识别它们的关联边。

`PropagatePQ` 如果其中一个顶点不是割点，且其嵌入树并非只由单个 P节点构成，则分别在两个匹配顶点的位置插入（“传播”）该嵌入树的两个副本。通过匹配对应的顶点，使嵌入树的内部节点保持同步。

`SimplifyMatching` 在剩余情况下，其中一个顶点不是割点，但其嵌入树是平凡的，也就是说，它在 SPQR树中只出现在一个平行骨架中，而不出现在任何刚性骨架中。如果该顶点（或者更确切地说，完全决定其旋转次序的平行骨架）可以适应任意旋转次序，我们就可以直接移除管道。唯一的例外是平行骨架的另一个极点也参与了匹配；此时我们可以跨越该平行骨架“短接”匹配关系。

总而言之，每个操作都会从匹配中移除一条管道，同时可能引入一些包含度更小的顶点的新管道。利用势函数可以证明，移除操作取得的进展始终足以抵消新引入管道带来的开销，而且移除所有管道所需的操作次数受所有已匹配顶点的总度数限制。此外，得到的不含管道的实例可以在线性时间内求解。如果所依赖的嵌入树已可用，这三种操作的运行时间均与未匹配顶点的度数成线性关系。本文的贡献在于高效地提供这些嵌入树；若采用朴素方法，则每一步都需要处理整个连通分量。借助Holm和Rotenberg提出的全动态SPQR树，可以用 $O(\Delta \cdot \log^3 n)$ 的多对数开销实现这一点，从而使总运行时间达到 $O(m \cdot \Delta \cdot \log^3 n)$。利用本文提出的节点扩展，我们可以改进运行时间：不再对线性数量的每个操作都花费与输入实例大小成线性关系的时间 ($O(m)$)，而是将每次操作的耗时降至与最大度成线性关系 ($O(\Delta)$)。从簇平面性归约得到的实例大小为 $O(n+d)$，其中已匹配顶点的总度数为 $O(d)$，对应于边跨越簇边界的总次数。请注意，虽然这意味着只需执行$O(d)$次操作即可得到归约后的实例，但在平面图中，边与簇边界之间的交叉次数可能达到顶点数的平方级。我们还指出，尽管与Holm和Rotenberg的方法相比，改进幅度仅为多对数级，但我们的数据结构在概念上更简单，因此也更有望在实践中提升性能。

6.2 使用节点扩展求解同步平面性

我们展示了如何利用扩展骨架分解及其动态操作 $\texttt{InsertGraph}_{\text{SPQR}}$，将 Bläsius 等人的同步平面性算法的运行时间从 $O(m^2)$ 改进到 $O(m \cdot \Delta)$，其中 Δ 是最大管道度。正如

上一节已经说明，算法的大部分运行时间都花在计算所谓的嵌入树上。这些嵌入树描述了平面图中单个顶点的所有可能旋转次序，并用于在同步旋转的顶点之间传递嵌入限制。得到嵌入树后，至多执行 $O(m)$ 次操作，每次操作的运行时间与其作用对象（管道或顶点）的度成线性关系，也就是 $O(\Delta)$ [8]。因此，通过维护生成这些嵌入树所依据的 SPQR 树来高效生成嵌入树，是我们为加速同步平面性算法所作的主要贡献。

对于双连通图 G 中的顶点 v，嵌入树 $\mathcal{T}_v$ 描述了在 G [12] 的所有平面嵌入中，与 v 关联的边可能具有的循环次序或旋转次序 。$\mathcal{T}_v$ 的叶节点是与 v 关联的边，而其内部节点分为两类：*Q*节点 定义了其关联树边的固定旋转次序（允许反转），而*P*节点 允许任意旋转；见图1d。为了生成嵌入树，我们采用 Bläsius 和 Rutter [10,第 2 节5] 中描述的 SPQR树与嵌入树之间关系的观察：v 的嵌入树中的 P 节点和 Q节点，分别与 G 的 SPQR树中 v 的键和三连通分配骨架一一对应。

▶ **引理 12**。 设 *S* 是一个 *SPQR*树，其平面表示图为 G_S。求顶点 $v \in G_S$ 的嵌入树所需时间为 $O(\deg(v))$。

