arXiv:2303.09524v1 [cs.CG] 16 Mar 2023

# 几何集合覆盖与击中集的在线与动态算法

Arindam Khan*[1]、Aditya Lonkar[1],Saladi Rahul[1],Aditya Subramanian†[1]和 Andreas Wiese[2]

[1]印度科学研究所，班加罗尔，印度[2]慕尼黑工业大学，德国

2023年3月17日

**摘要**

集合覆盖和击中集是组合优化中的基本问题，在线、离线和动态设置下均已得到深入研究。我们研究这些问题的几何版本，并提出相应的新在线与动态算法。在集合覆盖（分别为击中集）的在线版本中，给定 $m$ 集合（分别为 $n$ 点），而 $n$ 点（分别为 $m$ 集合）逐个在线到达。在动态版本中，点（分别为集合）既可以加入，也可以离开。我们的目标是维护一个集合覆盖（分别为击中集），并使所求解的大小最小。

对于任意大小的（轴平行）正方形的在线集合覆盖，我们给出了一个紧确的 $O(\log n)$-竞争算法。在相同设置下，对于击中集，假设所有点在 $[0, N)^2$ 中都具有整数坐标，我们给出了一个紧确的 $O(\log N)$-竞争算法。此前，这两种设置下都没有已知的在线算法，甚至对单位正方形也是如此（一般集合系统的已知在线算法除外）。

对于采用 $d$ 维超矩形的动态集合覆盖和击中集，我们获得了 $(\log m)^{O(d)}$-近似算法，其最坏情况更新时间为 $(\log m)^{O(d)}$。这部分回答了 Chan 等人 [SODA'22] 提出的一个开放问题。此前，即使在正方形的设置下，这两个问题也都没有已知的更新时间为多对数级的动态算法。我们的主要技术贡献是扩展四叉树 方法，以及频率归约 技术；该技术可将几何集合覆盖实例归约为频率有界的一般集合覆盖实例。

## 1 引言

几何集合覆盖是计算几何中的一个基础且经过深入研究的问题 [21,18, 34, 28, 30]。在这里，给定全集 $P$，其中包含 $\mathbb{R}^d$ 中的 $n$ 个点，以及一个由 $m$ 个集合构成的集合族 $\mathcal{S}$，其中每个集合 $S \in \mathcal{S}$ 都是一个几何对象（我们假设 $S$ 是 $\mathbb{R}^d$ 中的一个闭集合，并且 $S$ 覆盖了 $P \cap S$ 中的所有点），例如超矩形。我们的目标是从这些集合中选取一个集合 $\mathcal{S}' \subseteq \mathcal{S}$，使其包含（即覆盖） $P$ 中的所有元素，同时最小化 $\mathcal{S}'$ 的基数（示意图见图 1）。

*获 IUSSTF 虚拟中心"多项式作为算法范式"项目、Pratiksha Trust 青年研究者奖、Google 印度研究奖、Google ExploreCS 奖，以及 SERB 核心研究资助（CRG/2022/001176）"不可处理性与不确定性下的优化"项目的支持。

†部分获得 Kotak IISc AI-ML Centre（KIAC）博士奖学金的支持。

集合系统（$P$, $\mathcal{S}$）的频率 $f$ 定义为包含 $P$ 中某个元素的集合的最大数量。

在几何集合覆盖的某些情形中，离线设置下已知的近似比优于一般集合覆盖，例如，对于（轴平行）正方形 [31]，存在多项式时间近似方案（PTAS）。然而，对于几何集合覆盖的在线和动态变体，我们所知甚少。在线设定中，集合预先给定，点逐个到达；对于未覆盖点，我们必须立即且不可撤销地选择一个（覆盖该点的）集合。据我们所知，即使对于二维单位正方形，也没有已知的在线算法能取得渐近上优于一般在线集合覆盖 [4, 16] 的 $O(\log n \log m)$ 竞争算法的竞争比。在动态情形中，集合同样预先给定，并且每个时间步都会插入或删除一个点。在此，我们关注的是当输入发生变化时，能够快速更新当前解的算法。具体而言，更新时间为多对数级的算法更为理想。遗憾的是，几何集合覆盖领域几乎没有已知的此类算法。Agarwal 等人 [2] 首先研究了区间和二维单位正方形的动态几何集合覆盖，并分别提出了具有多对数更新时间的 $(1+\varepsilon)$ 近似算法和 $O(1)$ 近似算法。据我们所知，对于更一般的对象，例如矩形、三维立方体或更高维度中的超矩形，目前尚无此类动态算法。请注意，在动态几何集合覆盖中，插入的点以其坐标表示；与一般（动态）集合覆盖相比，这种表示更紧凑（在后者中，对于每个新点 $p$，我们都会得到一个包含该点 $p$ 的集合列表，因此仅仅读取该输入就可能需要 $\Omega(f)$ 的时间）。

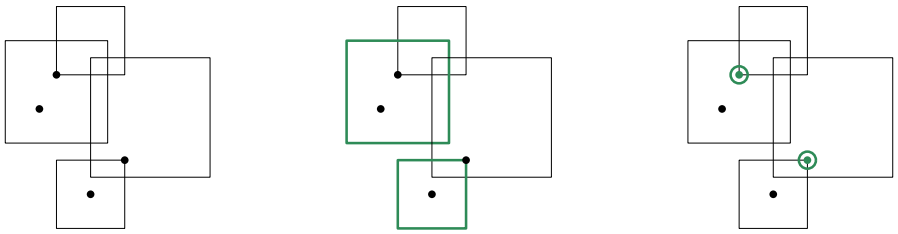


图 1：（a）一个正方形集合 $\mathcal{S}$ 和一个点集 $P$；（b）一个覆盖 $P$的集合覆盖（绿色） $\mathcal{S}' \subseteq \mathcal{S}$；（c）针对 $\mathcal{S}$的击中集（绿色的点） $P' \subseteq P$。

与集合覆盖相关的是击中集问题（图 1 展示了一个示例）：给定一个点集 $P$和一个集合族 $\mathcal{S}$，我们希望选出最少数量的点 $P' \subseteq P$，使其击中每个集合 $S \in \mathcal{S}$，也就是说，满足 $P' \cap S \neq \emptyset$。同样，在离线几何情形中，已知的近似比优于一般情形，例如，针对正方形 [31], 有一个多项式时间近似方案，针对矩形 [5]有一个$O(\log \log \mathrm{OPT})$-近似算法。然而，在线和动态情形中，已知的改进一般情形结果的研究寥寥无几。在线设定中，针对 $d = 1$（即区间），有一个 $O(\log n)$-竞争算法；针对单位圆盘 [23]，有一个 $O(\log n)$-竞争算法。动态情形中，已知算法仅适用于区间和单位正方形（因此也适用于象限），其近似比分别为 $(1+\varepsilon)$ 和 $O(1)$ [2]。

### 1.1 我们的研究成果

本文研究任意尺寸的正方形的几何集合覆盖与击中集在线算法；此前，即使对于单位正方形，也没有已知的改进结果。此外，我们

还提出了针对常数维超矩形（也称为 $d$-盒或正交多面体）的这些问题的动态算法。与此前研究的几何对象相比，这类对象的范围要广泛得多，例如区间 [2] 或二维正方形 [20]。

**正方形的在线集合覆盖** 在第2节中，我们研究任意大小轴平行正方形的在线集合覆盖，并给出一个 $O(\log n)$- 竞争算法。我们还匹配了下界 $\Omega(\log n)$（渐近意义下），因此我们的竞争比是紧的。在我们的在线模型中（与 [4] 中一样），我们假设集合（正方形）一开始就已给定，而元素（点）在线到达。

我们的在线算法基于一种新的单调离线算法；也就是说，如果我们向 $P$ 添加一个新点 $p$，该算法输出的正方形集合是仅给定 $P$ 而不包含 $p$ 时输出的正方形集合的超集。该算法基于四叉树分解。它从根节点遍历到叶节点，并针对仍有未覆盖点的每个单元 $C$，考察 $C$ 的每条边 $e$，然后选择包含 $e$ 且"最有用"的正方形，即与 $C$的交集最大的正方形。我们假设（全文均如此）所有点以及正方形的所有顶点在 $[0, N)^2$ 中都具有整数坐标，其中 $N$ 为给定值，并由此得到 $O(\log N)$ 的竞争比。如果我们知道所有插入的点都来自一个预先给定的候选点集合 $P_0$（例如 Alon 等人 [4] 的情形），则可将竞争比提高到 $O(\log n)$。对于这种情形，我们使用 Arya 等人 [7] 提出的 BBD 树数据结构。它将单元分解得比标准四叉树更复杂，我们以一种非平凡的方式对算法进行相应调整。由于我们的离线算法具有单调性，我们立即得到一个 $O(\log n)$-竞争的在线算法。

**正方形的在线击中集** 在第 3 节中，我们提出了一种针对任意大小正方形的在线击中集问题的 $O(\log N)$-竞争算法，其中点集预先给定，而正方形在线到达。这与已知的针对简单得多的区间情形的最佳 $O(\log N)$-竞争算法相匹配 [23]。此外，即使对于区间，也存在 $\Omega(\log N)$ 的匹配下界。

简而言之，如果对手插入一个新的正方形 $S$，我们会找出 $O(\log N)$ 个四叉树单元，其中 $S$ 包含该正方形的一条边。然后，我们从这些单元中选出最有用的点来击中这些正方形：也就是距离单元四条边最近的点。我们称这会激活该单元。在分析中，我们反过来论证：对于 OPT 中的每个点 $p \in$，如果插入的正方形 $S$ 被 $p$ 击中，那么最多只有$O(\log N)$ 个单元可能被激活。由此可得竞争比为 $O(\log N)$。

**动态集合覆盖和击中集，适用于 $d$维超矩形** 接着，我们在第4节和第5节介绍适用于 $d$ 维超矩形的集合覆盖和击中集动态算法。值得注意的是，即便对于矩形的集合覆盖，也尚无更新时间和近似比均为多对数级别的动态算法；Chan 等人 [20] 曾明确提出，是否存在这样的算法。因此，对于仅插入和删除点的情形，我们肯定地回答了这一问题。正如上文所述，当我们希望显式存储解时，这正是相关的情形。尽管我们考虑的对象非常一般，我们的算法仍只需多对数级别的最坏情况更新时间。相比之下，Abboud等人 [1] 证明，在强指数时间假设成立的条件下，任何一般的（动态）集合覆盖算法若具有 $O(f^{1-\varepsilon})$的摊销更新时间，其近似比必须达到 $\Omega(n^{\alpha})$，其中 $\alpha > 0$为某个常数，且$f$ 可以大至 $\Theta(m)$。

我们首先讨论集合覆盖算法。我们先将 $d$ 维超矩形的情形归约为 $[0, 4m]^{2d}$ 中具有整数顶点的 $2d$ 维超立方体。接下来，一种自然的

方法是将上文的正方形算法推广到 $2d$ 维超立方体。一种规范的推广方法是构建一棵四叉树，从根节点遍历到叶节点，并针对每个单元 $C$ 和 $C$ 的每个面 $F$，选取包含 $F$ 的最有用的超立方体 $S$，也就是与 $C$ 交集最大的超立方体 $S$。遗憾的是，这已不足以解决问题，甚至在三维中也不够：可能存在一个单元 $C$，对于它，必须选取仅包含 $C$ 的一条边、而不包含 $C$ 的任何一个面上的立方体（见图 2）。这里，我们提出一个关键的新思路：对于（标准）四叉树中的每个单元 $C$ 和每个维度 $i \in [2d]$，考虑沿维度 $i$ 对 $C$ 进行“边覆盖”的超立方体。基于这些超立方体，在除第 $i$ 维之外的所有维度上构建一个 $(2d-1)$ 维递归辅助结构（见图 14）。

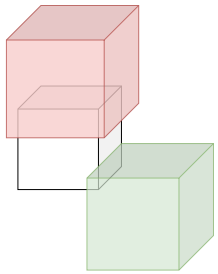

图 2：红色立方体是唯一覆盖未着色单元的一个面的立方体。绿色立方体（来自 OPT）只覆盖该单元的一条边。注意，OPT 中的立方体没有任何角落位于该单元内。选取红色立方体并不能覆盖绿色立方体与该单元的交集。

我们将得到的树称为扩展四叉树。尽管它比标准四叉树大得多，我们证明每个点只包含在 $(\log m)^{O(d)}$ 个单元中。此外，我们将其用于第二个关键思路，降低集合覆盖实例的频率：我们构建一个频率有界的一般集合覆盖辅助实例。它与给定的几何集合覆盖实例具有相同的点，但集合不同：对于扩展四叉树中对应于一维单元 $C$的每个节点，我们考察它的每个端点 $p$，并引入一个对应于覆盖 $p$的“最有用”超立方体的集合，也就是与 $C$交集最大的覆盖 $p$ 超立方体。由于每个点只包含在 $(\log m)^{O(d)}$ 个单元中，因此得到的频率至多为 $(\log m)^{O(d)}$。此外，我们证明辅助集合覆盖实例存在一个至多包含 $\mathrm{OPT}\cdot(\log m)^{O(d)}$ 个集合的解。随后，我们使用 [13] 中用于一般集合覆盖的动态算法，维护辅助实例的一个近似解，从而得到一个动态的 $(\log m)^{O(d)}$ 近似算法。

我们进一步将上述动态集合覆盖算法应用于 $d$维超矩形的击中集，其近似比为 $(\log n)^{O(d)}$。最后，我们将集合覆盖和击中集算法扩展到 $d$维超矩形的加权情形；假设输入中的所有集合/点的权重均属于 $[1, W]$，更新时间和近似比仅需额外增加一个 $(\log\ W)^{O(1)}$ 因子。我们的结果总结见下表。

## 1.2 其他相关工作

一般集合覆盖在在线和动态设置下都得到了充分研究。人们考虑了在线集合覆盖的若干变体和推广，例如在线子模覆盖 [27], 随机顺序到达下的在线集合覆盖 [24],带补救的在线集合覆盖 [25], 等。

对于动态集合 击问题方面，Gupta 等人 [25] 开启了相关研究，并给出了 $O(\log n)$-近似

| 问题 | 对象 | 竞争比 | 下界 |
|---|---|---|---|
| 集合覆盖 | 区间 | 2 [定理 7] | 2 [定理 8] |
| | 二维正方形 | $O(\log n)$ [定理 1] | $\Omega(\log n)$ [定理 1] |
| 击中集 | 区间 | $O(\log N)$ [23] | $\Omega(\log N)$ [23] |
| | 二维正方形 | $O(\log N)$ [定理 3] | $\Omega(\log N)$[23] |

