arXiv:2603.03909v1 [cs.DS] 4 Mar 2026

通过最近公共祖先方法进行超气泡枚举

Athanasios E. Zisis[1*]和 P°al Sætrom[1,2,3*]

[1*]计算机科学系，挪威科技大学，Sem Sælands vei 9，特隆赫姆，7034，挪威。[2]挪威科技大学临床与分子医学系，Erling Skjalgsons gate 1，特隆赫姆，7491，挪威。

[3]中央行政处，圣奥拉夫医院，特隆赫姆，7006，挪威。

*通讯作者。电子邮箱：athanas.zisis@gmail.com；
pal.satrom@ntnu.no；

摘要

泛基因组学使用基于图的模型来表示和研究同一物种个体之间或不同物种之间的遗传变异。在此类变异图中，穿过图的一条路径代表一个个体基因组。因此，编码局部不同路径的子图就是具有不同遗传变异的基因组区域，检测此类子图是研究遗传变异不可或缺的一环。

双边图是一种变异图，使用黑色和灰色两种边分别表示基因组序列以及序列之间的邻接关系。双边图中的超泡是极小子图，表示一组有限的序列变异，这些变异均以两条不同的序列为起点和末端；也就是说，超泡是无环的，且其中所有路径都通过两条不同的黑边进入和离开。因此，超泡是缠结的一种特殊情况；缠结是与图的其余部分通过两条黑边相连的极小子图。

在此，我们证明，任何双向图都可以转换为二部双边图，在该图中，最近公共祖先查询可以判定某个缠结是否为超泡。这由此得出一种 $O(Kn)$ 算法，可在一组 K 缠结中找出所有超泡；相较于先前在含有 n 个节点和 m 条边的双边图上采用的$O(K(n+m))$ 朴素方法，该算法有所改进。因此，我们在真实和合成变异图上开展的基准实验表明，该方法在环和死胡同路径较少的图，以及边较多的稠密图上，均能缩短

环和死胡同路径较少的图，以及边较多的稠密图的运行时间。

关键词： 超泡、缠结、双边图、变异图、泛基因组学

1 引言

泛基因组学不再依赖单一参考，而是研究来自同一物种的所有基因组。这种方法有助于揭示基因组的共有部分和变异部分，减少参考偏倚。它用于构建共享的基因组参考，在不同单倍型之间比对测序读段，并大规模比较基因组 [1]。

泛基因组涵盖物种内遗传物质的完整集合，包括所有个体中都存在的核心基因，以及在不同群体间存在差异的附属基因。这种对物种遗传多样性的全面视角，有助于更深入地理解不同遗传要素如何影响表型性状、适应和进化过程。通过研究泛基因组，研究人员不仅可以考察共有的遗传特征，还可以研究那些导致个体和群体层面差异的变异 [1]。

变异图是一种双向DNA序列图（见预备知识），为表示泛基因组提供了一种结构化方式，能够在统一框架内捕获遗传变异。节点表示DNA序列，并包含唯一标识符以及一个隐式的反向互补序列，用于表示该序列的相反朝向。节点的每一种朝向称为节点链。边表示其连接的节点序列之间存在的邻接关系，通过指定节点之间的链接方式来定义图的基础拓扑。图中的路径表示节点链上的序列或游走，从而有效编码具体序列，例如嵌入图中的基因组和单倍型。双边图是双向变异图的一种替代表示形式，但二者等价 [2]。在双边图中，DNA序列沿特殊黑边编码，而邻接关系则通过灰边编码。与单一线性参考基因组相比，使用此类图作为参考能够高效建模群体间的基因组变异，并减少参考偏倚 [3]。

在变异图中，表示等位基因多样性的部分子图包括气泡、superbubble 和超泡 [2, 4, 5], ，它们都是缠结的变体（见预备知识） [2]。有向图中的气泡是一个子图 X ，其中图内相同入口 s 和出口 t 节点之间有两条可选的、节点不相交的路径[5]。superbubble 将气泡推广为入口 s 和出口 t之间可以有多条但数量有限的内部路径，同时要求 X 是极小的，且 s 和 t 是 X [5]唯一的入口和出口节点。superbubble 具有重要意义，因为它们表示不同的变异子图，计算成本较低；由于路径数量有限且不存在死端，因此很可能没有错误 [6–8]。

BubbleGun [5],通过实现 Ondera算法 [9], 来枚举超气泡，因此也能枚举气泡。它会扫描（s, t）候选项，并在

子图包含环，或存在（除 t之外的）没有出边的节点时，中止扫描。该算法的平均时间复杂度为线性时间，在实际应用中处理大图时也几乎呈线性增长，但其最坏情况下的时间复杂度为二次，即 $O(n(n+m))$，其中 n 是节点数， m 是边数 [9]。

超泡通过将超泡子图 X 定义为由两条不同的黑边（2-BEC）与图的其余部分 [2] 相连，从而推广了双边图中的 superbubble。与 superbubble 类似，超泡 X 是极小且无环的，并且只有一个入口和一个出口，分别为 s 和 t。缠结进一步推广了超泡，它只要求子图满足 2-BEC 且为极小。

计算缠结时，首先构建一个仙人掌图 [2, 10] ，以概括图的最小割结构，然后生成缠结组件的基于树的嵌套。与针对所有端对进行朴素暴力搜索形成鲜明对比的是，这种方法运行效率很高（实际应用中接近线性）。基于仙人掌图的层次结构会返回一个非重叠的缠结分解集合，因此会省略一些彼此重叠、但根据正式的缠结定义 [2]仍然有效的缠结。

可以在缠结的子图中运行广度优先搜索（BFS）或深度优先搜索（DFS），检查其中是否存在环，或是否存在没有出灰边的节点（ s 或 t 除外），以判断该缠结是否为超泡。因此，假设双边图中的缠结数量为 K，则枚举所有超泡的复杂度为$O(K($ $n+m))$。

本文重点研究如何在给定一组缠结的变异图中识别所有超泡。给定一个包含 n 个节点和 m 条边的变异图中的 K 个缠结集合，我们提出一种 $O(Kn)$ 算法，用于枚举图中的所有超泡。该算法基于最近公共祖先查询，改进了朴素方法的 $O(K($ $n+m))$ 复杂度；在大多数实际图中，其运行时间接近 $O(K)$。

本文其余部分安排如下。第 2 节介绍预备知识，第 3 节阐述两类算法：第 3.1 节基于经典方法，第 3.2 节则提出我们的理论框架，并给出完整证明和复杂度界。第 4 节详细介绍材料与方法，包括数据集、预处理和实现细节。第 5 节报告结果，将我们的方法与现有的朴素方法进行比较，并结合理论阐释研究发现。第 6 节讨论并总结全文，第 7 节补充介绍前文各节提及的特定理论。

2 预备知识

有向图 $G=(V_G,\ E_G)$ 由一组节点 V_G 和一组将它们连接起来的有向边 E_G组成，其中每条边 $e=(v_i,\ v_j)\in E_G|v_i,\ v_j\in V_G$ 都表示一种具有特定方向的关系（即从节点 v_i 到节点 v_j的路径）。双向图将有向图的概念加以推广，允许每条边的每个末端各自具有朝向（“左侧”或“右侧”），以表示边所连接的节点一侧。

将双向图 $D=(V_D, E_D)$ 中的每个节点 v 替换为由特殊黑边连接的两个节点 v_{left}, v_{right} ，即可将双向图转换为

等价的双边图 $B(D) = (V_{B(D)}, E_{B(D)})$ [2]。D, E_D 的原始边称为灰边。因此，$E_{B(D)} = B_{B(D)} \cup G_{B(D)}$，其中 $G_{B(D)} = E_D$ 且 $B_{B(D)} = \{\{v_{\text{left}}, v_{\text{right}}\} | v \in V_D\}$。下文中，对于双边图中的任意节点 x，x' 指与 x 所连接的黑边另一端的节点。当朝向很重要时，L 和 R 分别指 v_{left} 和 v_{right}。

图中的环是指一条从同一节点出发并回到该节点的闭合路径，途中不重复任何节点或边 [11]。在有向图中，有向环是一条从同一节点出发并回到该节点、且不重复任何其他节点的闭合有向路径。此外，若允许自环，则 (v, v) 弧是长度为 1 的有向环 [12]。

连通分量（简称分量）是图的一个子图，其中每一对节点都由一条路径相连，且与子图外部的其他节点没有连接。图中的桥边是一条边，移除它会使图变得不连通，从而将其拆分为若干独立的分量。这条边对于维持图的整体连通性至关重要 [11]。

在有向图 $G = (V_G, E_G)$ 中，强连通分量是一个极大子图 $G' = (V_{G'}, E_{G'})$，其中 $V_{G'} \in V_G, E_{G'} \in E_G$，使得$G'$ 中的每一对节点都由一条路径相连。也就是说，对于 G' 中的每一对节点，两个方向上都应存在一条路径，并且在不破坏该性质的前提下，不能用来自 G 的额外节点或边扩展 G'。

在有向图中，没有入边的节点称为源点，而没有出边的节点称为汇点 [12]。源点和汇点都是仅包含单个节点的强连通分量。

缠结 [2] 是双边图的一个极小子图，由一对不同的节点 $\{x, y\}$ 确定，这两个节点之间有不同的黑边（即 x 和 y 之间没有黑边相连），并且 $\{x, y\}$ 满足以下性质：

- 分离性：移除与 x 和 y 相连的黑边后，图会被分成互不连通的分量。其中一个分量 X 包含 x 和 y，但不包含 x' 和 y'。
- 极小性：在 X 内，不存在一条黑边，移除该边后会将 X 分成两个或更多互不连通的分量。

在有向图 G 中，由一对不同的节点（x，y）确定的子图，若满足以下条件 [9]，则称为 superbubble：

- 可达性：存在一条从 x 到 y 的路径。
- 匹配性：不经过 y 时从 x 可达的节点集合，与不经过 x 时可到达 y 的节点集合相同；这里，经过一个节点是指进入并离开该节点。
- 无环：由 X 形成的子图不包含环。
- 极小：在 X 中，除 y 之外，没有其他节点能与 x 构成有效的超泡对；对 y 同样适用对称关系。

简言之，superbubble 是具有单一入口节点和出口节点的极小子图（也就是说，子图中没有其他源点或汇点），且这些节点之间的路径数量有限。

超泡将 superbubble 推广到了双边图中。具体而言，超泡是一种既无环又不含尖端的缠结，其中尖端是指没有入射或出射灰边（也就是说，没有关联灰边）的节点 [2]。

