arXiv:2601.08106v1 [cs.CG] 13 Jan 2026

带预测的德劳内三角剖分

Sergio Cabello* Timothy M. Chan† Panos Giannopoulos‡

2026年1月14日

摘要

我们研究计算几何中的带预测的算法，具体聚焦于计算二维德劳内三角剖分这一基本问题。给定一组 P 平面上的 n个点，以及一个作为德劳内三角剖分“预测”的三角剖分 G，当 G “接近” $DT(P)$时，我们希望利用 G更快地计算出正确的德劳内三角剖分德劳内三角剖分(P)。在不同的确定性和概率设定下，我们得到了一系列此类结果，包括以下内容：

1. 定义 D 为 G 中不属于 DT(P) 的边数。我们提出一种确定性算法，可在 $O(n + D\log^3 n)$ 时间内根据 G 计算 DT(P)；还提出一种随机化算法，可在$O(n + D\log n)$期望时间内完成计算，后者在 D方面是最优的。2. 令 R 为 DT(P) 中边的一个随机子集，其中每条边都以概率 ρ 独立选取。假设 G 是 P 的任意一个包含 R的三角剖分。我们提出一种算法，可在 $O(n\log\log n + n\log(1/\rho))$时间内，根据 G 计算 DT(P)，且该算法以高概率成功。3. 定义 $d$vio 为 P 中严格位于 G 某个三角形外接圆内的点的最大数量（如果 G 等于 DT(P)，则该数量为 0）。我们提出一种确定性算法，可在 $O(n\log^* n + n\log d\text{vio})$时间内，根据 G 计算 DT(P)。

我们还在类似的设定下，针对二维欧几里得最小生成树等相关问题取得了成果，并希望我们的工作能够开辟一条富有成果的未来研究方向。

*数学与物理学院，卢布尔雅那大学，斯洛文尼亚；数学、物理与力学研究所，斯洛文尼亚，电子邮箱：sergio.cabello@fmf.uni-lj.si

†西贝尔计算与数据科学学院，伊利诺伊大学厄巴纳-香槟分校，美国，电子邮箱：tmc@illinois.edu

‡计算机科学系，伦敦大学城市圣乔治学院，英国，电子邮箱：Panos.Giannopoulos@city.ac.uk

1 引言

1.1 迈向带预测的计算几何

带预测的算法 近年来受到了广泛关注。给定一个问题实例及其解的预测，我们希望设计一种算法来求解该问题，使得当预测“接近”正确解时，算法的表现“更好”。“更好”可以指运行时间更短、近似因子更优（对于优化问题），或竞争比更优（对于在线问题）。根据问题的不同，可以采用不同的“接近程度”度量；或者，我们也可以假设预测遵循某些概率模型。这一范式已应用于多个领域的问题，例如调度、经典图问题、资源分配、机制设计、设施选址和图搜索等；参见 [54] 中的综述，以及 [1]中定期更新的论文在线列表。

相比之下，带预测的计算几何 算法研究还十分稀少。本文旨在推动这一研究方向。我们主要关注如何利用预测来改善几何算法的运行时间。

由于经典计算几何中的许多基本问题，都可以看作是标准一维问题（如排序和搜索）在二维或更高维空间中的推广，因此我们从一维排序问题开始讨论。在这里，带预测的算法与历史悠久、研究充分的自适应排序 算法主题相关 [34]——如果给定的序列已经接近有序，我们就能更快地将其排序。衡量序列与有序状态之间距离的方法有多种。例如：

- 考虑 $D\mathrm{inv}$，即给定序列中的逆序对数量（也就是顺序错误的数对数量）。就这个参数而言，传统的插入排序运行时间为 $O(n + D\mathrm{inv})$，当 $D\mathrm{inv} = O(n)$ 时，其速度已经优于$O(n\log\ n)$ 算法。已知更快的算法，其运行时间为 $\Theta(n\log(1 + D\mathrm{inv}/n))$，这在比较模型下是最优的。

- 或者考虑 $D\mathrm{rem}$，即需要移除的最少元素数量（“错误”元素），使剩余列表有序。对于此情形，已知最优时间界为 $\Theta(n + D\mathrm{rem}\log n)$。

- 或者考虑 $D\mathrm{runs}$，即升序段的数量（也就是说，输入可以划分成的连续递增子序列的数量）。对于此情形，已知最优时间界为 $\Theta(n\log\ D\mathrm{runs})$。遗憾的是，即使 $D\mathrm{runs}$ 低至 $n^{0.1}$，在这种距离度量下也无法将复杂度降至 $n\log\ n$ 以下。

[34] 中的综述提供了详尽的背景介绍，并列出了算法文献中研究过的其他参数（篇幅颇长）。自适应排序仍在持续研究中（近期的一些随机示例包括 [4, 7, 55]），而近期的一些论文 [6, 49]则考察了带预测的排序的新模型。

1.2 二维德劳内三角剖分问题

我们能否在二维几何问题中获得类似的结果？本文选择计算几何中一个可以说是最基础的问题来开展这项研究：计算一组 P个平面上的点 [29]的德劳内三角剖分，记作德劳内三角剖分(P)。定义德劳内三角剖分(P)的一种方式是通过空圆性质：三个点 p、p'、$p'' \in P$ 构成德劳内三角形，当且仅当经过p、p'、p'' 的外接圆严格内部没有 P 中的任何点。（为简化起见，本文假设输入处于一般位置，例如，输入中不存在四个共圆的点。）

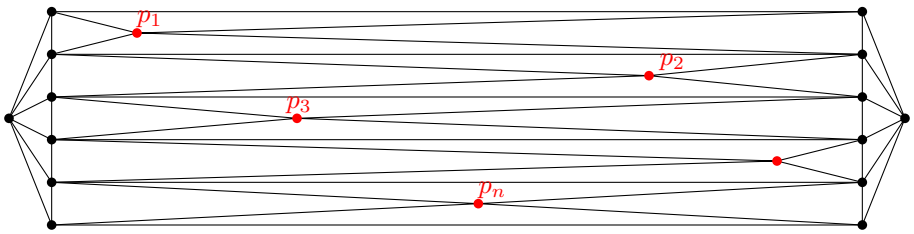

图1：对 $p_1, \ldots, p_n$ 的 x 坐标进行排序，可归约为计算图示三角剖分中 $O(n)$ 个顶点的德劳内三角剖分（假设通过缩放压缩 y 坐标）。

根据对偶性，德劳内三角剖分在组合意义上等价于沃罗诺伊图（采用欧几里得度量）。（在指定预测 G 或定义接近程度时，使用德劳内三角剖分比使用沃罗诺伊图稍微方便一些，因此我们的所有结果都将以德劳内三角剖分的形式表述。）德劳内三角剖分和沃罗诺伊图是最常见的几何结构之一，应用极其广泛，也是计算几何的核心基础。

在预测设定中，给定的不仅是输入点集 P，还有 P 的一个三角剖分 G（即将 P 的凸包分解为互不重叠的三角形，并使用 P 中的所有点作为顶点，但不使用斯坦纳点）。目标是设计一种算法，在 G 与 $DT(P)$ 之间的“距离”较小时，更快地计算 $DT(P)$。在传统设定中，已有几种标准的教科书算法可在 $O(n \log n)$ 时间内计算德劳内三角剖分，例如使用分治、平面扫描法和随机增量构造 [5, 8, 29, 36], ，其中随机增量构造在最坏情况下是最优的。即使没有接近性，P 的任意三角剖分 G 仍能提供一些关于输入的额外信息，但不足以将时间降至低于 $n \log n$：不难看出，通过归约一维排序，$\Omega(n \log n)$ 仍是一个下界（见图1）。当 G 接近时，我们希望做到优于 $n \log n$。正如我们将看到的，为二维德劳内三角剖分设计更快的自适应算法，在技术上比为一维排序设计这类算法更困难，因此也更有趣。

1.3 两个具体问题

我们可以用许多自然参数来衡量与德劳内三角剖分的距离，如表1所定义。其中一些定义（例如，$D, D_{\text{flip}}, D_{\text{cross}}$）可用于比较任意两个三角剖分，而另一些定义（例如，$D_{\text{vio}}, D_{\text{local}}$）则专用于德劳内三角剖分。对于所有这些参数，当且仅当 $G = DT(P)$ 时，其值为 0。这些参数之间的一些关系是显然的，例如，$D_{\text{cross}} \le O(nd_{\text{cross}})$ 和 $D_{\text{vio}} \le O(nd_{\text{vio}})$，并且已知

$$D_{\text{local}} \le D \le D_{\text{flip}} \le D_{\text{cross}}.$$

不等式 $D_{\text{local}} \le D$ 成立，是因为所有非局部德劳内边都不可能是真正的德劳内边，而最后一个不等式 $D_{\text{flip}} \le D_{\text{cross}}$ 并不平凡，已由 Hanke 等人 [38]证明。

存在一些例子，其中 D_{local} 是一个常数，而 D 是 $\Omega(n)$。非常粗略地说，在排序问题中，D_{local} 与 D_{runs}（升序段的数量）更为相似，而 D 与 D_{rem} 更为相似，D_{flip} 和 D_{cross} 则与 D_{inv} 更为相似。事实上，通过将排序归约到具有 $\Theta(D_{\text{local}})$ 个升序段的问题，或者等价地，归约到归并 $\Theta(D_{\text{local}})$ 个有序列表的问题（见图2），不难证明：就 D_{local}而言，计算德劳内三角剖分需要 $\Omega(n \log D_{\text{local}})$ 的下界。因此，D_{local} 可能不是该问题的理想参数（即使 D_{local} 小到 $n^{0.1}$，我们也无法超越 $n \log n$）。

参数	定义
D	G 中不属于德劳内三角剖分(P)的边数 （或等价地，德劳内三角剖分$DT(P)$中不属于 G的边数）
D_{flip}	将 G 转换为$DT(P)$所需的最少边翻转 次数 （在一次翻转中，与三角形 $\triangle pp'p''$ 和 $\triangle pp'p'''$ 相邻的边 pp' 会被替换为一条新边 $p''p'''$，假设 $pp''p'p'''$ 是一个凸四边形）
D_{cross}	G 与德劳内三角剖分$DT(P)$之间的边交叉数
d_{cross}	德劳内三角剖分$DT(P)$中被 G的一条边交叉的边的最大数量
D_{vio}	违反项数量，即（p'''，$\triangle pp'p''$），其中 $p''' \in P$和 $\triangle pp'p''$位于德劳内三角剖分(P)中使得 p'''严格位于经过 p，p'，p''的外接圆内
d_{vio}	严格位于.....内部的点的最大数量 经过 G中的一个三角形的外接圆
D_{local}	非局部德劳内边的数量，位于 G （即，边 pp' 与三角形 $\triangle pp'p''$ 和 $\triangle pp'p'''$ 在 G 中相邻，使得 p''' 位于经过 p、p'、p''的外接圆内）

表1：给定三角剖分 G与德劳内三角剖分$DT(P)$之间接近程度的不同度量。

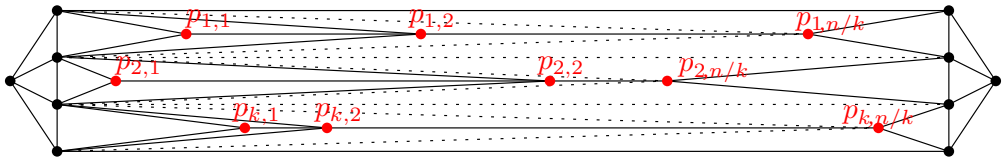

图2：归并 k x-排序列表$\langle p_{1,1}、.....、p_{1,n/k}\rangle$、.....、$\langle p_{k,1}、.....、p_{k,n/k}\rangle$可归结为计算所示三角剖分的 $O(n)$个顶点的德劳内三角剖分（假设进行重新缩放以压缩y坐标）。该三角剖分有 $D_{\text{local}} = \Theta(k)$条非局部德劳内边，以虚线表示。

在所有这些参数中，可以说最简单、最自然的参数是 D，即 G 中不属于$DT(P)$的“错误”边的数量（由于同一点集的所有三角剖分都具有相同数量的边，这也等于$DT(P)$中属于 G的边的数量）。为了开始研究针对德劳内三角剖分的带预测的算法，我们首先提出以下简单问题：

问题 *1*：运行时间以 n 和参数 D为函数进行分析时，最佳的二维德劳内三角剖分算法是什么？

这个简单问题事实证明颇具挑战性。根据上述不等式，如果能得到关于 D 的时间复杂度上界，就会自动得到关于 D_{flip} 和 D_{cross} 的上界。对于 d_{cross} 和 d_{vio}等其他参数，我们也可以提出类似的问题。

虽然问题 1 提供了一种对预测进行建模的方法，但误差通常具有概率性。因此，对于各种问题，带预测的算法有时会在概率模型下进行研究。为了定义一种简单的德劳内三角剖分预测概率模型，可以考虑抛掷一枚独立随机硬币，以确定$DT(P)$的一条边是否应被放入 G中（类似于 Cohen-Addad 等人 [28] 针对最大割问题研究的一种模型）。但对于我们的问题，有一个微妙之处：输入 G 应当是一个三角剖分，其中包含错误边（但不包含自交）。我们提出以下两阶段模型：首先，以给定概率 ρ将$DT(P)$的每条边独立地放入集合 R中；其次，将 R 补全为 P的一个三角剖分，但我们不对第二步的具体执行方式作任何假设，即 R 可以以对抗性地 的方式进行三角剖分（这使得该模型中的结果更具一般性）。

问题 *2*：在上述概率模型下，最佳的二维德劳内三角剖分算法是什么？其运行时间以 n和 ρ为参数进行分析。

在该模型中，“错误”边的期望数量 D 上界为 $(1-\rho)m$，其中 $m \le 3n-6$是 P的任意三角剖分中的边数。我们希望，相较于以参数 D为依据应用最坏情况算法，在概率模型下或许能更快地解决这个问题。

