

基于集合估计与覆盖优化的 无线电干扰源自动定位清除

摘 要

无线电干扰源的自动排查难在信息条件：源的数量、位置与有效接收半径均未知，单次检测至多给出一个带误差的方位，而“无信号”既可能来自距离，也可能来自定向源的朝向。面对这一条件，本文不猜测源的点位置，而是维护“真实位置可能落在哪里”的候选集合，并让每一次观测、移动与清除都作用在同一套集合约束上，由此建立由几何定位、主动测量、覆盖搜索与动作调度构成的闭环模型，并把移动、检测、换频与清除统一折算为虚拟耗时，使“能否清除”与“何时清除”成为可计算的判据。

针对问题一，本文将带误差的示向度改写为前向角域的半平面约束，经边界交点枚举、可行性筛选与凸包构造得到定位区域，并分别处理空集、无界与退化情形；在此基础上给出直径的顶点可达性与“直径圆覆盖该区域”的充要判据，并以反例说明覆盖与可清除并不等价，最终把清除条件落实到最小包围圆半径判据。

针对问题二，为兼顾接收保证与横向基线，本文由首次示向度与接收半径上界构造候选位置集，把“对任一候选源位置均保证接收”表示为圆盘之交，并用最小包围圆判据判断其非空；进而解析求出截断角域的最小包围圆，给出显式安全候选盘，并在其中以最坏后验半径与移动时间为准则选取第二检测点。

针对问题三，为在源数量未知时区分“尚未发现”与“根本不存在”，本文以原点与六个环站组成七点计划并证明其覆盖上界，使“全站无信号”成为频道缺席的解析证据；执行上为每个频道维护网格幸存集以融合正负观测，按期望信息增益与移动代价选点，并将覆盖、定位与清除三类任务统一建表后滚动重排。

针对问题四，针对定向源带来的无信号歧义，本文将发现条件提升为位置与朝向的前向半盘覆盖，采用最小命中集作为发现计划，以位置与朝向的联合幸存集保留遮挡解释；探点按可见性概率筛选，并在定位停滞时利用清除与朝向无关的性质执行预算化后备清除，另给出方格点集的连续覆盖证明作为备选方案。

全部模型与算法均在自建模拟器上完成了闭环验证，机制论证与定量结论取自可复现真实位置与逐动作观测的演练局，官方模拟器的正式测试仅用于附录中的最终效果验证。

关键词：有界误差；交会定位；最小包围圆；主动测量；定向覆盖

1 问题重述

1.1 问题背景

无线电干扰源的精准定位是频谱管理的末端任务。在初步监测和区域排查已确定目标分布范围后，仍需携带测向设备逐点检测，以缩小位置不确定性并完成清除。本题希望由一只搭载测向机和光学定位装置的机器狗自动完成这一过程，减少人工进入危险环境的需要，并缩短全部任务的完成时间 [1]。

目标区域为 $\Omega = \overline{B}(0, 1800)$ ，单位为米，坐标轴分别指向东、北。每个源占用 1 至 20 中的一个独立频道，源总数为 10 至 16。有效接收半径未知，介于 1000 与 1500 m 之间。检测返回无信号、近距离或示向度；示向度误差不超过 $\pm 1^\circ$ ，同一地点的重复检测不会改变误差。全向源在整个圆周发射；定向源仅在其未知朝向两侧各 90° 的闭半平面内发射。

机器狗从原点、频道 1 出发，以 5 m/s 沿直线移动，停机后才能检测。一次检测耗时 5 s，切换检测频道耗时 1 s。距源不超过 20 m 时，光学定位与清除共耗时 5 s；清除失败仅支付 3 s 光学检查时间。清除是否成功与源朝向无关。距源不超过 5 m 且能够接收信号时，返回近距离信息，可直接执行清除。圆域约束源的位置，机器狗允许到圆域外检测 [2]。

1.2 研究任务

- (1) 已知多个测点及同一源的示向度，计算交会定位区域的直径，判断以该直径为直径的圆能否覆盖定位区域。
- (2) 已知一个全向源的首次示向度，确定第二检测点的候选区域与选点策略，在保证接收的条件下获得较好的定位效果。
- (3) 对数量未知的全向源设计自动发现、定位和清除算法，在完成任务的前提下尽量减少总耗时，并报告测试结果。
- (4) 将策略推广到全向与定向源混合的情形，处理无信号的歧义，给出相应的搜索、定位、清除算法及测试结果。

问题三、四各完成三次正式测试。本文以自建模拟器演练局作为机制论证与定量分析的数据源，并以官方模拟器的六份正式导出包在附录中验证算法效率与结论，正式结果按题目表 1 的格式报告各次清除数量、平均定位清除时间与程序运行时间。

2 问题分析

四问并非彼此独立的技巧题，而是同一条“观测—证据—决策—执行”链路在不同信息条件下的展开，如图 1 所示。题面给定的目标区域、运动方式、动作耗时与源/频道

范围构成最上层的抽象；在线系统则分为证据层、决策层和执行层，分别维护候选集与幸存集、生成选点与任务表、串行执行动作；最下层是问题一至四各自的主线与发现、定位、清除三个闭环。后文的模型与算法都落在这条链路的某一层上：问题一、二提供证据层与决策层所需的几何判据，问题三、四解决时间约束下三个闭环的组织方式。

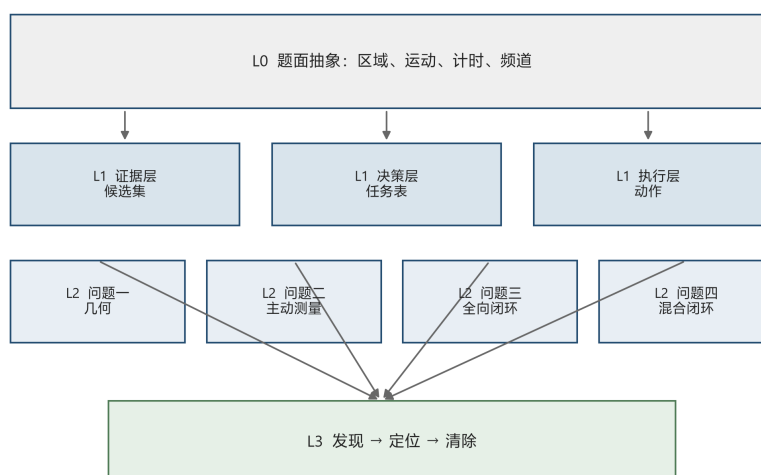


图 1 四问共用的模型链路：题面抽象（L0）、在线三层（L1）、四问主线（L2）与发现、定位、清除三个闭环（L3）。

问题一处于整条链路的几何起点。带误差的示向度只限制真实方位所在的角度区间，单条射线无法确定位置；若直接对射线求交，多解和数值敏感会同时出现。为避免这一点，本文将观测改写为前向角域的半平面约束，并通过直线交点枚举与凸包构造获得可行域。由这一改写还可进一步看出，可行域直径只反映区域尺度，并不等于覆盖该区域的圆半径：最远顶点对的中点未必能覆盖全部候选位置。正是这一差别，促使本文分别导出直径的顶点可达性与直径圆覆盖的充要判据，并以等边三角形反例说明“直径不超过 40 m”不能保证一次清除，最终把清除条件落到最小包围圆半径不超过 20 m 的充分判据上。

在问题一的几何基础上，问题二回答“第二测点选在哪里”。这里存在一对相互牵制的目标：若只追求大交会角，第二点可能远离源而收不到信号；若只求保证接收，又可能缺少横向基线。为兼顾二者，本文先由首次示向度与接收半径上界构造候选位置集，把“对未知真源均保证接收”表示为所有候选位置的 1000 m 圆盘之交，再用最小包围圆判据判断该交集是否非空；在此基础上证明截断角域的最小包围半径为 $1500/(2 \cos \delta)$ ，并在有效误差半角 1.005° 时给出半径约 249.885 m 的安全候选盘。在线执行时，用更紧的凸外包圆替代解析盘，在 37 个候选点中按最坏后验半径与移动时间择优。因此，问题二的接收保证是可证明的几何事实，定位收缩则通过有限候选近似完成。

问题三把单源判据推广到数量未知的全向源。此时真正的困难不是清除一个已发现

源，而是在有限时间内区分”尚未发现”与”根本不存在”。为此，本文以原点与六个环站组成七点计划，在环半径 1159.7 m 时证明最坏覆盖距离不超过 984.539 m，使”七站全无信号”成为该频道缺席的解析证据。实际执行不再逐站全频道扫描，而是为每个频道维护网格幸存集，统一融合正负观测：候选集为空即判缺席，并可跳过必为无信号的检测。定位点由期望信息增益与移动代价共同评分，覆盖、定位、清除三类任务统一建表后用最近邻与开放路径 2-opt 滚动重排，以减少发现任务与定位任务间的往返。

问题四进一步处理混合源，此时”无信号”既可能来自距离，也可能来自朝向。七点全向计划不再满足前向半盘下的发现要求，因此本文把发现条件提升为位置与朝向的联合覆盖，采用 31 点最小命中集，并按 15 m 位置间距和 72 个朝向档进行运行期校核；同时用逐频道的位置 $\times$ 朝向联合幸存集保留朝向遮挡解释。探点先按可见性概率筛选，再取移动最省者；定位停滞时，利用清除与朝向无关的性质，执行 20 m 方格、至多 400 格的预算化后备清除。作为备选，本文还由凸组合性质给出 49 点方格方案的连续覆盖证明。

上述链路在自建模拟器上闭环运行后，可归纳出四点结论：1. 带误差交会定位的严格清除判据是最小包围圆半径，而非定位区域直径；2. 主动测量应先保证接收，再在安全区域内优化横向基线与后验收缩；3. 多源搜索需把覆盖证明、正负观测滤波和滚动任务排序统一到同一时间目标下；4. 混合源场景必须保留位置与朝向的联合解释，并以有限后备清除兜底。上述闭环对应的观测更新与决策关系如图 2。

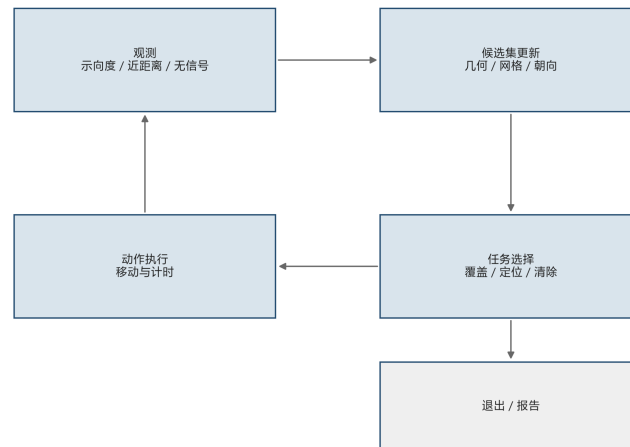


图 2 基于观测更新、集合估计与任务排序的闭环决策过程。

3 模型假设

假设 1 源的位置、频道、发射朝向和有效接收半径在一次测试中保持不变，不同频道互不干扰。这与静态测试案例及题设的频道独立性一致。

假设 2 采用二维欧氏距离，忽略高度差；机器人匀速直线移动，检测、换频、光学

定位与清除按给定时间串行执行。

假设 3 几何保证只使用有界误差和接收半径上下界。为覆盖接口两位小数的表示误差，在线角域取 $\delta_{\text{eff}} = 1^\circ + 0.005^\circ$ 。该附加量用于处理示向度的数值舍入误差。

假设 4 定位过程依据检测点坐标、频道及观测结果更新候选集合。

假设 5 信息增益与可见性评分采用均匀候选权重及均匀接收半径作为决策先验；几何可行性由测向误差和接收距离约束确定。

前三项中的物理约束取自题目和附件，数值余量与评分先验属于实现选择。

4 符号说明

表 1 主要符号及单位

符号	含义	单位
Ω, R	源位置圆域及其半径, $R = 1800$	m
g, s_i, q	真源位置、已有检测点、待选检测点	m
θ_i, δ	示向度及误差半角	度或弧度
$u(\alpha)$	单位方向向量 $(\cos \alpha, \sin \alpha)$	—
W_i, P	第 i 个前向角域及纯角域交集	—
K_c, P_c^{out}	频道 c 的可行位置集及保守凸外包	—
$D(P), m, \rho$	区域直径、最小包围圆心及半径	m
R_-, R_+	接收半径下、上界, 分别为 1000、1500	m
$\mathcal{Q}(K)$	对集合 K 保证接收的检测点区域	—
ℓ, ϵ_g	网格边长及格心位置裕量 $\ell/\sqrt{2}$	m
ψ, H	源朝向及朝向离散档数	度, 个
L, T	移动总距离及总虚拟耗时	m, s
N_m, N_s, N_c^+, N_c^-	检测、换频、成功清除、失败清除次数	次
$\bar{T}, T_{\text{real}}$	平均定位清除时间及程序运行时间	s/个, s

数学推导中三角函数的自变量采用弧度；带 $^\circ$ 的量先换算为弧度。记 $[a, b] = a_x b_y - a_y b_x$ 为二维叉积， $\bar{B}(c, r)$ 为闭圆盘。

5 计时模型与会话状态机

5.1 会话状态与合法动作

题目接口只有 /enter、/measure、/clear 与 /exit 四条指令。会话状态依次为未进入、进入倒计时、接口开放与断连，进入倒计时为 5 s，如图 3。只有 accepted=true 的响应推进虚拟时钟；被拒绝的请求返回的虚拟时刻为零，不能当作当前时刻使用，也

不改变位置与测向机频道。一次合法 /measure 结束后，测向机频道更新为本次检测频道；/clear 给出的是目标频道，既不改变测向机频道，也不产生换频耗时。因此清除之后原地再测的耗时取决于上一次 /measure 的频道，而不是清除的频道。

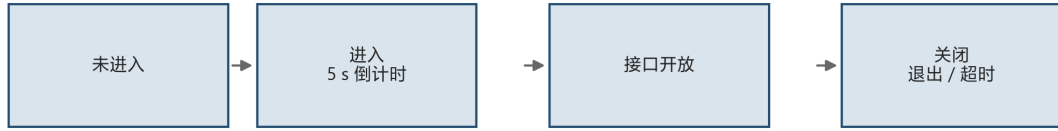


图3 会话状态机。进入后经 5 s 倒计时接口开放，主动退出或断连后进入关闭状态。

5.2 动作耗时与累计虚拟时间

题面将全部动作折算为虚拟耗时，见表 2。检测无论返回无信号、近距离还是示向度，均计 5 s；清除成功计 5 s，其中光学定位 3 s、激光清除 2 s，未命中目标时只支付光学检查的 3 s。移动与换频没有独立指令，由相邻动作的位置与频道变化推断。

表 2 动作耗时规则。移动耗时等于直线距离除以 5 m/s。

动作	指令	虚拟耗时/s	说明
进入	/enter	0	初始位置 (0, 0)、初始频道 1
检测	/measure	5+ 移动 + 换频	三种检测结果耗时相同
清除	/clear	5+ 移动 / 3+ 移动	成功 / 未命中目标，均不切换测向频道
退出	/exit	0	路径终点自由，无需返回原点
换频	由检测推断	1	相邻 /measure 的频道变化
移动	由动作推断	距离 / 5	相邻动作的位置变化

设整个会话的移动距离为 L ，检测、换频、成功清除与失败清除的次数分别为 N_m, N_s, N_c^+, N_c^- 。忽略每段移动时间的微秒量化时，总虚拟耗时为

$$T = \frac{L}{5} + 5N_m + N_s + 5N_c^+ + 3N_c^- \quad (1)$$

式 (1) 是后文任务排序的服务时间来源，也是由官方累计量反推移动时间（式 (38)）的依据。图 4 用附件的五步示例说明记账方式：首次检测含移动 100 s 与检测 5 s，单步 105 s；原地换到频道 2 检测单步 6 s，累计 111 s；清除目标频道 3 未命中，移动 80 s 加光学检查 3 s，单步 83 s，累计 194 s；清除不改变测向机频道，故随后原地以频道 2 复测只需 5 s，累计 199 s，与附件一致。

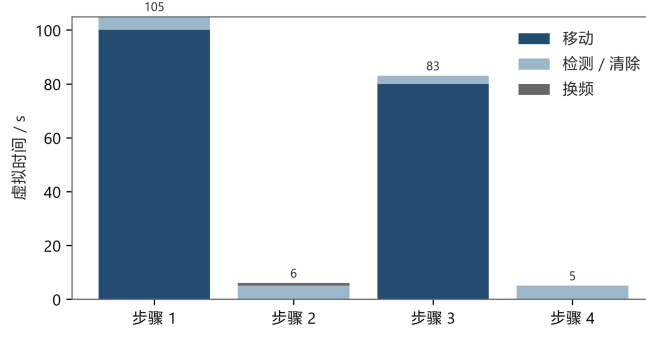


图 4 附件计时示例的逐动作构成。移动项为距离除以 5 m/s，标题数字为该步合计。

5.3 虚拟时间、现实预算与终止

虚拟时间度量任务完成速度，现实预算约束程序能否继续发出请求，两者不能混用。程序以 `/enter` 返回的 `remaining_real_duration_s` 为准，在每个动作前预留退出时间；虚拟时间以整数微秒累计，在线预测只用于规划与一致性核对。因此后文的“耗时”一律指式 (1) 的虚拟时间，程序运行时间在结果表中单列。策略在预算不足或协议异常时主动退出并报告未解决频道，不以程序正常结束代替任务完成。

6 问题一：有界误差交会定位与覆盖圆

6.1 从示向度到前向角域的半平面模型

交会定位的第一个困难在于，带误差的示向度只限制方向而不唯一确定位置。由题设可知，示向度误差不超过 $\pm 1^\circ$ ，故给定检测点 $s_i = (s_{ix}, s_{iy})$ 与读数 θ_i 后，真实方位必然落在 $[\theta_i - \delta, \theta_i + \delta]$ 内。据此，真实位置属于以前向方向 $u(\theta_i)$ 为中轴的角域

$$W_i = \left\{ g : \begin{cases} [u(\theta_i - \delta), g - s_i] \geq 0, \\ [u(\theta_i + \delta), g - s_i] \leq 0, \\ u(\theta_i) \cdot (g - s_i) \geq 0 \end{cases} \right\}. \quad (2)$$

三条不等式的意义并不相同。前两条把位置限制在左右边界之间，第三条显式保留“只在前方”这一方向性，从而使零误差的退化情形对应一条射线而不是整条直线。为便于统一求解，将其改写为 $ax + by \leq c$ 的形式，法向依次为

$$\begin{aligned} &(\sin(\theta_i - \delta), -\cos(\theta_i - \delta)), \\ &(-\sin(\theta_i + \delta), \cos(\theta_i + \delta)), \\ &(-\cos \theta_i, -\sin \theta_i), \end{aligned} \quad (3)$$

右端为相应法向与 s_i 的内积。采用正弦、余弦表示还有一个工程上的好处： 0° 与 360° 的跨界无需特判即可自动处理。

由于所有角域都是凸的半平面交，多个测点的信息可以直接合并。问题一的多边形定位区域因此定义为

$$P = \bigcap_{i=1}^n W_i = \{g : Ag \leq b\}. \quad (4)$$

式 (4) 根据各次测向角域确定问题一的定位区域；后续问题将在此基础上加入圆域 Ω 与接收距离约束。值得先说明的是，半平面交是闭凸集，可能为空集、单点、线段、有界多边形或无界集，这些情形必须分别处理，否则“直径”与“覆盖圆”都没有定义。

6.2 区域构造、空集及无界情形

要求解式 (4)，只需枚举约束边界之间的交点再筛选不可行者。令约束数 $k = 3n$ ，对任意两条不平行边界 $ax + by = c$ 、 $dx + ey = f$ ，候选交点为

$$x = \frac{ce - bf}{ae - bd}, \quad y = \frac{af - cd}{ae - bd}. \quad (5)$$

仅保留满足全部不等式的交点，再经去重和单调链凸包构造即得顶点集合。数值上，实现先平移与缩放坐标，以降低千米尺度坐标带来的误差；同时保留近平行边界的求交，因为其远端交点可能决定细长区域的直径。

顶点集确定之后，还需判断区域是否无界。为此对非空集合定义衰退锥

$$\mathcal{R} = \{d : Ad \leq 0\}. \quad (6)$$

若 $\mathcal{R}$ 含非零向量，则 $g + td \in P$ 对所有 $t \geq 0$ 成立，区域无界、直径为 $+\infty$ 。二维多面锥只需检查约束边界的两个切向 $(b, -a)$ 与 $(-b, a)$ 是否满足全部齐次不等式。由于本题至少一个前向窄角域会排除直线性，非空交集必存在极点，故在该输入族中不存在可行边界交点却判为空集的情况；无观测输入单独返回整个平面，不按空集处理。

有界非空区域中，一个顶点对应单点，两个顶点对应线段，其余为凸多边形。空集没有本文定义的有效直径与覆盖圆，程序以显式状态报告观测不相容，便于检查频道、误差界或输入数据。

