利用单位正方形覆盖点的最小重叠度覆盖问题的更快算法

Siddhartha Sarkar

Com计算机科学与自动化系，印度科学研究所，班加罗尔 u

arXiv:2301.13108v1 [cs.CG] 30 Jan 2023

摘要。 Biedl 等人在 CG 2021 上提出了最小重叠度覆盖问题，延续了 Erlebach 和 van Leeuwen 在 SODA 2008 上的开创性工作。他们证明，给定一组点和一组轴平行单位正方形，确定这组点的最小重数覆盖数是 NP-困难的，并针对最小重数覆盖数有常数上界的实例，给出了一个多项式时间的 2-近似算法。Durocher 等人最近针对一般情况提出了一个多项式时间的 $(8+\epsilon)$-近似算法，其中最小重数覆盖数为 $\omega(1)$，且 $\epsilon > 0$ 为任意固定值。他们采用标准的网格分解技术将问题划分为子问题，并设计了复杂的动态规划方案来求解每个子问题；每个子问题由一个单位边长正方形网格单元定义。随后，他们合并各子问题的解，得到最终的层覆盖。我们采用水平条带分解技术将问题划分为子问题。对于一般问题，当一个较小的常数 $\epsilon > 0$ 满足条件时，我们的算法利用简单的贪心启发式，得到一个 $(27+\epsilon)$-近似算法。与 Durocher 等人的算法相比，我们的算法运行速度显著更快。对于输入正方形被一条水平线相交的特殊情况，我们还给出了一个快速的 2-近似算法。该特殊情况的难度目前仍未解决。我们提出的算法有望推广到采用其他几何对象进行最小重叠度覆盖的情形，例如单位圆盘、全等矩形等。

1. 引言

集合覆盖是组合优化中的一个基本问题。给定一个范围空间（X，$\mathcal{R}$），它由一个集合 X 和一个称为范围的子集族 $\mathcal{R}$ 构成，这些子集都是 X 的子集。目标是计算 $\mathcal{R}$ 的一个最小基数子集，使其覆盖 X 中的所有点。将最小集合覆盖近似到对数因子以内是 NP-困难的 [Raz 和 Safra，1997年，Feige， 1998]。当范围由几何对象导出时，这个问题称为几何集合覆盖问题。即使对于简单的二维几何对象，例如平面上的单位正方形，计算最小基数集合覆盖仍然是 NP-困难的 [Fowler 等人， 1981]。关于为各种几何集合覆盖问题设计近似算法，已有大量研究（参见 [Agarwal 和 Pan，2014年；Clarkson 和 Varadarajan，2007年；Hochbaum 和 Maass，1985年；Chan 和 Grant，2014年；Mustafa 等人，2014年；Erlebach 和 van

电子邮箱：siddharthas1@iisc.ac.in。

Leeuwen, 2010]）。在许多情况下，几何版本的覆盖问题都可以高效求解或获得良好的近似。几何集合覆盖问题的许多变体可应用于设施选址、无线网络中的干扰最小化、超大规模集成电路设计等领域 [Demaine 等人，2006年，C˘alinescu 等人， 2004]。本文研究以下覆盖问题。

问题陈述。 给定一组几何对象 $\mathcal{S}$， $\mathcal{S}$ 的**重叠度** 是 $\mathcal{S}$ 中具有非空公共交集的任意子集的最大基数。若一组对象 $\mathcal{S}$ 覆盖一组点 P，则对于每个点 $p \in P$，都存在一个对象 $S \in \mathcal{S}$，使得 p 包含于 S 中。给定一组对象（例如，单位正方形） $\mathcal{S}$，以及平面上的一组点 P，目标是选取一个子集 $\mathcal{S}' \subseteq \mathcal{S}$，使得 P 中的每个点至少被 $\mathcal{S}'$ 中的一个对象覆盖，同时使重叠度最小。

本文研究最小重叠度覆盖问题，其中用于覆盖的对象是单位边长正方形。

1.1. 我们的贡献

我们设计了一种简单的贪心启发式算法，用于平面上的轴平行单位正方形，其近似因子为 $(27+\epsilon)$，其中 $\epsilon > 0$ 是一个较小的常数。我们的算法运行时间为 $O(n^2m^2)$，其中 n 是输入点的数量， m 是输入正方形的数量。与 Durocher 等人提出的动态规划算法相比，我们的算法快得多；该算法需要 $O(n + m^8k^4 \log\ k + m^8 \log\ m \log\ k)$ 的运行时间，其中 k 是最优重叠度值。我们的算法易于实现，也易于扩展到其他覆盖对象。

1.2. 相关工作

最小重叠度覆盖问题是最小成员数集合覆盖（即 MMSC）问题的一种推广。给定点集 P 和由 P 的子集构成的族 $\mathcal{S}$，MMSC 的目标是找到一个子集 $\mathcal{S}' \subseteq \mathcal{S}$，使其覆盖 P，同时最大限度地减少包含 P 中某个公共点的 $\mathcal{S}'$ 中元素的数量。 [Kuhn 等人 2005] 提出了这一一般问题，并给出一种基于线性规划的技术，得到近似因子 $\ln n$，该因子与困难性下界相匹配。在 MMSC 问题中，成员关系是在 P 的各个点处衡量的。 [Erlebach 和 van Leeuwen 2008] 证明，对于由单位圆盘或单位正方形覆盖点集的最小成员数集合覆盖问题，该问题是 NP-困难的，且无法以小于 2 的比率进行近似。对于单位正方形，他们提出了一种 5近似算法，前提是最小重叠度值有常数上界。在最小重叠度覆盖（简称 MPC）问题中，重叠度（即成员关系）是在平面上的任意点处衡量的。[Biedl 等人 2021] 证明，对于单位正方形和单位圆盘，MPC 问题均为 NP-困难；除非 P=NP，否则不存在近似比小于 2 的多项式时间近似算法。当最小重叠度值有常数上界时，他们针对单位正方形和单位圆盘分别给出了该问题的多项式时间 2-近似算法。

2. 最小层覆盖

我们的方法是将问题划分为若干不同的子问题并分别求解，然后合并这些解，得到原问题的层覆盖。首先，我们来定义并求解最小重叠度覆盖问题的一种特殊情况。这里使用的技巧今后也会派上用场。

2.1. 正方形与水平线相交

假设输入正方形都与一条水平线相交，且输入点都至少位于一个输入正方形内。参见图（1）。对于这种特殊情况，我们给出一种算法，计算出的最小重叠度覆盖的重叠度至多是最优层数的两倍。该算法表述和分析起来都很简单。首先，我们将问题划分为两个子问题：一个对应于水平线上方的输入点，另一个对应于水平线下方的输入点。

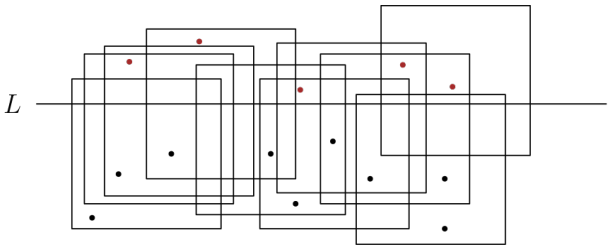


图*1.* 一种特殊情形，其中所有输入正方形 都与一条水平线相交 L，且点位于两侧 s.

定义2.1. 考虑一条水平线 L。现在考虑一组 $\mathcal{S}$ 与 L相交的单位正方形。设 P 为位于 L 下方的一组点，且每个点至少位于 $\mathcal{S}$中的一个正方形内。那么，**MPCSIHL1** 是这样一个问题：计算用 $\mathcal{S}$中的正方形对 P 进行覆盖所需的最小重叠度。

MPCSIHL1 代表由一条**水平**线**相交**的单位**正方形**的**最小重叠度覆**盖问题，其中点仅位于一 侧（即水平线的上方或下方）。参见图（2）

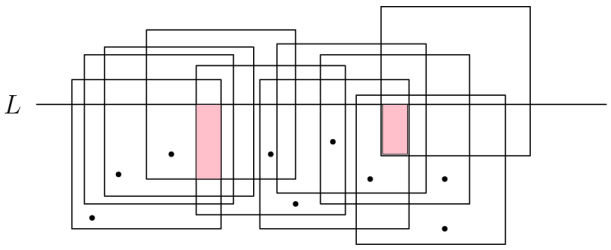


图*2*。 一个这样的 **MPCSIHL1** 问题实例，其中所有输入点都位于 L 下方。粉色区域是重叠区域，其重叠度值为 5。

对于一组正方形 $\mathcal{S}$，我们将平面上最大 深度 的区域称为重叠区域 ，并将其记为 $clique(\mathcal{S})$。因此，重叠区域 是直线 L 下方的公共交集区域。相应的深度就是重叠度，记为 $ply(\mathcal{S})$。注意，一组正方形 $\mathcal{S}$ 的重叠区域可能不唯一。参见图2。算法（1）是一种逐步计算最小重叠度覆盖的启发式算法。该启发式算法采用贪心规则。对于多组正方形 S_1、 S_2、.....、 S_k，我们定义以下贪心准则 来决定优先选择哪一组。

贪心准则。如果一组正方形有多个重叠区域，则将最右侧的重叠区域作为其代表项。使用以下优先规则进行选择并打破平局。

1. 优先选择具有最小重叠度值的正方形组。
2. 如果规则 (1) 导致平局，则优先选择最左侧重叠区域右边的正方形集合 de.

3. 如果规则 (2) 也导致平局，则优先选择重叠区域最窄的正方形集合。

4. 如果仍然出现平局，则任意选择一个候选正方形集合。

算法 1： MPCSIHL1 精确算法

输入：一条水平线 L，一个与 L 相交的 m 个单位正方形集合 $\mathcal{S}$，以及位于 L 下方的 n 个点集合 P，其中每个点都至少位于 $\mathcal{S}$ 中的一个正方形内。输出：返回一个覆盖 P 的正方形集合 $\mathcal{S}' \subseteq \mathcal{S}$，同时使重叠度最小。1 将 P 中的点按从左到右排序。2 设点的排序结果为 p_1、p_2、……、p_n。3 $T \leftarrow \infty$ // 一个二维数组 T，维度为 $n \times m$，其中每个条目初始化为 ∞ 或不可行。4 对 $i \leftarrow 1$ 到 n 执行 5 对 $j \leftarrow 1$ 到 m 执行 6 $T[i,j] = ComputeEntry(i, j, T)$ 7 结束 8 结束 9 根据贪心准则，获取 n 行中的最佳条目，记为 Sol。10 返回 Sol

过程 ComputeEntry(·) 展示了表中各个条目的计算方式。注意，条目 $T[i,j]$考虑了表 T中第 $i-1$行的所有可行解。如果对于某个 $k \in [m]$，$T[i,j]$ 是由 $T[i-1,k]$ 与 ^{s}j取并集得到的，那么我们就称 $T[i-1,k]$ 是 $T[i,j]$ 的父项，并且 $T[i,j]$ 是从 $T[i-1,k]$推导而来的。直观来说，$T[i,j]$ 表示针对 $P_i = \{p_1, \ldots, p_i\}$ 的最小重叠度覆盖，遵循贪心准则，使得点 p_i 由正方形 $s_j \in T[i,j]$覆盖。从 Sol开始，可以沿着父项追溯一条路径，到达表 T第一行中的某个条目。该路径在表的每一行中各包含一个节点。

计算结束后，在对应点 p_i 的 T 中第 i 行，可能有多个表项/覆盖达到相同的重叠度。在下一次迭代中，计算 $T[i+1,j]$ 时，我们将使用 贪心准则 从第 i 行中选出最佳表项。

将 $\{p_1, p_2, \ldots, p_i\}$ 中最左侧的 i 个输入点组成的集合记为 P_i。表项 $T[i,j]$ 是 P_i 的一个可行层覆盖，并且满足 ^{s}j 包含在解中的约束。算法（1）计算出 P 的一个集合覆盖 Sol。我们断言，我们的解是最优的。为了证明该断言的正确性，我们将证明以下**循环不变式**。

命题 2.2. （循环不变式）在算法（*1*）的第 *4* 行外层 *for* 循环第 i 次迭代结束时，对于每个满足 $1 \le l \le i$ 的行 l，都存在一个表项 $T(l,j)$，它对应于点集 P_l 的最小重叠度集合覆盖，并且按照贪心准则是最优的；j 对每一行可能不同。

函数 `ComputeEntry(i, j, T)`：**如果** $i == 1$，**则** **如果** $p_i \in s_j$，**则** **返回** $\{s_j\}$ **否则** **返回** ∞ 令 $\mathcal{F}_{i-1}$ 为第 $(i-1)$ 行中可行解的集合，即值有限的表项。**对于** $F \in \mathcal{F}_{i-1}$，**执行** 计算 $F \cup \{s_j\}$ 的重叠区域从第 9 行 for 循环中计算出的解里，按照贪心准则选出最优表项，记为 Sol_{ij}。 **返回** Sol_{ij}