表1：几何集合覆盖和击中集的在线算法。

| 问题 | 对象 | 近似比 | 更新时间 |
|---|---|---|---|
| 集合覆盖 | 二维单位正方形 | $O(1)$ [2] | $(\log n)^{O(1)}$ |
| | $d$-维超矩形 | $O(\log^{4d-1} m)\log W$ [定理 5] | $O(\log^{2d} m)\log^3(Wn)$ |
| 击中集 | 单位正方形 | $O(1)$ [2] | $(\log n)^{O(1)}$ |
| | $d$-D 超矩形 | $O(\log^{4d-1} n)\log W$ [定理 6] | $O(\log^{2d-1} n)\log^3(Wm)$ |

表2：几何集合覆盖和击中集的动态算法。 [2] 中的更新时间为摊还时间，且适用于无权情形。我们的结果针对最坏情况更新时间。

算法，其摊销更新时间为 $O(f\log n)$，即使在加权情形下也是如此。与我们的模型类似，他们的模型中集合是离线给定的，只有元素可以出现或离开。此后，出现了一系列相关研究 [1, 10, 12, 11, 13, 25, 26, 8]。

Bhattacharya 等人 [13] 提出了确定性 $(1+\varepsilon)$ $f$-近似算法，其摊还更新时间为 $O\left((f^2/\varepsilon^3) + (f/\varepsilon^2)\log(W)\right)$，最坏情况更新时间为 $O(f\log^2(Wn)/\varepsilon^3)$，其中 $W$ 表示权重最高的集合与权重最低的集合的权重之比。Assadi 和 Solomon [8] 提出了一种随机化 $f$-近似算法，其摊还更新时间为 $O(f^2)$。

Agarwal 等人 [2] 研究了几何集合覆盖的另一种动态设置，其中点和集合都可以到达或离开，并针对区间和单位正方形分别提出了具有次线性更新时间的 $(1+\varepsilon)$ 近似算法和 $O(1)$-近似算法。Chan 和 He [19] 将其扩展到任意正方形的集合覆盖。最近，Chan 等人 [20] 针对区间这一特殊情形，提出了 $(1+\varepsilon)$ 近似算法，其摊还更新时间为 $O(\log^3 n/\varepsilon^3)$。他们还针对单位正方形、任意正方形和带权区间的动态集合覆盖，分别提出了 $O(1)$-近似算法，摊还更新时间分别为 $2^{O(\sqrt{\log n})}n^{1/2+\varepsilon}$ 和 $2^{O(\sqrt{\log n}\log\log n)}$。

针对其他几何问题的动态算法也得到了深入研究，例如区间和超矩形的最大独立集 [29, 14, 17]，以及几何测度 [22]。

## 2 正方形覆盖

本节介绍我们针对正方形集合覆盖的在线与动态算法。给定一个 $m$ 正方形集合 $\mathcal{S}$，满足每个正方形 $S\in\mathcal{S}$ 在 $[0,N)^2$ 中具有整数顶点。不失一般性，假设 $N$ 是 2 的幂。我们首先描述一个 $O(\log N)$-近似的离线 $O$ 算法。然后基于该算法构造一个在线算法和一个动态算法，使二者的近似比同样均为 $O(\log N)$。对于离线算法，除 $\mathcal{S}$ 和 $N$ 之外，我们还给定一个需要覆盖的点集 $P$，满足 $P\subseteq[0,N)^2$，且每个点 $p\in P$ 具有整数坐标。

四叉树 我们首先给出四叉树的定义 $T = (V, E)$，与例如 [6, 9]中的定义类似。在 $T$中，每个节点 $v \in V$对应一个正方形单元 $C_v \subseteq [0, N)^2$，其顶点具有整数坐标。根节点 $r \in V$的 $T$对应于单元 $C_r := [0, N)^2$。递归地，考虑一个节点 $v \in V$，它对应于单元 $C_v$，并假设 $C_v = [x_1^{(1)}, x_2^{(1)}) \times [x_1^{(2)}, x_2^{(2)})$。如果 $C_v$是一个单位正方形，即 $|x_2^{(1)} - x_1^{(1)}| = |x_2^{(2)} - x_1^{(2)}| = 1$，则将 $v$定义为叶节点。否则，将 $v$定义为具有四个子节点 $v_1$、$v_2$、$v_3$、$v_4$，它们分别对应于将 $C_v$ 划分为四个大小相等的较小单元后得到的四个单元，即定义 $x^{(1)}\,\mathrm{mid} := (x_2^{(1)} - x_1^{(1)})/2$ 和 $x^{(2)}\,\mathrm{mid} := (x_2^{(2)} - x_1^{(2)})/2$，以及 $C_{v_1} = [x_1^{(1)}, x^{(1)}\,\mathrm{mid}) \times [x_1^{(2)}, x^{(2)}\,\mathrm{mid})$、$C_{v_2} = [x_1^{(1)}, x^{(1)}\,\mathrm{mid}) \times [x^{(2)}\,\mathrm{mid}\ x_2^{(2)})$、$C_{v_3} = [x^{(1)}\,\mathrm{mid}\, x_2^{(1)}) \times [x_1^{(2)}, x^{(2)}\,\mathrm{mid})$、和 $C_{v_4} = [x^{(1)}\,\mathrm{mid}\, x_2^{(1)}) \times [x^{(2)}\,\mathrm{mid}\, x_2^{(2)})$。注意，该树的深度为 $\log\ N$，其中树中节点的深度是其到 $T$根节点的距离，而 $T$的深度是$T$中任意节点的最大深度。根据构造，每个叶节点至多包含一个点，并且该点将位于相应单元的左下角。

**离线算法**在离线算法 $\mathcal{A}_{\text{off}}$中，我们按照广度优先顺序遍历 $T$，即按照节点到根节点 $r$ 的距离对 $V$中的节点排序，并按此顺序依次考察它们（以任意但固定的方式打破平局）。假设在某次迭代中，我们考察节点 $v \in V$，它对应一个单元 $C_v$。我们检查 $v$ 的祖先节点中选出的正方形是否覆盖了 $P \cap C_v$中的所有点。如果是，则本次迭代中我们不从 $\mathcal{S}$ 中选择任何正方形（对应于 $v$）。注意，因此在对应于 $T$ 中 $v$ 的后代节点的迭代中，我们也不会选择任何正方形（所以不妨直接跳过以 $v$为根节点的整棵子树）。

现在假设 $v$ 的祖先节点中选出的正方形没有覆盖 $P \cap C_v$中的所有点。我们称这样的节点已被算法探索。设 $e$ 为 $C_v$的一条边。我们称包含 $e$ 的正方形是针对 $e$的边覆盖正方形。我们从 $\mathcal{S}$ 中选择一个针对 $e$ 的边覆盖正方形，使其与 $C_v$ 的交集在 $\mathcal{S}$ 中所有此类正方形里最大（我们称这样的正方形为针对边 $e$的 $C_v$ 的最大面积覆盖正方形）。若出现平局，则以任意但固定的方式打破，例如按照任意给定的 $\mathcal{S}$排序，选择索引最小的正方形。如果 $\mathcal{S}$ 中没有针对 $e$ 的边覆盖正方形，则不选择对应于 $e$的正方形。我们对 $C_v$的四条边分别执行此操作。见图3。如果到达叶节点，且存在一个未覆盖点（注意，该点必定位于单元的左下角），则选择任意一个覆盖该点的正方形（这样的正方形必定存在，因为 OPT 中有某个正方形覆盖该点）。见图4。

**引理1.** $\mathcal{A}_{off}$ 为 $P$中的点输出一个可行集合覆盖。

证明。 反设 ALG 中没有任何正方形覆盖某个点 $p \in P$。由于 OPT 是一个可行集合覆盖，因此存在一个正方形 $S \in$ OPT 覆盖了 $p$。这里有两种情况： $p$ 恰好位于 $S$的某个角点上，或者不在角点上。在后一种情况下，注意到 $S$ 至少对一个包含 $p$的四叉树单元边覆盖。令 $C_v$ 为这样的单元（它包含 $p$，且其边 $e$ 包含在 $S$中）中深度最小的单元。现在，算法将遍历 $T$，直至到达包含 $p$的节点 $v$ （对应于单元 $C_v$）。由于算法为 $v$ 的祖先节点选出的正方形没有覆盖 $p$，我们将在 ALG 中为 $e$选择最大面积覆盖正方形 $S'$。由于（$S \cap C_v$） $\subseteq$ （$S' \cap C_v$）， $S'$ 将覆盖 $p$。这就产生了矛盾。现在考虑第一种情况，即 $p$ 位于 $S \in$ OPT 的某个角点上：要么存在一个包含它的叶节点 $v \in T$，且 $S$ 对 $C_v$边覆盖；要么对于这样的叶节点 $v$， $S$ 进行角覆盖。在这两种情况下， $\mathcal{A}_{\text{off}}$ 都会为 $v$ 或其某个祖先节点选取一个正方形，使该正方形覆盖 $p$。 □

**近似比** 令 ALG $\subseteq \mathcal{S}$ 表示选取的正方形集合，OPT 表示最优解。为证明该 $O(\log\ N)$-近似保证，主要思路如下:

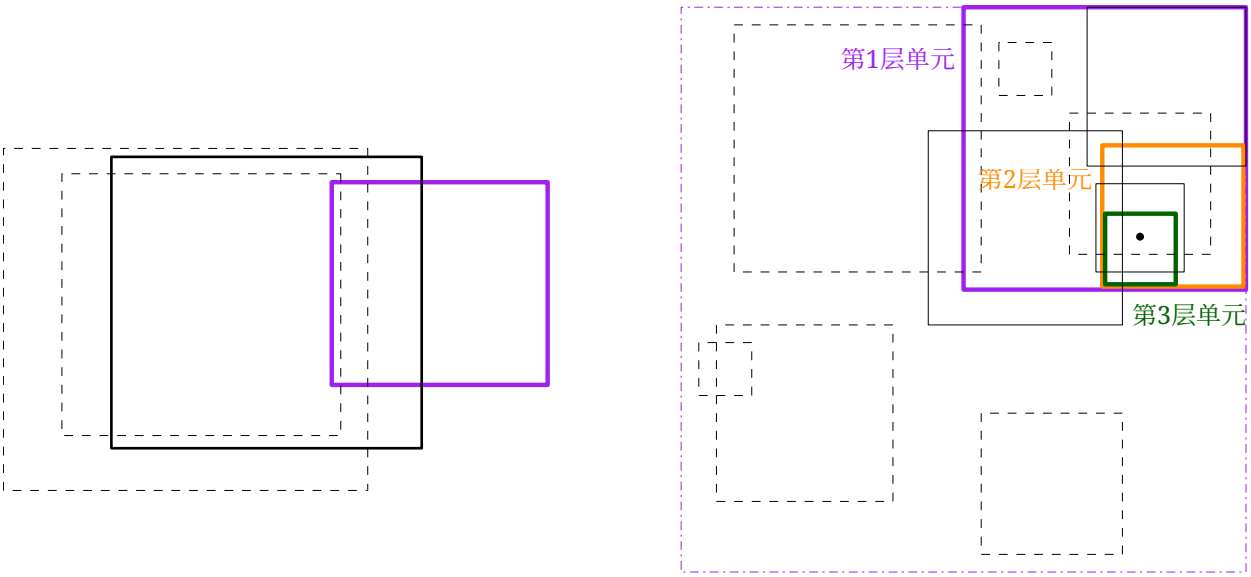


图3：左图展示了一个紫色的四叉树单元。选取了最大面积覆盖正方形（实线黑色），而没有选取其他边覆盖正方形（虚线）。右图展示了包含一个未覆盖点的四叉树单元（按层级以不同颜色标示）。按这些单元深度递增的顺序，每个单元至多同时选取4个最大面积覆盖正方形（实线黑色），直到该点被覆盖。

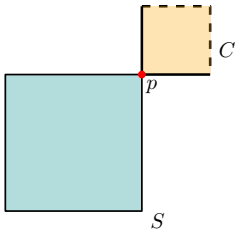


图4：点 $p$ 位于一个叶单元 $C$ 中（其中甚至可能没有任何边覆盖正方形）。在这种情况下，我们任意选取一个正方形 $S$ 来覆盖该点（因为这样的正方形总是存在）。

考虑一个节点 $v \in V$，并假设我们在与 $v$ 对应的迭代中至少选取了一个正方形。如果 $C_v$ 包含某个正方形 $S \in$ OPT 的一个角，那么我们就将为 $v$ 选取的（至多四个）正方形计入 $S$。否则，我们论证为 $v$ 选取的正方形所覆盖的 $C_v$ 至少与 OPT 中的正方形一样多，并且它们覆盖了 $P \cap C_v$ 中所有剩余的未覆盖点。特别地，我们不会在 $v$ 的后代节点中进一步选取正方形。为 $v$ 选取的正方形计入 $v$ 的父节点（该父节点包含某个正方形 $S \in$ OPT 的一个角）。由于每个正方形 $S \in$ OPT 的每个角都包含在 $O$（log $N$）个单元中，我们证明每个正方形 $S \in$ OPT 所承担的总计入量为 $O$（log $N$）。因此，我们得到以下引理。

**引理 2.** 我们有 $|\mathsf{ALG}| = O(\log N) \cdot |\mathsf{OPT}|$。

证明。 我们将 ALG 中选取的每个正方形计入 OPT 中的某个正方形。一个单元 $C_v$ 及其对应的节点 $v$，要么包含 OPT 中某个正方形的至少一个角，要么被 OPT 中至少一个正方形边覆盖，要么与 OPT 中的任何正方形都不相交。

- $C_v$ 包含 $S \in$ OPT 中的一个角点：在这种情况下，$\mathcal{A}$off 为该单元最多选取四个正方形，并将这些正方形计入该单元内的一个角点。如果 OPT 中有多个正方形的角点位于该单元内，则任意选取一个。即使 $C_v$ 对应的是叶节点，这一论断也成立。