6.3 直径与直径圆的判定

题目要求计算定位区域直径，并判断以该直径为直径的圆能否覆盖整个区域。对前者，可以证明无需在区域内部搜索。

命题 1 (顶点直径). 设 $P = \text{conv}\{v_1, \dots, v_m\}$ 非空有界，则

$$D(P) = \max_{i,j} \|v_i - v_j\|. \quad (7)$$

证明. 任意 $x, y \in P$ 可表示为凸组合 $x = \sum_i \lambda_i v_i$ 、 $y = \sum_j \mu_j v_j$ 。由于系数非负且各自和为 1，

$$\|x - y\| = \left\| \sum_{i,j} \lambda_i \mu_j (v_i - v_j) \right\| \leq \sum_{i,j} \lambda_i \mu_j \|v_i - v_j\| \leq \max_{i,j} \|v_i - v_j\|.$$

最远顶点对本身属于 P ，故上界可达。 $\square$

由命题可知，只需枚举顶点对即可，计算量为 $O(m^2)$ 。至于第二个问题，结论比直觉更严格：最远顶点对的中点未必能覆盖整个区域。

命题 2 (直径圆覆盖的充要条件). 取最远顶点对 a, b ，令 $D = \|a - b\|$ 、 $o = (a + b)/2$ 。存在直径等于 D 的圆盘覆盖 P ，当且仅当

$$\max_{v \in V(P)} \|v - o\| \leq D/2. \quad (8)$$

证明. 若半径 $D/2$ 的圆盘中心为 z 且包含 a, b ，则 $D \leq \|a - z\| + \|z - b\| \leq D$ 。等号迫使 $z = o$ 。反之，圆盘为凸集，包含全部顶点即包含它们的凸包。 $\square$

证明过程同时说明：圆心被 a, b 唯一确定，因此只需逐一检查其余顶点到该中点的距离。若条件不成立，输出圆外顶点及超出量 $e = \max_v \|v - o\| - D/2$ ，即得到可独立检查的否定证据。换言之，不能仅以直径端点在圆周上判断整个区域被覆盖。

图 5(a)(b) 给出一组三点交会算例及其实例区域的放大，(b) 中虚线为直径圆、实线为最小包围圆。该算例的定位区域由 5 个顶点确定，直径为 22.187 m，以该直径为直径的圆半径为 11.094 m，而最远顶点到圆心的距离超出该圆 4.140 m，故直径圆并不覆盖定位区域；其最小包围圆半径为 11.677 m，满足 20 m 的清除判据。这说明直径不超过 40 m 与能否清除是两个不同的判据，覆盖判定必须逐顶点检查。

6.4 反例与可清除半径

上述差别并非数值巧合，而是几何上的必然。为说明这一点，构造等边三角形

$$A = (0, 0), \quad B = (40, 0), \quad C = (20, 20\sqrt{3}). \quad (9)$$

其直径为 40 m。以 AB 为直径的圆心为 $(20, 0)$ ，半径为 20 m，但 C 到圆心的距离为 $20\sqrt{3}$ m，位于圆外。其最小包围圆圆心为 $(20, 20/\sqrt{3})$ ，半径 $40/\sqrt{3} \approx 23.094$ m，同样大于清除半径。图 5(c) 展示两类圆的差别。

该三角形并非人为设定的特例，它可以由题设的 $\pm 1^\circ$ 角域构造出来。例如检测点 $(-1000, 0)$ 、 $(540, -500\sqrt{3})$ 、 $(520, 520\sqrt{3})$ ，对应示向度为 1° 、 121° 、 241° 。各角域的一侧边界给出三角形的三条支撑线，另一侧边界包含整个三角形，故交集恰为上述三角形。

既然覆盖判据不足以保证清除，就需要一个与清除动作直接对应的指标。为此对一般候选区域定义最小包围圆

$$\rho(P) = \min_{z \in \mathbb{R}^2} \max_{g \in P} \|g - z\|, \quad m \in \arg \min_z \max_{g \in P} \|g - z\|. \quad (10)$$

其边界由至多三个支撑点确定，可用 Welzl 随机增量算法计算 [5]。实现采用固定随机种子及显式栈，并在输出前检查全部顶点的包含关系。在精确模型中，若 $\rho(P) \leq 20$ ，于

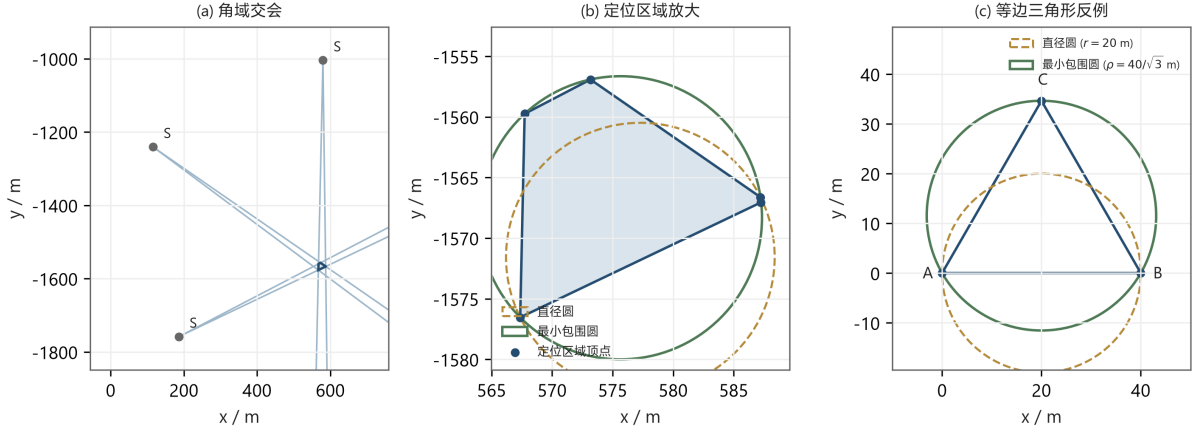


图5 问题一的几何算例。(a) 三点交会：三条浅色示向度的角域边界与它们的交集（定位区域），观测取自建模模拟器演练局 8VRC-ADW4-XDXM-F62L 的频道 12；(b) 该定位区域的放大：虚线为直径圆、实线为最小包围圆，几何量由生产求解器 `mcm26b.geometry.solve` 现算；(c) 等边三角形解析反例：虚线直径圆 $r = 20$ m 不覆盖顶点 C ，实线最小包围圆 $\rho = 40/\sqrt{3} \approx 23.094$ m 覆盖全部顶点，该图不是模拟器观测。

m 清除即可覆盖所有可能真源；考虑浮点计算后采用 $\rho(P) + \eta_{\text{num}} \leq 20$ 的充分判据。由此可见， $D(P) \leq 40$ 只是必要条件，并非充分条件。

6.5 算法一与复杂度

将上述判定串成流程，即得问题一的求解算法。

算法 1 有界测向角域交会与覆盖圆判定

输入： 同一频道的检测点、示向度和误差半角

输出： 区域状态、顶点、直径、直径圆判定与最小包围圆

- 1: **if** 观测为空 **then**
 - 2: **return** 无界
 - 3: **end if**
 - 4: 归一化坐标，生成 $3n$ 条半平面约束
 - 5: 枚举非平行边界交点，仅保留满足全部约束的点
 - 6: **if** 无可行点 **then**
 - 7: **return** 空集
 - 8: **end if**
 - 9: 检查衰退锥；若存在非零方向，**return** 无界及无穷直径
 - 10: 对可行交点去重并求凸包，区分单点、线段和二维多边形
 - 11: 枚举最远顶点对，计算 D 与中点 o ，检查全部顶点是否在 $\overline{B}(o, D/2)$ 内
 - 12: 计算最小包围圆 $\overline{B}(m, \rho)$ ，复核包含关系与数值余量
 - 13: **return** $\rho + \eta_{\text{num}} \leq 20$ 是否成立
-

边界交点枚举需 $O(k^2)$ 次，每点校验 k 条约束，主计算量为 $O(k^3)$ ；凸包最坏需 $O(k^2 \log k)$ ，直径枚举需 $O(m^2)$ 。Welzl 算法在随机排列下具有期望线性时间。该求解流程适合少量观测，当前入口限制最多 100 个检测点；更大规模可替换半平面交算法。

综上，问题一给出两点结论：1. 带误差交会定位的直径可由顶点对枚举精确得到，覆盖判定则必须逐顶点检查；2. 与清除动作真正对应的判据是最小包围圆半径不超过 20 m，直径约束只是它的必要条件。

7 问题二：保证接收的第二检测点选择

7.1 候选位置集合与安全区域

问题一说明了如何由多次观测确定位置，问题二则提出一个更早的问题：在只测过一次的情况下，第二点应当选在哪里。这里存在一对相互牵制的目标：若只追求大交会角，第二点可能远离源而收不到信号；若只求保证接收，又可能缺少横向基线。要同时处理两者，第一步是先刻画“由首次观测能推断出哪些位置”。设首次有效示向度为 (s, θ) ，则有效读数除限定角域外，还说明源距检测点不超过接收半径上界，故

$$K_1 = \Omega \cap W(s, \theta, \delta_{\text{eff}}) \cap \overline{B}(s, R_+). \quad (11)$$

其中 $R_+ = 1500$ 为有效接收半径上界。direction 对应距离大于 5 m；式 (11) 采用其凸松弛，使真实可行位置始终包含在所维护的候选集合中。采用凸松弛而非精确的“挖去 5 m 小圆”，是为了让后续的圆盘交运算保持凸性。

有了候选集合，第二步是把“保证接收”写成集合条件。对非空紧集 K 定义

$$F_K(q) = \max_{g \in K} \|q - g\|, \quad \mathcal{Q}(K) = \{q : F_K(q) \leq R_-\} = \bigcap_{g \in K} \overline{B}(g, R_-). \quad (12)$$

该区域内的任何点均在所有可能真源的最小接收范围内。 F_K 是凸函数且为 1-Lipschitz，因而 $\mathcal{Q}(K)$ 是闭凸集。由式 (10) 可知

$$\mathcal{Q}(K) \neq \emptyset \iff \rho(K) \leq R_-. \quad (13)$$

这一步把第二点的鲁棒可行性转化为问题一已经解决的最小包围圆判据 [6]，两个问题由此共用同一套几何工具。

在线执行时，还需把解析区域换成可快速裁剪的表示。为此用外切正多边形表示圆盘，再逐半平面裁剪。若边数为 M 、圆半径为 r ，外切多边形顶点的半径为 $r \sec(\pi/M)$ ，其最大径向超出量准确写为

$$e_M(r) = r(\sec(\pi/M) - 1). \quad (14)$$

默认 $M = 192$ 。由于整个圆盘已包含在多边形中，裁剪后得到 $K_1 \subseteq P^{\text{out}}$ 。算法求外包多边形 P^{out} 的最小包围圆 $\overline{B}(m, \rho_{\text{out}})$ 。

在此基础上，由三角不等式可直接得到一个保守但显式的安全盘。充分安全盘为

$$\mathcal{Q}_{\text{in}} = \overline{B}(m, R_- - \rho_{\text{out}} - \eta), \quad \eta = 10^{-3} \text{ m}. \quad (15)$$

任意 $q \in \mathcal{Q}_{\text{in}}$ 与真源 g 满足 $\|q - g\| \leq \|q - m\| + \|m - g\| \leq R_- - \eta$ 。因此第二次检测一定为 `direction` 或 `near`。该构造适用于 $\rho_{\text{out}} + \eta \leq R_-$ ；当候选区域较大时，程序先在其圆心追加观测，再更新可行区域。

7.2 首次截断角域的解析候选盘

上述安全盘依赖离散多边形，只能在程序内使用；题目还要求给出解析的候选区域。为此暂去掉 Ω ，考虑 $S = W(s, \theta, \delta) \cap \overline{B}(s, R_+)$ ，它的形状是一个被圆弧截断的角域。当 $0 \leq \delta \leq 45^\circ$ ，令

$$a = \frac{R_+}{2 \cos \delta}, \quad c = s + a u(\theta). \quad (16)$$

即可得到该区域的最小包围圆。

命题 3 (截断角域最小包围圆). 上述 S 的最小包围圆为 $\overline{B}(c, a)$ 。

证明. 任取 $g = s + tu(\theta + \varphi)$ ，其中 $0 \leq t \leq R_+$ 、 $|\varphi| \leq \delta$ 。有 $\|g - c\|^2 = t^2 + a^2 - 2at \cos \varphi$ 。该式关于 t 为凸二次式，最大值在区间端点取得。 $t = 0$ 时为 a^2 ； $t = R_+$ 时不超过 $R_+^2 + a^2 - 2aR_+ \cos \delta = a^2$ ，故全域被包含。

再取边界三点 $s, p_{\pm} = s + R_+ u(\theta \pm \delta)$ 。权重 $\lambda_{\pm} = 1/(4 \cos^2 \delta)$ 、 $\lambda_0 = 1 - 1/(2 \cos^2 \delta)$ 非负且和为 1，其加权中心为 c 。三点均距 c 为 a ，对任意圆心 z ，加权平方距离等于 $a^2 + \|z - c\|^2$ ，故至少一点距 z 不小于 a 。任何包含三点的圆半径均不小于 a 。□

由于 $K_1 \subseteq S$ ，于是

$$q \in \overline{B}\left(s + \frac{1500}{2 \cos \delta} u(\theta), 1000 - \frac{1500}{2 \cos \delta}\right) \quad (17)$$

即为题目所求的一种明确候选区域，实际使用时再内缩 1 mm。该圆盘构成 $\mathcal{Q}(K_1)$ 的一个显式凸子集。表 3 给出两种误差半角下的具体数值。可以看到，误差从 1° 放宽到 1.005° 时，圆心偏移仅增加约 0.001 m，候选盘半径仅减少约 0.001 m，说明该判据对题设的舍入余量并不敏感。

表 3 解析安全盘参数，由式 (17) 直接计算

误差半角	中心沿首次方向偏移 a/m	候选盘半径/m
1°	750.114246	249.885754
1.005°	750.115391	249.884609

在线算法继续使用 Ω 裁剪后的完整候选形状，通常能够获得更紧的源包围圆与更大的充分安全盘。两者的分工是清楚的：解析公式负责解释保证为何存在，多边形流程负责利用已有的全部几何约束。

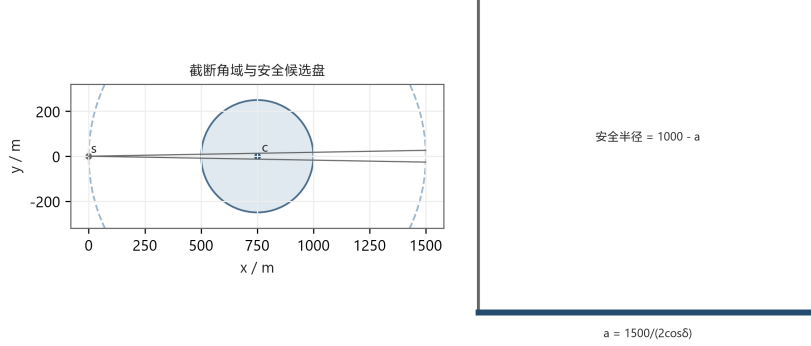


图 6 第二点解析候选区域。左图为首次截断角域 S 、其最小包围圆 $\bar{B}(c, a)$ 与保证接收的候选盘 $\bar{B}(c, 1000 - a)$ ；右图为半径分解， $a = 1500 / (2 \cos \delta)$ 。取 $\delta = 1^\circ$ 时 $a \approx 750.114$ m、 $1000 - a \approx 249.886$ m，与表 3 一致。

7.3 定位效果指标与有限候选求解

保证接收只解决了一半问题，另一半是”在安全区域内如何让第二次观测最有效”。理想的鲁棒选点模型为

$$\min_{q \in \mathcal{Q}_{in}} \left(\sup_{y \in \mathcal{Y}(q, K_1)} \rho(K_1 \cap O(q, y)), \frac{\|q - p\|}{5} \right)_{\text{lex}}, \quad (18)$$

其中 p 是当前机器人位置， y 表示可能观测， $O(q, y)$ 为相应观测可行集，下标表示按字典序比较：先最小化最坏后验半径，再减少移动耗时。方向分支取相应角域，近距离分支取 $\bar{B}(q, 5)$ 。这比仅优化交会角更直接地对应最终清除需求。

连续极小极大问题求解成本较高，当前算法采用有限候选近似。在式 (15) 的盘心及半径比例 0.4, 0.75, 1.0 的三个环上各取 12 个方向，角度为 $(k + \frac{1}{2})2\pi/12$ ，共 37 点。对每个候选 q ，在 P^{out} 边界以不大于 60 m 的间距采样，包含全部顶点。对边界采样点 g 枚举读数 $\text{bearing}(q, g) \pm \delta_{\text{eff}}$ ，计算新角域交后的最小包围半径，并另外计算 near 分支；取所得最大值作为 $\hat{J}(q)$ 。

选取使 $(\hat{J}(q), \|q - p\|/5)$ 最小的点，然后将边界采样间距加密至 15 m 复核其评分。需要强调的是， $\hat{J}$ 是最坏后验半径的离散近似，只用于比较有限候选点的定位效果；接收条件由式 (15) 保证，不受该近似影响。

若候选数为 N_q 、边界样本数为 N_g 、多边形顶点数为 m ，一次裁剪为 $O(m)$ ，总计算量主要为 $O(N_q N_g (m + C_{\text{MEC}}(m)))$ 。该算法复用问题一的角域模型与最小包围圆方法，通过多边形裁剪逐次更新候选集。

算法 2 保证接收的第二检测点选择

输入： 首次测点与示向度，或已更新的候选凸外包；当前位置

输出： 下一检测点、采样后验评分与接收保证状态

- 1: 初次调用时构造 K_1 的外包多边形；后续调用沿用已有外包
 - 2: 计算最小包围圆 $\bar{B}(m, \rho_{\text{out}})$ 与安全盘半径 $b = 1000 - \rho_{\text{out}} - 10^{-3}$
 - 3: **if** $b < 0$ **then**
 - 4: 在候选区域圆心追加观测
 - 5: **else**
 - 6: 生成 37 个安全候选点
 - 7: **for** 每个候选点 q **do**
 - 8: 枚举外包边界样本与方向端点读数，补入近距离分支
 - 9: 计算最坏后验半径 $\hat{J}(q)$
 - 10: **end for**
 - 11: 按后验评分优先、移动时间次优选点，并以 15 m 采样间距复核
 - 12: **end if**
 - 13: 获得新观测后更新候选集
 - 14: **if** 观测为 near **then**
 - 15: 就地清除
 - 16: **else**
 - 17: 根据最小包围半径决定清除或追加测量
 - 18: **end if**
-

综上，问题二得到两点结论：1. ”对所有可能真源均保证接收”等价于候选集合的最小包围圆半径不超过 1000 m，据此可给出半径约 249.885 m 的显式安全候选盘；2. 在安全盘内以最坏后验半径为第一目标、移动时间为第二目标选点，既避免了”收不到信号”的失败模式，又保证了交会基线。

8 问题三：全向多源的网格滤波与滚动调度

8.1 统一时间目标与任务状态

问题一、二分别给出了”能否清除”与”下一点选在哪里”的判据，问题三把它们放进一个连续运行的闭环：源的数量未知，机器人必须自己决定何时发现、何时定位、何时清除。在这样的任务中，优化的最终标准不是某一步的精度，而是完成任务所花的虚拟时间，因此第一步是把时间写成可优化的目标。按式 (1) 的计时记账，优化目标是在全部源清除的约束下最小化虚拟耗时 T 。题目统计的平均定位清除时间是 $\bar{T} = T/N_c^+$ ；由于清除动作本身的 5 s 是常数，该平均值不能代替总虚拟时间作为优化目标。清除请

求中的频道是目标频道，不改变测向机频道；进入与退出均不增加虚拟时间，因此路径终点自由，无需返回原点。这一自由度允许我们把任务序列当作开放路径而非回路来处理。

在时间目标之外，还需要一套能表达“这条频道到底还有没有源”的状态。为此各频道的状态按“未知、已发现、可清除、已清除、已证无源”管理。首次获得 direction 或 near 即确立该频道存在源。最小包围圆判据决定清除时机，成功响应将状态更新为已清除。频道的候选集为空，或其所有覆盖站检测结果均为无信号时，可判定该频道无源。后一条判据依赖下一小节的覆盖证明，是本问题效率的关键：只有能够严格举证“不存在”，才能放心地停止检测。

当前测试程序默认采用 BayesLoop：七点计划只作为解析覆盖与无源举证；证据层同时维护便于判据计算的凸候选集与逐频道网格幸存集，后者统一吸收正负观测，负观测按最小接收半径删除整格，候选集为空即给出缺席证明。定位测量按信息增益与移动代价选点，覆盖、定位与清除三类任务组成统一任务表，用开放路径 2-opt 滚动重排，并不执行七个站点对所有频道逐一全扫描的固定流程。

8.2 七点解析覆盖与无源证据

“证明某频道无源”要求对圆域内任意位置都保证有站点能收到信号。为达到这一目的，取站点集合

$$S(a) = \{(0,0)\} \cup \left\{ a \left(\cos \frac{k\pi}{3}, \sin \frac{k\pi}{3} \right) : k = 0, \dots, 5 \right\}. \quad (19)$$

即原点加正六边形顶点，共七点。其覆盖能力可以精确刻画。

命题 4 (七点全向覆盖). 当 $0 < a/\sqrt{3} \leq R$ 时，任意 $g \in \Omega$ 到最近站点的距离不超过

$$D_{\text{cov}}(a) = \max \left\{ \frac{a}{\sqrt{3}}, \sqrt{R^2 + a^2 - \sqrt{3}Ra} \right\}. \quad (20)$$

