计算几何课程
大作业选题报告

2021 年 4 月 29 日

1 小组成员

王宣润 2020215219
李　哲 2019310957
李玮祺 2020215216

2 问题描述

我们欲解决的问题是，给定二维平面上若干个点，求一个最大宽度的，边方向平行于坐标轴的正方形或矩形空环带，使得这个环带中不含任何点 [1]。

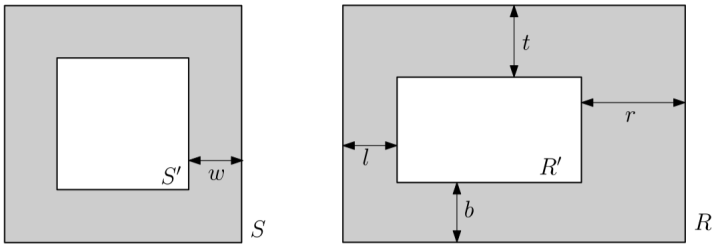

图 1: 正方形空环带和矩形空环带

正方形空环带的定义是，两个共中心的正方形套在一起，其宽度 w 为外面那个正方形 S 的边长减去里面那个正方形 S' 的边长，如图1左侧所示。矩形空环带的定义是，两个不相交的矩形 R, R' 套在一起，不必共中心，其宽度为内矩形四条边和外矩形四条边各自距离 t, r, l, b 的最小值，如图1右侧所示。图中灰色部分就是环带区域。注意为了让解有意义，内矩形/正方形内部或边界上、外矩形/正方形外部或边界上要至少有一个点。另外，正方形或者矩形都允许退化，即某条边延伸到无穷远，使得正方形或矩形退化成一条直条带或者一条 L 形的条带。

3 文献综述

最大空环问题是一个新兴的研究领域。之前有 [4] 第一个研究过最大空圆环的问题，并给出了时间复杂度为 $O(n^3logn)$ 的解法。最大矩形和正方形空环带问题也被 [5] 研究过，但该作者提出了一个错误的解法，他忽略了部分情况，错误地得到了一个 $O(n^2)$ 时间复杂度的解法。本次大作业我们想要实现的就是该问题最新最先进的解法，是今年发表在 Computational Geometry: Theory and Applications 期刊上的 [1]，它给出了针对求解最大宽度正方形空环带和矩形空环带问题的时间复杂度分别为 $O(n^3)$ 和 $O(n^2logn)$ 的算法，以下将详细介绍该论文的算法。

4 解决方案

4.1 符号定义

符号	意义
P	平面上的点集
n	$\|P\|$，即点集大小
p_i	平面点集中的第 i 个点
$x(p)$	点 p 的 x 坐标
$y(p)$	点 p 的 y 坐标

表 1: 符号表

我们将正方形空环带和矩形空环带的情况分开处理。

4.2 正方形空环带

首先可以证明 (证明略，见参考文献)，正方形空环带一定属于以下三种情况之一：

1. 外正方形的两条对边上各有一个点
2. 外正方形的两条邻边上各有一个点（L 形区域）
3. 外正方形的一条边上有一个点，其他三条边都是无穷（条带）

情况 3，空环带退化成了空条带，它的求解方式很简单，可以直接按照 x 坐标或者 y 坐标排序后扫描一遍，查找其最大的 gap，即为空条带的宽度。总时间复杂度 $O(nlogn)$。

情况 2，以每个点的 x 坐标和 y 坐标画垂直线和水平线，这些线之间有 n^2 个交点，对于每个这样的交点 o，找到以它为原点的第一象限里 (不含坐标轴)，x 坐标最小的点和 y 坐标最小的点，以此为限制找到 L 形区域，时间复杂度 $O(n^2logn)$[3]。

情况 1 较为复杂，先对所有点按 y 坐标排序，枚举外正方形顶边上的那个点 p_i 和底边上的那个点 p_j，然后以 $y = \frac{y(p_i)+y(p_j)}{2}$ 画线，这条线即是中线，正方形空环带的中心 c 一定在这条中线的一条线段上（因为需要顶边和底边都有 1 个点），且这条线段的左右端点会被 p_i 和 p_j 确定。设外正方形的边长为 $2r$，定义函数 $f_p(c)$ 表示当正方形空环带中心点位于位置 c 时，p 点对环带宽度的限制，即 $f_p(c) = ||p - c||_\infty$。

