Planar 2-Center Problem

袁泰凌, 卢嘉铭, 梁盾

[1] 清华大学计算机系

1. 问题描述

如图 1所示，某移动网络运营商想用 p 个基站覆盖某个城市的 n 个服务点。每个基站的覆盖范围是圆盘，覆盖范围越大，造价就越高，因此运营商想用尽量小的基站完成覆盖。为了方便批量生产和搭建，所有基站的大小相同。

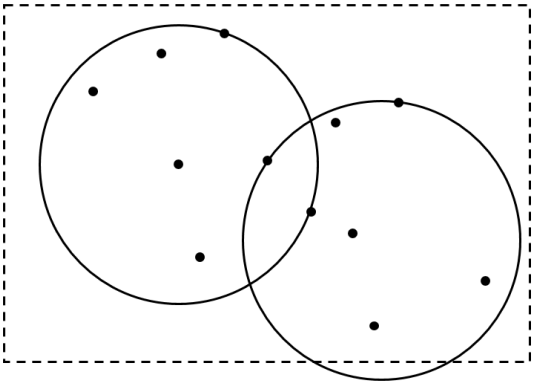

图 1. 背景问题

该问题可以进行如下抽象：设 S 为平面上包含 n 个点的点集，用半径尽可能小的 p 个相同的闭圆盘完全覆盖 S。

上述问题也称为平面内 p 圆覆盖问题。

当 p 作为问题的输入时，该问题是一个 NP 完全问题。那么可以认为当 p 大小固定时，问题复杂度是与 p 指数地相关。对于 p 固定的 p-center 问题，Hwang 1993 [1] 提出了一种复杂度为 $n^{O(\sqrt{p})}$ 的算法。$p = 1$ 时的问题被称为 “最小包围圆问题”，Megiddo 1983[3] 提出了线性时间解法。$p = 2$ 时的问题即 “2-center problem”，Sharir 1997[4] 提出了在并行处理机上 $O(nlog^9n)$ 的解法，需要使用 $O(n)$ 个并行处理器。

本实验实现了上述的近线性时间算法。

2. 算法

Sharir 1997[4] 提出的方法可以分为两个部分：

一个部分是对于给定的半径 r，判定是否存在两个闭圆盘能够覆盖 n 个点。这个判定问题被称为 “2DC problem”。对于这个判定问题，论文中给出算法的复杂

度为 $O(nlog^3n)$。在这一部分中，需要频繁对圆进行求交操作，论文中使用 $K(p)$ 树的方法动态地维护多个圆的交集；

第二个部分是基于上述判定问题，对 r 进行参数搜索，从而获得问题的解 r。

2.1. 参数搜索

Sharir 1997[4] 在并行处理该问题时，使用的参数搜索方法 [2] 比较复杂，而且现在的计算机是无法做到 O(n) 个处理器并行运行的。因此，我们改为在精度 ϵ 下求解 2-center problem。

利用 2DC problem，可以按如下方法在精度 ϵ（指相对于最远的两个点的距离的相对精度）下求解 2-center problem:

先算出包围盒对角线距离 L，在 [0, L] 范围内对 r 进行二分查找。在 $O(log\frac{1}{\epsilon})$ 步后，查找范围小于 $\frac{\epsilon L}{2}$，即可得到最终结果。

最终的总体时间复杂度是 $O(nlog\frac{1}{\epsilon}log^3n)$。

2.2. 动态维护平面结构

使用 $K(p)$ 树动态地维护一组闭圆盘的交集。如图 2 $Br(p)$ 表示 p 点对应的半径为 r 的圆盘，$Br^+(p)$ 表示圆盘及圆盘的上方部分，$Br^-(p)$ 表示圆盘及圆盘的下方部分。对于点集 S 的某个子集 P，我们维护 $K^+(p) = \bigcap_{p\in P} Br^+(p)$ 以及 $K^-(p) = \bigcap_{p\in P} Br^-(p)$。