证明. 设 g 距原点为 r ，其方位与最近环站的夹角不超过 $\pi/6$ 。若 $r \leq a/\sqrt{3}$ ，选取原点即可。若 $r \geq a/\sqrt{3}$ ，由余弦定理，距最近环站的平方不超过 $r^2 + a^2 - \sqrt{3}ar$ 。该凸二次式在 $[a/\sqrt{3}, R]$ 上的最大值于端点取得，分别为 $a^2/3$ 与 $R^2 + a^2 - \sqrt{3}aR$ 。□

命题给出了一个可调的覆盖半径。当前 bayes 策略取 $a = 1159.7$ m，由式 (20) 得 $D_{\text{cov}} = 984.539136$ m，比最小接收半径小约 15.461 m。因此每个合法全向源至少在一站可被接收。某频道在七站全部为无信号，就构成该频道不存在源的严格证据。以“站点、频道”为索引建立检测状态矩阵，记录各站对各频道的检测结果。该矩阵用于逐频道的严格缺席证明；实际执行不要求在每个站点重测所有频道，可由网格幸存集证明必为无信号的（站，频道）对会被跳过。

顺次从原点访问六个环站的开放路径长为 $6a$ ；任务总路径由覆盖站间移动以及定位、清除所需的局部移动共同构成。

8.3 正负观测的双重候选表示

覆盖证明只回答“源是否可能被听见”，要把一次具体检测转化为位置信息，还需要更新候选集合。每个已发现频道维护凸外包 P_c^{out} 。direction 将候选集与 $\bar{B}(s, 1500)$ 外切多边形及测向角域相交；near 与 $\bar{B}(s, 5)$ 外切多边形相交。难点在于全向源的 no_signal 观测，它意味着

$$\|g - s\| > r_{\text{eff}} \geq 1000, \quad (21)$$

即排除一个圆盘。圆盘的补集不凸，直接与凸候选集求交会破坏凸性。为此，采用网格幸存集 $\mathcal{G}_c$ 表示负观测带来的空间约束，同时保留便于几何运算的凸外包：前者负责吸收负信息，后者负责快速判据与选点。

网格边长为 $\ell = 14.65$ m，每格中心与格内点的最大距离为 $\epsilon_g = \ell/\sqrt{2}$ 。初始保留所有可能与 Ω 相交的格，按以下充分排除条件更新格心 z ：

$$\begin{aligned} \text{no_signal: } \|z - s\| \leq 1000 - \epsilon_g &\Rightarrow \text{删除;} \\ \text{near: } \|z - s\| > 5 + \epsilon_g &\Rightarrow \text{删除;} \\ \text{direction: } \|z - s\| > 1500 + \epsilon_g &\Rightarrow \text{删除.} \end{aligned} \quad (22)$$

三式的裕量 ϵ_g 保证只有整格必然越界时才删除，故真源所在格永不被误删。方向观测另外检查角差。若 $d = \|z - s\| > \epsilon_g$ ，源所在格引起的方位偏差可由 $\arcsin(\epsilon_g/d)$ 上界；实现采用距离内缩后的角误差界，并在测点附近保留所有可能方位。只有角差超过 δ_{eff} 加格内角裕量时才删除。

上述排除条件保持真源所在网络的可行性。因此网格幸存集为空时，该频道无源。若全部候选格心均满足 $\|z - s\| > 1500 + \epsilon_g$ ，则在 s 测量必无信号，且所有候选网格已在最小接收圆盘外，其观测的信息增益为零，可在检测状态矩阵中标记该推断结果，从而节省相应检测与换频时间。

8.4 信息增益选点与清除判据

候选集收窄之后，还需决定下一个测点。问题二的算法 2 提供满足接收约束的检测点；问题三的 BayesLoop 进一步采用网格幸存集上的信息增益，即优先选择最能缩小候选集的测点。存在保证接收圆盘时，在其内部生成环形候选并补充当前接近方向上的测点；其余情况下，在网格最小包围圆附近生成环形采样点。候选集合同时包含当前位置与探索点，分别根据预期信息增益和移动代价评分。对候选 q 计算

$$J_{\text{IG}}(q) = \widehat{\mathbb{E}} \left[\log \frac{|\mathcal{G}_c^{\text{post}}(q, Y)|}{|\mathcal{G}_c|} \right] + \lambda \left(\frac{\|p - q\|}{5} + \frac{\|q - m_c\|}{5} \right), \quad (23)$$

其中 $\lambda = 0.05$ ，空分支用正的小值截断。较小的候选集基数比对应较大信息增益，移动项限制为测量而产生的绕行。评分最多取 800 个子样本及 7 个代表假设，并以均匀

$r_{\text{eff}} \in [1000, 1500]$ 估计接收概率。该决策采用离散假设上的加权期望，接收半径的均匀分布作为选点先验。

清除判断使用完整候选集合。由于清除要求在 20 m 内完成，判据必须同时计入几何半径与数值余量。凸候选集的清除判据为

$$\rho_c + b_c + 0.5 \leq 20, \quad (24)$$

其中 $b_c \geq 0$ 为代码的累计附加余量。凸多边形包含真源， ρ_c 是其最小包围半径； b_c 与 0.5 m 共同提供附加数值裕量。若负观测使网格更紧，也可使用

$$\rho_{\text{grid}} + \epsilon_g + 0.5 \leq 20 \quad (25)$$

的网格清除判据，其中 ρ_{grid} 为候选格心的最小包围半径。满足任一充分条件即可选择清除点。利用剩余距离裕量，可将圆心向当前位置平移 Δ ；只要平移后的距离上界仍满足对应不等式，三角不等式便保证清除条件成立。

8.5 开放路径上的滚动任务排序

到这一步，每个频道都已知”是否需要检测、在哪里检测、何时清除”，剩下的问题是如何把这些动作排成一条省时的路线。统一任务表包含三类任务：覆盖站的批量扫频、已发现源的定位测量、满足半径条件的清除。站内检测优先使用测向机当前频道，减少换频次数。对任务序列 π ，排序代价为

$$C(\pi) = \frac{\|p - q_{\pi_1}\|}{5} + \sum_{j=1}^{k-1} \frac{\|q_{\pi_j} - q_{\pi_{j+1}}\|}{5} + \sum_{j=1}^k t_{\pi_j}(c_{j-1}), \quad (26)$$

其中服务时间 t_{π_j} 依赖进入任务时的频道，清除不改变该频道。从最近邻初序开始，默认最多进行 4 轮开放路径 2-opt 改进；每次只执行当前序列的首任务，得到新观测后重新生成和排序。这样新发现的源、收缩后的清除点和已证无源的频道都能及时改变路线，避免为一条已经失效的任务继续赶路。

已停靠位置可以顺带补测：要求此前未在该位置测过、距离保证成立、尚处于定位阶段，且预测半径有足够收缩。追加测量的费用为检测与换频时间，其收益来自节省额外移动。题设固定同地误差，因此优先选择新的检测位置。

8.6 算法三与完成条件

将上述模块串成闭环，即得问题三的求解算法。

”清除数达到 16”利用题设数量上界。对任意已存在的 16 个不同频道，也可提前排除其余未知频道，但已发现而未清除的源仍必须继续处理。

覆盖条件用于判定搜索范围，清除充分条件用于确定操作位置。当前每轮追加有效定位测量上限为 8；停滞、协议异常或预算耗尽时，记录各频道状态并终止。程序以接

算法 3 网格幸存集滤波、信息增益选点与滚动调度

输入： 题设参数、七站计划与合法观测接口

输出： 逐频道处理状态与会话统计

- 1: 检验七点覆盖条件，初始化频道状态、凸候选集、网格幸存集和检测状态矩阵
 - 2: 调用 */enter*，记录位置、频道和剩余现实预算
 - 3: **while** 存在未处理频道且预算充足 **do**
 - 4: 构建覆盖、定位和清除三类任务
 - 5: 用网格幸存集跳过必为无信号的检测
 - 6: 按最近邻与开放路径 2-opt 排序任务表
 - 7: 执行首任务，根据真实响应更新凸候选集与网格幸存集
 - 8: **if** 满足清除判据 **then**
 - 9: 执行清除；成功则更新状态，失败则追加观测
 - 10: **end if**
 - 11: **if** 已清除 16 个源或所有频道均被清除或证明无源 **then**
 - 12: **return** 报告完成状态
 - 13: **end if**
 - 14: **end while**
 - 15: 预算不足或异常时退出并报告未解决频道
-

口返回的现实预算为准，并预留退出时间，实际计时在整数微秒状态上累计，避免用实机运行秒数代替虚拟动作耗时。

综上，问题三得到三点结论：1. 七点计划在环半径 1159.7 m 时给出 984.539 m 的覆盖上界，使”全站无信号”成为该频道缺席的严格证据；2. 网格幸存集把非凸的负观测纳入集合估计，并使大量必然无信号的检测被跳过；3. 把覆盖、定位、清除统一为一条开放路径并滚动重排，是压缩总虚拟时间的主要手段。

9 问题四：混合源的前向半盘覆盖与联合信念

9.1 定向观测模型与无信号歧义

在问题三中，”无信号”只有一种解释：源太远。一旦引入定向源，这唯一的解释就被打破——源可能只是背对着检测点。因此本问的关键在于为无信号保留第二种解释。设定向源位于 g ，朝向为 ψ 。在检测点 q 可见的充要条件为

$$\|q - g\| \leq r_{\text{eff}}, \quad (q - g) \cdot u(\psi) \geq 0. \quad (27)$$

边界属于可见半平面。可见且距离不超过 5 m 时返回 *near*，否则返回 *direction*；任何一个条件失败均为 *no_signal*。问题二的可行区域满足距离约束，定向接收还需满足前

向半平面约束。对于已发现频道，位于该区域内的负观测反映朝向遮挡，因此不应被当作距离证据；区域外的负观测则同时包含距离与朝向两种解释。这一区分直接决定了后续信念表示的形式。

朝向约束也改变了覆盖的要求。问题三的七点计划只保证“至少一站距离足够近”；对定向源还要求该站位于源的前向半平面内，因此需要重新设计站点集。向外发射的边界源需要在圆域外布置接收站点。适用于混合源的站点集合 S 应满足

$$\forall g \in \Omega, \forall \psi \in [0, 2\pi), \exists s \in S: \quad \|s - g\| \leq 1000, \quad (s - g) \cdot u(\psi) \geq 0. \quad (28)$$

这是一项针对“位置、朝向”二元状态的全覆盖要求，而不是对位置的单变量覆盖。

问题三的七点计划不满足式 (28)；当前测试程序改用 31 点最小命中集与（位置 × 朝向）联合幸存集，定位停滞时以有限格后备清除兜底。

9.2 方格点集的连续覆盖性

在设计具体站点之前，先考虑一个容易证明的结构：方格点集。取 $S_{49} = \{(700i, 700j) : i, j = -3, \dots, 3\}$ 。其网格方域包含 Ω ，每个源落在某个边长为 700 m 的闭方格内。方格四角恰好同时满足距离与朝向两项要求。

命题 5 (方格四角的定向覆盖). S_{49} 满足式 (28)，对全向与任意朝向的定向源均保证发现。

证明. 设包含 g 的方格四角为 $v_1, \dots, v_4$ 。四角距 g 均不超过对角线长 $700\sqrt{2} < 1000$ 。由于 g 是四角的凸组合，存在 $\lambda_i \geq 0$ 、 $\sum_i \lambda_i = 1$ 使 $\sum_i \lambda_i (v_i - g) = 0$ 。对任意朝向 u 取内积，得 $\sum_i \lambda_i (v_i - g) \cdot u = 0$ ，故不可能四个角均满足 $(v_i - g) \cdot u < 0$ ；至少一角处于前向闭半平面，且同时满足距离条件。 $\square$

证明的关键是凸组合性质：四个方向的投影不可能同时为负。该方案具有显式站点坐标和覆盖距离上界。不过在同样覆盖 Ω 的前提下，网格方案的站点数并不经济，因此实际选点仍需借助集合覆盖模型。

9.3 前向半盘最小命中集的 31 点方案

把“每个位置与朝向组合至少被一个站点覆盖”写成整数规划，就可以用远比网格稀疏的站点集完成同样的任务。当前 directional 实现采用 q4_mincover_v2 计划。在候选站点 s_j 和离散目标状态 (g_i, ψ_i) 上定义关联矩阵

$$A_{ij} = \mathbf{1}\{\|s_j - g_i\| \leq r_{\text{design}}, (s_j - g_i) \cdot u(\psi_i) \geq \|s_j - g_i\| \sin \gamma_{\text{design}}\}. \quad (29)$$

其中角度余量 γ_{design} 把“恰好落在可见边界”的情形排除在外，使站点对朝向扰动保持稳健。站点筛选可写为集合覆盖模型

$$\min \sum_j z_j, \quad \sum_j A_{ij} z_j \geq 1 \quad (\forall i), \quad z_j \in \{0, 1\}. \quad (30)$$

采用集合覆盖的贪心算法与冗余站点删除求近似解 [7]，得到 31 个站点。设计阶段使用 965 m 距离与 5° 前向角余量，最终按 15 m 位置间距、72 个朝向采样复核。验证结果覆盖全部 3 256 200 个离散状态，最坏覆盖距离约为 990.051 m。

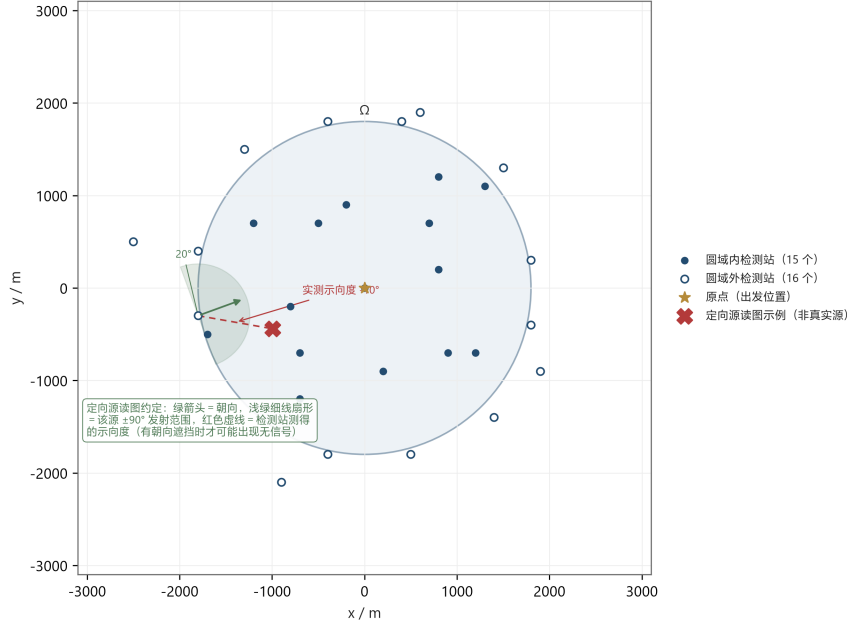


图 7 31 点最小命中集的几何分布与定向源的读图约定。实心圆点为圆域内检测站（15 个），空心圆点为圆域外的检测站（16 个），后者用于接收向外发射的边界源；浅色扇形区域为源位置圆域 Ω 。图中左侧以一个圆外检测站为例说明定向源的观测读法：绿箭头是该源的发射朝向、浅绿细线扇形是它的 $\pm 90^\circ$ 可见范围、红色虚线是检测站在该点测得的示向度，三者共同说明“有朝向遮挡时同一位置也可能测不到信号”。离散校验不等同于连续覆盖证明。

从离散覆盖推广到连续覆盖时，需同时计入位置与朝向的离散化误差。设任意实际位置距某采样位置不超过 ε ，实际朝向距其采样朝向不超过 α ，覆盖站点距采样位置为 d 。由三角不等式及单位向量差的上界，一组充分条件为

$$d + \varepsilon \leq 1000, \quad (s - g_i) \cdot u(\psi_i) \geq \varepsilon + 2d \sin(\alpha/2). \quad (31)$$

采样网格应覆盖 Ω 及其边界。本文对 31 点方案采用离散集合覆盖模型，对 49 点方案采用上述连续覆盖定理；连续近似误差的控制将在未来工作中讨论。

9.4 位置与朝向联合候选的保守更新

站点解决“能否发现”，而要完成清除还必须知道源在哪里、朝向如何。这正是问题三的候选集表示不足以处理的部分：位置可行并不代表朝向可行。对定向假设，在每个候选位置网格的中心 z 附加 $H = 24$ 个朝向档，档宽为 15° 。令 M_z 为仍可能的朝向掩码。方向与近距离观测先更新位置约束，再保留可能覆盖检测点的朝向；无信号时只有在整格处于 1000 m 保证接收盘内，才可删除必然可见的朝向。

记 $\beta = \text{bearing}(z \rightarrow s)$ 、 $\Delta_\psi = 7.5^\circ$ ，格内方位余量为 α_z ，查表舍入余量为 $\eta_\psi = 1.5^\circ$ 。以 d_o 表示圆周最小角差，保守保留条件为

$$\begin{aligned} \text{正观测: } d_o(\psi_k, \beta) &\leq 90^\circ + \Delta_\psi + \eta_\psi + \alpha_z, \\ \text{盘内负观测: } d_o(\psi_k, \beta) &> 90^\circ - \Delta_\psi - \eta_\psi - \alpha_z. \end{aligned} \quad (32)$$

第二行保留含有不可见方向的朝向区间，排除整个区间均可见的候选。若格心距测点过近，格内方位可覆盖整个圆周，此时保留全部朝向。在定向假设下，上述更新保持真源所在位置与朝向网格的可行性。

9.5 可见性筛选与探点选择

联合候选的第一个用途是给候选测点排序。在混合场景中，一个测点是否值得去，取决于那里的候选源有多大可能正对着它。对候选 q ，以均匀接收半径假设给出距离权重

$$w_r(d) = \begin{cases} 1, & d \leq 1000, \\ (1500 - d)/500, & 1000 < d < 1500, \\ 0, & d \geq 1500. \end{cases} \quad (33)$$

记 $\tilde{\mathcal{G}}_c$ 为位置样本集， $b_z(q)$ 为与检测点 q 相容的候选朝向档数，采用如下可见性权重：

$$V(q) = \frac{1}{|\tilde{\mathcal{G}}_c|} \sum_{z \in \tilde{\mathcal{G}}_c} w_r(\|z - q\|) \frac{b_z(q)}{24}. \quad (34)$$

分母 24 为初始朝向档数， $b_z(q)$ 由保留朝向集合与预计算可见角区间的交集确定。该权重用于衡量各候选点的相对检测价值，可直接避免”跑过去却因遮挡而无信号”的空跑。

候选族包括算法 2 给出的基础点、关于候选圆心的对跖点，以及 $0.5\rho, \rho, 1.5\rho$ 半径上的八方向环点。排除已被遮挡的重复点后，保留评分达到当前最大值 70% 的点，并从中选择

$$\frac{\|p - q\|}{5} + \frac{\|q - m\|}{5} + 6 \quad (35)$$

最小者。常数 6 是检测加换频的选点评分费用，真实结算仍按当时频道计算。该规则优先避免完全遮挡的点，再减少移动及后续接近距离。联合集不可用时，返回算法 2、对跖探点和环探的确定性顺序。持续遮挡时转入基于距离约束的网格遍历清除。

9.6 网格遍历清除与计算预算

遮挡可能使定位长期无法收缩，此时需要一条不依赖朝向信息的兜底路径。注意到清除是否成功只与距离有关，与源朝向无关，因此可以放弃”先定位再清除”的顺序，改为逐格试探。当一轮有效定位测量达到 8 次，或连续遮挡探点达到 4 次且候选区域仍较大，对保守凸候选多边形建立边长 20 m 的方格覆盖。保留所有与多边形相交的格子，

并以格心作为清除点。格内任一点到格心距离满足

$$\|g - z\| \leq 10\sqrt{2} < 20. \quad (36)$$

因清除与源朝向无关，只要真源保留在多边形中，逐格清除必有一次到达其 20 m 内。按最近邻顺序访问，首次成功即停止该频道的网格遍历。

兜底方案必须受预算约束，否则会挤占其他频道的处理时间。若外包框宽、高为 w, h ，保守格数上界为

$$B_{\text{cell}} = \left(\left\lceil \frac{w}{20} \right\rceil + 2 \right) \left(\left\lceil \frac{h}{20} \right\rceil + 2 \right). \quad (37)$$

当前格数硬上限为 400，现实预算需覆盖 $N_{\text{cell}} t_{\text{RTT},95} + t_{\text{reserve}}$ 。超过格数或预算上限即跳过并报告未解决；全部清除点失败则保留候选集并记录异常。在候选区域包含真源、格数满足上限且预算足够的条件下，逐格访问可保证完成该源的清除。

9.7 算法四与终止条件

算法 4 前向半盘最小命中集、联合信念与后备清除

输入： 混合源接口、定向覆盖计划与预算

输出： 已清除频道、未解决频道与停止依据

- 1: 加载覆盖计划；默认 31 点按 15 m/72 档校核，49 点备选有连续证明
 - 2: **while** 存在未解决频道且预算充足 **do**
 - 3: 选择最近待测站，当前频道优先，批量检测未知频道
 - 4: 更新凸位置候选与位置乘朝向联合幸存集；首次正观测后建立朝向候选
 - 5: **while** 频道尚未清除 **do**
 - 6: 按可见性概率筛选探点，取移动最省者执行测量
 - 7: **if** 满足半径判据 **then**
 - 8: 执行清除并跳出
 - 9: **else if** 遮挡或定位停滞 **then**
 - 10: 检查格数与预算，允许则执行 20 m 方格后备清除
 - 11: **end if**
 - 12: **end while**
 - 13: **end while**
 - 14: 主动退出时报告清除数与停止依据
-