我们也可以考虑其他概率模型。例如，一种变体是在德劳内三角剖分(P)中随机选取一条边；如果它的两个相邻三角形组成一个凸四边形，就进行翻转（否则不做任何操作），然后重复 $(1-\rho)m$ 步。在该模型中，翻转距离 D_{flip} 保证至多为 $(1-\rho)m$；相比之下，上述模型允许翻转距离可能更大，因此从某种意义上说更强。

1.4 我们的贡献

我们获得了一系列成果，回答了问题1和问题2及相关问题：

1. 我们提出了几种不同的算法，用于从 G 计算 德劳内三角剖分(P)，其运行时间取决于参数 D。一种确定性算法的运行时间为 $O(n + D\log^4 n)$，另一种确定性算法的运行时间为 $O(n + D\log^3 n)$。一种随机化算法的运行时间为 $O((n + D\log n)\log^* n)$，其中 $\log^*$ 是迭代对数函数（增长极其缓慢）。我们最后提出的随机化算法消除了 $\log^* n$ 因子，期望运行时间为 $O(n+D\log n)$，这是最优的，因为通过从排序 D 个数归约，可得出一个 $\Omega(\max\{n, D\log D\}) = \Omega(n + D\log n)$ 下界（我们只需将图1中的配置嵌入到一个包含 $\Theta(D)$ 个点的更大德劳内三角剖分中即可）。因此，我们已经彻底回答了问题 1（但仍需使用随机化）。我们的所有算法都不需要事先知道 D 的值。

值得注意的是，即使“错误”边的数量 D 多达 $n/\log n$，我们的最优随机化算法的运行时间仍为线性。

一个直接推论是，该问题可以在 $O(n+D_{\text{flip}}\log n)$ 期望时间内求解——只要翻转距离至多为 $n/\log n$，我们就能实现线性运行时间。

2. 对于问题 2 中的概率模型，我们得到另一种算法，可在 $O(n\log\log n + n\log(1/\rho))$ 时间内，以高概率根据 G 计算 德劳内三角剖分(P)（该算法始终正确）。对于任意常数 ρ，运行时间因此几乎是线性的，只有较小的 $\log\log n$ 项除外。与我们的 $O(n + D\log n)$ 算法相比，这一结果尤为显著，因为在这种概率设置下， D 预计接近 $(1-\rho)m$。例如，如果我们设定 $\rho = 0.1$（甚至设为 $\rho = 1/\log n$），即使大多数边都是错误的，仍然可以显著优于 $n\log n$！

对于前面提到的随机翻转模型，我们也能得到类似的结果。

3. 我们还获得了对其他参数敏感的结果。例如，就 d_{vio} 而言，我们描述了一种确定性算法，可以在 G 中以 $O(n\log^* n + n\log d_{\text{vio}})$ 的时间计算 德劳内三角剖分(P)。请注意，存在一些示例，其中 $d_{\text{vio}} = 1$，但 $D = \Omega(n)$（也就是说，一部分边是错误的！），因此我们的 D 敏感算法无法直接使用。参数 d_{vio} 与文献中一种称为阶-k 德劳内三角剖分[1] 的概念有关 [37]（其动机与我们的动机大相径庭）：当且仅当 $d_{\text{vio}} \le k$ 时，三角剖分 G 才是“阶-k 德劳内三角剖分”。我们的结果带来了以下有趣的推论：给定一个阶-$O(1)$德劳内三角剖分，我们可以在近线性（$O(n\log^* n)$）时间内恢复实际德劳内三角剖分！

4. 类似地，我们描述了一种 $O(n\log^* n + n\log d_{\text{cross}})$ 时间的确定性算法，可以根据 G 计算 德劳内三角剖分(P)。同样，存在一些示例，其中 $d_{\text{cross}} = 1$，但 $D = \Omega(n)$。尽管参数 d_{cross} 起初看起来可能并不十分自然，但事实证明，它是我们以优雅且统一的方式获得上述第 2 项和第 3 项的关键：不难证明，在该概率模型中， $d_{\text{cross}} = O((1/\rho)\log n)$ 以高概率成立，而且我们还能够证明 $d_{\text{cross}} \le O(d_{\text{vio}}^2)$ 始终成立。

5. 尽管我们一直专注于德劳内三角剖分问题，但对于一些相关问题，我们也可以获得类似的结果。例如，考虑在二维中计算一组 P个 n 点的欧几里得最小生成树 *EMST*(P) 的问题。给定一棵生成树 T，我们可以在 T中从 $O((n+D_{\text{emst}}\log n)\log^* n)$ 的期望时间内计算*EMST*(P)，其中 D_{emst} 是 T 中不属于 *EMST*(P) 的 “ 错误 ” 边的数量。事实上，这一结果以及最初使用生成树的思想，将对最终设计我们的 $O(n + D\log n)$ 时间随机化算法来计算德劳内三角剖分发挥关键作用。

对于概率模型中的欧几里得最小生成树，或从其他参数的角度来看，我们也得到了类似的结果；对于计算其他相关几何图（例如平面中的相对邻域图和 Gabriel 图）也是如此 [40]。对于计算三维中处于凸位置的n 个点的凸包问题，我们也得到了类似的结果（通过标准的提升映射，二维中的德劳内三角剖分会映射为三维中的凸包）——凸位置这一假设是必要的，因为可以证明，即使已知正确的凸包，判断 $n/2$ 个点是否都位于另外 $n/2$个点的凸包内部，也需要 $\Omega(n\log n)$ 的下界。

[1]这个名称在某种程度上具有误导性，因为它并不等价于对偶中的*order-k 沃罗诺伊图* 。order-k 德劳内三角剖分的所有候选三角形的集合确实对应于order-k 沃罗诺伊图的顶点，但一个order-k 德劳内三角剖分并不唯一，它由候选三角形的一个子集构成。

1.5 计算几何中的相关工作

据我们所知，计算几何领域此前没有研究明确探讨带预测的算法这一范式，只有 Cabello 和 Giannopoulos 最近发表的一篇关于在线几何搜索问题的论文 [10]（其中利用预测来获得竞争比界）。与之隐含相关的是自适应 数据结构，这类数据结构已在计算几何领域得到研究；例如，可参见 Iacono 和 Langerman 关于“近邻”平面点定位的研究 [39]（当查询者在答案附近有一个指针时，可以更快地回答查询）。

我们的工作与几何结构的验证 相关，也可视为这类工作的推广：如何使用比计算答案的算法更简单或更快的算法，检查给定答案是否确实正确。20 世纪 90 年代，受算法工程的推动，这一研究方向受到了计算几何学家的广泛关注 [31, 52, 53], 。对于德劳内三角剖分问题，只需检查所有边是否均为局部德劳内边，就可以在线性时间内轻松验证 G 是否等于德劳内三角剖分(G)。验证正确性对应于 $D = 0$这一情形，而我们的结果提供了更强大的扩展：我们可以在线性时间内纠正错误（最多可纠正 $n/\log n$ 条错误边）。从纠正输出中的错误 这一角度看待我们的结果，也能进一步说明其意义，即使读者未必对学习增强算法感兴趣。

一些近期工作（例如 [33] ）研究了在计算过程本身容许误差的模型下的几何算法（如存在噪声或故障比较），但我们的工作似乎与这一研究方向正交。另一方面，计算几何领域此前已有大量工作研究能够刻画输入中的错误或不精确性的模型。例如，Löffler 和 Snoeyink 的一项研究 [48],将不精确输入点建模为一个范围，具体而言是一个单位圆盘（其圆心表示该点的估计位置）；研究表明，可以在 $O(n\log n)$ 时间内预处理平面上的 n 个互不相交的单位圆盘，从而在给定位于这些圆盘中的 n 个实际点后，能够在 $O(n)$ 时间内计算它们的德劳内三角剖分。对于不同类型的区域和不同的问题，也已得到类似结果。我们的模型提供了另一种处理不精确输入的方式：在预处理阶段，我们只需计算估计点的德劳内三角剖分 G；当实际点 P 揭示后，我们便可以使用我们的算法进行纠正 G，得到德劳内三角剖分(P)。[2] 我们模型的一个优点是简单：无需为每个输入点提供任何几何范围或分布；相反，精度由一个参数 D来概括。

即便不考虑与算法范式或应对输入/输出错误相关的问题，问题1本身作为一个经典的计算几何问题也颇具吸引力。文献中已有不少论文探讨了在何种条件下能够以线性或近乎线性的时间计算德劳内三角剖分。例如，Devillers [30] 证明，给定二维点集的欧几里得最小生成树，可以在 $O(n\log^* n)$期望时间内计算其德劳内三角剖分（这一结果后来被改进为确定性线性时间 [24]）；两个德劳内三角剖分德劳内三角剖分(P_1)和德劳内三角剖分(P_2)可以在线性时间内合并为德劳内三角剖分($P_1 \cup P_2$) [42]；反过来，一个德劳内三角剖分德劳内三角剖分($P_1 \cup P_2$)也可以在线性时间内拆分为德劳内三角剖分(P_1)和德劳内三角剖分(P_2)（由 Chazelle 等人提出 [16]）；凸多边形顶点的德劳内三角剖分可以在线性时间内计算（由 Aggarwal 等人提出 [2, 22]）；在字 RAM 模型中，整数坐标二维点的德劳内三角剖分可以在与排序相同的时间复杂度内计算（由 Buchin 和 Mulzer 提出 [9]）；等等。Löffler 和 Mulzer 的一项重要研究 [47] 表明，二维德劳内三角剖分与四叉树构造及相关问题等价等价，从而统一了其中许多结果。尽管其中一些工具稍后会派上用场，但目前尚不清楚如何利用它们解决问题1和问题2。

[2]一个技术问题是，将 G 重新嵌入实际点集 P时可能会出现自交，但我们的算法中至少有一种即使在 G 存在自交时也能正常工作（见推论 3.5）。

1.6 我们的技术

虽然我们认为自己的主要贡献在于概念层面（为计算几何开辟了一个新的研究方向），但在我们看来，我们的算法在技术上也颇具意义。事实证明，这些解法比一维自适应排序算法或教科书中常见的德劳内三角剖分算法要困难得多：

- 我们的确定性 $O(n+D\log^4 n)$时间算法（见附录 A）巧妙地利用了动态几何数据结构。

- 我们的确定性 $O(n + D\log^3 n)$时间算法（见第 2 节）采用给定图中的平面图分隔器或 r-划分 G （对于经典计算几何问题，基于采样的 Clarkson–Shor 式分治比平面图分隔器更常用）。

- 我们的随机化 $O((n + D\log\ n)\log^* n)$时间算法（见第 3 节）是通过改造 Devillers 提出的上述随机化 $O(n\log^* n)$时间算法得到的，该算法利用欧几里得最小生成树计算德洛内三角剖分 [30] （而该算法又基于 Seidel [56] 以及 Clarkson、Tarjan 和 Van Wyck [27] 提出的、用于多边形三角剖分问题的随机化$O(n\log^* n)$时间算法）。这种改造基本上是直接了当的，但我们的贡献在于意识到这些随机化 $O(n\log^* n)$时间算法首先就与此相关。

- 我们最终的随机化 $O(n + D\log n)$ 时间算法（见第 4 节）通过采用更复杂的分治方法，消除了额外的 $\log^* n$因子。该方法涉及平面图分隔器或r-划分，这次应用于实际德洛内三角剖分，而非输入图 G。（有趣的是，某些输出敏感凸包算法 [13, 32] 中的思想也证明很有用。）

- 对于我们的确定性 $O(n\log^* n + n\log\ d\text{cross})$ 时间算法（见第 5 节），如前所述，它既是我们在概率模型中取得结果的关键，也是我们 $d$vio-敏感结果的关键，我们重新采用给定图 G中的 r-划分。$\log^* n$ 因子的由来颇为有趣：它并非源于使用 Seidel 等人已知的随机化 $O(n\log^* n)$算法，而是来自一种原创的递归方法。尽管我们的 D- 敏感算法不能直接导出 $d$cross- 敏感结果，但它最终会成为最终算法中的一个重要组成部分。

1.7 开放性问题

我们的工作还留下了许多有趣的问题（我们认为这是件好事，因为我们的目标是激励人们在这一研究方向上继续探索）：

- 能否在我们的一些结果中消除额外的 $\log^* n$ 因子?

- 我们关于概率模型的结果中，额外的 $\log\log\ n$ 因子能否去掉?

- 能否得到关于 D翻转次数的更优时间界，而不是 $O(n + D$翻转次数 $\log n)$? 例如，是否存在一种德劳内三角剖分算法，其运行时间为 $O(n + D$翻转次数$)$?

- 同样，是否存在一种德劳内三角剖分算法，能在 $O(n + D\text{cross})$ 时间内运行? 或者更好的是，在 $O(n\log(1 + D\text{cross}/n))$ 时间内运行? 又或者能在 $O(n + D\text{vio})$ 或 $O(n\log(1 + D\text{vio}/n))$ 时间内运行?

更一般地说，我们希望这项工作能促使人们研究预测的其他模型在德劳内三角剖分问题中的应用，或研究这些模型在计算几何及其他领域的基本问题中的类似应用。

2 确定性 $O(n + D\log^3 n)$ 算法

我们首先介绍一种确定性算法，可在 $O(n + D\log^3 n)$ 时间内计算德劳内三角剖分。该算法使用平面图分隔器，或者更一般地，使用输入三角剖分 G 上的 r-划分。（附录 A 中给出了另一种确定性算法；它实际上是我们最先发现的算法。虽然速度稍慢，但不需要使用分隔器。）

一个 r-划分 是指将具有 G 个 n 顶点的平面图划分为 G 个 $O(n/r)$ 个区域（由边诱导的子图），使得每个区域至多包含 $O(r)$ 个顶点，以及至多 $O(\sqrt{r})$个边界顶点，即被多个区域共享的顶点。可以在 r 时间内计算一个 $O(n)$ 划分；例如，可参见 Federickson 的经典结果 [35]，或 Klein、Mozes 和 Sommer 近期改进的结果 [44]。请注意，区域可能带有孔洞（即不属于 G的面的面）。我们还可以假设每个区域都是连通的。

定理 2.1。 给定一个由 P 个 n 平面上的点构成的点集的三角剖分 G，可以在 $O(n + D\log^3 n)$ 确定性时间内构造德劳内三角剖分 $DT(P)$，其中 D 是 G 中不属于 $DT(P)$的边的（未知）数量。

证明。 我们的算法按以下步骤进行：

1. 首先，我们用 $O(n)$ 时间计算输入三角剖分 G的对偶图的一个 t-划分 Γ，其中t 稍后确定。Γ 的每个区域 γ 都是由 $O(t)$ 个三角形组成的连通集合。令 V_γ 表示 γ 中所有三角形的顶点集合，B_γ 表示围绕 γ 的所有三角形的集合（这里，使用边界三角形比使用边界边更方便）。于是，$|V_\gamma| = O(t)$且$|B_\gamma| = O(\sqrt{t})$。此外，令 B 为 Γ 中所有边界三角形的集合，V_B 为 B中各边的顶点集合。则 $|V_B| = O(n/\sqrt{t})$。

2. 我们计算 点集 V_B 的德劳内三角剖分 $DT(V_B)$，所需时间为 $O(|V_B|\log |V_B|) \le O((n/\sqrt{t})\log n)$.