3 算法

在变异图的双边图表示中，黑边代表序列块，灰边代表它们之间的邻接关系。特定序列（例如个体基因组）被编码为穿过图的路径。任何此类图路径都始终交替经过黑边和灰边。一般而言，黑边和灰边都可以沿任一方向遍历，因此可以发生单个片段节点反转，也可以发生由多个此类反转构成的更长路径 [2, 4]。

下文中，我们假设每个双边图都是有向的；也就是说，所有边都只能沿一个方向遍历，从而避免倒位，包括跨越多个节点的长倒位。具体而言，我们定义以下遍历规则：

- 黑边只能从其左侧（L）节点遍历到右侧（R）节点；
- 灰边只能从黑边的一个 R 节点遍历到另一条黑边的 L 节点。

通过这种称为比对挽救的反转 [13]，的方法，我们得到了更简单的图拓扑。具体而言，双边图是二分图，由节点集 L 和 R 组成。从现在起，提到图的遍历时，我们始终指遵循上述遍历规则。

实际上，变异图（例如 GFA 格式的变异图）是双向图，但可以在线性时间内将其转换为仅向前流动的简单有向图。每个序列都保留正向；每当某条连接指向反方向时，我们就添加该序列的反向互补副本，并按照灰边（邻接关系）的相应朝向将其正常连接起来，然后将它们转换为具有所需特性的双边图；详情请参见第 7 节。

请注意，采用这种对一般双边图加以限制的方法不会造成信息损失，因为上述变换保留了给定双向图（GFA 格式）中的所有路径：其中的每条路径在有向图中都有对应的正向路径，而且结果始终可以映射回原始结构。因此，这种方法在简化图拓扑的同时仍然可靠，代价是在最坏情况下图中的边和节点数量会翻倍。

不失一般性，我们假设所有双边图都是连通的，因为对于非连通图，可以将每个连通分量作为单个连通双边图独立处理。我们假设每个双边图都有唯一的源点（下文称为“根”），从根出发遍历图即可到达图中的所有其他节点。由于根没有入边，它代表双边图所编码的泛基因组序列的起点。实际上，变异图可能没有唯一的根，但可以按照第 7 节所述转换图来解决这一问题。最后，我们假设双边图至少有一个汇点（“末端”）。如果有多个这样的节点，我们选择从根开始的广度优先搜索树中位于最深层级的汇点；如果这样的节点不止一个，

则任意选择其中一个。如果图中没有汇点，则可以按照第 7 节所述构造一个。

我们将分量 C_{start} 和 C_{end} 分别定义为双边图中包含根和末端节点的任意子图。显然， C_{start}=C_{end} 是可能的。我们在双边图中定义如下左右方向。当且仅当从根到 x 的最短路径比从根到 y 的最短路径更短时，节点 x 位于节点 y 的左侧。在双边图中，路径的长度是沿路径遍历的边数。

考虑左右方向和缠结 $\{x, y\}$，我们分别将最左侧和最右侧的边界节点称为 sn_1 和 sn_2。如果两个节点到根的最短路径长度相同，则我们任意指定 sn_1 和 sn_2。

由于此前定义的双边图遍历规则，在从根开始的广度优先搜索中，黑边的每个 R （右侧）节点只能通过同一黑边对应的 L （左侧）节点到达，因为禁止在 R 节点处出现入射灰边。此外，由于我们处理的图是连通的，且每个节点都能从根到达，因此每条黑边都是广度优先搜索树中的一条边。

根据我们之前的定义，在含有多个节点的双边图中，根始终只与一条黑边相连。因此，根是左节点，而与之相连的节点是右节点 R_r ，其与根的距离为1。此外，由于遍历规则和图的结构，黑边与灰边交替出现，因此每个 L 节点和 R 节点到根的距离分别为偶数和奇数。推论1 由此可得。

推论1。 由一个 R 节点和一个 L 节点定义的缠结，其边界 sn_1 和 sn_2到根的距离不可能相等。因此，我们总能根据两个边界节点中哪个离根更近，将此类缠结正确且唯一地排序表征为 $R-L$ 或 $L-R$ 。

每个有向无环图都有一个源点和一个汇点， [12],因此我们得到推论2。

推论2。 至少有一个源点和一个汇点的无环双边图（与有向图同构）。

根据缠结定义 [2], ，最小缠结是连接相邻黑边的两个节点的一条灰边。请注意，尽管这个最小缠结实际上并不表示图中的变异，但根据定义，它仍然是一个有效的缠结，也是一个有效的极小（平凡）超泡。

定义 n 1 我们将最小缠结称为平凡缠结（见图 g. 1**D**).

3.1 朴素方法中的超泡和缠结

给定一个双边图的缠结集合（表示为元组集合），其中每个元组包含缠结的边界节点 sn_1 和 sn_2 ，需要找出其中哪些是超泡，

我们需要针对每个缠结，检查由其边界节点 sn_1 和 sn_2 定义的子图中是否存在环和端点，例如可以使用深度优先搜索或广度优先搜索。如果缠结中不存在环和端点，则它是一个超泡；否则就不是。因此，假设缠结的数量上界为 K，那么朴素方法的复杂度为 $O(K($ $n+m))$，因为我们需要针对每个缠结运行一次深度优先搜索或广度优先搜索。

3.2 通过最近公共祖先查询确定超泡和缠结

在双边图中，源点和汇点的边天然是黑边；因此，源点和汇点分别是最左侧和最右侧黑边的左节点和右节点。此外，根据定义，源点不能有出灰边，汇点不能有入灰边。因此，根和汇点都构成端点，这就得到了引理 1 和定理1。

引理 1。 任何其子图包含 C_{start} 或 C_{end} 的缠结都不能构成超泡。

定理 1 *A snarl that is defined by 2 left (or 2 right) nodes of black edges can never form an ultrabubble (see Fig. 1**A**).*

Proof 假设我们有一个由 2 个左节点 x、 y 定义的缠结。如果该缠结包含 C_{start} 或 C_{end}，那么根据引理 1，该缠结不能构成超泡。假设这样的缠结是一个超泡，因此它既不包含 C_{start}，也不包含 C_{end}。那么该缠结将属于另一个连通分量 C_{mid}。由于它是一个超泡， C_{mid} 是一个无环双边图。但根据推论2， C_{mid} 应至少有一个源点和一个汇点。由于缠结的节点都是黑边的左节点，它们不能有出边，因为从 x 和 y 出发的出黑边均已被移除；并且根据定义，黑边的左节点不能有出灰边。因此， x 和 y 都构成 C_{mid} 中的汇点，所以 C_{mid} 应至少有一个源点 s，且该源点不同于定义该缠结的节点 x、 y。源点 s 不能有入灰边，因为这将意味着存在一个环（矛盾），或者意味着 s 不是源节点，因为存在另一个节点 s'，它与至少由 s'、一条黑边、另一个节点以及一条进入 s 的灰边组成的子图共同形成源点。注意，除了与缠结边界节点相连的两条边之外，我们没有删除任何其他黑边。因此，源节点是一个黑边的左节点，且没有自环汇入该节点，因为缠结中不存在环。但是，如上所述，该节点不与任何灰边相邻，因此它是一个端点，这与缠结是超泡的假设相矛盾。对于由 2 个右节点定义缠结的情况，证明相同，因为这些节点将构成源点，因此至少会存在一个汇点，而该汇点同样会构成一个端点。□

观察1。根据缠结 $\{x, y\}$ 的定义，删除 2 条黑边（x， x'）和（y， y'）后，双边图可从包含一个连通分量变为包含 2 个或 3 个连通分量，见图1C。当且仅当定义该缠结的节点所对应的 2 条黑边是双边图中的桥边时，才会出现 3 个分量的情况。在另一种包含 2 个分量的情况下，该缠结可能包含 (i) C_{start}、(ii) C_{end}、(iii) 两者，或 (iv) 两者均不包含 C_{start} and

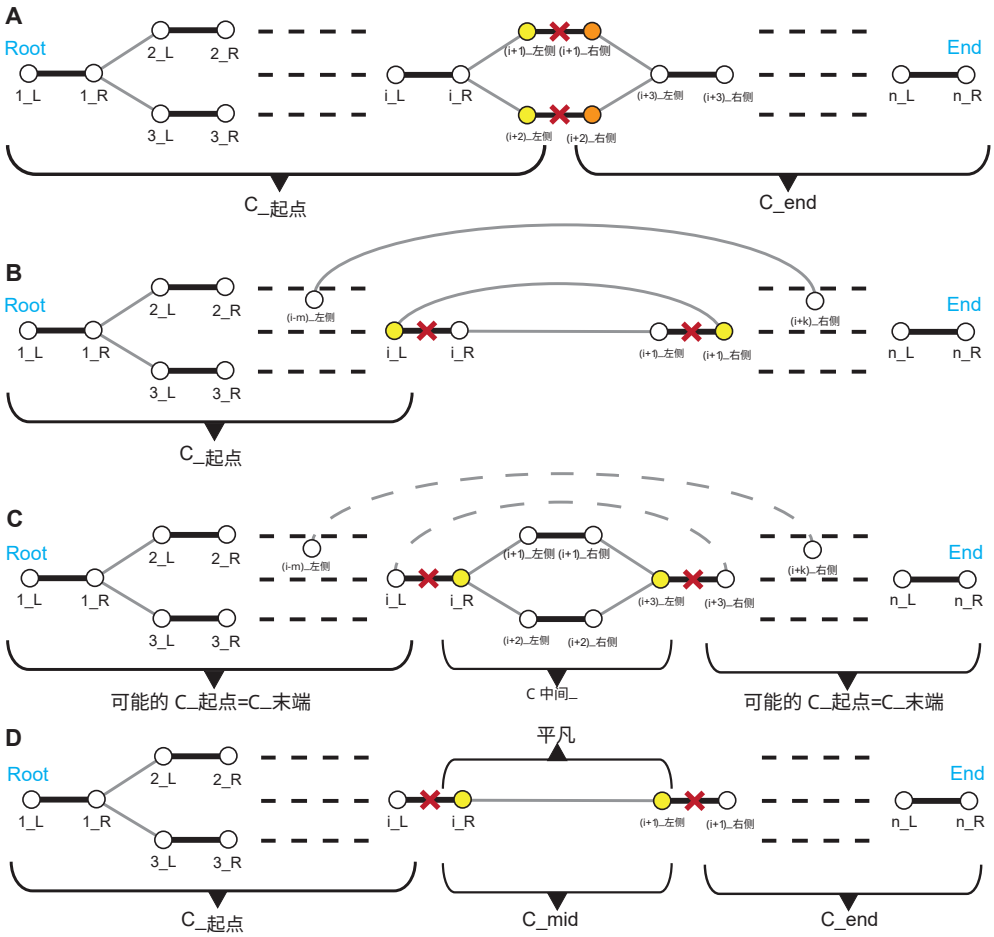

图 1 在双边图中，无法（**A-B**）以及可以（**C-D**）形成超泡的缠结。缠结以黄色或橙色标示；红色叉号表示相应被移除的黑边。（**A**）任何 $L-L$（左-左）或 $R-R$（右-右）缠结都无法形成超泡，因为它分别属于C_{start} 或 C_{end}连通分量。（**B**）任何 $L-R$缠结都无法形成超泡，因为作为该缠结一部分的 L 节点将位于 C_{start} 连通分量中。（**C**）任何 $R-L$ 缠结都会将图划分为三个或两个连通分量。只有在不存在连接 C_{start} 和 C_{end}的灰边时，才会出现前一种情况；此时，被移除的黑边是桥边。在后一种情况下，至少存在一条这样的灰边，于是 C_{start}=C_{end}。（**D**）由单条灰边连接的 $R-L$ 缠结既是平凡缠结，也是平凡超泡。