需要说明的是，问题四使用”本站批量扫频、最近候选圆心服务”的调度骨架，并未直接复用问题三的统一信息增益任务表。若要求严格的混合源全清证据，除采用连续覆盖计划外，还须补全类型联合表示或限制联合集排空的适用范围，并确保全部定位清除任务完成。本文区分当前实现、可证明条件与尚需完善之处，不把后者追溯为正式测试时已执行的功能。

综上，问题四得到三点结论：1. 定向源使无信号具有距离与朝向两种解释，候选集必须由位置扩展为位置与朝向的联合表示；2. 发现条件须由位置圆盘覆盖提升为前向半盘覆盖，31 点最小命中集以远少于网格的站点数完成该任务；3. 清除与朝向无关这一物理性质可用于构造有界预算的兜底清除，使遮挡不再导致任务无法完成。

10 实验结果与分析

10.1 实验设置与数据来源

本文的实验分两部分，职责严格区分。机制论证与定量结论取自项目内自建模拟器（referee，按题面附件 1、附件 2 复现的评测机）的演练局：演练局可提供真实源位置、机器人轨迹和逐次检测读数，因此能够支撑路径示意、机制分析与逐局的定量比较。最终效果验证取自官方模拟器的六份正式导出包，因无法获取真实位置点，只在附录 A 中报告清除数量、总虚拟时间与程序运行时间。

演练局由 `scripts/run_practice_episodes.py` 采集：脚本启动评测机、以固定种子开始一局演练，再用与 `mcm26b.cli run` 相同的生产策略对接接口，真值与行为日志全部由评测机自己写入，策略端只看见 HTTP 响应。采集批次共 44 局：问题三 15 局全向源、问题四 29 局混合源，源数覆盖 10 至 16、接收半径覆盖 1000 至 1500 m，问题四的定向源个数覆盖 3、6、8。判定标准为”清完全部源 $\wedge$ 无缺席误判 $\wedge$ 主动退出 $\wedge$ 零异常 $\wedge$ 零失败清除”，本批 44 局全部满足，因此可以直接作为当前实现的性能证据，累计清除 557 个源。所有时间均按式 (1) 的计时模型由动作序列独立复算，复算值与评测机日志记录的最大偏差为 0.000000 s，说明本文的时间口径与模拟器实现完全一致，后文所有耗时数字均可重算。为使”最优实现”的说法可检验，本文对每一问都准备了一个朴素基线，两者在同一算例上逐例配对比较：问题一的基线用包围盒对角线代替真实直径，并以包围盒中心近似覆盖圆；问题二的基线分别取”垂直首次示向度 1000 m 的 90° 交会点”与”候选集最小包围圆圆心”；问题三的基线用 49 点（700 m 步长）网格发现就近服务；问题四的基线则把问题三的两种策略原样搬用。基线并非稻草人：它们在部分场景中有效，暴露的是”缺了哪一步”。

10.2 问题一与问题二的配对对比

先看几何层的两个判据在数值上的表现。问题一用固定示例与随机交会算例（共 5 例固定、40 例随机）对比两种实现，见表 4。

由表可见，最优实现（半平面交 + 顶点对枚举 + 最小包围圆）在固定与随机两类算例上的直径误差均为零、覆盖判定全部正确；基线的直径平均绝对误差为 7.886 m（固定例）与 4.583 m（随机例），最大误差达 14.599 m，覆盖判定正确率仅 33.3% 与 52.5%。原因在于包围盒对角线只有在区域恰好与坐标轴对齐时才等于直径，而基线的覆盖判据几乎恒判”覆盖”，因此系统性偏乐观。计算代价方向相反：最优实现平均 0.086 ms，基

表 4 问题一：直径与覆盖判定的实现对比（时间单位：ms/例）

实现	状态正确率	直径平均误差/m	直径最大误差/m	覆盖正确率	平均耗时/ms
最优（固定 5 例）	100.0%	0.000	0.000	100.0%	0.064
最优（随机 40 例）	100.0%	0.000	0.000	100.0%	0.086
基线（固定 5 例）	80.0%	7.886	10.000	33.3%	14.202
基线（随机 40 例）	100.0%	4.583	14.599	52.5%	15.933

线 15.933 ms，说明精确判据并不以牺牲效率为代价。

问题二的差距出现在”保证接收”与”定位收缩”这对指标上，见表 5。入选标准是：安全盘内最坏后验半径越小越好，同时保证接收率必须为 100%。

表 5 问题二：第二检测点选点策略对比（30 例随机首次示向度）

选点方法	保证接收率	最坏后验均值/m	中位/m	平均收缩/m	终值比	移动均值/s
算法 2（默认选点）	100.0%	73.890	90.899	607.577	0.104	170.44
参考：盘内网格枚举	100.0%	75.627	91.349	605.841	0.106	170.28
基线：垂直 1000 m	0.0%	80.111	90.775	601.356	0.116	200.00
基线：候选圆心	100.0%	340.747	375.088	340.721	0.500	136.29

初始候选半径均值为 681.467 m。由表 5 可知：垂直 1000 m 基线的最坏后验半径与最优实现接近，但保证接收率只有 0.0%——当真源位于角域远端时必然返回无信号，这正是问题二所要避免的失败模式；候选圆心基线的保证接收率为 100.0%，但最坏后验半径达 340.747 m，终值比 0.500，相当于只收缩了一半，说明”保证接收”若脱离横向基线便几乎没有定位价值。最优实现同时取得 100.0% 的保证接收率与 73.890 m 的最坏后验均值，终值比 0.104，并与盘内网格枚举参考解（75.627 m）相差不到 2%。

10.3 问题三：演练局结果与基线对比

问题三的闭环在 18 个冻结场景上逐例配对运行，三种策略的结果见表 6。

表 6 问题三：三种策略在 18 个全向源场景上的对比（时间：s；三种策略的缺席误判均为 0）

策略	有效局	全清率	T 均值	T 中位	T 最大	检测均值	位移/km	后备清除
bayes（默认策略）	18/18	100.0%	3188.734	3111.256	3663.707	111.1	12.338	0.0
seven（七点批量）	18/18	100.0%	5361.707	5367.455	6462.351	100.8	23.508	0.0
grid（49 点网格）	18/18	100.0%	14935.659	15275.057	16691.406	430.6	61.550	0.0

三者的全清率都是 100%，因此差别完全体现在代价上。默认策略 bayes 的平均虚拟时间为 3188.734 s，七点批量策略为 5361.707 s，49 点网格基线为 14935.659 s；相对基线，bayes 的平均降幅为 78.4%、中位降幅 80.0%，并在 18/18 个场景中全部更快。代价的来源也很清楚：默认策略平均只需 111.111 次检测、位移 12.338 km，而网格基线需要 430.556 次检测、61.550 km。可见网格基线的开销主要来自”逐格走过去再检测”，而七点计划加网格幸存集把大量必然无信号的检测直接跳过，再配合滚动任务排序把移动量压到约五分之一。

这一机制在真实演练局中得到复现。表 7 给出问题三全部 15 局全清演练局的直接统计。

表 7 问题三：全向源演练局结果（按总虚拟时间升序）

序号	案例编码	源数	清除数	总虚拟时间/s	移动/km	检测数	换频数	平均清除/(s/个)
1	6LST-ZFVS-2HHW-TQQ4	13	13	2794.212	10.426	109	99	214.939
2	NMRC-85ZV-3FFE-GBXY	13	13	2860.706	10.974	101	96	220.054
3	LZDX-68DJ-XM4P-A6XL	10	10	2954.390	11.072	117	105	295.439
4	QS6H-CC29-PDY3-ZA54	16	16	2987.385	12.042	85	74	186.712
5	MULB-4YTU-VQYG-YAS6	10	10	3064.020	11.250	129	119	306.402
6	6V2P-HJCS-GJ9A-83MA	16	16	3174.578	12.458	103	88	198.411
7	7ZXN-N53X-HAXQ-EGY4	10	10	3228.653	12.393	119	105	322.865
8	VE37-SRPD-NCFR-4EQF	10	10	3240.474	12.277	124	115	324.047
9	HGBA-XGH9-UNZJ-7YQ3	10	10	3340.025	12.810	123	113	334.003
10	JW47-AVPH-XGHH-RLLV	13	13	3389.152	13.506	106	93	260.704
11	8VRC-ADW4-XDXM-F62L	13	13	3407.039	13.695	102	93	262.080
12	MT3R-68KH-CNY5-6X73	13	13	3527.713	14.044	112	94	271.363
13	E27E-M8WQ-LDT5-2XDA	16	16	3531.065	14.210	104	89	220.692
14	R2N7-QMLC-26QX-RXYT	16	16	3556.397	14.257	106	95	222.275
15	ARBP-6XFZ-5YJU-STC8	16	16	3867.872	15.514	116	105	241.742

15 局的源数覆盖 10 至 16，全部实现清除数等于源数，且没有一次失败清除；总虚拟时间集中在 2794.212 至 3867.872 s，平均 3261.579 s，平均定位清除时间 258.782 s/个。随源数增加，总虚拟时间只从约 3.0×10^3 s 增长到约 4.0×10^3 s，说明发现—定位—清除三个闭环的组织方式对源数量并不敏感。

图 8 给出其中一局（QS6H-CC29-PDY3-ZA54，16 个源）的完整路径。机器人先按七点计划完成覆盖与批量扫频，再围绕候选区域执行定位与清除；清除点与源真值基本吻合，轨迹在多个源之间跳跃后逐步收敛。该局共检测 85 次、换频 74 次，移动 12.042 km，总虚拟时间 2987.385 s，程序运行时间仅 2.052 s。

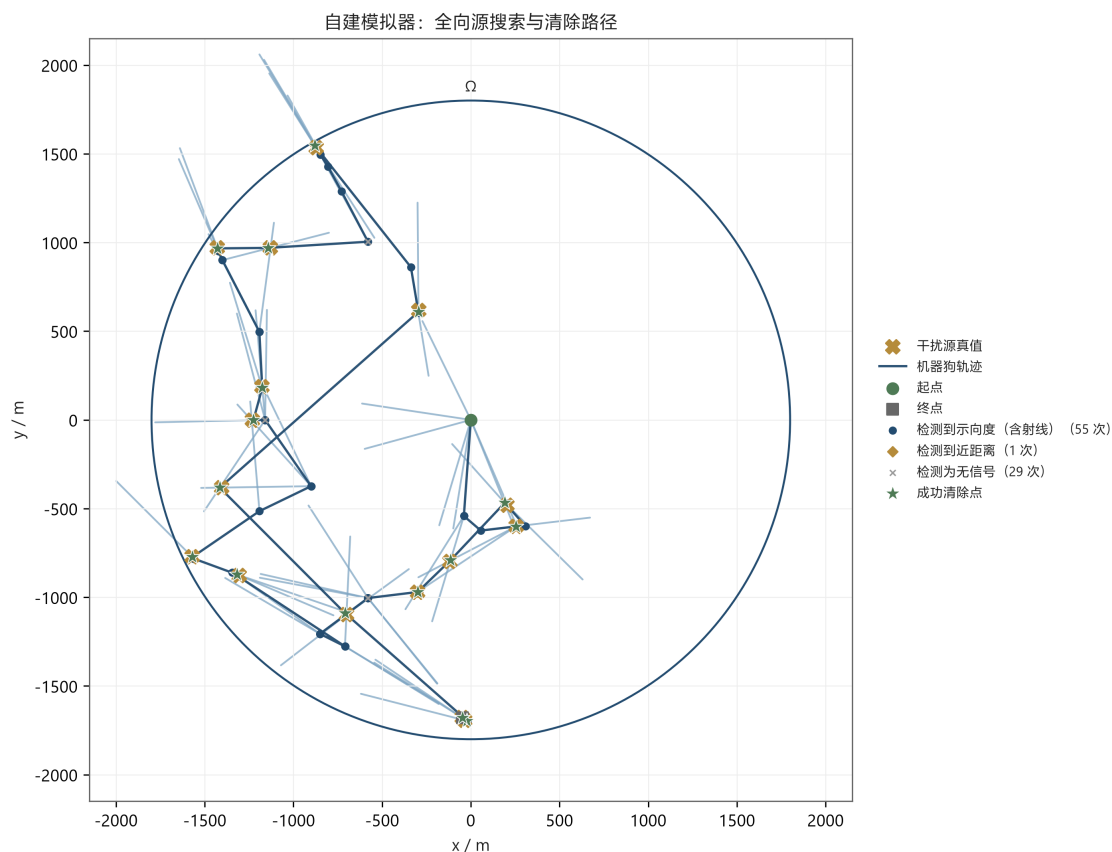


图 8 自建模拟器演练局 QS6H-CC29-PDY3-ZA54（16 个全向源）的搜索与清除路径。折线为机器狗轨迹；每条浅色细射线表示一次 direction 检测测得的示向度，射线起点即发起该次检测的位置；圆点为测到示向度的检测点（55 次），菱形为测到近距离的检测点（1 次），灰色叉号为无信号检测点（29 次）；星号为成功清除点，金色叉号为干扰源真值。全部数据由自建模拟器真值输出。

10.4 问题四：演练局结果与基线对比

问题四的困难来自遮挡，因此对比的重点不只是时间，还包括”是否真的全部清除”。表 8 给出三种策略在 11 个混合源场景（含 5 个边界定向源场景）上的结果。

表 8 问题四：三种策略在 11 个混合源场景上的对比（时间：s）

策略	有效局	全清率	缺席误判	T 均值	T 中位	T 最大	检测均值
directional（默认策略）	11/11	100.0%	0	9069.871	8803.261	12081.148	355.6
grid（问题三网格）	0/11	66.6%	0	18179.173	18345.164	20457.858	573.7
bayes（问题三策略）	0/11	77.0%	24	3890.801	3646.178	5054.382	132.0

由表 8 可见，把问题三的策略直接搬用会产生两类不同后果。bayes 的七点全向缺席判据会把朝外发射的定向源误判为”不存在”，因而出现 24 次缺席误判、全清率只有 77.0%，其平均虚拟时间（3890.801 s）虽然最短，却是”快而错”，不能作为有效结果；49 点网格策略的发现是完备的，但没有后备清除，定位停滞时只能停在”已发现”状态，

全清率 66.6%，属于”稳而不全”。默认策略 `directional` 在全部 11/11 局有效，全清率 100.0%，缺席误判为零，平均虚拟时间 9069.871 s，平均需要 24.818 次后备清除——正是这部分兜底动作换来了网格与批量策略都达不到的完备性。

演练局进一步显示了遮挡造成的额外代价。表 9 给出全部 29 局全清混合源演练的统计。

表 9 问题四：混合源演练局结果（按总虚拟时间升序）

序号	案例编码	源数	清除数	总虚拟时间/s	移动/km	检测数	换频数	平均清除/(s/个)
1	SWA3-C3FB-C5QC-FLW7	16	16	4780.432	18.987	159	108	298.777
2	QL6B-KEUW-DJR2-P84B	16	16	5229.931	20.430	184	144	326.871
3	CEL3-XQV8-M7KY-3CSW	16	16	5806.596	23.313	184	144	362.912
4	2Y25-4LD6-5A4P-64CV	16	16	6685.802	26.469	226	182	417.863
5	QDDJ-S89Y-7TKY-E6VJ	16	16	7316.173	29.811	221	169	457.261
6	3XM4-65TW-CH39-XWTZ	16	16	7327.981	30.165	211	160	457.999
7	6LEP-YVQ6-AYP5-N359	16	16	7982.569	31.403	280	222	498.911
8	7HG2-UEBM-K3D2-LZWT	10	10	8146.581	29.393	377	333	814.658
9	6US3-A76N-KC8B-DMFM	10	10	8171.259	29.676	371	331	817.126
10	TKCY-2VT2-WYUX-2JXU	10	10	8260.712	29.719	385	342	826.071
11	RLK4-E82Q-UHT6-2M9M	10	10	8342.135	30.471	374	328	834.214
12	DX2D-KDF9-BGYU-XPMG	13	13	8642.081	33.710	314	265	664.775
13	JY84-364B-45UL-AFBK	10	10	8704.597	30.413	436	392	870.460
14	T5ZL-5GEP-FDCK-3P8T	13	13	8719.661	33.248	344	285	670.743
15	ZLKB-GVZT-NQDE-CU5F	10	10	8723.919	31.780	395	343	872.392
16	JH76-MEWS-BR6Q-BQ6S	13	13	8732.841	33.504	336	287	671.757
17	VJ68-ADZ6-STFT-LHYT	13	13	8783.099	32.805	369	312	675.623
18	US79-DFNV-UQVW-NPLL	10	10	8795.904	31.740	408	358	879.590
19	B8HR-3HJH-2UYH-YDVB	13	13	8840.642	33.073	371	306	680.049
20	XN57-MH FY-V58F-XCMY	10	10	9188.211	32.756	438	397	918.821
21	2M5J-QWHS-BYGB-MPYF	13	13	9191.437	35.577	343	296	707.034
22	SE9X-67GA-6YC6-QB7B	10	10	9241.603	34.013	406	359	924.160
23	QXVN-L63H-6WLX-KLEJ	10	10	9307.231	34.556	399	351	930.723
24	GSG4-6HTJ-AH5L-ZYGM	10	10	9488.522	34.348	438	379	948.852
25	96M4-SK2V-PDWN-YGHB	13	13	9593.184	36.986	364	311	737.937
26	R2E7-LF5E-J5K6-BSVQ	13	13	9809.921	38.810	340	283	754.609
27	JDAY-2SU7-RTSJ-NQFR	13	13	9813.925	38.555	351	283	754.917
28	A5YU-4SM6-4PK7-M5H8	10	10	9854.818	37.839	382	327	985.482
29	X3Y4-S9D2-XWR9-V6CZ	13	13	10160.134	39.831	365	304	781.549

与问题三相比，混合源场景的检测次数增加约 3 倍（平均 336.9 次对 110.4 次），移

动距离约 31.841 km（对 12.729 km），总虚拟时间平均 8401.445 s，平均定位清除时间 708.350 s/个，约为全向源场景的 2.6 倍。原因在机制上是明确的：一次无信号观测不再等于”源在远处”，而可能只是朝向遮挡，因此必须先用若干探点排除遮挡解释，才能继续收缩位置候选集。本批 29 局全部做到”清完且无失败清除”，说明可见性筛选加后备清除在定向源个数 3 至 8 的范围内是稳定的。

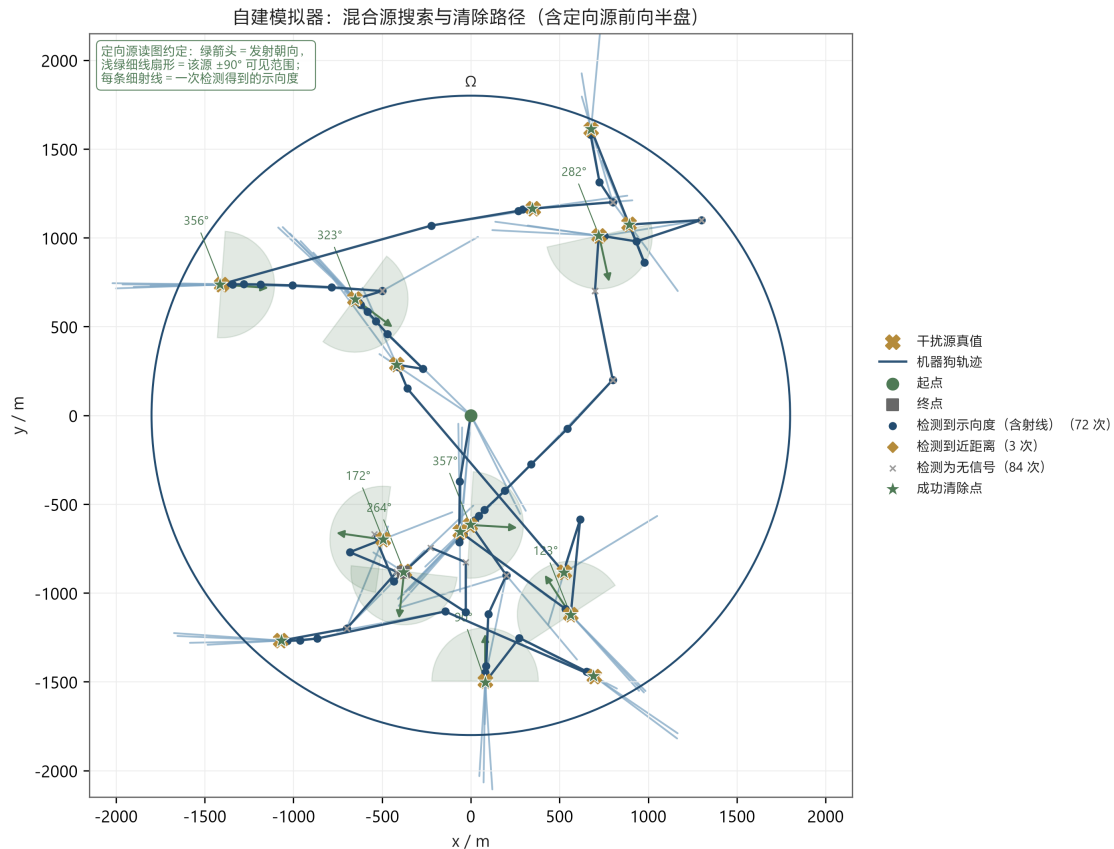


图 9 自建模拟器演练局 SWA3-C3FB-C5QC-FLW7（16 个源，其中 8 个定向源）的搜索与清除路径。 X 为干扰源真值，每个定向源用”绿箭头（发射朝向）+ 浅绿细线扇形（ $\pm 90^\circ$ 可见范围）+ 角度标注”表示；每条浅色细射线表示一次 direction 检测测得的示向度，射线起点即发起检测的位置；圆点为测到示向度的检测点（72 次），菱形为测到近距离的检测点（3 次），灰色叉号为无信号检测点（84 次，其中相当一部分由朝向遮挡造成），星号为成功清除点。