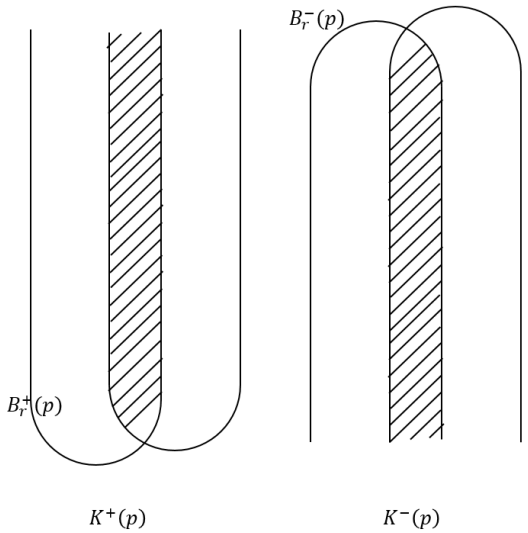

图 2. $K^+(p)$ **与** $K^-(p)$

下面说明如何维护 $K^+(p)$。对于 $K^-(p)$ 的情况可以对称地处理。

将 S 中的所有点作为 $K^+(P)$ 叶子节点，构造一个最小高度二叉树。$K^+(P)$ 的内部节点 v 表示了对应的 P 的子集 P_v 的交。$K^+(P)$ 的内部节点中包含下列信息：

1. 这个节点的 x 范围。

2. 这个节点的对应的左右孩子的交的坐标。

对于某个 $K^+(P)$，可以通过 $O(logn)$ 的时间判定某个点 p 是否在 $K^+(P)$ 中。过程如下：找到树的根节点，判断节点中的 x 范围是否为空，若是，则返回否。否则判断 p 的 x 坐标是否在 x 范围中，若否，则返回否。否则比较节点存储的交点 q 的 x，若 $p.x < q.x$，则进入左支对应的子树，若 $p.x > q.x$，则进入右支对应的子树，若 $p.x = q.x$，若 $p.y >= q.y$，则返回是，否则返回否。

对于某个 $K^+(P)$，可以通过 $O(log^2 n)$ 的时间插入一个点或删除一个点。当插入一个点时，找到这个叶子节点对应的从根节点到它本身的路径，自底向上地进行更新。更新每个内部节点时，求该节点对应的左右孩子的交。讨论左右孩子对应的分支交点及相邻的弧的相交情况，可以不断排除不可能的分支缩小规模。最终每次更新内部节点的复杂度为 $O(logn)$。所以每次插入、删除操作的总复杂度为 $O(log^2 n)$。

对于某个 $K(P)$ 树求交时，需要对 $K^+(P)$ 和 $K^-(P)$ 求一次交。求交时，考虑 $K^+(P)$ 中的所有断点（即交集的弧上的交点），通过判断断点是否在 $K^-(P)$ 中，可以将断点对应到 $K^-(P)$ 的某段弧上。通过类似于 $K^+(P)$ 内部节点更新的方法讨论，可以将交点确定到 $K^+(P)$ 的左子树或右子树，不断缩小问题规模。最终可以判断是否与 $K^+(P)$ 相交，并得到与 $K^+(P)$ 的交点。总的复杂度为 $O(log^2 n)$。

2.3. 2DC **问题**

对于给定的 r，判定是否存在两个圆覆盖点集 S。假定两个圆为 D_1, D_2，它们的圆心分别为 c_1, c_2。可以分为如下三种情况讨论：

1. $|c_1c_2| > 3r$
2. $|c_1c_2| < r$
3. $r < |c_2c_2| < 3r$

2.3.1. $|c_1c_2| > 3r$ **的情况**

在该种情况下我们进行如下的考虑：

δ 是一个足够小的角度，比如可以取 $1°$，取 $j = 0, 1, ..., \frac{2\pi}{\delta}$ 将原来的点集旋转 $j\delta$，在某个角度上，一定存在一条竖直线将两个圆完全分开。如图 3所示。

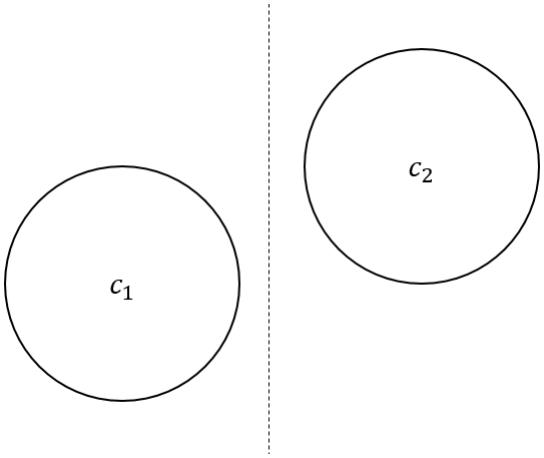

图 3. $|c_1c_2| > 3r$

在这个角度下，我们将点集 S 按 x 坐标排序，从左到右扫描，记扫描的当前点为 q。q 左边的点作为 S_L，q 右边的点做为 S_R。在扫描每一个点时，更新 $K(S_L)$ $K(S_R)$ 并判定 S_L S_R 是否有交。当 $K(S_L)$ $K(S_R)$ 均不为空时，说明存在满足条件的两个圆。