C_{end}。因此，若缠结既不包含 C_{start} 也不包含 C_{end}，则要么 2 条黑边（x，x'）和（y，y'）是桥边（3 个分量的情况），要么该缠结嵌套在图中一个不包含 C_{start} 也不包含 C_{end}的区域内；见图 1**C**。

引理 2. 任何缠结，若其最左侧边界 sn_1 是一条黑边的左节点（L），则不能形成超泡（见图 *1B*）。

Proof 该 sn_1 节点根据定义是该缠结的两个边界节点中最接近根的节点。此外，虽然 sn_1由于是左节点，可以有一条或多条入射灰色边，但其对侧节点 sn_1'位于与 sn_1相连的黑边的另一端，不能有任何入射灰色边，因为根据定义不允许沿黑边回溯。因此，删除黑边（sn_1, sn_1'）后形成的、且包含节点 sn_1 的子图也将包含根；也就是说，根与 sn_1 属于同一连通分量 C_{start}。因此，根据引理 1，这样的缠结不能形成超泡。□

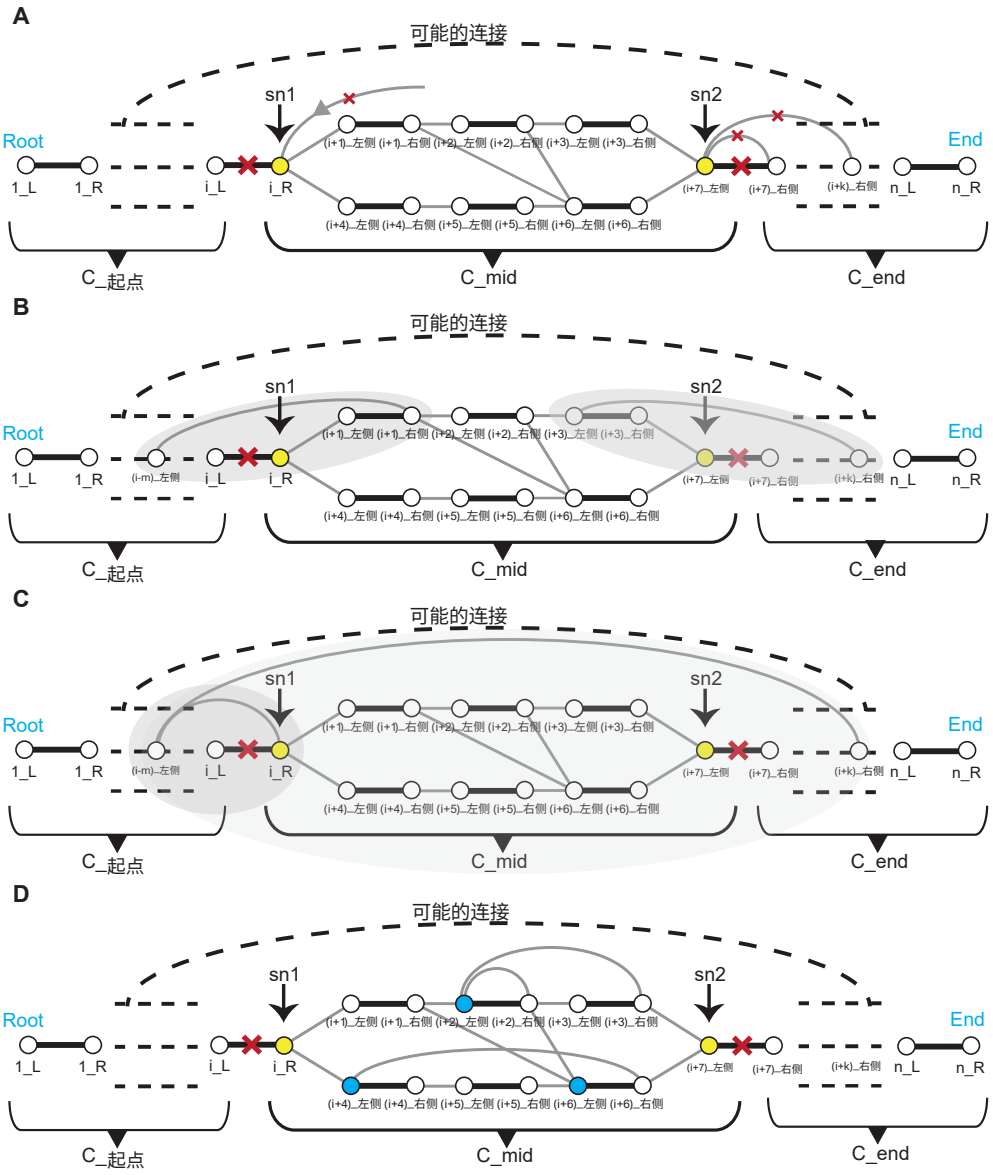

图2 与缠结（**A-C**）不兼容、与缠结（**D**）兼容的环。**(A)** 缠结的边界节点 sn_1, sn_2 不能是环闭合节点。根据定义， sn_1 不能有入灰边。对于 sn_2，若存在一条入环闭合灰边，就会违反 snarl 定义，因为这会使 sn_2 和 sn_2'（此处分别为（$i+7$）L 和（$i+7$） R）处于同一连通分量中。**(B-C)** 缠结的内部节点不能 **(B)** 成为一条部分位于缠结内部、部分位于缠结外部的环的环闭合节点，也不能 **(C)** 有灰边连接到缠结外部的节点，否则就会违反 snarl 定义中的分离准则。**(D)** 缠结中位于其边界节点 sn_1 和 sn_2 之间的环闭合节点，表示一个嵌套在该缠结边界内的环。请注意，识别出的环闭合节点取决于深度优先搜索执行；此处， （$i+4$）L 或（$i+6$） L 都表示同一个嵌套环。

根据定理 1 和引理 2，我们可以直接排除所有 $R-R$、 $L-L$和 $L-R$缠结，将其判定为非超泡。因此，我们只需检查$R-L$ 缠结是否符合超泡准则。

假设我们有一个先前定义的有根双边图，其中节点的标签各不相同。从根节点开始，针对遵循我们遍历规则的有向图运行经过修改的深度优先搜索（参见第 7 节），我们可以在线性时间内找出图中每个环的所有端点和所有环闭合节点，并将它们标记为蓝色。请注意，由于遍历规则的限制，只有左节点才能成为环闭合节点。

定理 2 *An $R-L$ snarl $\{x,y\}$ where $sn_1=x$ is a right node and $sn_2=y$ is a left ,,**A** node cannot have x or y as cycle closing nodes (see Fig. 2).*

Proof 首先，考虑最右侧边界 $sn_2=y$。假设 y 有一条闭环边。这意味着存在一条从节点 c 出发的入灰边，而 c 比 y 离根节点更远，因此必定存在一条从 y 到 c 的路径（否则就不会形成环）。从 y出发，该路径必须以黑边（y,y'）开始，因为离开 y 别无他法： y 是左节点，而左节点没有出灰边。因此，存在一条从 y' 到 c 的路径。按照缠结定义，移除黑边（y,y'）后，环会被破坏，但从c到 y仍然存在一条灰边。因此，移除黑边 yy'后， y'、 c和 y属于同一个连通分量，这违反了缠结的分离准则。其次，在 $R-L$ 缠结中，最左侧边界 x 不可能是环闭合节点，因为它是黑边的右节点，根据定义不能有入灰边。□

根据定理2，我们得到引理 3。

引理 3. 如果一个环闭合节点出现在一个 $R-L$ 缠结中，则其对应的环完全嵌套在该缠结的子图中（见图2***B,C***）。

*Proof*设 c为某个 $R-L$缠结的边界节点 x、y之间的一个环闭合节点。假设其对应的环并未完全嵌套在该缠结的子图中。则该环的一条路径必须位于snarl子图之外，或在缠结的右侧，或在其左侧。由于该缠结通过 x 和 y与图的其余部分相连，至少存在一个节点 d 与 $d=y'$ 或 $d=x'$共同参与该环，两种情况均有可能。但这样一来，便会存在一条连接 d 与 c的路径。因此，若 d 位于缠结的右侧或左侧，则分别地， y'、 d、 c与 y ，或 d、 x'、 c与 x，将处于同一连通分量中。这两种情况均违反了snarl 定义的分离准则。□

引理4。 位于 $R-L$ 缠结子图中的尖端可以是该子图中的任意节点，但不能是 sn_2 的后代。

*Proof*在一个 $R-L$ 缠结 $\{sn_1,sn_2\}$中， sn_2 只能有入灰边，因为它是一个左节点。假设其中一条入灰边位于一条始于尖端的路径上——也就是说，该节点没有任何关联的灰边。那么，从根出发就无法按要求遍历这个尖端以及通往 sn_2 的相应路径，因为这意味着存在一条出灰边

来自左节点（sn_2），因此我们对根的假设不成立（见图5**A**）。如果存在这样的尖端，则有如下观察2 。□

观察2。与左节点通过灰边相连的尖端，是双边图适配序列方向的潜在根。因此，该图有多个潜在根（见图5**B**；我们将在第 4 节和第 7 节讨论这一真实案例。）显然，sn_1作为右节点，可以有一条通向尖端的出灰边，而该尖端在双边图中将是一个汇点。

超泡是一种 $R-L$ 缠结 $\{sn_1, sn_2\}$，其中不含环或端点。根据引理3，缠结子图中存在环，当且仅当环闭合节点位于 sn_1 和 sn_2之间。此外，根据引理4，缠结中的端点位于 sn_1 和 sn_2之间，因为它可以是 sn_1 的后代，但不能是sn_2的后代。因此，要检查缠结是否为超泡，只需检查 sn_1 和 sn_2之间是否存在端点或环闭合节点。因此，端点和环闭合节点都会针对同一属性进行检查，所以我们将二者都放入一个名为 ftip 的集合中。因此，当且仅当缠结子图中至少有一个节点 $t \in$ftip 位于 sn_1 和 sn_2 之间时，该缠结不是超泡。换句话说， t 是 sn_1 的后代，但不是 sn_2 的后代（见图3**B**）。

