arXiv:2601.22609v1 [cs.CG] 30 Jan 2026

# 凸位置中心圆盘图中的支配集计算*

Anastasiia Tkachenko† Haitao Wang‡

**摘要**

给定平面上的 $P$ 个 $n$ 点，以及一个以这些点为中心的圆盘集合，圆盘图 $G(P)$的顶点集为 $P$；若两个顶点对应的圆盘相交，则这两个顶点之间有一条边。我们研究 $G(P)$中的支配集问题，并考虑$P$中的点处于凸位置这一特殊情形。一般圆盘图中的该问题是 NP-hard。然而，在凸位置假设下，我们给出了该问题的首个多项式时间算法。具体而言，我们设计了一个运行时间为 $O(k^2n\log^2 n)$的算法，其中 $k$ 表示最小支配集的大小。对于加权版本，即每个圆盘都关联一个权重，目标是计算总权重最小的支配集，我们得到一个运行时间为 $O(n^5\log^2 n)$的算法。

关键词： 圆盘图、支配集、凸位置

## 1 引言

设 $P=\{p_1,\ldots,p_n\}$ 为平面中 $n$ 个点的集合，其中每个点 $p_i\in P$被赋予半径$r_{p_i}\geq 0$。令 $D_{p_i}$ 表示以 $p_i$ 为中心、半径为 $r_{p_i}$的圆盘。圆盘图 $G$（$P$）以 $P$为其顶点集；若两个圆盘 $D_{p_i}$与 $D_{p_j}$相交，即 $|p_ip_j|\leq r_{p_i}+r_{p_j}$，则 $p_i$与$p_j$之间存在一条边，其中$|p_ip_j|$ 为 $p_i$ 与$p_j$的欧几里得距离。圆盘图（包括所有圆盘半径相同的单位圆盘情形）自然出现在众多应用领域中，例如无线传感器网络（具有异构传输范围的设备在彼此范围内时进行通信） [3,18,24,25],监控网络 [32], 最优设施选址 [10,34], 等。

圆盘图上的许多经典问题仍然是NP-hard的 [10]，并且已经发展出近似算法，例如顶点覆盖 [14,22,33], 独立集 [14], 反馈顶点集 [4], 支配集 [17], 等。一个例外是，在单位圆盘图中寻找最大团可以在多项式时间内求解 [10,13,15]。但一般圆盘图上的团问题能否在多项式时间内求解，长期以来一直是一个开放问题；参见 [6,19] 以了解该问题的一些最新进展。

我们研究圆盘图中的支配集问题。 $G(P)$的支配集 是满足以下条件的子集 $S\subseteq P$： $G(P)$中的每个顶点要么属于 $S$ ，要么与 $S$中的某个顶点相邻。支配集问题 是求基数最小的支配集。在加权情形下， $P$ 中的每个点都有一个权重，目标是找到总权重最小的支配集。即使在单位圆盘图中，该问题也是 NP-hard 的 [10]。已有近似算法，例如[2,11,17,23]。

*本文的预印本将发表于第17届拉丁美洲理论信息学研讨会论文集（*LATIN 2026*）。

†卡勒特计算学院，犹他大学，美国犹他州盐湖城，邮编 84112。anastasiia.tkachenko@utah.edu
‡卡勒特计算学院，犹他大学，美国犹他州盐湖城，邮编 84112。haitao.wang@utah.edu

### 1.1 凸位置设定与以往工作

鉴于上述困难性结果，探索可能支持高效算法的结构化情形尤为重要。本文研究圆盘图中的支配集问题，考虑凸位置 情形，其中 $P$ 中的所有点都处于凸位置，也就是说， $P$ 中的每个点都是 $P$的凸包的一个顶点。这种情形可用于建模沿周界的部署（例如围栏、海岸线或传感器环），而且尤为重要的是，它揭示了强有力的结构关系。

在这一凸位置设定下，Tkachenko 和 Wang [31]研究了单位圆盘情形下的问题。具体而言，他们为加权支配集问题给出了一个 $O(n^3\log^2 n)$ 时间算法，并为非加权情形给出了一个 $O(kn\log\ n)$ 时间算法，其中 $k$ 是最小支配集的大小。

凸位置下的其他几何问题也得到了研究。对于独立集问题，即寻找 $G(P)$ 的顶点的一个最大子集，使得任意两个顶点之间都没有边， [31] 在凸位置设定下给出了一个 $O(n^{7/2})$ 时间算法。平面点集上的经典 $k$-中心问题，是寻找 $k$ 个半径最小的全等圆盘来覆盖所有点。该问题是 NP-hard。Choi、Lee 和 Ahn [9] 研究了凸位置设定下的该问题，并提出了一个 $O(n^3\log n)$ 时间算法。对于该问题的离散版本，即要求圆盘中心位于给定点集中的情形，Tkachenko 和 Wang 在 $O(n^2\log^2 n)$ 时间内解决了该问题 [31]。给定平面上的一个点集和一个数值 $k$，离散化问题是寻找 $k$ 个点，使它们之间的最小两两距离最大化。该问题同样是 NP-hard [34]。Singireddy、Basappa 和 Mitchell [29] 研究了凸位置设定下的该问题，并给出了一个$O(n^4k^2)$ 时间算法。在 [31]中，后来推导出了一个运行时间为 $O(n^{7/2}\log n)$ 的改进算法。

即使是一般情形下已可在多项式时间内求解的问题，在凸位置设定中也受到关注，例如 Aggarwal、Guibas、Saxe 和 Shor 提出的经典线性时间算法，可用于为处于凸位置的一组二维点构造 Voronoi 图 [1] 。关于凸位置设定下的其他研究，请参阅 [5,7,8,21,27,30]。

### 1.2 我们的结果

我们研究凸位置设定下圆盘图中的支配集问题。尽管单位圆盘情形已有多项式时间算法 [31], ，但允许圆盘半径各不相同会带来重大挑战。例如，虽然三十多年前就有人提出了计算单位圆盘图中最大团的多项式时间算法 [10], ，但对于一般圆盘图，这个问题能否在多项式时间内求解仍是未知的。

尽管如此，我们尝试将 [31] 中的技术扩展到一般圆盘图情形。这绝非易事，因为单位圆盘情形中的许多性质不再成立。即便如此，我们仍发现了新的结构性观察，并开发出新颖的算法技术。因此，我们证明了凸位置设定下圆盘图中的支配集问题可以在多项式时间内求解。

具体而言，对于加权情形，我们提出了一种运行时间为 $O(n^5\log^2 n)$ 的算法。此外，给定规模上限 $k$，我们可以在 $O(k^2n^3\log^2 n)$ 时间内，计算一个规模至多为 $k$ 的最小权支配集（如果存在）。我们的方法是一种动态规划算法，它利用了我们证明的一个关键结构性质，我们称之为直线可分性质。该性质使我们能够将原问题分解为适合动态规划求解的更小子问题。

对于非加权情形，我们采用贪心策略设计了一种更高效的算法，其运行时间为 $O(k^2n\log^2 n)$，其中 $k$ 表示最小支配集的规模。特别地，当 $k = O(1)$ 时，我们的算法运行时间为 $O(n\log^2 n)$。

尽管此前已有研究探讨凸位置设定下的单位圆盘图，但据我们所知，关于凸位置下的一般圆盘图，相关研究仍然很少。因此，我们希望本研究成果能够推动这一方向的进一步研究。

**概述**。本文其余部分安排如下。第2节介绍记号，第3节介绍加权算法和非加权算法所依赖的一些结构性质。加权情形和非加权情形的算法分别在第4 节和5节给出。

## 2 记号

除了第 1 节中已经定义的记号外，我们还引入一些将在全文中使用的记号，例如 $P$、 $n$、 $G$（$P$）以及 $D_{p_i}$。

对于平面中的任意两个点 $p$ 和 $q$，我们用 $|pq|$ 表示它们之间的（欧几里得）距离。我们还用 $pq$ 表示连接它们的线段。

令 $\mathcal{H}$（$P$）表示 $P$ 的凸包。我们假设 $P$ 中的点处于凸位置。因此，$P$ 可以看作围绕 $\mathcal{H}$（$P$）的循环序列。具体而言，令 $P = \langle p_1、 p_2、 \ldots\ldots、 p_n\rangle$ 表示沿 $\mathcal{H}$（$P$）按逆时针顺序排列的点的循环列表。我们用子列表指代 $P$ 的一个连续子序列。如果多个子列表的连接仍是一个子列表，则称它们是连续的。对于 $P$ 中的任意两个点 $p_i$ 和 $p_j$，我们将 $P[i, j]$ 定义为 $P$ 中从 $p_i$ 沿逆时针方向到 $p_j$（包括两端点）的子列表，即如果 $i \le j$，则为 $P[i, j] = \langle p_i、 p_{i+1}、 \ldots\ldots、 p_j\rangle$；否则为 $P[i, j] = \langle p_i、 p_{i+1}、 \ldots\ldots、 p_n、 p_1、 \ldots\ldots、 p_j\rangle$。注意，如果 $i = j$，则 $P[i, j] = \langle p_i\rangle$。我们还将 $P$（$i$、 $j$] 表示 $P[i, j]$ 中排除 $p_i$ 后的子列表，其他变体同理，例如 $P[i, j)$ 和 $P$（$i$、 $j$）。

对于任意两个点 $p_i$、 $p_j \in P$，我们定义 $|D_{p_i}D_{p_j}| = |p_ip_j| - (r_{p_i} + r_{p_j})$，并将其称为 $D_{p_i}$与 $D_{p_j}$之间的圆盘距离 。如果这两个圆盘相交，则 $|D_{p_i}D_{p_j}| \le 0$；否则， $|D_{p_i}D_{p_j}| > 0$。

对于任意子集 $P' \subseteq P$，令 $\mathcal{D}(P') = \{D_{p_i} \mid p_i \in P\}$。有时我们也称 $\mathcal{D}(P)$ 的圆盘为输入圆盘。我们强调，尽管输入圆盘的中心，即 $P$中的点，处于凸位置，但圆盘本身未必如此；也就是说，某个输入圆盘 $D_{p_i}$ 可能位于 $\mathcal{D}(P)$ 中所有圆盘的凸包内部。

## 3 结构性质

我们首先考察 $G(P)$ 中支配集的结构性质。我们针对加权情形介绍这些性质，它们同样适用于非加权情形。

对于 $P$ 的子列表 $\alpha$，如果对所有点 $p \in \alpha$，圆盘 $D_{p_i}$ 都与 $D_p$ 相交，我们就称点 $p_i \in P$ 支配 $\alpha$。对于两个点 $p_i$、 $p_j \in P$，如果 $D_{p_i}$ 与 $D_{p_j}$ 相交，我们也称 $p_i$ 支配$p_j$（同理，$p_j$ 支配 $p_i$）。

设 $S \subseteq P$ 是 $G$（$P$）的一个支配集。令 $\mathcal{A}$ 为 $P$ 的一个划分，其中各子列表均非空且两两不相交，并且对于每个子列表 $\alpha \in \mathcal{A}$， $S$ 中都存在一个支配 $\alpha$ 的点。分配是一个映射 $\phi$： $\mathcal{A} \to S$，它将每个子列表 $\alpha \in \mathcal{A}$ 恰好分配给一个点 $p_i \in S$，使得 $p_i$ 支配 $\alpha$。对于每个 $p_i \in S$，我们将 $p_i$ 的组记作 $\mathcal{A}_{p_i}$，定义为在 $\phi$ 下分配给 $p_i$ 的所有子列表$\alpha \in \mathcal{A}$ 的集合。根据上下文， $\mathcal{A}_{p_i}$ 也可以指 $P$ 的所有子列表中的点集。例如，如果点 $p_j$ 位于 $\mathcal{A}_{p_i}$ 的一个子列表中，我们可以写作 “$p_j \in \mathcal{A}_{p_i}$”。根据定义， $S$ 中所有点的组两两不相交，它们共同构成 $\mathcal{A}$ 的一个划分，也构成 $P$ 的一个划分。

分配 $\phi$： $\mathcal{A} \to S$ 是直线可分的，当对于任意两个点 $p_i$， $p_j \in S$，都存在一条直线 $\ell$，使得 $\mathcal{A}_{p_i}$ 中的点完全位于直线 $\ell$的一侧，而 $\mathcal{A}_{p_j}$ 中的点位于另一侧。

以下引理证明了线可分性质。正如我们将看到的，这一性质对我们的支配集算法至关重要。

**引理 1.** 假设 $S$ 是 $G(P)$的一个最优支配集。那么，存在 $P$ 的一个划分 $\mathcal{A}$ 以及一个线可分配 $\phi$: $\mathcal{A} \to S$，使得 *(1)* 对每个点 $p_i \in S$， $p_i \in \mathcal{A}_{p_i}$，即组$\mathcal{A}_{p_i}$ 包含其自身 $p_i$；*(2)* $\mathcal{A}$中任意两个相邻子列表都被分配给 $S$中的不同点。