图 9 给出含定向源的一局（SWA3-C3FB-C5QC-FLW7，16 个源、其中 8 个定向）的路径。定向源的前向半盘造成大量无信号观测，轨迹比全向源局密集得多；可见性筛选把探点优先放在朝向相容的位置，后备清除则在定位停滞时兜底完成清除。该局共检测 159 次、换频 108 次，移动 18.987 km，总虚拟时间 4780.432 s。

10.5 动作耗时分解

上面的对比说明”哪种策略更快”，还需要说明”时间花在哪里”。把式 (1) 重排，得到累计移动时间

$$T_{\text{move}} = T - 5N_m - N_s - 5N_c^+ - 3N_c^-. \quad (38)$$

据此可把每局的总虚拟时间拆成移动、检测、换频与清除四项，见表 10。

表 10 演练全清局的动作耗时分解，括号内为占总虚拟时间的比例

问题	局数	移动/s (占比)	检测/s (占比)	换频/s (占比)	清除/s (占比)
问题三 (全向源)	15	38185.679 (78.1%)	8280.000 (16.9%)	1483.000 (3.0%)	975.000 (2.0%)
问题四 (混合源)	29	184675.902 (75.8%)	48855.000 (20.1%)	8301.000 (3.4%)	1810.000 (0.7%)

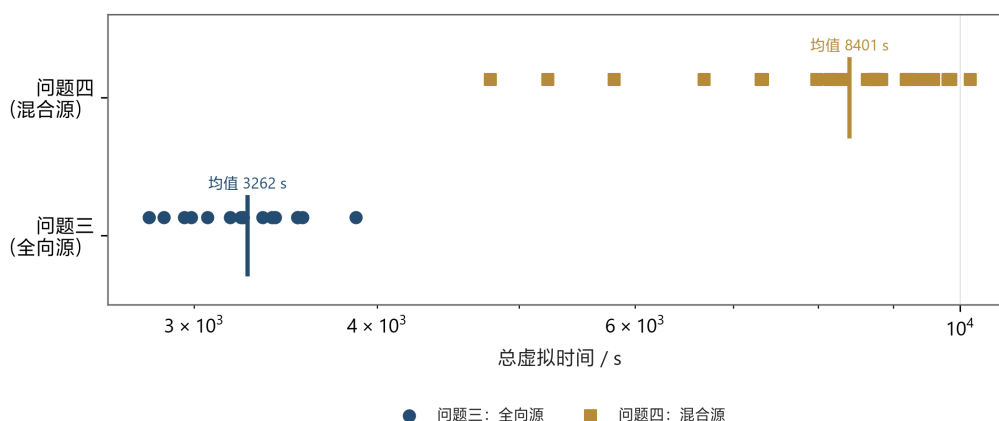


图 10 自建模拟器演练局的总虚拟时间分布。每个点代表一局，实线段为该问题全清局的均值；横轴取对数刻度，问题四（混合源）的整体耗时明显高于问题三（全向源）。

由图 10 与表 10 可见，两类场景中移动时间都是主要成本，分别占 78.1% 与 75.8%，因此缩短总耗时的主攻方向是减少移动，这与问题三相对基线的降幅主要来自位移下降相互印证。清除动作本身只占 2.0% 与 0.7%，说明”清除”不是时间瓶颈，真正的成本在于把机器人送到能清除的位置。检测与换频合计占比不高，但因换频次数与移动路径耦合，仍需一并计入路径排序，不能单独优化。

10.6 本节结论

综合以上配对对比与演练局复算，可提炼出四点结论：1. 几何层的精确判据不需要以效率为代价——问题一的最优实现误差为零而耗时更短，问题二则在保证接收率为 100% 的同时把最坏后验半径压到基线的约五分之一；2. 多源搜索的效率来自”证明不需要检测”，网格幸存集与七点覆盖使默认策略的检测次数和位移相对网格基线分别降到约四分之一和五分之一，平均虚拟时间下降 78.4%；3. 混合源场景必须为无信号保留朝向解释，否则会出现缺席误判；默认策略以全清率 100%、缺席误判 0 和平均 9069.871

s 的代价换取了这种完备性；4. 本次采集的 44 局演练中，两类场景的全部源都被清除且失败清除为零，移动时间占总耗时的 78.1% 与 75.8%，是继续优化的首要对象；清除动作本身只占 2.0% 与 0.7%，不是当前的时间瓶颈。

11 模型评价与推广

11.1 模型的有效性

本文采用集合估计处理有界测向误差，将每次观测转化为真实位置的可行约束，这一条主线贯穿四问。问题一通过凸包顶点计算直径，以最小包围圆半径给出清除充分条件；问题二将接收距离要求表示为闭圆盘的交集，并由三角不等式构造显式可行子集。几何定位、接收约束与清除距离因此由统一的集合运算联系起来，而不是各自独立的经验规则。

多源搜索采用覆盖点集与检测状态矩阵维护各频道的搜索进度。网格表示可处理负观测产生的非凸可行区域，信息增益用于选择有利于收缩候选集的测点。开放路径排序将移动、检测、换频和清除纳入统一目标，并在新观测到来后重新求解任务次序。

对于定向源，位置与朝向的联合约束提供可见性信息，凸组合性质给出方格点集的连续覆盖证明，网格遍历清除则利用清除半径与方向无关的物理条件。这些结论在自建模拟器上完成了闭环验证：本次采集的 44 局演练全部清完全部源、失败清除为零，问题三平均总虚拟时间 3261.579 s、问题四 8401.445 s；配对基线对比进一步说明，几何判据的精确性、覆盖证明与任务排序三者共同决定了最终效率。六份官方正式测试则在附录 A 中提供了正式环境下的效率验证。

11.2 参数影响的解析讨论

主要参数的影响可由模型公式分析。由 $a(\delta) = 1500/(2 \cos \delta)$ 得 $a'(\delta) = 750 \sec \delta \tan \delta$ ，其中 δ 以弧度计。误差半角增大会扩大源位置的包围圆，并缩小保证接收圆盘。表 3 给出了两位小数舍入余量对首次可行区域的影响： 1° 与 1.005° 之间的差别在毫米量级，说明该判据对舍入余量稳健。

由式 (20)，环半径影响覆盖路径长度和最大覆盖距离。默认半径保留约 15.461 m 的距离裕量。减小网格边长可降低 ϵ_g 并收紧位置误差上界，而二维网格数量大致与 ℓ^{-2} 成正比，因此应在精度、内存和计算时间间取得平衡。同理，问题四的 31 点计划在站点数量与覆盖余量之间构成一对可调参数：设计阶段预留的 5° 前向角余量越大，所需站点越多，但对朝向扰动越稳健。

参考文献

- [1] 全国大学生数学建模竞赛组委会. 2026 年高教社杯全国大学生数学建模竞赛 B 题：无线电干扰源的快速自动定位与清除 [Z]. 2026.

- [2] 赛题配套材料. 模拟器使用说明；模拟器通信接口说明及编程指南 [Z]. 2026.
- [3] 本文自建模拟器（评测机实现）. 无线电干扰源环境模拟器协议与场景规范 [Z]. 2026.
- [4] 全国大学生数学建模竞赛组委会. 全国大学生数学建模竞赛论文格式规范（2026 年修订稿） [Z]. 2026.
- [5] WELZL E. Smallest enclosing disks (balls and ellipsoids)[C]// New Results and New Trends in Computer Science. LNCS 555. Berlin: Springer, 1991: 359–370. <https://doi.org/10.1007/BFb0038202>.
- [6] BOYD S, VANDENBERGHE L. Convex Optimization[M]. Cambridge: Cambridge University Press, 2004. <https://web.stanford.edu/~boyd/cvxbook/>.
- [7] CHVÁTAL V. A greedy heuristic for the set-covering problem[J]. Mathematics of Operations Research, 1979, 4(3): 233–235. <https://doi.org/10.1287/moor.4.3.233>.
- [8] 官方无线电干扰源环境模拟器. 问题三、四正式行为日志与配套统计（客户端 1.1.0，六个案例） [DS]. 2026-09-12–2026-09-13. 案例编码及文件校验值见附录 A。

A 官方正式测试的效果验证

A.1 数据选择及对应关系核对

正文的机制论证与定量结论取自自建模拟器演练局，本节则用官方模拟器的正式测试验证算法在正式环境下的效率与正文结论是否一致。实验样本严格限定为官方模拟器 Jammers-simulator-full/test 下的六份正式导出包，问题三、四各有三次。文件名、案例编码与完整包校验值见表 14，核对说明见 docs/paper/data/audit.md。由于官方记录不公开真实源位置和机器人轨迹，本节不绘制搜索/清除路径，只使用清除数量、总虚拟时间、程序运行时间与动作计数四类可直接观测的量。这样处理的原因是：正式测试能够检验“算法是否在真实时限内完成清除”，但不能检验“定位是否收敛到真值”，后者已由正文的演练局分析承担。

评价指标包括清除数量 N_c^+ 、总虚拟时间 T 、程序运行时间 T_{real} ，以及检测、换频和清除的动作次数。其中 T 按整数微秒累计， T_{real} 采用官方记录的毫秒值；平均定位清除时间仅作辅助分析，不替代总虚拟时间与程序运行时间这两项直接指标。六次测试均以主动退出 (user_exit) 结束，统计提交状态为已确认。

A.2 问题三与问题四的正式结果

表 11 问题三正式测试结果（时间单位：s）

测试	测试案例编码	清除数	总虚拟时间/s	程序运行时间/s
测试 1	9HU7-UWJR-BPTR-VYAT	13	3094.978	5.596
测试 2	7TCG-C9EC-53VF-HZ5H	15	3493.335	3.024
测试 3	UVUZ-B4MQ-XSUV-HQFP	10	2906.863	2.680

三次正式测试分别清除 13、15、10 个源，总虚拟时间分别为 3094.978、3493.335、2906.863 s，合计 9495.176084 s；程序运行时间分别为 5.596、3.024、2.680 s，合计 11.300 s。按清除数量加权的平均定位清除时间为 249.873 s/个，仅作辅助解释。第三次的总虚拟时间最短（2906.863 s），但其清除数也最少（10 个），因此判断优化程度时应以总虚拟时间与清除数量并列报告，不能只看单一平均值。

表 12 问题四正式测试结果（时间单位：s）

测试	测试案例编码	清除数	总虚拟时间/s	程序运行时间/s
测试 1	VQUM-ETQ5-74PP-ZQ3J	13	9362.226	7.668
测试 2	5K5T-TRWZ-BPR2-ECAV	10	9271.302	5.724
测试 3	K58B-V5VC-E4EF-CRTW	14	8608.382	6.051

表 12 给出问题四的三次正式结果。三次分别清除 13、10、14 个源，总虚拟时间合

计 27241.909918 s，程序运行时间合计 19.443 s，加权平均定位清除时间 736.268 s/个。第三次清除数最多（14 个）且总虚拟时间最短（8608.382 s），与自建模拟器演练局中”可见性筛选加后备清除能够压缩定向场景耗时”的结论方向一致；但三次案例的随机源分布与朝向不同，仍应以绝对时间与清除数量为准。

A.3 正式测试的动作耗时分解

按式 (38) 由动作计数反推移动时间，得到表 13。其中 Q3-1 表示问题三第一次正式测试，其余编号依此类推。

表 13 六次正式测试的动作计数与移动耗时

案例	检测数	换频数	失败清除	总虚拟时间/s	移动时间/s	移动占比
Q3-1	105	96	0	3094.978	2408.978	77.8%
Q3-2	117	108	0	3493.335	2725.335	78.0%
Q3-3	117	107	0	2906.863	2164.863	74.5%
Q4-1	349	275	0	9362.226	7277.226	77.7%
Q4-2	361	315	0	9271.302	7101.302	76.6%
Q4-3	264	213	0	8608.382	7005.382	81.4%

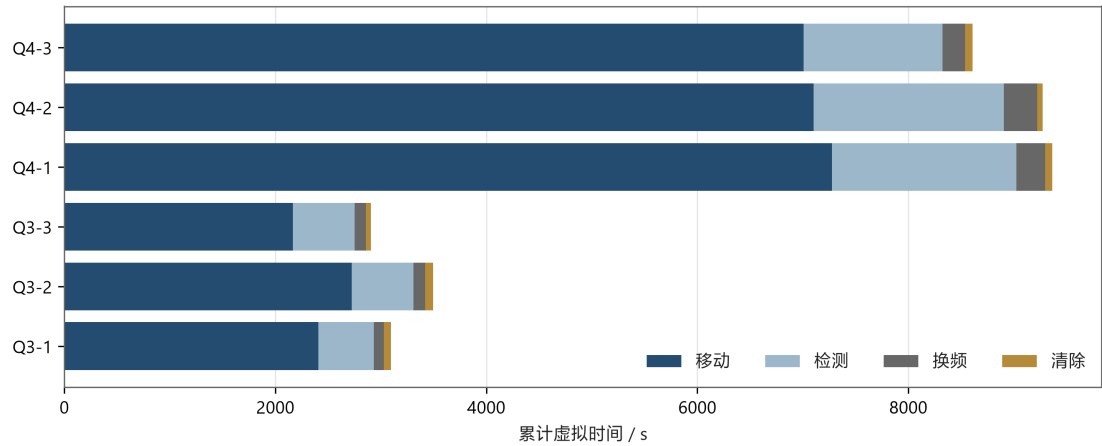


图 11 六次正式测试的累计虚拟耗时分解（数据仅来自正式测试的原始动作计数与总虚拟时间）。

六次测试的清除失败计数均为零，移动时间占总虚拟耗时的 74.47% 至 81.38%，与演练局的构成一致，说明”移动是主要成本”这一结论在正式环境中同样成立。由 $L_{eq} = 5T_{move}$ 反推等效移动距离：问题三依次为 12.045、13.627、10.824 km，问题四依次为 36.386、35.507、35.027 km；对照表 7 与表 9，正式测试的移动量级与演练局相符（问题三约 12–15 km，问题四约 30–33 km），表明演练局并非经过挑选的乐观样本。该等效距离由动作计数与移动速度换算得到，不是独立轨迹测得的长度。

A.4 正式测试结论

六次正式测试均主动退出且清除失败计数为零。以题目要求的直接时间指标衡量：问题三累计清除 38 个源，总虚拟时间 9495.176084 s，程序运行时间合计 11.300 s；问题四累计清除 37 个源，总虚拟时间 27241.909918 s，程序运行时间合计 19.443 s；六次合计清除 75 个源、总虚拟时间 36737.086002 s。由此可得结论：1. 全向源场景下，七点解析覆盖与逐频道网格幸存集使三次正式测试均在时限内完成，总虚拟时间为 2906.863 至 3493.335 s，与演练局量级一致；2. 混合源场景下，前向半盘最小命中集、位置与朝向联合幸存集和后备清除使三次正式测试的总虚拟时间为 8608.382 至 9362.226 s，明显高于全向场景，印证了正文关于”定向遮挡仍是主要时间成本”的判断；3. 平均定位清除时间与移动时间分解只用于解释成本结构，优化程度的结论应以总虚拟时间、程序运行时间和清除数量为准。由于正式数据未公开源总数与类型组成，本文不据此计算全清率或做跨问题性能排名。

表 14 所选正式导出文件与完整包哈希

案例	原始文件名及 SHA-256
Q3-1	formal-p3-1-9HU7-UWJR-BPTR-VYAT.jlog fbd7ab92f78a019887f603172ab6a39e274adb1f917fc2681f1e8073ff47d911
Q3-2	formal-p3-2-7TCG-C9EC-53VF-HZ5H.jlog 5d83ce87455ce22fadbcdbdb60814d9b1f0be54de7bc9d0463777ba47b3922ef
Q3-3	formal-p3-3-UVUZ-B4MQ-XSUV-HQFP.jlog eeffe419c3df005d0ea78c7ec2c10afed29c4bec782327538537dd7ba7914424
Q4-1	formal-p4-1-VQUM-ETQ5-74PP-ZQ3J.jlog 9a102295d324b5fb5aeed1c6036d13ae7e08a7d7fddb60ceb8dead24e62b15d3
Q4-2	formal-p4-2-5K5T-TRWZ-BPR2-ECAV.jlog 370aac9c2c5d324dd382e57e541ef0da7c5dd4759660d9ea0cba8373ce0ea9a1
Q4-3	formal-p4-3-K58B-V5VC-E4EF-CRTW.jlog 1c7f1d111f8fd1a1d0f608d9847c0946b1b0a45bdb7b3f487e0c6d4f4586885f

B 自建模拟器演练局数据清单

本节列出正文定量结论所依据的全部演练局，共 44 局，由 scripts/run_practice_episodes.py 采集、scripts/build_practice_data.py 从评测机行为日志复算得到，复算口径为题面计时模型，与日志记录的最大偏差为 0.000000 s。采集时以固定种子逐局运行生产策略，判定标准为”清完全部源 ^ 无缺席误判 ^ 主动退出 ^ 零异常 ^ 零失败清除”；

本表所列 44 局全部满足该标准。

表 15 演练局汇总（仅统计全清且无失败清除的局）

问题	全清局数	累计清除	平均总虚拟时间/s	时间区间/s	平均定位清除时间/(s/个)	移动占比
问题三（全向源）	15	195	3261.579	2794.212–3867.872	258.782	78.1%
问题四（混合源）	29	362	8401.445	4780.432–10160.134	708.350	75.8%

表 16 全部演练局的逐局复算结果（“全清”列为是否达到该局源数且无失败清除）

案例编码	问题	源数	清除数	失败清除	总虚拟时间/s	检测数	换频数	移动/km	全清
6LST-ZFVS-2HHW-TQQ4	3	13	13	0	2794.212	109	99	10.426	是
6V2P-HJCS-GJ9A-83MA	3	16	16	0	3174.578	103	88	12.458	是
7ZXN-N53X-HAXQ-EGY4	3	10	10	0	3228.653	119	105	12.393	是
8VRC-ADW4-XDXM-F62L	3	13	13	0	3407.039	102	93	13.695	是
ARBP-6XFZ-5YJU-STC8	3	16	16	0	3867.872	116	105	15.514	是
E27E-M8WQ-LDT5-2XDA	3	16	16	0	3531.065	104	89	14.210	是
HGBA-XGH9-UNZJ-7YQ3	3	10	10	0	3340.025	123	113	12.810	是
JW47-AVPH-XGHH-RLLV	3	13	13	0	3389.152	106	93	13.506	是
LZDX-68DJ-XM4P-A6XL	3	10	10	0	2954.390	117	105	11.072	是
MT3R-68KH-CNY5-6X73	3	13	13	0	3527.713	112	94	14.044	是
MULB-4YTU-VQYG-YAS6	3	10	10	0	3064.020	129	119	11.250	是
NMRC-85ZV-3FFE-GBXY	3	13	13	0	2860.706	101	96	10.974	是
QS6H-CC29-PDY3-ZA54	3	16	16	0	2987.385	85	74	12.042	是
R2N7-QMLC-26QX-RXYT	3	16	16	0	3556.397	106	95	14.257	是
VE37-SRPD-NCFR-4EQF	3	10	10	0	3240.474	124	115	12.277	是
2M5J-QWHS-BYGB-MPYF	4	13	13	0	9191.437	343	296	35.577	是
2Y25-4LD6-5A4P-64CV	4	16	16	0	6685.802	226	182	26.469	是
3XM4-65TW-CH39-XWTZ	4	16	16	0	7327.981	211	160	30.165	是
6LEP-YVQ6-AYP5-N359	4	16	16	0	7982.569	280	222	31.403	是
6US3-A76N-KC8B-DMFM	4	10	10	0	8171.259	371	331	29.676	是
7HG2-UEBM-K3D2-LZWT	4	10	10	0	8146.581	377	333	29.393	是
96M4-SK2V-PDWN-YGHB	4	13	13	0	9593.184	364	311	36.986	是
A5YU-4SM6-4PK7-M5H8	4	10	10	0	9854.818	382	327	37.839	是
B8HR-3HJH-2UYH-YDVB	4	13	13	0	8840.642	371	306	33.073	是
CEL3-XQV8-M7KY-3CSW	4	16	16	0	5806.596	184	144	23.313	是
DX2D-KDF9-BGYU-XPMP	4	13	13	0	8642.081	314	265	33.710	是
GSG4-6HTJ-AH5L-ZYGM	4	10	10	0	9488.522	438	379	34.348	是
JDAY-2SU7-RTSJ-NQFR	4	13	13	0	9813.925	351	283	38.555	是
JH76-MEWS-BR6Q-BQ6S	4	13	13	0	8732.841	336	287	33.504	是
JY84-364B-45UL-AFBK	4	10	10	0	8704.597	436	392	30.413	是
QDDJ-S89Y-7TKY-E6VJ	4	16	16	0	7316.173	221	169	29.811	是
QL6B-KEUW-DJR2-P84B	4	16	16	0	5229.931	184	144	20.430	是
QXVN-L63H-6WLX-KLEJ	4	10	10	0	9307.231	399	351	34.556	是
R2E7-LF5E-J5K6-BSVQ	4	13	13	0	9809.921	340	283	38.810	是
RLK4-E82Q-UHT6-2M9M	4	10	10	0	8342.135	374	328	30.471	是
SE9X-67GA-6YC6-QB7B	4	10	10	0	9241.603	406	359	34.013	是
SWA3-C3FB-C5QC-FLW7	4	16	16	0	4780.432	159	108	18.987	是
T5ZL-5GEP-FDCK-3P8T	4	13	13	0	8719.661	344	285	33.248	是
TKCY-2VT2-WYUX-2JXU	4	10	10	0	8260.712	385	342	29.719	是
US79-DFNV-UQVW-NPLL	4	10	10	0	8795.904	408	358	31.740	是
VJ68-ADZ6-STFT-LHYT	4	13	13	0	8783.099	369	312	32.805	是
X3Y4-S9D2-XWR9-V6CZ	4	13	13	0	10160.134	365	304	39.831	是
XN57-MHXY-V58F-XCMY	4	10	10	0	9188.211	438	397	32.756	是
ZLKB-GVZT-NQDE-CU5F	4	10	10	0	8723.919	395	343	31.780	是