- OPT 中的某个正方形 $S \in$ 是 $C_v$ 的边覆盖正方形（且 $C_v$ 中没有 OPT 中正方形的角点）：如果 $\mathcal{A}$off 没有为这样的单元选取边覆盖正方形，那么就没有问题。否则，如果 $\mathcal{A}$off 为这样的单元选取了正方形，我们声称它覆盖了该单元中的所有点。这是因为，由于 OPT 中不存在正方形的角点，该单元中的任意点都被 OPT 中一个针对 $C_v$ 的边覆盖正方形覆盖。因此，当 $\mathcal{A}$off 选取与该单元交集最大的边覆盖正方形时，任意 OPT 中的正方形 $S' \in$ 与 $C_v$ 的交集也会被覆盖）。于是，算法不会进一步探索 $v$ 的任何子节点。这也意味着，树中 $v$ 的父节点 $v'$ 将包含 $S$ 的一个角点（因为 $C_{v'}$ 与 $S$ 相交，但无法被其边覆盖）。我们将 $\mathcal{A}$off 在 $C_v$ 处选取的所有正方形（最多四个）计入父节点中的这个特定角点。如果有多个这样的角点，则任意选取一个。

- 没有来自 OPT 的正方形与 $C_v$相交：在这种情况下，$C_v$ 不包含 $P$中的任何点。因此，$\mathcal{A}$off 不会为这样的单元选取任何正方形。

现在我们注意到，OPT 中任意正方形的一个角点，至多落在四叉树的 $\log N$ 个单元中。对于这些单元中的每一个角点，在该单元选取的正方形中至多被计费四次，在它的四个子节点中的每个子节点中也至多被计费四次。因此，每个单元的每个角点总计至多被计费 20 次。所以，OPT 中的每个正方形至多被计费 20（每个角点、每个单元）$\times 4$（角点）$\times \log N$（单元）$= 80 \log N$次。因此，ALG 中至多有 $80 \log N \cdot |\mathsf{OPT}|$ 个正方形。

□

## 2.1 正方形的在线集合覆盖

### 2.1.1 $O(\log N)$-近似在线算法

我们希望将离线算法 $\mathcal{A}_{\text{off}}$ 转化为在线算法 $\mathcal{A}_{\text{on}}$，并假设对手在每轮 引入一个新点。关键在于，上述算法是单调的，也就是说，如果向 $P$中添加一个点，那么它输出的正方形集合就会包含此前（仅在 $P$上运行时）输出的 $\mathcal{S}$中的正方形。对于给定的点集 $P$，令 ALG($P$) $\subseteq \mathcal{S}$表示我们的（离线）算法输出的正方形集合。

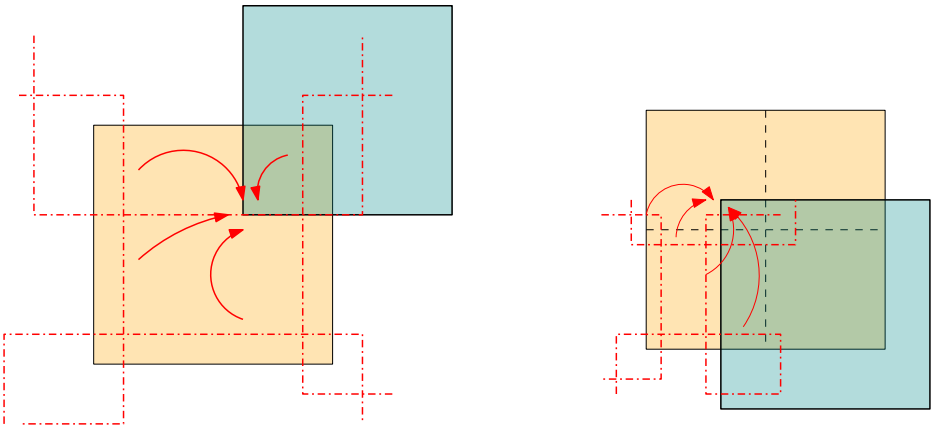

图 5：将选取的（红色）边覆盖正方形归因于 OPT 中（青色）正方形的角。左图中，（黄色）单元格包含 OPT 中某个正方形的一个角；右图中，该单元格的父节点包含这样的一个角。

**引理 3.** 考虑一个点集 $P$和一个点 $p$。那么 $\mathsf{ALG}(P) \subseteq \mathsf{ALG}(P \cup \{p\})$。

证明。 反设存在某个正方形 $S$ 属于 ALG($P$)，但不属于 ALG($P \cup \{p\}$)。根据 $\mathcal{A}_{\text{off}}$的描述，我们可以推断，算法在某次迭代中选取了 $S$，因为它是某个单元 $C_v$ 的最大面积覆盖正方形（该单元对应于 $T$中的节点 $v$），而该单元包含对手引入的一个点 $p' \in P$。此外，$\mathcal{A}_{\text{off}}$ 在运行过程中必定遍历了 $T$中 $v$ 的所有祖先节点。注意，算法只有在遍历包含 $p'$的单元时，才能在运行过程中覆盖任何这样的点 $p'$。这是因为，一旦我们选取了与四叉树中某个单元相关联的正方形，就只会计算这些正方形在该单元内部覆盖的面积。因此，如果 $\mathcal{A}_{\text{off}}$ 在这一步没有遍历 $C_v$，那么它也不会遍历 $T$中 $v$ 的子节点。于是，点 $p'$ 就不会被覆盖，这与假设矛盾。

□

因此，现在很容易导出一个用于正方形集合覆盖的在线算法。初始时，$P = \emptyset$。如果对手引入一个点 $p$，那么我们计算 ALG($P$)（其中 $P$ 表示之前的点集，即不包括 $p$）和 ALG($P \cup \{p\}$)，并将 ALG($P \cup \{p\}$)\ALG($P$) 中的正方形加入我们的解。因此，根据引理 2 和引理 3，我们得到一个 $O$(log $N$)-竞争性在线算法。

### 2.1.2 $O$(对数 $n$)近似的正方形在线集合覆盖

现在假设给定一个集合 $\tilde{P} \subseteq \mathbb{R}^2$，其中有 $|\tilde{P}| = n$，并且每轮都会将一个点从 $\tilde{P}$插入到 $P$，也就是说，每轮之后都有 $P \subseteq P$。在这种情况下，我们希望将竞争比控制在 $O$(log $n$)。如果 $N = n^{O(1)}$，则结论显然成立。否则，我们将算法扩展为使用 Arya 等人提出的平衡盒分解树（即 BBD 树）数据结构 [7]，而不是四叉树。第一轮之前，$P = \emptyset$，我们初始化 BBD 树，得到一棵树$\tilde{T} = (V, E)$，它具有以下性质：

- 每个节点 $v \in \tilde{V}$对应一个单元 $\tilde{C}_v \subseteq [0, N)^2$，该单元由一个外框 $b_O \subseteq [0, N)^2$ 和一个内框 $b_I \subseteq b_O$描述；二者都是轴平行矩形，且 $\tilde{C}_v = b_O \setminus b_I$（注意，$b_I$ 可能是空集）。

- $b_O$的纵横比，即 $b_O$最长边与最短边的长度之比，不超过 3。

- 若 $b_I \neq \emptyset$，则 $b_I$ 是粘性的，直观而言，这意味着在每个维度中，$b_I$ 到 $b_O$ 边界的距离要么为 0，要么至少为 $b_I$的宽度。形式化地，假设$b_O = [x(1)_O, x(2)_O] \times [y(1)_O, y(2)_O]$ 且 $b_I = [x(1)_I, x(2)_I] \times [y(1)_I, y(2)_I]$。那么 $x(1)_O = x(1)_I$或 $x(1)_I - x(1)_O \geq x(2)_I - x(1)_I$。此外，$x(2)_O = x(2)_I$或 $x(2)_O - x(2)_I \geq x(2)_I - x(1)_I$。对于$y$坐标也有类似条件。

- 每个节点 $v \in \tilde{V}$要么是叶节点，要么有两个子节点 $v_1$，$v_2 \in \tilde{V}$；在后一种情况下 $\tilde{C}_v = \tilde{C}_{v_1} \dot{\cup} \tilde{C}_{v_2}$。

- $\tilde{T}$的深度为 $O(\log\ n)$，且每个点 $q \in [0, N)^2$ 都包含在 $O(\log\ n)$个单元中。

- 每个叶节点 $v \in V$ 在 $P$中至多包含一个点。

在 BBD 树的构造中，我们令同一深度的单元彼此不相交，使得一个点$p$ 在某一深度上恰好包含于一个单元中。因此，对于单元$C_v = b_O \setminus b_I$，我们假设 $b_O$ 和 $b_I$ 都是闭集，即外框 $b_O$ 的边界属于该单元，而内框 $b_I$ 的边界不属于该单元。现在，我们描述对第2节中的离线算法所做的调整：使用 $T$ 而非 $T$。与之前类似，我们按广度优先顺序遍历 $T$。假设在某次迭代中，我们考察对应于单元$C_v$的节点 $v \in V$。我们检查 $v$ 的祖先节点中选出的正方形是否覆盖了 $P \cap C_v$中的所有点。如果是这样，则在本次迭代中，我们不会从 $\mathcal{S}$ 中为 $v$选取任何正方形。

现在假设 $v$ 的祖先中选出的正方形没有覆盖 $P \cap C_v$中的所有点。与第2节类似，我们希望为 $C_v$ 选择 $O(1)$ 个正方形，使得如果 $C_v$ 中不包含任何正方形 $S \in \text{OPT}$ 的角点，那么我们为 $C_v$ 选择的正方形应覆盖 $P \cap C_v$ 中的所有点。与之前类似，对于 $b_O$ 的每条边 $e$，我们从 $\mathcal{S}$ 中选择一个包含 $e$ 且与 $b_O$ 的交集在 $\mathcal{S}$ 所有此类正方形中最大的正方形。若出现平局，则以任意但固定的方式打破平局。然而，由于 $C_v$ 可能不是正方形且可能存在孔洞（由 $b_I$ 导致），除了边覆盖正方形外，我们还需要考虑 OPT 中与 $C_v$ 有非空重叠的两类额外正方形：(a) 穿过 $C_v$，即与 $b_O$ 的两条平行边相交的正方形；(b) 有一个或两个角点位于 $b_I$ 内部的正方形。

以下贪心子程序 $\mathcal{G}$ 将在我们的算法中用于处理此类棘手情况。设 $R$ 为一个宽度为 $w$、高度为 $h$ 的盒，满足 $w/h \leq B$，其中 $B \in \mathbb{N}$为某个常数；设$P_R$ 为 $R$ 内的一个点集，可以用一组垂直穿越（即同时与 $R$的两条水平边相交）的正方形 $\mathcal{S}'$覆盖。那么，根据 $\mathcal{G}$选出的正方形将以下列方式覆盖 $P_R$：

- 只要存在未覆盖点 $p' \in P_R$:
  - 考虑最左侧的此类未覆盖点 $p \in P_R$。
  - 选择与 $p$ 相交且最右侧边最靠右的垂直穿越正方形（根据假设，这样的正方形存在）。

（上述子程序用于寻找垂直穿越正方形。要寻找水平穿越正方形，我们可以将输入 $90°$ 适当逆时针旋转，然后应用同一子程序。）接下来，我们给出关于上述子程序的以下结论。

**命题 1.** 设 $R$ 是一个宽度为 $w$、高度为 $h$ 的盒，满足 $w/h \leq B$，其中 $B \in \mathbb{N}$为某个常数；且 $P_R$ 是盒 $R$ 内的一个点集，可由一组垂直贯穿的（即它们

同时与盒 $R$的两条水平边相交）正方形 $\mathcal{S}'$覆盖。那么，我们可以从 $\mathcal{S}'$ 中找出至多 $B+1$ 个正方形，覆盖盒 $R$内的所有点。

当 $h/w \le B$ 时，我们对水平穿越正方形也有一个类似的命题。

证明。 考虑贪心子程序 $\mathcal{G}$ 的每次迭代。我们将尚未被截至目前 $\mathcal{G}$ 选出的正方形覆盖的最左侧点 $p \in P_R$ 称为枢轴点。那么，对于 $R$，所有选出的垂直穿越正方形都恰好包含一个在算法执行过程中某个时刻被识别为枢轴点的点。由于纵横比有界于 $B$，且这些正方形是垂直穿越正方形（即其垂直长度大于 $R$ 的垂直长度），因此枢轴点至多有 $B+1$ 个。因此，由于 $R$，我们至多选出 $B+1$ 个穿越正方形。由此得到一个可行集合覆盖。

□

现在我们描述我们的算法。首先，处理可能穿越 $b_O$的正方形。因此，我们在 $b_O$ 上应用贪心子程序 $\mathcal{G}$。由于 $b_O$ 的纵横比有界且为 3，根据命题 1，我们得到至多 $(3+1)+(1+1) = 6$ 个可能垂直或水平穿越 $C_v$ 的正方形。如果 $b_I = \emptyset$，我们便不再选取其他正方形。否则，我们需要处理在 $b_I$ 内可能有一个或两个角点的正方形。设 $\ell_1$、$\ell_2$、$\ell_3$、$\ell_4$ 分别表示包含$b_I$四条边的四条直线。注意，$\ell_1$、$\ell_2$、$\ell_3$、$\ell_4$ 将 $b_O$ 划分为至多九个矩形区域，其中一个与 $b_I$ 完全相同。对于每个这样的矩形区域 $R$，如果它与 $b_I$ 共享一条水平边，我们再次使用$\mathcal{G}$ 来选取垂直穿越正方形。否则，如果 $R$ 与 $b_I$ 共享一条垂直边，我们适当地使用子程序 $\mathcal{G}$ 来选取水平穿越正方形。这样就处理了在 $b_I$ 内有两个角点的正方形。否则，如果矩形区域 $R$ 与$b_I$不共享任何边，我们就检查是否存在一个正方形 $S \in \mathcal{S}$，其有一个角点位于 $b_I$ 内且完全包含 $R$。我们也将 $S$ 加入解中。至此，便处理了正方形只有一个角点位于 $b_I$ 内的情况。

最后，为了完成我们的算法，在执行之前，我们进行如下操作：对于每个至多包含一个点 $p \in P$ 的叶节点 $v$ for˜which $C_v$，我们为其关联一个覆盖 $p$ 的固定正方形。然后，如果算法在遍历过程中到达某个叶节点 $v$，且其中存在一个未覆盖点 $p$，我们就选取与该叶节点关联且覆盖该点的正方形。算法中的这一条件保证了可行性。