证明。我们使用定理 10 中的旋转信息，并且为 G_S 中的每个顶点维护一个（任意选取的）分配顶点。要计算从 v 的分配顶点 u 开始的顶点 v 的嵌入树，我们将使用 u 的一条关联边上的 twinE 遍历 SPQR树，然后将所得边的一个端点作为 v 的下一个分配顶点。如果 u 的度为 2，那么它属于一个多边形骨架，而该骨架不会在嵌入树中产生节点。因此，我们转向它相邻的分配骨架，并以同样的方式跳过遇到的其他多边形骨架。如果 u 的度至少为 3，我们检查任意两条关联边：如果它们通向同一顶点，则 u 是一个键的极点，我们生成一个 P节点。否则，它属于一个三连通分量，我们生成一个 Q节点。接下来，我们遍历 u 的关联边；对于三连通分量，按 u 的旋转给定的顺序遍历。对于每条实边，我们将相应的叶节点连接到新生成的节点。与该叶节点对应的图边可通过 origE 获得。对于每条虚边，我们递归处理相应的相邻骨架，并将递归生成的节点连接到当前节点。由于 u 至多属于 deg(u) 个骨架，这些骨架构成 T_S 的一棵子树，而 u 的分配顶点总共只有 O(deg(u)) 条虚边和实边与之关联，因此，该过程可在线性于其度的时间内生成 u 的嵌入树。◀

现在，我们可以使用该数据结构来缩短求解同步平面性的运行时间：预先生成一棵 SPQR树，在执行所有操作的过程中维护它，并从 SPQR树中导出所需的嵌入树。

▶ **定理13.** 同步平面性可在 $O(m \cdot \Delta)$时间内求解，其中 m 为边数， Δ 为管道的最大度。

证明。 该算法通过不断拆分表示同步约束的管道，直至其小到可以平凡处理。它根据同步顶点对周围的图结构，穷尽地应用以下三种操作 `EncapsulateAndJoin`、 `PropagatePQ` 和 `SimplifyMatching` 。正如 Bläsius 等人所指出的，如果已知所用的嵌入

树，则这些操作的运行时间与其所作用的管道的度呈线性关系，并且 $O(m)$ 次操作足以解决给定实例 [8]。我们的改动是为每个双连通分量维护一棵 SPQR树，然后根据需要使用引理 12 在线性时间内生成所需的嵌入树。有关后文修改的同步平面性操作的更多背景，请参见第6.1节。

如果同步顶点v 周围的图允许 v任意旋转，也就是说， v 的嵌入树是平凡的，那么就可以执行操作 `SimplifyMatching` 。在这种情况下，无需修改图结构即可移除管道。因此，我们现在可以轻松检查此操作的前置条件，而无需对SPQR树进行任何更改。

`PropagatePQ` 会取一个同步顶点 v 的非平凡嵌入树，并分别在 v 及其配对元素的位置插入该树的副本。插入树的内部顶点上的同步约束用于确保这些树以相同的方式嵌入。我们还使用 $\texttt{InsertGraph}_{\texttt{SPQR}}$ 将嵌入树插入相应的SPQR树，并用轮表示Q节点。在传播到割点时，我们还需要检查两个或更多关联块是否合并。我们对关联块构造等价类：若两个块满足以下任一条件，则它们属于同一类：1）由各自边诱导的两个子树至少共享两个节点；2）两个诱导子树都包含一个在这两个子树中的度均至少为2的C节点。同一等价类中的块最终会归入同一个双连通分量，具体如下：我们构造由该等价类中所有边诱导的子树，并为类中的每个块各添加一个额外节点，将所有叶节点连接到相应块的节点（这些叶节点所代表的边通向该节点）。我们计算该双连通图的SPQR树，然后应用推论9，将各个块的SPQR树合并到其中。由于 $\texttt{InsertGraph}_{\texttt{SPQR}}$ （以及类似的所有 $\texttt{Merge}_{\texttt{SPQR}}$ 操作）的运行时间与所插入PQ树的大小成线性关系，而该大小受其表示的顶点的度限制，因此这不会对该操作的运行时间产生负面影响。

操作 `EncapsulateAndJoin` 会生成一个新的二分分量，用以表示两个同步割点所关联的块的边如何彼此匹配。该分量的大小与同步顶点的度成线性关系。因此，我们可以在线性时间内重新计算所生成分量的 SPQR树，这也不会对运行时间产生负面影响。