3. 我们称区域 γ 中的 Γ 为良好区域，如果：(i) B_γ 的所有边界三角形都属于 $DT(V_B)$；并且 (ii)，γ 及其直接邻域中的三角形均为局部德劳内（即，对于 $\triangle p_1p_2p_3$ 和 $\triangle p_1p_2p_4$ 中任意一对相邻三角形 G，且至少有一个位于 γ 内部（另一个可以位于内部，也可以是边界三角形），都有 p_4 位于经过 p_1、p_2、p_3的外接圆之外）；我们称其为不良区域，否则。令

$$V_{good} = \left(\bigcup_{\gamma \text{ is good}} V_\gamma\right) \setminus V_B \text{ and } V_{bad} = \left(\bigcup_{\gamma \text{ is bad}} V_\gamma\right) \setminus V_B.$$

注意，条件 (i) 可以使用标准数据结构进行测试，而条件 (ii) 可以在线性时间内进行测试。

4. 我们计算德劳内三角剖分$(V_B \cup V_{good})$。这可以通过修改德劳内三角剖分(V_B)来完成。具体来说，对于每个优良区域γ，我们用γ的三角形替换其内部的部分 B_γ（根据定义，B_γ 属于德劳内三角剖分(V_B)）。这样得到的三角剖分是局部德劳内的，因此必定与德劳内三角剖分$(V_B \cup V_{good})$一致（因为整个三角剖分满足局部德劳内性质，意味着它就是德劳内三角剖分 [29]）。这一步需要线性时间。

5. 我们计算德劳内三角剖分(V_{bad})。这一步将从头开始，以 $O(|V_{bad}|\log |V_{bad}|)$时间完成。对于每个坏区域 γ， G 内部 γ 或其边界三角形上至少有一条非德劳内边。因此，坏区域的数量至多为 $O(D)$，这意味着 $|V_{bad}| \leq O(Dt)$。所以，这一步需要$O(Dt\log\ n)$时间。

6. 最后，我们通过归并德劳内三角剖分的标准算法，在 线性时间内归并 德劳内三角剖分($V_B \cup V_{good}$)和 $DT(V_{bad})$。（ Chazelle [15], 以及之后的 Chan [11], 更一般地证明了， $\mathbb{R}^3$ 中的两个凸包可以在线性时间内归并，但 Kirkpatrick [42] 更早的结果已足以处理德劳内三角剖分的情形 $\mathbb{R}^2$。）

总运行时间为 $O(n + (n/\sqrt{t} + Dt)\log\ n)$。取 $t = \log^2 n$ 即可得到所需的 $O(n + D\log^3 n)$ 上界。 □

3 一种随机化 $O((n + D\log\ n)\log^* n)$ 算法

本节将介绍一种完全不同且更快的随机化算法，该算法根据参数 D计算德劳内三角剖分，其与最优结果仅相差一个 $\log^* n$ 因子。该算法改编自 Devillers [30], 提出的一种算法；该算法利用欧几里得最小生成树构造德劳内三角剖分，期望时间几乎为线性时间 ($O(n\log^* n)$)。更一般地，可以从德劳内三角剖分(P) 的任意最大度为$O(1)$的生成树，在 $O(n\log^* n)$ 期望时间内构造德劳内三角剖分。我们进一步推广了他的结果，证明即使给定的生成树包含一些错误边，甚至不再要求最大度条件，也同样可以达到这一结果。

Devillers 算法与 Seidel [56] 以及 Clarkson、Tarjan 和 Van Wyk [27], 提出的、较为知名的用于多边形三角剖分的 $O(n\log^* n)$ 随机化算法类似；这些算法是标准$O(n\log\ n)$ 随机增量算法或随机采样算法的变体 [20, 25, 26], ，但加快了点定位步骤。我们将介绍基于随机采样的算法。

在三维空间中处理会更方便，因为二维德劳内三角剖分和沃罗诺伊图可以映射为三维凸包和平面的下包络。我们先介绍一些标准术语:

对于一组平面 Π in $\mathbb{R}^3$，令 LE(Π) 表示 Π 的下包络，也就是位于 Π 中某个平面上、但其下方没有 Π 中任何平面的点集。LE(Π) 的垂直分解记为 Decomp(Π)，它将 LE(Π) 下方的空间划分为垂直的截断棱柱；这些棱柱向下无界，向上则由包含在 LE(Π) 中的三角形面围成。围成棱柱的三角形顶点位于 LE(Π) 上，并按某种规范规则确定，例如，将 LE(Π) 的每个面剖分为三角形，所用对角线均从该面上 x 坐标最小的点引出。（由于这些面是凸的，因此可以进行这样的三角剖分。）我们还需要对 LE(Π) 的无穷面进行“三角剖分”，也就是说，将它们划分为由三条边界定的区域。我们将这些棱柱称为单元。

对于每个点 $p = (\alpha, \beta) \in \mathbb{R}^2$，令 p^* 表示由 $z = -2\alpha x - 2\beta y + \alpha^2 + \beta^2$在 $\mathbb{R}^3$ 中定义的平面。平面 p^* 与抛物面 $z = -x^2 - y^2$ 在点 p处相切。对于任意点集 P，定义$P^* = \{p^* \mid p \in P\}$。

对于平面上的一组 P点， P^* 的下包络在 xy方向上的投影是 P的沃罗诺伊图（与德劳内三角剖分德劳内三角剖分 (P) 对偶）。下包络 (P^*) 的顶点和边对应于平面上 P 的沃罗诺伊图的顶点和边。由于我们假设处于一般位置，下包络(P^*)的每个顶点都位于 P^*中的三个平面上。分解(P^*)中一个单元在 xy方向上的投影是一个三角形，该三角形包含在一个沃罗诺伊单元内，并由三个沃罗诺伊顶点定义。

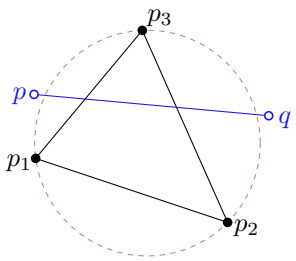

图3：引理3.1的证明。

对于顶点 $v = p_1^* \cap p_2^* \cap p_3^* \in \mathbb{R}^3$， v的冲突列表 记作 P_v^*，指的是 P^* 中严格位于 v下方的所有平面的列表。注意， p^* 属于冲突列表 P_v^*当且仅当 p严格位于经过 p_1、 p_2和 p_3的外接圆内部。

引理3.1。 设 p_1、 p_2、 p_3 是三角形 $\triangle$ 的顶点，且 p、 q 是满足 pq 与三角形 $\triangle$ 的内部相交的点。若 pq 是 $DT(\{p_1, p_2, p_3, p, q\})$ 的一条边，则 p^* 或 q^* 必须位于 $p_1^* \cap p_2^* \cap p_3^*$ 的冲突列表中。

证明。 见图3。若 p 和 q 不在 $\triangle$ 的外接圆内部，则 $\triangle$ 的边是德劳内三角剖分($\{p_1, p_2, p_3, p, q\}$) 的边。由于 pq 的端点不在 $\triangle$ 内部， pq 会与 $\triangle$ 的某条边相交。这将与德劳内三角剖分($\{p_1, p_2, p_3, p, q\}$) 的平面性相矛盾。 □

对于单元 $\tau \subset \mathbb{R}^3$， τ 的冲突列表记为 P_τ^*，它指 P^* 中与 τ 相交的所有平面的列表。注意， τ 的冲突列表正好是 τ 三个顶点的冲突列表的并集。

在介绍我们的主要算法之前，我们先给出以下引理。该引理利用给定的生成树 T 来加速冲突列表的计算。为处理 T 的最大度没有上界的情况，我们将使用加权样本，或者等价地，使用多重集的样本。

引理3.2。 设 P 是由 n 个点组成的集合， T 是 P的一棵生成树，其中每个顶点 $p \in P$ 周围的边均已按角度排序。设 $\widehat{P}$ 是一个多重集，其中每个点的重数为 $deg_T(p)$。令 R 为从 P 中随机选取的、大小为 r的（多重）子集，并令 $\widehat{R}$ 为去除重复元素后的集合 R。假设已给出下包络 $LE(R^*)$。那么，我们可以在 $O(n + D_T \log n)$期望时间内，计算 $Decomp(R^*)$ 中所有单元的冲突列表，其中 D_T 是 T 中不属于德劳内三角剖分 $DT(P)$的边数。

证明。 我们对 $DT(R)$ 进行预处理，以支持点定位 [43]。预处理需要 $O(r)$ 时间，而一次点定位查询可在 $O(\log r) \leq O(\log n)$ 时间内回答。

对于每个点 $p \in P$， 令 γ_p 表示其冲突列表包含 p^*的分解(R^*)顶点数。

我们首先介绍如何为每个点 $p \in P$ 计算包含 p 的三角形 $\triangle_p \in$ 德劳内三角剖分(R)。（严格来说，我们应在 R 和 P 中都添加一个足够大的包围三角形的 3 个顶点，以确保 $\triangle_p$ 始终存在，并处理分解 (R^*) 中 " 位于无穷远处"的顶点。）

为此，我们遍历生成树 T 的边。对于 T 的每条边 pq，已知 $\triangle_p$ 后，我们可以通过在 德劳内三角剖分(R) 中进行线性遍历到达 $\triangle_q$。但是，如果我们穿过 德劳内三角剖分(R) 中超过 $\log n$ 个三角形，就改为对 q 执行点定位查询。所需时间为 $O(\min\{c_{pq}, \log n\})$，其中 c_{pq} 是被 pq 穿过的 德劳内三角剖分(R) 中三角形的数量。遍历 T 中的所有边后，我们得到所有 $p \in P$ 的 $\triangle_p$。总时间为

$$O\left(\sum_{pq \in T} \min\{c_{pq}, \log n\}\right). \tag{1}$$

如果 pq 是 德劳内三角剖分(P) 的一条边，那么定理3.1意味着 c_{pq} 有界于 $\gamma_p + \gamma_q$。因此

$$\sum_{pq \in T \cap DT(P)} c_{pq} \;\le\; \sum_{p \in P} (\gamma_p \cdot \deg_T(p)),$$

其与冲突列表总大小成正比，其中包括重复项。根据 Clarkson和Shor 的分析， [26], 冲突列表总大小（包括重复项）的期望值为 $O(r \cdot |P|/r) \;=\; O(n)$， 这里我们利用了 $|P| = O(\sum_p \deg_T(p)) \;= O(n)$这一事实。

另一方面，

$$\sum_{pq \in T \setminus DT(P)} \min\{c_{pq}, \log n\} \;\le\; O(D_T \log n).$$

将这些结果代入式（1），即可得到所声称的期望时间界。

上文未考虑的一种情况是 $p \in R$。要针对给定的边 pq，从 p 开始沿着 q 行进，我们需要找到德劳内三角剖分德劳内三角剖分(R)中与 p 相邻且被射线 $\overrightarrow{pq}$穿过的三角形 $\triangle_{pq} \in$ 。为此，在预处理期间，对于每个 $p \in R$，我们可以将 T 在 p 周围的边的排序列表，与德劳内三角剖分(R)在 p周围的边的排序列表进行归并；归并的总成本为线性。之后，每个三角形 $\triangle_{pq}$ 都可以在常数时间内确定。

现在， $\triangle_p$ 映射到*LE*(P^*)的一个顶点，该顶点的冲突列表包含 p^*。通过图搜索，对于每个 $p \in P$，我们可以生成*LE*(P^*)的所有顶点，这些顶点的冲突列表包含 p^*。通过反演，我们得到*LE*(P^*)中所有顶点的冲突列表，进而得到*Decomp*(P^*)中所有单元的冲突列表。所有这些操作所需的额外时间与冲突列表的总大小成线性关系；根据Clarkson和Shor的界，该大小的期望值为 $O(n)$。□

定理 3.3。 给定平面上点集 P 的生成树 T，可以在 $O((n + D_T \log n) \log^* n)$ 期望时间内构造点集 P的德劳内三角剖分 $DT(P)$，其中 D_T是T 中不属于$DT(P)$的边数。

证明。 令 P为一个多重集，其中每个点 $p \in P$的重数为deg$_T$(p)；注意， $|P| = O(n)$。 定义随机样本序列 $R_1 \subset R_2 \subset \cdots \subset R_\ell \subset P$，其中 R_i的大小为 $|P|/s_i$，对应序列$\sim s_1, s_2, \ldots, s_\ell$，其取值将在稍后确定。令 R_i 为去除重复元素后的集合 R_i。

给定下包络(R_i^*)，我们描述如何计算下包络(R_{i+1}^*)。首先，根据 O 中的定理 3.2，我们用 $n + D_T \log n$ 的期望时间计算分解(R_i^*)中所有单元的冲突列表。对于分解(R_i^*)中的每个单元 $\tau \in$，我们使用标准下包络算法在 $O(|(R_{i+1}^*)_\tau| \log |(R_{i+1}^*)_\tau|)$ 时间内计算下包络$((R_{i+1}^*)_\tau) \cap \tau$。最后，我们将所有这些包络拼接起来，得到下包络($R_{i+1}^*$)。运行时间为

$$O\left(\sum_{\tau \in Decomp(R_i^*)} |(R_{i+1}^*)_\tau| \log |(R_{i+1}^*)_\tau| \right),$$