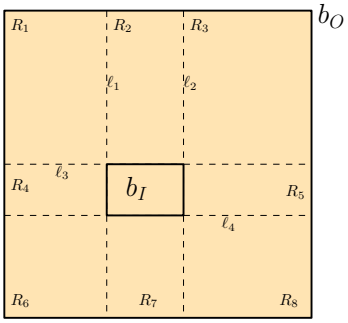


图 6：由于内框 $b_I$，外框 $b_O$ 被划分为至多 9 个矩形。

˜**引理 4.** 设 $C_v$ 是一个单元，使得在 $v$ 的祖先节点中选取的正方形 不 覆盖 $P \cap C_v$ 中的所有点˜。那么

˜*(a)* 我们为 $C_v$ 至多选取 $O(1)$ 个正方形，并且

˜ ˜*(b)* 如果 $C_v$ 不包含正方形 $S \in$ OPT的任何角，那么为 $C_v$ 选择的正方形会覆盖 $P \cap C_v$中的所有点˜。

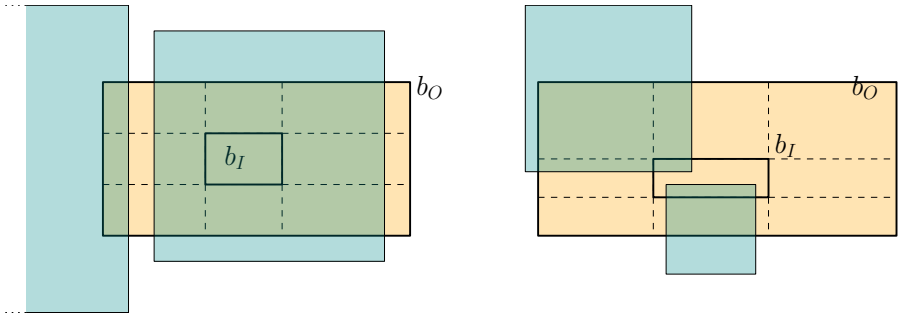


图7：OPT 中的一个（青色）正方形与一个单元的可能交集，且该正方形没有角点位于该单元内。左图展示了边覆盖和穿越正方形。右图展示了有两个角点之一位于 $b_I$ 内的正方形。

˜证明。 首先，我们证明部分（a）。如果 $C_v$ 对应一个叶节点，我们至多选择一个正方形。˜否则，我们为$C_v$ 至多选择 4 个边覆盖正方形。根据命题 1，我们为 $C_v$ 选择 $b_O$中水平或垂直交叉的 $O(1)$个˜正方形。如果 $b_I = \emptyset$，则不再选择更多正方形。

因此，考虑另一种情况： $b_I \neq \emptyset$。令 $R$ 为将 $b_O$ 按照 $\ell_1$、$\ell_2$、$\ell_3$、$\ell_4$ 划分后得到的（至多）四个矩形区域之一，这些区域与 $b_I$ 共用一条边。令 $w$、$h$ 分别为 $R$ 的宽度和高度。不失一般性，假设 $R$ 与 $b_I$ 共用一条水平边。由于 $b_I$ 具有粘性，且 $b_I$ 和 $b_O$ 的纵横比有界，为 3，可以看出 $R$ 也具有一个 $w/h \leq 3$ （类似地，如果 $R$ 与 $b_I$ 共用一条垂直边，则 $h/w \leq 3$）。再次利用命题 1，我们为 $R$ 选取 $O(1)$ 个垂直穿越正方形。对其他此类区域执行类似操作。现在考虑将 $b_O$ 按照 $\ell_1$、$\ell_2$、$\ell_3$、$\ell_4$ 划分后得到的其余（至多四个）区域，这些区域不与 $b_I$ 共用边。我们为每个区域至多选取一个正方形。因此，总共为 $C_v$ 选取至多 ˜ $O(1)$ 个正方形。

˜现在我们证明部分（b）。如果 $C_v$ 不包含正方形 $S \in$ OPT的任何角点，那么
˜ ˜OPT中所有与 $C_v$ 相交的正方形，要么是边覆盖正方形 $b_O$，要么是穿越正方形 $C_v$，要么在 $b_I$内包含一个或两个角点。然而，由于我们为 $b_O$选取了极大的边覆盖正方形，它们包含了 $C_v$ 中所有被 OPT 中的边覆盖正方形覆盖的˜点。

类似地，根据我们在贪心子程序 $\mathcal{G}$中选取的正方形（它们垂直或水平穿越 $b_O$），我们覆盖了 OPT 中所有与 $b_O$相交的这类穿越正方形所能覆盖的点。

对于至少有一个角点位于 $b_I$内的正方形，注意到它们必定穿过由划分 $b_O$ 得到、且与 $b_I$共边的某个矩形区域。事实上，对于这样一个有角点位于 $b_I$内的正方形 $S$，在这四个矩形区域中，存在一个矩形区域 $R$ （设其宽度为 $w$ 、高度为 $h$），使得要么 $S$ 是相对于 $R$ 垂直穿越的，并且 $R$与 $b_I$ 共用一条水平边（此时 $w/h \leq 3$），要么 $S$ 是相对于 $R$ 水平穿越的，并且 $R$ 与 $b_I$ 共用一条垂直边（此时 $h/w \leq 3$）。于是，利用命题 1，我们覆盖了这类正方形所覆盖的所有点。此外，一个恰好有一个角点位于 $b_I$ 内的正方形，可能完全包含由划分 $b_O$ 得到的另一个矩形区域（该区域与 $b_I$不共边）。对于这类情况，如果存在这样的正方形，我们同样通过选取一个正方形将其覆盖。

□

因此，我们可以建立与第2节类似的收费方案。为了支付我们的解，我们对 OPT 正方形的每个角 $q$ 最多收费 $O(\log n)$ 次。因此，我们的近似比为 $O(\log n)$。与第2节类似，我们可以将上述离线算法修改为在线算法，其近似比为 $O(\log n)$。

**定理 1.** 对于任意大小的轴平行正方形集合覆盖，存在一个确定性的 $O(\log n)$-竞争在线算法。

## 2.2 下界

一个自然的问题是，对于正方形在线集合覆盖，能否设计出竞争因子优于 $O(\log n)$ 的算法。我们的答案是否定的，即使在单位正方形的情形下，甚至对于随机化算法也是如此。我们还指出，区间集合覆盖存在一个紧确的 2-竞争在线算法（见附录 A）。

### 2.2.1 单位正方形与象限

给定一个集合覆盖实例 $(P, \mathcal{F})$，对每个 $F \in \mathcal{F}$ 定义一个变量 $x_F$，其取值为 $\in [0,1]$。对于一个点 $p \in P$，令 $\mathcal{F}_p \subseteq \mathcal{F}$ 表示覆盖它的集合。在 分数集合覆盖问题中，目标是为变量 $x_F$ 赋值，使得对于所有点 $p \in P$，$\sum_{F \in \mathcal{F}_p} x_F \geq 1$。

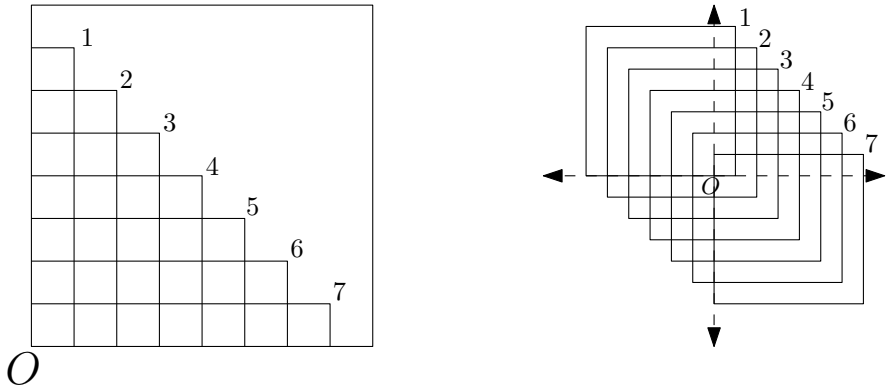


图 8：针对象限和单位正方形的 $\Omega(\log m)$ 困难实例

**引理 5**。 存在一个关于 $m$ 象限的在线分数覆盖问题实例（象限是一个以其某个角点为原点的矩形），使得任何在线确定性算法在该实例上的竞争比均为$\Omega(\log m)$。

证明。考虑一组 $m$ 象限（见图 8），其右上角如下所示：$Q_1 = (1, m)$ $Q_2 = (2, m-1) \ldots, Q_k = (k, m-k+1) \ldots, Q_m = (m, 1)$（并令集合覆盖实例中对应的变量依次为 $x_1, x_2, \ldots, x_m$）。现在考虑点 $p_{i,j} = (i - 0.5, m - j + 0.5)$。我们声称该点恰好与索引从 $i^{th}$ 到 $j^{th}$ 的象限相交。由于 $p(x) > i - 1$, $p_{i,j}$ 不与前面的 $(i-1)$ 个象限中的任何一个相交。此外，由于$p(y) = m - j + 0.5 < m - i - 1$，它确实与象限 $i$ 相交；这一结论一直成立到象限$j$（因为，$p(y) = m - j + 0.5 < m - i - 1$）。另外，$p_{i,j}$ 不与最后 $m - j + 1$ 个象限中的任何一个相交，因为 $p(y) > m - j$。

现在考虑一个对手按如下方式引入点：$P_1 = p_{1,\ m}$。如果算法为变量赋值，使得 $\sum_{i=1}^{m/2} x_i \geq \sum_{i=m/2+1}^{i=m} x_i$，则给出点 $P_2 = p_{m/2+1,\ m}$；否则给出点 $P_2 = p_{1,\ m/2}$。对手重复这一过程，将相交的象限集合不断减半，并将下一个点放在总和较小的范围内，直到只剩下一个象限，即象限 $j$。

最优解本应只将 $x_j$ 赋值为 1，其余变量赋值为 0。但任何在线算法在每一步都只能将潜在最优解的集合缩小一半，

同时至少要给非最优象限赋予 $1/2$ “代价”。因此，任何在线算法的代价都是 $\Omega(\log m)$。

□

**推论 1.** 存在一个关于 $m$ 单位正方形的在线分数覆盖问题实例，使得任何确定性在线算法在此实例上的竞争比为 $\Omega(\log m)$。

证明. 我们可以沿左下方向适当延伸引理 5 中的象限，从而得到一个类似的、关于边长为 $m$ 的正方形的实例。更准确地说，设与象限 $i$ 对应的正方形 $i$ 的左下角为 $(i-m, 1-i)$。对手引入的点与象限实例中的点相同。

现在，将这个关于正方形的实例适当缩小 $m$ 倍，得到所需的单位正方形实例。

□

利用标准技术，如 [15]，所示，我们可以将分数变体中确定性算法的下界推广为整数变体中随机化算法的下界。

**推论 2.** 存在一个关于 $m$ 单位正方形的在线（整数）集合覆盖问题实例，使得任何随机化在线算法在此实例上的竞争比为 $\Omega(\log m)$。

由于在我们的下界构造中， $n = \Theta(m^2)$、$\log n = \Theta(\log\ m)$，因此我们得到如下定理。

**定理 2.** 对于单位正方形的集合覆盖，任何确定性或随机化在线算法的竞争比均为$\Omega(\log n)$，即使所有正方形都包含原点，且所有点都位于同一象限内。

## 3 正方形的在线击中集

我们提出用于正方形击中集问题的在线算法。假设给定一个具有整数坐标的固定点集 $P \subseteq [0, N)^2$。我们维护一个已选点集 $P'$，初始时 $P' := \emptyset$。每一轮中，我们会得到一个正方形 $S \subseteq [0, N)^2$，其角点的坐标均为整数。

不失一般性，假设 $N$ 是 2 的幂。令 $Q$ 为$[0, N)^2$,中所有坐标为整数的点，即 $P \subseteq Q$。对于每个点 $q \in Q$，我们称 $q = (q_x, q_y)$ 的层级为 $\ell$，如果 $q_x$和$q_y$都是 $N/2^\ell$的整数倍。我们构建与第2节中相同的四叉树。若单元 $C_v$ 的高度和宽度均等于 $N/2^\ell$，则称其层级为 $\ell$ 。

现在给出我们的算法。假设在某一轮中给定了一个新的正方形 $S$。如果 $S \cap P' \neq \emptyset$，那么我们不会向 $P'$ 中添加任何点。现在假设 $S \cap P' = \emptyset$。令 $q$ 为 $Q \cap S$ 中层级最低的点（如果有多个这样的点，则任意选取 $Q \cap S$ 中层级最低的一个点）。直观地说，我们将 $q$ 视为原点，并把平面划分为四个象限。我们定义 $O_{TR} := \{(p_x, p_y) \mid p_x \geq q_x,\ p_y \geq q_y\}$，并定义 $S_{TR} := O_{TR} \cap S$；类似地定义 $O_{TL}$、$O_{BR}$、$O_{BL}$以及 $S_{TL}$、$S_{BR}$、$S_{BL}$。考虑 $O_{TR}$和 $S_{TR}$。对于每个层级$\ell = 0, 1, \ldots, \log N$，我们执行以下操作。按照某个固定顺序，依次考虑层级为 $\ell$的每个单元 $C$，使得 $C \subseteq O_{TR}$，并且 $S_{TR}$ 是 $C$的一条边 $e$ 的边覆盖。接着，对每条边，找出 $P \cap C$中距其底边（顶边、左边和右边）最近的点 $p_b$（分别为 $p_t$、$p_l$、$p_r$）。如果 $p_b$、$p_t$、$p_l$、$p_r$中至少一个包含在 $S_{TR}$中，我们就将这些点（至多 4 个）加入解中（见图9）。如果以这种方式将该单元 $C$ 的至少一个此类点 $p$ 加入 $P'$，我们就称 $C$ 被激活。注意，即使 $p_b$、$p_t$、$p_l$、$p_r$中只有一个可能被包含在其中，我们仍可能将它们全部加入 $P'$，尽管可能只有一个

包含在 $S_{TR}$中。这是为了确保在在线算法的一次运行中， $C$ 至多被激活一次。我们将在断言2中证明这一点，而这最终将有助于证明我们的算法具有 $O(\log\ N)$-竞争比。如果在当前层级 $\ell$，我们激活了至少一个层级为 $\ell$的单元 $C$，那么就停止循环，不再考虑其他层级 $\ell+1$、 . . . 、$\log\ N$。否则，我们继续处理层级 $\ell+1$。我们对点对（$O_{TL}$、 $S_{TL}$）、（$O_{BR}$、 $S_{BR}$）和（$O_{BL}$、 $S_{BL}$）执行对称操作。现在我们证明该算法的正确性，并证明其竞争比为 $O(\log\ N)$。