证明。 首先，我们考虑基本情形。对于点 p_1，我们的算法会在 T 的第一行填入可行解，每个解都只包含一个正方形；也就是说，对于每个覆盖 p_1 的正方形 s_j，对应的表项 $T(1, j)$ 是一个单元素集合 $\{s_j\}$。每个可行解的重叠度为 1，这显然是最优的。第 1 行中对应于包含 p_1 的最左侧正方形的表项，是基于我们的 贪心准则 得到的 P_1 的最优解。

我们固定一个 $1 < i < n$。我们的归纳假设是，算法已经计算出外层 *for* 循环前 $(i-1)$ 次迭代的最优解。具体而言，我们假设在前 $(i-1)$ 次迭代之后，我们的 循环不变式 成立。

我们需要证明，在外层 *for* 循环第 i 次迭代结束时，存在一个包含点 p_i 的正方形 s_j，使得表项 $T[i, j]$ 是满足贪心准则的 P_i 的最优层覆盖。我们将这样的解称为最优参数化解。

在外层*for* 循环的第 i 次迭代后，令 Sol_i 表示存储在表 T 第 i 行、针对 $1 \le i \le n$ 的最优集合覆盖（按照贪心准则）。为证明矛盾，假设存在一个更优的解 OPT_{ij^*}，针对 P_i 和 $p_i \in s_{j^*}$ 以及 $s_{j^*} \in OPT_{ij^*}$。这意味着以下情况之一成立：

$$ply(OPT_{ij^*}) < ply(Sol_i) \tag{1}$$

Or

$$ply(OPT_{ij^*}) = ply(Sol_i), \quad \text{but } OPT_{ij^*} \text{ is better than } Sol_i \text{ as per the } \textit{greedy criteria}. \tag{2}$$

我们先考虑情况 (1)。我们的算法本质上是将正方形 s_j 与针对 P_{i-1} 的可行解 $T(i-1, j')$ 结合起来构造 Sol_i。如果 $s_j \in T(i-1, j')$ 或 s_j 与 $T(i-1, j')$ 的重叠区域不相交，那么 $T(i-1, j') \cup \{s_j\}$ 就是针对 P_i、重叠度为 $ply(T(i-1, j'))$ 的可行解。

令 $OPT_{(i-1)j''} \subseteq OPT_{ij^*}$ 为 P_{i-1} 的一个集合覆盖（我们并不声称它具有最小重叠度），且其中没有冗余正方形。从 OPT_{ij^*} 构造 $OPT_{(i-1)j''}$ 的一种方法，是直接移除覆盖 p_i 的正方形（如果存在）。

完全来自 OPT_{ij^*}。令 $s_{j''} \in OPT_{(i-1)j''}$ 和 $p_{i-1} \in s_{j''}$。根据子集关系，我们有

$$ply(OPT_{(i-1)j''}) \leq ply(OPT_{ij^*}) \tag{3}$$

根据归纳假设，我们有

$$ply(Sol_{i-1}) \leq ply(OPT_{(i-1)j''}) \tag{4}$$

由于我们的算法对每个新点至多添加一个正方形，因此我们有

$$ply(Sol_i) \leq 1 + ply(Sol_{i-1}) \tag{5}$$

因此，由（1）、（3）、（4）和（5），我们得出

$$ply(OPT_{(i-1)j''}) = ply(OPT_{ij^*}) = ply(Sol_{i-1}) = ply(Sol_i) - 1$$

现在考虑针对 P_{i-1} 的解 $OPT_{(i-1)j''}$ 和针对 P_i 的解 OPT_{ij^*}。OPT_{ij^*} 与 $OPT_{(i-1)j''}$ 之间存在两种关系。

情况1：∃ 一个覆盖 p_i 的正方形 $s_{j^*} \in OPT_{ij^*} \setminus OPT_{(i-1)j''}$，使得 s_{j^*} 不与 $OPT_{(i-1)j''}$ 的重叠区域相交。

情况1：$OPT_{(i-1)j''}$ 已经覆盖点 p_i，因此 $OPT_{ij^*} = OPT_{(i-1)j''}$。

图 (3) 展示了这两种情况。

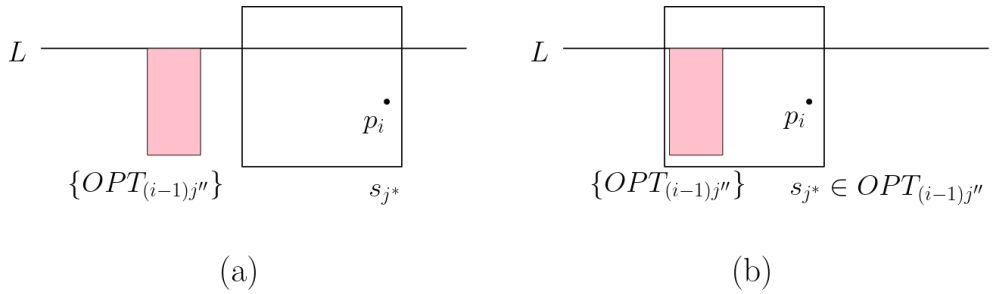
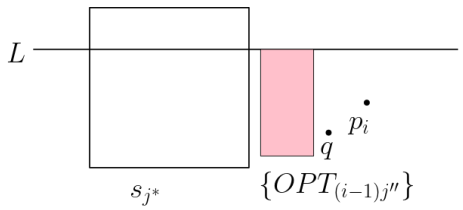


图 *3.* $OPT_{(i-1)j''}$ 覆盖点 p_i 时的两种情况。(a) 情况1，(b) 情况 2。粉色区域 $\{OPT_{(i-1)j''}\}$ 表示 $OPT_{(i-1)j''}$ 的重叠区域。

首先考虑**情况1**。考虑一个正方形 $s_{j''} \in OPT_{(i-1)j''}$，它覆盖 $OPT_{(i-1)j''}$ 中的点 p_{i-1}。根据归纳假设，

$$ply(Sol_{i-1}) \leq ply(OPT_{(i-1)j''})$$

正方形 s_{j^*} 不可能完全位于 $OPT_{(i-1)j''}$ 最右侧重叠区域的左边界左侧。假设它确实如此。$OPT_{(i-1)j''}$ 的重叠区域的左边界由参与重叠的最右侧正方形决定，设为 s_r。由于 $OPT_{(i-1)j''}$ 中没有正方形是冗余的，因此 s_r 必须独自覆盖某个点，设为 $q \in P_{i-1}$。根据我们的假设，s_{j^*} 位于 s_r 的左边界左侧，而 p_i 位于 q 右侧，因此 s_{j^*} 无法覆盖 p_i。由此便产生矛盾。示意图见图（4）。



F图 4. $s_{j^*} \ni p_i$ 完全位于 $OPT_{(i-1)j''}$ 的重叠区域左侧的正方形是不可能存在的。

因此，s_{j^*} 必定完全位于 $OPT_{(i-1)j''}$最右侧重叠区域的右边界右侧。根据我们的归纳假设（每次迭代均遵循贪心准则），$Sol_{i-1} \cup \{s_j\}$ 是 P_i 的可行解，其重叠度为 ply（Sol_{i-1}）。因此，

$$ply(Sol_i) = ply(Sol_{i-1}) \leq ply(OPT_{(i-1)j''}) = ply(OPT_{ij^*})$$

我们得出了一个矛盾。因此，**情况1** 不成立。

现在我们考虑**情况 2**，此时 p_i 已被 $OPT_{(i-1)j''}$覆盖。对于一个正方形 s，我们用 $x_L(s)$ $x_R(s)$分别表示 s 左、右边界的 x坐标。类似地，用 $y_T(s)$ $y_B(s)$分别表示 s上、下边界的 y坐标。我们假设任意两个正方形的左边界都不相同。更具体地说，输入正方形按从左到右的顺序排列为 $\prec$。若 s_1位于 s_2左侧，我们记作 $s_1 \prec s_2$，即 $x_L(s_1) < x_L(s_2)$。若 $i < j$，则 p_i 位于p_j左侧，即 $x(p_i) < x(p_j)$。

令 s_k 为在 $OPT_{(i-1)j''}$ 中覆盖 p_i 的正方形中最右侧的正方形。由于 $s_k \in OPT_{(i-1)j''}$，必定存在至少一个位于 p_i 左侧的点，该点仅由 $OPT_{(i-1)j''}$ 中的 s_k 覆盖。

令 $p_{i'}$ 为在 $OPT_{(i-1)j''}$ 中仅由 s_k 覆盖的点中最左侧的点，其中 $i' < i$。由于最左侧的 $(i'-1)$ 个点都不是 s_k 独占的，因此 $OPT_{i'} \setminus OPT_{i'-1} = \{s_k\}$。如果 s_k 对重叠度有贡献，即 s_k 与 $OPT_{i'-1}$ 的重叠区域相交，那么 $[OPT_{i'}] = [OPT_{i'-1}] + 1$；否则 $[OPT_{i'}] = [OPT_{i'-1}]$。

命题 2.3. 按照从左到右的顺序 $\prec$，正方形 s_k 是 $OPT_{(i-1)j''}$ 中最右侧的正方形。

证明。假设存在一个相对于 s_k 位于右侧的正方形 $s_o \in OPT_{(i-1)j''}$，即 $s_k \prec s_o$。s_o 的左边界必须位于 p_{i-1} 的左侧，否则 s_o 将变得冗余。如果 $y_T(s_o) > y_T(s_k)$，那么 s_o 在 $OPT_{(i-1)j''}$ 中将变得冗余，因为 s_k 覆盖了 s_o 的相关区域。另一种可能是 $y_T(s_o) < y_T(s_k)$。但在这种情况下，正方形 s_o 将包含 p_i。这违反了 s_k 的定义，即 s_k 是 $OPT_{(i-1)j''}$ 中包含 p_i 的正方形里最右侧的正方形。因此，这种情况不成立。所以，s_k 是$OPT_{(i-1)j''}$ 中最右侧的正方形。参见图 (5)。

□

命题 2.4。 $p_{i'}$ 与 p_i 之间的所有点都必须位于正方形 s_k内。

证明。设存在一个点 p_r，使得 $i' < r < i$ 且 $p_r \notin s_k$。本质上，p_r 位于 s_k的下边界之下。必定存在一个覆盖 p_r的正方形 $s \in OPT_{(i-1)j''}$。根据我们先前的断言，s_k 是 $OPT_{(i-1)j''}$中最右侧的正方形。因此，$s \prec s_k$。所以，s 也必须覆盖 $p_{i'}$，但这与 $p_{i'}$ 在 $OPT_{(i-1)j''}$中仅由 s_k 覆盖相矛盾。

□

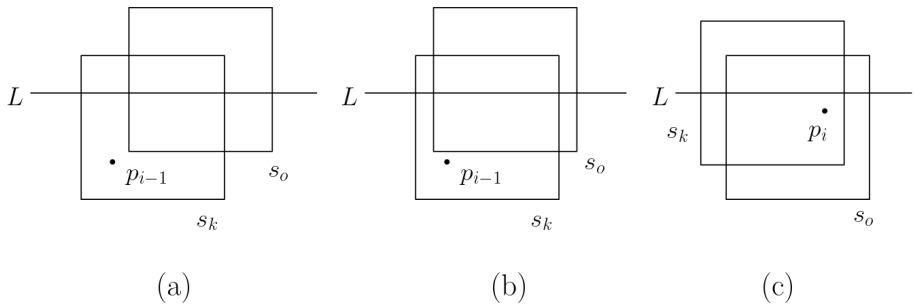


图*5.* s_k 和 s_o的相对位置。(a) s_o 位于 s_k 上方，但 p_{i-1} 位于 s_o的左边界左侧。因此， s_o 是冗余的。(b) s_o 位于 s_k 上方，但 s_o 位于 p_{i-1} 左侧的整个区域都被 s_k覆盖。因此， s_o 是冗余的。(c) s_o 位于 s_k下方，此处 s_o 也覆盖了 p_i ，这与 s_k的定义矛盾。因此， s_k 是 $OPT_{(i-1)j''}$ 中最右侧的正方形。

断言2.5。 条目 $T(i', k)$ 是 P_i 的一个可行集合覆盖，且与解 OPT_{ij^*}同样好。数学上，$ply(T(i', k)) = ply(OPT_{ij^*})$。