根据 Clarkson和Shor 的界 [26]，其期望值至多为

$$O(|\widehat{R}_i| \cdot (|\widehat{R}_{i+1}|/|\widehat{R}_i|) \log(|\widehat{R}_{i+1}|/|\widehat{R}_i|)) \;=\; O((n/s_{i+1}) \log(s_i/s_{i+1})).$$

总期望时间的上界为 $O(\ell(n + D_T \log\, n) + \sum_{i=1}^{\ell-1} (n/s_{i+1}) \log\, s_i)$。令 $s_1 = n$,$s_{i+1} = \lfloor \log\, s_i \rfloor$, ..., $s_\ell = 1$，其中 $\ell = O(\log^* n)$，即可得到该定理。 □

推论 3.4. 给定集合 P 中 n 个平面上的点的一种三角剖分 G，可以在 $O((n+D\log n)\log^* n)$ 期望时间内构造德劳内三角剖分 $DT(P)$，其中 D 是 G 中不属于 $DT(P)$ 的边数（未知）。

证明。令 T 为 G的一棵生成树。我们只需应用定理 3.3，并注意到 $D_T \le D$（因为 $T \subseteq G$，且 G 中每个顶点周围的边已经排好序）。 □

（Chazelle 和 Mulzer [18] 针对另一个问题得到了类似的期望时间上界 $O(n\log^* n + k\log n)$：在已知可能大得多的点集的德劳内三角剖分的情况下，计算具有 k 个连通分量的 n 条边子集的德劳内三角剖分。）

由于即使 T存在自交，定理 3.3 仍然成立，因此即使给定的 G 是可能存在自交的组合三角剖分，我们的结果也仍然成立。

推论 3.5。 给定平面上的 n 个点构成的集合 P，以及一个顶点集为 P、可能存在自交的组合三角剖分 G，可以在 $O((n+D\log n)\log^* n)$ 期望时间内构造德劳内三角剖分 $DT(P)$，其中 D 是 G 中不属于 $DT(P)$的边数（未知）。

证明。令 d_p 表示 G 中与 p 关联、但不是 德劳内三角剖分(P) 中三角形的三角形面数量。G 中每个顶点 p 的邻居几乎都已正确排序，至多有 O 个（d_p）错位元素；利用已知的自适应排序算法 [34]，可在 $O(\deg_G(p) + d_p\log d_p)$ 时间内完成排序。因此，总排序时间为 $O(n+\sum_p d_p\log n) = O(n+D\log n)$。

令 T 为 G 的任意一棵生成树。现在可以像之前一样将定理 3.3 应用于 T。 □

关联地，我们还得到了一个关于错误边数量敏感的欧几里得最小生成树算法：

推论 3.6。 给定一个由平面上的 n 个点组成的集合 P 的生成树 T，欧几里得最小生成树 $EMST(P)$ 可以在 $O((n+D_{\text{emst}}\log n)\log^* n)$ 期望时间内构造，其中 D_{emst} 是 T 中不属于 $EMST(P)$ 的边数（该数量未知）。

证明。由于欧几里得最小生成树(P)的最大度不超过 6，因此在 T 中，度至少为 7 的每个顶点 p 都至少与 $\deg_T(p) - 6$ 条不属于欧几里得最小生成树(P)的边相邻。这意味着 $\sum_p \max\{\deg_T(p)-6,0\} \le O(D_{\text{emst}})$。我们可以对每个顶点周围的 T边进行排序，总时间为 $O(\sum_p \deg_T(p)\log\deg_T(p)) \le O(n+\sum_p \max\{\deg_T(p)-6,0\}\log n) = O(n+D_{\text{emst}}\log n)$。现在可以应用定理 3.3，注意到$D_T \le D_{\text{emst}}$（因为欧几里得最小生成树$(P)$ $\subseteq$ 德劳内三角剖分(P)）。

计算出德劳内三角剖分(P)后，我们就可以通过计算德劳内三角剖分(P)的最小生成树，得到欧几里得最小生成树(P)。为此，我们可以使用 Matsui [51] 或 Cheriton 和 Tarjan [21]针对平面图的线性时间算法（无需使用 Karger、Klein 和 Tarjan [41] 提出的通用线性时间随机化算法）。 □

我们也可以得到关于相对邻域图 [40]的类似结果：pq 当以 p 和 q 为圆心、以 $\|p-q\|$ 为半径的两个圆盘的交集内部不含任何 P时，相对邻域图包含欧几里得最小生成树，因此是连通的。

推论 3.7。 给定平面上的一个 P n 点集，以及一个顶点集为 P 的连通平面直线图[3] G，相对邻域图 $RNG(P)$ 可在 $O((n+D_{\text{rng}}\log n)\log^* n)$ 期望时间内构造，其中 D_{rng} 是 G 中不属于 $RNG(P)$ 的边数（未知）。

[3]PSLG 是平面直线图的缩写；定义请参见附录 A。

证明。 令 T 为 G 的任意一棵生成树。我们只需应用定理 3.3，并注意到 $D_T \le D_{\text{rng}}$（因为 $T \subseteq G$ 且 相对邻域图(G) $\subseteq$ 德劳内三角剖分(G)，而 G 中围绕每个顶点的边已经排好序）。计算出 德劳内三角剖分(P) 后，我们便可在 [46] 线性时间内得到相对邻域图(G)。 □

类似的结果也适用于 Gabriel 图，或者更一般地，适用于任意 $\beta \in [1,2]$ [40] 的 β-骨架。

4 一种最优随机化 O($O(n + D\log n)$ 算法

本节将介绍一种关于参数 D 计算德劳内三角剖分的最优随机化算法，从而消除上一节算法中的额外 $\log^* n$ 因子。这一点很有趣，因为对于多边形三角剖分等相关问题，要去除 Clarkson、Tarjan 和 Van Wyk 的算法或 Seidel 算法中的额外 $\log^* n$ 因子 [27, 56] 是很困难的（参见 Chazelle 的线性时间多边形三角剖分算法 [14]）。

我们采用的新方法是：在定理 3.3 的证明中，经过固定轮数后提前停止迭代算法，此时冲突列表的大小已经足够小（例如为 $O(\log\log n)$），然后切换到另一种策略。对于这部分，我们在实际的德劳内三角剖分（随机子集的德劳内三角剖分）上使用平面图分隔器（t-划分）。

定理 4.1。 给定一组 G 平面上的 n 点的三角剖分 P，可以在 $O(n + D\log n)$ 期望时间内构造德劳内三角剖分 $DT(P)$，其中 D 是G 中不属于 $DT(P)$的边数（该数量未知）。

证明。 注意， G 中的三角形至多有 $O(D)$ 个不属于 德劳内三角剖分(P)；等价地，德劳内三角剖分(P) 中的三角形至多有 $O(D)$ 个不属于 G（因为 G和 德劳内三角剖分(P)的三角形数量相同）。

若一个三角形属于 德劳内三角剖分(P)，则称其为良好；若不属于 G，则称其为不良。

令 T 为 G 的任意生成树。由于 T中至多有 D 条边不属于 德劳内三角剖分(P)，我们可以应用定理 3.3 中的算法。不过，我们将在进行 ℓ 次迭代后停止该算法，并以非常数的 s_ℓ 取代 $s_\ell = 1$。计算出 下包络(R_ℓ^*) 后，我们将改用另一种策略来完成 下包络(P*) 的计算，如下所述。

首先，我们构造分解(R_ℓ^*) 的冲突列表，依据定理 3.2，在 $O(n + D\log n)$ 期望时间内完成 .

若某个单元的冲突列表大小超过 bs_ℓ，则称其为异常单元，其中参数 b 将在之后设定。我们直接为 Decomp(R_ℓ^*) 中的所有异常单元构造下包络 LE(P_τ^*)，耗时为 $\cap\,\tau$in $O(|P_\tau^*|\log |P_\tau^*|)\tau \in$。根据 Chazelle 和 Friedman 的“指数衰减引理” [17],，冲突列表大小超过 is_ℓ 的单元的期望数量为 $O(2^{-\Omega(i)}\cdot n/s_\ell)$，因此这一步的期望总成本至多为 $O(\sum_{i\ge b} 2^{-\Omega(i)}\cdot(n/s_\ell)\cdot(is_\ell)\log(is_\ell)) = O(2^{-\Omega(b)}\cdot n\log s_\ell)$。

对于某个参数 t，我们在 $O(n)$ 时间内，对 Decomp(R_ℓ^*) 的 xy-投影的平面图的对偶图计算一个 t-划分。由此得到 $O(n/(s_\ell t))$ 个区域，我们称之为超单元；每个超单元由 Decomp(R_ℓ^*) 中的 $O(t)$ 个单元构成，并有 $O(\sqrt{t})$ 面边界墙。

我们为每个边界墙 β 计算 下包络(P_β^*) $\cap\,\beta$。这需要 $O(n/(s_\ell t)\cdot\sqrt{t}\cdot(bs_\ell)\log(bs_\ell))$ 时间（因为异常单元已经处理完毕）。

当返回到$\mathbb{R}^2$时，下包络(P*)在其边界墙处的顶点会映射到德劳内三角剖分(P)中的边链。（过去一些输出敏感的凸包算法也曾将此类边链用于分治 [3, 13, 32],，但我们的用法不同。）这些边将德劳内三角剖分(P)划分为多个区域，每个区域对应一个超单元（不过一个超单元可能对应多个区域）。我们可以移除“尖刺”（即两侧都邻接同一区域的边）。对于每个这样的区域，我们检查其所有边界边是否都出现在 G中；如果是，再检查该区域内 G 的所有三角形是否

都是局部德劳内的（即，对于区域内 G 中的每一对相邻三角形 $\triangle p_1p_2p_3$ 和 $\triangle p_1p_2p_4$，都有 p_4 位于经过 p_1、p_2、p_3的外接圆之外）。如果这些检查均通过，我们就知道区域内 G 的所有三角形都是正确的德劳内三角形[4]，并将该区域标记为此类区域。所有这些检查的总耗时为线性时间。

对于每个超单元 γ，如果其对应的区域被标记为正确，那么我们就可以免费获得 LE(P^*) $\cap\,\gamma$ 中的所有顶点，并称该超单元 γ 为良好。否则，该超单元 γ 为不良，我们只需通过对 γ 中的每个单元 τ分别计算 LE(P^*_τ)$\cap\tau$，来计算 LE(P^*)$\cap\gamma$中的顶点，所需时间为 $O(t\cdot(bs_\ell)\log(bs_\ell))$（同样，异常单元已预先处理）。

注意，如果 $LE(P^*) \cap \gamma$中所有顶点对应的德劳内三角形都是良好的，那么超单元 γ必定是良好的。因此，每个不良超单元至少包含一个顶点，该顶点位于 $LE(P^*)$ 中，且其德劳内三角形是不良的。因此，不良超单元的数量至多为 $O(D)$。

由此可知，根据 $LE(P^*_\ell)$，我们可以在期望时间内计算出 $LE(P^*)$

$$O(n + D\log n + n/(s_\ell t)\cdot\sqrt{t}\cdot(bs_\ell)\log(bs_\ell) + D\cdot t\cdot(bs_\ell)\log(bs_\ell) + 2^{-\Omega(b)}\cdot n\log s_\ell).$$

例如，我们可以选择 $b = s_\ell$和 $t = s_\ell^4$，得到时间上界 $O(n + D\log n + Ds_\ell^{O(1)})$。

回想一下，定理 3 3 $O\ \ell\ n + D\quad n\ +$. 的证明中的算法，其期望时间上界为 $\sum_{i=1}^{\ell-1} n/s_{i+1}$ (log) ( ) log)。令 s_i , log , log log with, $s_1 = n\ s_2 =\quad n\ s_3 =\quad n\quad \ell = 3$ 则总时间上界为 $O(n + D\log\ n + D(\log\log\ n)^{O(1)}) = O(n + D\log\ n)$。□

当给定图 G 不是三角剖分，而只是一个 PSLG（可能不连通）且缺少边时，同样的结果也成立。

推论 4.2. 给定平面上的 n 个点组成的集合 P 以及顶点集为 P的 *PSLG* G，可以在 $O(n + D\log\ n)$ 期望时间内构造德劳内三角剖分 $DT(P)$，其中 D 重新定义为 $DT(G)$ 中不属于 G的边的（未知）数量。

证明。 缺失边的数量至多为 D，我们可以先轻松地在 G 中对 O的非三角形面进行三角剖分，所需时间为$D\log D$，然后运行定理 4.1 中的算法。 □

5 个 $O(n\log^* n + n\log\ d_{\text{cross}})$ 算法

本节中，我们介绍一种计算德劳内三角剖分的算法，该算法对参数 d_{cross}具有敏感性。与第 2 节中的确定性算法类似，该算法使用输入图 G中的平面图分隔器或 r-划分，但分治方式有所不同。困难在于，即使d_{cross} 很小，错误边的数量 D 仍可能很大。我们的思路是递归地计算每个区域的正确德劳内三角剖分，并将它们合并，得到一个错误边数量更少的新三角剖分（$O(n/\text{polylog}\ n)$）；随后，我们可以应用先前的 D-敏感算法来修正这个三角剖分。递归深度为 $O(\log^* n)$（但这与第 3 节随机化算法中的 $\log^* n$ 因子无关）。

我们从射线射击的已知子程序开始：

引理 5.1。 给定平面上总共有 n 个顶点的 h 个不相交简单多边形，我们可以在 $O(n + h^2\log^{O(1)} h)$ 时间内构建一个数据结构，从而可以在 $O(\log n)$ 时间内回答射线射击查询。

[4]如果一个三角剖分 G 在区域 γ内部满足局部德劳内性质，并且 γ 的所有边界边都是正确的德劳内边，那么 G 内部的部分 γ 就是正确的德劳内三角形。这一事实可以直接由约束德劳内三角剖分的已知性质推出，例如 [45], 将 $\partial\gamma$ 作为约束边时。