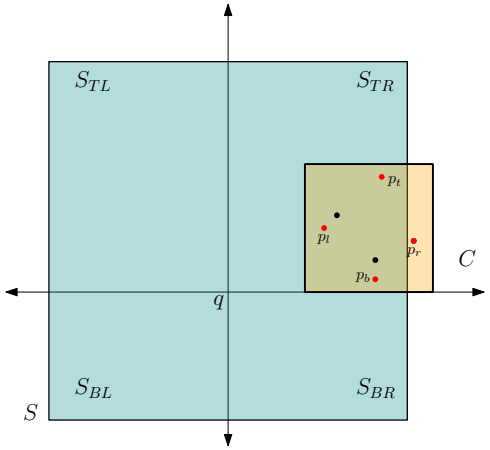


图9：位于 $O_{\text{TR}}$ 中的单元 $C$ 里，红色点由算法选取。

**引理6。** 每轮结束后，集合 $P'$ 是迄今已添加的正方形的击中集。

证明。 我们将证明，在 $S\cap P'=\emptyset$ 的情况下插入正方形 $S$ 时，至少有一个单元 $C$ 会被激活。注意，存在一个最优击中集中的点 $p$，使得 $p\in S$。不失一般性，假设 $p$ 属于 $S_{TR}$。接下来，考虑包含点 $p$的单元格集合 $\mathcal{C}'_p$ 。由于$S$ 的边长至少为1（其角点的坐标为整数），因此存在一个单元 $C'\in\mathcal{C}'_p$，使得 $S_{TR}$覆盖 $C'$的一条边 $e$。因此，在 $\{0,1,...,$中存在一个层级 $\ell$，满足$\log N\}$，使得存在一个层级为 $\ell$ 的单元 $C''\subseteq O_{TR}$ ，其边 $S$ （例如底边 $e'$）被覆盖，且$p\in C''\cap S$。于是，我们的算法为 $C''$选取了 $p_b$ （距底边最近的点），使得$p_b\in C''\cap S$。

□

现在我们证明，每一轮都会向 $P'$ 中添加 $O(1)$ 个点。

**引理7.** 每一轮我们都会向 $P'$ 中添加 $O(1)$ 个点。

证明。 我们证明，对于给定的正方形 $S$，使得 $S\cap P'=\emptyset$ （$P'$ 是我们的算法维护的击中集），我们的算法至多激活 4 个单元。为此，只需注意，在四个象限 $O_{TR},O_{BR},O_{TL},O_{BL}$ 中，我们至多各激活 1 个单元。对于这些单元中的每一个，我们至多选取 4 个点，因此任意一轮中我们至多添加 16 个点。

□

将插入正方形的最后一轮之后的最优解记为 OPT。

**引理 8.** 设 $p\in$OPT。那么存在 $O(\log\ N)$ 轮，其中插入了满足 $p\in S$ 的正方形 $S$，且在该轮开始时 $P'\cap S=\emptyset$。

证明。 首先，将层级为 $\ell$ 的两个单元之间的水平距离定义为其左边缘的$x$坐标之间的距离。类似地，定义垂直距离。现在，我们为点 $p$定义一组单元 $\mathcal{C}_p$ ，并将其初始化为 $\emptyset$。稍后我们将证明，在某个轮次中，如果

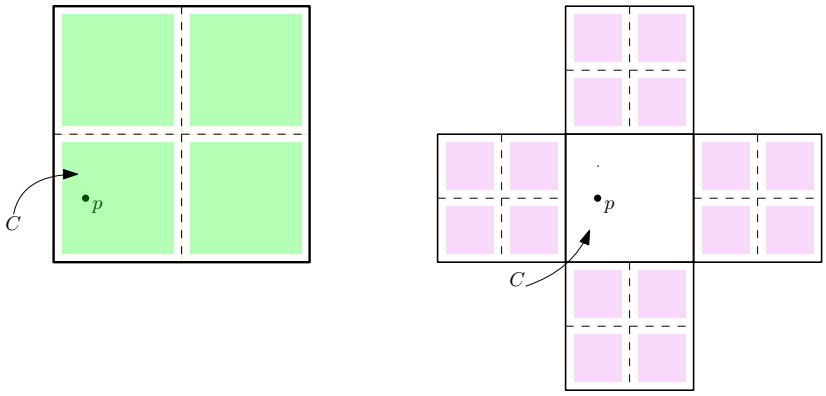


图10：单元格 $C$ 包含点 $p$。主要单元格以绿色突出显示，次要单元格以粉色突出显示。

对手引入的正方形 $S$ 包含 $p$，且我们的算法至少激活一个单元，那么 $\mathcal{C}_p$ 中的一个单元也会被激活。对于每个层级 $\ell \in \{0, 1, \ldots, \log\ N\}$，将以下单元纳入集合 $\mathcal{C}_p$：包含 $p$ 的层级为 $\ell$ 的单元$C$，以及层级为 $\ell$ 、与$C$拥有相同父节点的其他单元（如果存在）。我们称这些单元为主要单元。此外，对于层级 $\ell \in \{0, 1, \ldots, \log\ N\}$，考虑包含 $p$的层级为 $\ell$ 的单元 $C$。接着，考虑所有与其水平距离为 $N/2^\ell$、垂直距离为0的层级为 $\ell$ 的单元；同时，也考虑所有与 $C$垂直距离为 $N/2^\ell$、水平距离为0的单元。这样的单元至多有4个。对于每个这样的单元，将其所有子节点纳入 $\mathcal{C}_p$（每个单元有4个子节点）。我们称这些单元为次要单元。因此，每个层级至多选取 $4 + 4 \times 4 = 20$ 个单元。因此，$|\mathcal{C}_p| = O(\log\ N)$。

我们首先注意到，一旦一个单元 $\hat{C}$ 被激活后，就不会再次被激活。

**断言2。** 我们的算法不会使单元格 $C \in T$ 被激活超过一次。

**断言3。** 反设 $C$ 已在之前某一轮中因正方形 $S'$而被激活。在当前轮次，对手引入了正方形 $S$，使得 $S \cap P' = \emptyset$。如果 $C$ 再次被激活，那么 $S$ 覆盖了 $C$ 的一条边（不失一般性，假设这是底边）。如果 $C$ 已在之前某一轮中被激活，那么算法必定选择了分别最靠近 $C$ 底边、顶边、左边和右边的点，并且至少有一个点命中了该正方形。在这些点中，将最靠近底边的点记为 $p'$。如果$C$ 因 $S$再次被激活，根据定义，它显然至少包含一个命中 $S$ 的点。于是，$p'$ 就会命中 $S$，因为它在$C$中的 $y$ 坐标最低。

现在我们要证明，若正方形 $S$在某一轮中被插入，且满足 $S \cap P' = \emptyset$但不满足 $p \in S_{TR}$，那么 $\mathcal{C}_p$ 中的一个单元会被激活，而 $C \notin \mathcal{C}_p$ 中满足 $C \subseteq O_{TR}$ 的单元则不会被激活。证明这一点后，注意到在每一轮这样的轮次中，我们都会激活 $\mathcal{C}_p$中的一个单元。然后，利用上述结论——在这样的轮次中，$\mathcal{C}_p$ 中不会有单元再次被激活——我们可以保证，经过 $|\mathcal{C}_p|$ 轮后，对手引入的每个包含 $p$ 的正方形 $S'$，都至少被算法维护的击中集中的一个点击中。

对于 $p \in S_{TL}$、$p \in S_{BR}$和 $p \in S_{BL}$这些情况，我们也可以进行对称论证。每种情况都得出如下结论：若添加正方形 $S$时满足 $S \cap P' = \emptyset$但不满足 $p \in S$，那么 $\mathcal{C}_p$中的某个单元会被激活。因此，这样的轮次最多只有 $|\mathcal{C}_p| = O(\log\ N)$轮。由此，最后只剩下证明该结论。

**断言4.** 如果在某一轮插入正方形 $S$，且 $S \cap P' = \emptyset$，但 $p \in S_{TR}$，那么 $\mathcal{C}_p$ 中的一个单元会被激活，但不会有满足 $C \subseteq O_{TR}$ 的单元 $C \notin \mathcal{C}_p$ 被激活。

**断言 5.** 将 $Q \cap S$ 中层级最低的点之一 $q$ 的层级记为 $\ell'$。反设单元 $\hat{} C \notin \mathcal{C}_p$（满足 $\hat{} C \subseteq O_{TR}$）被激活，并且不失一般性地， $S$ 覆盖了它的底边。令 $\hat{}C$ 的层级为 $\ell$。令 $C$ 为包含 $p$ 的层级为 $\ell$ 的单元，并令 $C'$ 为其父节点（其层级为 $\ell - 1$）。根据构造，我们知道 $\hat{} C \cap C' = \emptyset$，因此 $\hat{}C$ 的父节点不是 $C'$。于是，$\hat{}C$ 的父节点（记为 $C''$）与 $C'$ 是层级为 $\ell - 1$ 的单元，二者在水平方向或垂直方向上的距离至少为 $N/2^{\ell-1}$。不失一般性，假设 $C'$ 位于 $C''$ 的右侧。根据我们的假设，$\hat{}C$ 的右边不在 $C'$ 的左边右侧（两者可能重合）。$\hat{}C$ 的任一角点都是层级至多为 $\ell$ 的点。因此，我们知道 $q$ 的层级（它是 $\ell'$）至多为 $\ell$。现在分两种情况。

第一种情况下， $q$ 与 $C'$ 的左边之间的水平距离至少为 $N/2^{\ell-1}$ （见图 ***11(a)***）。那么 $\ell' \leq \ell - 1$，因为此时 $S_{TR}$ 覆盖了 $C'$的左下角。在这种情况下，根据我们的假设， $S_{TR}$ 覆盖了 $C''$ 的底边，而底边上也至少有一个点可以击中它。这与本轮次激活单元的层级矛盾，因为 $C''$ 的层级为 $\ell - 1$。

另一种情况下， $q$ 与 $C'$ 的左边之间的水平距离严格小于$N/2^{\ell-1}$ （见图 ***11(b)***）。在这种情况下， $q$ 的层级仍恰好为 $\ell' = \ell - 1$。于是，$\hat{}C$ 的右边与 $C'$的左边重合。因此， $C''$ 位于 $C'$ 左侧，距离恰好为 $N/2^{\ell-1}$，而$\hat{}$本应作为次级单元添加。因此， $C \in \mathcal{C}_p$。这是一个矛盾。

这就完成了该引理的证明。 □

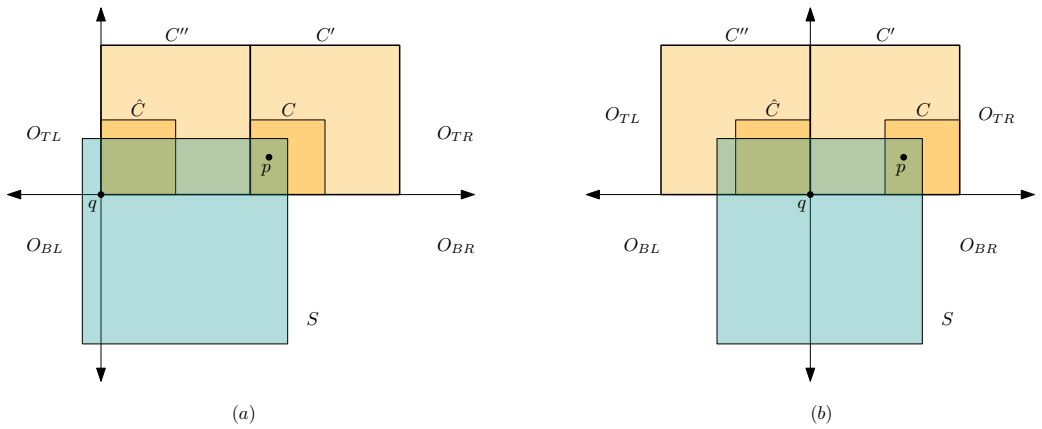


图11：断言4的证明。(a) 要么 $C''$ 与 $C'$ 之间的距离很大，此时 $S$ 是 $C''$的边覆盖；要么 (b) $\hat{} C \in \mathcal{C}_p$。

因此，引理7和引理 8表明，对于 OPT 中的每个点 $p \in$，我们向$P'$中添加 $O(\log\ N)$个点。因此，我们的竞争比为 $O(\log\ N)$。

**定理 3。** 存在一种 $O(\log\ N)$-竞争的确定性在线算法，用于求解任意大小轴对齐正方形的击中集问题。

这是紧的，因为即使对于区间，Even 等人 [23] 也已证明存在 Ω(log $N$) 的下界。

## 4 $d$维超矩形的动态集合覆盖

在本节中，我们将设计一种算法，动态维护$d$维超矩形的近似集合覆盖。本节证明的主要结果如下。

**定理 4.** 经过耗时 $O(m\log^{2d} m)$ 的预处理步骤后，存在一种适用于 $d$维超矩形动态集合覆盖的算法，其近似因子为$O(\log^{4d-1} m)$，更新时间为 $O(\log^{2d+2} m)$。

我们的目标是改造本文前几节中设计的基于四叉树的算法。为此，我们首先对问题进行变换，使 $\mathbb{R}^d$ 中的点和超矩形映射为 $\mathbb{R}^{2d}$ 中的点和 超立方体，从而得到一个新问题：用这些超立方体覆盖 $\mathbb{R}^{2d}$ 中的点。正如引言中所述，在这些超立方体上构造一个简单的 $2d$ 维四叉树无法满足我们的需求。我们从两个方面对四叉树进行扩展：(a) 在每个节点处，收集相对于该节点构成边覆盖的超立方体，并“忽略”它们构成边覆盖的那个维度；(b) 基于剩余的 $2d-1$ 个维度，在这些超立方体上递归构造一个 $(2d-1)$ 维四叉树。我们将这一新结构称为扩展四叉树。由此得到的一个优良性质是，$\mathbb{R}^{2d}$ 中的任意点只会属于扩展四叉树中的 $O$ 个单元（$\log^{2d} m$）。此外，在扩展四叉树的一维单元处，我们会为每个单元确定 $O$ 个(1)“最有用”的超立方体。这确保任意点只属于 $O$ 个 $(\log^{2d} m)$ 最有用的超立方体。因此，我们可以利用这些最有用的超立方体构造一个“有界频率”集合系统。Bhattacharya 等人 [13]（针对一般集合覆盖）的动态算法能够高效处理有界频率集合系统，将其应用于我们的场景即可得到一个 $O(\log^{4d-1} m)$ 近似算法。