证明。 根据归纳假设， $ply(Sol_{i'}) \leq ply(OPT_{i'})$。由于 $[Sol_{i'}]$ 是覆盖 $P_{i'}$的最优重叠度，且根据我们函数 Compute Entry(·) 的定义，我们有

$$[Sol_{i'}] \leq [T(i', k)] \leq [Sol_{i'}] + 1 \tag{6}$$

因此，我们只需考虑两种可能的情况：i) $[T(i', k)] = [Sol_{i'}]$ ，以及 ii) $[T(i', k)] = [Sol_{i'}] + 1$。

i) $[T(i', k)] = [Sol_{i'}]$: 由于根据断言 (2.4)， $p_{i'}$与 p_i之间的所有点都位于 s_k中，因此 $T(i', k)$是 P_i的一个可行解。如果我们选择了 s_k，即为 $p_{i'}$ 选择 $T(i', k)$，那么问题就解决了。如果我们没有选择 s_k，那么设我们选择了不包含 p_i的$s_{k'}$。表中的条目 $T(i', k)$将一直保留到第（$i - 1$）行。最终，在某次第（i''）次迭代中，算法会选择 $T(i', k)$，因为它覆盖了 p_i 且不会增加重叠度，其中 $i' \leq i'' \leq i$。因此，在表的第（$i - 1$）行中，我们有一个可行解 $T(i', k)$，并且该解会一直保留到第 i行，而 $[T(i', k)] \leq [OPT_i]$。因此， $[Sol_i] > [OPT_{ij^*}]$ 被排除。

ii) $[T(i', k)] = [Sol_{i'}] + 1$: 由于 $[T(i', k)] \leq [Sol_{i'-1}] + 1$，我们有 $[Sol_{i'}] = [Sol_{i'-1}]$。换言之，选择 s_k 来覆盖 $p_{i'}$ 会导致当前解的重叠度值增加。因此， s_k 参与了 $T(i', k)$的重叠度。换言之， s_k 与 $Sol_{i'-1}$的重叠区域相交。根据归纳假设， $\{Sol_{i'-1}\}$ 相对位于 $\{OPT_{i'-1}\}$的左侧。由于 s_k 覆盖 $p_{i'}$ 和 p_i，因此 s_k 也与 $\{OPT_{i'-1}\}$相交。由于$OPT_{i'} = OPT_{i'-1} \cup \{s_k\}$，我们有 $[OPT_{i'}] = [OPT_{i'-1}] + 1$。

因此， $[T(i', k)] = [Sol_{i'-1}] + 1 \leq [OPT_{i'-1}] + 1 = [OPT_{i'}]$。如果我们选取了 s_k，即，对于 $p_{i'}$， $T(i', k)$，那么问题就解决了。如果我们没有选取 s_k，那么假设我们选取了不包含 p_i 的 $s_{k'}$。表项 $T(i', k)$ 将继续保留在表中，直到第 i 行。最终，在第 i 次迭代中，我们的算法会选取它，因为它不会导致重叠度增加。因此，我们的表中有了 $[T(i', k)] \leq [OPT_i]$。因此， $[Sol_i] > [OPT_{ij^*}]$ 被排除。

□

至此，循环不变式维护的证明完成。 □

关于可能性（2）的论证类似。现在考虑算法（1）的运行时间。对点进行排序需要 $O(n \log\ n)$ 时间。需要计算 T 中的 $n \cdot m$ 个表项。为了计算表项 $T(i,\ j)$，需要考虑第 $(i-1)$ 行中的所有可行表项（至多 m 个）。给定一组正方形 F，计算 $F \cup \{s_j\}$ 的重叠区域需要 $O(1)$ 时间。因此，总体运行时间为 $O(nm^2)$，其中 n 是点的数量，m 是正方形的数量。

定理 2.6. 算法（*1*）是求解***MPCSIHL1*** 问题（*2.1*）的多项式时间精确算法。

证明。 我们已经看到，算法（1）可在多项式时间内运行。根据断言（2.2），i行在T中包含 P_i的一个最优解。现在考虑与点 p_n对应的表 T 的第 n行。实现最小重叠度的表项就是我们的输入点集 $P = P_n$的最优解。 □

定理 2.7. 对于给定的、位于一条水平线与给定一组正方形的跨度相交区域内的点集，其最小重叠度覆盖可以在多项式时间内近似到2 倍以内。

证明。 我们运行算法（1）两次：一次针对位于相交水平线下方的点所对应的子问题，另一次针对位于相交水平线上方的点所对应的子问题。然后返回两个解的并集，记为 $Sol_a \cup Sol_b$。显然，我们的解覆盖了所有输入点，因此是一个可行集合覆盖。现在考虑平面上的任意一点。它最多被 $[Sol_a] + [Sol_b]$ 个正方形覆盖，其中 $ply(Sol_a)$（分别为 $ply(Sol_b)$）是针对水平线上方（分别为下方）点定义的问题的重叠度。由于 $ply(Sol_a) \leq ply(OPT)$且 $ply(Sol_b) \leq ply(OPT)$，因此我们的解是一个2-近似。

□

2.2. 点位于单位高度水平带内

我们考虑这样一种特殊情况：输入点位于高度为 1 的水平带内，且输入正方形与水平带的上边界或下边界相交。参见图（6）。对于这种情况，我们给出一种 $(9+\epsilon)$-因子近似算法，用于计算最小重叠度覆盖，其中 $\epsilon > 0$ 是一个较小的常数。该算法的表述和分析都很简单。我们正式定义这个问题。

定义 2.8. 考虑两条相距单位距离的水平线 L_1 和 L_2，其中 L_2 位于 L_1上方。现在考虑一组 $\mathcal{S}$ 单位正方形，其中每个正方形都与 L_1 或 L_2相交。设 P是一组位于 L_2下方但在 L_1上方的点，且每个点都位于 $\mathcal{S}$中的至少一个正方形内。目标是用 $\mathcal{S}$中的正方形计算 P的最小重叠度覆盖。

定理 2.9. 对于单位正方形，如果问题 *2.8* 存在一个 c-近似算法，那么最小重叠度覆盖问题就存在一个 $3c$- 近似算法。

证明。 我们将平面划分为高度为单位长度的水平条带。如果有 n 个输入点，那么至多有 n 个水平条带包含至少 1 个点。假设我们将这些条带从下到上记为H_1、 H_2、.....、 H_n。我们分别求解每个条带，并将各个解的并集作为最终输出。对于所有输入点，我们的解是一个可行解。考虑平面上的任意一点。该点位于某个水平条带内，设为 H_i。该点可能位于某个最大团中，即 H_i 的重叠区域。同时，该点也可能位于针对 H_{i-1} 和 H_{i+1} 的解的最大团中。令 Sol_i 表示我们的算法为条带 H_i 返回的解

。令 OPT_i 表示条带 H_i 的最优解。令 $Sol = \cup_i Sol_i$。显然，对于所有 i，$ply(OPT_i) \leq ply(OPT)$，其中 OPT 是整个输入的最小重叠度。因此

$$\begin{aligned} ply(Sol) &= \max_i (ply(Sol_{i-1}) + ply(Sol_i) + ply(Sol_{i+1}) \\ &\leq c \cdot (ply(OPT_{j-1}) + ply(OPT_j) + ply(OPT_{j+1})) \quad [j : \text{index at which the sum is max.}] \\ &\leq 3c \cdot ply(OPT) \end{aligned}$$

□

我们使用与上一节类似的贪心算法。这里的贪心平局打破规则略有修改，如下所示。

- 选择重叠度值最小的覆盖。
- 平局时，优先选择浮动重叠区域，而非固定重叠区域。
- 若仍然平局，则选择重叠区域右侧位置最靠左的覆盖。
- 若仍然平局，则选择重叠区域最窄的覆盖。
- 如果再次出现平局，则从并列的覆盖中任意选择一个。

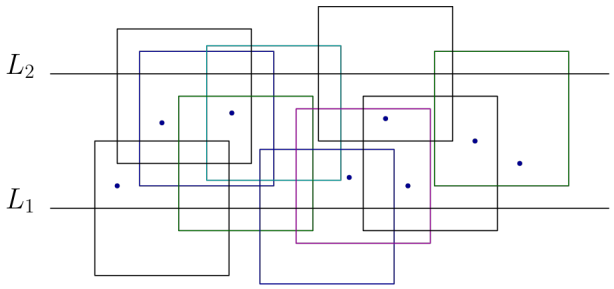


图 6。 单位高度水平带子问题。

对于 $1 \leq i \leq n$、$1 \leq j \leq m$，我们将表中的第（i，j）项记为 T（i，j）。我们将第 i 行中最小重叠度解的重叠度记为 ply_i。回顾一下，我们用 P_i 表示最左侧的 i 个输入点。在本节中，我们将证明以下引理。

引理 2.10。 贪心算法是问题（*2.8*）的 （$9 + \epsilon$）-近似算法。

断言 2.11。 对于任意 j，如果解 T（$i + 1$、j） 对 P_{i+1} 是可行的，则 T（$i + 1$、j） 的重叠度介于 ply_i 和 $ply_i + 1$ 之间。记作：

$$ply_i \leq ply(T(i + 1, j)) \leq ply_i + 1 \tag{7}$$

证明。 上界很容易证明。由于解 $T(i + 1,\ j)$ 可以由第 j 个正方形与第 i 行中的最小重叠度解的并集构成，因此 $T(i + 1,\ j)$ 的重叠度至多为一

多于 ply_{i}。
我们通过矛盾证明下界。假设对于某个 j， $T(i+1,\ j)$ 的重叠度严格小于 ply_{i}。因此，在我们的算法计算出的表的第 i 行中，s_j 不属于最优解。可能有两种情况。

情况1：处理点 p_{i+1}时， s_j 可以与第 i 行中的一个最小重叠度解合并，例如 $T(i,\ k)$，使得 $T(i,\ k) \cup \{s_j\}$ 是 P_{i+1}的一个可行解，并且 s_j 不与 $T(i,\ k)$ 的重叠区域相交，同时至少有一个参与构成 $T(i,\ k)$ 重叠区域的正方形被丢弃。因此 $ply(T(i+1,\ j)) < ply_{i}$。
情况 2：处理点 p_{i+1}时， s_j 可以与第 i 行中的一个最小重叠度解合并，例如 $T(i,\ k)$，使得 $T(i,\ k) \cup \{s_j\}$ 是 P_{i+1}的一个可行解，并且 s_j 与 $T(i,\ k)$ 的重叠区域相交，同时至少有两个参与构成 $T(i,\ k)$ 重叠区域的正方形被丢弃。因此 $ply(T(i+1,\ j)) < ply_{i}$。
任何解的重叠区域必定是其上一行父解的重叠区域的子集。因此，在上述两种情况下，如果 s_j 在第（$i+1$）行是更好的选择，我们的贪心算法就会更早选中它。因此，我们得出了矛盾。

□

如果一组正方形有公共交集，我们就称它们构成一个团。我们将该团的公共交集区域称为重叠区域。首先，我们将解中的团分成若干不同类型。设所考虑的团的大小为 l，即 l 个正方形有公共交集。如果一个正方形 s_1 的上边高于另一个正方形 s_2的上边，我们就称 s_1 位于 s_2上方。等价地，也可以说s_2 位于 s_1下方。一组正方形 $s_1, s_2, \ldots, s_k$，若对所有 $1 \leq i \leq k-1$， s_i 都位于 s_{i+1} 上方，则称为一组下降正方形。一组正方形 $s_1, s_2, \ldots, s_k$，若对所有 $1 \leq i \leq k-1$， s_i 都位于 s_{i+1} 下方，则称为一组升序 正方形。如果一个正方形 s_1 的左边位于另一个正方形s_2的左边左侧，我们就称 s_1 位于 s_2左侧。我们用 $s_1 \prec s_2$表示这一关系。以下是可能出现的团类型。

- **顶部锚定**：在此，团中的所有组成正方形都与顶线相交 L_2。如图（7）所示，共有三种子类型。

  – **顶部锚定 ASC**：正方形从左至右升序排列。换句话说，对于团中任意两个正方形 s_1, s_2，只要满足 $s_1 \prec s_2$，正方形 s_2 就在 s_1上方。– **顶部锚定 DESC**：正方形从左至右降序排列。换句话说，对于团中任意两个正方形 s_1, s_2，只要满足 $s_1 \prec s_2$，就有 s_2 在 s_1下方。– **顶部锚定 DESC|ASC**：存在一个 $k \geq 2$，使得构成该团的正方形起初从 s_1 到 s_k依次下降。随后，正方形 s_{k+1} 位于 s_k上方。接着，正方形 s_{k+1} 到 s_l 依次升序排列。我们将正方形 s_{k+1} 称为过渡正方形。这种类型的团可以看作一个下降团与一个上升团按此顺序合并而成。

命题 2.12。 禁止出现类型为升序|升序的顶部锚定团。

证明。 假设结论不成立。假设，左侧上升序列由 k 个正方形组成。那么第一 ASC 序列的最右侧正方形 s_k 将变得冗余，因为第一 ASC 序列的倒数第二个正方形 s_{k-1} 和过渡正方形 s_{k+1} 将完全覆盖 s_k 的相关区域。这是矛盾的，因为 s_k 是冗余的。示例请参见图（8）。