证明. 我们先说明如何构造划分 $\mathcal{A}$ 和分配 $\phi$，然后论证$\phi$ 是直线可分的。

构造 $\mathcal{A}$ 和 $\phi$。令 $\mathcal{VD}(S)$ 为点集 $S$的加性加权沃罗诺伊图，其中每个点 $p \in S$ 的权重为 $-r_p$；也就是说，任意点 $q$ 与$p$之间的加权距离 $d_p(q)$定义为 $|pq| - r_p$。令 $R(p)$表示 $\mathcal{VD}(S)$中点 $p$的沃罗诺伊单元。注意， $R(p)$是星形的，且包含 $p$ [16, 28]。

由于 $S$ 是一个最优支配集， $S$ 中不存在两个点 $p_i$ 和 $p_j$ 满足 $D_{p_i} \subseteq D_{p_j}$（否则， $S \setminus \{p_i\}$ 仍是 $G(P)$的一个支配集，这与 $S$的最优性矛盾）。因此， $S$ 中的每个点 $p$ 在 $\mathcal{VD}(S)$中都有一个非空的沃罗诺伊单元，且该单元包含 $p$ [16,28]，也就是说， $p \in R(p)$。

对于每个点 $p_i \in S$，将包含在 $R(p_i)$中的每个点 $p \in P$ 分配给组$\mathcal{A}_{p_i}$，其中 $p$ 。如果 $p$ 位于两个沃罗诺伊单元的边界上，则将 $p$ 分配给其中任意一个单元。这样，每个点 $p \in P$ 恰好被分配给 $S$中的一个点。此外，由于 $p_i \in R(p_i)$， $p_i$ 被分配到其自身的组 $\mathcal{A}_{p_i}$ （这保证了引理陈述中的第一个条件）。

对于每个点 $p_i \in S$，我们将 $\mathcal{A}_{p_i}$ 中每段连续的极大点序列作为一个子列表（这保证了引理陈述中的第二个条件）。令 $\mathcal{A}$ 为所有点 $p_i \in S$ 对应的 $\mathcal{A}_{p_i}$ 的并集。分配 $\phi$ 直接由上述组的定义 $\mathcal{A}_{p_i}$ 得出。注意， $\mathcal{A}$ 满足引理陈述中的两个条件。

要证明 $\mathcal{A}$ 是 $S$ 的 $P$ 的一个所需划分，只需证明：对于任意点 $p_i \in S$， $D_{p_i}$ 与所有点 $p \in \mathcal{A}_{p_i}$ 对应的 $D_p$ 相交。事实上，考虑一个点 $p \in \mathcal{A}_{p_i}$。由于 $S$ 是一个支配集， $D_p$ 必与某个点 $p_j \in S$ 对应的 $D_{p_j}$ 相交。因此， $d_{p_j}(p) \leq r_p$。由于 $p \in \mathcal{A}_{p_i}$，根据我们对 $\mathcal{A}_{p_i}$ 的构造， $p \in R(p_i)$ 。于是，我们有 $d_{p_i}(p) \leq d_{p_j}(p) \leq r_p$，因此 $D_p$ 与 $D_{p_i}$ 相交。

**证明线可分性质**。接下来要证明分配 $\phi$ 是线可分的。反设存在两个组 $\mathcal{A}_{p_i}$ 和 $\mathcal{A}_{p_j}$ （其中 $p_i$, $p_j \in S$）不可线可分。那么，由于 $p_i \in \mathcal{A}_{p_i}$ 和 $p_j \in \mathcal{A}_{p_j}$，必定存在一个点 $p \in \mathcal{A}_{p_i}$ 和一个点 $p' \in \mathcal{A}_{p_j}$，使得 $p_ip$ 与 $p_jp'$ 相交。另一方面，由于 $p \in \mathcal{A}_{p_i}$， $p$ 位于 $R(p_i)$ 中。由于 $R(p_i)$ 相对于 $p_i$ [16,28]是星形的，因此 $p_ip \subseteq R(p_i)$ 。同理， $p_jp' \subseteq R(p_j)$ 。由于 $R(p_i)$ 与 $R(p_j)$ 内部不相交，$p_ip$不可能与 $p_jp'$ 相交。因此得出矛盾。 □

按照引理 1中的记号，对于每个中心 $p_i \in S$，我们将 $\mathcal{A}_{p_i}$ 中包含 $p_i$的子列表称为 的主子列表 $p_i$，而 $\mathcal{A}_{p_i}$ 中的所有其他子列表称为 次级子列表。

**备注**。 在单位圆盘情形下，对于任意最优支配集 $S$，总存在一个划分 $\mathcal{A}$和一个对 $\phi$ 的分配，使得对于每个 $p_i \in S$，组 $\mathcal{A}_{p_i}$ 至多包含两个子列表 [31]。这一性质使得 [31]中的单位圆盘情形可以采用更高效的算法。遗憾的是，对于一般圆盘问题，这一性质不再成立。图 1 给出了一个例子，其中有一个大圆盘 $D_{p_i}$ ，而其他圆盘彼此不相交，并以一种交替相交与不相交的方式排列 $D_{p_i}$。假设所有点的权重都相同， $p_i$ 将属于任意最优支配集，而 $p_i$ 将被分配到$\Omega(n)$ 个子列表。与单位圆盘情形相比，这种“无界”的组大小大大增加了设计高效算法的难度。尽管如此，我们将在下文证明，仅依靠引理 1 中的线可分性质，仍可推导出一个多项式时间算法。

我们还得到以下引理，它将对算法设计发挥重要作用。

**引理 2.** 设 $S$ 是 $G(P)$的一个最优支配集，并设 $\phi: \mathcal{A} \to S$ 是引理 *1*给出的分配。那么， $S$ 有一个点 $p_i$ ，其所属的组 $\mathcal{A}_{p_i}$ 只有一个子列表（即 $p_i$的主子列表）。

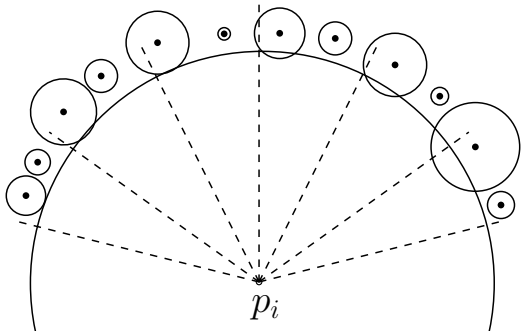



图 1： 举例说明 $|\mathcal{A}_{p_i}| = \Omega(n)$ 的情况。

证明。 对于 $\mathcal{A}$中的每个子列表 $\alpha$，我们选取 $\alpha$ 中的一个点作为其代表点。对于每个点 $p_i \in S$，我们沿着 $\mathcal{H}(P)$对 $\mathcal{A}_{p_i}$ 的子列表排序，并将排序中每对相邻子列表的代表点用一条线段连接起来，称为 $P$的一条对角线 。令 $E$ 为上述针对所有 $p_i \in S$定义的对角线集合。

由于分配 $\phi$ 是直线可分的，因此 $E$ 的任意两条对角线都不会相交。因此，必定存在一条对角线 $d \in E$，将 $\mathcal{H}(P)$ 分成两个子多边形，其中一个不包含其他任何对角线；令 $R$ 表示这样的子多边形。注意， $d$ 连接的是某个点 $p_i$ 所属组 $\mathcal{A}_{p_i}$ 中两个子列表的代表点。根据引理 1 的第二个性质，$\mathcal{A}_{p_i}$ 中没有两个子列表沿着 $\mathcal{H}(P)$ 相邻，因此 $R$ 必定包含一个子列表 $\alpha$，它属于满足 $p_j \neq p_i$ 的组 $\mathcal{A}_{p_j}$。由于 $R$ 不包含除 $d$ 以外的任何对角线， $\alpha$ 的代表点不会通过 $E$ 的任何对角线与其他点相连。这意味着 $\alpha$ 是 $\mathcal{A}_{p_j}$ 中唯一的子列表。因此，引理得证。 □

# 4 加权支配集问题

本节中，我们介绍用于在圆盘图 $G(P)$ 中计算最小权支配集的算法。

对于每个点 $p_i \in P$，令 $w_i$ 表示其权重。我们假设 $w_i > 0$，否则 $p_i$ 总可以包含在解中。对于任意子集 $P' \subseteq P$，定义 $w(P') = \sum_{p_i \in P'} w_i$。

我们将重点讨论以下有界规模问题：给定一个整数 $k$，在满足 $|S| \leq k$ 的条件下，求 $G(P)$ 中总权重最小的支配集 $S \subseteq P$。使用 $k = n$ 求解该问题，可得到 $G(P)$ 的最小权支配集。令 $W^*$ 表示大小至多为 $k$ 的最小权支配集的总权重。

本节其余部分安排如下。第 4.1 节首先介绍一个新概念——秩-$t$中心，它对我们的算法至关重要；同时还概述了该算法。第 4.2 节详细介绍算法，而正确性证明见第 4.3 节。最后，第 4.4 节讨论算法实现和时间分析。

## 4.1 第 $t$ 阶中心与算法概述

设 $S$ 是 $G(P)$ 的一个 $|S| \leq k$，并设 $\phi : \mathcal{A} \to S$ 是引理 1 给出的分配（注意，该引理原本针对的是不受规模限制的最优支配集，但其证明同样适用于受规模限制的最优支配集）。

设 $\alpha_1$、$\alpha_2$、……、$\alpha_m$ 为 $\mathcal{A}$ 中按照 $P$ 的顺序排列的子列表，即沿着 $\mathcal{H}(P)$ 逆时针排列。对于任意两个子列表 $\alpha_g$ 和 $\alpha_h$，我们将 $\mathcal{A}[g,h]$ 定义为从 $\alpha_g$ 到 $\alpha_h$ （含）的子列表集合。视上下文而定，$\mathcal{A}[g,h]$ 也可以指这些子列表的并集，它们构成 $P$ 的一个子列表。我们将 $S[g,h]$ 定义为点 $p \in S$ 的子集，其中点的组 $\mathcal{A}_p$ 包含 $\mathcal{A}[g,h]$ 的一个子列表。注意，如果 $g = h$，那么 $\mathcal{A}[g,h]$ 指的是 $\{\alpha_g\}$。

**定义 1**。 若点 $p_i \in S$ 满足以下条件，则称其为 $\mathcal{A}[g,h]$ 的秩为$t$ 中心：*(1)* $|S[g,h]| \le t$；*(2)* $\alpha_g$ 和 $\alpha_h$ 中至少有一个属于 $\mathcal{A}_{p_i}$；*(3)* $p_i$ 的主子列表属于 $\mathcal{A}[g,h]$；*(4)* 若 $t>1$，则对于每个点 $p_j \in S[g,h]\setminus\{p_i\}$，$\mathcal{A}_{p_j} \subseteq \mathcal{A}[g,h]$，即组 $\mathcal{A}_{p_j}$ 中的所有子列表都属于 $\mathcal{A}[g,h]$；否则，$\mathcal{A}_{p_i} \subseteq \mathcal{A}[g,h]$。

注意，该定义不要求 $\mathcal{A}_{p_i} \subseteq \mathcal{A}[g,h]$ if $t>1$，因此 $\mathcal{A}_{p_i}$ 的某些次级子列表可能不属于 $\mathcal{A}[g,h]$。我们还称上述定义中的 $\mathcal{A}[g,h]$ 为 $p_i$的秩-$t$ 子列表 。此外，在定义的第二个条件中，如果 $\alpha_g$ 和 $\alpha_h$ 中恰有一个属于 $\mathcal{A}_{p_i}$，则称 $p_i$ 为秩-$t$ 开放 中心；如果 $\alpha_g$ 和 $\alpha_h$ 都属于 $\mathcal{A}_{p_i}$，则称 $p$ 为秩-$t$ 闭合中心。

为说明秩-$t$ 中心这一概念为何有用，请注意，每个点 $p_i \in S$ 都是某个 $P$的子列表 $\mathcal{A}[g,h]$ 的秩-$k$ 中心。事实上，令 $\alpha_g \in \mathcal{A}$ 为某个索引 $g$对应的 $p_i$的主子列表。那么，由于 $|S| \le k$, $p_i$ 是 $\mathcal{A}[g-1,g]$ 或 $\mathcal{A}[g,g+1]$的秩-$k$ 中心，其中运算“$g-1$”和“$g+1$”是模 $m$（本节余下部分均遵循这一约定）。注意，$\mathcal{A}[g-1,g]$和 $\mathcal{A}[g,g+1]$ 都是 $P$（并且 $S[g-1,g]$ 和 $S[g,g+1]$ 都是 $S$）。

此外，至少有一个点 $p_i \in S$ 是某个子列表的秩-1 中心，该子列表属于 $\mathcal{A}$。事实上，根据引理 2，存在一个中心 $p_i \in S$，其组 $\mathcal{A}_{p_i}$ 仅包含其主子列表，记为 $\alpha_{g_i}$。于是，$p_i$ 是子列表 $\mathcal{A}[g,g]=\{\alpha_g\}$的秩-1 中心。

**算法概述**。 我们的算法是一种动态规划，共进行 $k$ 次迭代。在第 $t$ 次迭代中，$1 \le t \le k$，我们为每个点 $p_i \in P$ 计算一个集合 $\mathcal{L}_t(i)$，其中包含 $O(n^2)$ 个 $P$ 的子列表；每个子列表 $L \in \mathcal{L}_t(i)$ 都关联一个值 $w'(L)$ 和一个子集 $S_L \subseteq P$，并满足以下算法不变式：(1) $w(S_L) \le w'(L)$；(2) $S_L$ 支配 $L$；(3) $p_i \in S_L$；(4) $|S_L| \le t$。定义 $\mathcal{L}_t = \cup_{p_i \in P}\mathcal{L}_t(i)$。