## 4.1 向超立方体的变换（$\mathbb{R}^{2d}$ 中）。

回忆一下，输入是一个点集 $P$，而 $\mathcal{S}$ 是 $\mathbb{R}^d$ 中的一个超矩形集合。算法的第一步是将 $\mathcal{S}$ 中的超矩形变换为 $\mathbb{R}^{2d}$ 中的超立方体。考虑一个超矩形 $S \in \mathcal{S}$，其 $a = (a_1, \ldots, a_d)$ 和 $b = (b_1, \ldots, b_d)$ 分别是 $S$ 的“左下”角点和“右上”角点。令 $\Delta = \max_{j=1}^{d}(b_j - a_j)$。于是，$S$ 被变换为 $\mathbb{R}^{2d}$ 中的一个超立方体 $S'$，其边长为 $\Delta$，且“右上”角点为 $(-a_1, -a_2, \ldots, -a_d, b_1, b_2, \ldots, b_d)$。令 $\mathcal{S}'$ 为这些 $m$ 变换后的超立方体的集合。令 $P'$ 为 $\mathbb{R}^{2d}$ 中的点集 $n$，它是通过将每个点 $p = (p_1, \ldots, p_d) \in P$ 变换为 $p' = (-p_1, \ldots, -p_d, p_1, \ldots, p_d)$ 得到的。

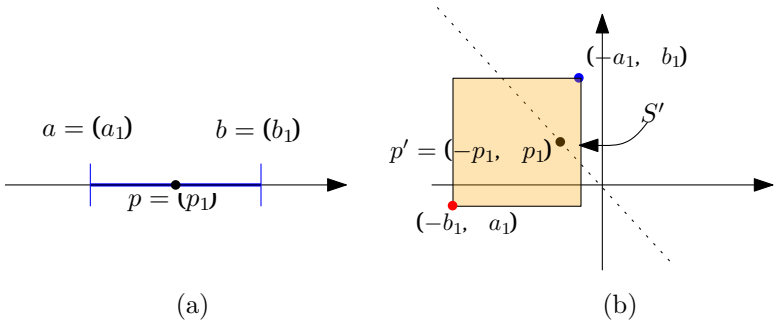


图12：（a）位于区间 $S = [a_1, b_1]$ 内的一维点 $p$，以及（b）将 $p$变换为点 $p' = (-p_1,\ p_1)$，并将 $S$ 变换为二维正方形 $S'$。

**观察1。** 当且仅当点 $p = (p_1, \ldots, p_d)$ 位于 $S$ 内时，$p' = (-p_1, \ldots, -p_d, p_1, \ldots, p_d)$位于 $S'$内。

证明。 假设 $p$ 位于 $S$内部。考虑满足 $i \le d$的 $p'$ 的第 $i$个坐标。由于 $b_i - a_i \le \Delta$意味着 $-a_i - \Delta \le -b_i$，我们观察到 $-a_i - \Delta \le -b_i \le -p_i \le -a_i$。因此，对于所有$1 \le i \le d$，都有 $-a_i - \Delta \le -p_i \le -a_i$。

现在考虑满足 $i > d$的 $p'$ 的第 $i$个坐标。由于 $b_i - a_i \le \Delta$ 意味着 $b_i - \Delta \le a_i$，我们观察到 $b_i - \Delta \le a_i \le p_i \le b_i$。因此，对于所有 $d+1 \le i \le 2d$，都有 $b_i - \Delta \le p_i \le b_i$。由此，我们断言，如果点 $p$ 位于 $S$内部，那么 $p'$ 将位于 $S'$内部。

证明另一个方向很容易。如果 $p$ 位于 $S$之外，那么至少存在一个坐标（设为 $i$），使得 $p_i < a_i$ 或 $p_i > b_i$。如果 $p_i < a_i$，那么 $-p_i > -a_i$，因此 $p'$ 位于 $S'$之外。另一方面，如果 $p_i > b_i$，那么同样有 $p'$ 位于 $S'$之外。 □

通过标准秩空间归约，我们可以假设 $\mathcal{S}$中的超矩形的角点位于网格 $[0, 2m]^d$上。应用上述变换后，我们注意到，$\mathcal{S}'$ 中的超立方体角点的坐标将位于网格 $[-4m, 0]^d \times [-2m, 2m]^d$上：显然，$\Delta + a_i \le 4m$，因此 $-4m \le -a_i - \Delta \le -a_i \le 0$。此外，$-2m \le b_i - \Delta \le b_i \le 2m$。对网格进行适当平移后，我们将假设 $\mathcal{S}'$ 中的所有超立方体角点都位于网格 $[0, 4m]^{2d}$上。

### 4.2 构造有界频率集合系统。

现在我们将介绍一种选取集合$\hat{\mathcal{S}} \subseteq \mathcal{S}'$ 的技术，使其具有以下性质：

1. (有界频率) $P'$ 中的任意点都位于$\hat{\mathcal{S}}$中的 $O(\log^{2d} m)$个超立方体内。 2. 对$(P', \hat{\mathcal{S}})$的 $\alpha$-近似动态集合覆盖算法，可推出对$(P', \mathcal{S}')$的 $O(\alpha \log^{2d-1} m)$-近似动态集合覆盖算法。 3. 更新集合系统$(P', \hat{\mathcal{S}})$的解所需时间为 $O(\log^{2d} m \cdot \log^2 n)$。 4. 构造集合$\hat{\mathcal{S}}$所需时间为 $O(m \log^{2d} m)$。

#### 4.2.1 二维正方形的扩展四叉树。

给定一个正方形集合 $\mathcal{S}'$，构造一个二维四叉树 $\mathbb{T}$ （定义见第2节），使其根节点包含 $\mathcal{S}'$中的所有正方形。为简单起见，我们假设 $\mathcal{S}'$中任意两个输入正方形都不共享角。这样，我们就可以对输入点进行轻微扰动，使得没有点 $p \in P'$ 落在四叉树的任何网格点上，同时每个正方形仍包含与之前相同的点集。考虑一个节点 $v \in \mathbb{T}$ 和一个正方形 $S \in \mathcal{S}'$。令 $C$ 和 $par(C)$分别为节点$v$ 和 $v$的父节点所对应的单元。令$\text{proj}_i(C)$、$\text{proj}_i(par(C))$和$\text{proj}_i(S)$分别表示 $C$、 $par(C)$和 $S$在第 $i$维上的投影。那么，当且仅当$\text{proj}_i(C) \subseteq \text{proj}_i(S)$，但$\text{proj}_i(par(C)) \not\subseteq \text{proj}_i(S)$时， $S$ 在 $v$ 处是 $i$-长的。见图13(a)。对于所有 $u \in \mathbb{T}$，令 $\mathcal{S}(u, i) \subseteq \mathcal{S}'$表示在节点 $u$处 $i$-长的正方形。从直观上说，这些正方形覆盖了 $C$ 在第 $i$维上的边，但不覆盖 $par(C)$在第 $i$维上的任何边。现在，我们将在 $\mathbb{T}$ 的每个节点处构造两个次级结构，具体如下：第一个结构是在 $\mathcal{S}(u, 1)$中的正方形在第二维上的投影上构建的一维四叉树，第二个结构是在 $\mathcal{S}(u, 2)$中的正方形在第一维上的投影上构建的一维四叉树。

在每个次级结构中，当且仅当节点 $u$ 是深度最小的节点（根节点的深度为零），且 $I$ 与单元 $C_u$ 的左端点或右端点之一相交时，区间 $I$ （对应一个正方形 $S \in \mathcal{S}'$）才会被分配给节点 $u$ 。见图 13(b)。根据此定义，每个区间在次级结构中最多会被分配给两个节点。

现在我们将使用 $\mathbb{T}$ 所诱导的次级结构集合 $\mathcal{S}_{\text{sec}}$ 的节点集。对于任意节点 $u \in V_{\text{sec}}$，在其被分配的、与单元 $C_u$左（分别为右）端点相交的区间中，找出 最大的 区间 $I_\ell$ （分别为 $I_r$），即

与 $C_u$重叠最大的区间。见图 13(c)。接着，我们对 $V_{\text{sec}}$中的所有节点执行以下操作：对于节点 $u \in V_{\text{sec}}$，分别用 $S'$ 和 $S''$ 表示与被分配的区间 $I_\ell$ 和 $I_r$对应的正方形。此外，令 $w$ 为 $\mathbb{T}$中的节点， $u$ 的次级结构就是在该节点上构建的。然后，我们将矩形 $S_1 \cap C_w$ 和$S_2 \cap C_w$纳入 $\mathcal{S}$ 。

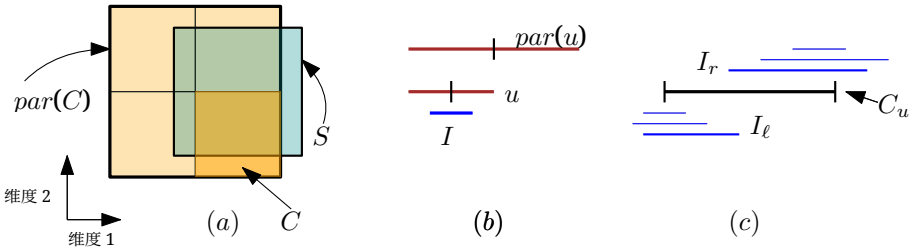


图 13: （a）节点 $v$ 处边长为 1 的正方形 $S$（对应的单元 $C$ 以较深的橙色突出显示）；（b） $I$ 被分配给 $v$ 的两个子节点；（c） $C_v$处的极大区间 $I_\ell$ 和 $I_r$ 。

#### 4.2.2 用于 $2d$维超立方体的扩展四叉树。

在本节中，我们需要将第2节中定义的四叉树推广。对于 $d' > 2$，$d'$ 维四叉树的定义与第2节中定义的四叉树类似，但每个内部节点现在有 $2^{d'}$ 个子节点，而不是四个。假设根据归纳法，我们已经定义了如何构造维度小于或等于 $2d-1$ 的扩展四叉树。（基本情形是为二维正方形构造的扩展四叉树。）现在，我们定义如何为 $2d$ 维超立方体构造该结构。首先，为超立方体集合 $\mathcal{S}'$ 构造常规的 $2d$ 维四叉树 $\mathbb{T}$ 。考虑任意节点 $v \in \mathbb{T}$。将先前的定义推广可得，对于任意 $1 \le i \le 2d$，超立方体 $S \in \mathcal{S}'$ 在节点 $v$ 处是 $i$ 长的，当且仅当 $\text{proj}_i(C) \subseteq \text{proj}_i(S)$，但 $\text{proj}_i(par(C)) \not\subseteq \text{proj}_i(S)$。对于所有 $v \in \mathbb{T}$，令 $\mathcal{S}(v,i) \subseteq \mathcal{S}'$ 表示在节点 $v$ 处 $i$ 长的超立方体。现在，我们将在 $\mathbb{T}$ 的每个节点处按如下方式构造 $2d$ 个次级结构：对于所有 $1 \le i \le 2d$，第 $i$ 个次级结构是一个建立在 $\mathcal{S}(v,i)$ 上的 $(2d-1)$ 维扩展四叉树，使用其除第 $i$ 维以外的所有 $2d$ 个维度。具体而言，形如 $\ell_1 \times \cdots \times \ell_i \times \cdots \times \ell_{2d}$ 的任意超立方体$S \in \mathcal{S}(v,i)$ 都被投影为一个 $(2d-1)$ 维超立方体 $\hat{}$ $\ell_1 \times \cdots \times \ell_{i-1} \times \ell_{i+1} \times \cdots \times \ell_{2d}$。令 $\mathcal{S}_v$ 表示使用例程 . 在 $v \in \mathbb{T}$ 处为次级结构归纳选取的 $(2d-1)$ 维超矩形的集合。定义函数 $g$ ，将作为集合 $\hat{} \mathcal{S}_v$ （对于某个 $v \in \mathbb{T}$）一部分选取的 $(2d-1)$ 维超矩形映射到其对应的 $2d$ 维超立方体 $S \in \mathcal{S}'$。现在，我们定义由 $2d$ 维超矩形构成的集合 $\hat{} \mathcal{S}$ :

$$\hat{\mathcal{S}} \leftarrow \bigcup_{v \in \mathbb{T}} \left( \bigcup_{S' \in \hat{\mathcal{S}}_v} (g(S') \cap C_v) \right).$$

**Claim 6.** （可行性）任意点 $p \in P'$ 至少被集合 $\hat{}\mathcal{S}$ 中的一个集合覆盖 .

证明。 我们首先针对二维正方形的情形证明这一断言。对于任意点 $p \in P'$，我们知道至少有一个覆盖它的正方形 $S \in \mathcal{S}'$ 。根据我们的假设， $p$ 不在四叉树的任何网格点上。接着，我们断言，对于某个单元 $C_v$ ，$S$ 是其边覆盖，其中 $v$ 是一个叶节点，并且

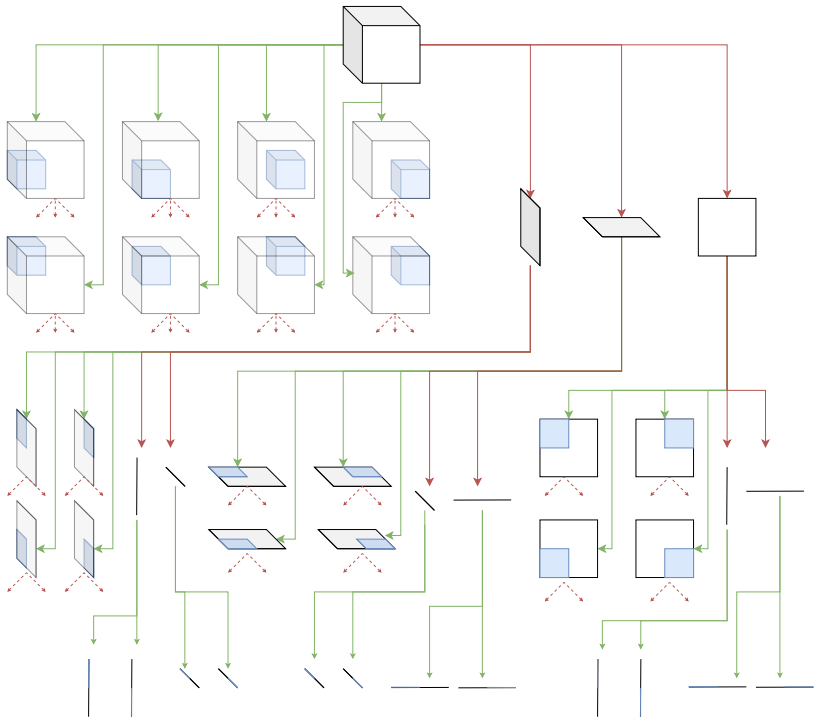

图 14：以一个 $2 \times 2 \times 2$ 立方体为根节点的扩展四叉树。

$C_v$ 包含 $p$。然后，根据 $i$-长的定义，存在 $v$ 在 $\mathbb{T}$ 中的一个祖先 $u$ （也可能是 $v$ 本身），使得对于某个 $i \in [2]$， $S$ 相对于 $C_u$是 $i$-长。这意味着 $\hat{\mathcal{S}}$ 由一个（极大）正方形 $S'$ 组成，使得 $S' \in \mathcal{S}$（$u$， $i$）且 $S' \cap C_u \supseteq S \cap C_u$。然后，根据我们构造扩展四叉树的方式， $S' \cup C_u$ 属于集合 $\hat{\mathcal{S}}$。因此，该断言对二维正方形成立。推广这一思路，对于 $d \geq 1$， $2d$维超立方体的情形也能保证可行性。 □