□

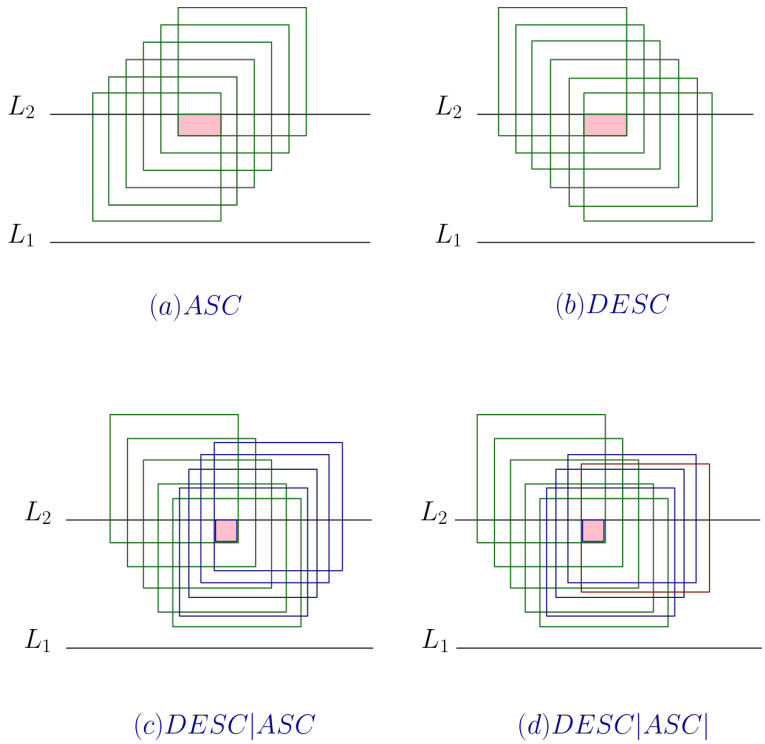


图 7。(a)、(b) 和 (c) 展示了三种不同类型的顶部锚定团。(d) 展示了一个无效团，其中一个 DESC|ASC 顶部锚定团后接着一个过渡正方形。

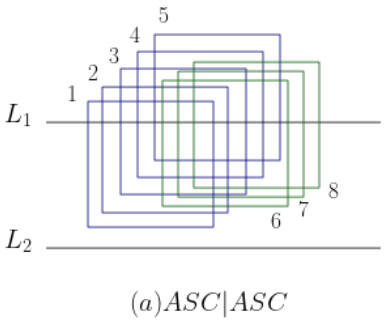


图 *8*。 一个类型为顶部锚定 ASC|ASC 的禁用团。这里，正方形 5 是冗余的，因为 4 和 6 完全覆盖了 5 的相关区域。

命题 2.13。 类型为 *DESC|DESC* 的顶部锚定团是禁用的。

证明。 假设结论不成立。假设，左侧下降序列由 k 个正方形组成。那么第二 DESC 序列的最左侧正方形 s_{k+1}将变得冗余，因为第一 DESC 序列的最后一个正方形 s_k 和正方形 s_{k+2} 将完全覆盖 s_{k+1}的相关区域。这是矛盾的，因为 s_{k+1} 是冗余的。示例请参见图 9(a)。

□

命题 2.14。 类型为 *ASC|DESC* 的顶部锚定团是禁用的。

证明。 证明过程与命题 (2.12) 的证明类似。示例请参见图 9(b)。 □

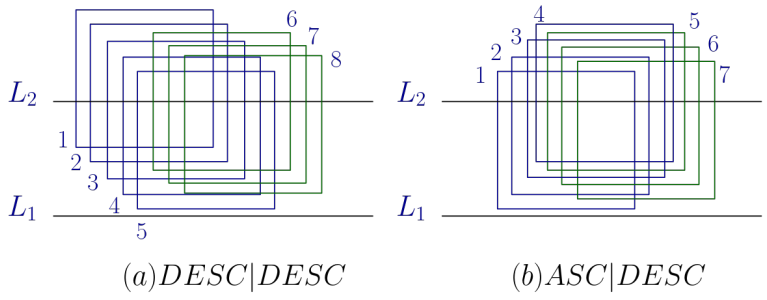


图*10*。(a) 一种顶部锚定降序|降序类型的禁用团。这里，正方形 6 是冗余的，因为正方形 5 和 7 完全覆盖了正方形 6 的相关区域。(b) 一种顶部锚定升序|降序类型的禁用团。这里，正方形 4 是冗余的，因为正方形 3 和 5 完全覆盖了正方形 4 的相关区域。

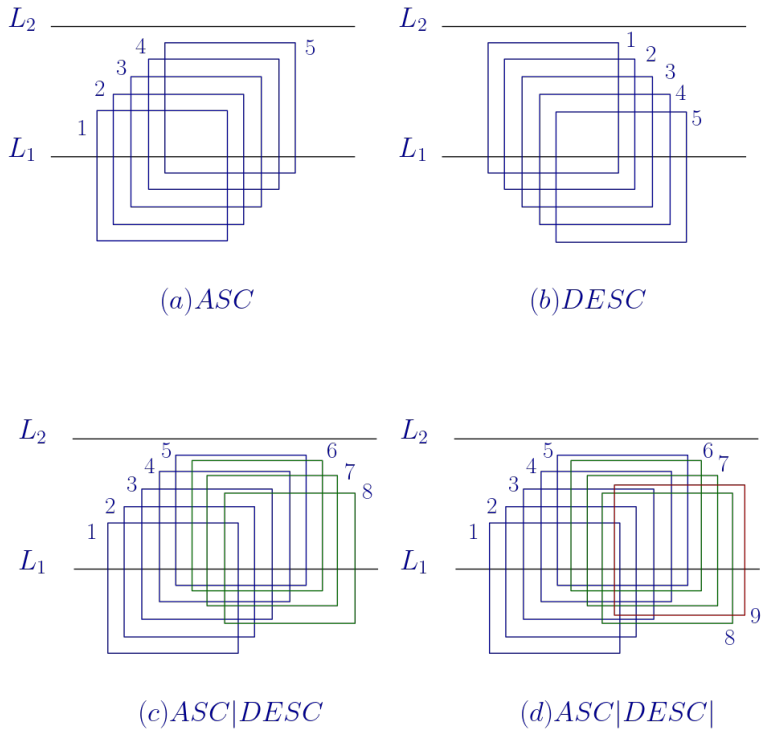


图*10*。(a)、(b) 和 (c) 展示了三种不同类型的底部锚定团。(d) 展示了一个无效团，其中一个升序|降序底部锚定团后接着一个过渡正方形。

- **底部锚定**：这里，团中的所有组成正方形都与底线相交 L_1。如图（10）所示，共有三种子类型。

  – **底部锚定 ASC**：这里的正方形从左到右依次升序排列。换言之，对于团中任意两个正方形 s_1，s_2，若满足 $s_1 \prec s_2$，则正方形 s_2 位于 s_1上方。– **底部锚定 DESC**：这里的正方形从左到右依次下降。换言之，对于团中任意两个正方形 s_1, s_2，若满足 $s_1 \prec s_2$，则正方形 s_2 位于 s_1下方。– **底部锚定 ASC|DESC**：存在一个 $k \geq 2$，使得构成该团的正方形起初从 s_1 到 s_k 依次升序排列。然后，正方形 s_{k+1} 位于 s_k下方。接着，正方形 s_{k+1} 到 s_l 依次下降。

命题 2.15. 类型为 *ASC|ASC*、*DESC|ASC* 或 *DESC|DESC* 的底部锚定团是禁用的。

证明。 在这三种情况下，都会有一些正方形变得冗余，从而导致矛盾。证明与命题 (2.12) 的证明类似。示例见图（11）和图（12）。 □

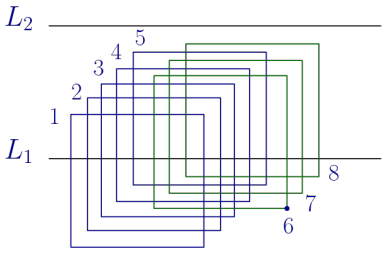
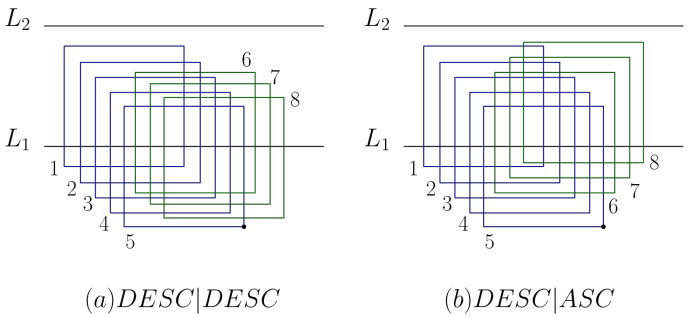


图 *11.* 一个类型为底部锚定 ASC|ASC 的禁用团。这里，正方形 6 是冗余的，因为正方形 5 和 7 完全覆盖了正方形 6 的相关区域。



$(a) DESC|DESC$ $(b) DESC|ASC$

图 *12.* (a) 一个类型为底部锚定 DESC|DESC 的禁用团。这里，正方形 5 是冗余的，因为正方形 4 和 6 完全覆盖了正方形 5 的相关区域。(b) 一个类型为底部锚定 DESC|ASC 的禁用团。这里，正方形 5 是冗余的，因为正方形 4 和 6 完全覆盖了正方形 5 的相关区域。

• **浮动**：如果一组正方形有公共交集，且其中一些正方形与顶线相交L_2，另一些与底线相交 L_1，我们就称该公共交集为浮动 团。在下面的子类型中，至少有一个正方形与底线相交 L_1，且至少有一个正方形与顶线相交 L_2。

  – **浮动上升**：此处，最左侧正方形 s_1 与底线 L_1 相交。从左到右的正方形呈升序。换言之，对于团中任意两个正方形 s_1, s_2，若满足 $s_1 \prec s_2$，则正方形 s_2 位于 s_1 上方。最右侧正方形 s_l 必须与顶线 L_2相交。请参阅图（13(a)）。– **浮动下降**：此处，最左侧正方形 s_1 与顶线 L_2 相交。从左到右的正方形呈下降。换言之，对于团中任意两个正方形 s_1, s_2，若满足 $s_1 \prec s_2$，则正方形 s_2 位于 s_1 下方。最右侧正方形 s_l 必须与底线 L_1相交。请参阅图（13(b)）。– **浮动上升|升序**：存在一个 $k \geq 2$，使得构成该团的正方形从 s_1 到 s_k起初呈升序。随后，正方形 s_{k+1} 位于 s_k 下方。然后，正方形 s_{k+1} 到 s_l 再次呈升序。请参阅图（14）。此类团可以视为两个单调上升团的合并，其中第一个团由正方形 s_1 至 s_k 组成，第二个团

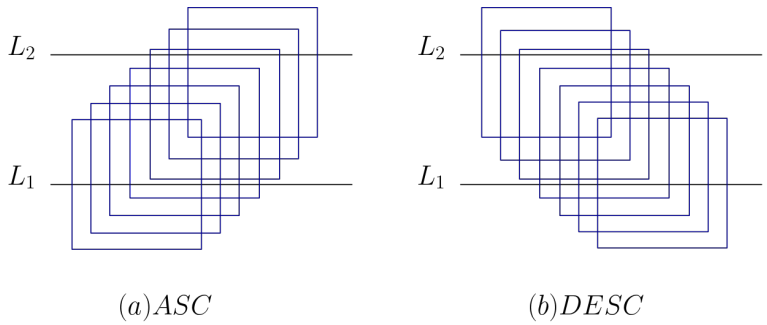


图*13.* 单调浮动团的类型。

由正方形 s_{k+1} 至 s_l组成。这类团的结构遵循以下断言所规定的若干规则。

断言 2.16. 在类型为 *ASC|ASC* 的浮动团中，第一条上升序列中至多有一个正方形与顶线 L_2 相交，第二条上升序列中至多有一个正方形与底线 L_1 相交。

证明. 反设不然。第一条上升序列中至少有两个正方形与 L_2 相交。那么，第一条上升序列中最右侧的两个正方形 s_{k-1} 和 s_k 肯定与 L_2 相交。根据定义， s_{k+1} 位于 s_k 下方。如果 s_{k+1} 与顶线 L_2 相交，那么 s_k 是冗余的，因为 s_{k-1} 和 s_{k+1} 覆盖了 s_k 的相关区域。参见图14(a)。如果 s_{k+1} 与底线 L_1 相交，则分两种情况。情况1： s_{k+2} 也与底线 L_1 相交，那么 s_{k+1} 是冗余的，因为 s_1， s_k 和 s_{k+2} 覆盖了 s_{k+1} 的相关区域。参见图14(b)。情况 2： s_{k+2} 与顶线 L_2 相交，那么 s_k 是冗余的，因为 s_{k-1}， s_{k+1} 和 s_{k+2} 覆盖了 s_k 的相关区域。参见图14(c)。
现在考虑断言的第二部分。假设与 L_1 相交的第二个上升序列中至少有两个正方形。那么，其中最左侧的两个正方形 s_{k+1} 和 s_{k+2} 肯定与L_1相交。正方形 s_{k+1} 是冗余的，因为 s_1， s_k 和 s_{k+2} 覆盖了 s_{k+1}的相关区域。参见图14(d)。

□

– 浮动升序|下降：存在一个 $k \geq 2$ ，构成该团的正方形最初从 s_1 到 s_k按升序排列。随后，正方形 s_{k+1} 位于 s_k下方。接着，正方形 s_{k+1} 到 s_l 按降序排列。参见图（15）。这种类型的团可以看作是一个单调上升团后接一个单调下降团的合并，其中第一个团由正方形 s_1 到 s_k 组成，第二个团由正方形 s_{k+1} 到 s_l组成。这类团的结构遵循以下断言所规定的某些规则。

断言 2.17. 在一个类型为升序|降序的团中，至多有两个正方形与顶线L_2相交。

证明。 反设不然。至少有三个正方形与 L_2 相交。正方形 s_k 是该团中最高的正方形，因此 s_k 肯定与 L_2相交。共有三种情况。
情况1： s_{k-2}， s_{k-1}， s_k 与 L_2相交。正方形 s_{k-1} 是冗余的，因为 s_{k-2}， s_k 和 s_{k+1} 覆盖了 s_{k-1}的相关区域。参见图15(a)。