更一般地，给定参数 $\delta \in [0,1]$，我们可以在 $O(n + h^{2-\delta}\log^{O(1)} h)$时间内构建一个数据结构，从而可以在 $O(h^{\delta}\log n)$ 时间内回答射线射击查询。

证明。 第一部分归功于 Chen and Wang [19]。第二部分通过将输入划分为 h^{δ} 个子集、每个子集包含 $h^{1-\delta}$ 个多边形，并对每个子集应用该数据结构即可得出：预处理时间变为 $O(n+h^{\delta}\cdot(h^{1-\delta})^2\log^{O(1)} h)$，查询时间变为 $O(h^{\delta}\log n)$（因为射线射击是一个“可分解”问题——知道针对每个 h^{δ} 子集的查询答案，就能得出整体答案）。 □

引理 5.2。 给定一个含有 n 个顶点的连通平面直线图 G，以及一个由 Q 条线段组成的集合，我们可以在 $O(n + (|Q| + |A|)^{2-\delta}\log^{O(1)} n)$ 时间内计算出 G 中所有与 Q 相交的边所组成的子集 A，其中常数 $\delta > 0$ 足够小。

证明。 令 $r \in [1, |Q|]$ 为一个稍后选定的参数。令 R 为一个 $(1/r)$-网，其由 Q 组成，大小为 $O(r\log r)$，即一个子集 $R \subset Q$，使得任何与 $|Q|/r$ 条线段相交的线段，都必须与 Q 中至少一条线段相交。根据关于 $(1/r)$-网的已知结果， [50], 这需要 $O(|Q|r^{O(1)})$ 确定性时间（若允许随机化，则随机样本以较高概率已满足该性质）。

根据引理 5.1，我们对 G 进行预处理，耗时 $O(n)$，从而能够在 $O(\log n)$ 时间内回答射线射击查询（因为可以通过引入切割将 G 转化为树，而树可以通过遍历每条边两次视为一个简单多边形）。对于每条线段 $q \in R$，我们通过反复进行射线射击，在 $O(\log n)$ 乘以交叉数量的时间内找出 G 中所有与 q 交叉的边。这一步的总耗时至多为 $O(n + (r\log r)\cdot|A|\cdot\log n)$。

令 A_R 表示与 R 交叉的 G 的所有边组成的子集。接下来，我们对 $G - A_R$ 进行预处理，耗时 $O(n + |A|^{2-\delta}\log^{O(1)} n)$，从而能够根据引理 5.1，在 $O(|A|^{\delta}\log n)$ 时间内回答射线射击查询（因为 $G - A_R$ 至多有 $O(|A_R|) \le O(|A|)$ 个连通分量，并且可以视为 $O(|A|)$ 个简单多边形）。对于每条线段 $q \in Q$，我们通过反复进行射线射击，在 $O(|A|^{\delta}\log n)$ 乘以交叉数量的时间内找出 $G - A_R$ 中所有与 q 交叉的边。由于 R 是一个 $(1/r)$- 网， $G - A_R$ 中的每条边至多与 Q 的 $|Q|/r$ 条线段交叉；因此， $G - A_R$ 与 Q 之间的交叉总数至多为 $O(|A|\cdot|Q|/r)$。这一步的总耗时至多为 $O(n + |A|^{2-\delta}\log^{O(1)} n + |A|\cdot|Q|/r\cdot|A|^{\delta}\log n)$。令 $r = |Q|^{2\delta}$ 并选择足够小的 $\delta > 0$，即可得到该引理（在我们的应用中，无需优化 δ 的取值）。 □

引理 5.3。 任意一个含有 n 个顶点的平面图（不一定连通）至多有 $3n - 6$ 条边，且至少有 $3n - 6 - O(z)$ 条边，其中 z 是非三角形面中出现的边数与孤立顶点数之和。

证明。 通过添加 $O(z)$ 条边，可以将该图转化为完全三角剖分（即极大平面图），它恰好有 $3n - 6$ 条边。 □

定理 5.4。 给定一组 P 个平面上的 n 个点的三角剖分 G，可以在 $O(n\log^* n + n\log d_{\text{cross}})$ 确定性时间内构造德劳内三角剖分 $DT(P)$，其中 d_{cross} 是德劳内三角剖分 $DT(P)$ 中一条边可能与 G 的一条边相交的最大边数（未知）。

证明。 考虑一个区域 γ_0，它是 G 中一个连通集合的并，包含至多 μ 个三角形，且至多有 $c\sqrt{\mu}$ 条边界边，其中 c 是一个足够大的常数。（初始时， γ_0 是 P 的整个凸包。）令 V_{γ_0} 表示 γ_0 中所有三角形的顶点集合。我们描述一种递归算法来计算德劳内三角剖分(V_{γ_0}):

1. 首先，我们用 $O(\mu)$ 时间，计算 γ_0 内部 G 的对偶图的一个 t-划分 Γ，其中 t 将在后文确定。Γ 的每个区域 γ 都是一个连通集合的并，包含至多 t 个三角形，且至多有 $c\sqrt{t}$ 条边界边。令 V_γ 表示 γ 中所有三角形的顶点集合。

2. 对于每个区域 γ，我们递归地计算德劳内三角剖分(V_γ)。

3. 对于每个区域 γ，令 X_γ 为德劳内三角剖分(V_γ)中所有穿过边界 $\partial\gamma$ 的边组成的集合，并令 Y_γ 为德劳内三角剖分(V_γ)中完全包含于 γ 的所有边组成的集合。

我们通过一个间接论证来界定 Y_γ 中"错误"边的数量，具体考察德劳内三角剖分(P)：我们知道，穿过 $\partial\gamma$ 的德劳内三角剖分(P)的边和三角形数量至多为 $O(d_{\text{cross}}\sqrt{t})$。根据引理 5.3，完全包含于 γ 的德劳内三角剖分(P)的边数至少为 $3|V_\gamma| - O(d_{\text{cross}}\sqrt{t})$。由于德劳内三角剖分($P$)中完全包含于 γ 的所有边同时属于 Y_γ 和德劳内三角剖分(V_{γ_0})，因此 Y_γ 中至少有 $3|V_\gamma| - O(d_{\text{cross}}\sqrt{t})$ 条边属于德劳内三角剖分(V_{γ_0})。这意味着，Y_γ 中不属于德劳内三角剖分(V_{γ_0})的"错误"边至多有 $3|V_\gamma| - (3|V_\gamma| - O(d_{\text{cross}}\sqrt{t})) = O(d_{\text{cross}}\sqrt{t})$ 条。

特别地，我们还有 $|Y_\gamma| \geq 3|V_\gamma| - O(d_{\text{cross}}\sqrt{t})$，以及 $|X_\gamma| \leq 3|V_\gamma| - |Y_\gamma| = O(d_{\text{cross}}\sqrt{t})$。
我们可以将引理 5.2 应用于三角剖分 德劳内三角剖分(V_γ) 和 $\partial\gamma$ 中的线段，以 $O(t + (d_{\text{cross}}\sqrt{t})^{2-\delta}\log^{O(1)} t)$ 的运行时间计算出 X_γ。随后，我们可以通过图遍历，再用 $O(t)$ 的额外时间生成 Y_γ。这一步的总运行时间为 $O(\mu/t \cdot (t + (d_{\text{cross}}\sqrt{t})^{2-\delta}\log^{O(1)} t))$ $= O\left(\mu + d_{\text{cross}}^{2-\delta}\mu/t^{\delta/2}\right)\log^{O(1)} t$。

4. 对于每个区域 γ，我们扩展 Y_γ，得到 γ 的完整三角剖分 G'_γ：使用 V_γ 中的所有顶点（但不使用斯坦纳点），对 γ 内 $Y_\gamma \cup \partial\gamma$ 中的非三角形面进行三角剖分。由于 $\partial\gamma$ 的规模为 $O(\sqrt{t})$，且 $|Y_\gamma| \geq 3|V_\gamma| - O(d_{\text{cross}}\sqrt{t})$，非三角形面的复杂度为 $O(d_{\text{cross}}\sqrt{t})$，因此可在 $O(d_{\text{cross}}\sqrt{t}\log(d_{\text{cross}}\sqrt{t}))$ 时间内完成三角剖分。这部分开销可由其他步骤的开销吸收。

5. 我们合并所有 γ in Γ 对应的三角剖分 G'_γ，得到 $DT(V_{\gamma_0})$ 的完整三角剖分 G'：对 $Y_{\gamma_0} \cup \partial CH(V_{\gamma_0})$ 中的非三角形面进行三角剖分。这些非三角形面的总复杂度为 $O(\sqrt{\mu})$，因此可以在 $O(\sqrt{\mu}\log\mu)$ 时间内完成三角剖分。

6. 最后，应用我们之前的定理 2.1 算法，从 G 得到 $DT(V_{\gamma_0})$ G'.
G' 中不属于 $DT(V_{\gamma_0})$ 的"错误"边总数至多为 $O(\mu/t \cdot d_{\text{cross}}\sqrt{t} + \sqrt{\mu})$ $= O\left(d_{\text{cross}}\mu/\sqrt{t} + \sqrt{\mu}\right)$ $O\left(\mu + d_{\text{cross}}\mu/\sqrt{t} + \sqrt{\mu}\right)^3\mu$ $\log\mu$。因此，这一步需要 $(\log)$ 确定性时间。

总时间满足以下递推式：

$$T(\mu) = \max_{\substack{\mu_1,\mu_2,\ldots\leq t:\\ \mu_1+\mu_2+\cdots\leq\mu}} \left(\sum_i T(\mu_i) + O(\mu + (d_{\text{cross}}^{2-\delta}/t^{\delta/2} + d_{\text{cross}}/\sqrt{t})\mu\log^{O(1)}\mu)\right).$$

令 b 为稍后选定的参数。令 $t = \max\{\log^c\mu,\ b^c\}$，其中常数 c 充分大（取决于 δ），则有

$$T(\mu) = \max_{\substack{\mu_1,\mu_2,\ldots\leq\max\{\log^c\mu,\,b^c\}:\\ \mu_1+\mu_2+\cdots\leq\mu}} \left(\sum_i T(\mu_i) + O((1 + d_{\text{cross}}^2/b^2)\mu)\right).$$

结合基本情形 $\mu \leq b^c$ 时的朴素界 $T(\mu) \leq O(\mu\log\mu) \leq O(\mu\log b)$，递推式的解为 $T(\mu) = O((1 + d_{\text{cross}}^2/b^2)\mu\log^*\mu + \mu\log b)$。

如果已知 d_{cross}，我们只需设置 $b = d_{\text{cross}}$。否则，可以通过一个递增序列来“猜测”它的值。更准确地说，对于 $j = \lceil \log(\log^* \mu) \rceil$, $\lceil \log(\log^* \mu) \rceil + 1, \ldots$，我们设置 $b = 2^{2^j}$，并运行算法，将运行时间上限设为 $O(\mu 2^j)$。当 j 达到 $\max\{\lceil \log(\log^* \mu) \rceil, \lceil \log \log d_{\text{cross}} \rceil\}$ 时，算法至迟会成功；总运行时间由最后一次迭代所耗费的时间主导，即 $O(\mu \log^* \mu + \mu \log d_{\text{cross}})$。 □

同样的方法也可用于欧几里得最小生成树：

定理 5.5. 给定一组包含 P 个 n平面上点的三角剖分 G，可以在 $O(n(\log^* n)^2 + n \log d_{\text{cross-emst}})$期望时间内构造欧几里得最小生成树 $EMST(P)$ 和德劳内三角剖分 $DT(P)$，其中 $d_{\text{cross-emst}}$ 是 $EMST(P)$ 中一条边可能与 G 的一条边相交的最大边数（该值未知）。

证明。我们修改定理 5.4 证明中的算法，将 $DT(\cdot)$ 替换为 $EMST(\cdot)$，并将“triangula- tions”替换为“生成树”。所有系数 3 都替换为 1。在第 6 步中，我们改为调用推论 3.6（这会引入额外的 $\log^* n$ 因子）。猜测 $d_{\text{cross-emst}}$ 时，如果将运行时间限制为 $C\mu 2^j$（其中 C 是足够大的常数），那么对于给定的 s， j 需要达到 $\max\{\lceil \log(\log^* \mu) \rceil, \lceil \log \log d_{\text{cross-emst}} \rceil + s\}$ 的概率为 $O(1/C)^s$，因此总期望运行时间为

$$\sum_{s \geq 0} O(1/C)^s \cdot C(\mu \log^* \mu + 2^s \mu \log d_{\text{cross-emst}}) = O(\mu \log^* \mu + \mu \log d_{\text{cross-emst}}).$$

计算出欧几里得最小生成树(P)后，例如可通过 Devillers 算法 [30] 在 $O(n \log^* n)$ 的额外期望运行时间内得到德劳内三角剖分(P)。 □

我们也可以得到涉及多个参数的结果；例如，不难调整定理 5.4 的证明，从而得到一种德劳内三角剖分算法，其运行时间为 $O(n \log^* n + D \log d_{\text{cross}})$。

6 概率模型下的 $O(n \log \log n + n \log(1/\rho))$ 算法

利用我们的 d_{cross}-敏感算法，我们可以立即得到一种用于概率模型下德劳内三角剖分的高效算法。

引理 6.1。给定平面上的一个包含 n 个点的集合 P，以及 $\rho \in (0,1)$，令 R 为德劳内三角剖分 $DT(P)$ 的边的一个子集，其中每条边都以概率 ρ 独立选取。令 G 为包含 R 的 P 的任意一个三角剖分。令 d_{cross} 为德劳内三角剖分 $DT(P)$ 的边与 G 的一条边可能相交的最大边数。那么，$d_{\text{cross}} \leq O((1/\rho) \log n)$ 以高概率成立。

证明。这可由标准的基于采样的击中集或 ε-网论证得出：G cross R的边都不属于其中。对于固定的 $p, q \in P$，若 pq crosses $DT(P)$ 中超过 $(c/\rho) \ln n$ 条边，则 pq不与 Rcross 的概率至多为 $(1-\rho)^{(c/\rho) \ln n} \leq (e^{-\rho})^{(c/\rho) \ln n} = n^{-c}$。根据并合界， $d_{\text{cross}} > (c/\rho) \ln n$的概率至多为 n^{-c+2}。 □