**引理9**。 （有界频率） $P'$ 中的任意点都位于 $O$（$\log^{2d} m$） 个集合中 $\hat{\mathcal{S}}$.

证明。 对于二维正方形的扩展四叉树，令 $\tau(2)$ 表示 $\hat{\mathcal{S}}$中包含点 $p = (p_x, p_y) \in P'$的集合的最大数量。根据标准四叉树的性质，二维四叉树中对应单元包含 $p$ 的节点数量为 $O(\log m)$。在每个这样的节点处，每个次级结构中对应单元包含 $p$ （ $p_x$ 或 $p_y$）的投影的节点数量为 $O(\log m)$。对于次级结构中包含 $p$的每个单元，我们至多选取两个（极大）超矩形加入 $\hat{\mathcal{S}}$。因此， $\tau(2) = O(\log^2 m)$。

一般而言，对于 $d'$维超立方体的扩展四叉树，令 $\tau(d')$ 表示 $\hat{\mathcal{S}}$ 中包含任意给定 $d'$维点的集合的最大数量。由于我们在每个节点处构建 $2d$ 个次级结构，因此得到以下递推关系：

$$\tau(2d) = O((2d)\log m) \cdot \tau(2d-1) = O(\log^{2d} m),$$

其中大-$O$-记号中隐藏的常数取决于 $d$。 □

**引理 10.** 如果存在针对$(P', \hat{\mathcal{S}})$ 的 $\alpha$-近似动态集合覆盖算法，那么就存在针对$(P', \mathcal{S}')$的$O(\alpha \log^{2d-1} m)$-近似动态集合覆盖算法。

证明。 令 OPT 和 $\widehat{\text{OPT}}$ 分别表示 $(P', \mathcal{S}')$ 与 $(P', \hat{\mathcal{S}})$ 的最优集合覆盖的大小。对于 $\mathcal{S}'$ 中的任意超立方体 $S$ ，我们将证明 $\hat{\mathcal{S}}$ 中存在 $O(\log^{2d-1} m)$ 个超矩形，其并集

将完全覆盖 $S$。因此，$\widehat{\text{OPT}} = \text{OPT} \cdot O(\log^{2d-1} m)$。注意，对于每个集合 $S' \in \hat{\mathcal{S}}$，都存在一个对应的超立方体 $S \in \mathcal{S}'$，其覆盖的点至少包含 $P'$ 中由 $S'$ 覆盖的点。对于 $(P', \mathcal{S}')$，$(P', \hat{\mathcal{S}})$ 的一个 $\alpha$-近似集合覆盖意味着近似因子为：

$$\frac{b}{\text{OPT}} = \frac{b \cdot O(\log^{2d-1} m)}{\widehat{\text{OPT}}} = O(\alpha \log^{2d-1} m),$$

其中，最后一个等式由 $b \leq \alpha \cdot \text{OPT}$ 得出。

最后，我们证明覆盖性质。我们将通过对维度 $\hat{}$ 大小进行归纳法来证明。作为基本情形，对于二维中的正方形，令 $\mu(2)$ 表示在 $\mathcal{S}$ 中所需集合的数量，使得这些集合的并集完全覆盖一个正方形 $S \in \mathcal{S}'$。对于任意 $S \in \mathcal{S}'$，令 $\text{long}(S)$ 表示 $\mathbb{T}$ 中满足 $S$ 为 1-long 的节点集。根据四叉树的标准性质，有 (a) $|\text{long}(S)| = O(\log m)$，以及 (b) $S \leftarrow \bigcup_{v \in \text{long}(S)} (S \cap C_v)$，其中 $C_v$ 是与 $v$ 对应的单元。现在考虑任意节点 $v \in \text{long}(S)$。令 $I$ 为在基于 $\mathcal{S}(v,1)$ 构建的 $v$ 次级结构中与 $S$ 对应的区间。通过在次级节点处选择极大区间，显然存在两个极大区间覆盖 $I$。因此，$\mu(2) \leq 2 \cdot |\text{long}(S)| = O(\log m)$。

一般而言，令 $\mu(2d)$ 表示所需集合的数量，使得 $\hat{\mathcal{S}}$ 这些集合的并集能够完全覆盖超立方体 $S$ 的数量，使得 $S'$。于是我们断言

$$\mu(2d) = O(2^d \log m) \times \mu(2d-1),$$

其中 $O(2^d \log m)$ 是 $\mathbb{T}$ 中的节点数，其中 $S$ 的长度为 1。解该递推关系可得 $\mu(2d) = O(\log^{2d-1} m)$。 □

### 4.3 最终算法

对于一般的动态集合覆盖问题，Bhattacharya 等人 [13] 给出了一种 $O(f)$-近似算法，其最坏情况更新时间为 $O(f \log^2 n)$。回忆一下，$f$ 是该集合系统的频率。我们将他们的算法作为黑盒，应用于集合系统 $(P', \hat{\mathcal{S}})$。令 ALG 表示他们的算法报告的集合。然后，我们的算法也会将 ALG 报告为 $(P', \mathcal{S}')$ 的一个集合覆盖。该解是可行的，因为 $\hat{\mathcal{S}}$ 中的每个集合也都属于 $\mathcal{S}'$。

**引理 11.** 我们的算法的近似因子为 $O(\log^{4d-1} m)$。

证明。 对于集合系统 $(P', \hat{\mathcal{S}})$，其频率为 $f = O(\log^{2d} m)$，因此，[13] 的算法为该集合系统带来 $O(\log^{2d} m)$ 近似。利用引理 10，这意味着集合系统 $(P', \mathcal{S}')$ 的近似因子为 $O(\log^{2d} m) \cdot O(\log^{2d-1} m) = O(\log^{4d-1} m)$。 □

**引理 12.** 更新时间为 $O(\log^{2d} m \cdot \log^2 n) = O(\log^{2d+2} m)$。

证明。 插入或删除一个点时，可以通过遍历树 $\mathbb{T}$，在 $O(\log^{2d} m)$ 时间内找到 $\hat{\mathcal{S}}$ 中包含该点的 $O(\log^{2d} m)$ 个集合。[13] 的算法更新时间为 $O(f \log^2 n) = O(\log^{2d} m \cdot \log^2 n) = O(\log^{2d+2} m)$。这是因为 $d$ 维超立方体的对偶维度为 $O(d)$ [32, 33]，因此 $O(\log n) = O(\log m)$。 □

**引理 13.** 构造集合 $\hat{\mathcal{S}}$ 所需的时间为 $O(m \log^{2d} m)$。

证明。 令 $T(m, d)$ 表示在 $d$ 维空间中的 $m$ 个超立方体上构建扩展四叉树所需的时间。作为基本情形，我们首先计算 $T(m, 1)$。构建一维四叉树的骨架结构需要 $O(m)$ 时间，因为区间的端点位于整数网格 $[0, 4m]$ 上。然后，将每个区间“分配”给这棵四叉树中的一个节点，需要 $O(m \log m)$ 时间。

对于一个被分配了 $m_v$ 个区间的节点 $v$，找到两个极大区间需要 $O(m_v)$ 时间。因此，$T(m, 1) = O(m \log m) + \sum_v O(m_v) = O(m \log m)$。

现在考虑 $2d$ 维空间中 $m$ 超立方体上的扩展四叉树。同样，构建四叉树骨架只需 $O(m)$ 时间。对于任意 $1 \le i \le 2d$，找到 $\mathbb{T}$ 中超立方体"长度为 $i$"的节点需要 $O(2^{2d} \log m) = O(\log m)$ 时间。因此，

$$\begin{aligned} T(m, 2d) &= O(m \log m) + \sum_v T(m_v, 2d-1) \\ &= O(m \log m) + \sum_v m_v \log^{2d-1} m_v \qquad \text{(by induction)} \\ &= O(m \log m) + \sum_v m_v \log^{2d-1} m \\ &= O(m \log^{2d} m), \qquad \text{since, } \sum_v m_v = O(m \log m). \end{aligned}$$

□

## 4.4 加权情形

我们将算法简单扩展到每个超矩形 $S \in \mathcal{S}$ 都具有权重 $w_S \in [1, W]$ 的情形。首先，我们将每个集合 $S$ 的权重取整至大于或等于 $w_S$ 的最小 2 的幂。这会产生 $O(\log W)$ 个不同的权重类别。接下来，对于每个权重类别，在将该类别的超矩形从 $d$ 维归约为 $2d$ 维超立方体后（如上一节所示），基于该类别中的超立方体构建一棵扩展四叉树。最后，令 $\mathcal{S}$ 为从所有 $O(\log W)$ 棵扩展四叉树中得到的（极大）超立方体的集合。在 $(P', \hat{\mathcal{S}})$ 上运行 Bhattacharya等人 [13] 的动态集合覆盖算法。

**引理14**。 该算法的近似因子为 $O(\log^{4d-1} m \cdot \log W)$。

证明。 考虑 $(P', \mathcal{S}')$ 的最优解，并令 OPT 为其最优权重。通过对最优解中每个集合的权重进行舍入，其总权重至多变为 $2 \cdot$OPT。因此，舍入后集合系统中的最优解权重至多为 $2 \cdot$OPT。与无权情形相比，现在集合系统 $(P', \hat{\mathcal{S}})$ 的频率增加了 $O(\log W)$ 倍。因此，我们得到的近似因子为 $O(\log^{4d-1} m \cdot \log W)$。

□

**引理15**。 该算法的更新时间为 $O(\log^{2d} m \log^3(Wm))$。

证明。 插入或删除一个点时，可以通过遍历 $O(\log W)$ 棵扩展四叉树，在 $O(\log^{2d} m \log W)$ 时间内找到 $\hat{\mathcal{S}}$ 中包含该点的 $O(\log^{2d} m \log W)$ 个集合。 [13] 的算法的更新时间为 $O(f \log^2 Wn) = O(\log^{2d} m \log W \log^2 Wn) = O(\log^{2d} m \log^3(Wm))$。

□

**定理 5**。对于 $d$ 维超矩形的加权动态集合覆盖，存在一种算法，其近似因子为 $O(\log^{4d-1} m \cdot \log W)$，更新时间为 $O(\log^{2d} m \cdot \log^3(Wm))$。

对于 $2d$ 维超立方体的动态集合覆盖，如果所有角点都是整数且在某个常数 $c > 0$ 的范围内有界，我们还可以得到以下推论。

**推论3**。 对于 $2d$ 维超立方体的加权动态集合覆盖，存在一种算法，其近似因子为 $O(\log^{4d-1} m \cdot \log W)$，更新时间为 $O(\log^{2d} m \cdot \log^3(Wm))$；其中，所有角点的坐标均为整数，且在 $[0, cm]^{2d}$ 内有界，$c > 0$ 为固定值。

## 5 $d$ 维超矩形的动态击中集

本节介绍一种用于 $d$ 维超矩形击中集的动态算法。我们将把该问题归约为 $2d$ 维空间中的一个动态集合覆盖实例，并使用上一节设计的算法（定理 3）。回顾一下， $P$ 是点集， $\mathcal{S}$ 是超矩形集。假设 $P$ 中的所有点以及 $\mathcal{S}$ 中的超矩形都位于盒 $[0,N]^d$ 内。对于所有 $p \in P$，我们将 $p=(p_1,\ldots,p_d)$ 变换为一个 $2d$ 维超立方体 $\mathcal{S}(p)$，其边长为 $N$，且“左下”角点为 $p'=(-p_1,\ldots,-p_d,p_1,\ldots,p_d)$。令 $\mathcal{S}(P)$ 表示变换后的超立方体。接下来，我们将每个超矩形（例如 $S \in \mathcal{S}$）变换为一个 $2d$ 维点$P(S)=(-a_1,\ldots,-a_d,b_1,\ldots,b_d)$；该超矩形的“左下”角点为 $a=(a_1,\ldots,a_d)$，“右上”角点为 $b=(b_1,\ldots,b_d)$。令 $P(\mathcal{S})$ 表示变换后的点。参见图 15。

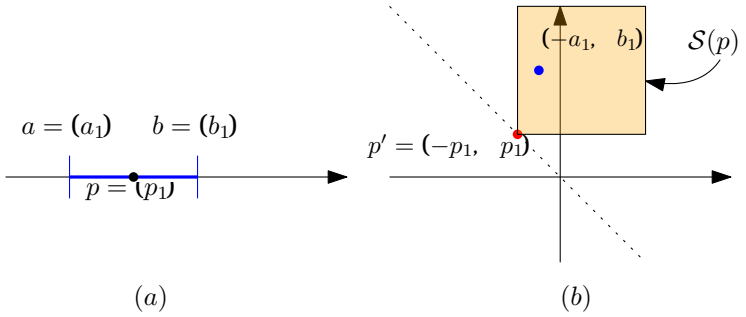


图 15：（a）一维中位于区间 $S=[a_1,b_1]$ 内的点 $p$，以及（b）将 $p$ 变换为正方形 $\mathcal{S}(p)$，并将 $S$ 变换为二维中的点 $(-a_1,\ b_1)$。