为论证该算法的正确性，我们将证明：对于每个点 $p_i \in P$ 以及 $p_i$的每个秩为 $t$ 的子列表 $\mathcal{A}[g,h]$，$\mathcal{L}_t(i)$ 都必须包含一个子列表 $L$，满足 $\mathcal{A}[g,h] \subseteq L$，且 $w(S_L) \le w'(L) \le w(S[g,h])$。由于每个点 $p_i \in S$ 都是某个满足 $P$ 的子列表的秩为 $k$ 的中心，因此经过第 $k$ 次迭代后，$\mathcal{L}_k$ 必须包含一个子列表 $L$，满足 $P$，且 $w(S_L) \le w'(L) \le w(S) = W^*$，这意味着 $S_L$ 是一个最优支配集（大小至多为 $k$）。因此，经过 $k$ 次迭代后，在 $\mathcal{L}_k$ 中所有满足 $P$ 的子列表里，我们找到具有最小 $W'(L)$ 的那个子列表 $L$，并将 $S_L$ 作为最优支配集返回。注意，由于至少有一个点 $p_i \in S$ 是 $\mathcal{A}$ 的某个子列表的秩为 1 的中心，我们的算法从 $t=1$ 开始。

本节其余部分将使用以下记号。

**定义 2**。 对于两个点 $p_i$、$p_j \in P$（$p_i = p_j$ 是可能的），将 $a_i^j$ 定义为从 $p_j$ 起沿逆时针方向的 $P$ 中满足 $|D_{p_i}D_p| > 0$ 的第一个点 $p$ 的索引，并将 $b_i^j$ 定义为从 $p_j$ 起沿顺时针方向的 $P$ 中满足 $|D_{p_i}D_p| > 0$ 的第一个点 $p$ 的索引（如果 $|D_{p_i}D_{p_j}| > 0$，则 $a_i^j = b_i^j = j$）。如果对于所有点 $p \in P$ 都有 $|D_{p_i}D_p| \le 0$，则令 $a_i^j = b_i^j = 0$。

## 4.2 算法描述

现在我们描述带有 $1 \le t \le k$ 的算法第 $t$ 次迭代的详细过程。

初始时，$t=1$，并且对于每个 $p_i \in P$，我们的算法根据定义 2 计算两个索引 $a_i^i$ 和 $b_i^i$。我们将在引理 10 中证明，这可以在 $O(n\log^2 n)$ 时间内完成。然后，对于每个 $p_i \in P$，令 $L = P(b_i^i, a_i^i)$，$S_L = \{p_i\}$、$w'(L) = w_i$，以及 $\mathcal{L}_1(i) = \{L\}$。显然，对于 $L$，所有算法不变式均成立。至此，算法的第一次迭代完成。

假设对于每个 $t'$、$1 \le t' \le t-1$，我们都已计算出集合 $\mathcal{L}_{t'}(i)$，其中包含针对每个 $p_i \in P$ 的 $O(n^2)$ 个子列表，且每个子列表 $L \in \mathcal{L}_{t'}(i)$ 都关联一个值 $w'(L)$ 和一个点集 $S_L \subseteq P$，使得 $L$ 的算法不变式成立，即 $w(S_L) \le w'(L)$，$S_L$ 支配 $L$, $p_i \in S_L$，且 $|S_L| \le t'$。

算法的第 $t$ 次迭代如下进行。对于每个点 $p_i \in P$，我们针对每个点 $p_j$ 执行逆时针/顺时针处理过程，并执行双向预处理过程。具体细节如下。

### 4.2.1 逆时针/顺时针处理过程

对于其他每个点 $p_j \in P$，我们执行一个逆时针处理过程，具体如下。对于每个$t'$ 满足 $1 \le t' \le t-1$的，以及每个点 $p_z \in P[i,j]$，我们执行以下操作。参见图 2。

我们首先对 $\mathcal{L}_{t'}(i)$ 执行最小值包围子列表查询，以找到满足 $P[i,z] \subseteq L_1$ 的、 $w'(L_1)$ 最小的子列表 $L_1 \in \mathcal{L}_{t'}(i)$；我们将在引理9中讨论如何求解该查询。令 $p_{z_1}$ 为 $L_1$ 的逆时针端点。接着，我们对 $\mathcal{L}_{t-t'}$ 执行另一次最小值包围子列表查询，以找到满足 $P[z_1+1,j] \subseteq L_2$ 的、 $w'(L_2)$ 最小的子列表 $L_2 \in \mathcal{L}_{t-t'}$。令 $p_{z_2}$ 为 $L_2$的逆时针端点。接下来，我们计算索引 $a_i^{z_2+1}$ 。

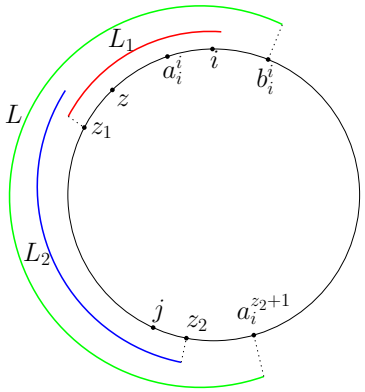



图2： 展示 $P$ 中各点的相对位置（仅显示其索引）：圆表示 $\mathcal{H}(P)$。

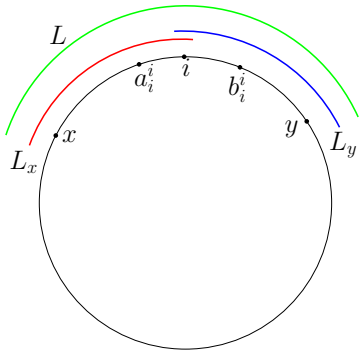



图3： 展示 $L_x$、 $L_y$和 $L$。

根据构造，以下四个子列表依次相邻，因此构成$P$的一个子列表： $P(b_i^i,\ a_i^i)$ 、 $L_1$、 $L_2$和 $P(z_2,\ a_i^{z_2+1})$。将这个合并后的子列表记为 $L(t',\ m)$；若上下文已明确$t'$ 和 $m$，则简记为 $L$。令$S_L = S_{L_1} \cup S_{L_2}$，并令 $w'(L) = w'(L_1) + w'(L_2)$。

**观察1**。 所有算法不变式对 $L$均成立。

证明。 我们证明算法不变式对 $L$成立，即 $w(S_L) \le w'(L)$， $L$ 被 $S_L$、$p_i \in S_L$和 $|S_L| \le t$支配。事实上，由于$L_1 \in \mathcal{L}_{t'}(i)$，根据 $t'$的算法不变式， $w(S_{L_1}) \le w'(L_1)$，$L_1$ 被 $S_{L_1}$、$p_i \in S_{L_1}$和 $|S_{L_1}| \le t'$支配。同理，由于 $L_2 \in \mathcal{L}_{t-t'}$，根据 $t-t'$的算法不变式， $w(S_{L_2}) \le w'(L_2)$， $L_2$ 被 $S_{L_2}$和 $|S_{L_2}| \le t-t'$支配。此外，根据定义， $P(b_i^i,\ a_i^i)$ 和 $P(z_2,\ a_i^{z_2+1})$ 中的点被 $p_i$支配。综合以上各点可知，所有算法不变式对 $L$均成立。 □

在所有满足 $1 \le t' \le t-1$ 和 $p_z \in P[i,j]$的此类子列表 $L(t',m)$ 中，我们保留使 $w'(L(t',m))$ 最小的那个，并将其加入 $\mathcal{L}_t(i)$。至此完成针对 $p_j$ 的逆时针处理过程。

对称地，我们也为 $p_j$ 执行顺时针处理过程，该过程至多向 $\mathcal{L}_t(i)$ 添加一个子列表。

对所有 $p_j \in P$ 执行这两个过程，会向 $\mathcal{L}_t(i)$ 添加 $O(n)$ 个子列表。

### 4.2.2 双向处理过程

我们还对执行双向处理过程，其中 $p_i$。对于每一对点$(p_x,\ p_y)$，满足$p_y,\ p_i,\ p_x$沿着 $\mathcal{H}(P)$按逆时针顺序排列；对于每个 $t'$，满足 $2 \le t' \le t-1$，我们执行以下操作。参见图3。

我们在 $\mathcal{L}_{t'}(i)$ 上执行最小值包围子列表查询，以找到满足 $P[i,x] \subseteq L_x$ 的子列表 $L_x \in \mathcal{L}_{t'}(i)$，使 $w'(L_x)$ 最小。类似地，我们在 $\mathcal{L}_{t+1-t'}(i)$ 上执行最小值包围子列表查询，以找到满足 $P[y,i] \subseteq L_y$ 的子列表 $L_y \in \mathcal{L}_{t+1-t'}(i)$，使 $w'(L_y)$ 最小。根据定义，$P(b_i^i, a_i^i)$、$L_x$ 和 $L_y$ 是连续的，因此构成 $P$ 的一个子列表，记作 $L(x, y, t')$，或简记为 $L$。我们令 $S_L = S_{L_x} \cup S_{L_y}$，并令 $w'(L) = w'(L_x) + w'(L_y) - w_i$。

**观察 2.** 所有算法不变式对 $L$均成立。

证明。事实上，根据 $t'$ 和 $t - t' + 1$ 的算法不变式，子列表 $L_x$ 和 $L_y$ 分别被 $S_{L_x}$ 和 $S_{L_y}$ 支配，$w(S_{L_x}) \leq w'(L_x)$，$w(S_{L_y}) \leq w'(L_y)$，$p_i \in S_{L_x}$，$p_i \in S_{L_y}$，$|S_{L_x}| \leq t'$，以及$|S_{L_y}| \leq t - t' + 1$。由于 $S_{L_x}$ 和 $S_{L_y}$ 都包含 $p_i$，且 $P(b_i^i, a_i^i)$ 被 $p_i$ 支配，$S_L$ 的大小至多为 $t$，并支配 $L$。此外，$w(S_L) \leq w(S_{L_x}) + w(S_{L_y}) - w_i \leq w'(L_x) + w'(L_y) - w_i = w'(L)$。因此，算法不变式对 $L$成立。 □

对于一个固定的数对 $(p_x, p_y)$，在所有子列表 $L(x, y, t')$，$2 \leq t' \leq t-1$ 中，我们保留使$w'(L(x, y, t'))$ 最小的那个，并将其加入 $\mathcal{L}_t(i)$。这样，针对 $p_i$ 的双向处理过程会向 $\mathcal{L}_t(i)$ 中添加 $O(n^2)$ 个子列表。

### 4.2.3 总结

总之，算法的第 $t$ 次迭代会为每个 $\mathcal{L}_t(i)$ 计算 $O(n^2)$ 个子列表，并总共为 $\mathcal{L}_t$ 计算 $O(n^3)$ 个子列表。

在第 $k$ 次迭代之后，在 $\mathcal{L}_k$ 中所有满足 $P$ 的子列表中，我们找出一个 $L^*$，使其 $w'(L^*)$ 最小，并将 $S_{L^*}$ 作为大小至多为 $k$ 的最优支配集返回。如果 $\mathcal{L}_k$ 中没有子列表满足 $P$，则我们报告不存在大小至多为 $k$ 的支配集。

## 4.3 算法正确性

在本节中，我们 将证明以下引理，该引理确立了算法的正确性 .