每次更新 $K(S_L)$ $K(S_R)$，两棵树各需要增加一个点，删除一个点，复杂度为 $O(log^2(n))$。各需要判定一次是否有交，复杂度为 $O(log^2(n))$。故这类情况总的复杂度为 $O(nlog^2(n))$。

2.3.2. $r < |c_1c_2| < 3r$ 的情况

通过一定角度的旋转之后，取一些固定的竖直线，一定能够找到一条 λ，使得圆心 c_1 在 λ 的左侧，同时 λ 左侧的所有点都在 D_1 内。当两圆不相交时，容易知道上述情况成立。当两圆相交时，我们假定交点为 v_1, v_2。如图 4所示。当 v_1 v_2 右侧时，取过 v_1 的竖直线将 S 分为两部分，其中 $S_L \subset D_1$，$S_R \subset D_2$。通过第一种情况可以判定。只需考虑 v_1 v_2 左侧时。由于 S 中的点 x 最大值和最小值的差一定在 5r 以内，那么 $x(v_1) - x(c_1) > 0.4r$。在 v_1 c_1 之间的直线即可满足条件。

通过这条竖直线 λ，将 S 分为 S_L S_R。我们计算 S_L 的 $K(S_L) = \bigcap_{p \in S_L} B_r(p)$。对于 S_R 中的每个点 p，计算 $B_r(p)$ 与 $\partial K(S_L)$ 的交，最多有两个交点。我们将 $\partial K(S_L)$ 中的点和这些交点进行排序放入一个列表 Γ。

对于上述的列表 Γ 进行遍历，对与 Γ 中的某个点 v，将圆心 c_1 设在 v 处，所有不被这个圆覆盖的点组合为集合 $S'(c_1)$。若 $S'(c_1)$ 能被某个圆覆盖，说明存在两个半径 r 的圆盘可以覆盖 S。若遍历所有的 λ 和对应的 Γ 之后，仍然找不到满足的情况，说明不存在这样的两个圆盘。

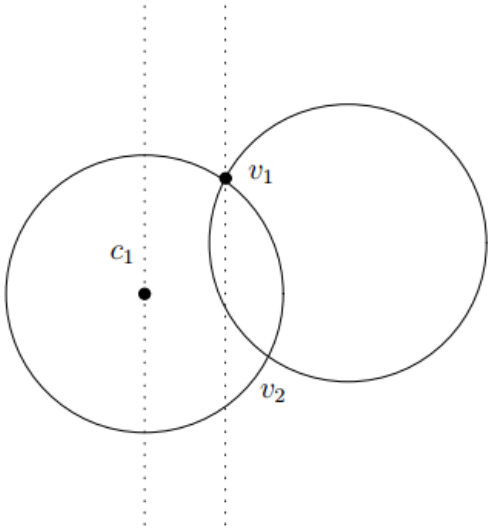

图 4. $r < |c_1c_2| < 3r$

2.3.3. $|c_1c_2| < r$ **的情况**

这种情况下，$D_1 \bigcap D_2$ 的面积最少为 $r^2\sqrt{\frac{3}{2}} \approx 0.866r^2$. 点集 S 可以被一个边与坐标轴平行，半径为 3r 的正方形 R 覆盖。我们在正方形 R 中生成有限个点，其中至少有一个点是位于 $D_1 \bigcap D_2$ 中。