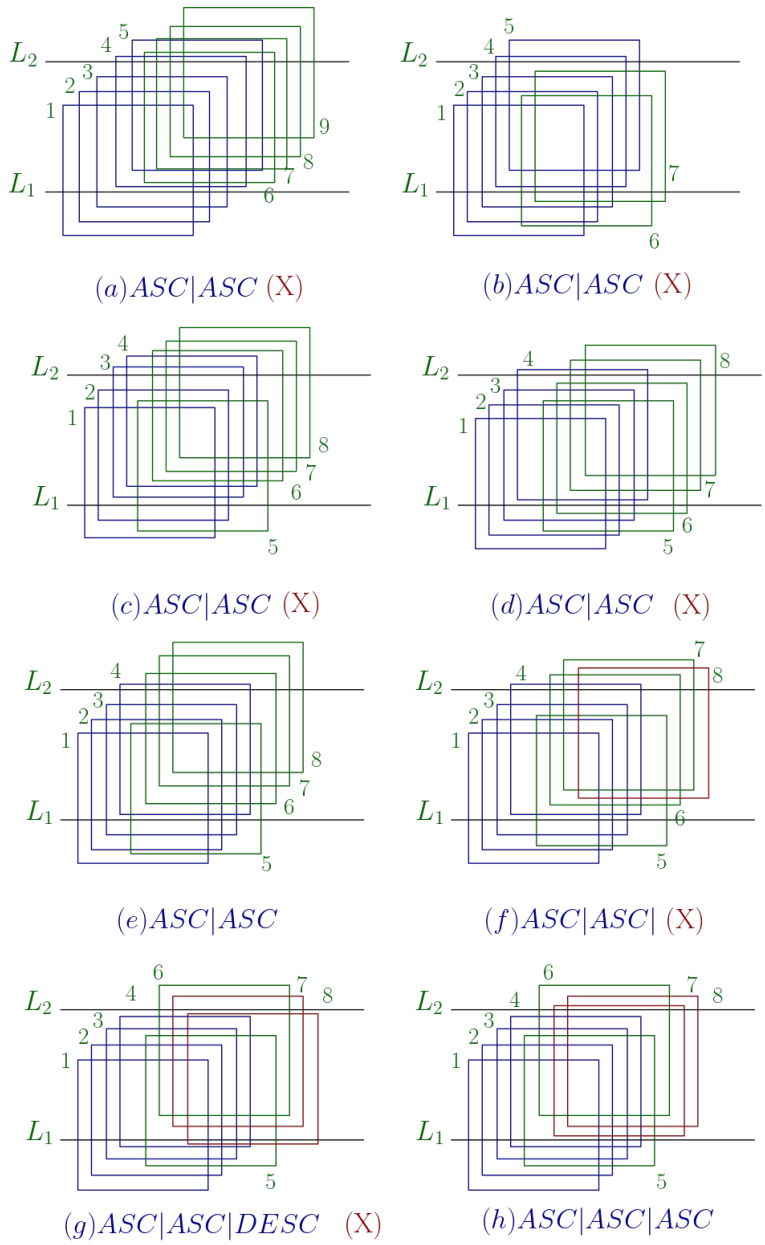


图*14.* 浮动上升|升序团。在 (a)、(b) 和 (c) 中，第一个升序序列有多个正方形与 L_2 相交。(a) 展示了一个升序|升序类型的无效团，其中第一个升序序列的最右侧正方形 5 被 4 和 6 覆盖。第二个升序序列有多个正方形与 L_1 相交。(b) 展示了一个升序|升序类型的无效团，其中过渡正方形 6 被 1、5 和 7 覆盖。(c) 展示了一个升序|升序类型的无效团，其中第一个升序序列的最右侧正方形 4 被 3、5 和 6 覆盖。(d) 展示了一个无效团，其中升序|升序的第二个上升序列有多个正方形与 L_1 相交。这里，5 的相关区域被 1、4 和 6 覆盖。(e) 展示了一个升序|升序类型的有效团。(h) 展示了一个后接过渡正方形的升序|升序类型的无效团。(g) 展示了一个升序|升序|降序类型的无效团。(h) 展示了一个升序|升序|升序类型的有效团。

情况 2： s_{k-1}， s_k， s_{k+1} 与 L_2相交。正方形 s_k 是冗余的，因为 s_{k-1} 和 s_{k+1} 覆盖了 s_k的相关区域。参见图15(b)。

情况3： s_k, s_{k+1}, s_{k+2} 相交于 L_2。正方形 s_{k+1} 是冗余的，因为 s_1, s_k 和 s_{k+2} 覆盖了 s_{k+1}的相关区域。参见图15(c)。

因此，我们在每种情况下都得出了矛盾。证毕。 □

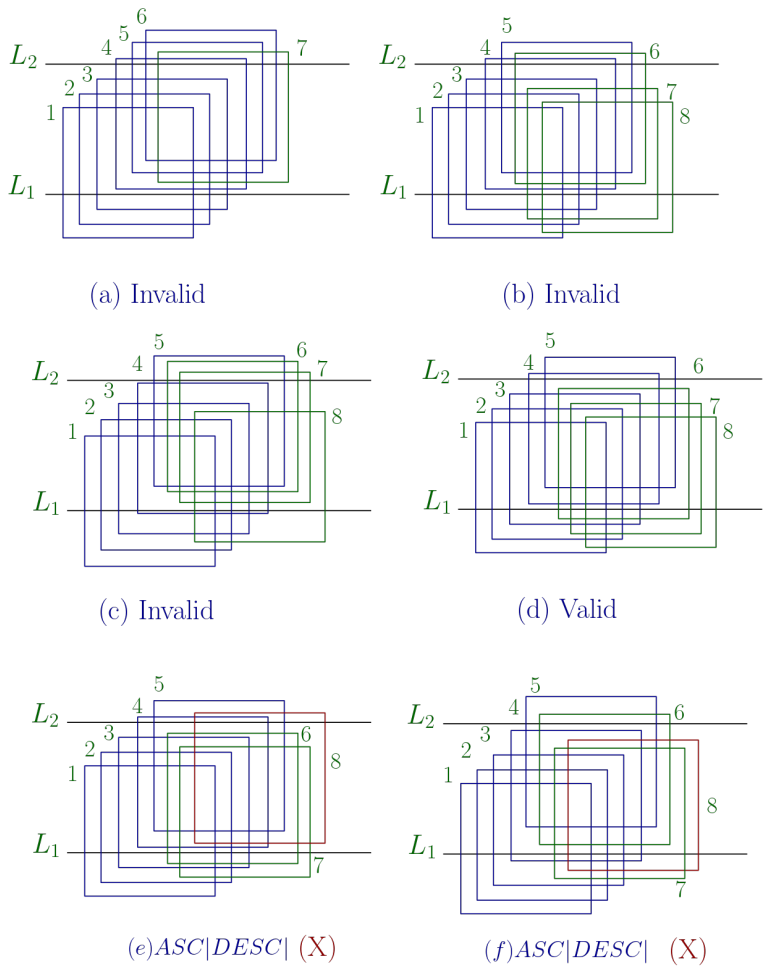


图*15.* (a)、(b) 和 (c) 展示了无效团，其中有超过 2 个正方形与顶线相交。(a) 中，5 的相关区域由 4、6 和 7 覆盖。(b) 中，5 的相关区域由 4 和 6 覆盖。(c) 中，6 的相关区域由 1、5 和 7 覆盖。(d) 是一个有效的浮动上升|降序团，其中有 2 个正方形与顶线相交。(e) 展示了一个无效团，其中一个上升|降序团后接一个过渡正方形。过渡正方形 8 与 L_2 相交。这里，6 的相关区域由 1、5、7 和 8 覆盖。(f) 展示了一个无效团，其中一个上升|降序团后接一个过渡正方形。过渡正方形 8 与 L_1 相交。这里，7 的相关区域由 1、6 和 8 覆盖。

– **浮动下降|升序**：存在一个 $k \geq 2$，使得构成该团的正方形最初从 s_1 到 s_k呈下降排列。然后，正方形 s_{k+1} 位于 s_k上方。接着，正方形 s_{k+1} 到 s_l 呈升序排列。参见图16。此类团可以视为一个单调下降团与另一个单调上升团的合并，其中第一个团由正方形 s_1 至 s_k 组成，第二个团由正方形 s_{k+1} 至 s_l组成。此类团的结构遵循某些规则，具体如下述断言所述。

断言2.18. 在类型为降序|升序的浮动团中，团内至多有两个正方形与底线 L_1相交。

证明. 假设不然。至少有三个正方形与 L_1相交。正方形 s_k 是团中最靠下的正方形，因此 s_k 必定与 L_1相交。有三种情况。
情况1： s_{k-2}， s_{k-1}， s_k 与 L_1相交。 s_{k+1} 不能与 L_1 相交，否则 s_k 将成为冗余。

因此， s_{k+1} 必须与 L_2相交。现在，正方形 s_{k-1} 是冗余的，因为 s_{k-2}， s_k 和 s_{k+1} 覆盖了 s_{k-1}的相关区域。参见图16(a)。情况 2： s_{k-1}， s_k， s_{k+1} 与 L_1相交。正方形 s_k 是冗余的，因为 s_{k-1} 和 s_{k+1} 覆盖了 s_k的相关区域。参见图16(b)。情况3： s_k， s_{k+1}， s_{k+2}与 L_1相交。正方形 s_{k+1}是冗余的，因为 s_1， s_k和 s_{k+2}覆盖了 s_{k+1}的相关区域。参见图16(c)。因此，在每种情况下我们都得出了矛盾。证毕。

□

– **浮动下降|降序**：存在一个 $k \geq 2$ ，使得构成该团的正方形最初从 s_1 到 s_k呈下降排列。然后，正方形 s_{k+1} 位于 s_k上方。接着，正方形 s_{k+1} 到 s_l 再次呈下降排列。参见图（17）。这种类型的团可以看作由两个单调下降团合并而成，其中第一个团由正方形 s_1 至 s_k 组成，第二个团由正方形 s_{k+1} 至 s_l组成。这种团的结构遵循以下断言中所述的特定规则。

命题2.19. 在类型为浮动团*DESC*|降序的团中，第一个下降序列中至多有一个正方形与底线 L_1 相交，第二个下降序列中至多有一个正方形与顶线 L_2相交。

证明. 反设不然。第一个下降序列中至少有两个正方形与 L_1相交。那么，第一个升序序列中最右侧的两个正方形 s_{k-1} 和 s_k 必定与 L_1相交。根据定义， s_{k+1} 位于 s_k上方。由于 s_{k-1}， s_k 都与 L_1相交，因此 s_1 必定与 L_2相交，否则 s_{k-1} 将变得冗余。如果 s_{k+1} 与底线 L_1相交，那么 s_k 就是冗余的，因为s_{k-1} 和 s_{k+1} 覆盖了 s_k的相关区域。参见图17(a)。另一方面，如果 s_{k+1} 与顶线 L_2相交，则有两种情况。

情况1： s_{k+2} 也与顶线 L_2相交。那么 s_{k+1} 是冗余的，因为 s_1， s_k 和 s_{k+2} 覆盖了 s_{k+1}的相关区域。参见图17(b)。
情况 2： s_{k+2} 与底线 L_1相交，那么 s_k 是冗余的，因为 s_{k-1}， s_{k+1} 和 s_{k+2} 覆盖了 s_k的相关区域。参见图17(c)。
现在考虑断言的第二部分。假设第二下降序列中至少有两个正方形与 L_2相交。那么其中最左侧的两个正方形 s_{k+1} 和 s_{k+2} 必定与L_2相交。正方形 s_{k+1} 是冗余的，因为 s_1， s_k 和 s_{k+2} 覆盖了 s_{k+1}的相关区域。参见图17(d)。