根据上述结论，我们采用基于最近公共祖先（LCA）的方法，对给定的 $R-L$ 缠结进行超泡分类。具体而言，当且仅当 $LCA(t, sn_1) = sn_1$ 且$LCA(t, sn_2) \neq sn_2$时，节点 t 位于 $R-L$ 缠结 $\{sn_1, sn_2\}$ 内，其中 $LCA(x, y)$ 是 x 和 y 的最近公共祖先（见图 3）。因此，使用 LCA 判断缠结是否为超泡，最坏情况下需要进行 $O(n)$ 次最近公共祖先查询。

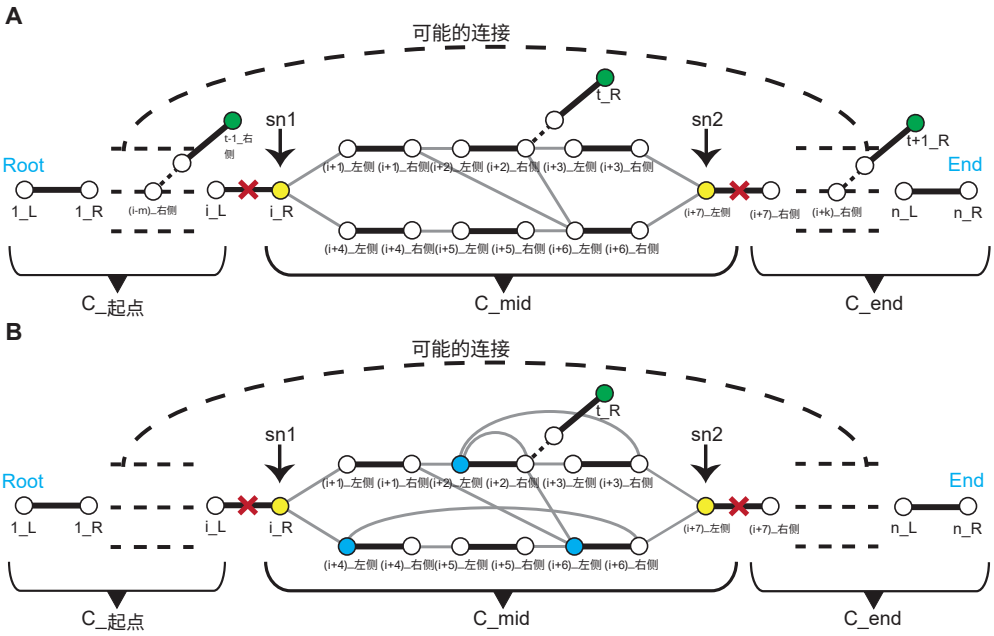

图 3 最近公共祖先（LCA）查询可确定某个节点是否位于 $R-L$ 缠结（黄色）内。（**A**）对于分别位于缠结左侧、内部和右侧的三个端点（绿色）$(t-1)$ R、t R 和 $(t+1)$ R，以下关系成立：$LCA((t-1)\ R, sn_1) \neq sn_1$，$LCA\ t\ R\ sn_1 = sn_1$ $LCA\ t\ R\ sn_2 \neq sn_2$ $LCA\ t+1\ R\ sn_2 = sn_2$ - (,), (,), and (( ),). （**B**）同一组最近公共祖先检查适用于端点（绿色）和环闭合节点（蓝色）。

在有向无环图（DAG）中，最近公共祖先定义明确，[14]，但给定的双边图不一定是有向无环图。在一般有向图中，可以先收缩图的强连通分量（SCC），构造一个称为凝聚有向无环图的有向无环图，从而高效地回答最近公共祖先查询。随后在该有向无环图上回答查询，并使用固定的代表元规则将每个结果映射回原图中的一个节点；详见第 7.2 节。不过，由于我们只关心判断给定节点 t 是否位于给定缠结的边界 sn_1 和 sn_2 之间，因此可以改为在从双边图根节点出发的广度优先搜索树中执行最近公共祖先查询（定理 3）。

定理 3 *Given a biedged graph B with a unique root and corresponding BFS tree B_t, an $R-L$ snarl $\{sn_1, sn_2\}$, and a node t located within the snarl, the LCA queries between t and sn_1 and t and sn_2 in B_t will answer if t is between sn_1 and sn_2 and not more left and outside the frontier than sn_1 (see Fig. 4).* ***B***

Proof 双边图中的每个 $R-L$ 有向缠结 B 显然都有唯一的入口节点和唯一的出口节点（分别为边界节点 sn_1 和 sn_2），这是由定义和遍历规则决定的。假设LCA查询 $LCA(sn_1, t) = c_1$ 位于广度优先搜索树 B_t 的

该图中，并且 c_1 比 sn_1更靠左。由于 t 是该缠结子图中 sn_1 的后代，这一假设意味着存在两条不同的路径 P_1 和 P_2 ，分别从 c_1 通向 sn_1 和 t，并且 sn_1 不属于 P_2。 P_1 必须包含sn_1'，因为通往 sn_1 的唯一路径要经过黑边(sn_1', sn_1)（根据遍历规则和定理 2）。此外， P_2 不能包含(sn_2, sn_2')，因为这条边方向相反，不能用于进入缠结 $\{sn_1, sn_2\}$。因此， P_2 必须包含某条将 t 连接到 c_1的其他边，这意味着 sn_1' 和 sn_1 仍属于同一连通分量，从而违反了snarl 定义的分离准则。同样的逻辑也适用于 t 与 sn_2之间的最近公共祖先查询，因为二者都是 sn_1的后代；也就是说，对于 $LCA(sn_2, t) = c_2$，c_2 必须属于该缠结的子图，除非除(sn_1', sn_1)和(sn_2, sn_2')之外，还存在某条将 $\{sn_1, sn_2\}$ 连接到 B 其余部分（从而将 c_2 连接到 t）的其他边，从而违反分离准则。□

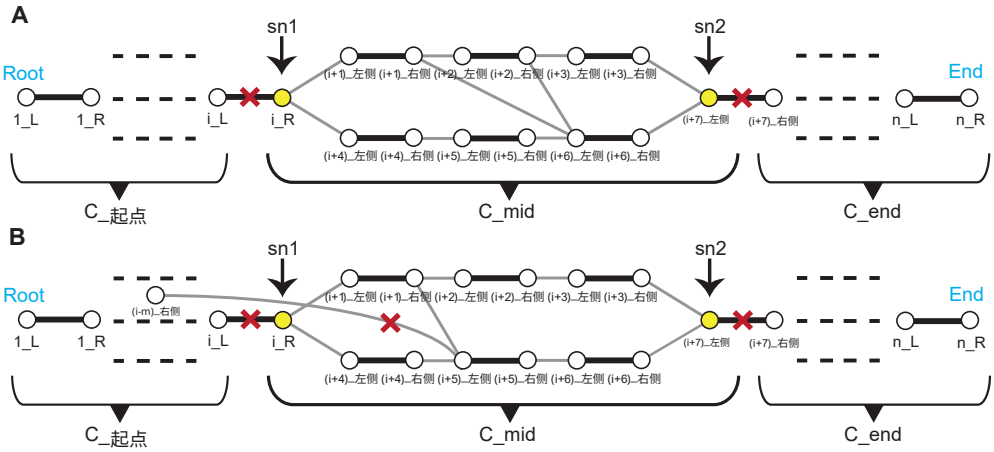

图 4 在广度优先搜索（BFS）树 (LCA_{B_t}) 中进行最近公共祖先（LCA）查询，可确定某个节点是否位于 $R-L$ 缠结（黄色）内。（**A**） LCA_{B_t} 可能给出一个比双边图中的最近公共祖先 (LCA_B LCA_{B_t} $i+$) 更靠近根的结果；3 L $i+6$ L $=$ $i+1$ R LCA_B $i+3$ L $i+6$ L $=$ $i+2$ R LCA_{B_t} 例如，(() ,( )) ( )，而 (() ,( )) ( )。（**B**）永远不会给出高于 sn_1的答案，因为这样的答案会违反缠结的分离准则。

根据定理 3，我们有推论 3。

推论 3。对于给定双边图 B 的任意 R–L 缠结 $\{sn_1, sn_2\}$，只需在该图的广度优先搜索树 B_t 中进行最近公共祖先查询，即可确定集合 $ftip$ 中是否存在节点位于 $\{sn_1, sn_2\}$ 内，尽管在广度优先搜索树 B_t 中进行最近公共祖先查询所得答案可能与在 B 中进行的查询不同。

根据推论 3，我们得到算法 1：给定一个满足遍历规则的双边图 B、其中的 S_{RL} 个 $R-L$ 缠结的集合 B、其中 $ftip$ 个端点和环闭合节点的集合 B，以及 B 的广度优先搜索树 B_t，返回 U 中超泡的集合 B。由于树中的最近公共祖先查询可在常数时间内完成，该算法的复杂度为 $O(K \cdot |ftip|)$，最坏情况下为 $O(Kn)$。

算法1 在双边图中枚举超泡 B

要求： S_{RL} $R-L$ 中缠结的集合、 B中的集合 $ftip$以及 B的广度优先搜索树 B_t。
要求： $LCA_{B_t}(a,\ b)$： B_t中 a 和 b 的最近公共祖先。

$U \leftarrow \emptyset$
对每个 $\{sn_1, sn_2\} \in S_{RL}$ **执行**
    F ←**为真**
    对每个 $t \in ftip$ **执行**
        如果 $LCA_{B_t}(t,\ sn_1) = sn_1$ 且 $LCA_{B_t}(t,\ sn_2) \neq sn_2$ **则**
            F ←**为假**
            跳出
        结束如果
    结束循环
    如果 F =**为真，则**
        $U \leftarrow U \cup \{(sn_1,\ sn_2)\}$
    结束如果
结束循环
返回 U

4 材料与方法

4.1 图

我们从可用来源下载了 GFA 格式的双向变异图（表1），并使用自定义 Python 脚本构建了具有不同规模、气泡复杂度以及不同环和端点数量的合成双边图（表 2-3）。由于部分变异图并非连通图，我们提取并使用了其中部分连通分量或其子图；例如，lpa120.gfa 是 LPA.gfa 的一个包含 120 个节点的子图。