**引理 16。** 在 $\mathbb{R}^d$ 中，点 $p$ 位于超矩形 $S$ 内，当且仅当在 $\mathbb{R}^{2d}$ 中，超立方体 $\mathcal{S}(p)$ 包含点 $P(S)$。

证明。 如果且仅如果对所有 $1 \le i \le 2d$ 都有 $q_i \ge q'_i$，我们就称点 $q(q_1,\ldots,q_{2d})$ 支配 另一个点 $q'(q'_1,\ldots,q'_{2d})$。假设 $p$ 位于 $S$ 内。那么，对所有 $1 \le i \le d$，都有 $p_i \ge a_i$ 和 $p_i \le b_i$。这意味着 $P(S)=(-a_1,\ldots,-a_d,b_1,\ldots,b_d)$ 支配点 $p'(-p_1,\ldots,-p_d,p_1,\ldots,p_d)$，该点是 $\mathcal{S}(p)$ 的左下角。

$\mathcal{S}(p)$ 的右上角坐标为 $(-p_1+N,\ldots\ldots,-p_d+N,p_1+N,\ldots\ldots,p_d+N)$。对于所有 $1 \le i \le d$，由于 $N \ge p_i - a_i$，因此可推出 $-p_i+N \ge -a_i$。对于所有 $1 \le i \le d$，由于$p_i+N \ge N \ge b_i$，因此可推出 $p_i+N \ge b_i$。最终可推出，$\mathcal{S}(p)$ 的右上角支配 $P(S)$。因此， $\mathcal{S}(p)$ 包含 $P(S)$。

假设 $p$ 位于 $S$ 之外。那么存在一个维度 $i$，使得 $p_i < a_i$ 或 $p_i > b_i$。如果$p_i < a_i$，那么 $-p_i > -a_i$，这意味着 $P(S)$ 不可能支配 $\mathcal{S}(p)$ 的左下角。如果 $p_i > b_i$，同样， $P(S)$ 不可能支配 $\mathcal{S}(p)$ 的左下角。这意味着 $P(S)$ 位于 $\mathcal{S}(p)$ 之外。 □

我们将使用上述归约，把 $P$中的点转化为 $\mathbb{R}^{2d}$ 中的超立方体，并把 $\mathcal{S}$ 中的超矩形转化为 $\mathbb{R}^{2d}$中的点。因此， $\mathbb{R}^d$ 中针对超矩形的击中集问题已归约为 $\mathbb{R}^{2d}$ 中针对超立方体的集合覆盖问题。再采用与上一节所述类似的秩空间归约后，该实例中的所有点以及超立方体的角点都具有整数坐标。随后，使用推论3求解该集合覆盖实例。其正确性由引理 16保证。注意，对于 $2d$ 维超立方体， $O(\log n)=O(\log m)$，算法的性能总结如下。

**定理 6.** 经过耗时 $O(n\log^{2d} n)$ 的预处理步骤后，存在一种针对 $d$ 维超矩形击中集问题的算法，其近似因子为 $O(\log^{4d-1} n)$，更新时间为 $O(\log^{2d+2} n)$。在加权情形下，近似因子为 $O(\log^{4d-1} n \cdot \log W)$，更新时间为 $O(\log^{2d} n \log^3(Wn))$。

# 6 未来工作

在本工作的第一部分中，我们研究了二维正方形的在线几何集合覆盖和击中集。这为未来开辟了一条有趣的研究方向。我们提出几个开放问题：

1. 作为二维正方形的一种自然推广，是否可以为三维立方体设计一个 $o(\log n \log m)$-竞争算法? 本文中用于二维正方形的技术似乎无法推广到三维立方体。另一个值得研究的情形是几何对象为二维圆盘时。我们能否为其获得一个 $o(\log n \log m)$-竞争的在线集合覆盖算法？ 2. 为矩形的集合覆盖（分别为击中集）设计一个竞争比为 $o(\log^2 n)$ 的在线算法，或者证明一个几乎匹配的下界 $\Omega\left(\log^2 n \log\log n\right)$（该下界适用于一般情形下的在线集合覆盖 [4]）？ 3. 作为上述问题的推广，是否可以为具有“常数”VC 维度的集合系统，获得竞争比为 $\overline{o(\log^2 n)}$ 的集合覆盖和击中集在线算法？ 4. 对于在线集合覆盖的加权情形，即使是单位正方形，我们能否获得竞争比为 $o(\log n \log m)$ 的算法？ 5. 为正方形的击中集设计一个竞争比为 $O(\log n)$ 的在线算法，从而改进我们算法的 $O(\log N)$ 竞争比（正方形的角点为整数且包含在 $[0, N)$ [2]。

在我们工作的第二部分中，我们研究了针对$d$维超矩形的动态几何集合覆盖和击中集。这一研究方向巧妙地结合了数据结构、计算几何和近似算法。最后，我们提出了动态设置中的几个开放问题：

1. 改进二维矩形情形下动态集合覆盖的近似因子。具体而言，能否在多对数更新时间内实现 $O(\log n)$ 近似？在这种设定中，矩形是固定的，但点是动态的。 2. 对于二维矩形情形下的加权动态集合覆盖，能否实现与 $W$ 无关的近似和更新时间界（其中 $W$ 是输入中权重最高的矩形与权重最低的矩形的权重之比）？ 3. 对于 [3, 19, 20]，中研究的（全）动态集合覆盖情形，当集合为矩形时，我们能否得到更新时间为次线性、近似因子为多对数级的算法（正如 [20] 中最初提出的问题）？

# 参考文献

[1] Amir Abboud、Raghavendra Addanki、Fabrizio Grandoni、Debmalya Panigrahi 和 Barna Saha。动态集合覆盖：改进的算法与下界。载于第*51*届年度 *ACM SIGACT* 计算理论研讨会（*STOC*）论文集，第 114–125 页，2019年。[2] Pankaj K. Agarwal、Hsien-Chih Chang、Subhash Suri、Allen Xiao 和 Jie Xue。动态几何集合覆盖与击中集。载于第*36*届国际计算几何研讨会（*SoCG 2020*），LIPIcs 第 164 卷，第 2:1–2:15 页。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik，2020年。[3] Pankaj K Agarwal、Hsien-Chih Chang、Subhash Suri、Allen Xiao 和 Jie Xue。动态几何集合覆盖与击中集。*arXiv* 预印本 *arXiv:2003.00202*，2020年。[4] Noga Alon、Baruch Awerbuch 和 Yossi Azar。在线集合覆盖问题。载于第*35*届年度 *ACM* 计算理论研讨会（*STOC*）论文集，第 100–105 页，2003年。[5] Boris Aronov、Esther Ezra 和 Micha Sharir。轴平行矩形与长方体的小规模 \eps-网。*SIAM* 计算杂志，39(7):3248–3282，2010年。[6] Sanjeev Arora。欧几里得旅行商问题及其他几何问题的多项式时间近似方案。*ACM* 学报（*JACM*），45(5):753–782，1998年。[7] Sunil Arya、David M. Mount、Nathan S. Netanyahu、Ruth Silverman 和 Angela Y. Wu。固定维度下近似最近邻搜索的最优算法。*ACM* 学报（*JACM*），45(6):891–923，1998年。[8] Sepehr Assadi 和 Shay Solomon。通过超图极大匹配实现全动态集合覆盖：通过局部方法获得最优近似。载于第*29*届年度欧洲算法研讨会（*ESA 2021*），第 8 页。Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH，Dagstuhl Publishing，2021年。[9] Mark de Berg、Marc van Kreveld、Mark Overmars 和 Otfried Schwarzkopf。计算几何。载于计算几何，第 1–17 页。Springer，1997年。[10] Sayan Bhattacharya、Monika Henzinger 和 Giuseppe F. Italiano。用于顶点覆盖与匹配的确定性全动态数据结构。*SIAM* 计算杂志，47(3):859–887，2018年。[11] Sayan Bhattacharya、Monika Henzinger 和 Giuseppe F. Italiano。通过原始-对偶方法实现动态算法。信息与计算，261:219–239，2018年。[12] Sayan Bhattacharya、Monika Henzinger 和 Danupon Nanongkai。一种用于动态集合覆盖的新确定性算法。载于*2019 IEEE* 第*60*届年度计算机科学基础研讨会（*FOCS*），第 406–423 页。IEEE，2019年。[13] Sayan Bhattacharya、Monika Henzinger、Danupon Nanongkai 和 Xiaowei Wu。动态集合覆盖：改进的摊还与最坏情况更新时间。载于*2021*年 *ACM-SIAM* 离散算法研讨会（*SODA*）论文集，第 2537–2549 页。SIAM，2021年。[14] Sujoy Bhore、Jean Cardinal、John Iacono 和 Grigorios Koumoutsos。动态几何独立集。载于日本离散与计算几何、图论及博弈会议，2021年。

[15] Marcin Bienkowski、Jarosław Byrka、Christian Coester 和 Łukasz Je˙z。针对弱对手的 k- 服务器无界下界。载于第 *52* 届 *ACM SIGACT* 计算理论研讨会（*STOC*）论文集，第 1165–1169页，2020年。[16] Niv Buchbinder 和 Joseph Naor。通过原始-对偶方法设计竞争性在线算法。计算机科学理论与趋势，3(2-3)：93–263，2009年。[17] Jean Cardinal、John Iacono 和 Grigorios Koumoutsos。最坏情况下高效的动态几何独立集。载于第*29*届欧洲算法研讨会（*ESA 2021*）。Schloss Dagstuhl-Leibniz-Zentrum f¨ur Informatik，2021年。[18] Timothy M. Chan、Elyot Grant、Jochen K¨onemann 和 Malcolm Sharpe。通过改进的准均匀采样实现加权容量约束、优先级及几何集合覆盖。载于第二十三届*ACM-SIAM*离散算法研讨会（*SODA*）论文集，第1576–1585页。SIAM，2012年。[19] Timothy M. Chan 和 Qizheng He。用于几何集合覆盖、具有次线性更新时间的更多动态数据结构。载于第*37*届国际计算几何研讨会（*SoCG 2021*）。Schloss Dagstuhl-Leibniz-Zentrum f¨ur Informatik，2021年。[20] Timothy M. Chan、Qizheng He、Subhash Suri 和 Jie Xue。重新审视动态几何集合覆盖。载于*2022*年*ACM-SIAM*离散算法研讨会（*SODA*）论文集，第3496–3528页。SIAM，2022年。[21] Kenneth L. Clarkson 和 Kasturi Varadarajan。改进的几何集合覆盖近似算法。载于第二十一届计算几何年会（*SoCG*）论文集，第135–141页，2005年。[22] Justin Dallant 和 John I acono。动态几何测度问题的条件下界。*arXiv*预印本 *arXiv:2112.10095*，2021年。[23] Guy Even 和 Shakhar Smorodinsky。在线击中集与顶点排序。载于算法——*ESA 2011*——第*19*届欧洲算法年会，《计算机科学讲义》第6942卷，第347–357页。Springer，2011年。[24] Anupam Gupta、Gregory Kehne 和 Roie Levin。随机顺序在线集合覆盖与离线集合覆盖一样简单。载于*2021*年*IEEE*第*62*届计算机科学基础年会（*FOCS*），第1253–1264页。IEEE，2022年。[25] Anupam Gupta、Ravishankar Krishnaswamy、Amit Kumar 和 Debmalya Panigrahi。集合覆盖的在线与动态算法。载于第*49*届*ACM SIGACT* 计算理论研讨会（*STOC*）论文集，第537–550页，2017年。[26] Anupam Gupta 和 Roie Levin。具有有界回溯的全动态次模覆盖。载于*2020*年*IEEE*第*61*届计算机科学基础年会（*FOCS*），第1147–1157页。IEEE，2020年。[27] Anupam Gupta 和 Roie Levin。在线次模覆盖问题。载于第十四届*ACM-SIAM*离散算法年会（*SODA*）论文集，第1525–1537页。SIAM，2020年。[28] Sariel Har-Peled 和 Mira Lee。重新审视加权几何集合覆盖问题。《计算几何杂志》，3(1)：65–85，2012年。

[29] Monika Henzinger、Stefan Neumann 和 Andreas Wiese。区间、超立方体和超矩形的动态近似最大独立集。载于第*36*届国际计算几何研讨会（*SoCG 2020*），*LIPIcs*第164卷，第51:1–51:14页。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik，2020年。[30] Nabil H. Mustafa、Rajiv Raman 和 Saurabh Ray。确定几何集合覆盖的APX-困难性。载于*2014*年*IEEE*第*55*届计算机科学基础年会（*FOCS*），第541–550页。IEEE，2014年。[31] Nabil H. Mustafa 和 Saurabh Ray。通过局部搜索求解几何击中集问题的多项式时间近似方案。载于第*25*届*ACM*计算几何研讨会（*SoCG*）论文集，第17–22页。ACM，2009年。[32] Norbert Sauer。关于集合族的密度。组合理论杂志*A*辑，13(1)：145–147，1972年。[33] Saharon Shelah。一个组合问题：无穷语言中模型与理论的稳定性和序。太平洋数学杂志，41(1)：247–261，1972年。[34] Kasturi Varadarajan。通过准均匀采样实现加权几何集合覆盖。载于第*42*届*ACM*计算理论研讨会（*STOC*）论文集，第641–648页，2010年。

## 一种在线区间集合覆盖算法

本节针对区间集合覆盖问题，介绍一种紧确的 2-竞争算法。

该算法从一个空集覆盖开始。每次迭代中，当新点 $p$ 到达时，如果它已被覆盖，我们便不采取任何操作；否则，我们从覆盖 $p$ 的区间中，选出右端点最靠右的区间和左端点最靠左的区间。

该算法的正确性显然成立，因为对于每个新出现的未覆盖点，我们都会选取一个覆盖它的区间。在算法后续步骤中，我们不会从解中移除任何区间，因此算法终止时，所有点都已被覆盖。

**定理 7.** 存在一种用于在线区间集合覆盖问题的 *2*-竞争算法。

证明。 考虑最优解 OPT 中的一个区间 $I$。当它覆盖的第一个未覆盖点到达输入时，我们的算法会选取两个区间，并确保这两个区间覆盖所有的 $I$。因此，对于 OPT 中的每个区间，我们的解至多选取 2 个区间，从而得到一个 2-竞争解。

□

**定理 8。** 存在一个关于区间集合覆盖问题的实例，使得任何在线算法（无补救）在该实例上的竞争比至多为 *2*。

证明。 考虑给定的区间集合： $A := [0,1], B := [1,2], C := [2,3], D := [3,4]$。第一个到达的点是 $p_1 := 2$。如果算法选取两个或更多集合，那么结论成立，因为 OPT 的大小为 1。否则，为覆盖 $p$，算法可以选取区间 $B$ 或 $C$。若选前者，第二个点应为 $p_2 := 3$；若选后者，则应为 $p_3 := 1$。可见，在这两种情况下，OPT 的大小均为 1，但在线算法都被迫选取两个区间。

□