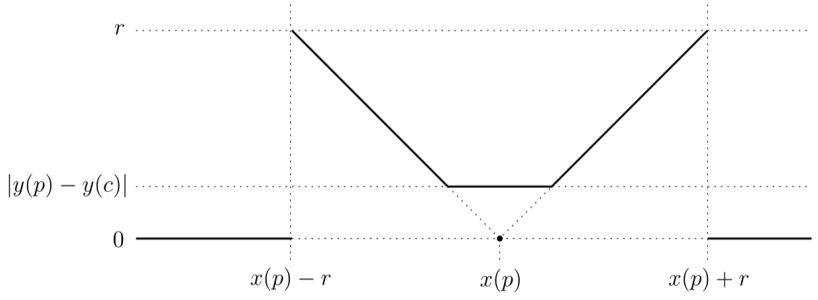

图 2: $f_p(c)$ 的函数图像

假设环带中心点 c 在中心线上的 $x = c$ 这个位置上，那么要使得 p 这个点不在空环带中，则环带宽度不能大于 $r - ||p - c||_\infty$，那么 f 的图像应如图2所示，图中中间平的一段的物理意义是此时 y 方向距离比 x 方向距离大，旁边为 0 的区域表示如果 p 落在了外正方形外，则其对结果没有限制。

那么，我们把所有 $p \in P$ 的 $f_p(c)$ 都求出来，然后求一个包络线，即

$$f(c) = \max_{p \in P} f_p(c)$$

求 $f(c)$ 的最小值点，即可知道 c 为何值时 $w = r - f(c)$ 最大。

求上述包络线的过程，我们可以发现上述每个点的图像都可以拆成三个部分，即斜率为 $1, 0, -1$ 的三个部分。将三个部分分开处理，再合并起来就能得到总体的包络线了。另外，由于一开始就把所有的点按照 x 和 y 分别排过序了，所以这里处理包络线时也不需要进行排序。包络线只会有 $O(n)$ 段，故这里处理包络线的过程是 $O(n)$ 的。加上之前枚举 p_i 和 p_j 的时间消耗，情况 1 处理的总时间复杂度是 $O(n^3)$ 的。

由于这三种情况可以囊括所有的可能情况，所以求解正方形空环带算法的整体时间复杂度是 $O(n^3)$。

4.3 矩形空环带

矩形的情况略为复杂，因为每个方向的宽度都不一样。首先可以证明

- 内矩形的每条边上要么至少有一个点，要么是无穷
- 外矩形的每条边上要么至少有一个点，要么是无穷

这是因为如果外矩形的某条边上没有点，那可以向外平移这条边，从而使其碰到一个点或者到无穷远，而此时该矩形空环带是不会更窄的；如果内矩形的某条边上没有点，那也可以向内平移这条边，从而碰到一个点，此时矩形空环带也不会变窄。

由于矩形空环带的宽度由其最窄的部分确定，所以那些宽的方向可以扩展，使得其每个方向的宽都是 w。如图3，若扩展前矩形顶边的宽度最小且为 w，此时该环带的宽度由内外矩形的顶边确定。不失一般性可以只考虑这一种情况。

接着我们可以考虑枚举外矩形顶边的点 t 和底边的点 b，继续寻找空环带的最大宽度 w。注意到 w 满足单调性：如果给定一对 t 和 b，且已知 $w_2 > w_1$ 且 w_2 是可行的，则 w_1 也是可行的。

怎样判决一个 w 值是可行的呢？分为以下两个步骤：

1. 判断顶边下方和底边上方的 w 宽度内的合法点范围，x 是上述确定的 $t(b)$ 点的横坐标，$|y_1 - y_2| = w$，如左图可以找到竖直线段 $(x, y_1) - (x, y_2)$ 往左平移遇到的第一个点 $q_{1t}(q_{1b})$ 和往右平移遇到的第一个

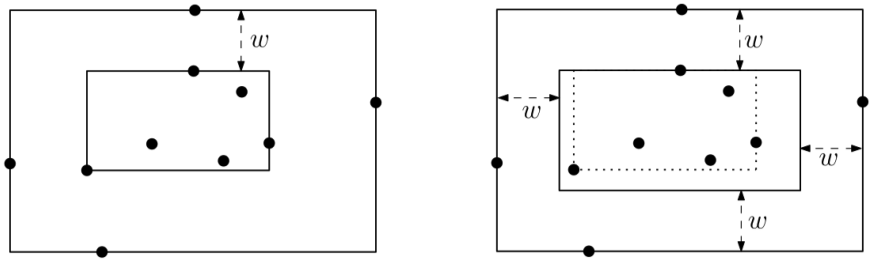

图 3: 扩展矩形空环带，使其四边的宽度相等

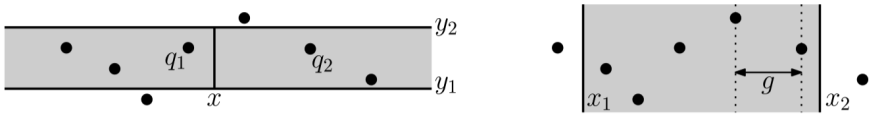

图 4: 判断 w 值是否可行的两个步骤图示

点 $q_{2t}(q_{2b})$。由于我们需要确保 $t(b)$ 点是在外矩形上，所以 $t, b, q_{1t}, q_{2t}, q_{1b}, q_{2b}$ 这几个点需要满足合适的位置关系，即 q_{XX} 不能出现在 $x(t)$ 到 $x(b)$ 之间；且 q_{XX} 实际上卡住了这个矩形，因为如果空环带宽为 w，那么这个条带里是不能有点的，为此矩形最多延伸到 q_{XX} 处。