4.2 图预处理

给定一个双向图 D （GFA 格式），我们采用以下步骤将其转换为结构符合遍历规则的双边图。首先，将 D 中的每个片段节点替换为一个左节点和一个右节点，并用黑边（L、 R）连接这两个节点。其次，删除所有连接两个右节点或两个左节点的边。我们采用这种更简单的方法来处理包含此类 $R-R$ 和 $L-L$边的图，而没有按照第7.1节所述的方法转换图，因为这些图仅用于基准测试。转换后的图是一个结构与先前定义的遍历规则一致的双边图，因此与有向图同构。第三，如果图具有多个源点，我们按照图 7 所示构造一个根。完成这些转换后，我们实验中使用的所有变异图都具有一个根，但请注意，第7.2 节介绍了在与有向图同构的双边图中构造单一源点的一般方法。

表 1 实验所用的数据集

Name	源点 URL	参考文献
C4-90.gfa	https://zenodo.org/records/6617246	[15]
chr19.pan.fa.a2fb268.e820cd3.9ea71d8.smooth.gfa	https://42basepairs.com/browse/s3/human-pangenomics/pangenomes/scratch/2021年7月30日 PGGB/染色体?file=chr19.pan.fa.a2fb268.e820cd3.9ea71d8.smooth.gfa.gz	[16], [17]
chr6.pan.fa.a2fb268.2ff309f.1300c8a.smooth.gfa	https://s3-us-west-2.amazonaws.com/human-pangenomics/index.html?prefix=泛基因组/scratch/2021 05-06-PGGB/gfas/	[16], [17]
chr6.C4.gfa	https://github.com/pangenome/ODGI/tree/master/test	[18]
chrM.pan.fa.6626ff2.7748b33.72587dd.smooth.gfa	https://s3-us-west-2.amazonaws.com/human-pangenomics/index.html?prefix=泛基因组/scratch/2021年7月30日 PGGB/染色体/	[16], [17]
LPA.gfa	https://github.com/泛基因组/ODGI/tree/master/test	[18]
lpa120.gfa	https://github.com/pangenome/ODGI/tree/master/test	[18]
MHC-57b.gfa	https://zenodo.org/records/6617246	[15]
酿酒酵母.fa.gz.d1a145e.417fcdf.7493449.smooth.final.gfa	http://hypervolu.me/~erik/酵母/酿酒酵母.fa.gz.d1a145e.417fcdf.7493449.smooth.final.gfa.gz 或 https://doi.org/10.5281/zenodo.18484366	[17], [19]

4.3 识别缠结

我们采用两种方法来识别预处理图中的缠结。首先，我们使用暴力朴素方法，对图中的每一对节点检查其是否符合缠结定义中的分离性和极小性标准。该方法具有立方时间复杂度，但能够识别所有缠结，包括嵌套缠结。由于其时间复杂度较高，我们仅在部分图上使用了这种朴素方法（表2）。

其次，我们使用了vg工具中基于Paten等人 [2]的识别缠结选项，该选项会生成缠结的分层嵌套结构。具体而言，该工具将缠结集合输出为边界节点的元组，并额外给出其中哪些是无环的信息。我们仅保留无环snarl以供进一步处理，因为它们是唯一可能的超泡。由于vg工具所用算法的实际复杂度接近线性时间，我们对所有图都使用了vg工具。

4.4 超泡识别

我们使用朴素方法（第3.1节）和算法1 来识别一组缠结中的超泡。朴素方法检查每个缠结是否存在端点或环。对于算法1，我们首先使用广度优先搜索树计算从根到缠结边界节点的距离，以识别 $R-L$ 缠结集合，依据推论1。其次，我们在线性时间内找出所有端点以及环闭合节点及其对应的闭合灰边，详见第7.3节。第三，我们通过范围最小值查询（RMQ）–欧拉游历组合技术，预先计算从根到所有节点的全部路径，如第7.4节所述，从而使每次LCA查询都能在 O(1)时间内得到解答。

4.5 软件

我们使用 Bandage [https://rrwick.github.io/Bandage/，版本 0.8.1] [20] 来可视化 GFA文件，并使用 vg [https://github.com/vgteam/vg，版本 1.62.0] [3]生成嵌套snarl集合。

所有 Python 脚本均在一台配备 Intel® Core™ i7-1280p 处理器的笔记本电脑上运行，该处理器包含 14 个核心和 20 个线程，电脑配有 16 GB 内存。

我们使用 GPT（由 OpenAI 开发的 ChatGPT）辅助代码开发。代码可在此处获取：https://github.com/athanasios-zisis/UltraLCA。

5 结果

我们使用一组真实的变异图和合成双边图，对基于最近公共祖先的方法（算法 1）与朴素方法（第 3.1 节）进行基准测试，以便在一组缠结中识别超泡；关于图的处理方式，详见方法部分。我们使用 vg工具识别缠结；对于一组小型图，我们还使用朴素的暴力方法识别缠结，因为根据缠结的定义，该方法能够识别所有缠结（表 2）。正如预期，朴素方法识别出的缠结更多，但两组缠结包含的超泡完全相同（比较表 2 和 3 ）。

在大多数情况下，算法 1的表现优于朴素方法（表 3）。然而，当 ftip 集合非常大时，算法 1 会更慢，因为我们要对集合 t 中的每个 $ftip$ 都运行一次 LCA查询。

6 讨论与结论

我们提出了一个理论框架，说明可以使用最近公共祖先查询来判断有根二分双边图中的缠结是否为超泡。该方法的复杂度为 $O(Kn)$，优于朴素方法的 $O(K(n+m)$ 复杂度，其中 K、 n 和 m 分别表示缠结数、节点数和边数。从理论上看，与朴素方法相比，对于端点和环较少的图，或边数较多的稠密图，这种基于最近公共祖先的方法应能缩短运行时间；我们在实际图上进行的基准测试实验也支持这一结论。值得注意的是，将基于最近公共祖先的方法集成到现有的缠结枚举方法中，或许还能进一步缩短运行时间。

表2 使用最近公共祖先和朴素方法，在不同图中由朴素方法生成的缠结内枚举所有超泡的运行时间。”NAME(.gfa)”是图的文件名；”NODES”和”EDGES”分别是图中的节点数和边数；”REMOVED LL&RR EDGES”是预处理期间移除的 $L-L$ 和 $R-R$ 边数；”TIPS/CYCLES/FTIP”分别是尖端数、环闭合节点数和 $ftip$ 集合中的节点数；”SNARLSNAIVE”是朴素算法识别出的缠结数；”SN TIME”是识别缠结所需的运行时间（秒）；”UL.”是超泡数；”REJ”是被拒绝的 $R-L$ 缠结数；”PRE TABLE”是计算RMQ距离表所需的运行时间；”ALGO”和”NAIVE”分别是算法1 和朴素方法的运行时间。

名称（.gfa）	节点	边	已移除左左&右右边	端点/环/FTIP	缠结朴素	SN 时间	UL.	REJ	PRE 表格算法	朴素	
C4-90_VG_rem_nroot	19	20	10	7/0/7	29	0	7	22	0	**0**	0.001
lpa120.gfa	120	169	-	2/0/2	82	6	56	26	0	**0**	0.010
MHC-57b_VG	1068	1496	-	2/0/2	592	3417	376	216	0	**0**	0.59
chrM.pan.fa.6626ff2.7748b33.72587dd.smooth	1395	1887	-	2/0/2	1339	11577	456	883	0	**0**	2.85
chr6.pan.fa.a2fb268.2ff309f.1300c8a.smooth_comp6_rem_nroot	1700	2283	1	4/0/4	1643	25855	584	1059	0.1	**0.01**	9.188
chr6.C4.gfa	1748	2366	-	2/2/3	830	52070	582	248	0	**0**	0.81
SYNAL1	236	289	-	6/4/9	225	23.9	138	87	0	**0**	0.058
SYNAL2	439	569	-	6/10/14	379	148	233	146	0	**0.01**	0.214
SYNAL3	598	767	-	6/12/16	511	373	337	174	0	**0.01**	0.33
SYNAL4	1107	1426	-	5/13/17	860	2376.3	596	264	0	**0.01**	0.81

7 补充材料

7.1 从 GFA 格式的双向图构建一个与有向图同构的双边图

对于 GFA 格式的双向图，我们可以按如下方式将其转换为不存在倒位和/或回溯的双边图。我们只需扫描一次图形片段组装文件（GFA）。对于每个链接，强制两端都采用正向方向（“++”）。如果某一端为反向（“–”），则将该端重定向到一个新节点；该节点存储相同序列片段的反向互补序列，并在首次需要时创建。如果链接包含重叠字符串，则仅当恰有一端翻转时，才反转其操作顺序；如果两端都翻转或都未翻转，则保持不变。以同样的方式更新路径，将每个反向步骤替换为相应的正向方向反向互补节点。由于每一行仅读取和重写一次，且每个序列或重叠字符串最多转换一次，整个转换的复杂度为线性时间。

通过上述过程，我们最终得到一个与有向图同构的双边图，具体说明见本文。

表3 在不同图中，使用最近公共祖先方法和朴素方法枚举 vg 工具生成的缠结中的所有超泡所需的运行时间。此处，vg 使用 -T 标志运行，该标志会输出所有嵌套且无环的缠结，以及平凡缠结；“VG SNARLS -T”表示此类缠结的数量；其他列的详细信息请参见表2 。

名称（.gfa）	节点	边	已移除左侧和右侧边	端点	VG 缠结	-T	UL.	REJ	PRE 表格	算法	朴素	
C4-90_VG_rem_nroot	19	20	10	7	9	7	2	0	0	0	0	
lpa120.gfa	120	169	-	2	56	56	0	0	**0**	0.001		
MHC-57b_VG	1068	1496	-	2	376	376	0	0	**0**	0.008		
chrM.pan.fa.6626ff2.7748b33.72587dd.smooth	1395	1887	-	2	456	456	0	0	**0**	0.007		
chr6.pan.fa.a2fb268.2ff309f.1300c8a.smooth_comp6_rem_nroot	1700	2283	1	4	586	584	2	0	**0**	0.008		
chr6.C4.gfa	1748	2366	-	2	584	582	2	0	**0**	0.009		
LPA.gfa	3751	5195	-	2	1305	1305	0	0	**0.01**	0.018		
chr6.pan.fa.a2fb268.2ff309f.1300c8a.smooth_comp5_nroot	5475	7374	-	2	1896	1894	2	0	**0.01**	0.025		
cerevisiae.fa.gz.d1a145e.417fcdf.7493449.smooth.final_comp11_rem_nroot	54721	73869	3	9	18172	18162	10	0.4	**0.19**	0.27		
cerevisiae.fa.gz.d1a145e.417fcdf.7493449.smooth.final_comp4_nroot	72349	97855	-	7	23907	23898	9	0.6	**0.27**	0.343		
cerevisiae.fa.gz.d1a145e.417fcdf.7493449.smooth.final_comp7_rem_nroot	216342	292679	7	55	71457	71395	62	2	3.41	**1.172**		
chr19.pan.fa.a2fb268.e820cd3.9ea71d8.smooth_comp14_rem_nroot	2858719	3985788	786	458	904944	904804	140	35.8	325.85	**17.608**		
chr6.pan.fa.a2fb268.2ff309f.1300c8a.smooth_comp1_rem_nroot	4906290	6856764	280	606	1555547	1555411	136	70.1	847.49	**29.743**		
Synth1	8091	10133	-	200	4662	4453	209	0	0.68	**0.04**		
Synth2	173879	219006	-	152	102585	102421	164	1.5	12.11	**0.978**		
Synth3	27886	35086	-	7	16542	16536	6	0.2	**0.13**	0.149		
Synth4	37105	47488	-	7	5041	5035	6	0.3	**0.08**	0.14		
Synth5	834549	1071462	-	157	467294	467110	184	8.7	58.37	**5.485**		
Synth6	211938	273297	-	3	119602	119601	1	2	**0.68**	1.355		