表 17 演练局逐局动作耗时分解（时间单位：s）

编号	总虚拟时间	移动	占比	检测	占比	换频	占比	清除	占比
Q3-1	2794.212	2085.212	74.6%	545.000	19.5%	99.000	3.5%	65.000	2.3%
Q3-2	3174.578	2491.578	78.5%	515.000	16.2%	88.000	2.8%	80.000	2.5%
Q3-3	3228.653	2478.653	76.8%	595.000	18.4%	105.000	3.3%	50.000	1.5%
Q3-4	3407.039	2739.039	80.4%	510.000	15.0%	93.000	2.7%	65.000	1.9%
Q3-5	3867.872	3102.872	80.2%	580.000	15.0%	105.000	2.7%	80.000	2.1%
Q3-6	3531.065	2842.065	80.5%	520.000	14.7%	89.000	2.5%	80.000	2.3%
Q3-7	3340.025	2562.025	76.7%	615.000	18.4%	113.000	3.4%	50.000	1.5%
Q3-8	3389.152	2701.152	79.7%	530.000	15.6%	93.000	2.7%	65.000	1.9%
Q3-9	2954.390	2214.390	75.0%	585.000	19.8%	105.000	3.6%	50.000	1.7%
Q3-10	3527.713	2808.713	79.6%	560.000	15.9%	94.000	2.7%	65.000	1.8%
Q3-11	3064.020	2250.020	73.4%	645.000	21.1%	119.000	3.9%	50.000	1.6%
Q3-12	2860.706	2194.706	76.7%	505.000	17.7%	96.000	3.4%	65.000	2.3%
Q3-13	2987.385	2408.385	80.6%	425.000	14.2%	74.000	2.5%	80.000	2.7%
Q3-14	3556.397	2851.397	80.2%	530.000	14.9%	95.000	2.7%	80.000	2.2%
Q3-15	3240.474	2455.474	75.8%	620.000	19.1%	115.000	3.5%	50.000	1.5%
Q4-16	9191.437	7115.437	77.4%	1715.000	18.7%	296.000	3.2%	65.000	0.7%
Q4-17	6685.802	5293.802	79.2%	1130.000	16.9%	182.000	2.7%	80.000	1.2%
Q4-18	7327.981	6032.981	82.3%	1055.000	14.4%	160.000	2.2%	80.000	1.1%
Q4-19	7982.569	6280.569	78.7%	1400.000	17.5%	222.000	2.8%	80.000	1.0%
Q4-20	8171.259	5935.259	72.6%	1855.000	22.7%	331.000	4.1%	50.000	0.6%
Q4-21	8146.581	5878.581	72.2%	1885.000	23.1%	333.000	4.1%	50.000	0.6%
Q4-22	9593.184	7397.184	77.1%	1820.000	19.0%	311.000	3.2%	65.000	0.7%
Q4-23	9854.818	7567.818	76.8%	1910.000	19.4%	327.000	3.3%	50.000	0.5%
Q4-24	8840.642	6614.642	74.8%	1855.000	21.0%	306.000	3.5%	65.000	0.7%
Q4-25	5806.596	4662.596	80.3%	920.000	15.8%	144.000	2.5%	80.000	1.4%
Q4-26	8642.081	6742.081	78.0%	1570.000	18.2%	265.000	3.1%	65.000	0.8%
Q4-27	9488.522	6869.522	72.4%	2190.000	23.1%	379.000	4.0%	50.000	0.5%
Q4-28	9813.925	7710.925	78.6%	1755.000	17.9%	283.000	2.9%	65.000	0.7%
Q4-29	8732.841	6700.841	76.7%	1680.000	19.2%	287.000	3.3%	65.000	0.7%
Q4-30	8704.597	6082.597	69.9%	2180.000	25.0%	392.000	4.5%	50.000	0.6%
Q4-31	7316.173	5962.173	81.5%	1105.000	15.1%	169.000	2.3%	80.000	1.1%
Q4-32	5229.931	4085.931	78.1%	920.000	17.6%	144.000	2.8%	80.000	1.5%
Q4-33	9307.231	6911.231	74.3%	1995.000	21.4%	351.000	3.8%	50.000	0.5%
Q4-34	9809.921	7761.921	79.1%	1700.000	17.3%	283.000	2.9%	65.000	0.7%
Q4-35	8342.135	6094.135	73.1%	1870.000	22.4%	328.000	3.9%	50.000	0.6%
Q4-36	9241.603	6802.603	73.6%	2030.000	22.0%	359.000	3.9%	50.000	0.5%
Q4-37	4780.432	3797.432	79.4%	795.000	16.6%	108.000	2.3%	80.000	1.7%
Q4-38	8719.661	6649.661	76.3%	1720.000	19.7%	285.000	3.3%	65.000	0.7%
Q4-39	8260.712	5943.712	72.0%	1925.000	23.3%	342.000	4.1%	50.000	0.6%
Q4-40	8795.904	6347.904	72.2%	2040.000	23.2%	358.000	4.1%	50.000	0.6%
Q4-41	8783.099	6561.099	74.7%	1845.000	21.0%	312.000	3.6%	65.000	0.7%
Q4-42	10160.134	7966.134	78.4%	1825.000	18.0%	304.000	3.0%	65.000	0.6%
Q4-43	9188.211	6551.211	71.3%	2190.000	23.8%	397.000	4.3%	50.000	0.5%
Q4-44	8723.919	6355.919	72.9%	1975.000	22.6%	343.000	3.9%	50.000	0.6%

C 支撑材料与运行方法

表 18 支撑材料目录及用途

文件或目录	用途
src/mcm26b/	四问几何、信念、覆盖、调度与串行通信实现
src/mcm26b/guarantee/plans/	第四问已实现的覆盖计划及验证参数
project/	四问“最优实现 / 基线实现”配对对比框架与结果
referee/	自建模拟器（评测机）实现，与官方协议同一边界
referee-data/sessions/	正文引用的演练局行为日志（含真值）
scripts/build_practice_data.py	演练局计时复算与论文数据、宏文件生成
scripts/plot_paper_figures.py	正文插图 matplotlib 生成脚本
scripts/audit_paper_formal.py	正式文件对应核对、匿名快照及表格复算
docs/figures/manifest.json	插图来源与生成命令清单
docs/paper/data/	演练局与正式案例的可读统计、表格及校验值
docs/paper/sections/	分章论文源文件
Jammers-simulator-full/test/	六份原名官方正式日志
support-manifest.json	支撑包文件清单及逐文件 SHA-256

正文数据与插图的复算命令如下（在项目根目录执行）：

```
python scripts/build_practice_data.py          # 演练局复算，生成正文数据与宏文件
python scripts/plot_paper_figures.py           # 生成正文全部插图
python scripts/audit_paper_formal.py --snapshot docs/paper/data/formal-results.json
python scripts/build_paper_support.py          # 附录正式数据表与支撑包
```

第一条命令读取 referee-data/sessions/ 下的行为日志，按题面计时模型复算每局的移动、检测、换频与清除耗时，并输出正文引用的宏、表格行与插图数据；第二条命令据此与覆盖计划重新绘制插图，论文不手抄图内任何数字。第三条命令按已有快照复算正式统计并核对正式包的校验值；第四条命令生成附录的正式数据表并打包支撑材料。

Python 运行时需要 requests, numpy 提供可选加速。将 src 加入 Python 模块路径后，可调用 `python -m mcm26b.geometry` 进行问题一几何分析，或通过 `mcm26b.cli` 运行在线策略。在线测试由模拟器界面启动，接口就绪后执行策略入口。具体运行指令和编译步骤见支撑包的 README.md。

D 核心程序代码

四问的求解核心集中在约 5000 行的单文件程序中，按公共层、几何与集合估计层、覆盖保证层、策略层四层组织；评测机、命令行外壳、数据采集与绘图脚本、单元测试以及未被采用的方案不属于求解核心，不在此列出。完整可执行程序为支撑材料中的 `mcm26b-.ipynb`, `src/mcm26b/` 保留与之逐字对应的模块化版本，两者函数名一一对应。

下列代码为无改动的摘录：函数体与注释与程序一致，只按核心程度取舍。未收录的部分包括独立评测机（按附件协议复现模拟器，与求解代码零耦合）、命令行入口、数据采集与论文绘图脚本、单元测试，以及经实测未采用的候选方案。

动作计时与开放路径排序

三段耗时按整数微秒累计；后备清除的走格顺序用最近邻打底并做 2-opt 改进。

CostBreakdown：一次合法动作的三段耗时：移动 + 换频道（仅检测）+ 动作本身。

```

1 class CostBreakdown:
2     """一次合法动作的三段耗时 [题面]：移动 + 换频道（仅检测）+ 动作本身。"""
3
4     move_s: float
5     switch_s: float
6     action_s: float
7     @property
8     def total_s(self) -> float:
9         return self.total_us / 1_000_000
10
11     @property
12     def total_us(self) -> int:
13         """逐段量化后按整数微秒累计，与官方虚拟时钟一致。"""
14         return sum(round(value * 1_000_000)
15                     for value in (self.move_s, self.switch_s, self.action_s))

```

measure_cost：合法 /measure 的耗时；移动按直线距离，换频道只看前后频道是否不同。

```

1 def measure_cost(prev_pos: Vec2, pos: Vec2, prev_channel: int, channel: int,
2                  cost: CostConfig = CostConfig()) -> CostBreakdown:
3     """合法 /measure 的耗时。移动按直线距离；换频道只看前后频道是否不同。"""
4     return CostBreakdown(
5         move_s=dist(prev_pos, pos) / cost.speed_mps,
6         switch_s=0.0 if channel == prev_channel else cost.switch_s,
7         action_s=cost.measure_s,
8     )

```

clear_cost：合法 /clear 的耗时；清除不切换测向机频道，故不含换频项。

```

1 def clear_cost(prev_pos: Vec2, pos: Vec2, success: bool,
2                cost: CostConfig = CostConfig()) -> CostBreakdown:
3     """合法 /clear 的耗时。"""
4     return CostBreakdown(
5         move_s=dist(prev_pos, pos) / cost.speed_mps,
6         switch_s=0.0,
7         action_s=cost.clear_success_s if success else cost.clear_fail_s,
8     )

```

move_time：任意两点间的移动耗时，供规划使用。

```

1 def move_time(a: Vec2, b: Vec2, cost: CostConfig = CostConfig()) -> float:
2     """任意两点间的移动耗时（规划用）。"""
3     return dist(a, b) / cost.speed_mps

```

nearest_neighbor_order：最近邻开放路径，作为路径排序的启发式上界。

```

1 def nearest_neighbor_order(points: list[Vec2],
2                             start: Vec2 = (0.0, 0.0)) -> list[Vec2]:
3     """最近邻开放路径 ( $O(n^2)$ )，启发式上界。"""
4     remaining = list(points)
5     order: list[Vec2] = []
6     position = start
7     while remaining:
8         nxt = min(remaining, key=lambda q: math.dist(position, q))
9         remaining.remove(nxt)
10        order.append(nxt)
11        position = nxt
12    return order

```

two_opt_open：开放路径 2-opt 改进；终点自由，只反转内部片段。

```

1 def two_opt_open(order: list[Vec2], start: Vec2 = (0.0, 0.0),
2                  max_passes: int = 8) -> list[Vec2]:
3     """开放路径 2-opt（终点自由：只反转内部片段，不做首尾相接）。"""
4     best = list(order)

```

```

5     n = len(best)
6     if n < 3:
7         return best
8     for _ in range(max_passes):
9         improved = False
10        for i in range(n - 1):
11            prev = best[i - 1] if i > 0 else start
12            pi = best[i]
13            d_prev_i = math.dist(prev, pi)
14            for j in range(i + 1, n):
15                pj = best[j]
16                nxt = best[j + 1] if j + 1 < n else None
17                d_prev_j = math.dist(prev, pj)
18                removed = d_prev_i
19                added = d_prev_j
20                if nxt is not None:
21                    removed += math.dist(pj, nxt)
22                    added += math.dist(pi, nxt)
23                if added < removed - 1e-9:
24                    best[i:j + 1] = best[i:j + 1][::-1]
25                    improved = True
26                    pi = best[i]
27                    d_prev_i = math.dist(prev, pi)
28            if not improved:
29                break
30    return best

```

问题一：角域交会、直径与最小包围圆

半平面交求定位区域，顶点对枚举直径，Welzl 算法求最小包围圆，并输出直径圆覆盖判定与清除半径判据。

Measurement：一次示向度观测：测点坐标、示向度与误差半角（题设为 1° ）。

```

1 class Measurement:
2
3     x: float
4     y: float
5     bearing_deg: float
6     half_angle_deg: float = 1.0

```

halfplanes：一次观测的三个半平面约束：角域两侧边界与前向约束，法向量已单位化。

```

1 def halfplanes(m: Measurement) -> list[tuple[float, float, float]]:
2     """Return a*x+b*y<=c with unit normals, including the forward..."""
3     theta = m.bearing_deg % 360
4     lower, upper = radians(theta - m.half_angle_deg), radians(theta + m.half_angle_deg)
5     middle = radians(theta)
6     normals = [(sin(lower), -cos(lower)), (-sin(upper), cos(upper)),
7               (-cos(middle), -sin(middle))]
8     return [(a, b, a * m.x + b * m.y) for a, b in normals]

```

convex_hull：单调链凸包（逆时针、无重复末点），输入坐标先平移缩放到同一量级。

```

1 def convex_hull(points: list[Point], eps: float = EPS) -> list[Point]:
2     """Monotone chain, CCW, no repeated final vertex; retain segm..."""
3     ordered = sorted(points)
4     unique = []
5     for p in ordered:
6         if not unique or distance(p, unique[-1]) > eps:
7             unique.append(p)
8     if len(unique) <= 2:
9         return unique
10
11    def chain(seq):
12        out = []
13        for p in seq:
14            while len(out) >= 2:
15                a = (out[-1][0] - out[-2][0], out[-1][1] - out[-2][1])
16                b = (p[0] - out[-1][0], p[1] - out[-1][1])
17                # Distance-to-line tolerance, not an unscaled area tolerance.
18                if cross(a, b) > eps * max(hypot(*a), hypot(*b), eps):
19                    break
20                out.pop()
21            out.append(p)
22    return out

```

```

23
24     return chain(unique)[-1] + chain(reversed(unique))[-1]

```

diameter: 顶点对枚举区域直径; 有界凸集的最远点必为顶点。

```

1 def diameter(points: list[Point]):
2     if not points:
3         raise ValueError("空点集没有本程序定义下的直径")
4     a = b = points[0]
5     value = 0.0
6     for p, q in combinations(points, 2):
7         d = distance(p, q)
8         if d > value:
9             value, a, b = d, p, q
10    return value, (a, b)

```

smallest_enclosing_circle: Welzl 随机增量最小包围圆; 固定随机种子保证结果可复现。

```

1 def smallest_enclosing_circle(points: list[Point], eps: float = EPS):
2     """Welzl's randomized incremental algorithm, using an explicit..."""
3     if not points:
4         raise ValueError("空集不计算最小包围圆")
5     if not isfinite(eps) or eps <= 0:
6         raise ValueError("eps必须为有限正数")
7     if any(len(p) != 2 or any(isinstance(x, bool) or not isinstance(x, (int, float))
8                               or not isfinite(x) for x in p) for p in points):
9         raise ValueError("点集必须由有限二维坐标组成")
10    origin = points[0]
11    scale = max(distance(origin, p) for p in points)
12    if scale == 0:
13        return tuple(origin), 0.0
14    local = [(p[0]-origin[0])/scale, (p[1]-origin[1])/scale) for p in points]
15    tolerance = min(eps/scale, EPS)
16    Random(20260910).shuffle(local)
17    # Frame = (remaining prefix length, forced boundary, returning from child).
18    stack = [(len(local), (), False)]
19    circle = None
20    while stack:
21        n, boundary, returning = stack.pop()
22        if n == 0 or len(boundary) == 3:
23            # Forced support points must lie ON the boundary. For three points
24            # use their circumcircle, even when their unconstrained MEC is smaller.
25            circle = _circumcircle(*boundary) if len(boundary) == 3 else None
26            if circle is None and boundary:
27                circle = _enclosing_circle_reference(list(boundary), tolerance)
28            elif not returning:
29                stack.append((n, boundary, True))
30                stack.append((n-1, boundary, False))
31            elif not _circle_covers(circle, local[n-1], tolerance):
32                stack.append((n-1, boundary + (local[n-1],), False))
33        assert circle is not None
34        center = (origin[0]+circle[0][0]*scale, origin[1]+circle[0][1]*scale)
35        # Recompute after transforming back: returned circle contains all input points
36        # to distance() precision, including any roundoff from the large translation.
37        return center, max(distance(center, p) for p in points)

```

solve: 问题一主入口: 半平面交求定位区域, 区分空集、无界与退化情形, 再给出直径、直径圆覆盖判定与最小包围圆。

```

1 def solve(measurements: list[Measurement], eps: float = EPS) -> dict:
2     """Solve Q1's pure angular intersection (no disk/range prior...)"""
3     if not isfinite(eps) or eps <= 0:
4         raise ValueError("eps必须为有限正数")
5     if not measurements:
6         return {"status": "unbounded", "vertices": [], "diameter_m": None,
7               "diameter_infinite": True, "recession_direction": [1.0, 0.0],
8               "message": "没有观测, 整个平面均可行"}
9     if len(measurements) > 100:
10        raise ValueError("参考实现最多接收100个检测点; 更大规模应使用半平面交算法")
11    ox = sum(m.x for m in measurements) / len(measurements)
12    oy = sum(m.y for m in measurements) / len(measurements)
13    scale = max(1.0, max(hypot(m.x - ox, m.y - oy) for m in measurements))
14    local = [Measurement((m.x - ox) / scale, (m.y - oy) / scale,
15                        m.bearing_deg, m.half_angle_deg) for m in measurements]
16    planes = [h for m in local for h in halfplanes(m)]
17