**引理3。** $S_{L^*}$ 是一个最优支配集，且 $W^* = w'(L^*)$。

根据算法不变式，子列表 $L^*$（即 $P$）被 $S_{L^*}$ 支配，其中$|S_{L^*}| \leq k$ 且 $w(S_{L^*}) \leq w'(L^*)$。因此，我们得到 $W^* \leq w(S_{L^*}) \leq w'(L^*)$。为证明该引理，下面我们将证明 $w'(L^*) \leq W^*$。请注意，我们的证明还说明，必定存在这样一个子列表$L^* \in \mathcal{L}_k$，且满足 $L^* = P$。

设 $S$ 是一个大小至多为 $k$ 的最优支配集，并设 $\phi$: $\mathcal{A} \to S$ 是引理1所保证的线可分配。按照第 $\mathcal{A} = \langle \alpha_1, \alpha_2, \ldots\ldots, \alpha_m \rangle$ 沿逆时针方向排列子列表$\mathcal{H}(P)$，如第 4.1 节所述（我们也沿用该节的记号，例如 $\mathcal{A}[g,h]$、$\mathcal{A}_{p_i}$、$S[g,h]$）。我们将证明以下引理。

**引理4。** 对于任意点 $p_i \in S$ 以及任意满足 $1 \leq t \leq k$的 $t$，对于 $p_i$的任意秩为$t$ 的子列表 $\mathcal{A}[g,h]$，$\mathcal{L}_t(i)$ 必定包含一个子列表 $L$，满足 $\mathcal{A}[g,h] \subseteq L$ 且 $w'(L) \leq w(S[g,h])$。

引理4将推出 $w'(L^*) \leq W^*$，从而证明引理3。事实上，正如前面所讨论的，$p_i$必定有一个秩为$k$ 的子列表 $\mathcal{A}[g,h]$，它满足 $P$和 $S[g,h] = S$。由于 $\mathcal{L}_k(i)$包含一个子列表 $L$，满足$\mathcal{A}[g,h] \subseteq L$ 且 $w'(L) \leq w(S[g,h])$，因此我们得到 $L = P$且 $w'(L) \leq w(S) = W^*$。根据定义，$L^* = P$ 且 $w'(L^*) \leq w'(L) \leq w(S) = W^*$。由此证明引理3。

本节余下部分将通过对 $t = 1, 2, \ldots\ldots, k$进行归纳法来证明引理 4。回顾引理 2，基本情形从 $t = 1$ 开始，因为 $S$ 至少有一个点在 $\mathcal{A}$中具有秩为 1 的子列表。

对于基本情形 $t = 1$，假设 $p_i \in S$ 有一个秩为 1 的子列表 $\mathcal{A}[g,h]$。根据定义，$\mathcal{A}[g,h]$ 是组 $\mathcal{A}_{p_i}$中唯一的子列表，并且它是 $p_i$的主子列表，这意味着 $p_i \in \mathcal{A}[g,h]$。因此，$w(S[g,h]) = w_i$。由于 $p_i \in \mathcal{A}[g,h]$，且 $P(b_i^i, a_i^i)$ 是包含 $p_i$的 $P$的极大子列表，根据

定义 $a_i^i$和 $b_i^i$，我们有 $\mathcal{A}[g,h] \subseteq P(b_i^i,\ a_i^i)$。根据我们的算法，$\mathcal{L}_1(i)$ 由一个子列表组成，即 $L = P(b_i^i, a_i^i)$，其中 $S_L = \{p_i\}$ 且 $w'(S_L) = w_i$。因此，引理4 对基本情形$t = 1$成立。

现在假设引理4对每个 $t' \in [1, t-1]$ 都成立，即，对于作为某个 $p_i \in S$ 的秩为 $t'$ 的子列表的每个子列表 $\mathcal{A}[g',h']$，$\mathcal{L}_{t'}(i)$ 都包含一个满足以下条件的子列表 $L'$：$\mathcal{A}[g',h'] \subseteq L'$，且 $w'(L') \leq w(S[g',h'])$。下面，我们证明引理4对 $t$ 成立，即，对于作为某个 $p_i \in S$ 的秩为 $t$ 的子列表的每个子列表 $\mathcal{A}[g,h]$，$\mathcal{L}_t(i)$ 都包含一个满足以下条件的子列表 $L$：$\mathcal{A}[g,h] \subseteq L$，且 $w'(L) \leq W(S[g,h])$。

假设 $p_i$ 是 $\mathcal{A}[g,h]$的秩为 $t$ 的中心。根据定义，$\mathcal{A}[g,h]$ 的两个端子列表中至少有一个属于组 $\mathcal{A}_{p_i}$，而 $\mathcal{A}[g,h]$ 包含 $p_i$的主子列表。根据 $p_i$ 的主子列表是否为 $\mathcal{A}[g,h]$的端子列表，分为两种情况。

### 4.3.1 情况（1）：$p_i$ 的主子列表是$\mathcal{A}[g,h]$的一个端子列表

不失一般性，我们假设 $\mathcal{A}[g,h]$ 的顺时针端子列表是 $p_i$的主子列表，即 $\alpha_g$。我们有以下引理，可将 $\mathcal{A}[g,h]$ “分解”为更小的子列表。参见图 4。

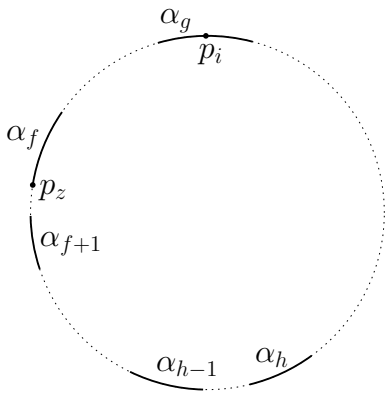



图4：$\mathcal{A}$ 的子列表示意图，以黑色实线弧表示（点状圆表示 $\mathcal{H}(P)$）。

**引理5。1.（开放情形）**如果 $p_i$ 是 $\mathcal{A}[g,h]$ 的秩为 $t$ 的开放中心，那么存在 $t' \in [1,t-1]$、$f \in [g,h-1]$ 和一个点 $p_{i'} \in S[g,h]$，使得 $p_i$ 是 $\mathcal{A}[g,f]$ 的秩为 $t'$ 的中心，且 $p_{i'}$ 是 $\mathcal{A}[f+1,h]$ 的秩为 $(t-t')$ 的中心。此外，$w(S[g,h]) = w(S[g,f]) + w(S[f+1,h])$。

**2.（闭情形）**若 $p_i$ 是 $\mathcal{A}[g,h]$ 的秩为 $t$ 的闭中心，则存在 $t' \in [1,t-1]$、$f \in [g,h-1]$ 以及一个点 $p_{i'} \in S[g,h]$，使得 $p_i$ 是 $\mathcal{A}[g,f]$ 的秩为 $t'$ 的中心，且 $p_{i'}$ 是 $\mathcal{A}[f+1,h-1]$ 的秩为 $(t-t')$ 的中心。此外，$w(S[g,h]) = w(S[g,f]) + w(S[f+1,h-1])$。

证明。我们首先讨论开放情形。在这种情形下，子列表 $\alpha_h$ 必须属于另一个点 $p_{i'} \in S[g,h] \setminus \{p_i\}$ 的 $\mathcal{A}_{p_i'}$。令 $\alpha_f$ 为分配给 $p_i$ 的 $\mathcal{A}[g,h]$ 中最远的逆时针子列表。令 $t' = |S[g,f]|$。由于 $\alpha_g$ 是 $p_i$ 的子列表（即 $\alpha_g \in \mathcal{A}_{p_i}$），根据线可分性质，$S[f+1,h]$ 中的任何点都不能在 $\mathcal{A}[g,f]$中拥有子列表。这意味着 $1 \leq t' \leq t-1$ 和 $|S[f+1,h]| = t - t'$，

考虑一个点 $p \in S[f+1,h]$。根据定义，$\mathcal{A}_p$的所有子列表都在 $\mathcal{A}[g,h]$中。由于 $\mathcal{A}_p$不能在 $\mathcal{A}[g,f]$中有子列表，因此 $\mathcal{A}_p$的所有子列表都在 $\mathcal{A}[f+1,h]$中。由于 $\alpha_h \in \mathcal{A}_{p_i'}$，根据定义，$p_{i'}$是 $\mathcal{A}[f+1,h]$ 的一个秩为（$t-t'$）的中心。

另一方面，由于 $\mathcal{A}[g,f]$的两个端子列表都来自 $\mathcal{A}_{p_i}$，同样根据线可分性质，$S[g,f] \setminus \{p_i\}$中的点都不可能在 $\mathcal{A}[g,f]$之外有子列表。由于 $|S[g,f]| = t'$，根据定义，$p_i$是 $\mathcal{A}[g,f]$的一个秩为（$t'$）的（闭）中心。

上述讨论也意味着，$S[g,f]$和 $S[f+1,h]$构成了 $S[g,h]$的一个划分。因此，$w$（$S[g,h]$）$= w$（$S[g,f]$）$+ w$（$S[f+1,h-1]$）。

现在讨论闭情形。在这种情况下，$\alpha_h$ 也被分配给 $p_i$，即 $\alpha_h \in \mathcal{A}_{p_i}$。设 $p_{i'}$ 为满足 $\alpha_{h-1} \in \mathcal{A}_{p_i'}$的点。我们将$\alpha_f$ 定义为 $\mathcal{A}[g,h-1]$中最远的逆时针子列表

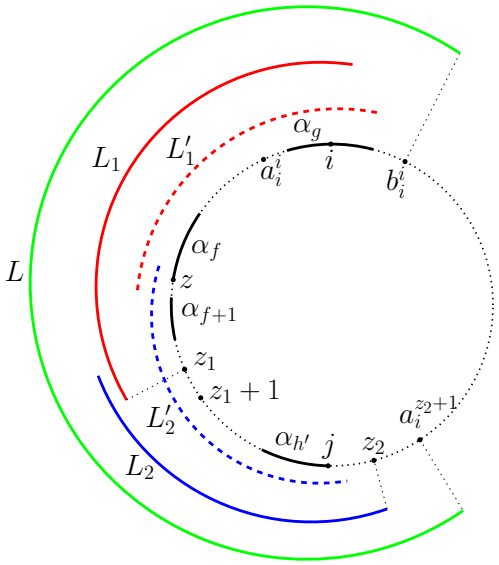



图5： 引理 6的证明示意图。对于 $P$中的点，仅显示其索引。红色（相应地，蓝色）虚线弧表示 $L'_1$ （相应地， $L'_2$）。红色（相应地，蓝色、绿色）实线弧表示 $L_1$ （相应地， $L_2$、 $L$）。点状圆上的黑色实线弧表示 $\mathcal{A}$的子列表。

分配给 $p_i$。设 $t' = |S[g, f]|$。由于$\alpha_g$和$\alpha_f$都在 $\mathcal{A}_{p_i}$中，根据线可分性质， $S[g, f] \setminus \{p_i\}$中的点都不可能在 $\mathcal{A}[g, f]$之外有子列表。因此， $p_i$是 $\mathcal{A}[g, f]$的秩为$t'$（闭）的中心。类似地，由于$\alpha_f$和 $\alpha_h$都在 $\mathcal{A}_{p_i}$中， $S[f + 1, h - 1]$中的点都不可能在 $\mathcal{A}[f + 1, h - 1]$之外有子列表。根据 $f$的定义， $S[f + 1, h - 1]$ 的任何子列表都不在$\mathcal{A}_{p_i}$中。因此， $|S[f + 1, h - 1]| = t - t'$和 $p_{i'}$是 $\mathcal{A}[f + 1, h - 1]$的秩为 $(t - t')$ 的中心。上述讨论还表明， $S[g, f]$ 和 $S[f + 1, h - 1]$ 构成 $S[g, h]$的一个划分。因此， $w\left(S[g, h]\right) = w\left(s[g, f]\right) + w\left(s[f + 1, h - 1]\right)$ 。 □

定义 $h'$：如果 $p_i$是开放中心，则为 $h$；如果 $p_i$是闭中心，则为 $h - 1$。在这两种情况下， $p_{i'}$都是 $(t - t')$ 阶中心，属于 $\mathcal{A}[f + 1, h']$。令 $p_j$为 $\mathcal{A}[f + 1, h']$的逆时针端点。令 $p_z$为引理 $\alpha_f$ 中的 5 的逆时针端点（见图 4）。

现在，考虑算法在逆时针处理过程中处理 $p_i$ 时的第 $t$ 次迭代，并检查三元组 $(p_j, p_z, t')$。在此步骤中，算法为 $\mathcal{L}_t(i)$ 构造子列表 $L$，其为以下四个子列表的并集（见图2）： $P(b_i^i, a_i^i)$、包含 $P[i, z]$的 $\mathcal{L}_{t'}(i)$ 的最小值子列表 $L_1$、包含 $P[z_1 + 1, j]$的 $\mathcal{L}_{t-t'}$ 的最小值子列表 $L_2$，以及 $P(z_2, a_i^{z_2+1})$，其中 $p_{z_1}$ 和 $p_{z_2}$ 分别是 $L_1$ 和 $L_2$ 的逆时针端点。此外，$S_L = S_{L_1} \cup S_{L_2}$ 和 $w'(L) = w'(L_1) + w'(L_2)$。下面的引理完成了我们对情况（1）的证明。

**引理6。** $\mathcal{A}[g, h] \subseteq L$ 和 $w'(L) \leq w(S[g, h])$。

证明。根据归纳假设， $\mathcal{L}_{t'}(i)$包含一个子列表 $L'_1$，使得 $\mathcal{A}[g, f] \subseteq L'_1$ 且 $w'(L'_1) \leq w(S[g, f])$；类似地，$\mathcal{L}_{t-t'}(i')$包含一个子列表 $L'_2$，使得 $\mathcal{A}[f + 1, h'] \subseteq L'_2$ 且 $w'(L_2) \leq w(S[f + 1, h'])$。见图5。

由于 $P[i, z] \subseteq L_1$和 $p_i \in \alpha_g \subseteq P(b_i^i, a_i^i)$，我们有 $P[i, z] \subseteq \mathcal{A}[g, f] \subseteq P(b_i^i, a_i^i) \cup L_1$。由于 $\mathcal{A}[g, f] \subseteq L'_1$， $P[i, z] \subseteq L'_1$。因为 $L_1$是包含 $P[i, z]$的 $\mathcal{L}_{t'}(i)$的最小值子列表，而 $L'_1$是包含 $P[i, z]$的 $\mathcal{L}_{t'}(i)$的子列表，我们得到 $w'(L_1) \leq w'(L'_1)$。由于 $w'(L'_1) \leq w(S[g, f])$，因此

$$w'(L_1) \leq w(S[g, f]). \tag{1}$$

我们断言， $p_{z_1+1}$ 必须属于 $\mathcal{A}[f + 1, h']$的一个子列表。反设并非如此。那么，由于 $\mathcal{A}[g, f]$和 $\mathcal{A}[f + 1, h']$相邻，且 $\mathcal{A}[g, f] \subseteq P(b_i^i, a_i^i) \cup L_1$，我们有