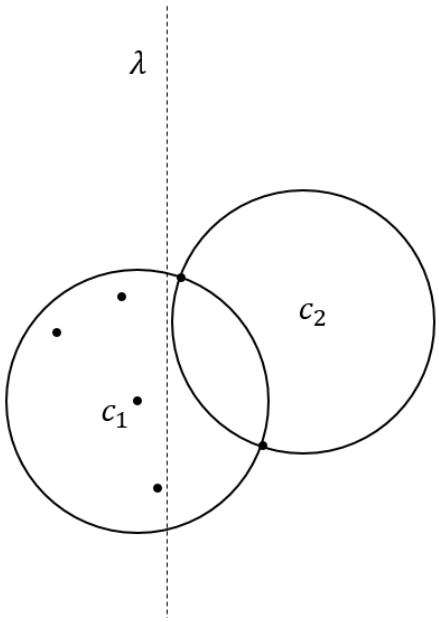

图 5. $|c_1c_2| > r$

参考图 5。设这个点为 z，用通过点 z 的水平线可以将点集 S 分成两部分 Q^+, Q^-。将 Q^+, Q^- 中的点按照与 z 连线的夹角排序，（Q^+ 中的点按照顺时针方

向，Q^- 中的点按照逆时针方向)。论文中证明了，一定存在一个 Q^+ 的前缀 S_L^+ 和 Q^- 的前缀 S_L^-，使得 $S_L^+ \cup S_L^- \subset D_1$ 且 $S\ (S_L^+ \cup S_L^-) \subset D_2$。

事实上，对与 $D_1 \bigcap D_2$ 中的所有点，都一定可以找到某种划分使得上述情况成立。所以，基于取得的 $D_1 \bigcap D_2$ 中的点 z，在 Q^- 和 Q^+ 中进行搜索。当找到一种前缀的划分使得上述情况成立时，说明存在两个半径 r 的圆盘可以覆盖 S。当找不到这种划分时，说明不存在这样的两个圆盘。总体复杂度为 $O(nlog2n)$

3. 实现

在我们的实现中，有如下两处与论文所述不同。

3.1. 参数搜索

Sharir 1997 在并行处理该问题时，使用的参数搜索方法 [2] 比较复杂，而且现在的计算机是无法做到 O(n) 个处理器并行运行的。因此，我们改为在精度 ϵ 下求解 2-center problem。

利用 2DC problem，可以按如下方法在精度 ϵ（指相对于最远的两个点的距离的相对精度）下求解 2-center problem:

先算出包围盒对角线距离 L，在 [0, L] 范围内对 r 进行二分查找。在 $O(log\frac{1}{\epsilon})$ 步后，查找范围小于 $\frac{\epsilon L}{2}$，即可得到最终结果。

最终的总体时间复杂度是 $O(nlog\frac{1}{\epsilon}log^3n)$。

3.2. $|c_1c_2| < r$ 的情况

论文中对于 $|c_1c_2| < r$ 的情况没有进行旋转。事实上，对于某个可行的半径 r，对与 $D_1 \bigcap D_2$ 中的所有点，都一定可以找到某种划分使得上述情况成立的条件还需要保证 z 的高度（y 方向）在两个交点的高度之间。论文中未经过旋转，有可能两个交点高度差很小，候选 z 的分布需要无穷密。但通过旋转一定角度后，一定可以满足该条件，使得能够找到一个可行的划分。

4. 实验

图 6, 图 7, 图 8列出了在一些输入点集上的实验结果。

5. 总结

我们实现了有限精度下的两圆覆盖问题的算法，时间复杂度为 $O(nlog\frac{1}{\epsilon}log^3n)$。在实现算法的过程中，我们发现了一些问题。