推论 6.2。给定由 n 个平面上的点组成的集合 P 以及 $\rho \in (0,1)$，令 R 为 $DT(P)$ 的边的一个子集，其中每条边都以概率 ρ独立选取。令 G 为包含 R的 P 的任意三角剖分。给定 G，可以在 $O(n \log \log n + n \log(1/\rho))$时间内以高概率构造德劳内三角剖分 $DT(P)$。

证明。我们只需结合前述引理应用定理 5.4。 □

类似的方法也适用于欧几里得最小生成树：

引理 6.3。给定平面上的 n 个点组成的集合 P，以及 $\rho \in (0,1)$，令 R 为欧几里得最小生成树(P) 的边的一个子集，其中每条边都以概率 ρ 独立选取。令 T 为任意一棵包含 R 且没有交叉的生成树。令 $d_{\text{cross-emst}}$ 为欧几里得最小生成树(P)中一条边可能与 T 的边交叉的最大边数。那么，高概率下 $d_{\text{cross-emst}} \leq O((1/\rho)\log n)$。

推论 6.4。给定平面上的 n 个点组成的集合 P，以及 $\rho \in (0,1)$，令 R 为欧几里得最小生成树(P) 的边的一个子集，其中每条边都以概率 ρ 独立选取。令 T 为任意一棵包含 R 且没有交叉的生成树。给定 T，欧几里得最小生成树(P)和德劳内三角剖分(P) 可在高概率下于 $O(n\log\log n + n\log(1/\rho))$ 时间内构造出来。

证明。我们可以从 T计算出 P的三角剖分 G，例如使用 Chazelle 算法在线性时间 [14] 内完成，或使用 Seidel 算法在 $O(n\log^* n)$ 期望时间 [56]内完成。然后，我们可以结合前面的引理应用定理 5.5。□

在随机翻转模型中也有类似的结果：

引理 6.5。给定一组 n 个平面上的点 P 以及 $\rho \in (0,1)$，考虑以下过程：

> *(*)* 初始化 $G =$ 德劳内三角剖分(P)。令 m 为 G中的边数。对于 $i = 1, \ldots, (1-\rho)m$：从 G中随机选取一条边 e_i；如果 G 中与 e_i 相邻的两个三角形构成一个凸四边形，则翻转 e_i 中的 G。

过程结束时，令 d_{cross} 为 G 的一条边可能与德劳内三角剖分(P) 相交的边的最大数量。那么以高概率有 $d_{\text{cross}} \leq O((1/\rho)\log n)$。

证明。设 $t = (1-\rho)m$。为便于分析，我们为 G 中的每条边 e 指定一个不同的标签 $\ell(e) \in \{1, \ldots, m\}$。翻转一条边时，我们为新边赋予与旧边相同的标签，以保持不变式。于是 $\ell(e_1), \ldots, \ell(e_t)$ 是来自 $\{1, \ldots, m\}$ 的相互独立且均匀分布的随机变量。考虑一个固定的 $p, q \in P$，使得 pq 穿过 DT(P) 中超过 $b := (c/\rho)\ln n$ 条边。设 B 是这些边中固定的 b 子集的标签。pq 出现在最终 G 中的概率，至多等于 $\{\ell(e_1), \ldots, \ell(e_t)$ 包含 B 的概率，即为 $\binom{t}{b} \cdot \frac{b!}{m^b} = \frac{t!}{m^b (t-b)!} \leq \frac{t^b}{m^b} = (1-\rho)^b \leq e^{-\rho b} = n^{-c}$。根据并合界，$d_{\text{cross}} > (c/\rho)\ln n$ 的概率至多为 n^{-c+2}。□

推论 6.6. 给定平面上的 P 个点以及 $\rho \in (0,1)$，令 G 为通过上述过程 *(*)* 生成的 P 的一个三角剖分。给定 G，德劳内三角剖分 $DT(P)$ 可在 $O(n\log\log n + n\log(1/\rho))$ 时间内以高概率构造出来。

7 一种 $O(n\log^* n + n\log d_{\text{vio}})$ 算法

使用我们的 d_{cross}-敏感算法，我们还可以立即得到一种用于德劳内三角剖分的 d_{vio}-敏感算法，只需证明 d_{cross} 总是有上界 $O(d_{\text{vio}}^2)$。

引理 7.1. 给定点集 P（由 n 个平面上的点组成）的一种三角剖分 G，令 d_{vio} 为三角剖分 P 中，某个三角形的外接圆所能包含的点的最大数量。考虑 G 中的一个三角形 $\triangle qq''q'''$。德劳内三角剖分(P) 中与 q 相接且与 $q''q'''$ 相交的边数至多为 d_{vio}。

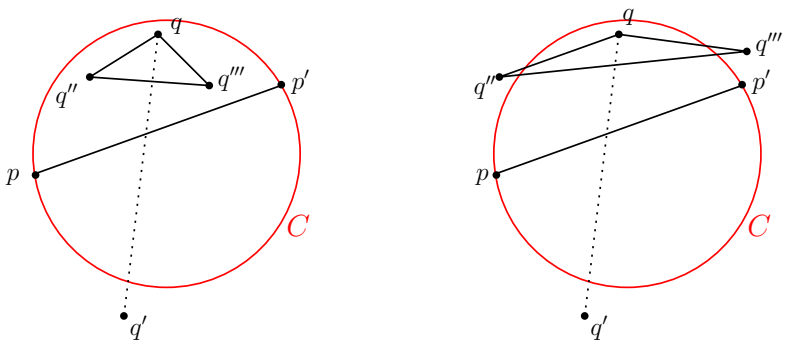

图4　：引理7.2的证明。左侧示例属于情形 (i)；右侧示例属于情形 (ii)。

证明。 假设 qq' 是德劳内三角剖分 $DT(P)$ 中与 $q''q'''$ 相交的一条边。那么 q 必须位于 $\triangle qq''q'''$ 的外接圆内（否则， qq' 和 $q''q'''$ 都会是德劳内三角剖分 $DT(\{q, q', q'', q'''\})$ 的边，这与德劳内三角剖分的平面性矛盾）。因此， q' 至多有 d_{vio} 种选择。 □

引理 7.2。 给定点集 P 的一个三角剖分 G，其中有 n 个平面上的点。设 d_{cross} 为德劳内三角剖分(P) 中一条边最多可能与 G 的一条边相交的次数，设 d_{vio} 为 P 中最多可能落在 G 的一个三角形外接圆内的点数。那么 $d_{\text{cross}} \leq O(d_{\text{vio}}^2)$。

证明。 考虑 G 的一条边 pp' 。设 G 中与这条边相邻的一个三角形为 $\triangle pp'p''$ 。设 $pp'p''$ 的外接圆为 C 。设 G 中由位于 C 内部的点诱导的子图为 H 。

假设 qq' 是德劳内三角剖分德劳内三角剖分(P) 的一条边，且与 pp' 相交。那么 q 和 q' 中至少有一个必须位于 C 内部（否则， pp' 和 qq' 都会是德劳内三角剖分$(\{p, p', q, q'\})$ 的边，这与德劳内三角剖分的平面性矛盾）。不失一般性，假设 q 位于 C 内部。设 G 中与 q 相邻、且 q' 位于 $\overrightarrow{qq''}$ 和 $\overrightarrow{qq'''}$ 之间的角域内的三角形为 $\triangle qq''q'''$ 。见图4。注意， $\triangle qq''q'''$ 与 pp' 不相交（因为 G 是平面的）。因此， qq' 必须与 $q''q'''$ 相交。根据引理 7.1，对于每个固定的 $\triangle qq''q'''$， q' 至多有 d_{vio} 种选择。

此外，必有以下情形之一： （i） q'' 或 q''' 位于 C内部；或（ii） p 和 p' 位于 $\overrightarrow{qq''}$ 与 $\overrightarrow{qq'''}$之间的角域内。在情形（i）中，对于每个固定的q， $\triangle qq''q'''$ 至多有 $2 \cdot \deg_H(q)$种选择。在情形（ii）中，每个 q只有一种 $\triangle qq''q'''$ 选择。因此， $\triangle qq''q'''$ 的选择总数为$O(\sum_q 2\deg_H(q) + 1))$，即 $O(d_{\text{vio}})$（因为 H 是一个至多有 d_{vio} 个顶点的平面图）。 qq' 的选择总数为 $O(d_{\text{vio}}^2)$。 □

推论 7.3。 给定点集 P （由 n 个平面上的点组成）的一种三角剖分 G，可以在 $O(n \log^* n + n \log d_{\text{vio}})$ 确定性时间内构造德劳内三角剖分 $DT(P)$，其中 d_{vio} 是 P 中的点可能被 G 的一个三角形的外接圆包含的最大数量（未知）。

或者，证明不等式 $d_{\text{cross-emst}} \leq O(d_{\text{vio}})$ 会更容易（因为欧几里得最小生成树的最大度是常数）；然后我们可以应用定理 5.5 得到几乎相同的结果，只是多了一个 $\log^* n$ 因子。

资助

本研究部分由斯洛文尼亚研究与创新署（P1-0297、J1-2452、N1-0218、N1-0285）资助，部分由欧盟资助（欧洲研究委员会，KARST，项目编号 101071836）。观点

和意见仅代表作者本人，并不一定反映欧盟或欧洲研究委员会的观点。欧盟和授予资助的机构均不对此承担责任。

致谢

作者感谢 Kostas Tsakalidis 提出带预测的算法问题这一研究方向。

参考文献

[1] 带预测的算法。https://algorithms-with-predictions.github.io/。由 Alexander Lindermayr 和 Nicole Megow 领导并管理。[2] Alok Aggarwal、Leonidas J. Guibas、James B. Saxe 和 Peter W. Shor。用于计算凸多边形沃罗诺伊图的线性时间算法。离散与计算几何，4:591–604，1989年。doi:10.1007/ BF02187749。[3] Nancy M. Amato、Michael T. Goodrich 和 Edgar A. Ramos。高维凸包的并行算法。载于第*35*届计算机科学基础年度研讨会（*FOCS*）论文集，第683–694页，1994年。doi:10.1109/SFCS.1994.365724。[4] Nicolas Auger、Vincent Jugé、Cyril Nicaud 和 Carine Pivoteau。论 TimSort 的最坏情况复杂度。载于第*26*届欧洲算法研讨会（*ESA*）论文集，第4:1–4:13页，2018年。doi:10.4230/ LIPICS.ESA.2018.4。[5] Franz Aurenhammer 和 Rolf Klein。沃罗诺伊图。载于 Jörg-Rüdiger Sack 和 Jorge Urrutia 编，《style id='12'>计算几何手册，第201–290页。North Holland / Elsevier，2000年。doi:10.1016/ b978-044482537-7/50006-1。[6] Xingjian Bai 和 Christian Coester。带预测的排序。载于神经信息处理系统进展第*36*卷（*NeurIPS*），2023年。网址：http://papers.nips.cc/paper_files/paper/2023/hash/544696ef4847c903376ed6ec58f3a703-Abstract-Conference.html。[7] Jérémy Barbay 和 Gonzalo Navarro。论排列压缩与自适应排序。理论计算机科学，513:109–123，2013年。doi:10.1016/J.TCS.2013.10.019。[8] Jean-Daniel Boissonnat 和 Mariette Yvinec。《style id='22'>算法几何》。剑桥大学出版社，1998年。[9] Kevin Buchin 和 Wolfgang Mulzer。 $O(\mathrm{sort}(n))$ 时间及更多条件下的德劳内三角剖分。*ACM*杂志，58(2):6:1– 6:27，2011年。doi:10.1145/1944345.1944347。[10] Sergio Cabello 和 Panos Giannopoulos。利用预测在欧几里得空间中搜索。载于第*22*届近似与在线算法研讨会（*WAOA*）论文集，《style id='32'>计算机科学讲义第15269卷，第31–45页，2024年。doi:10.1007/978-3-031-81396-2_3。[11] Timothy M. Chan。用于求三维空间中两个凸多面体交集的更简单线性时间算法。离散与计算几何，56(4):860–865，2016年。doi:10.1007/s00454-016-9785-3。[12] Timothy M. Chan。通过浅层切割构建动态几何数据结构。离散与计算几何，64(4):1235–1252，2020年。doi:10.1007/s00454-020-00229-5。