此外，由于现在生成嵌入树时不再需要遍历整个连通分量，我们也不再需要确保这些分量不会变得太大。因此，我们还可以直接使用推论9收缩两个不同双连通分量之间的管道，而不必使用 `PropagatePQ`插入 PQ树。这可能会改善实际运行时间，因为 `PropagatePQ` 可能需要进一步执行操作来清理生成的管道，而直接收缩会完全移除管道，且不会生成新的管道。◀

▶ **推论14.** 簇平面性可在 $O(n + d \cdot \Delta)$时间内求解，其中 d是簇边界与边之间的交叉总数，Δ 是单个簇边界上的最大边交叉数。

证明。 注意，对于不含平行边的图，要成为平面图，其边数必须与顶点数呈线性关系。我们采用 Bläsius 等人所述的、从 簇平面性 到 同步平面性 的归约 [8]。忽略 CD 树生成的平行边后，我们可以在总计 $O(n)$ 时间内，为所得实例的每个分量生成一棵 SPQR树。该实例为每个簇边界包含一条管道，其中管道的度对应于穿过相应簇边界的边数。因此，Bläsius 等人所述的势函数 [8], 其求和

所有管道的度，并乘以一个取决于每条管道端点的常数因子，其值为 $O(d)$。求解同步平面性实例时执行的每次操作都在 $O(\Delta)$ 时间内完成（其中 Δ 为管道的最大度），并至少使势函数减少 1。因此，可以在 $O(d \cdot \Delta)$ 时间内得到一个不含管道的归约后实例，该实例可在线性时间内求解。◀

参考文献

1 P. Angelini、T. Bläsius 和 I. Rutter。平面图互对偶性测试。《计算几何与应用国际期刊》, 24(4):325–346, 2014。 arXiv:1303.1640,doi:10.1142/S0218195914600103. 2 P. Angelini 和 G. Da Lozzo。带管道的聚类平面性。Algorithmica, 81(6):2484–2526, 2019。 doi:10.1007/s00453-018-00541-w. 3 P. Angelini、G. Di Battista 和 M. Patrignani。以 $O(n^4)$ 时间求平面图的最小深度嵌入。Algorithmica, 60(4):890–937, 2009。doi:10.1007/s00453-009-9380-6. 4 P. Angelini、G. D. Lozzo、G. Di Battista 和 F. Frati。嵌入平面图的条带平面性测试。Algorithmica, 77(4):1022–1059, 2016。 doi:10.1007/s00453-016-0128-9. 5 T. C. Biedl、G. Kant 和 M. Kaufmann。满足四连通约束的平面图三角剖分。Algorithmica, 19(4):427–446, 1997。 doi:10.1007/PL00009182. 6 D. Bienstock 和 C. L. Monma。平面图中的最优包围区域。Networks, 19(1):79–94, 1989。 doi:10.1002/net.3230190107. 7 D. Bienstock 和 C. L. Monma。关于以最小化某些距离度量为目标嵌入平面图的复杂性。Algorithmica, 5(1):93–109, 1990。 doi:10.1007/bf01840379. 8 T. Bläsius、S. D. Fink 和 I. Rutter。同步平面性及其在受约束平面性问题中的应用。载于第29届欧洲算法年度研讨会（ESA'21）论文集, 《计算机科学讲义》第204卷, 第19:1–19:14页, 2021。 doi:10.4230/LIPIcs.ESA.2021.19. 9 T. Bläsius 和 I. Rutter。将聚类平面性视为组合嵌入问题的新视角。《理论计算机科学》, 609:306–315, 2016。 arXiv:1506.05673, doi:10.1016/j.tcs.2015.10.011. 10 T. Bläsius 和 I. Rutter。同时PQ排序及其在约束嵌入问题中的应用。《ACM算法汇刊》, 12(2):16:1–16:46, 2016。 doi:10.1145/2738054. 11 T. Bläsius、I. Rutter 和 D. Wagner。具有凸弯曲成本的最优正交图绘制。《ACM算法汇刊》, 12(3):33:1–33:32, 2016。 doi:10.1145/2838736. 12 K. S. Booth 和 G. S. Lueker。使用PQ树算法测试连续1性质、区间图和平面图。《计算机与系统科学杂志》, 13(3):335–379, 1976。 doi:10.1016/s0022-0000(76)80045-1. 13 G. Brückner、M. Himmel 和 I. Rutter。用于向上平面性的类SPQR树嵌入表示。载于 D. Archambault 和 C. D. Tóth 编，第27届图绘制与网络可视化国际研讨会（GD'19）论文集, 《计算机科学讲义》第11904卷，第517–531页。施普林格，2019。 doi:10.1007/978-3-030-35802-0_39. 14 Z.-Z. Chen、X. He 和 C.-H. Huang。在线性时间内求平面图的双欧拉迹[CMOS VLSI电路设计]。载于第40届计算机科学基础年度研讨会（FOCS'99）论文集。IEEE，1999。 doi:10.1109/sffcs.1999.814603. 15 P. F. Cortese、G. Di Battista、F. Frati、M. Patrignani 和 M. Pizzonia。c-连通聚类图的C-平面性。《图算法与应用杂志》, 12(2):225–262, 2008。 doi:10.7155/jgaa.00165. 16 G. Di Battista 和 R. Tamassia。增量平面性测试。载于第30届计算机科学基础年度研讨会（FOCS'89）论文集，第436–441页。IEEE，1989。 doi:10.1109/sfcs.1989.63515.