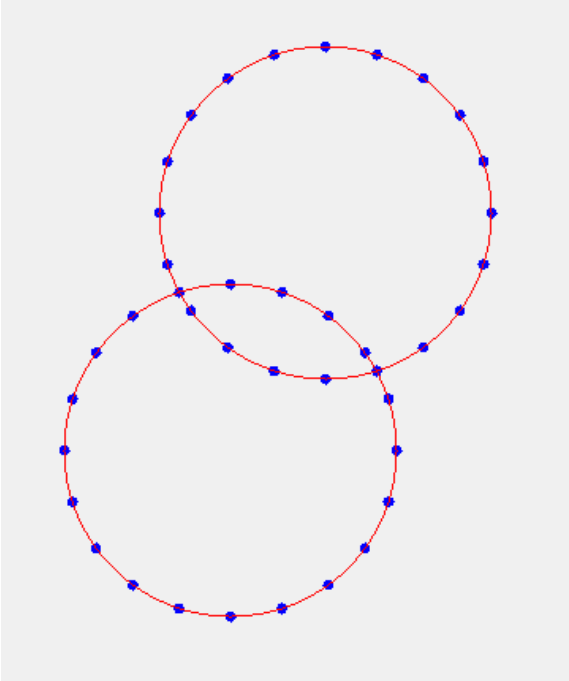

图 6. 圆

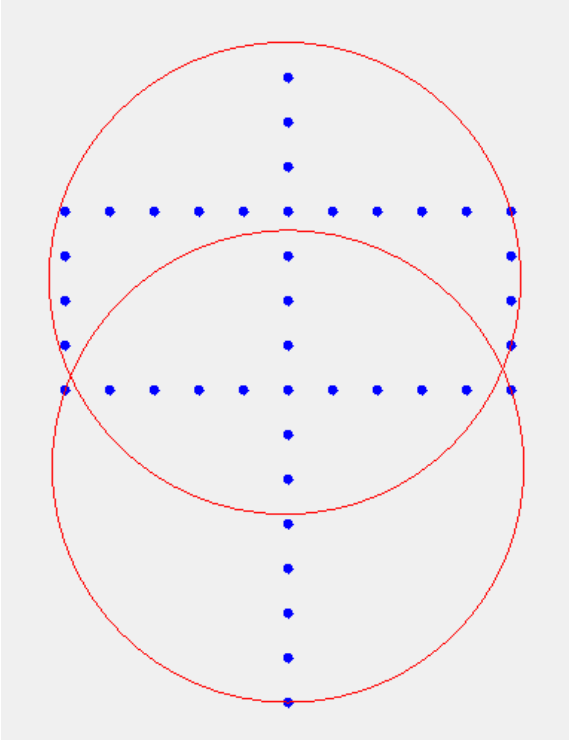

图 7. 方形

5.1. 精度问题

在我们的实现中存储点采用了 double 类型。由于浮点数的精度有限，在运算时会出现误差的累积，有可能出现精度导致的错误结果。在 $K(P)$ 树的维护和求

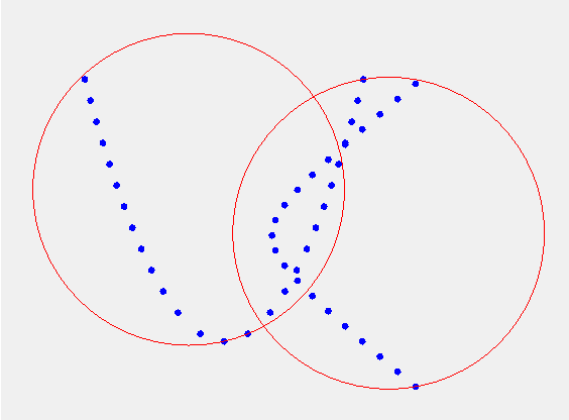

图 8. 抛物线

交中，存在大量的浮点数比较和四则运算操作。要保证正确的结果需要对各类操作误差大小进行细致的分析。由于点集 S 是确定的，通过上述误差分析以及通过点集 S 的信息对误差进行修正，可以优化目前的算法实现。

5.2. 运行时间

论文中提出算法在理论上的复杂度是近线性的。但实际的测试结果中，该算法的运行速度并不快。原因是对于每个情况算法需要测试多个角度，多条直线，直到找到一种满足的情况，或是遍历完所有的情况。虽然这些角度和直线的数目都是常数，但最终综合起来，这个算法的常数项是非常大的。在测试点集数量不大的情况下，该算法的效率并没有明显的运行速度上的优势。

Sharir 给出的算法在理论上是近线性时间的复杂度。但在实际的实现中，$O(n)$ 的并行处理机目前并不存在，需要考虑复杂的误差分析和精度处理，同时算法的常数项非常大，在小测试点集的下没有运行速度的优势。

参考文献

[1] R. Z. Hwang, R. C. T. Lee, and R. C. Chang. The slab dividing approach to solve the euclideanp-center problem. Algorithmica, 9(1):1–22, 1993.

[2] Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. In Foundations of Computer Science, 1981. Sfcs '81. Symposium on, pages 399–408, 1981.

[3] Nimrod Megiddo, Nimrod Megiddo, Nimrod Megiddo, and Nimrod Megiddo. Linear-time algorithms for linear programming in r3 and related problems. In Foundations of Computer Science, 1982. Sfcs '08. Symposium on, pages 329–338, 1982.

[4] Micha Sharir. A near-linear algorithm for the planar 2-center problem. In Twelfth Symposium on Computational Geometry, pages 106–112, 1996.