$\mathcal{A}[g,h'] = \mathcal{A}[g,f] \cup \mathcal{A}[f+1,h'] \subseteq P(b_i^i, a_i^i) \cup L_1$。根据定义，$|S_{L_1}| \le t' = |S[g,f]| < t$。由于 $p_i$ 位于 $S_{L_1}$中，且 $p_i$ 支配 $P(b_i^i, a_i^i)$，若将 $S$ 中的 $S[g,h']$ 替换为 $S_{L_1}$中的点，我们仍可得到一个新的支配集 $S'$，它针对 $P$ 且大小至多为 $k$。但由于 $w(S_{L_1}) \le w'(L_1) \le w(S[g,f])$，新的支配集 $S'$的权重为 $w(S') = w(S \setminus S[g,h']) + w(S_{L_1}) < w(S \setminus S[g,h']) + w(S[g,f]) + w(S[f+1,h']) = w(S)$，这与 $S$的最优性矛盾。因此，上述断言成立。

由上述结论可得 $P[z_1+1,j] \subseteq \mathcal{A}[f+1,h']$。由于 $\mathcal{A}[f+1,h'] \subseteq L_2'$, $P[z_1+1,j] \subseteq L_2'$。因为 $L_2$ 是包含 $P[z_1+1,j]$ 和 $L_2' \in \mathcal{L}_{t-t'}$ 的 $\mathcal{L}_{t-t'}$ 的最小值子列表，且 $i'$，所以 $\subseteq \mathcal{L}_{t-t'}$,$w'(L_2) \le w'(L_2')$ 成立。由于 $w'(L_2') \le w(S[f+1,h'])$，因此

$$w'(L_2) \le w(S[f+1,h']). \tag{2}$$

回顾一下，$L$ 是以下四个子列表的并集：$P(b_i^i, a_i^i)$、$L_1$、$L_2$ 和 $P(z_2, a_i^{z_2+1})$。根据 $p_i$ 是 $\mathcal{A}[g,h]$ 的开或闭秩-$t$ 中心，有两种情况。

- 如果 $p_i$ 是一个开放中心，则 $h' = h$，从而 $\mathcal{A}[g,h] = \mathcal{A}[g,f] \cup \mathcal{A}[f+1,h']$。上文已证明，$\mathcal{A}[g,f] \subseteq P(b_i^i, a_i^i) \cup L_1$。由于 $\mathcal{A}[f+1,h']$ 的顺时针端点是 $p_{z+1}$且 $p_z \in L_1$，我们得到 $\mathcal{A}[f+1,h'] \subseteq L_1 \cup P[z_1+1,j]$。因此，$\mathcal{A}[g,h] \subseteq P(b_i^i, a_i^i) \cup L_1 \cup P[z_1+1,j]$。由于$P[z_1+1,j] \subseteq L_2$，可推出 $\mathcal{A}[g,h] \subseteq P(b_i^i, a_i^i) \cup L_1 \cup L_2 \subseteq L$。

  根据引理5，$w(S[g,h]) = w(S[g,f]) + w(S[f+1,h'])$。由于 $w'(L) = w'(L_1) + w'(L_2)$，由不等式（1）和（2）可推出 $w'(L) \le w(S[g,h])$。

- 如果 $p_i$ 是一个闭中心，那么 $h' = h-1$ 和 $\mathcal{A}[g,h] = \mathcal{A}[g,f] \cup \mathcal{A}[f+1,h'] \cup \alpha_h$。与上述开放情形一样，$\mathcal{A}[g,f] \cup \mathcal{A}[f+1,h'] \subseteq P(b_i^i, a_i^i) \cup L_1 \cup L_2 \subseteq L$。

  如果 $\mathcal{A}[g,h] \subseteq P(b_i^i, a_i^i) \cup L_1 \cup L_2$，那么我们仍有 $\mathcal{A}[g,h] \subseteq L$。否则，必定有 $p_{z_2+1} \in \alpha_h$。因此，由于 $p_i$支配 $\alpha_h$，且 $p_j$是 $\mathcal{A}[f+1,h']$的逆时针端点，根据 $a_i^{z_2+1}$的定义，$\alpha_h \subseteq P[j+1, a_i^{z_2+1})$。由于 $\mathcal{A}[g,f] \cup \mathcal{A}[f+1,h'] \subseteq P\ b_i^i\ a_i^i\ \cup L_1 \cup L_2$ $\mathcal{A}[g,h] = \mathcal{A}[g,f] \cup \mathcal{A}[f+1,h'] \cup \alpha_h \subseteq P\ b_i^i\ a_i^i\ \cup L_1 \cup L_2 \cup \alpha_h \subseteq (,)$，我们得到 $(,)(b_i^i, P$ $a_i^i) \cup L_1 \cup L_2 \cup P[j+1, a_i^{z_2+1})$。根据定义，$p_j \in L_2$，且 $z_2$是 $L_2$ $L_2 \cup P[j+1, a_i^{z_2+1} \subseteq L_2 \cup P\ z_2\ a_i^{z_2+1}$ $P\ b_i^i\ a_i^i\ \cup L_1 \cup L_2 \cup$的逆时针端点，因此我 $P[j+1, a_i^{z_2+1} \subseteq P\ b_i^i\ a_i^i\ \cup L_1 \cup L_2 \cup P\ z_2\ a_i^{z_2+1} = L$ $\mathcal{A}[g,h] \subseteq L$ 们有 $)(,)$。因此，$(,))(,)(,)$。由此可得 。

  由引理5，$w(S[g,h]) = w(S[g,f]) + w(S[f+1,h'])$。与上述情形一样，由于 $w'(L) = w'(L_1) + w'(L_2)$，根据不等式（1）和（2），最终可推出 $w'(L) \le w(S[g,h])$。

总之，在这两种情形下，我们都有 $\mathcal{A}[g,h] \subseteq L$ 和 $w'(L) \le w(S[g,h])$。因此引理得证。 □

#### 4.3.2 情形（2）：$p_i$ 的主子列表不是$\mathcal{A}[g,h]$的端子列表

由于 $p_i$ 是一个秩-$t$ 中心，且 $\mathcal{A}[g,h]$的两个端子列表之一 $\mathcal{A}[g,h]$ 必属于组 $\mathcal{A}_{p_i}$。不失一般性，我们假设 $\mathcal{A}[g,h]$的逆时针端子列表，即 $\alpha_h$，属于$\mathcal{A}_{p_i}$。注意，$\alpha_h$ 必是 $p_i$的次子列表。令 $\alpha_f$ 为 $p_i$的主子列表。根据定义，$\alpha_f \in \mathcal{A}[g,h]$。我们有以下引理。见图 6。

**引理 7**。存在 $t' \in [2, t-1]$，使得 $p_i$ 是 $\mathcal{A}[f,h]$ 的秩-$t'$ 中心，并且是 $\mathcal{A}[g,f]$ 的秩-$(t-t'+1)$中心。此外，$w(S[g,h]) = w(S[f,h]) + w(S[g,f]) - w_i$。

证明。定义 $t' = |S[f,h]|$。由于 $\alpha_h$ 和 $\alpha_f$ 都是 $\mathcal{A}_{p_i}$的子列表，$S[f,h] \setminus \{p_i\}$ 非空，因此 $t' \ge 2$；此外，由于线可分性质，$S[f,h] \setminus \{p_i\}$ 的任何中心都不可能拥有位于 $\mathcal{A}[f,h]$之外的子列表。因此，$p_i$ 是 $\mathcal{A}[f,h]$的秩-$t'$中心。

任取一个中心 $p_{i'} \in S[g,f] \setminus \{p_i\}$。我们声称，$\mathcal{A}_{p_{i'}}$的所有子列表都在 $\mathcal{A}[g,f]$中。事实上，由于 $p_i$ 是 $\mathcal{A}[g,h]$ 和 $p_{i'} \in S[g,h]$中的一个秩为 $t$ 的中心，$\mathcal{A}_{p_{i'}}$的所有子列表都在 $\mathcal{A}[g,h]$中。由于 $\alpha_h$

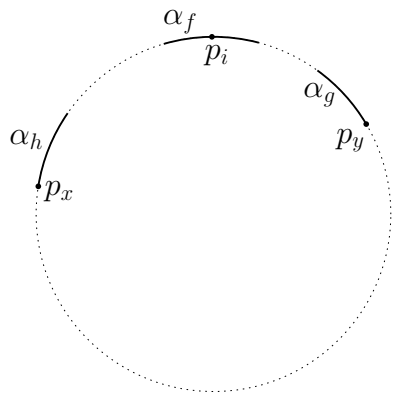



图 6： 说明 $\mathcal{A}$的子列表，其中黑色实线弧表示这些子列表（点状圆表示 $\mathcal{H}(P$ )).

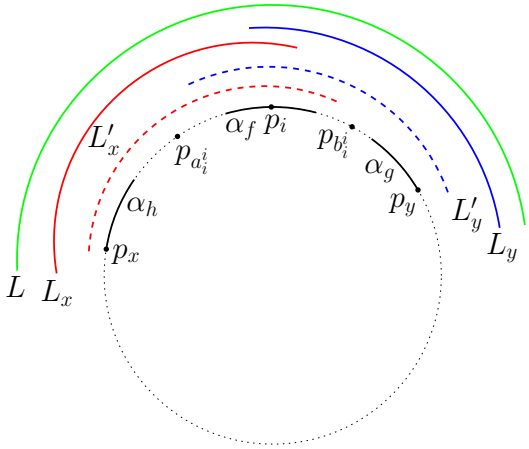



图 7： 引理 8 证明示意图。红色（分别为蓝色）虚线弧表示 $L'_1$ （分别为 $L'_2$）。红色（分别为蓝色、绿色）实线弧表示 $L_1$ （分别为 $L_2$、 $L$）。点状圆上的黑色实线弧表示 $\mathcal{A}$的子列表。

和 $\alpha_f$ 都是 $p_i$的子列表，根据线可分性质， $\mathcal{A}_{p_{i'}}$ 不可能在 $\mathcal{A}[f,h]$中有子列表。因此， $\mathcal{A}_{p_{i'}}$的所有子列表都必须在 $\mathcal{A}[g,f]$ 中。

鉴于上述结论，由于 $p_i$同时属于 $S[g,f]$和 $S[f,h]$，因此有 $|S[g,f]| = t+1-|S[f,h]| = t+1-t'$。由于 $\alpha_f$ 是 $\mathcal{A}[g,f]$的端子列表， $p_i$是 $\mathcal{A}[g,f]$的秩-$(t+1-t')$中心。

上述讨论还意味着， $S[f,h]\backslash\{p_i\}$和 $S[g,f]\backslash\{p_i\}$构成 $S[g,h]\backslash\{p_i\}$的一个划分。由于 $p_i$属于 $S[f,h]$、 $S[g,f]$和 $S[g,h]$，因此有 $w(S[g,h]) = w(S[f,h]) + w(S[g,f]) - w_i$。 □

由引理 $t'$ 7定义。令 $p_x$ 和$p_y$ 分别表示 $\mathcal{A}[g,h]$的逆时针端点和顺时针端点；见图。

现在考虑算法的第 $t$次迭代，在该次迭代中， $p_i$ 通过双向处理过程进行处理，并考虑三元组（$p_x$， $p_y$， $t'$）。在此步骤中，算法构造子列表 $L \in \mathcal{L}_t(i)$，它是以下三个子列表的并集： $P(b_i^i,\ a_i^i)$、 $\mathcal{L}_{t'}(i)$中包含 $P[i,x]$的最小值包围子列表 $L_x$，以及 $\mathcal{L}_{t+1-t'}(i)$中包含$P[y,i]$的最小值包围子列表 $L_y$。此外， $S_L = S_{L_x} \cup S_{L_y}$和 $w'(L) = w'(L_x) + w'(L_y) - w_i$。以下引理完成了我们对情形（2）的证明。

**引理8。** $\mathcal{A}[g,h] \subseteq L$ 和 $w'(L) \leq w(S[g,h])$。

证明。根据关于 $t'$ 的归纳假设， $\mathcal{L}_{t'}(i)$ 包含一个子列表 $L'_x$，使得 $\mathcal{A}[f,h] \subseteq L'_x$ 且 $w'(L'_x) \leq w(S[f,h])$。类似地， $\mathcal{L}_{t+1-t'}(i)$ 包含一个子列表 $L'_y$，使得 $\mathcal{A}[g,f] \subseteq L'_y$ 且 $w'(L'_y) \leq w(S[g,f])$。见图 7。

回忆一下，$\alpha_f$ 是 $p_i$ 的主子列表。由于 $p_i \in \alpha_f$，根据 $b_i^i$ 和 $a_i^i$ 的定义，$\alpha_f \subseteq P(b_i^i, a_i^i)$ 成立。由于 $P[i,x] \subseteq L_x$，我们有 $\mathcal{A}[f,h] \subseteq P(b_i^i, a_i^i) \cup L_x$。此外，根据定义，$P[i,x] \subseteq \mathcal{A}[f,h]$。由于 $\mathcal{A}[f,h] \subseteq L'_x$，我们有 $P[i,x] \subseteq L'_x$。由于 $L'_x \in \mathcal{L}_{t'}(i)$，且 $L_x$ 是 $\mathcal{L}_{t'}(i)$ 中包含 $P[i,x]$ 的最小值子列表，因此可得 $w'(L_x) \leq w'(L'_x)$。因为 $w'(L'_x) \leq w(S[f,h])$，我们得到 $w'(L_x) \leq w(S[f,h])$。