7.2 构建变异泛基因组图的根（源点）和末端（汇点）

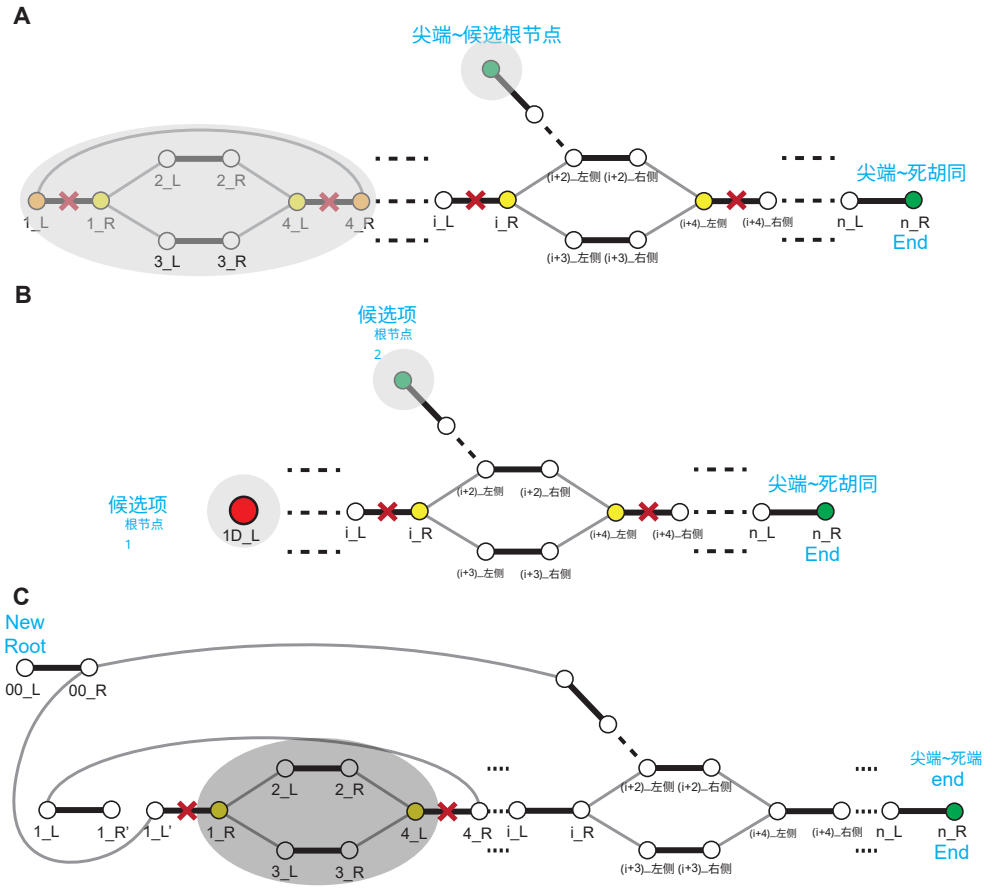

图 5 A)图中展示了一个没有工作根的图，这意味着图中存在环结构、无法到达图中所有节点的候选根，或两者兼有。请注意，$(i+1)\,L$ 和 $n\ R$ 都是尖端，但第一个是候选根，因为它的入度为=0；而第二个是死端，因为它的出度为=0。**B)** 在双边图中，首先计算强连通分量（SCC），并构建凝聚图，将每个 SCC 作为超节点。随后，入度为=0的超节点（如 $1D\ L$ ）和普通节点（如 $(i+1)\ L$ ）都属于候选根。**C)** 添加一条端点为 $00\ L$、$00\ R$ 的人工黑边，并按照前文针对凝聚图中超节点对应的 SCC 所述的过程，将 $00\ R$ 通过灰边连接到凝聚图的每个候选根。这样一来，$00\ L$ 就成为该图的工作根。请注意，此过程可能会改变图的 snarl 集，例如，snarl $(i\ R, (i+4)\ L)$不再是 snarl，但所有超泡，如 $(1\ R, 4\ L)$，及其边界节点均不受影响。

在一般图 G 中创建单一源点的概念性策略是将图转换为有向无环图 G'，找出其中所有入度为零的 N^0 节点，然后

添加一个单独的节点，并向每个 N^0节点添加出边。构造单一汇点也需要采取类似的方法，不同之处在于，新汇点节点要从 G'中所有出度为零的节点接收入边。重要的是，为了实现我们的目的， G 中的每个超泡都必须存在于 G'中；也就是说，如果图 G 中的超泡集合为 $UB(G)$，那么 $UB(G) \subseteq UB(G')$。

以下步骤描述了该构造算法的细节。

1. 将 G 变换为有向无环图。任意图 G都可以通过将 G中的每个强连通分量收缩为单个节点而变换为有向无环图。变换后的图 G_C 称为 G的凝缩图。识别所有强连通分量、收缩这些分量，并保留G_C 中的节点与 G 之间的映射 f ，例如借助 Tarjan 算法 [21]，其复杂度为 $O(m+n)$。然而，由于若 G 的至少一个强连通分量包含一个超泡，则 $UB(G) \supset UB(G_C)$，我们使用 G_C 作为中介，以识别 G 中将在变换后的图里连接到源点的所有强连通分量（见图 5A）。2. 识别 G_C中所有入度为零的节点 N_C^0。这一简单操作的复杂度为$O(n+m)$，并识别出所有必须连接到源点的强连通分量（见图 5B）。3. 将 N_C^0 中的每个强连通分量连接到源点。注意，对于 N_C^0 中映射到单个节点 $v \in G$的每个节点，可以通过添加一条从源点到 v的出灰边，将 v 直接连接到源点。由其余节点 $v \in N_C^0$ 所表示的强连通分量各自使用以下过程进行连接（见图 5C 和图 6）：(a) 若由 v 映射的强连通分量（即 $f(v)$）不包含$R-L$ 缠结（因此也不包含超泡），则可将源点连接到 $f(v)$中的任意（左侧）节点，其复杂度为 $O(1)$。(b) 否则，识别 $f(v)$的非嵌套 $R-L$缠结集合 S_{RL}，并选择黑边 e ，该边连接 S_{RL} 中任意缠结的右节点 r 与 $f(v)$中的某个节点 n 。创建两个新节点 r' 和 n' ，并将 e 拆分为 e_l 和 e_r ，使得 e_l 连接 n 和 r' ，且 e_r 连接 n' 和 r。用 e的标签标记 e_l 和 e_r ，并将源点连接到 n'。后一项操作，即创建 r' 和n'、拆分 e以及连接源点，其复杂度为 $O(1)$。

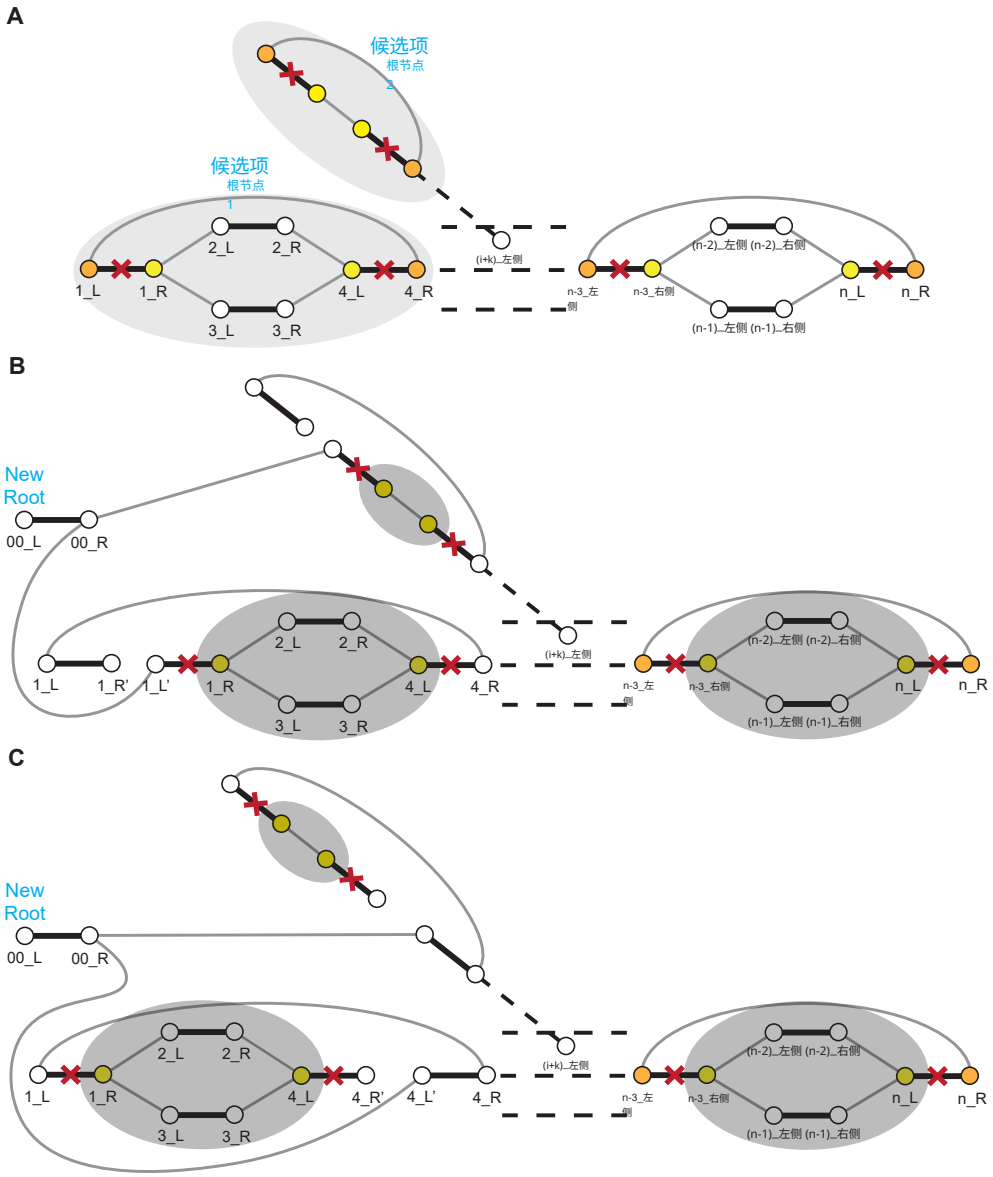

图6 A)没有任何候选根的图至少有一个强连通分量。这里我们有3个强连通分量，其中2个是候选根。**B)** 按照我们添加人工根的方法，此处黑边(i L,i R)和(1 L,1 R)被拆分。请注意，添加人工根后，所有超泡均不受影响。**C)**按照我们添加人工根的方法，此处黑边(($i+1$ L $i+$),( ))和(,) 1 R 4 L 4 R 被拆分。请再次注意，添加人工根后，所有超泡均不受影响。