20 节点扩展下维持三连通分量

17 G. Di Battista 和 R. Tamassia。基于SPQR树的在线图算法。载于第17届自动机、语言与编程国际研讨会（ICALP'90）论文集，第598–611页。施普林格，1990。 doi:10.1007/bfb0032061. 18 G. Di Battista 和 R. Tamassia。基于SPQR树的三连通分量在线维护。Algorithmica, 15(4):302–318, 1996。 doi:10.1007/bf01961541. 19 G. Di Battista 和 R. Tamassia。在线平面性测试。SIAM计算杂志, 25(5):956–997, 1996。 doi:10.1137/s0097539794280736. 20 W. Didimo、G. Liotta、G. Ortali 和 M. Patrignani。在线性时间内绘制平面三图的最优正交图。载于第14届ACM-SIAM离散算法年度研讨会（SODA'20）论文集，第806–825页。SIAM，2020。 doi:10.1137/1.9781611975994.49. 21 D. Eppstein、Z. Galil、G. F. Italiano 和 T. H. Spencer。基于分隔器的稀疏化。《计算机与系统科学杂志》, 52(1):3–27, 1996。 doi:10.1006/jcss.1996.0002. 22 M. Fedarko、J. Ghurye、T. Treagen 和 M. Pop。Metagenomescope：基于Web的宏基因组组装图分层可视化。载于 F. Frati 和 K.-L. Ma 编，第25届图绘制与网络可视化国际研讨会（GD'17）论文集，第630–632页。施普林格，2017。（海报）。URL: https://gd2017.ccis.northeastern.edu/files/posters/fedarko-metagenomescope.pdf, doi:10.1007/978-3-319-73915-1. 23 Q.-W. Feng、R. F. Cohen 和 P. Eades。聚类图的平面性。载于 P. G. Spirakis 编，第3届欧洲算法年度研讨会（ESA'95）论文集，《计算机科学讲义》第979卷，第213–226页。施普林格，1995。 doi:10.1007/3-540-60313-1_145. 24 D. Franken、J. Ochs 和 K. Ochs。含多端口元件网络的波数字结构生成。IEEE电路与系统汇刊I：常规论文, 52(3):586–596, 2005。 doi:10.1109/tcsi.2004.843056. 25 R. Fulek、J. Kynčl、I. Malinović 和 D. Pálvölgyi。聚类图平面性测试再探。《电子组合学杂志》, 22(4), 2015。 doi:10.37236/5002. 26 R. Fulek 和 C. D. Tóth。原子可嵌入性、聚类平面性与可增厚性。ACM学报, 69(2):13:1–13:34, 2022。 arXiv:1907.13086v1, doi:10.1145/3502264. 27 Z. Galil、G. F. Italiano 和 N. Sarnak。全动态平面性测试及其应用。ACM学报, 46(1):28–91, 1999。 doi:10.1145/300515.300517. 28 C. Gutwenger。SPQR树在图绘制平面化方法中的应用。博士论文，2010。URL: https://eldorado.tu-dortmund.de/bitstream/2003/27430/1/diss_gutwenger.pdf. 29 C. Gutwenger、M. Jünger、S. Leipert、P. Mutzel、M. Percan 和 R. Weiskircher。聚类图C-平面性测试的进展。载于 S. G. Kobourov 和 M. T. Goodrich 编，第10届图绘制国际研讨会（GD'02）论文集，《计算机科学讲义》第2528卷，第220–235页。施普林格，2002。 doi:10.1007/3-540-36151-0_21. 30 C. Gutwenger 和 P. Mutzel。SPQR树的线性时间实现。载于第8届图绘制国际研讨会（GD'20）论文集，第77–90页。施普林格，2001。 doi:10.1007/3-540-44541-2_8. 31 J. Holm 和 E. Rotenberg。多对数时间全动态平面性测试。载于 K. Makarychev 、 Y. Makarychev 、 M. Tulsiani 、 G. Kamath 和 J. Chuzhoy 编，第 52 届 ACM SIGACT计算理论年度研讨会（STOC'20）论文集，卷号abs/1911.03449，第167–180页。ACM，2020。 arXiv:1911.03449,doi:10.1145/3357713.3384249. 32 J. Holm 和 E. Rotenberg。最坏情况下多对数增量式SPQR树：嵌入、平面性与三连通性。载于第14届ACM-SIAM离散算法年度研讨会（SODA'20）论文集，第2378–2397页。SIAM，2020。 doi:10.1137/1.9781611975994.146. 33 J. E. Hopcroft 和 R. E. Tarjan。将图分解为三连通分量。SIAM 计算杂志 , 2(3):135–158, 1973。 doi:10.1137/0202012. 34 T. Lengauer。分层平面性测试算法。ACM学报, 36(3):474–509, 1989。 doi:10.1145/65950.65952.