通过对称论证，我们可以得到 $\mathcal{A}[g,f] \subseteq P(b_i^i, a_i^i) \cup L_y$ 以及 $w'(L_y) \leq w(S[g,f])$。

结合上述结果可得 $\mathcal{A}[g,h] = \mathcal{A}[f,h] \cup \mathcal{A}[g,f] \subseteq P(b_i^i, a_i^i) \cup L_x \cup L_y = L$ 以及 $w'(L) = w'(L_x) + w'(L_y) - w_i \leq w(S[f,h]) + w(S[g,f]) - w_i = w(S[g,h])$，其中最后一个等式由引理 7得出。引理得证。 □

### 4.4 时间分析和实现

在第 $t$ 次迭代中，$1 \leq t \leq k$，该算法执行 $O(tn^3)$ 次最小值包围子列表查询：给定一个子列表 $L$（属于 $P$），计算包含 $L$ 的最小值子列表，该子列表属于集合 $\mathcal{L}$。该问题已在 [31]中研究。我们在下面的引理中总结相关结果。

**引理 9.** ( [31]) 给定由 $P$ 的 $m$ 个子列表组成的集合 $\mathcal{L}$，且每个子列表都关联一个值，我们可以在 $O(m\log m)$ 时间内为 $\mathcal{L}$ 构建数据结构，其中 $m = |\mathcal{L}|$，从而使每次最小值包围子列表查询都能在 $O(\log^2 m)$ 时间内得到回答。

在我们的算法中，每次迭代时，需要为其构建上述数据结构的相关集合 $\mathcal{L}$ 的总大小为 $O(n^3)$。因此，在算法的第 $t$ 次迭代中，这些查询（包括构建数据结构所需的时间）总共耗时 $O(tn^3 \log^2 n)$。

此外，在第 $t$ 次迭代中，算法会执行 $O(tn^2)$ 次操作，以计算索引 $a_i^j$ 和 $b_i^j$。为此，下面的引理给出了一种数据结构。

**引理 10.** 我们可以在 $O(n\log^2 n)$ 时间内为 $P$ 构建一种数据结构，使得对于任意两个点 $p_i$、$p_j \in P$，都能在 $O(\log^2 n)$ 时间内计算出索引 $a_i^j$ 和$b_i^j$。

证明。 该算法与 [31] 中针对单位圆盘情形给出的算法类似，不同之处在于，我们用最远圆盘 Voronoi 图替换最远点 Voronoi 图 [26]。为完整起见，我们在下文给出详细内容。

首先构造一棵完全二叉树 $T$，其叶节点从左到右分别存储 $p_1$，$p_2$，…，$p_n$。对于每个节点 $v \in T$，令 $P_v$ 表示 $P$ 中位于以 $v$为根的子树叶节点上的点的子集。我们在 $\mathcal{D}(P_v)$ [26], 上构造最远圆盘 Voronoi 图 $\mathcal{FVD}(v)$，即以 $P_v$中的点为中心的圆盘集合。我们还在 $\mathcal{FVD}(v)$ [12, 20]上构建点定位数据结构。构造 $\mathcal{FVD}(v)$需要 $O(|P_v|\log |P_v|)$时间 [26]。构造点定位数据结构需要 $O(|P_v|)$时间 [12, 20]。对所有节点 $v \in T$执行上述操作需要 $O(n\log^2 n)$时间。

给定两个点 $p_i$，$p_j \in P$，我们可以按如下方式在 $O$ 中用 $^2 n$ 的对数时间计算索引 $a_i^j$。首先检查 $a_i^j = 0$ 是否成立，即对于所有点 $p \in P$，$|D_{p_i}D_p| \leq 0$ 是否成立。为此，我们通过查找包含 $p_i$ 的 $T$ 的根 $v$ 所对应的 $\mathcal{FVD}(v)$ 单元，计算距离 $p_i$ 最远的 $\mathcal{D}(P)$ 的圆盘 $D_{p'}$。注意，当且仅当 $|D_{p'}D_{p_i}| \leq 0$ 时，$a_i^j = 0$。接着，我们通过测试 $|D_{p_i}D_{p_j}| > 0$ 是否成立来检查 $a_i^j = j$。下文假设 $a_i^j$ 不等于 0 或 $j$。根据 $a_i^j$ 大于还是小于 $j$，分为两种情况。

- 我们首先考虑 $a_i^j > j$的情况。设$v_j$ 为存储$p_j$的 $T$ 的叶节点。从$v_j$开始，我们沿着 $T$ 向上移动，直到遇到第一个节点 $v$，其右子节点 $u$满足以下性质：$p_i$ 与其最远的圆盘 $D_{p'}$ 之间的距离对于 $p' \in P_u$ 大于 $r_i$（这意味着 $|D_{p_i}D_{p'}| > 0$）。令 $v^*$表示这样的节点 $v$。具体而言，从$v = v_j$开始，如果 $v$ 没有右子节点，则令 $v$ 为其父节点。否则，令 $u$ 为 $v$的右子节点。通过对$\mathcal{FVD}$（$P_u$）执行点定位查询，我们找到 $p_i$ 中满足 $p' \in P_u$的最远圆盘 $D_{p'}$。如果 $|D_{p'}D_{p_i}| \leq 0$，则令 $v$ 为其

父节点；否则，令 $v^* = v$。由于 $a_i^j > j$，因此可以保证找到 $v^*$ 。接下来，从 $v^*$的右子节点开始，我们执行自顶向下搜索过程。对于每个节点 $v$，令 $u$ 为其左子节点。我们找到 $p_i$ 中满足 $p' \in P_u$的最远圆盘 $D_{p'}$ 。如果 $|D_{p'}D_{p_i}| > 0$，则令 $v$ 为 $u$；否则，令 $v$ 为其右子节点。该过程最终将到达叶节点 $v$，其存储点 $a_i^j$。总时间为 $O(\log^2 n)$，因为该搜索过程调用点定位查询 $O(\log n)$次，而每次点定位查询需要 $O(\log n)$时间。

- 对于情形 $a_i^j < j$， $D_{p_i}$ 与右侧任意叶节点中的点为中心的圆盘相交，因此上述算法中的自底向上过程最终会到达根节点。注意， $a_i^j$是 $P[1, j-1]$ 中第一个其圆盘与 $D_{p_i}$不相交的点。接下来分情况讨论。若 $|D_{p_1}D_{p_i}| > 0$，则 $a_i^j = 1$ ，算法可以停止。否则，从 $T$最左侧的叶节点 $v = v_1$ 开始，按照上述情形使用相同的算法，即先执行自底向上的过程，再执行自顶向下的过程。该算法的总运行时间为 $O(\log^2 n)$。

以类似的方式，可以在 $O(\log^2 n)$ 时间内计算 $b_i^j$ 。 □

总之，算法的第 $t$ 次迭代需要 $O(tn^3\log^2 n)$ 时间。由于共有 $k$ 次迭代，算法的总运行时间为 $O(k^2n^3\log^2 n)$。因此，我们得到以下定理。

**定理 1.** 给定一个数 $k$ 以及一组 $n$带权圆盘，其中心在平面中处于凸位置，我们可以在 $O(k^2n^3 \log^2 n)$ 时间内，在圆盘图中找到规模至多为 $k$ 的最小权支配集，或者报告不存在这样的支配集。

将定理 1 应用于 $k = n$ ，可得如下结果。

**推论 1.** 给定一组 $n$ 带权圆盘，其中心在平面中处于凸位置，我们可以在 $O(n^5\log^2 n)$ 时间内计算出圆盘图中的最小权支配集。

# 5 非加权情形

在本节中，我们考虑非加权情形。目标是计算圆盘图 $G(P)$。通过将 $P$ 的所有点的权重设为 1 并应用推论 1，即可在 $O(n^5\log^2 n)$ 时间内求解该非加权问题。下面，我们给出一个时间复杂度为$O(k^2n \log n)$ 的算法，其中 $k$ 为最小支配集的规模。

我们沿用加权情形的动态规划方案，但利用 $P$ 中所有点权重相同这一性质，融入贪心策略。第5.1节首先介绍算法。正确性证明见第5.2节。第5.4 节最后讨论实现和时间分析。

## 5.1 算法描述

我们仍然按迭代 $t = 1$、2.....进行。在第 $t$ 次迭代中，我们为每个 $p_i \in P$计算一个包含$\mathcal{L}_t$个 $O$子列表的集合$i$，其中每个子列表 $L$ 都与一个子集 $S_L \subseteq P$相关联，并维持以下算法不变式：(1) $S_L$ 支配 $L$；(2) $p_i \in S_L$；(3) $p_i \in L$；(4) $|S_L| \leq t$。定义$\mathcal{L}_t = \cup_{p_i \in P}\mathcal{L}_t(i)$。因此， $|\mathcal{L}_t| = O(nt)$。

令 $k$ 为最小支配集的大小。我们将证明，经过 $k$ 次迭代后， $\mathcal{L}_k$ 必定包含一个满足 $P$的子列表。因此，如果算法首次计算出满足 $P$ 的子列表，我们就可以停止算法。

最初， $t = 1$，我们的算法对每个 $p_i \in P$ 执行以下操作。我们计算 $L = P(b_i^i, a_i^i)$，并设置 $S_L = \{p_i\}$。我们令 $\mathcal{L}_1(i) = \{L\}$。显然，对于 $L$，所有算法不变式均成立。

一般而言，假设对于每个 $t' \in [1, t-1]$，我们都已计算出一个集合 $\mathcal{L}_{t'}(i)$，其中包含针对每个 $p_i \in P$ 的 $O(t')$ 个子列表，每个子列表 $L' \in \mathcal{L}_{t'}(i)$ 都与一个子集 $S_{L'} \subseteq P$ 相关联，且算法不变式成立，即 $L'$ 被 $S_{L'}$、$p_i \in S_{L'}$、$p_i \in L'$ 和 $|S_{L'}| \leq t'$ 支配。

第 $t$ 次迭代的步骤如下。对于每个点 $p_i \in P$，我们执行逆时针/顺时针处理过程以及双向预处理过程。请注意，与加权情形不同，这些过程现在会采用贪心策略。具体细节如下。

### 5.1.1 逆时针/顺时针处理过程

对于每个点 $p_i \in P$，对于每个 $t' \in [1, t-1]$，我们执行以下逆时针处理过程。仍可参照图2（但可忽略图中的记号 $z$）。

根据针对 $t'$ 的算法不变式，$\mathcal{L}_{t'}(i)$ 的每个子列表都包含 $p_i$。我们首先找出 $\mathcal{L}_{t'}(i)$ 中的子列表，其逆时针端点沿逆时针方向距 $p_i$ 最远。令 $L_1$ 表示该列表，令 $p_{z_1}$ 表示 $L_1$ 的逆时针端点。请注意，如果明确维护 $\mathcal{L}_{t'}(i)$ 中逆时针端点距 $p_i$ 最远的子列表（例如，在第 $t'$ 个过程中，每当我们向 $\mathcal{L}_{t'}(i)$ 添加一个列表时，就更新逆时针端点距 $p_i$ 最远的子列表），则可在 $O(1)$ 时间内找到 $L_1$。接着，我们对 $\mathcal{L}_{t-t'}$ 执行逆时针最远包围子列表查询，以计算一个包含 $p_{z_1+1}$ 的子列表，使其逆时针端点沿逆时针方向距 $p_{z_1+1}$ 最远。令 $L_2$ 表示该子列表，令 $z_2$ 表示 $L_2$ 的逆时针端点。

接下来，我们计算索引 $a_i^{z_2+1}$。根据构造，以下四个子列表的并集是 $P$的一个（连续）子列表：$P(b_i^i, a_i^i)$、$L_1$、$L_2$和 $P(z_2, a_i^{z_2+1})$。将该合并后的子列表记为 $L(t')$，或简记为 $L$。我们令$S_L = S_{L_1} \cup S_{L_2}$。

**观察3.** 所有算法不变式对 $L$成立。

证明. 我们论证算法不变式对 $L$成立，即 $L$ 被 $S_L$、$p_i \in S_L$、$p_i \in L$和 $|S_L| \leq t$支配。事实上，根据 $t'$的算法不变式，$L_1$ 被$S_{L_1}$、$p_i \in S_{L_1}$、$p_i \in L_1$和 $|S_{L_1}| \leq t'$支配。由于 $L_2 \in \mathcal{L}_{t-t'}$，根据 $t-t'$ 的算法不变式，$L_2$ 被 $S_{L_2}$ 和$|S_{L_2}| \leq t-t'$支配。此外，根据定义，两个子列表 $P(b_i^i, a_i^i)$和 $P(z_2, a_i^{z_2+1})$ 被$p_i$支配。综合以上，算法不变式对 $L$成立。 □