一种判断 $R-L$ 缠结（r_i,l_i）是否嵌套在另一个$R-L$缠结（r_j,l_j） $i \neq j$中的方法，是考察从 r_j开始对 f（v）进行深度优先搜索时， r_j、 r_i、 l_i、 l_j的完成时间 t_f。首先，根据 $R-L$ 缠结的定义，对于任意$R-L$ 缠结（r_j,l_j），从 r_j开始进行深度优先搜索时，l_j 是对应深度优先搜索森林中 r_j的真后代，因此 t_f（r_j） $> t_f$（l_j）。此外，对于强连通分量 f（v）中的任意 $R-L$缠结（r_j,l_j），若从由（r_j,l_j）定义的子图之外的任意位置开始进行深度优先搜索，则 t_f（r_j） $> t_f$（l_j）。相反，在由（r_j,l_j）定义的子图内任意位置开始深度优先搜索（$\backslash r_j$即排除 r_j，但包含 l_j），会得到 t_f（r_j） $< t_f$（l_j）。因此，对于强连通分量中嵌套在另一个 $R-L$缠结（r_j,l_j） $i \neq j$中的任意 $R-L$缠结（r_i,l_i），从 r_i 开始进行深度优先搜索时，将有 t_f（r_j） $< t_f$（l_j）。反过来，对于嵌套在另一个 $R-L$缠结（r_j,l_j） $i \neq j$ 中的任意缠结 $R-L$缠结（r_i,l_i），若从 r_j开始进行深度优先搜索，则 r_i 和 l_i 是对应深度优先搜索森林中 r_j 的真后代，并且 t_f（r_j） $> t_f$（r_i） $> t_f$（l_i）。更具体地说，如果这些嵌套缠结位于同一个强连通分量中，那么对于任何从由深度优先搜索路径（从 r_j 到 r_i）定义的子图之外开始的深度优先搜索， f_t（r_j） $> f_t$（r_i）。因此，对于强连通分量 f（v）中的嵌套缠结，从任意 $R-L$缠结（r_i,l_i）的节点 r_i 开始进行深度优先搜索， $\in f$（v）可能出现两种情况：（i）对于所有 $R-L$缠结（r_j,l_j）， $\in f$（v） t_f（r_j） $> t_f$（l_j），此时缠结（r_i,l_i）没有嵌套在另一个 $R-L$缠结中；（ii）对于某个 $R-L$缠结（r_j,l_j）， $\in f$（v） t_f（r_j） $< t_f$（l_j），此时缠结（r_i,l_i）嵌套在 $R-L$缠结（r_j,l_j）中。情况（i）下，我们已识别出一个非嵌套缠结，可以继续执行构造算法。情况（ii）下，（r_i,l_i）可能嵌套在多个 $R-L$缠结中，它们都满足t_f（r_j） $< t_f$（l_j）。假设这类嵌套 $R-L$缠结的集合为 S_N（即，$\forall$（r_i,l_i） $\in S_N$，t_f（r_i） $< t_f$（l_i））， 则 S_N中最外层缠结的 R节点 r_i将具有最高的完成时间。识别集合 S_N 的复杂度为 O（$V_{f(v)}$），其中 $V_{f(v)}$ 是 f（v）中的节点数；同理，由于 S_N 中的 R 节点已按完成时间排序，识别 S_N 中完成时间最高的 R 节点也适用此结论。执行初始深度优先搜索的复杂度为 O（$E_{f(v)}+V_{f(v)}$），其中 $E_{f(v)}$ 是f（v）中的边数，因此整个图的复杂度为 O（$n+m$）。

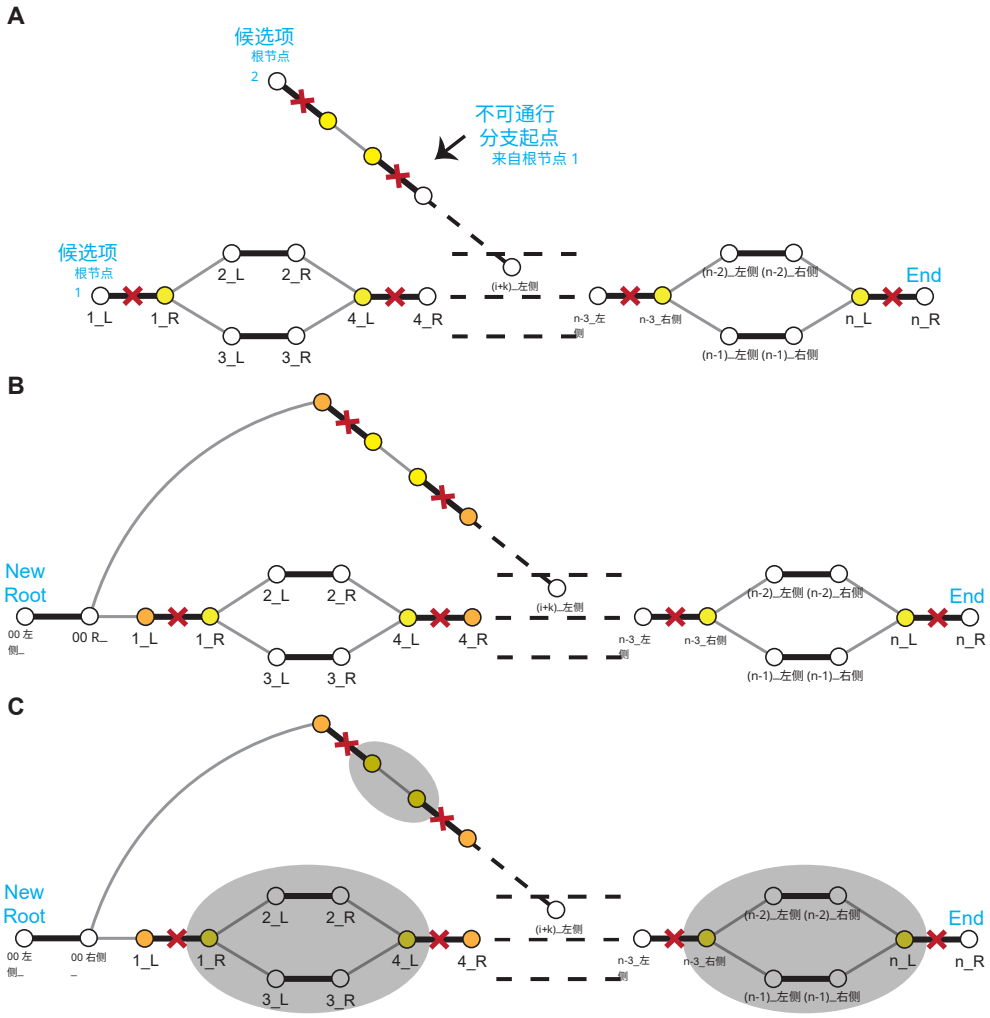

图7 A) 一个包含多个候选根节点的双边图，无法从其中任何一个根节点完整遍历，因为总会存在一条不可通行的分支。**B)**添加一条连接节点00 L和 00 R 的人工黑边，然后用灰边将 00 R 与双边图的每个候选根节点相连，最终得到 00 L 作为工作根。**C)** 添加人工根后，尽管缠结集发生了变化，但所有超泡（1 R, 4 L）、（i R, （$i+1$） L）和（（$\bar{n}-3$） R, $\underline{n}$ L）均不受影响。

7.3 计算环闭合节点及相应的灰色环闭合边

有向图中的环可以通过深度优先搜索（深度优先搜索）轻松检测。在标准的深度优先搜索中，节点分为未访问、位于递归栈中或已完全探索（通常分别记为白色、灰色和黑色），而Tarjan证明了 [21] 期间

深度优先搜索中的任意回边（从一个节点指向递归栈中某个灰色祖先的边）都表明存在一个环。后来，Tarjan 还证明了 [22] 上述方法对于包含嵌套环和重叠环的图同样适用，因为他证明了每个有向环都至少包含一条回边。Johnson 算法 [23], 能够完整枚举有向图中的所有简单初等环，但其时间复杂度 $O((V+E)(C+1))$与环的数量成正比 C，而环的数量可能相对于图中节点的数量呈指数级增长。然而，根据我们的方法以及引理3，我们只关注环闭合节点及其对应的（灰色）闭合边，而不关注显式的环路径。因此，我们采用 Tarjan 的方法 [21] ，沿着白色、灰色、黑色边进行遍历；该方法的完备性由 [22] 保证，因此在一次深度优先搜索执行中，我们可以在线性时间内 $O(V+E$标记给定双边图中每个环的所有环闭合节点及其对应的环闭合灰边，包括嵌套环中的环。