35 G. Liotta、I. Rutter 和 A. Tappini。同步 FPQ 排序与混合平面性测试。载于第 46 届信息学理论与实践当前趋势国际会议（SOFSEM'20）论文集，第 617–626 页。施普林格，2020年。 doi:10.1007/978-3-030-38919-2_51。36 S. Mac Lane。平面组合图的结构特征。《杜克数学杂志》，3(3):460–472，1937年。 doi:10.1215/S0012-7094-37-00336-3。37 P. Mutzel。图绘制中的SPQR树数据结构。载于 J. C. M. Baeten、J. K. Lenstra、J. Parrow 和 G. J. Woeginger 编，第 30 届自动机、语言与编程国际研讨会（ICALP'03）论文集，《计算机科学讲义》第 2719 卷，第 34–46 页。施普林格，2003年。 doi:10.1007/3-540-45061-0_4。38 J. A. L. Poutré。图的三连通分量维护。载于第 19 届自动机、语言与编程国际研讨会（ICALP'92）论文集，第 354–365 页。施普林格，1992年。 doi:10.1007/3-540-55719-9_87。39 J. A. L. Poutré。用于增量平面性测试的 Alpha 算法（初步版本）。载于第 26 届 ACM 计算理论研讨会年会（STOC'94）论文集。ACM Press，1994年。 doi:10.1145/195058.195439。40 J. Vanhatalo、H. Völzer 和 J. Koehler。精化过程结构树。《数据与知识工程》，68(9):793–818，2009年。 doi:10.1016/j.datak.2009.02.015。41 A. von Manteuffel 和 C. Studerus。Reduze 2——分布式费曼积分约化。2012年。arXiv:1201.4330。42 R. Weiskircher。SPQR 树在图绘制中的新应用。博士论文，萨尔兰大学，2002年。 doi:10.22028/D291-25752。43 J. Westbrook。快速增量平面性测试。载于第 19 届自动机、语言与编程国际研讨会（ICALP'92）论文集，第 342–353 页。施普林格，1992年。 doi:10.1007/3-540-55719-9_86。44 Y. Zhang、W. Luk、H. Zhou、C. Yan 和 X. Zeng。用于多重图形化光刻的成对着色布局分解。载于 J. Henkel 编，IEEE/ACM 计算机辅助设计国际会议（ICCAD'13）论文集，第 170–177 页。IEEE，2013年。doi:10.1109/ICCAD.2013.6691115。