在上文针对所有 $t' \in [1, t-1]$ 计算出的 $O(t)$ 个子列表 $L(t')$ 中，我们保留逆时针端点距 $p_i$ 最远的那个 $L$，并将其加入 $\mathcal{L}_t(i)$。

上文针对 $p_i$ 描述的逆时针处理过程至多向 $\mathcal{L}_t(i)$ 添加一个子列表。类似地，我们也针对 $p_i$ 执行顺时针处理过程，该过程也至多向 $\mathcal{L}_t(i)$ 添加一个子列表。

### 5.1.2 双向处理过程

对于每个点 $p_i \in P$，对于每个 $t' \in [2, t-1]$，我们执行以下双向处理过程。仍可参见图 3（但可以忽略图中的记号 $x$ 和 $y$）。

我们在 $\mathcal{L}_{t'}(i)$ 中找到子列表 $L_x$，其逆时针端点距 $p_i$ 最远（沿逆时针方向）；同时在 $\mathcal{L}_{t+1-t'}(i)$ 中找到子列表 $L_y$，其顺时针端点距 $p_i$ 最远（沿顺时针方向）。以下三个子列表构成 $P$ 的一个（连续）子列表：$P(b_i^i, a_i^i)$、$L_x$ 和 $L_y$。令 $L$ 为上述三个子列表的并集。我们设定 $S_L = S_{L_x} \cup S_{L_y}$，并将 $L$ 加入 $\mathcal{L}_t(i)$。

**观察 4.** 所有算法不变式对 $L$ 均成立。

证明。 我们证明所有算法不变式对 $L$都成立。根据 $t'$ 和 $t+1-t'$的算法不变式，$L_x$被 $S_{L_x}$支配，$L_y$被 $S_{L_y}$支配，$|S_{L_x}| \leq t'$、$|S_{L_x}| \leq t+1-t'$和 $p_i$同时属于 $S_{L_x}$和 $S_{L_y}$，

而且 $p_i$ 也同时属于 $L_x$ 和 $L_y$。因此， $S_L$ 支配 $L$、 $|S_L| \leq t$、 $p_i \in S_L$和 $p_i \in L$。由此，算法不变式对 $L$ 成立。 □

上述过程向 $\mathcal{L}_t(i)$ 添加了 $O(t)$ 个子列表。

### 5.1.3 总结

第 $t$ 次迭代针对每个点 $p_i \in P$，为 $\mathcal{L}_t(i)$ 计算 $O(t)$ 个子列表。因此，在第 $t$ 次迭代中，共向 $\mathcal{L}_t$ 添加了 $O(tn)$ 个子列表。如果任一子列表 $L \in \mathcal{L}_t$ 为 $P$，则我们停止该算法，并返回 $S_L$ 作为最小支配集。否则，我们继续进行下一次迭代。

## 5.2 算法正确性

We证明以下引理，该引理确立了算法的正确性 .

**引理11.** 如果算法首次计算出一个为 $P$ 的子列表 $L$，则 $S_L$ 是 $G(P)$的最小支配集。

设 $k$ 是 $G$（$P$）的最小支配集大小。只需证明该算法将在 $k$ 次迭代内停止，即存在一个子列表 $L \in \mathcal{L}_k$ 使得 $L = P$。许多论证与第 4.3 节中加权情形的证明类似。不同之处在于，我们需要论证贪心策略确实有效。

设 $S$ 是一个最小支配集（因此 $|S| = k$），并令 $\phi$ : $\mathcal{A} \to S$ 是引理 1所保证的线可分配。将 $\mathcal{A} = \langle \alpha_1、 \alpha_2.....、 \alpha_m \rangle$ 沿$\mathcal{H}$（$P$）按逆时针方向排列，如前文所述（我们也沿用前文的记号，例如 $\mathcal{A}[g,h]$、 $\mathcal{A}_{p_i}$、 $S[g,h]$）。我们将证明以下引理。

**引理 12**。 对于任意点 $p_i \in S$ 和任意 $t$ （满足 $1 \leq t \leq k$），对于 $p_i$中的任意秩为 $t$ 的子列表 $\mathcal{A}[g,h]$，$\mathcal{L}_t$（$i$）必然包含一个子列表 $L$，满足 $\mathcal{A}[g,h] \subseteq L$。

引理 12 将推出引理 11。事实上，如前所述， $P$ 必为秩为 $k$ 的子列表 $\mathcal{A}[g,h]$，属于 $p_i$，且满足 $\mathcal{A}[g,h] = P$ 与 $S[g,h] = S$。由于 $\mathcal{L}_k(i)$ 包含一个子列表 $L$，其满足 $\mathcal{A}[g,h] \subseteq L$ （根据引理 12 ），我们得到 $L = P$。因此，我们的算法将找到大小为 $k$ 的最优支配集，于第 $k$次迭代中。

在本节的其余部分，我们证明引理 12，即通过对 $t = 1、2、.....、k$ 进行归纳 .

对于基本情形 $t = 1$，假设 $p_i \in S$ 有一个秩为 1 的子列表 $\mathcal{A}[g,h]$。根据定义， $\mathcal{A}[g,h]$ 是组 $\mathcal{A}_{p_i}$ 中唯一的子列表，并且它是 $p_i$ 的主子列表。由于 $p_i \in \mathcal{A}[g,h]$ 且 $P(b_i^i, a_i^i)$ 是 $P$ 中包含 $p_i$ 的极大子列表，我们有 $\mathcal{A}[g,h] \subseteq P(b_i^i, a_i^i)$。根据我们的算法， $\mathcal{L}_1(i)$ 由单个子列表 $L = P(b_i^i, a_i^i)$ 组成，且满足 $S_L = \{p_i\}$。因此，引理 12 对 $t = 1$ 成立。

现在假设引理 12 对每个 $t' \in [1, t-1]$都成立，即对于每个子列表 $\mathcal{A}[g', h']$，它是某个 $t'$ 的秩为 $p_i \in S$ 的子列表， $\mathcal{L}_{t'}(i)$ 都包含一个满足 $L'$ 的子列表 $\mathcal{A}[g', h'] \subseteq L'$。接下来，我们证明该引理对 $t$成立，即对于每个子列表 $\mathcal{A}[g,h]$，它是某个 $t$ 的秩为 $p_i \in S$ 的子列表， $\mathcal{L}_t(i)$ 都包含一个满足 $L$ 的子列表 $\mathcal{A}[g,h] \subseteq L$。

假设 $p_i$ 是子列表 $\mathcal{A}[g,h]$的秩为 $t$ 的中心。根据定义， $\mathcal{A}[g,h]$ 的两个端子列表中至少有一个属于组 $\mathcal{A}_{p_i}$，并且 $\mathcal{A}[g,h]$ 包含 $p_i$的主子列表。根据 $p_i$ 的主子列表是否为 $\mathcal{A}[g,h]$的端子列表，可分为两种情况。

## 5.3 情形（1）： $p_i$ 的主子列表是$\mathcal{A}[g,h]$的端子列表 ]

不失一般性，我们假设 $\mathcal{A}[g,h]$ 的顺时针端子列表是 $p_i$的主子列表，即 $\alpha_g$。引理5 仍然成立。

定义 $h'$ 为 $h$，若 $p_i$ 是开放中心；若 $h-1$，则 $p_i$ 是闭中心。因此，在两种情况下，$p_{i'}$ 始终是 $(t-t')$ 阶的 $\mathcal{A}[f+1,h']$中心，其中 $p_{i'}$ 和 $f$ 来自引理 5。令 $p_j$ 为 $\mathcal{A}[f+1,h']$的逆时针端点。

现在，考虑我们的算法的第 $t$次迭代：在逆时针处理过程中处理 $p_i$，并检查值 $t'$。在此步骤中，算法为 $\mathcal{L}_t(i)$构造子列表$L$，作为四个子列表的并集：$P(b_i^i, a_i^i)$、$\mathcal{L}_{t'}(i)$的一个子列表 $L_1$（其逆时针端点距 $p_i$最远）、$\mathcal{L}_{t-t'}$ 中包含 $p_{z_1+1}$ 且其逆时针端点距 $p_{z_1+1}$最远的子列表 $L_2$，以及 $P[z_2, a_i^{z_2+1})$，其中 $p_{z_1}$和 $p_{z_2}$分别是 $L_1$和 $L_2$的逆时针端点。此外，$S_L = S_{L_1} \cup S_{L_2}$。下面的引理完成了我们对情况（1）的证明。

**引理13.** $\mathcal{A}[g,h] \subseteq L$.

证明。根据归纳法，$\mathcal{L}_{t'}(i)$ 包含一个子列表 $L_1'$，使得 $\mathcal{A}[g,f] \subseteq L_1'$；类似地，$\mathcal{L}_{t-t'}(i')$ 包含一个子列表 $L_2'$，使得 $\mathcal{A}[f+1,h'] \subseteq L_2'$。仍可参考图5（但记号 $z$ 在图中可以忽略）。

由于 $L_1' \in \mathcal{L}_{t'}(i)$，且 $L_1$ 是 $\mathcal{L}_{t'}(i)$ 中逆时针端点距离 $p_i$ 最远的子列表，因此我们有 $L_1' \subseteq P(b_i^i, a_i^i) \cup L_1$。由于 $\mathcal{A}[g,f] \subseteq L_1'$，我们得到 $\mathcal{A}[g,f] \subseteq P(b_i^i, a_i^i) \cup L_1$。

我们声称 $p_{z_1+1}$ 必须属于 $\mathcal{A}[f+1,h']$ 的某个子列表。论证与引理6中的相同（例如，假设该引理中的所有点权重都相同）。因此，$P[z_1+1, j] \subseteq \mathcal{A}[f+1,h']$。由于 $\mathcal{A}[f+1,h'] \subseteq L_2'$，我们有 $P[z_1+1,j] \subseteq L_2'$。由于 $L_2$ 是 $\mathcal{L}_{t-t'}$ 中包含 $p_{z_1+1}$ 且逆时针端点距离 $p_{z_1+1}$ 最远的子列表，而 $L_2'$ 位于 $\mathcal{L}_{t-t'}$ 中且也包含 $p_{z_1+1}$，因此我们得到 $P[z_1+1,j] \subseteq L_2$。由于 $\mathcal{A}[g,f] \subseteq P(b_i^i, a_i^i) \cup L_1$，$z_1$ 是 $L_1$ 的逆时针端点，而 $j$ 是 $\mathcal{A}[g,h']$ 的逆时针端点，我们可以推出 $\mathcal{A}[g,h'] = \mathcal{A}[g,f] \cup \mathcal{A}[f+1,h'] \subseteq P(b_i^i, a_i^i) \cup L_1 \cup P[z_1+1,j] \subseteq P(b_i^i, a_i^i) \cup L_1 \cup L_2$。

回顾一下，$L$ 是以下四个子列表的并集：$P(b_i^i, a_i^i)$、$L_1$、$L_2$和 $P(z_2, a_i^{z_2+1})$。根据 $p_i$ 是开放的还是闭的秩为$t$的中心，分为两种情况。

- 如果 $p_i$ 是开放中心，则 $h' = h$，从而 $\mathcal{A}[g,h] = \mathcal{A}[g,h']$。上文已经证明，$\mathcal{A}[g,h'] \subseteq P(b_i^i, a_i^i) \cup L_1 \cup L_2 \subseteq L$。因此，$\mathcal{A}[g,h] \subseteq L$ 成立。
- 如果 $p_i$ 是闭中心，则 $h' = h-1$ 且 $\mathcal{A}[g,h] = \mathcal{A}[g,h'] \cup \alpha_h$。如果 $\mathcal{A}[g,h] \subseteq P(b_i^i, a_i^i) \cup L_1 \cup L_2$，那么我们仍有 $\mathcal{A}[g,h] \subseteq L$。否则，必有 $p_{z_2+1} \in \alpha_h$。因此，由于 $p_i$支配 $\alpha_h$，且$p_j$是$\mathcal{A}[g,h']$的逆时针端点，根据 $a_i^{z_2+1}$的定义，$\alpha_h \subseteq P[j+1, a_i^{z_2+1})$。由于 $\mathcal{A}[g,h'] \subseteq P(b_i^i, a_i^i) \cup L_1 \cup L_2$，我们得到 $\mathcal{A}[g,h] = \mathcal{A}[g,h'] \cup \alpha_h \subseteq P(b_i^i, a_i^i) \cup L_1 \cup L_2 \cup \alpha_h \subseteq P(b_i^i, a_i^i) \cup L_1 \cup L_2 \cup P[j+1, a_i^{z_2+1})$。由于 $p_j \in L_2$ 且 $z_2$是 $L_2$的逆时针端点，我们有 $L_2 \cup P[j+1, a_i^{z_2+1}) \subseteq L_2 \cup P(z_2, a_i^{z_2+1})$。因此，$P(b_i^i, a_i^i) \cup L_1 \cup L_2 \cup P[j+1, a_i^{z_2+1}) \subseteq P(b_i^i, a_i^i) \cup L_1 \cup L_2 \cup P(z_2, a_i^{z_2+1}) = L$。所以，我们得到 $\mathcal{A}[g,h] \subseteq L$。