7.4 在线性时间内回答最近公共祖先查询

如第 3.2 节所述，在一般有向图中，无需枚举所有路径即可高效处理最近公共祖先查询。我们首先将所有强连通分量（SCC）收缩，以构建凝聚有向无环图；该图无环，并且保留可达性 [24]。在该有向无环图上，两个节点的最近公共祖先是任意一个极小公共祖先；也就是说，它是一个不存在同样能够到达这两个节点的后代的公共祖先，而这样的最近公共祖先可能构成一个集合，而非单个节点 [25]。如果不存在公共祖先，则最近公共祖先集合为空。为了针对每次查询返回一个节点，我们可以从构成平局处理规则的极小公共祖先连通分量中选择一个确定性代表节点。对于少量查询，在凝聚有向无环图上进行两次反向搜索并求交集即可；而对于大量查询，预先计算祖先信息（集合或稠密位集）可以实现快速查找。对于具有 $O(1)$次查询的全对方法，存在一些需要大量预处理的有向无环图专用算法：一种经典的次立方框架 [25], an $O(nm)$全对方法，在稀疏有向无环图上有效 [26], ；以及当前最佳的次立方上界 $O(n^{2.447})$ [27] ，但其实现更加困难。这里的 n 和 m 分别表示凝聚有向无环图的顶点数和边数。

为在树中寻找最近公共祖先，我们考虑三种经典方法。Harel 和 Tarjan [24]提出的最近公共祖先方法支持在线最近公共祖先查询，在经过 $O(n)$预处理后，可在 O时间内完成查询；但该方法实现起来相对困难。更简单的替代方案是倍增（跳跃指针）：该方法在 $O(n\log\ n)$时间内预处理 2^i-祖先，并通过先将深度调整至相同，再依据 [25]中的跳跃指针表，按二的递减次幂同时向上跳跃两个节点，从而在 $O(n\log\ n)$时间内回答每次 LCA 查询。欧拉游历 $+$区间最小值查询方法使用 ±1深度数组 [28] ，将最近公共祖先问题转化为欧拉深度序列上的区间最小值查询；该序列的相邻差值为 ±1 ，其预处理复杂度为 $O($ $n)$，查询复杂度为 $O(1)$。尽管 Harel-Tarjan 算法与欧拉游历 $+$区间最小值查询方法的渐近复杂度相同，我们选择了欧拉游历 $+$ 区间最小值查询方法，因为其实现更简单直接。我们没有实现 [29] 提出的节省空间的区间最小值查询方法。该方法是同一方法的一个更节省空间的变体；由于我们的关注点是运行时间目标，实现它会增加实现复杂度。

参考文献

[1] Abondio, P., Cilli, E., Luiselli, D.：《人类泛基因组学：分布式基因组参考的前景与挑战》。Life 13(6), 1360 (2023) https://doi.org/10. 3390/life13061360[2] Paten, B., Eizenga, J.M., Rosen, Y.M., Novak, A.M., Garrison, E., Hickey, G.：超气泡、超泡与仙人掌图。《计算生物学杂志》25(7), 649–663 (2018) https://doi.org/10.1089/cmb.2017.0251[3] Garrison, E., Sir´en, J., Novak, A.M., Hickey, G., Eizenga, J.M., Dawson, E.T., Jones, W., Garg, S., Markello, C., Lin, M.F., Paten, B., Durbin, R.：变异图工具包通过在参考序列中表示遗传变异改进读段比对。《自然·生物技术》36(9), 875–879 (2018) https://doi.org/10.1038/nbt.4227[4] Mwaniki, N., Garrison, E., Pisanti, N.：在泛基因组图中消除气泡。arXiv (2024) https://doi.org/10.48550/arxiv.2410.20932 2410.20932[5] Dabbaghie, F., Ebler, J., Marschall, T.：BubbleGun：枚举基因组图中的气泡和超气泡。《生物信息学》38(17), 4217–4219 (2022) https: //doi.org/10.1093/bioinformatics/btac448[6] Liao, X., Li, M., Zou, Y., Wu, F., Yi-Pan, Wang, J.：从头组装的当前挑战与解决方案。《定量生物学》7(2), 90–109 (2019) https: //doi.org/10.1007/s40484-019-0166-9[7] Zerbino, D.R., Birney, E.：Velvet：使用德布鲁因图进行从头短读段组装的算法。《基因组研究》18(5), 821–829 (2008) https://doi.org/ 10.1101/gr.074492.107[8] Bankevich, A., Nurk, S., Antipov, D., Gurevich, A.A., Dvorkin, M., Kulikov, A.S., Lesin, V.M., Nikolenko, S.I., Pham, S., Prjibelski, A.D., Pyshkin, A.V., Sirotkin, A.V., Vyahhi, N., Tesler, G., Alekseyev, M.A., Pevzner, P.A.：SPAdes：一种新的基因组组装算法及其在单细胞测序中的应用。《计算生物学杂志》19(5), 455–477 (2012) https://doi.org/10. 1089/cmb.2012.0021[9] Onodera, T., Sadakane, K., Shibuya, T.：检测组装图中的超气泡。arXiv (2013) https://doi.org/10.48550/arxiv.1307.7925 1307.7925[10] Paten, B., Earl, D., Nguyen, N., Diekhans, M., Zerbino, D., Haussler, D.：Cactus：基因组多序列比对算法。《基因组研究》21(9), 1512–1528 (2011) https://doi.org/10.1101/gr.123356.111[11] Wallis, W.D.：《图论及其应用》（J. A. Bondy 和 U. S. R. Murty）。《SIAM评论》21(3), 429–429 (1979) https://doi.org/10.1137/1021086

[12]Bang-Jensen, J., Gutin, G.Z.：《有向图：理论、算法与应用》。Springer 数学专著（2009）https://doi.org/10.1007/978-1-84800-998-1

[13]Romain, S., Dubois, S., Legeai, F., Lemaitre, C.：探究泛基因组图中倒位的拓扑模式。bioRxiv，2025–0314643331（2025）https://doi.org/10.1101/2025.03.14.643331

[14] Bender, M.A., Farach-Colton, M., Pemmasani, G., Skiena, S., Sumazin, P.：树和有向无环图中的最低公共祖先。《算法杂志》**57**(2), 75–94（2005）https://doi.org/10.1016/j.jalgor.2005.08.001

[15]Li, H.：用于测试序列到图比对的示例图与序列。https://doi.org/10.5281/zenodo.6617246 . https://doi.org/10.5281/zenodo.6617246

[16] Liao, W.-W., Asri, M., Ebler, J., Doerr, D., Haukness, M., Hickey, G., Lu, S., Lucas, J.K., Monlong, J., Abel, H.J., Buonaiuto, S., Chang, X.H., Cheng, H., Chu, J., Colonna, V., Eizenga, J.M., Feng, X., Fischer, C., Fulton, R.S., Garg, S., Groza, C., Guarracino, A., Harvey, W.T., Heumos, S., Howe, K., Jain, M., Lu, T.-Y., Markello, C., Martin, F.J., Mitchell, M.W., Munson, K.M., Mwaniki, M.N., Novak, A.M., Olsen, H.E., Pesout, T., Porubsky, D., Prins, P., Sibbesen, J.A., Tomlinson, C., Villani,F., Vollger, M.R., Consortium, H.P.R., Bourque, G., Chaisson, M.J., Flicek, P., Phillippy, A.M., Zook, J.M., Eichler, E.E., Haussler, D., Jarvis, E.D., Miga, K.H., Wang, T., Garrison, E., Marschall, T., Hall, I., Li, H., Paten, B.: 人类泛基因组参考序列草案 (2022) https://doi.org/10.1101/2022.07.09.499321[17] Garrison, E., Guarracino, A., Heumos, S., Villani, F., Bao, Z., Tattini, L., Hag- mann, J., Vorbrugg, S., Marco-Sola, S., Kubica, C., Ashbrook, D.G., Thorell, K., Rusholme-Pilcher, R.L., Liti, G., Rudbeck, E., Golicz, A.A., Nahnsen, S., Yang, Z., Mwaniki, M.N., Nobrega, F.L., Wu, Y., Chen, H., Ligt, J.d., Sud- mant, P.H., Huang, S., Weigel, D., Soranzo, N., Colonna, V., Williams, R.W., Prins, P.: 构建泛基因组图。《自然·方法》**21**(11), 2008–2012 (2024)https://doi.org/10.1038/s41592-024-02430-3[18] Guarracino, A., Heumos, S., Nahnsen, S., Prins, P., Garrison, E.: ODGI：理解泛基因组图。《生物信息学》**38**(13), 3319–3326 (2022) https://doi.org/10.1093/bioinformatics/btac308[19] Yue, J.-X., Li, J., Aigrain, L., Hallin, J., Persson, K., Oliver, K., Bergstr¨om, A., Coupland, P., Warringer, J., Lagomarsino, M.C., Fischer, G., Durbin, R., Liti, G.: 家养酵母与野生酵母之间基因组进化动态的对比。《自然·遗传学》**49**(6), 913–924 (2017)https://doi.org/10.1038/ng.3847[20] Wick, R.R., Schultz, M.B., Zobel, J., Holt, K.E.: Bandage：从头组装基因组的交互式可视化。《生物信息学》**31**(20), 3350–3352 (2015)

https://doi.org/10.1093/bioinformatics/btv383

[21] Tarjan, R.：深度优先搜索与线性图算法。载于：第12届年度开关与自动机理论研讨会（swat 1971），第114–121页（1971）。https://doi.org/10.1109/SWAT.1971.10[22] Tarjan, R.：有向图基本回路的枚举。《SIAM计算杂志》2(3)，211–216（1973）。https://doi.org/10.1137/0202017[23] Johnson, D.B.：《寻找有向图的所有基本回路》.pdf。《SIAM计算杂志》4(1)，77–84（1975）。https://doi.org/10.1137/0204007[24] Harel, D., Tarjan, R.E.：查找最近公共祖先的快速算法。《SIAM计算杂志》13(2)，338–355（1984）。https://doi.org/10.1137/ 0213024[25] Bender, M.A., Farach-Colton, M.：层级祖先问题的简化版。《理论计算机科学》321(1)，5–12（2004）。https://doi.org/10.1016/j.tcs.2003. 05.002[26] Czumaj, A., Kowaluk, M., Lingas, A.：在有向无环图中查找最近公共祖先的更快算法。《理论计算机科学》380(1-2)，37–46（2007）。https://doi.org/10.1016/j.tcs.2007.02.053[27] Grandoni, F., Italiano, G.F.，Lukasiewicz, A., Parotsidis, N., Uzna´nski, P.：有向无环图中的所有点对最近公共祖先：突破 $O(nˆ{2.5})$ 界限。arXiv（2020）。https://doi.org/10.48550/arxiv.2007.08914 2007.08914[28] Bender, M.A.,Farach-Colton, M.：重新探讨最近公共祖先问题。载于：Gonnet, G.H., Viola, A.（编），LATIN 2000：理论信息学，第88–94页。Springer，柏林、海德堡（2007）。https://doi.org/10.1007/10719839 9[29] Fischer, J., Heun, V.：静态数组区间最小值查询的空间高效预处理方案。《SIAM计算杂志》40(2)，465–492（2011）。https://doi.org/10.1137/090779759