因此，在每种情况下我们都得出了矛盾。证毕。 □

引理 2.20。 如果三个正方形单调序列 S_1、 S_2、 S_3分别从左到右合并形成一个团，那么 S_2 至多由2 个正方形组成。

证明。正方形序列 S_2 要么按升序排列，要么按降序排列。我们先考虑升序情形。假设存在矛盾，即序列 S_2中至少有 3 个正方形。我们将最左边的三个正方形

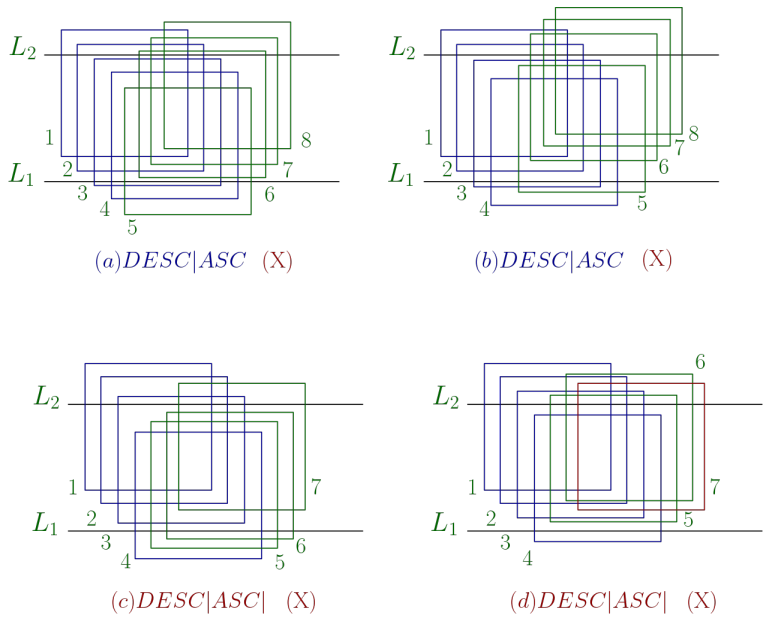


图*16.* (a)、(b) 和 (c) 展示了无效的降序|升序团，其中有超过 2 个正方形与底线相交。(a) 中，4 的相关区域由 3、5 和 6 覆盖。(b) 中，4 的相关区域由 3 和 5 覆盖。(c) 中，5 的相关区域由 1、 4 和 6 覆盖。(d) 展示了一个无效团，其中一个降序|升序团后接一个过渡正方形。过渡正方形 7 会使正方形 5 或正方形 6 变得冗余。

从左到右记为 s_{k+1}， s_{k+2} 和 s_{k+3}。根据断言 (2.15)、(2.16) 和 (2.18)，我们知道 S_2 不可能是底部锚定团。以下是几种可能情况，

i) S_2 是顶部锚定升序：假设 $|S_1| = k, |S_1| + |S_2| = l$。如果 ($S_2$， S_3) 的过渡正方形，即 s_{l+1}，是顶部相交的，那么 s_l 将由 s_{l-1} 和 s_{l+1}覆盖。因此， s_l 将变得冗余。矛盾。如果 s_{l+1} 是底部相交的，那么 s_{l-1} 的区域将由 s_{k+1}， s_l 和 s_{l+1}覆盖。因此， s_{l-1} 将变得冗余。矛盾。

ii) S_2 是浮动上升：如果 S_1 是底部锚定DESC，则 S_1的最右侧正方形，即 s_k 将变得冗余，因为其相关区域将被 s_{k-1} 和 s_{k+1}覆盖。因此， S_1 要么是浮动，要么是底部锚定 ASC。现在可能出现以下情况。

(a) 如果 s_{k+2}、 s_{k+3} 以及（S_2， S_3）的过渡正方形，即 s_{l+1} 均为顶部相交，则 s_{k+3}的相关区域将被 s_{k+2} 和 s_{l+1}覆盖。因此， s_{k+3} 将变得冗余。
(b) 如果 s_{k+2}， s_{k+3} 为顶部相交，但（S_2， S_3）的过渡正方形，即 s_{l+1} 为底部相交，则存在两种情况。情况 1：如果（S_1， S_2）的过渡正方形，即 s_{k+1} 为底部相交，则 s_{k+1} 的相关区域将完全被 s_1、 s_k，s_{k+2} 和 s_{l+1}覆盖。因此， s_1 将变得冗余。情况 2：如果 S_1， S_2之间的过渡正方形，即 s_{k+1} 为顶部相交，则 s_{k+2} 的区域将完全被s_{k+1}、 s_{k+3} 和 s_{l+1}覆盖。因此， s_{k+2} 将变得冗余。

（c）如果 s_{k+1} 和 s_{k+2} 是底部相交的， s_{k+3} 是顶部相交的，并且 (S_2， S_3) 的过渡正方形，即 s_{l+1}) 是顶部相交的，那么有两种情况。
情况1： S_1的最右侧正方形，即 s_k ，是底部相交的，因此 s_{k+1} 的相关区域将被 s_k 和 s_{k+2}完全覆盖。因此， s_{k+1} 将变得冗余。

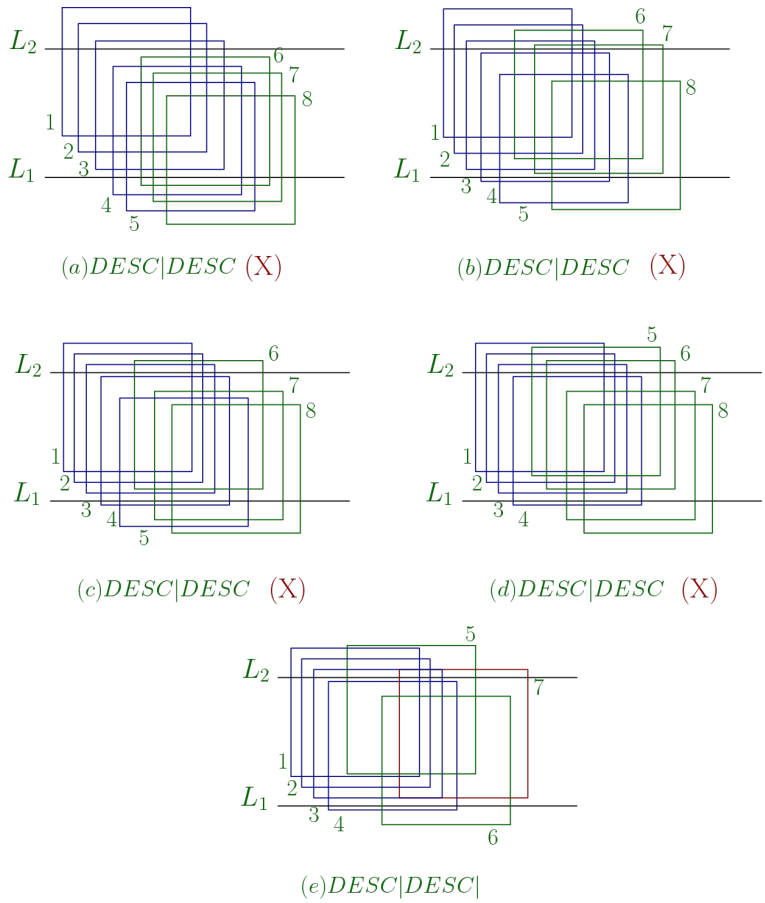


图*17.* (a)、(b) 和 (c) 展示了无效的 DESC|DESC 团，其中第一条 DESC 序列中的多个正方形与底线 L_1相交。(a) 中，5 的相关区域由 4 和 6 覆盖。(b) 中，6 的相关区域由 1、5 和 7 覆盖。(c) 中，5 的相关区域由 4、6 和 7 覆盖。(d) 展示了一个无效的 DESC|DESC 团，其中第二条 DESC 序列中的多个正方形与顶线 L_2相交。此处，5 的相关区域由 1、4 和 6 覆盖。(d) 展示了一个无效团，其中 DESC|DESC 团后接一个过渡正方形。过渡正方形 7 会使正方形 5 或正方形 6 成为冗余。

情况 2： S_1的最右侧正方形，即 s_k，是底部相交的，因此 s_{k+3} 的相关区域将被 s_k， s_{k+2} 和 s_{l+1}完全覆盖。因此， s_{k+3} 将变得冗余。

(d) 如果 s_{k+1} 和 s_{k+2} 是底部相交的， s_{k+3} 是顶部相交的，并且 (S_2、 S_3) 的过渡正方形（即 l_{l+1} ）是底部相交的，那么存在两种情况。

情况1： S_1、 S_2之间的过渡正方形，即 s_{k+1} ，与底部相交，则 s_{k+2} 的区域将完全被 s_{k+1} 和 s_{k+3}覆盖。因此， s_{k+2} 将变得冗余。

情况 2： S_1、 S_2之间的过渡正方形，即 s_{k+1} ，与顶部相交，则 s_{k+3} 的区域将完全被 s_{k+1}、 s_{k+2}、 s_{k+4}和 s_{k+5}覆盖。因此， s_{k+3} 将变得冗余。

我们已经证明，当 S_2 是一个上升类型的序列时，每种可能情况都会导致矛盾。当 S_2 是一个下降正方形序列时，也可以采用类似的论证。 □

断言 2.21。 我们的算法会不可撤销地选择一个正方形 s，其位置位于 s 的最右侧专属点处或其左侧。

证明。 考虑一个正方形 $s \in Sol$。正方形 s 必定是在处理某个点 $p \in s$ 时被纳入 Sol 的。令 p_r 为 s 在 Sol 中的最右侧专属点。假设在截至点 p_r 的所有点所得到的部分解中不存在 s。那么， s 必定是在 p_r 右侧的某个点 p_q 处被纳入的。处理点 p_q 时，算法必定丢弃了某些正方形，使得 p_r 及 $Excl$（s）中的其他点可以成为 s 的独占点。由此得到的解是 P_q 的一个可行解。这意味着，如果 s 是针对 p_q 的更好选择，我们的贪心算法早就会选它。因此，我们得出了矛盾。

□

命题 2.22。 单独考虑一个团时，团中所有非极端正方形的专属区域都是矩形或 L-形。至多有两个非极端正方形可能具有两个不同的连通专属区域。

证明。 考虑任意一个非极端正方形 s_i。正方形 s 的左侧有一个正方形 s_{i-1}，右侧有一个正方形 s_{i+1}。共有 4 种情况。
i) 三个正方形均按升序排列。此时， s_i 的专属区域必须位于其左上角和/或右下角附近。

ii) 三个正方形均按降序排列。此时， s_i 的专属区域必须位于其右上角和/或左下角附近。

iii) s_i 位于 s_{i-1} 和 s_{i+1}的下方：此时， s_i 只有一个专属区域，该区域为矩形或 L-形。iv) s_i 位于 s_{i-1} 和 s_{i+1}的上方：此时， s_i 只有一个专属区域，该区域为矩形或 L-形。由于水平带的高度为 1，因此最多有两个正方形具有两个不同的连通专属区域。具体而言，在单调降序团中，是与 L_2 相交的最右侧正方形和与 L_1相交的最左侧正方形；在单调升序团中，则是与 L_1 相交的最右侧正方形和与 L_2 相交的最左侧正方形。

□

由于我们解中的所有正方形都是必要的，因此对于每个正方形 $s \in Sol$，都存在一组点 $Excl(s)$，使得 $Excl(s)$ 中的点仅包含在 s 中，而不包含在 Sol 中的其他任何正方形内。 $Excl(s)$ 中的这些点称为 s 的专属点。我们对专属点提出以下关键断言。

命题 2.23。 设 s_1, s_2 是 Sol 的一个最大团中的相邻正方形，且 $s_1 \prec s_2$，并且 s_1 不是 Sol 中最左侧的正方形。没有任何输入正方形 s 能包含 $Excl$ 中的所有点$(s_1) \cup Excl(s_2)$。

证明。 我们已经证明，解中不含冗余正方形的任何团只能属于少数几种类型。相邻正方形 s_1 和 s_2 有 6 种可能情况。下面逐一分析。对于以下每种情况，为了导出矛盾，假设存在一个正方形 s，使得 s 覆盖了 $Excl(s_1) \cup Excl(s_2)$ 中的所有点。

1. 顶部锚定 DESC：此处 s_1 和 s_2 与顶线 L_2 和 $s_1 \prec s_2$ 相交，且 s_1 位于 s_2上方。同样有两种子情况。

   (a) 如果 s 与底线 L_1相交：那么在 s_2的最左侧专属点 p 处，我们的算法必须在 s 和 s_2之间做出选择。由于根据断言 2.21，已经选取了 s_1，因此我们的算法将

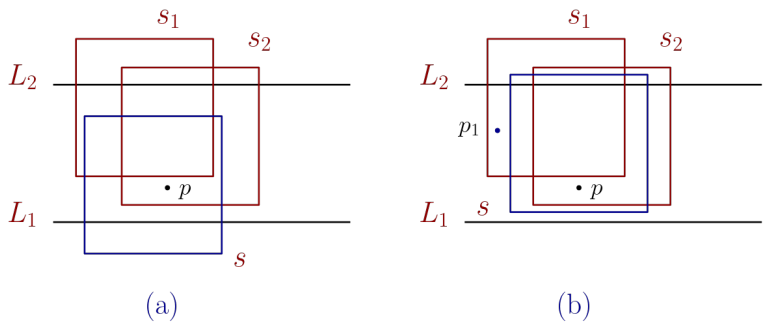


图*18*。正方形 s_1 和 s_2 是顶部锚定的，并且在所考虑的团中按降序排列。

优先选择 s 而非 s_2，因为选择 s_2 会得到一个浮动团。而我们的算法之后不会再选择 s_2。请记住，我们的贪心算法优先选择浮动团，而非锚定团。图 18(a) 展示了一个示例。