[13] Timothy M. Chan、Jack Snoeyink 和 Chee-Keng Yap。原始划分与对偶剪枝：四维多胞体和三维沃罗诺伊图的输出敏感构造。离散与计算几何，18(4):433–454，1997年。doi:10.1007/PL00009327。[14] Bernard Chazelle。在线性时间内对简单多边形进行三角剖分。离散与计算几何，6:485–524，1991年。doi:10.1007/BF02574703。[15] Bernard Chazelle。求三维凸多面体交集的最优算法。*SIAM*计算杂志，21(4):671–696，1992年。doi:10.1137/0221041。[16] Bernard Chazelle、Olivier Devillers、Ferran Hurtado、Mercè Mora、Vera Sacristán 和 Monique Teillaud。在线性时间内分割德劳内三角剖分。*Algorithmica*，34(1):39–46，2002年。doi:10.1007/ S00453-002-0939-8。[17] Bernard Chazelle 和 Joel Friedman。随机抽样的确定性视角及其在几何学中的应用。组合学，10(3):229–249，1990年。doi:10.1007/BF02122778。[18] Bernard Chazelle 和 Wolfgang Mulzer。计算遗传凸结构。离散与计算几何，45(4):796–823，2011年。doi:10.1007/S00454-011-9346-8。[19] Danny Z. Chen 和 Haitao Wang。多边形区域中的可见性与射线发射查询。计算几何，48(2):31–41，2015年。doi:10.1016/J.COMGEO.2014.08.003。[20] Otfried Cheong、Ketan Mulmuley 和 Edgar Ramos。随机化与去随机化。载于 Jacob E. Good- man、Joseph O'Rourke 和 Csaba D. Tóth 编，《style id='23'>离散与计算几何手册。CRC出版社，2017年。网址：https://www.csun.edu/~ctoth/Handbook/chap44.pdf。[21] David R. Cheriton 和 Robert Endre Tarjan。寻找最小生成树。*SIAM*计算杂志，5(4):724–742，1976年。doi:10.1137/0205051。[22] L. Paul Chew。在线性期望时间内为凸多边形构建沃罗诺伊图。技术报告 PCS- TR90-147，达特茅斯学院数学与计算机科学系，汉诺威，新罕布什尔州，1986年。[23] L. Paul Chew。约束德劳内三角剖分。*Algorithmica*，4(1):97–108，1989年。doi:10.1007/ BF01553881。[24] Francis Y. L. Chin 和 Cao An Wang。在线性时间内求简单多边形的约束德劳内三角剖分和约束沃罗诺伊图。*SIAM*计算杂志，28(2):471–486，1998年。doi:10.1137/ S0097539795285916。[25] Kenneth L. Clarkson。随机化几何算法。载于 D.-Z. Du 和 F. K. Hwang 编，《style id='37'>欧几里得几何计算，第117–162页。世界科技出版公司，1992年。[26] Kenneth L. Clarkson 和 Peter W. Shor。随机抽样在计算几何中的应用，第二部分。离散与计算几何，4:387–421，1989年。doi:10.1007/BF02187740。[27] Kenneth L. Clarkson、Robert Endre Tarjan 和 Christopher J. Van Wyk。用于三角剖分简单多边形的快速拉斯维加斯算法。离散与计算几何，4:423–432，1989年。doi:10.1007/BF02187741。[28] Vincent Cohen-Addad、Tommaso d'Orsi、Anupam Gupta、Euiwoong Lee 和 Debmalya Panigrahi。最大割及相关问题的学习增强近似算法。载于神经信息处理系统进展第*38*卷（*NeurIPS*），2024年。网址：http://papers.nips.cc/paper_files/paper/2024/hash/2db08b94565c0d582cc53de6cee5fd47-Abstract-Conference.html。[29] Mark de Berg、Otfried Cheong、Marc J. van Kreveld 和 Mark H. Overmars。《style id='50'>计算几何：算法与应用。施普林格，第3版，2008年。

[30] Olivier Devillers。随机化为困难的 Ω(n) 问题带来了简单的 $O(n\log^* n)$ 算法。国际计算几何与应用杂志，2(1):97–111，1992年。doi:10.1142/S021819599200007X.[31] Olivier Devillers、Giuseppe Liotta、Franco P. Preparata 和 Roberto Tamassia。多胞体凸性与剖分平面性的检验。计算几何，11(3-4):187–208，1998年。doi:10.1016/ S0925-7721(98)00039-X.[32] Herbert Edelsbrunner 和 Weiping Shi。用于三维凸包问题的 $O(n\log^2 h)$ 时间算法。SIAM计算杂志，20(2):259–269，1991年。doi:10.1137/0220016.[33] David Eppstein、Michael T. Goodrich 和 Vinesh Sridhar。概率噪声原语操作下的计算几何。载于第19届国际算法与数据结构研讨会（WADS）论文集，第24:1–24:20页，2025年。doi:10.4230/LIPIcs.WADS.2025.24.[34] Vladimir Estivill-Castro 和 Derick Wood。自适应排序算法综述。ACM计算调查，24(4):441–476，1992年。doi:10.1145/146370.146381.[35] Greg N. Federickson。平面图最短路径的快速算法及其应用。SIAM计算杂志，16(6):1004–1022，1987年。doi:10.1137/0216064.[36] Steven Fortune。沃罗诺伊图与德劳内三角剖分。载于 Jacob E. Goodman 和 Joseph O'Rourke 编，《离散与计算几何手册》，第705–721页。Chapman and Hall/CRC，第3版，2017年。URL: https://www.csun.edu/~ctoth/Handbook/chap27.pdf.[37] Joachim Gudmundsson、Mikael Hammar 和 Marc J. van Kreveld。高阶Delaunay三角剖分。计算几何，23(1):85–98，2002年。doi:10.1016/S0925-7721(01)00027-X.[38] Sabine Hanke、Thomas Ottmann 和 Sven Schuierer。三角剖分的边翻转距离。通用计算机科学杂志，2(8):570–579，1996年。doi:10.3217/JUCS-002-08-0570.[39] John Iacono 和 Stefan Langerman。近邻平面点定位。载于第19届ACM计算几何研讨会（SoCG）论文集，第220–226页，2003年。doi:10.1145/777792.777826.[40] Jerzy W. Jaromczyk 和 Godfried T. Toussaint。相对邻域图及其相关图。IEEE会刊，80(9):1502–1517，1992年。doi:10.1109/5.163414.[41] David R. Karger、Philip N. Klein 和 Robert Endre Tarjan。求解最小生成树的随机化线性时间算法。ACM杂志，42(2):321–328，1995年。doi:10.1145/201019.201022.[42] David G. Kirkpatrick。连续骨架的高效计算。载于第20届年度计算机科学基础研讨会（FOCS）论文集，第18–27页，1979年。doi:10.1109/SFCS.1979.15.[43] David G. Kirkpatrick。平面剖分中的最优搜索。SIAM计算杂志，12(1):28–35，1983年。doi: 10.1137/0212002.[44] Philip N. Klein、Shay Mozes 和 Christian Sommer。在线性时间内对平面图进行结构化递归分隔器分解。载于第45届ACM计算理论研讨会（STOC）论文集，第505–514页，2013年。doi:10.1145/2488608.2488672.[45] D. T. Lee 和 A. K. Lin。平面图的广义Delaunay三角剖分。离散与计算几何，1:201– 217，1986年。doi:10.1007/BF02187695.[46] Andrzej Lingas。由德劳内三角剖分在线性时间内构造相对邻域图。计算几何，4:199–208，1994年。doi:10.1016/0925-7721(94)90018-3.[47] Maarten Löffler 和 Wolfgang Mulzer。正方形三角剖分与三角形正方形化：四叉树与德劳内三角剖分等价。SIAM计算杂志，41(4):941–974，2012年。doi:10.1137/110825698。

[48] Maarten Löffler 和 Jack Snoeyink。预处理后在线性时间内完成不精确点的德劳内三角剖分。计算几何，43(3):234–242，2010年。doi:10.1016/J.COMGEO.2008.12.007.[49] Pinyan Lu、Xuandi Ren、Enze Sun 和 Yubo Zhang。带预测的广义排序。载于第*4*届算法简约性研讨会（*SOSA*）论文集，第111–117页，2021年。[50] Jiˇrí Matoušek。ε-网与计算几何。载于 János Pach 编，《离散与计算几何新趋势》，第69–89页。施普林格，1993年。doi:10.1007/978-3-642-58043-7_4.[51] Tomomi Matsui。平面图上的最小生成树问题。离散应用数学，58(1):91–94，1995年。doi:10.1016/0166-218X(94)00095-U.[52] Ross M. McConnell、Kurt Mehlhorn、Stefan Näher 和 Pascal Schweitzer。认证算法。计算机科学评论，5(2):119–161，2011年。doi:10.1016/j.cosrev.2010.09.009.[53] Kurt Mehlhorn、Stefan Näher、Michael Seel、Raimund Seidel、Thomas Schilz、Stefan Schirra 和 Christian Uhrig。几何程序的检验，或几何结构的验证。计算几何，12(1-2):85–103，1999年。初步版本发表于SoCG 1996。doi:10.1016/S0925-7721(98)00036-4.[54] Michael Mitzenmacher 和 Sergei Vassilvitskii。带预测的算法。载于 Tim Roughgarden 编，超越算法的最坏情况分析，第646–662页。剑桥大学出版社，2020年。doi:10. 1017/9781108637435.037.[55] J. Ian Munro 和 Sebastian Wild。近乎最优的归并排序：快速实用且能根据已有运行段实现最优自适应的排序方法。载于第*26*届欧洲算法研讨会（*ESA*）论文集，第63:1–63:16页，2018年。doi:10.4230/LIPICS.ESA.2018.63.[56] Raimund Seidel。一种简单快速的增量随机算法，用于计算梯形剖分和多边形三角剖分。计算几何，1:51–64，1991年。doi:10.1016/0925-7721(91)90012-4。

A 个确定性 $O(n + D$对数$^4 n)$ 时间算法

在本附录中，我们介绍一种用于计算德劳内三角剖分的不同的确定性 D敏感算法。尽管它比第 2 节中的算法慢一个对数因子，但该方法不使用平面图分隔器，因而可能具有独立的研究价值（它使用某些动态几何数据结构；如果能够改进这些数据结构，运行时间或许还能进一步缩短）。

在此引入更一般的平面直线图概念会很方便，简称为*PSLG*，即所有边均绘制为直线边的图的平面嵌入。因此，平面点集 P 的三角剖分是一个满足以下条件的连通平面直线图：(i) 其顶点集恰好为 P；(ii) 其所有有界面均为三角形；(iii) 无界面是凸包$CH(P)$的补集。需要指出的是，一个内部含有额外顶点（这些顶点可能是孤立的）的三角形，并不是平面直线图的三角形面。一个面的组合复杂度，是界定该面的边数与顶点数之和。面内的孤立顶点也计入该数量。

对于任意三个点 p、q、r，$C(p, q, r)$ 表示经过 p、q、r 的唯一圆。对于三角形 $\triangle$，我们将其外接圆记为 $C(\triangle)$。

设 G 是一个平面直线图，f 是 G 的一个面。我们用 $N_G(f)$ 表示 f 在 G 的对偶图中的邻面集合。因此，面 f' 属于 $N_G(f)$，当且仅当 f 和 f' 共享一条边。令 $V(N_G(f))$ 表示 $N_G(f)$ 中各面的顶点集。注意，如果 $N_G(f)$ 的一个面在其内部包含孤立顶点，那么这些顶点也属于 $V(N_G(f))$。见图 5。

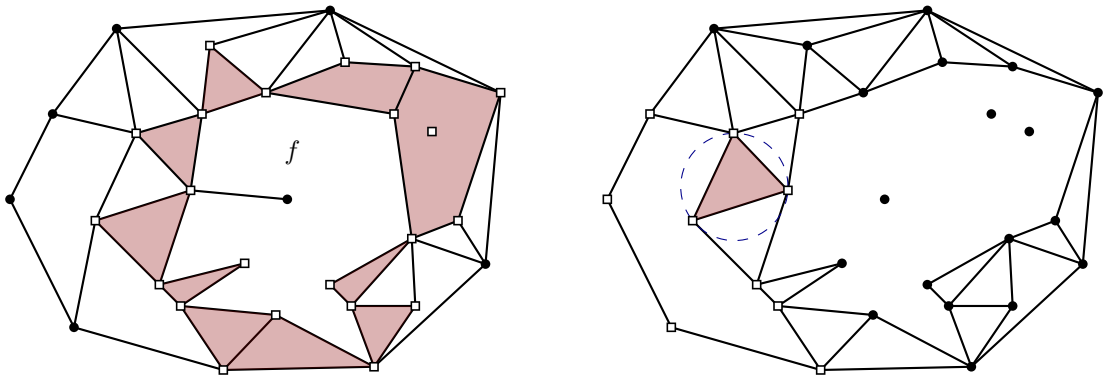

图 5：左图：为面 f 标记了 $N_G(f)$，并用空心方块标记了 $V(N_G(f))$ 中的点。左侧的平面直线图不满足引理 A.1 的条件，而右侧的平面直线图满足。要验证右侧阴影三角形是否满足引理 A.1 的假设，我们必须检查用空心方块标记的点。

德劳内三角剖分的标准空圆性质（见引言）意味着，可以在线性时间内检验给定的三角剖分是否为德劳内三角剖分。我们还需要下面这个扩展，以识别德劳内三角剖分的子图。假设条件见右侧的图 5。

引理 A.1。设 G 是一个平面直线图，其顶点集为 P，并且 G 的每条边都位于 $CH(P)$ 的边界上，或与一个三角形面相邻。假设，对于 G 中的每个三角形 $\triangle$，$V(N_G(\triangle))$ 中的所有点都在外接圆 $C(\triangle)$ 之外。那么 G 的每条边都是德劳内三角剖分 $DT(P)$ 的一条边。

证明。将 G 扩展为一个三角剖分 T，并使用将 T 转换为德劳内三角剖分的算法，通过边翻转使边合法，具体描述见 [29, 第 9]章。我们证明，在整个算法过程中，G 的所有边始终合法。

假设，为了导出矛盾，$e = p_i p_j$ 是 G 中第一条因变为不合法而被翻转的边。设 T_e 是翻转边 $e = p_i p_j$之前的三角剖分。 e不可能属于$CH(P)$的边界，因为这些边永远不会变得不合法，也不会被翻转。因此， e 不在$CH(P)$的边界上。根据假设， G中有一个三角形 $\triangle p_i p_j p_k$ 。设 G 的另一个面为 f，其边界上有$p_i p_j$； f 是一个有界面。

由于目前尚未翻转 G的任何边， $\triangle$和 f 的边仍然是当前三角剖分 T_e中的边。这意味着 f 是 T_e 中若干三角形的并集。对于 T_e 中以$p_i p_j$ 为一条边的另一个三角形 $\triangle p_i p_j p_l \neq \triangle p_i p_j p_k$ ，必有 p_l 是 f 的一个顶点。然而，由于 f中没有顶点位于圆 $C(p_i, p_j, p_k)$内，点 p_l 不在 $C(p_i, p_j, p_k)$的内部。这意味着$e = p_i p_j$不可能是 T_e中的非法边，矛盾。 □

引理 A.2. 设 G 是 P 的一个三角剖分，且 E' 是 E（G）的一个子集，其中有 k 条边。从 G中移除 E' 的边，然后再移除所有两侧都邻接有界非三角形面的边。总共移除 O（k）条边。

证明。 移除 E' 的每条边都可能产生一个额外的非三角形面，并使非三角形面上的顶点总数增加 $O(1)$。因此，移除 E' 的所有边后，我们有 $O(k)$个非三角形面，它们总共包含 $O(k)$个顶点。