总之，$\mathcal{A}[g,h] \subseteq L$ 在两种情形下均成立。因此引理得证。 □

### 5.3.1 情形（2）：$p_i$ 的主子列表不是$\mathcal{A}[g,h]$的端子列表

由于 $p_i$是一个秩为$t$ 的中心，$\mathcal{A}[g,h]$的两个端子列表之一必定属于组 $\mathcal{A}[g,h]$ $\mathcal{A}_{p_i}$。不失一般性，假设 $\mathcal{A}[g,h]$的逆时针端子列表，即 $\alpha_h$，属于 $\mathcal{A}_{p_i}$。注意，$\alpha_h$ 必定是 $p_i$的次子列表。引理7 在此处仍然适用。令 $t'$ 和 $f$ 取自引理7。

现在考虑算法的第 $t$ 次迭代，其中通过双向处理过程处理 $p_i$，并检查值 $t'$。在这一步中，算法构造子列表 $L \in \mathcal{L}_t(i)$，它是以下三个子列表的并集：$P(b_i^i, a_i^i)$、$\mathcal{L}_{t'}(i)$ 中逆时针端点距离 $p_i$ 最远的子列表 $L_x$，以及 $\mathcal{L}_{t+1-t'}(i)$ 中顺时针端点距离$P$最远的子列表 $L_y$。此外，$S_L = S_{L_x} \cup S_{L_y}$。下面的引理完成了情形（2）的证明。

**引理 14.** $\mathcal{A}[g,h] \subseteq L$.

证明。 根据关于 $t'$， $\mathcal{L}_{t'}(i)$ 的归纳假设，其中包含一个子列表 $L'_x$ 满足 $\mathcal{A}[f,h] \subseteq L'_x$；同样，$\mathcal{L}_{t+1-t'}(i)$ 包含一个子列表 $L'_y$ 满足 $\mathcal{A}[g,f] \subseteq L'_y$。仍可参见图 7（但图中的记号$p_x$ 和$p_y$ 可以忽略）。

由于 $L_x$ 是 $\mathcal{L}_{t'}(i)$中逆时针端点距离 $p_i$ 和 $L'_x \in \mathcal{L}_{t'}(i)$最远的子列表，我们得到 $L'_x \subseteq P(b_i^i, a_i^i) \cup L_x$。类似地，我们有 $L'_y \subseteq P(b_i^i, a_i^i) \cup L_y$。因此，由于 $\mathcal{A}[f,h] \subseteq L'_x$和 $\mathcal{A}[g,f] \subseteq L'_y$，我们可以推导出
$\mathcal{A}[g,h] = \mathcal{A}[g,f] \cup \mathcal{A}[f,h] \subseteq L'_x \cup L'_y \subseteq L_x \cup L_y \cup P(b_i^i, a_i^i) = L$。 □

### 5.4 时间分析与实现

在每个 $t$次迭代中，当 $1 \le t \le k$时，我们执行 $O(tn)$ 次操作来计算索引 $a_i^j$ 和 $b_i^j$ 以及$O(tn)$ 次逆时针/顺时针最远包围子列表查询：给定点 $p \in P$，从集合 $\mathcal{L}$ 的子列表中找出包含 $p$ 且其逆时针/顺时针端点距离$p$最远的子列表。计算索引 $a_i^j$ 和 $b_i^j$ 需要 $O(\log^2 n)$ 时间，依据引理10。逆时针/顺时针最远包围子列表查询问题已在 [31]中得到研究；我们在以下引理中总结该结果。

**引理 15.** ( [31]) 给定一个由 $P$ 的 $m$ 个子列表组成的集合 $\mathcal{L}$，我们可以在 $O$ 上构建一个用于 $\mathcal{L}$ 的数据结构，耗时为 $m\log\ m)$；这样，每次逆时针/顺时针最远包围子列表查询都可以在 $O(\log m)$ 时间内得到回答。

在我们的算法中，对于每次 $t$ 次迭代，需要构建上述数据结构的相关集合 $\mathcal{L}$ 的总大小为 $O(tn)$。因此，每次 $t$ 次迭代的总耗时为 $O(tn\log^2 n)$。由于共有 $k$ 次迭代，整个算法的运行时间为 $O(k^2 n \log^2 n)$。我们以以下定理结束本节。

**定理 2.** 给定平面上一组中心处于凸位置的 $n$ 个圆盘，可以在 $k^2 n\log^2 n)$ 时间内计算出该圆盘图的一个最小支配集，其中 $k$ 为最小支配集的大小。

## 参考文献


[1] Alok Aggarwal、Leonidas J. Guibas、James Saxe 和 Peter W. Shor。用于计算凸多边形 Voronoi 图的线性时间算法。《离散与计算几何》，4:591–604，1989年。 doi:10.1007/BF02187749。 2[2] Frank Ángel Hernández Mira、Ernesto Parra Inza、José Marıia Sigarreta Almira 和 Nodari Vakhania。图中最小支配集的多项式时间近似算法。《理论计算机科学》，930:142–156，2022年。 doi:10.1016/j.tcs.2022.07.020。 1[3] Paul Balister、Béla Bollobás、Amites Sarkar 和 Mark Walters。随机 k 近邻图的连通性。《应用概率进展》，37(1):1–24，2005年。 doi:doi:10.1239/aap/1113402397。 1[4] Sayan Bandyapadhyay、William Lochet、Daniel Lokshtanov、Saket Saurabh 和 Jie Xue。针对 h-小子式自由图上的割与环命中问题的亚指数参数化算法。载于*2022*年 *ACM-SIAM* 离散算法年度研讨会（*SODA*）论文集，第 2063–2084 页，2022年。 doi:10.1137/1.9781611977073.82。 1[5] Ahmad Biniaz、Mahdi Amani、Anil Maheshwari、Michiel Smid、Prosenjit Bose 和 Jean-Lou De Carufel。处于凸位置的点的平面 1.88-伸展子图。《计算几何杂志》，7(1):520–539，2016年。 doi:10.20382/jocg.v7i1a21。 2

[6]Marthe Bonamy、Édouard Bonnet、Nicolas Bousquet、Pierre Charbit、Panos Giannopoulos、Eun Jung Kim、Paweł Rzazewski、Florian Sikora 和 Stéphan Thomassé。圆盘图和单位球图上最大团的高效多项式时间近似方案及亚指数算法。ACM 汇刊，68(2):1–38，2021年。doi:10.1145/3433160。1[7] Onur Çağırici 和 Bodhayan Roy。凸位置下圆盘的最大团。载于《计算几何：青年研究者论坛》（CG:YRF）摘要集，第 21:1–21:3 页，2018年。网址：https://www.computational-geometry.org/old/YRF/cgyrf2018.pdf。2[8] Bernard Chazelle、Herbert Edelsbrunner、Michelangelo Grigni、Leonidas Guibas、Micha Sharir 和 Emo Welzl。凸集弱 $\epsilon$-网的改进界。载于第 25 届 ACM 计算理论研讨会（STOC）论文集，第 495–504 页，1993年。doi:10.1145/ 167088.167222。2[9] Jongmin Choi、Jaegun Lee 和 Hee-Kap Ahn。处于凸位置的平面点的高效 $k$-中心算法。载于第 18 届算法与数据结构研讨会（WADS）论文集，第 262–274 页，2023 年。doi:10.1007/978-3-031-38906-1_18。2[10] Brent N. Clark 、 Charles J. Colbourn 和 David S. Johnson。单位圆盘图。《离散数学》，86:165–177，1990年。doi:10.1016/0012-365X(90)90358-O。1, 2[11] Minati De 和 Abhiruk Lahiri。通过局部搜索求解几何支配集与集合覆盖。计算几何，113(C):102007，2023年。doi:10.1016/j.comgeo.2023.102007。1[12] Herbert Edelsbrunner、Leonidas J. Guibas 和 J. Stolfi。单调剖分中的最优点定位。SIAM 计算杂志，15(2):317–340，1986年。doi:10.1137/0215023。13[13] David Eppstein。计算几何问题的图论解法。载于第 35 届图论计算机科学概念国际研讨会（WG）论文集，第 1–16 页，2009年。doi:10.1007/978-3-642-11409-0_1。1[14] Thomas Erlebach、Klaus Jansen 和 Eike Seidel。几何交图的多项式时间近似方案。SIAM 计算杂志，34(6):1302–1323，2005年。doi:10.1137/S0097539702402676。1[15] Jared Espenant、J. Mark Keil 和 Debajyoti Mondal。在圆盘图中寻找最大团。载于第 39 届计算几何国际研讨会（SoCG）论文集，第 30:1–30:17 页，2023年。doi:10.4230/LIPIcs.SoCG.2023.30。1[16] Steven Fortune。用于 Voronoi 图的扫描线算法。Algorithmica，2:153–174，1987年。doi:10.1007/BF01840357。4[17] Matt Gibson 和 Imran A. Pirwani。圆盘图中支配集的算法：突破对数 $n$ 障碍。载于第 18 届欧洲算法研讨会（ESA）论文集，第 243–254 页，2010年。doi:10.1007/978-3-642-15775-2_21。1[18] Karl Kammerlander。C 900——一种具有最优频率利用率的先进移动无线电话系统。IEEE 车辆技术汇刊，33(3):205–213，2013年。doi:10.1109/T-VT.1984.24007。1[19] J. Mark Keil 和 Debajyoti Mondal。圆盘图中的最大团问题变得简单。载于第 41 届计算几何国际研讨会（SoCG）论文集，第 63:1–63:16 页，2025年。doi:10.4230/LIPIcs.SoCG.2025.63。1

[20] 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。13[21] Andrzej Lingas。关于凸平面点集的贪心三角剖分的近似表现与实现。载于第2届计算几何研讨会（SoCG）论文集，第72–79页，1986年。doi:10.1145/10515.10523。2[22] Daniel Lokshtanov、Fahad Panolan、Saket Saurabh、Jie Xue 和 Meirav Zehavi。圆盘图近似方案框架。载于2023年 ACM-SIAM 离散算法年度研讨会（SODA）论文集，第2228–2241页，2023年。doi:10.1137/1.9781611977554.ch84。1[23] Nabil H. Mustafa 和 Saurabh Ray。几何击中集问题的改进结果。《离散与计算几何》，44:883–895，2010年。doi:10.1007/s00454-010-9285-9。1[24] Ngoc Doanh Nguyen、Thi Ha Duong Phan、Thi Ngoc Anh Nguyen、Alexis Drogoul 和 Jean- Daniel Zucker。基于圆盘图的模型：一种连接基于主体的模型与动力系统的图论方法。载于国际计算与通信技术、研究、创新及未来愿景会议（IEEE RIVF）论文集，第1–4页，2010年。doi:10.1109/RIVF.2010.5633033。1[25] Charles E. Perkins 和 Pravin Bhagwat。自组网按需距离矢量路由。载于第2届 IEEE 移动计算系统与应用研讨会（WMCSA）论文集，第90–100页，1999年。doi:10.1109/MCSA.1999.749281。1[26] David Rappaport。计算一组圆盘的最远站点Voronoi图。载于第1届算法与数据结构研讨会（WADS）论文集，第57–66页，1989年。doi:10.1007/ 3-540-51542-9_7。13[27] Dana S. Richards 和 Jeffrey S. Salowe。凸点集的直线Steiner最小树算法。载于第2届斯堪的纳维亚算法理论研讨会（SWAT）论文集，第447卷，第201–212页，1990年。doi:10.1007/3-540-52846-6_90。2[28] Micha Sharir。一组平面圆盘的相交与最近点对问题。《SIAM 计算杂志》，14(2):448–468，1985年。doi:10.1137/0214034。4[29] Vishwanath R. Singireddy、Manjanna Basappa 和 Joseph S.B. Mitchell。 $k$算法：平面上处于凸位置的点的分散。载于第9届国际算法与离散应用数学会议（CALDAM）论文集，第59–70页，2023年。doi:10.1007/978-3-031-25211-2_5。2[30] Anastasiia Tkachenko 和 Haitao Wang。计算单位圆盘图中的最大团。载于第37届加拿大计算几何会议（CCCG）论文集，第283–291页，2025年。URL: https://cccg-wads-2025.eecs.yorku.ca/cccg-papers/6B2.pdf。2[31] Anastasiia Tkachenko 和 Haitao Wang。凸位置平面点的支配集、独立集、离散k-中心、分散及相关问题。载于第42届理论计算机科学专题国际研讨会（STACS）论文集，第73:1–73:20页，2025年。doi:10.4230/LIPIcs.STACS.2025.73。2, 4, 13, 18[32] Viet Xuan Truong、Hiep Xuan Huynh、Minh Ngoc Le 和 Alexis Drogoul。基于单位圆盘图技术的监控网络建模——湄公河三角洲地区昆虫入侵监测应用。载于第15届多主体系统原理与实践国际会议（PRIMA）论文集，第228–242页，2012年。doi:10.1007/978-3-642-32729-2_16。1

[33] Erik Jan van Leeuwen。圆盘图的更优近似方案。载于第*10*届斯堪的纳维亚算法理论研讨会（*SWAT*）论文集，第316–327页，2006年。doi:10.1007/11785293_30。1

[34] Da-Wei Wang 和 Yue-Sun Kuo。关于两个几何定位问题的研究。《信息处理快报》，28(6)：281–286，1988年。doi:10.1016/0020-0190(88)90174-3。1，2