```

```

18 def inside(p):
19     tol = eps * max(1.0, abs(p[0]), abs(p[1]))
20     return all(a * p[0] + b * p[1] <= c + tol for a, b, c in planes)
21
22 feasible_vertices = []
23 for (a, b, c), (d, e, f) in combinations(planes, 2):
24     det = a * e - b * d
25     if det == 0.0:
26         continue # 精确平行: 无交点 (近平行对的交点是合法远端顶点, 保留)
27     p = ((c * e - b * f) / det, (a * f - c * d) / det)
28     if isfinite(p[0]) and isfinite(p[1]) and inside(p):
29         feasible_vertices.append(p)
30
31 if not feasible_vertices:
32     return {"status": "empty", "vertices": [], "diameter_m": None,
33            "diameter_infinite": False, "tolerance_m": eps * scale,
34            "message": "测向角域没有公共点; 检查频道、数据及误差界"}
35
36 # A nontrivial 2D recession cone has a ray tangent to an input halfplane.
37 # Use a much tighter angular threshold than the positional feasibility epsilon.
38 ray = None
39 for a, b, _ in planes:
40     for direction in ((b, -a), (-b, a)):
41         if all(d * direction[0] + e * direction[1] <= 2e-14
42              for d, e, _ in planes):
43             ray = direction
44             break
45 if ray is not None:
46     break
47
48 hull = convex_hull(feasible_vertices, eps)
49
50 def world(p):
51     return [ox + p[0] * scale, oy + p[1] * scale]
52
53 result = {"status": "unbounded" if ray else
54          ("point" if len(hull) == 1 else "segment" if len(hull) == 2 else "polygon"),
55          "vertices": [world(p) for p in hull], "tolerance_m": eps * scale,
56          "diameter_infinite": ray is not None}
57 if ray is not None:
58     result.update(diameter_m=None, recession_direction=list(ray),
59                  message="存在非零衰退方向, 区域无界, 直径为无穷大")
60     return result
61
62 d, pair = diameter(hull)
63 midpoint = ((pair[0][0] + pair[1][0]) / 2, (pair[0][1] + pair[1][1]) / 2)
64 distances = [distance(midpoint, p) for p in hull]
65 witness_index = max(range(len(hull)), key=distances.__getitem__)
66 max_distance = distances[witness_index]
67 center, radius = smallest_enclosing_circle(hull, eps)
68 area = abs(sum(cross(hull[i], hull[(i + 1) % len(hull)])
69               for i in range(len(hull)))) / 2
70 result.update(diameter_m=d * scale, diameter_pair=[world(p) for p in pair],
71              area_m2=area * scale ** 2,
72              diameter_circle={"center": world(midpoint), "radius_m": d * scale / 2,
73                              "max_vertex_distance_m": max_distance * scale,
74                              "covers": max_distance <= d / 2 + eps * 4,
75                              "coverage_tolerance_m": 4 * eps * scale,
76                              "max_excess_m": (max_distance - d / 2) * scale,
77                              "outside_vertex_indices": [i for i, value in enumerate(distances)
78                                                         if value > d / 2 + 4 * eps],
79                              "witness_vertex": world(hull[witness_index])
80                              if max_distance > d / 2 + 4 * eps else None},
81              minimum_enclosing_circle={"center": world(center), "radius_m": radius * scale},
82              guaranteed_clearable_at_mec_center=radius * scale <= 20 - eps * scale * 4,
83              clearance_certificate={"model": "angular_intersection_only",
84                                   "clear_radius_m": 20.0,
85                                   "numerical_margin_m": 4 * eps * scale,
86                                   "radius_with_margin_m": (radius + 4 * eps) * scale,
87                                   "remaining_margin_m": 20 - (radius + 4 * eps) * scale,
88                                   "center": world(center)})
89 return result

```

问题二：候选集与保证接收的第二检测点

候选集用圆盘外切多边形与角域半平面裁剪表示；安全盘半径取 $1000 - \rho$ ，盘内候选按最坏后验包围半径、同分按移动代价取优。

CandidateRegion：某频道候选集合的保守凸外包多边形（顶点按 CCW；允许退化为 1-2 点）。

```
1 class CandidateRegion:
2     """某频道候选集合的保守凸外包多边形（顶点按 CCW；允许退化为 1-2 点）。"""
3
4     @classmethod
5     def disk(cls, center: Vec2, radius_m: float, sides: int,
6             slack_m: float | None = None) -> "CandidateRegion":
7         if slack_m is None:
8             slack_m = radius_m * (1 - math.cos(math.pi / sides))
9         return cls(circumscribed_polygon(center, radius_m, sides), slack_m)
10
11     @classmethod
12     def initial(cls, station: Vec2, config: ProblemConfig) -> "CandidateRegion":
13         """首个 direction 后的基础集:  $\Omega$  B(station, r_max) 的保守外包。"""
14         field_cfg, sides = config.field, config.search.polygon_sides
15         slack = field_cfg.omega_radius_m * (1 - math.cos(math.pi / sides))
16         region = cls.disk((0.0, 0.0), field_cfg.omega_radius_m, sides, slack)
17         region.clip_disk(station, field_cfg.r_eff_max_m, sides)
18         return region
19
20     def clip_disk(self, center: Vec2, radius_m: float, sides: int) -> None:
21         """与圆盘 B(center, radius) 的外切多边形求交（近似误差入账）。"""
22         for a, b, c in polygon_halfplanes(circumscribed_polygon(center, radius_m, sides)):
23             self._clip(a, b, c)
24             self.slack_m += radius_m * (1 - math.cos(math.pi / sides))
25
26     def clip_wedge(self, station: Vec2, bearing_deg: float,
27                  half_angle_deg: float) -> None:
28         """与测向角域 W(station, bearing $\pm$ half_angle) 求交（半角须  $<90^\circ$ ）。
29
30         两个边界半平面与 localization.halfplanes 同口径， $0^\circ/360^\circ$  跨界由
31         sin/cos 直接处理，无需分支。
32         """
33         if not 0 < half_angle_deg < 90:
34             raise ValueError("角域半角须在 (0, 90) 度")
35         lower = math.radians(bearing_deg - half_angle_deg)
36         upper = math.radians(bearing_deg + half_angle_deg)
37         sx, sy = float(station[0]), float(station[1])
38         self._clip(math.sin(lower), -math.cos(lower),
39                   math.sin(lower) * sx - math.cos(lower) * sy)
40         self._clip(-math.sin(upper), math.cos(upper),
41                   -math.sin(upper) * sx + math.cos(upper) * sy)
42
43     def mec(self) -> tuple[Vec2, float]:
44         """最小包围圆（圆心，半径）。半径是真实候选集 的上界。"""
45         if self._mec is None:
46             if not self.polygon:
47                 raise InconsistentBelief("候选区域为空")
48             self._mec = smallest_enclosing_circle(self.polygon)
49         return self._mec
50
51     def witnesses(self, spacing_m: float) -> list[Vec2]:
52         """顶点 + 边界按间距细分的采样点（最坏后验评估用）。"""
53         if len(self.polygon) < 3:
54             return list(self.polygon)
55         points = list(self.polygon)
56         count = len(self.polygon)
57         for index in range(count):
58             p, q = self.polygon[index], self.polygon[(index + 1) % count]
59             length = math.dist(p, q)
60             steps = int(length // spacing_m)
61             for k in range(1, steps + 1):
62                 ratio = k / (steps + 1)
63                 points.append((p[0] + ratio * (q[0] - p[0]),
64                               p[1] + ratio * (q[1] - p[1])))
65         return _dedupe(points, tol=1e-9)
66
67     def contains(self, point: Vec2, tol_m: float = 0.05) -> bool:
68         """点是否在候选集内（带容差；用于真源保留性检验，不做删点依据）。"""
69         if not self.polygon:
70             return False
71         if len(self.polygon) == 1:
```

```

72         return math.dist(point, self.polygon[0]) <= tol_m + self.slack_m
73     if len(self.polygon) == 2:
74         p, q = self.polygon
75         return _point_segment_distance(point, p, q) <= tol_m + self.slack_m
76     for index, p in enumerate(self.polygon):
77         q = self.polygon[(index + 1) % len(self.polygon)]
78         if (q[0] - p[0]) * (point[1] - p[1]) - (q[1] - p[1]) * (point[0] - p[0])
79             < -tol_m - self.slack_m:
80             return False
81     return True

```

ClearCertificate: 清除充分条件: 候选集外包半径与近似误差之和不超过清除半径。

```

1 class ClearCertificate:
2     """清除证书: 候选集外包半径 (含近似误差) 不超过 20m 阈值。"""
3
4     channel: int
5     center: Vec2
6     radius_m: float
7     limit_m: float
8     certified: bool

```

clear_certificate: 由候选集最小包围圆判定是否可以就地清除。

```

1 def clear_certificate(region: CandidateRegion, channel: int,
2                       config: ProblemConfig = ProblemConfig()) -> ClearCertificate:
3     """_poly <= 20 - 外包近似误差 - 工程余量 才允许发起清除。"""
4     center, radius = region.mec()
5     limit = (config.field.clear_radius_m - region.slack_m
6             - config.search.clear_margin_m)
7     return ClearCertificate(channel, center, radius, limit, radius <= limit)

```

candidate_points: 保证接收盘内的候选点: 盘心加各比例半径上的均匀方向点。

```

1 def candidate_points(center: Vec2, safe_radius_m: float, rings, angles: int) -> list[Vec2]:
2     """保证接收盘内的候选点: 盘心 + 各比例半径 × 均匀方向。"""
3     points = [tuple(center)]
4     for fraction in rings:
5         radius = fraction * safe_radius_m
6         if radius <= 1e-9:
7             continue
8         step = 2 * math.pi / angles
9         for k in range(angles):
10             angle = step * k + step / 2
11             points.append((center[0] + radius * math.cos(angle),
12                           center[1] + radius * math.sin(angle)))
13     return points

```

safe_disk_radius: 保证接收盘半径 $1000 - \rho$; 小于零表示没有保证接收的区域。

```

1 def safe_disk_radius(region: CandidateRegion,
2                      config: ProblemConfig = ProblemConfig()) -> float:
3     """保证接收盘 B(mec, safe) 的半径; <0 表示无保证 ( 超过最小接收半径)。"""
4     return config.field.r_eff_min_m - region.mec()[1] - 1e-3

```

select_measurement_point: 在保证接收盘内选下一检测点: 主指标为最坏后验包围半径, 同分取移动代价较小者。

```

1 def select_measurement_point(region: CandidateRegion, from_position: Vec2,
2                             config: ProblemConfig = ProblemConfig()) -> SecondPointPlan:
3     """在保证接收盘内选择下一检测点 (P3 核心: 对任意候选集通用)。"""
4     field, search = config.field, config.search
5     center, radius = region.mec()
6     half_angle = field.bearing_error_half_deg + search.bearing_rounding_margin_deg
7
8     def move_cost(point: Vec2) -> float:
9         return move_time(from_position, point, config.cost)
10
11     # 1mm 数值余量在 safe_disk_radius 内记账。
12     safe_radius = safe_disk_radius(region, config)
13     if safe_radius < 0:
14         return SecondPointPlan(tuple(center), radius, move_cost(center),
15                                "无保证接收盘 (>1000): 探索 MEC 圆心, 可能 no_signal")
16     coarse = region.witnesses(search.witness_spacing_m)
17     samples = candidate_points(center, safe_radius,
18                               search.candidate_rings, search.candidate_angles)

```

```

19     near_radius, sides = field.near_radius_m, search.polygon_sides
20     best_key, best_point, best_worst = None, None, None
21     for point in samples:
22         worst = _worst_posterior_radius(region, point, coarse, half_angle,
23                                         near_radius, sides)
24         key = (round(worst, 6), round(move_cost(point), 6))
25         if best_key is None or key < best_key:
26             best_key, best_point, best_worst = key, point, worst
27     dense = _worst_posterior_radius(region, best_point,
28                                     region.witnesses(search.witness_spacing_m / 4),
29                                     half_angle, near_radius, sides)
30     reason = (f"安全盘 B(centre,{safe_radius:.0f}m) 内 {len(samples)} 候选; "
31              f"粗采样最坏后验 {best_worst:.1f}m, 1/4 间距加密复核 {dense:.1f}m")
32     return SecondPointPlan(tuple(best_point), dense, move_cost(best_point), reason)

```

select_second_point: 问题二入口: 由首次示向度构造候选集并给出第二检测点。

```

1 def select_second_point(client: SimulatorClient, first_position: Vec2,
2                         first_bearing_deg: float,
3                         config: ProblemConfig = ProblemConfig()) -> SecondPointPlan:
4     """P3 入口: 由首次 direction 构造 K1, 返回保证接收的第二检测点方案。"""
5     region = CandidateRegion.initial(first_position, config)
6     half_angle = config.field.bearing_error_half_deg
7     + config.search.bearing_rounding_margin_deg
8     region.clip_wedge(first_position, first_bearing_deg, half_angle)
9     return select_measurement_point(region, client.state.position, config)

```

问题三: 覆盖计划、检测记录与清除证书

覆盖计划与运行期抽样校验、逐频道检测记录, 以及七点解析覆盖与全向 / 定向两种覆盖语义的证书判据。

CoverageCertificate: 覆盖证书: 采样分辨率、位置与朝向目标数、缺口数与最坏命中距离。

```

1 class CoverageCertificate:
2     """覆盖证书: 采样语义 + 缺口 + 最坏命中距离。"""
3
4     kind: str # "full_disk" | "forward_half_disk"
5     target_step_m: float
6     headings: int
7     r_eff_min_m: float
8     targets: int
9     uncovered: int
10    worst_hit_distance_m: float
11    nn_tour_m: float
12    generator_sha256: str = ""
13    verify_heading_margin_deg: float = 0.0
14    @property
15    def valid(self) -> bool:
16        return self.uncovered == 0 and self.targets > 0
17
18    def as_dict(self) -> dict:
19        return {"kind": self.kind, "target_step_m": self.target_step_m,
20                "headings": self.headings, "r_eff_min_m": self.r_eff_min_m,
21                "targets": self.targets, "uncovered": self.uncovered,
22                "worst_hit_distance_m": round(self.worst_hit_distance_m, 3),
23                "nn_tour_m": round(self.nn_tour_m, 1),
24                "nn_tour_virtual_s": round(self.nn_tour_m / SPEED_MPS, 1),
25                "generator_sha256": self.generator_sha256,
26                "verify_heading_margin_deg": self.verify_heading_margin_deg}

```

CoveragePlan: 覆盖计划: 点集、覆盖语义与证书。

```

1 class CoveragePlan:
2     """一份覆盖计划: 点集 + 语义 + 证书 (可为解析证明)。"""
3
4     name: str
5     problem: int
6     points: tuple[Vec2, ...]
7     kind: str # "analytic" | "certified"
8     coverage: str # "full_disk" | "forward_half_disk"
9     note: str = ""
10    certificate: CoverageCertificate | None = None
11    generator: dict = field(default_factory=dict)
12    def as_dict(self) -> dict:
13        return {"name": self.name, "problem": self.problem, "kind": self.kind,

```

```

14         "coverage": self.coverage, "note": self.note,
15         "generator": self.generator,
16         "certificate": (self.certificate.as_dict()
17                         if self.certificate is not None else None),
18         "points": [[p[0], p[1]] for p in self.points]}

```

coverage_points: 49 点覆盖网格, 单元对角 $700\sqrt{2} \approx 989.95 \leq 1000$ m。

```

1 def coverage_points(config: ProblemConfig = ProblemConfig()) -> list[Vec2]:
2     """49 点覆盖网格生成 (纯几何, 可独立测试) 。"""
3     step = config.search.grid_step_m
4     half = config.search.grid_half_cells
5     return [(step * i, step * j)
6             for j in range(half, -half - 1, -1)
7             for i in range(-half, half + 1)]

```

CoverageLedger: 逐频道检测记录, 按 (频道, 网格点) 二维记账。

```

1 class CoverageLedger:
2     """逐频道覆盖账本: (频道, 网格点) -> 是否已测。"""
3
4     config: ProblemConfig = field(default_factory=ProblemConfig)
5     points: list[Vec2] = field(default_factory=coverage_points)
6     measured: set = field(default_factory=set) # {(channel, point_index)}
7     skipped: dict = field(default_factory=dict) # channel -> 原因
8     def mark_measured(self, channel: int, point_index: int) -> None:
9         self.measured.add((channel, point_index))
10
11     def measured_at(self, channel: int, point_index: int) -> bool:
12         return (channel, point_index) in self.measured
13
14     def skip_channel(self, channel: int, reason: str) -> None:
15         """记录整频道跳过原因 (如 already_cleared / located_pending_clear) 。"""
16         self.skipped[channel] = reason
17
18     def pending_points(self, channel: int) -> list[int]:
19         """该频道尚未测量的网格点下标。扫描调度策略未实现 (P4) 。"""
20         return [index for index in range(len(self.points))
21                 if not self.measured_at(channel, index)]
22
23     def coverage_complete(self, channel: int) -> bool:
24         """该频道覆盖是否完成 (含显式跳过) 。"""
25         return channel in self.skipped or not self.pending_points(channel)

```

seven_points: 原点与环上六个等分点的七站覆盖集。

```

1 def seven_points(ring_radius_m: float = RING_RADIUS_M) -> list[tuple[float, float]]:
2     """原点 + 环上 6 等分点的七站覆盖集。"""
3     return [(0.0, 0.0)] + [(ring_radius_m * math.cos(k * math.pi / 3),
4                             ring_radius_m * math.sin(k * math.pi / 3)) for k in range(6)]

```

seven_plan: 第三问解析覆盖计划; 环半径 1159.7 m 时最坏覆盖距离不超过 984.539 m。

```

1 def seven_plan(ring_radius_m: float) -> CoveragePlan:
2     """第三问解析计划 (覆盖证明由 validate_seven_coverage 给出) 。"""
3     return CoveragePlan(
4         name=f"seven-r{ring_radius_m:.1f}", problem=3, kind="analytic",
5         coverage="full_disk",
6         note=(f"解析证明  $\max(r/\sqrt{3}, \sqrt{r^2+r^2-\sqrt{3}Rr}) < 1000$ ; "
7              f"r [1122.9, 1732] 时成立, 巡检半径越小巡回越短",
8         points=tuple(seven_points(ring_radius_m)))

```

verify_full_disk: 第三问覆盖语义: 圆域内每点都落在某个被访点的最小接收圆盘内。

```

1 def verify_full_disk(points: list[Vec2], step_m: float = VERIFY_STEP_M,
2                       r_eff: float = R_EFF_MIN_M) -> CoverageCertificate:
3     """第三问语义:  $\Omega$  内每点都在某个被访点的  $r_{\text{eff}}$  圆盘内。"""
4     points = list(points)
5     positions = omega_positions(step_m)
6     table = _bucket(points, r_eff)
7     missing, worst = 0, 0.0
8     for (px, py) in positions:
9         ci, cj = int(px // r_eff), int(py // r_eff)
10        best = None
11        for di in (-1, 0, 1):
12            for dj in (-1, 0, 1):

```

```

13         for c_index in table.get((ci + di, cj + dj), ()):
14             cx, cy = points[c_index]
15             distance = math.hypot(cx - px, cy - py)
16             if distance <= r_eff and (best is None or distance < best):
17                 best = distance
18     if best is None:
19         missing += 1
20     else:
21         worst = max(worst, best)
22     return CoverageCertificate("full_disk", step_m, 1, r_eff, len(positions),
23                               missing, worst, nn_tour_length(points))

```

verify_forward_half_disk: 第四问覆盖语义: 任意位置与朝向都存在前向半盘内的被访点。

```

1 def verify_forward_half_disk(points: list[Vec2], step_m: float = VERIFY_STEP_M,
2                               headings: int = VERIFY_HEADINGS,
3                               r_eff: float = R_EFF_MIN_M,
4                               heading_margin_deg: float = 0.0) -> CoverageCertificate:
5     """第四问语义: (p,h) 存在前向半盘内的被访点。"""
6     points = list(points)
7     targets = forward_targets(step_m, headings)
8     table = _bucket(points, r_eff)
9     missing, worst = 0, 0.0
10    sin_margin = math.sin(math.radians(max(heading_margin_deg, 0.0)))
11    for (px, py, heading) in targets:
12        ux, uy = math.cos(heading), math.sin(heading)
13        ci, cj = int(px // r_eff), int(py // r_eff)
14        best = None
15        for di in (-1, 0, 1):
16            for dj in (-1, 0, 1):
17                for c_index in table.get((ci + di, cj + dj), ()):
18                    cx, cy = points[c_index]
19                    dx, dy = cx - px, cy - py
20                    distance_sq = dx * dx + dy * dy
21                    if distance_sq > r_eff * r_eff:
22                        continue
23                    if dx * ux + dy * uy < sin_margin * math.sqrt(distance_sq) - 1e-9:
24                        continue
25                    distance = math.sqrt(distance_sq)
26                    if best is None or distance < best:
27                        best = distance
28    if best is None:
29        missing += 1
30    else:
31        worst = max(worst, best)
32    return CoverageCertificate("forward_half_disk", step_m, headings, r_eff,
33                              len(targets), missing, worst,
34                              nn_tour_length(points))

```

plan_instance: 按名称取覆盖计划; 证书计划在运行期按同一分辨率复核。

```

1 def plan_instance(name: str, verify: bool = True) -> CoveragePlan:
2     """统一入口: 按名称取覆盖计划 (内置解析计划 / 内嵌证书计划)。"""
3     if name == "square49":
4         return square_49_plan()
5     if name == "seven":
6         return seven_plan(1200.0)
7     if name not in EMBEDDED_PLANS:
8         raise FileNotFoundError(f"未找到覆盖计划 {name!r}; 现有: {plan_names()}")
9     plan = embedded_plan(name)
10    if verify and plan.kind == "certified":
11        certificate = verify_plan(plan)
12        if not certificate.valid:
13            raise ValueError(
14                f"覆盖计划 {name!r} 运行期校验失败: "
15                f"{certificate.uncovered}/{certificate.targets} 个 (p,h) 未被覆盖")
16    plan = CoveragePlan(
17        name=plan.name, problem=plan.problem, kind=plan.kind,
18        coverage=plan.coverage, note=plan.note, generator=plan.generator,
19        certificate=CoverageCertificate(
20            kind=certificate.kind, target_step_m=certificate.target_step_m,
21            headings=certificate.headings, r_eff_min_m=certificate.r_eff_min_m,
22            targets=certificate.targets, uncovered=certificate.uncovered,
23            worst_hit_distance_m=certificate.worst_hit_distance_m,
24            nn_tour_m=certificate.nn_tour_m,
25            generator_sha256=(plan.certificate.generator_sha256
26                             if plan.certificate else ""),