现在，我们删除两侧均为有界非三角形面的边。见图 5，右侧的平面直线图由左侧的平面直线图得到。如果该边两侧是两个不同的面，那么删除它会合并两个非三角形面，但不会改变与这些面关联的顶点数。因此，这种情况至多发生 $O(k)$次，因为如果起初有 ℓ个非三角形面，这一操作最多可以进行 $\ell - 1$次。此后，我们得到 $O(k)$个面，它们总共包含 $O(k)$个顶点。

现在，我们删除两侧属于同一个面的边。删除这些边中的每一条，都不会改变面数，也不会改变与非三角形面关联的顶点数，但会减少与非三角形面关联的边数。由于这些非三角形面构成一个具有 $O(k)$个顶点的平面图，它们有 $O(k)$条边，因此这些删除操作可以进行 $O(k)$次。 □

我们将使用以下由 Chan [12] 通过通常的提升到抛物面 $z = x^2 + y^2$得到的数据结构。对于第二部分，我们只需在 $\mathbb{R}^3$中使用点-平面对偶。

引理 A.3。我们 可以维护一个动态集合 P 的 n平面上的点，并支持以下操作 :

- 插入或删除一个点需要 $O(\log^4 n)$ 摊还时间;
- 对于查询圆 C，我们可以检测 P 是否有某个点位于圆 C 内，这需要 $O(\log^2 n)$ 时间。

我们可以维护一个动态集合 $\mathcal{C}$ 的 n 平面上的圆，并支持以下操作:

- 插入或删除一个圆需要 $O(\log^4 n)$ </style>摊还时间;
- 对于查询点 p，我们可以在 $O(\log^2 n)$ </style>时间内检测 $\mathcal{C}$ 的某个圆是否包含 p 。

接下来我们展示主要结果。

定理 A.4. 设 P 是由 n 个平面上的点组成的集合， G 是一个顶点集为 P 的平面直线图。德劳内三角剖分 $DT(P)$ 可在 $O(n + D\log^4 n)$ 时间内构造出来，其中 D 重新定义为 $DT(P)$ 中不属于 G 的边数（该数量未知）。

证明. 第一步是在 $O(n + D\log^4 D)$ 时间内，找出 G 的一个边子集 E'，其中含有 $O(D)$ 条边，使得 $G - E'$ 的所有边都属于德劳内三角剖分 $DT(P)$。子集 E' 也可能包含 $DT(P)$ 的一些边；因此，我们计算的是“错误”边的一个超集。识别 E' 的判据是，$G - E'$ 应满足引理 A.1 的假设。这里用到了两个基本思路：

- 检查三角形面的外接圆，判断其中是否包含相邻面上的某个点。由于相邻面可能发生变化，同一个三角形可能需要检查多次；为高效完成这一工作，我们使用数据结构。

- 如果一条边位于两个非三角形面的边界上，我们就没有可用于检查该边的三角形。在这种情况下，我们只需将这条边加入 E'，但根据引理 A.2，这种情况可能发生 $O(D)$ 次。

现在我们来说明细节。对于 G 的任意子图 G'，令 $F_{\geq 4}(G')$ 表示 G' 中非三角形且有界的面。因此，$CH(P)$ 的补集永远不会属于 $F_{\geq 4}(G')$。如果 G 是一个三角剖分，那么 $F_{\geq 4}(G)$ 就是空集。开始时，$F_{\geq 4}(G)$ 有 $O(D)$ 个顶点；即使 G 是 德劳内三角剖分(P) 的子图，我们仍然必须向 G 添加 D 条边，才能得到 德劳内三角剖分(P)；而要将含有 $\ell > 3$ 个顶点的每个非三角形面三角剖分，还需要添加 $\ell - 3 > 0$ 条边。我们并不显式维护 $F_{\geq 4}(G)$，但为它引入一个记号很有用。

我们从 $E' = \emptyset$ 和 PSLG $G' = G$开始；我们保持不变式：$G = G' + E'$。因此，E' 是待移除的边集。

G' 的三角形被标记为待检查 或已检查。初始时，我们将G 的所有三角形标记为待检查。（如果$CH(P)$是一个三角形，则外面会被标记为已检查，并且始终保持该标记。）

令 Q 为位于 $F_{\geq 4}$（G'）各面中的点集。我们将 Q 维护在引理 A.3 的第一个数据结构中。初始时，Q 含有 O（D）个元素。我们还维护集合 τ，其中包含与 $F_{\geq 4}$（G'）的某个面共享一条边且被标记为已检查的三角形。对于 τ 中的每个三角形，其外接圆都存储在引理 A.3 的第二个数据结构中。初始时，τ 不含任何元素，因为所有三角形面都被标记为待检查。我们保持不变式：对于任意 $\triangle \in \tau$，Q 中没有点位于 C（$\triangle$）的内部。

我们遍历 G' 中标记为待检查的三角形 $\triangle$，并对每个三角形执行以下操作。我们要判断 $C(\triangle)$ 是否包含 $V(N_{G'}(\triangle))$ 中的某个点。如果 $\triangle$ 的某条边 e 与 G' 中的另一个三角形 $\triangle'$ 共享，我们就测试 $\triangle'$ 的顶点是否位于外接圆$C(\triangle)$ 的内部。我们还会查询存储 Q 的数据结构，以查看 $C(\triangle)$ 是否包含 Q 中的某个点。这样也涵盖了与 $\triangle$ 相邻的某个面 f 非三角形的情况，因为 $V(f) \subseteq Q$。通过这些操作，我们已测试 $V(N_{G'}(\triangle))$ 中的任意点是否包含在 $C(\triangle)$ 中；我们还测试了一些额外的点，即 $Q \setminus V(N_{G'}(\triangle))$。

如果我们检测到 $C(\triangle)$ 的内部存在某个点，就会在检测到该点后立即删除 $E(\triangle)$ 的三条边；删除操作属于对 $\triangle$ 的迭代过程。请注意，当我们决定删除 $E(\triangle)$ 时，可以确定 $\triangle$ 的边中至少有一条不在 $DT(P)$ 中。因此，这样最多会删除 $3D$ 条边。

如果不删除 $\triangle$的边，那么我们将 $\triangle$ 标记为 已检查。在这种情况下，如果 $\triangle$ 与 $F_{\geq 4}$（G'）的某个面相邻，我们就将其加入 τ，并将其外接圆 C（$\triangle$）加入相应的数据结构。

现在考虑对于三角形 $\triangle$，需要如何删除 E（$\triangle$）。我们从 G' 中移除 E（$\triangle$）上不在 CH（P）边界上的边，并将它们加入 E'。三角形 $\triangle$ 现在成为一个新面的组成部分，我们将其记为 f，它属于 $F_{\geq 4}$（G'）。我们通过以下操作恢复不变式：

- 我们将此前不在 Q 中的$Q_{\text{新}}$ 顶点添加到 Q；此类新点至多有 $O(1)$个

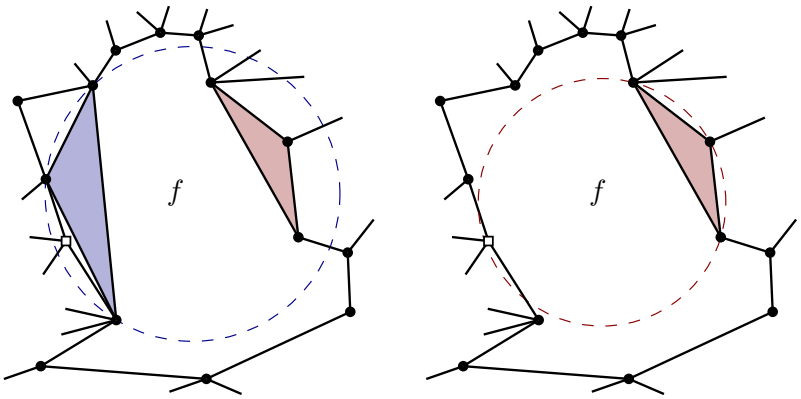

图6：检查蓝色阴影三角形 $\triangle$时，我们决定删除 $E(\triangle)$，因为 $C(\triangle)$ 包含 $V(f) \subseteq Q$中的某个点。空心方块标记的点成为 Q 的一个新点，它位于 $C(\triangle')$ 的内部，其中 $\triangle'$ 以红色阴影标出。因此， $\triangle'$ 被标记为待检查（可能是再次标记）。

添加到 Q ，因为除了 $\triangle$的顶点外，只有 $\triangle$ 的邻接三角形可能向 Q贡献一个新点。

- 面 f 可能有邻接三角形 $\tau_{\text{新}}$ 标记为已检查，需要将它们添加到 τ。同样，这类新的三角形面有 O个，因为只有与 $\triangle$ 相邻的三角形面才可能为 τ生成新的三角形面。因此，我们可以在$O(1)$时间内识别出这些适用于 τ 的新三角形面 $\tau_{\text{新}}$。我们还会将 $\tau_{\text{新}}$ 中的三角形的外接圆添加到相应的数据结构中。

- 我们必须重新检查 $\tau_{\text{新}}$ 中的每个三角形 $\triangle$，因为它们的邻面发生了变化，并且 Q 包含了可能位于 $C(\triangle)$ 内部的新点。对于 $\tau_{\text{新}}$中的每个三角形 $\triangle$，我们查询存储Q 的数据结构，以确定 $\triangle$ 内部是否包含 Q 中的某个点。如果包含，我们就将 $\triangle$ 标记为待检查；之后会再次检查并将其删除，除非由于其他删除操作，它不再是 G' 中的三角形。

- 与 τ相邻的三角形必须重新检查，因为 V（f）包含新的点，即$Q_{\text{新点}}$。示例请参见图6。我们使用存储 C（$\triangle'$）的$\triangle' \in \tau$数据结构来完成此操作。我们使用$Q_{\text{新点}}$ 中的每个点多次查询该数据结构，以找出

$$\tau' = \{\triangle' \in \tau \mid C(\triangle') \text{ contains some point of } Q_{\text{new}} \text{ in the interior}\}.$$

我们从 τ中删除 τ'的三角形；实际上，这已经在计算 τ'期间完成了。我们将 τ'的三角形标记为待检查。之后将再次测试 τ'的每个三角形并将其删除，除非由于其他删除操作，它在 G' 中不再是三角形。

至此，我们暂时完成了对 G' 和 E' 计算过程的描述。对于每个被删除的三角形，它至少有一条边不属于 德劳内三角剖分(P)。因此，我们至多删除 $3D$ 条边。对于每个被删除的三角形，我们向 Q 添加 $O(1)$ 个新点，并向 τ 添加 $O(1)$ 个三角形。这样，我们对存储 Q 和 τ 中三角形外接圆的数据结构进行 $O(D)$ 次修改，这些数据结构见引理 A.3。在 Q 中我们不进行删除，而在 τ 中，当我们确定某个三角形将被删除时，就会进行删除。对于每个被删除的三角形，我们会将一些三角形标记为 待检查，但这只会在我们确定该三角形将被删除时发生。因此，我们最多将 $O(D)$ 个三角形从 已检查重新标记为 待检查。这表明我们总共处理了 $O(n + D)$ 个三角形面。

只有在删除三角形面时，才会查询外接圆 $\{C(\triangle') \mid \triangle' \in \tau\}$；每次此类删除都会使用 Q_{new} 执行 $O(1)$ 次查询，因此，对 τ 的外接圆共有 $O(D)$ 次此类查询。查询 Q 时，会针对三角形 $\triangle$ 使用圆 $C(\triangle)$，因为 $\triangle$ 与 $F_{\geq 4}(G')$ 的一个面相邻。这些三角形中的每一个，要么与 $F_{\geq 4}(G)$ 的一个面相邻（即初始图 G），要么是 τ_{new} 中因另一个三角形被删除而必须考虑的 $O(1)$ 个三角形之一。第一种情况发生 $O(D)$ 次，因为 $F_{\geq 4}(G)$ 的所有面总共至多有 $O(D)$ 条边。第二种情况发生 $O(D)$ 次，因为每次删除三角形都会触发 $O(1)$ 次新查询，而我们会删除 $O(D)$ 个三角形。总共，我们对存储 Q 的数据结构进行了 $O(D)$ 次查询。由此可知，对存储 Q 和 $\{C(\triangle') \mid \triangle' \in \tau\}$ 的数据结构进行查询，总共需要 $O(D \log^4 D)$ 时间。

我们总结当前的状态：我们已在 $O(n + D\log^4 D)$ 时间内计算出 E' 个 $O(D)$ 边的集合 G，使得对于 $G' = G - E'$ 的所有三角形面 $\triangle$，外接圆 $C(\triangle)$ 不包含 $Q = V(F_{\geq 4}(G'))$ 中的任何点，也不包含与 $\triangle$ 相邻的三角形中的任何点。现在，我们从 G' 中移除两侧均为非三角形面的所有边，并将其添加到 E' 中。由于此操作不会改变 $Q = \bigcup_{f \in F_{\geq 4}(G')} V(f)$，也不会改变 τ，因此我们为完成任务而需要满足的条件未发生变化：每个三角形的外接圆内部都不包含 $V(F_{\geq 4}(G'))$ 中的任何顶点，也不包含其邻接三角形中的任何顶点。根据引理 A.2，我们另外删除了 $O(3D) = O(D)$ 条边，并将其添加到 E' 中。

我们已从输入平面直线图中计算出一个包含 $O(D)$ 条边的集合 E'，使其 $G - E'$ 满足引理 A.1 的假设。因此，$G - E'$ 的所有边都是 $DT(P)$ 边。随后，我们使用 Chew [23] 的算法，为 $G - E'$ 的各个面计算约束 Delaunay 三角剖分；这些面内部可能包含一些孤立点或连通分量。首先，我们花费 $O(n)$ 时间来检测 $G - E'$ 中哪些面不是三角形。由于 $G - E'$ 具有 $O(D)$ 个非三角形面，共包含 $O(D)$ 个顶点，因此，对所有非三角形面计算约束 Delaunay 三角剖分总共需要 $O(D \log D)$ 时间。

□