(b) 如果 s 与顶线 L_2相交：那么在 s_2的最左侧专属点 p 处，我们的算法会选择 s 而不是 s_2，因为选择 s 会使 s_1 变得冗余。由于 s_1 的独占区域是矩形，且正方形 s 覆盖了 $Excl(s_1)$中的所有点，因此 s 位于 s_2的左侧，即 $s \prec s_2$。因此， s 覆盖了 $s_1 \cap s_2$ 中位于 p左侧的每个点。考虑一个点 $p_1 \in s_1 \setminus s_2$，它被另一个正方形 $s_0 \in Sol$ 覆盖，但推测并未被 s覆盖。显然， s_0 的所有专属点都位于 p的左侧。根据断言 2.21， s_0 必定已被我们的解选中。考虑一个点 $p_2 \in s_2 \setminus s_1$，它被另一个正方形 $s_0 \in Sol$ 覆盖，但推测并未被 s覆盖。如果$s_2 \prec s_0$，那么 s_0 中的所有点都位于 p的右侧。因此，在此阶段我们无需担心覆盖 p_2。否则，如果 $s_0 \prec s_1$，那么 $s_0 \cap s_2$ 将包含在 $s_1 \cap s_2$ 中，而这样的点 p_2 不可能存在。因此，在处理 p 时选择 s 不会增加活动重数，我们的算法将贪心地选择 s。图18(b)展示了示意图。

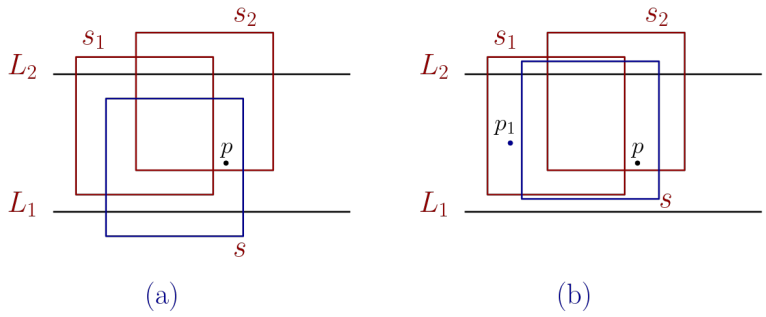


图*19.* 在所考虑的团中，正方形 s_1 和 s_2 顶部锚定，且按升序排列。

2. 顶部锚定升序：这里， s_1 和 s_2 与顶线 L_2 相交，且 $s_1 \prec s_2$ 和 s_1 位于 s_2上方。同样有两种子情况。

(a) 如果 s 与底线 L_1相交：那么在 s_2的最左侧专属点 p 处，我们的算法本应选择 s 而不是 s_2。原因与上述情况1(a)中的论证完全相同。(b) 如果 s 与顶线 L_2相交：在处理 s_2的最左侧专属点 p 时，我们的算法已经

选择了 s_1，因为 s_1 的最右侧专属点必定位于 p的左侧。在 p 处，我们的算法本应选择 s 而不是 s_2，因为选择 s 也会使 s_1 变得冗余。有两种可能性。第一，如果 $s_2 \prec s$，那么我们的算法会优先选择 s 而不是 s_2，因为这样可以得到一个大小相同但更窄的团。第二，如果 $s_1 \prec s \prec s_2$，那么 s 覆盖 $s_1 \cap s_2$中的每个点。考虑一个点 $p_1 \in s_1 \setminus s_2$，它被另一个正方形 $s_0 \in Sol$ 覆盖，但推测未被 s覆盖。如果 $s_0 \prec s_1$，那么当我们的算法处理 p时， s_0 已经被选中。如果 $s_2 \prec s_0$，那么这样的 p 不可能存在。参见图19。考虑一个点 $p_2 \in s_2 \setminus s_1$，它被另一个正方形 $s_0 \in Sol$覆盖，但推测未被 s覆盖。如果 $s_2 \prec s_0$，那么由于 s 覆盖 p， s 覆盖 p_2。否则，如果 $s_0 \prec s_1$，那么这样的点 p_2 不可能存在。因此，在处理 p 期间选择 s 不会导致活动重叠度增加，我们的算法将贪心地选择 s。

3. 底部锚定 DESC： s_1 和 s_2 与底线 L_1 和 $s_1 \prec s_2$ 相交，而 s_1 位于 s_2上方。再次分为两个子情况。

（a）如果 s 与顶线 L_2相交：那么在 s_2的最左侧专属点 p 处，我们的算法本应选择 s 而不是 s_2。原因与上述情况1(a)中的论证完全相同。（b）如果 s 与底线 L_1相交：在处理 s_2的最左侧专属点 p 时，我们的算法已经选择了 s_1，因为 s_1 的最右侧专属点必定位于 p的左侧。在 p处，我们的算法会选择 s 而不是 s_2，因为选择 s 会使 s_1 变得冗余。这里有两种可能性。第一，如果 $s_2 \prec s$，那么我们的算法会优先选择 s 而不是 s_2，因为这样会得到一个大小相同但更窄的团。第二，如果 $s_1 \prec s \prec s_2$，那么 s 覆盖 $s_1 \cap s_2$中的每个点。考虑一个点$p_1 \in s_1 \setminus s_2$，它被另一个正方形 $s_0 \in Sol$覆盖。如果 $s_0 \prec s_1$，那么当我们处理 p时， s_0 已经被我们的算法选中。如果 $s_2 \prec s_0$，那么这样的 p 不可能存在。参见图（）。考虑一个点 $p_2 \in s_2 \setminus s_1$，它被另一个正方形 $s_0 \in Sol$覆盖。如果 $s_2 \prec s_0$，那么这样的点 p_2 位于 p的右侧。否则，如果 $s_0 \prec s_1$，那么这样的点 p_2 不可能存在。因此，为 p选择 s 不会导致活动重数增加，这与我们的贪心选择一致。

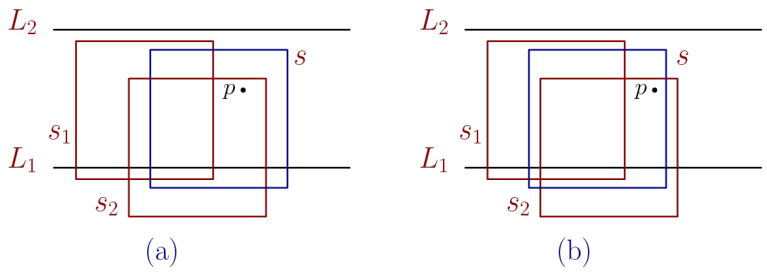


图*20*。正方形 s_1 和 s_2 在所考虑的团中均为底部锚定，并按降序排列。

4. 底部锚定升序： s_1 和 s_2 与底线 L_1 和 $s_1 \prec s_2$ 及 $x_T(s_1) < x_T(s_2)$相交。同样有两种子情况。

(a) 如果 s 与顶线 L_2相交：那么在 s_2的最左侧专属点 p 处，我们的算法必须在 s 和 s_2之间作出选择。由于已经选取了 s_1，我们的算法会优先选择 s 而非 s_2，因为选择s_2 会得到一个浮动团。而且我们的算法之后绝不会再选择 s_2。回想一下，我们的贪心算法会优先选择浮动团，而不是锚定团。

(b) 如果 s 与底线 L_1相交：那么在 s_2的最左侧专属点 p 处，我们的算法会选择 s 而不是 s_2，因为选择 s 会使 s_1 变得冗余。由于s_1 的独占区域是矩形，且正方形 s 覆盖了 $Excl$（s_2）中的所有点，因此 s 位于 s_2的左侧，即$s \prec s_2$。因此， s 覆盖了 $s_1 \cap s_2$ 中位于 p左侧的每个点。考虑一个由另一个正方形 $s_0 \in Sol$覆盖的点 $p_1 \in s_1 \setminus s_2$。显然， s_0 的所有专属点都位于 p的左侧。根据引理（2.21），我们的解必定已经选取了 s_0。再考虑一个由另一个正方形 $s_0 \in Sol$覆盖的点 $p_2 \in s_2 \setminus s_1$。如果 $s_2 \prec s_0$，那么 s_0 中的所有点都位于p的右侧。因此，在此阶段我们无需担心覆盖 p_2。否则，如果 $s_0 \prec s_1$，那么 $s_0 \cap s_2$ 将包含在 $s_1 \cap s_2$ 中，而这样的点 p_2 不可能存在。因此，在处理 p 期间选取 s 不会增加活动重叠度，我们的算法会采用贪心策略选取 s。

5. 浮动降序： s_1 与顶线 L_2 相交，且 s_2 与底线 L_1 相交，且 $s_1 \prec s_2$。同样，这里有两种子情况。

(a) 正方形 s 与顶线 L_1相交：如果 $s_2 \prec s$，那么在 s_1的最左侧专属点 p 处，我们的算法会选择 s 而不是 s_1，因为选择 s 也会得到一个大小相同但更窄的团。如果 $s_1 \prec s \prec s_2$，那么在 s_1的最左侧专属点 p 处，我们的算法会选择 s 而不是 s_1，因为选择 s 会得到一个大小相同但更窄的团。如果存在一个点 $p_1 \in s_1 \cap s_2$，它没有被 Sol中的任何其他正方形覆盖，那么在处理 p_1时，我们的算法会选择 s_2 而不是 s_1，因为这样会得到一个更窄的团。因此，我们的算法绝不会选择 s_1。如果 $s \prec s_1 \prec s_2$，那么在 s_2的最左侧专属点 p 处，我们的算法会选择 s 而不是 s_2，并丢弃 s_1。如果存在一个点 $p_1 \in s_1 \cap s_2$，它没有被 Sol 中的任何其他正方形或 s覆盖，那么在处理 p_1时，我们的算法会选择 s_2 而不是 s_1。因此，我们的算法绝不会选择 s_1。

(b) 正方形 s 与底线 L_1相交：适用与情形 5(a) 完全相同的论证。

6. 浮动上升： s_1 与底线 L_1 相交，且 s_2 与顶线 L_2 和 $s_1 \prec s_2$相交。此情形适用与浮动下降情形中所述的类似论证。

由于 s_1、 s_2 和 s没有其他可能的情况，因此我们的断言得证。 □

引理 2.24. 考虑我们的解 Sol中的一个最大团，例如 K。为了覆盖构成 K的正方形中的独占点，任何可行集合覆盖都必须选取 $\lfloor k/3 \rfloor$ 个正方形，其中 $|K| = k$。

证明。 根据命题 (2.23)，必须使用 $\lfloor k/3 \rfloor$ 个正方形覆盖团 K 中 Sol 个正方形的专属点。我们已经证明，升序团（分别是下降团）中的专属点从左到右单调升序（分别单调下降），但最多有 4 个正方形可能例外。首先讨论 K 是升序类型的单调团的情况。考虑 K 中连续的四个正方形，例如 s_1、 s_2、 s_3 和 s_4，它们都与顶线 L_2 相交。没有正方形 s 能覆盖这 4 个正方形中的专属点。否则，该正方形 s 最终会覆盖 $Excl(s_1) \cup Excl(s_2)$ 中的所有点。参见图 21。如果这些正方形都与底部相交，也可用类似的论证。唯一的例外情况是部分正方形与顶部相交、部分正方形与底部相交，如图 22 所示。在这种情况下，一个正方形 s 可能覆盖 s_1、 s_2、 s_3、 s_4 中所有正方形的专属点。但出于类似原因，不可能覆盖 5 个正方形的专属点

出于类似原因可能是不可能的。
现在从左到右将最大团中的正方形分成若干组，每组包含 3 个。每组至少需要 1 个正方形，但在从上到下或从下到上的过渡处，至多有一组例外。若存在过渡正方形，类似的论证同样适用。因此，$\lfloor k/3 \rfloor$ 个正方形是必要的。 □

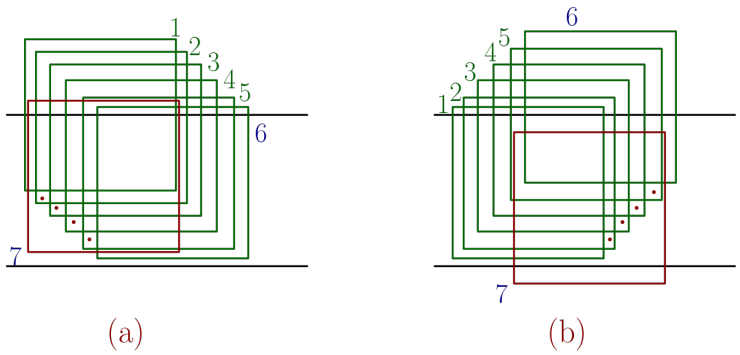


图 *21.* (a) 考虑这个顶部锚定单调下降团中的 4 个相邻正方形 2, 3, 4,5。正方形 7 覆盖了全部四个正方形中的专属点。具体而言，它覆盖了 $Excl(3) \cup Excl(4)$ 中的所有点。(b) 考虑这个顶部锚定单调上升团中的 4 个相邻正方形 2, 3, 4, 5。正方形 7 覆盖了全部四个正方形中的专属点。具体而言，它覆盖了 $Excl(3) \cup Excl(4)$ 中的所有点。

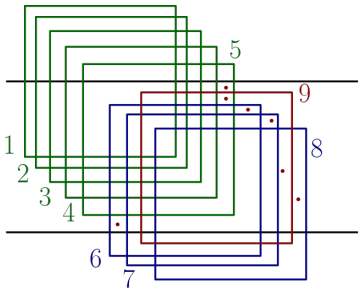


图 *22.* 这个浮动团中的正方形 9 覆盖了正方形 5、6、7、8 的专属点，并且只覆盖其中一个正方形的全部专属点，即正方形 7。

引理 2.25. 在覆盖 Sol 中大小为 k 的团所需的正方形中，至少有 $\lfloor \frac{k}{9} \rfloor - 1$ 个正方形具有公共交集。