2. 以 q_{XX} 和 t、b 的 x 坐标作为边界参考量，寻找在 t 和 b 之间的这些点的最大 x 坐标 gap，如果 gap 值大于 w，说明宽度为 w 是可行的。这是因为可以把矩形空环带的左右两边卡在这个 gap 里，而这个 gap 里在 $y(t)$ 和 $y(b)$ 之间是一定没有任何一个点的。

下面介绍上述两个步骤的具体实现。

步骤 1 实际上是一个经典问题：segment-dragging，即给定一条二维平面上的线段，让其沿着其垂线方向扫描，求扫描遇到的第一个点。这个算法可以以 $O(nlogn)$ 的时间预处理，然后每次 $O(logn)$ 查询。但由于其实现难度较大，在我们的实际处理中预计使用 $O(nlogn)$ 预处理，$O(log^2n)$ 查询的较简单的方式。具体则是 range-tree 的应用：所谓 range-tree，是把所有的点按照坐标顺序放到一棵二叉树的叶子上，然后树的非叶子节点记录了这个区间 (它的子孙们) 的所有信息。那么，我们按照 P 中所有点的 x 坐标建立 range-tree，然后将这些点的 y 坐标依次维护在这棵 range-tree 的节点上。上面的问题实际上转化为，在 x 区间中，找比 y 大或小的第一个值。那么在 range-tree 上，我们可以找到 $O(logn)$ 个节点，代表查询的 x 区间，然后在每个节点上进行一次二分查找，所以每次查询总时间复杂度是 $O(log^2n)$ 的。预处理由于需要排序，且每个点只会出现在 $O(logn)$(树高) 个节点里，所以时间复杂度是 $O(nlogn)$ 的。

步骤 2 也可以用 range-tree 来实现。在 range-tree 的每个节点里保存当前区间的最大 gap 值，每次查询也只会访问 $O(logn)$ 个节点。故查询的时间复杂度也是 $O(logn)$ 的。

具体实现中，我们不必每次都去枚举外矩形顶边的点和底边的点，再从头构建数据结构。我们首先枚举顶边的点 t，由于内外矩形的顶边都有点，故枚举 t 后，w 最多只有 n 个取值。然后依次枚举底边的点 b，在枚举 b 的过程中，不从头开始枚举 w，而是直接递增 w 的大小：这是由于前述 w 的单调性，如果较大的 w 能够满足，则较小的 w 也一定能够满足，而不可能在递增 w 的过程中发现本来不合法，w 变大之后反而合法。实际上，这样的操作虽然枚举了 t, b, w，但 w 是单调的这个性质使得实际的运算次数是 $O(n^2)$ 级别的。内层每次判断是否合法的时间复杂度是 $O(log^2n)$ 的，故总时间复杂度是 $O(n^2log^2n)$。论文提出的最优时间

复杂度是 $O(n^2 logn)$，是由于其在 segment-dragging 问题中使用了更快更复杂的算法 [2]。

5 技术难点

我们认为，本项目的技术难点主要在以下几个方面。

5.1 分情况讨论复杂

无论是正方形还是矩形的空环带区域，其都分为了多种情况，对于每种情况的处理可能不尽相同。恰当而全面地处理这些情况是一个挑战。

5.2 算法和数据结构能力要求较高

在求解正方形环带区域时，要求对排序、归并等操作理解透彻；在求解矩形环带区域时，需要对 range-tree 的相关操作和二分查找算法理解深刻。如果有余力，我们打算实现原文中 $O(logn)$ 的针对 segment-dragging 问题的查询操作，我们查阅了相关论文，发现这个算法的实现难度很高，如果实现，也是个不小的挑战。

6 组员分工

本分工为初步协商的分工结果，后续可能会根据各种情况而发生变更。

王宣润 算法实现

李哲 UI 实现和美化

李玮祺 文档的撰写、测例的收集、文献调研

7 参考资料

参考文献

[1] Sang Won Bae, Arpita Baral, and Priya Ranjan Sinha Mahapatra. Maximum-width empty square and rectangular annulus. *Computational Geometry*, 96:101747, 2021.

[2] Bernard Chazelle. An algorithm for segment-dragging and its implementation. *Algorithmica*, 3(1):205–221, 1988.

[3] Siu-Wing Cheng. Widest empty l-shaped corridor. *Information Processing Letters*, 58(6):277–283, 1996.

[4] José Miguel Díaz-Báñez, Ferran Hurtado, Henk Meijer, David Rappaport, and Joan Antoni Sellarès. The largest empty annulus problem. *International Journal of Computational Geometry & Applications*, 13(04):317–325, 2003.

[5] Priya Ranjan Sinha Mahapatra. Largest empty axis-parallel rectangular annulus. *Journal of Emerging Trends in Computing and Information Sciences*, 3(6), 2012.