证明. 首先，考虑一个顶部锚定的升序团。除最左侧正方形处可能例外外，专属点也按单调升序排列。在这个团中，考虑一个非极端专有点 p。如图 23(b) 所示，它要么可以由左侧的顶部相交正方形覆盖，要么可以由右侧的底部相交正方形覆盖。这意味着，覆盖非极端专有点的所有底部相交正方形彼此相交。同理，覆盖非极端专有点的所有顶部相交正方形也彼此相交。因此形成了两个团。应用鸽巢原理，其中一个团的大小至少为 $\frac{\lfloor k/3 \rfloor - 2}{2} = \lfloor \frac{k}{6} \rfloor - 1$。

对于单调浮动团，或由两个单调正方形序列组成的团，即类型为 ASC|DESC 或 DESC|ASC 的团，独占点可能具有两个不同的单调序列。因此，我们需要采用不同的论证。

考虑一个类型为底部锚定 ASC|DESC 的团 Q。注意，Q 中的独占点有两个单调序列——第一个序列升序排列，第二个序列降序排列。和之前一样，需要两个团来覆盖第一个独占点单调序列中的点。将这两个团分别记为 Q_1 和 Q_2，其中 Q_1 为顶部锚定，Q_2 为底部锚定。此外，OPT 中还有另外两个团，用于覆盖第二个单调序列中的独占点。将这两个团分别记为 Q_3 和 Q_4，其中 Q_3 为底部锚定，Q_4 为顶部锚定。所有 L_1（即底线）相交正方形，也就是 Q_2 和 Q_3 中的正方形，都有一个公共交集区域。参见图 23(d)。因此，在 OPT 中存在一个大小为

$$\max(|Q_1|, |Q_2| + |Q_3|, |Q_4|)$$

根据鸽巢原理，其中一个团的大小至少为 $\lfloor k/3 \rfloor_3 - 1 = \lfloor \frac{k}{9} \rfloor - 1$。 —— –

考虑一个类型为浮动降序|降序的团 Q。注意，Q 中有两个由独占点构成的单调序列——这两个序列都是降序的。与之前一样，至少需要两个团才能覆盖第一个由独占点构成的单调序列。将这两个团记为 Q_1 和 Q_2，其中 Q_1 是底部锚定的，Q_2 是顶部锚定的。此外，OPT 中还有另外两个团，用于覆盖第二个单调序列中的独占点。将这两个团记为 Q_3 和 Q_4，其中 Q_3 是底部锚定的，Q_4 是顶部锚定的。这里的几何关系是：Q_2 中所有与 L_2 相交的正方形，都与 Q_3中和 L_1相交的正方形相交。参见图 23(e)。因此，在 OPT 中存在一个大小为

$$\max(|Q_1|, |Q_2| + |Q_3|, |Q_4|)$$

根据鸽巢原理，其中一个团的大小至少为 $\lfloor k/3 \rfloor_3 - 1 = \lfloor \frac{k}{9} \rfloor - 1$。 —— – □

定理 2.26。 给定平面上的 n 个点和 m 个轴平行单位正方形，我们的算法可在 $O((nm)^2)$ 时间内计算出最小重叠度覆盖的一个 $(27+\epsilon)$近似因子近似解，其中 $\epsilon > 0$ 是一个较小的正常数。

证明。 近似因子直接由定理 2.9 和引理 2.25 得出。水平条带子问题的算法会计算一个包含 nm 个表项的表。每个表项可在 $O(m)$ 时间内计算。因此，每个子问题需要 $O(nm^2)$ 时间。子问题至多有 n 个。因此，所需总时间为 $O((nm)^2)$。

□

3. 结论

本文为轴平行单位正方形的最小重叠度覆盖问题提出了一种运行速度快的算法技术。我们刻画了解中任意团的结构，并将其与最优解的交图中的最大团进行了比较。近似比或许还能进一步改进。我们认为，这种技术可以推广，从而为更广泛的一类对象构造多项式时间近似算法。

致谢。我想感谢 Sathish Govindarajan，就最小层覆盖问题展开了许多有益的讨论，并提出了宝贵意见。我还要感谢 Aniket Basu Roy 和 Shirish Gosavi 抽出时间与我讨论该问题。

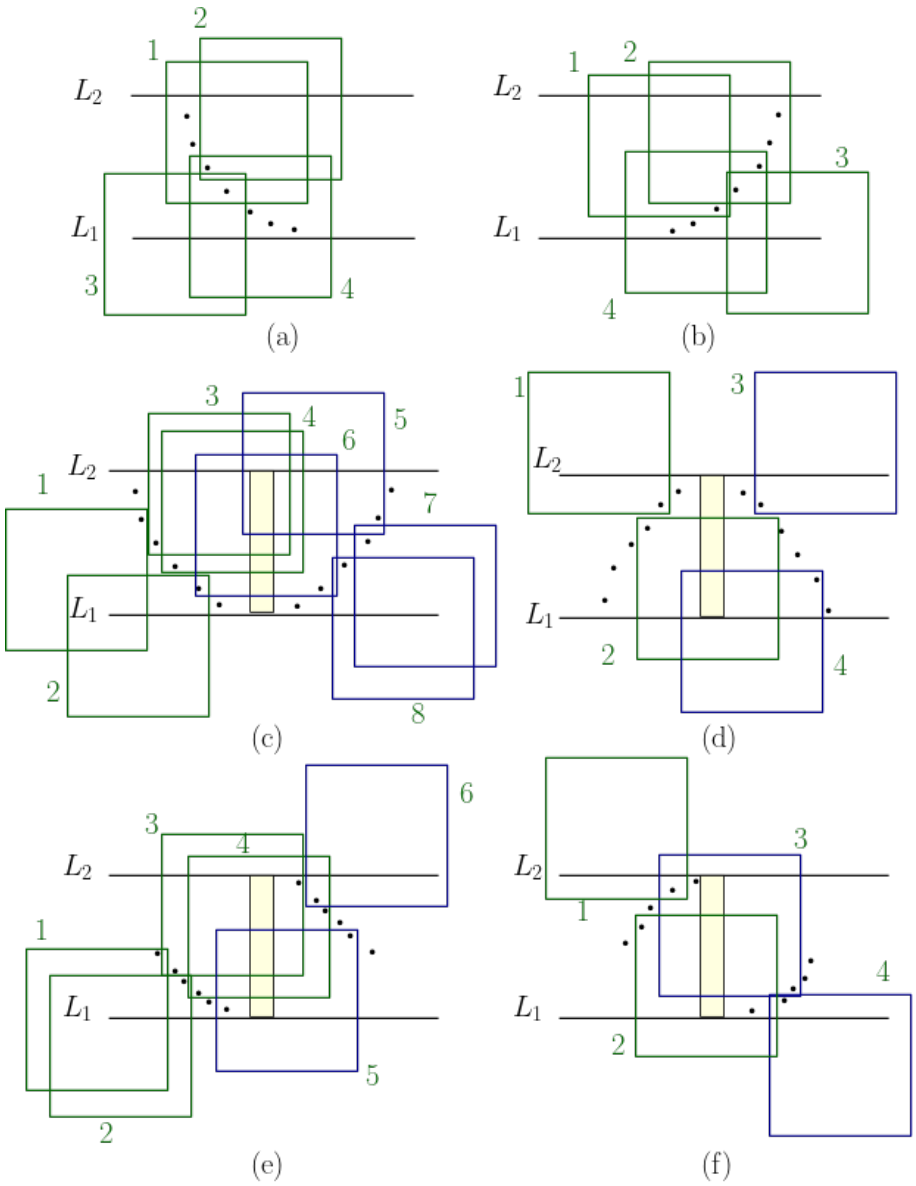


图*23*。 展示了引理（2.25）证明中的不同情形。

参考文献

Pankaj K. Agarwal 和 Jiangwei Pan。几何命中集与集合覆盖的近线性算法。载于第三十届计算几何年度研讨会论文集，SOCG'14，第 271–279 页，美国纽约州纽约市，2014年。计算机协会。国际标准书号 9781450325943。数字对象标识符：10.1145/2582112.2582152。网址：https://doi.org/10.1145/2582112.2582152。

Therese Biedl、Ahmad Biniaz 和 Anna Lubiw。使用圆盘和正方形对点进行最小层覆盖。计算几何，94:101712，2021。ISSN 0925-7721。数字对象标识符：https://doi.org/10.1016/j.comgeo.2020.101712。URL：https://www.sciencedirect.com/science/article/pii/S0925772120301061。

Gruia C˘alinescu、Ion I。M˘andoiu、Peng-Jun Wan 和 Alexander Z. Zelikovsky。《选择转发邻居》 s

无线自组网络中的转发邻居选择。移动网络与应用，9(2):101–111，2004年4月。ISSN 1572- 8153。数字对象标识符: 10.1023/B:MONE.0000013622.63511.57。URL https://doi.org/10.1023/B:MONE.0000013622.63511.57。

Timothy M. Chan 和 Elyot Grant。几何装箱与覆盖问题的精确算法及 APX-困难性结果。计算几何，47(2, Part A):112–124，2014年。ISSN 0925-7721。数字对象标识符：https://doi.org/10.1016/j.comgeo.2012.04.001。URL https://www.sciencedirect.com/science/article/pii/S0925772112000740。专题：第 23 届加拿大计算几何会议（CCCG11）。

Kenneth L. Clarkson 和 Kasturi Varadarajan。改进的几何集合覆盖近似算法。离散与计算几何，37(1):43–58，2007年 1 月。ISSN 1432-0444。数字对象标识符：10.1007/s00454-006-1273-8。URL https://doi.org/10.1007/s00454-006-1273-8。

E. D Demaine、Mohammad T. Hajiaghayi、U. Feige 和 M. R Salavatipour。《组合可能很难：唯一覆盖问题的近似难度》。载于第十七届 *ACM-SIAM* 离散算法研讨会论文集，第 162–171 页，2006/// 2006。

Stephane Durocher、J. Mark Keil 和 Debajyoti Mondal。单位正方形的点最小重叠度覆盖。*CoRR*，abs/2208.06122，2022。数字对象标识符：10.48550/arXiv.2208.06122。网址：https://doi.org/10.48550/arXiv.2208.06122。

Thomas Erlebach 和 Erik Jan van Leeuwen。几何覆盖问题的近似算法。载于第十九届 *ACM-SIAM* 离散算法研讨会论文集，SODA '08，第 1267–1276 页，美国，2008 年。工业与应用数学学会。

Thomas Erlebach 和 Erik Jan van Leeuwen。单位正方形上的加权集合覆盖的多项式时间近似方案。载于 Maria Serna、Ronen Shaltiel、Klaus Jansen 和 Jos´e Rolim 编，《近似、随机化与组合优化：算法与技术》，第 166–177 页，柏林，海德堡，2010 年。Springer Berlin Heidelberg。国际标准书号 978-3-642-15369-3。

Uriel Feige。集合覆盖近似的 ln n 阈值。《ACM学报》，45(4):634–652，1998年7月。ISSN 0004-5411。数字对象标识符：10.1145/285055.285059。网址：https://doi.org/10.1145/285055.285059。

Robert J. Fowler、Michael S. Paterson 和 Steven L. Tanimoto。平面上的最优装箱与覆盖是 NP完全问题。《信息处理快报》，12(3):133–137，1981年。ISSN 0020-0190。数字对象标识符：https://doi.org/ 10.1016/0020-0190(81)90111-3。网址：https://www.sciencedirect.com/science/article/ pii/0020019081901113。

Dorit S. Hochbaum 和 Wolfgang Maass。用于图像处理和超大规模集成电路中覆盖与装箱问题的近似方案。《*ACM*学报》，32(1):130–136，1985年1月。ISSN 0004-5411。数字对象标识符：10.1145/2455.214106。网址：https://doi.org/10.1145/2455.214106。

Fabian Kuhn、Pascal von Rickenbach、Roger Wattenhofer、Emo Welzl 和 Aaron Zollinger。《蜂窝网络中的干扰：最小成员集合覆盖问题》。载于 Lusheng Wang 编，计算与

组合学，第 188–198 页，柏林、海德堡，2005 年。Springer Berlin Heidelberg。国际标准书号 978-3-540-31806- 4。

Nabil H. Mustafa、Rajiv Raman 和 Saurabh Ray。确定几何集合覆盖的 APX-困难性状态。载于第*55*届 *IEEE*计算机科学基础年会，*FOCS 2014*，费城，美国宾夕法尼亚州，*2014*年*10*月*18*日至*21*日，第541–550页。IEEE计算机学会，2014年。数字对象标识符：10.1109/FOCS.2014.64。网址：https://doi.org/10.1109/FOCS.2014.64。

Ran Raz 和 Shmuel Safra。一种亚常数错误概率低次测试，以及一种 NP 的亚常数错误概率 PCP 刻画。载于第二十九届 *ACM*计算理论年会论文集，STOC '97，第475–484页，美国纽约州纽约，1997年。计算机协会。国际标准书号 0897918886。数字对象标识符：10.1145/258533.258641。网址：https://doi.org/10.1145/258533.258641。