```

```

27         points=plan.points)
28     return plan

```

问题三：网格候选集

逐频道的网格幸存集，按格心裕量 $\ell/\sqrt{2}$ 内缩后再删格，据此支持负观测更新、跳测规则与缺席证明。

ChannelParticles：单个频道的网格候选集：格心裕量 $\ell/\sqrt{2}$ ，所有删除判定内缩该裕量，故真源所在格永不被删。

```

1  class ChannelParticles:
2      """单个频道的网格幸存集。"""
3
4      def cell_center(self, index: int) -> Vec2:
5          i, j = divmod(index, self.width)
6          return (self.x0 + (i + 0.5) * self.spacing_m,
7                  self.x0 + (j + 0.5) * self.spacing_m)
8
9      def discard(self, indices: list[int]) -> None:
10         """公开删除入口：联合信念把"朝向不可能"的格一并删除时使用。"""
11         self._discard(indices)
12
13     def _discard(self, indices: list[int]) -> None:
14         if not indices:
15             return
16         coarse_w = self._coarse_width()
17         for index in indices:
18             if index not in self.alive:
19                 continue
20             self.alive.discard(index)
21             key = (index // self.width // _COARSE_FACTOR) * coarse_w
22                   + (index % self.width) // _COARSE_FACTOR
23             count = self._coarse.get(key, 0) - 1
24             if count <= 0:
25                 self._coarse.pop(key, None)
26             else:
27                 self._coarse[key] = count
28         self.version += 1
29         self._mec = None
30
31     def _kill_disk(self, center: Vec2, radius_m: float) -> None:
32         """删除格心距 center <= radius_m 的全部幸存格（整格判据见模块注释）。"""
33         if not self.alive:
34             return
35         reach = radius_m + _DIST_EPS_M
36         ilo = max(0, int(math.floor((center[0] - reach - self.x0) / self.spacing_m)))
37         ihi = min(self.width - 1,
38                   int(math.ceil((center[0] + reach - self.x0) / self.spacing_m)))
39         jlo = max(0, int(math.floor((center[1] - reach - self.x0) / self.spacing_m)))
40         jhi = min(self.width - 1,
41                   int(math.ceil((center[1] + reach - self.x0) / self.spacing_m)))
42         limit_sq = reach * reach
43         doomed = []
44         for i in range(ilo, ihi + 1):
45             dx = self.x0 + (i + 0.5) * self.spacing_m - center[0]
46             base = i * self.width
47             for j in range(jlo, jhi + 1):
48                 index = base + j
49                 if index not in self.alive:
50                     continue
51                 dy = self.x0 + (j + 0.5) * self.spacing_m - center[1]
52                 if dx * dx + dy * dy <= limit_sq:
53                     doomed.append(index)
54             self._discard(doomed)
55
56     def _kill_outside_disk(self, center: Vec2, radius_m: float) -> None:
57         """删除格心距 center > radius_m 的全部幸存格（保留盘内）。"""
58         if not self.alive:
59             return
60         reach = radius_m + _DIST_EPS_M
61         doomed = [index for index in self.alive
62                   if math.dist(self.cell_center(index), center) > reach]
63         self._discard(doomed)
64
65     def _kill_outside_wedge(self, station: Vec2, reading_deg: float) -> None:

```

```

66     """删除角向偏离读数超出  $r_{eff}$  + 角向格裕量的幸存格。
67
68     角向格裕量 =  $\arcsin(m / \max(\text{dist}-m, 1))$ : 格内任一点相对格心的方位
69     偏差不超过它 (点距测点还可再近  $m$ , 故分母再内缩  $m$ )。
70     """
71     if not self.alive:
72         return
73     sx, sy = float(station[0]), float(station[1])
74     doomed = []
75     for index in self.alive:
76         gx, gy = self.cell_center(index)
77         dist = math.hypot(gx - sx, gy - sy)
78         if dist <= self.margin_m + 1.0:
79             continue # 格贴测点: 格内方位可指向任意方向, 不可删
80         angular_margin = math.degrees(
81             math.asin(min(0.9999, self.margin_m / (dist - self.margin_m))))
82         gap = angular_gap(math.degrees(math.atan2(gy - sy, gx - sx)), reading_deg)
83         if gap > self.half_angle_deg + angular_margin + _ANG_EPS_DEG:
84             doomed.append(index)
85     self._discard(doomed)
86
87     def apply_measure(self, station: Vec2, result: str,
88                     svd_deg: float | None = None, *,
89                     directional: bool = False) -> None:
90         """按读数更新幸存集。真源保留性由 Voronoi 裕量保证 (模块注释)。
91
92         ``directional=True`` (第四问): no_signal **不删空间候选**——定向源
93         的静默还可能来自朝向遮挡, 单独删空间候选会误删真源; 朝向维度由
94         :class:`~mcm26b.belief.joint.JointBelief` 维护。
95         """
96         if result == NO_SIGNAL:
97             if directional:
98                 return
99             # no_signal dist > r_eff >= 1000: 整格落入 B(s, 1000-m) 的删。
100             self._kill_disk(station, self.r_eff_min_m - self.margin_m)
101         elif result == NEAR:
102             self._kill_outside_disk(station, self.near_radius_m + self.margin_m)
103         elif result == DIRECTION:
104             if svd_deg is None:
105                 raise ValueError("direction 读数必须携带 svd_deg")
106             # direction 隐含 dist <= r_max [题面]
107             self._kill_outside_disk(station, self.r_eff_max_m + self.margin_m)
108             self._kill_outside_wedge(station, svd_deg)
109         else:
110             raise ValueError(f"未知测向结果: {result!r}")
111
112     def alive_points(self, limit: int | None = None) -> list[Vec2]:
113         sorted_alive = sorted(self.alive)
114         if limit is not None and len(sorted_alive) > limit:
115             step = len(sorted_alive) / limit
116             indices = {sorted_alive[int(k * step) % len(sorted_alive)]
117                        for k in range(limit)}
118             indices = sorted(indices)
119         else:
120             indices = sorted_alive
121         return [self.cell_center(index) for index in indices]
122
123     def nearest_alive_distance(self, point: Vec2, limit_m: float) -> float | None:
124         """最近幸存格心距离; > limit_m 时返回 None (粗格环扫 + 下界剪枝)。
125
126         返回值语义: 恰为最近幸存格心距离 (若 <= limit_m); None 表示所有
127         幸存格心都超出 limit_m。粗格聚合半径已计入扫描范围, 不漏判。
128         """
129         if not self.alive:
130             return None
131         coarse_cell_m = self.spacing_m * _COARSE_FACTOR
132         coarse_reach = coarse_cell_m * math.sqrt(2) / 2
133         cw = self._coarse_width()
134         ci_base = int(math.floor((point[0] - self.x0) / coarse_cell_m))
135         cj_base = int(math.floor((point[1] - self.y0) / coarse_cell_m))
136         ring_max = int(math.ceil((limit_m + coarse_reach) / coarse_cell_m)) + 1
137         best: float | None = None
138         for ring in range(ring_max + 1):
139             lower_bound = max(0.0, ring - 1) * coarse_cell_m - coarse_reach
140             if best is not None and lower_bound >= best:
141                 break # 更外环的所有格心都 >= 下界 >= 当前最优
142             for ci in range(max(0, ci_base - ring), min(cw - 1, ci_base + ring) + 1):

```

```

143         for cj in range(max(0, cj_base - ring), min(cw - 1, cj_base + ring) + 1):
144             if max(abs(ci - ci_base), abs(cj - cj_base)) != ring:
145                 continue # 只扫当前环 (Chebyshev 环)
146             if (ci * cw + cj) not in self._coarse:
147                 continue
148             ccx = self.x0 + (ci + 0.5) * coarse_cell_m
149             ccy = self.x0 + (cj + 0.5) * coarse_cell_m
150             if math.hypot(ccx - point[0], ccy - point[1])
151                 > limit_m + coarse_reach:
152                 continue
153             i0, j0 = ci * _COARSE_FACTOR, cj * _COARSE_FACTOR
154             for i in range(i0, min(i0 + _COARSE_FACTOR, self.width)):
155                 base = i * self.width
156                 for j in range(j0, min(j0 + _COARSE_FACTOR, self.width)):
157                     index = base + j
158                     if index not in self.alive:
159                         continue
160                     dist = math.dist(self.cell_center(index), point)
161                     if dist <= limit_m and (best is None or dist < best):
162                         best = dist
163         return best
164
165     def guaranteed_no_signal(self, station: Vec2) -> bool:
166         """真源与该测点距离必 > r_max (>= r_eff) 必 no_signal, 测了也无信息。"""
167         return self.nearest_alive_distance(
168             station, self.r_eff_max_m + self.margin_m) is None
169
170     def mec(self) -> tuple[Vec2, float]:
171         """幸存集最小包围圆 (圆心, 半径); 按 version 缓存。"""
172         if self._mec is None:
173             if not self.alive:
174                 raise ValueError("幸存集为空 (该频道已可证明缺席)")
175             points = [self.cell_center(index) for index in self.alive]
176             self._mec = smallest_enclosing_circle(points)
177         return self._mec
178
179     def subsample(self, size: int) -> list[Vec2]:
180         """按索引序均匀抽取的子采样 (索引序 空间序, 代表点空间铺开)。"""
181         return self.alive_points(limit=size)
182
183     def expected_survival(self, point: Vec2, subsample,
184                           representatives) -> float:
185         """E[log 幸存占比]: 越小该测点的期望信息增益越大。
186
187         r_eff 不确定性解析积分: dist (1000,1500) 时按 r_eff-U(1000,1500)
188         取 p_ns=(dist-1000)/500 为 no_signal 分支概率。
189         - no_signal 分支: 删 B(point, 1000-m), 幸存 = 子采样中盘外比例;
190         - direction 分支: 以代表点方位为期望读数, 角域 B(point,1500+m) 内幸存;
191         - near 分支 (dist <= 5+m): 可行集收缩到 5m 盘, 幸存占比趋近 0。
192         """
193         if not self.alive or len(subsample) == 0 or len(representatives) == 0:
194             return 0.0
195         if _np is not None and isinstance(subsample, _np.ndarray)
196             and isinstance(representatives, _np.ndarray):
197             return self._expected_survival_vector(point, subsample, representatives)
198         eps = 1.0 / len(subsample)
199         keep_ns = max(1.0 - self.fraction_within(subsample, point,
200                                                    self.r_eff_min_m - self.margin_m),
201                      eps)
202         log_keep_ns = math.log(keep_ns)
203         total = 0.0
204         for rep in representatives:
205             dist = math.hypot(rep[0] - point[0], rep[1] - point[1])
206             if dist <= self.near_radius_m + self.margin_m:
207                 total += math.log(eps)
208                 continue
209             p_ns = min(1.0, max(0.0, (dist - self.r_eff_min_m)
210                               / (self.r_eff_max_m - self.r_eff_min_m)))
211             reading = math.atan2(rep[1] - point[1], rep[0] - point[0])
212             survivors = 0
213             for p in subsample:
214                 if math.dist(p, point) > self.r_eff_max_m + self.margin_m:
215                     continue
216                 gap = angular_gap(math.degrees(
217                     math.atan2(p[1] - point[1], p[0] - point[0])),
218                     math.degrees(reading))
219                 if gap <= self.half_angle_deg + _ANG_EPS_DEG:

```

```

220         survivors += 1
221         keep_dir = max(survivors / len(subsample), eps)
222         total += p_ns * log_keep_ns + (1.0 - p_ns) * math.log(keep_dir)
223     return total / max(len(representatives), 1)
224
225 def fraction_within(self, samples, point: Vec2, radius_m: float) -> float:
226     if len(samples) == 0:
227         return 0.0
228     reach = radius_m + _DIST_EPS_M
229     if _np is not None and isinstance(samples, _np.ndarray):
230         distance = _np.hypot(samples[:, 0] - point[0], samples[:, 1] - point[1])
231         return float(_np.count_nonzero(distance <= reach)) / len(samples)
232     count = sum(1 for p in samples if math.dist(p, point) <= reach)
233     return count / len(samples)
234
235 def expected_posterior_radius(self, point: Vec2, subsample,
236                             representatives) -> float:
237     """一次测量后期望后验半径（米）：选点主指标，直指清除证书阈值。
238
239     对每个代表假设（真值候选）按  $r_{\text{eff}} \sim U(1000, 1500)$  解析积分
240     no_signal/direction 分支，near 分支（dist <= 5+m）后验为 5m 盘；
241     各分支半径用幸存子采样的质心最大距离近似（偏差 << 格距，排序用）。
242     与几何版 worst-posterior 同族，但作用于含负观测的非凸幸存集，
243     且以期望代替最坏（代表点覆盖幸存集主轴）。
244     """
245     if not self.alive or len(subsample) == 0 or len(representatives) == 0:
246         return 0.0
247     if _np is not None and isinstance(subsample, _np.ndarray)
248         and isinstance(representatives, _np.ndarray):
249         return self._expected_posterior_radius_vector(
250             point, subsample, representatives)
251     eps_radius = self.spacing_m / 2.0
252     total = 0.0
253     for rep in representatives:
254         dist = math.hypot(rep[0] - point[0], rep[1] - point[1])
255         if dist <= self.near_radius_m + self.margin_m:
256             total += self.near_radius_m + self.margin_m
257             continue
258         p_ns = min(1.0, max(0.0, (dist - self.r_eff_min_m)
259                             / (self.r_eff_max_m - self.r_eff_min_m)))
260         members_ns = [p for p in subsample
261                       if math.dist(p, point) > self.r_eff_min_m - self.margin_m]
262         reading = math.atan2(rep[1] - point[1], rep[0] - point[0])
263         members_dir = []
264         for p in subsample:
265             if math.dist(p, point) > self.r_eff_max_m + self.margin_m:
266                 continue
267             gap = angular_gap(math.degrees(
268                 math.atan2(p[1] - point[1], p[0] - point[0])),
269                 math.degrees(reading))
270             if gap <= self.half_angle_deg + _ANG_EPS_DEG:
271                 members_dir.append(p)
272         r_ns = self._branch_radius(members_ns) if members_ns else eps_radius
273         r_dir = self._branch_radius(members_dir) if members_dir else eps_radius
274         total += p_ns * r_ns + (1.0 - p_ns) * r_dir
275     return total / max(len(representatives), 1)

```

问题三：信息增益选点与滚动任务排序

统一任务表由站点扫频、定位测量与清除三类任务组成，每执行一项即重排；定位点按期望信息增益加移动代价评分。

BayesLoop：问题三闭环：网格候选集的正负观测更新、信息增益选点与滚动任务排序。

```

1 class BayesLoop(OmnidirectionalLoop):
2     """问题3贝叶斯闭环：负观测感知 + 信息增益选点 + 滚动任务排序。"""
3
4     def _build_tasks(self) -> list[Task]:
5         tasks: list[Task] = []
6         for index, point in enumerate(self.ledger.points):
7             pending = self._fused_channels_at(index)
8             if not pending:
9                 continue
10            switches = len(pending) - (1 if self.client.state.channel in pending else 0)
11            tasks.append(Task("station", tuple(point),
12                             self.config.cost.measure_s * len(pending))

```

```

13         + self.config.cost.switch_s * max(switches, 0),
14         station_index=index, station_channels=tuple(pending)))
15 for channel in self.belief.channels_by_status(ChannelStatus.DETECTED):
16     cert = self._best_certificate(channel)
17     if cert is not None and cert.certified
18         and channel not in self._clear_failed:
19         tasks.append(Task("clear", cert.center,
20                           self.config.cost.clear_success_s,
21                           channel=channel))
22     continue
23     if self._defer_to_station(channel):
24         continue # 下一条方位线可由"本就要去的站点"顺路提供
25     point = self._select_point(channel)
26     if point is None:
27         continue
28     tasks.append(Task("locate", point,
29                       self.config.cost.measure_s + self.config.cost.switch_s,
30                       channel=channel))
31 return tasks
32
33 def _execute_station(self, index: int) -> None:
34     """单站批量扫频：只测待测频道（跳测规则已过滤），扫完即返回重排。
35
36     P0-1: 本站同时测量"已发现未定证"频道，使第二/第三条方位线在巡回
37     途中顺路取得；证书一旦成立即就地清除，不额外绕行。
38     """
39     point = self.ledger.points[index]
40     pending = self._fused_channels_at(index)
41     unknown = [c for c, b in self.belief.channels.items()
42                if b.status is ChannelStatus.UNKNOWN]
43     self.skipped_measures += len(unknown) - len(
44         [c for c in pending if self.belief.channels[c].status
45          is ChannelStatus.UNKNOWN])
46     # 当前频道优先，减少换频；同站 near 直接清除不中断扫频
47     pending.sort(key=lambda c: (c != self.client.state.channel, c))
48     for channel in pending:
49         if self.stop_reason is not None or self._count_limit_reached()
50             or self._virtual_cap_reached():
51             return
52         detected_before = (self.belief.channels[channel].status
53                            is ChannelStatus.DETECTED)
54         outcome = self.client.measure(point, channel)
55         self.ledger.mark_measured(channel, index)
56         self._record_measure(channel, point, outcome)
57         if outcome.result in (DIRECTION, NEAR):
58             self._may_clear.add(channel)
59         if outcome.result == NEAR:
60             region = self.belief.candidate_set(channel)
61             cert = clear_certificate(region, channel, self.config)
62             if cert.certified:
63                 if self._attempt_clear(channel, point, cert):
64                     self._clear_failed.discard(channel)
65                 else:
66                     self._clear_failed.add(channel)
67                     self._may_clear.discard(channel)
68             else:
69                 self._anomaly("near_without_certificate", channel, position=point)
70             continue
71         if not detected_before:
72             continue
73         if outcome.result == NO_SIGNAL:
74             # 该频道已被发现：保证接收盘内不可能 no_signal，属守恒性异常
75             region = self.belief.candidate_set(channel)
76             safe = (safe_disk_radius(region, self.config)
77                    if region is not None else -1.0)
78             if safe >= 0 and math.dist(point, region.mec()[0]) <= safe:
79                 self._anomaly("no_signal_in_safe_disk", channel,
80                               position=point)
81             continue
82         cert = self._best_certificate(channel)
83         if cert is not None and cert.certified:
84             if self._attempt_clear(channel, cert.center, cert):
85                 self._clear_failed.discard(channel)
86             else:
87                 self._clear_failed.add(channel)
88                 self._may_clear.discard(channel)
89

```

```

90 def _record_measure(self, channel: int, position: Vec2, outcome) -> None:
91     """belief (凸证书) 与 PF (幸存集) 同步更新; PF 空即证缺席。"""
92     self.belief.update_measure(channel, position, outcome.result,
93                               outcome.svd_deg)
94     # 同一 0.5m 测点 × 同一频道的重复检测: 历史上一次官方演练有 98.84%
95     # 的虚拟时间耗在空复扫上, 本计数是该问题的回归指标。
96     key = _point_key(position)
97     seen = self._measured_points.setdefault(channel, set())
98     if key in seen:
99         self.redundant_rescan_measures += 1
100     seen.add(key)
101     self._clear_failed.discard(channel) # 新证据落地: 允许再次尝试清除
102     belief = self.belief.channels[channel]
103     if belief.status in (ChannelStatus.CLEARED, ChannelStatus.ABSENT):
104         return # 已证毕: 清除/缺席后的 no_signal 与距离无关, 不进 PF
105     pf = self.particles[channel]
106     pf.apply_measure(position, outcome.result, outcome.svd_deg)
107     if not pf:
108         if belief.status is ChannelStatus.UNKNOWN:
109             self.belief.mark_absent(channel)
110             self.absence_evidence[channel] = "pf_empty"
111             if channel not in self.pf_absent_channels:
112                 self.pf_absent_channels.append(channel)
113         else:
114             self._anomaly("pf_empty_while_detected", channel, position=position)
115     self._apply_count_bound()
116
117 def _best_certificate(self, channel: int) -> ClearCertificate | None:
118     """凸证书优先; 负观测挖洞使 PF 幸存集更紧时用网格证书补位。"""
119     region = self.belief.candidate_set(channel)
120     cert = clear_certificate(region, channel, self.config)
121     if region is not None else None
122     if cert is not None and cert.certified:
123         return self._near_clear_target(cert)
124     if self.config.bayes.grid_cert_enabled:
125         pf = self.particles[channel]
126         if len(pf):
127             center, radius = pf.mec()
128             limit = (self.config.field.clear_radius_m - pf.margin_m
129                     - self.config.search.clear_margin_m)
130             if radius <= limit:
131                 return self._near_clear_target(
132                     ClearCertificate(channel, center, radius, limit, True))
133     return cert
134
135 def _select_point(self, channel: int) -> Vec2 | None:
136     """定位测量点: 期望 log 幸存占比 + · 移动时间 最小的候选。"""
137     context = self._probe_context(channel)
138     if context is None:
139         return None
140     candidates, info_score, center, position = context
141     bayes = self.config.bayes
142
143     def score(q: Vec2) -> float:
144         # 一次定位后仍需接近源才能清除; 仅计到检测点的移动会偏好绕行。
145         move = bayes.lambda_move_s * (move_time(position, q, self.config.cost)
146                                       + move_time(q, center, self.config.cost))
147         return info_score(q) + move
148
149     return min(candidates, key=lambda q: (score(q), _point_key(q)))
150
151 def _localize_and_clear(self, channel: int) -> bool:
152     """阻塞式定位子循环 (resolve 阶段安全网): PF 选点, 失败退凸选点。"""
153     search = self.config.search
154     self._may_clear.add(channel)
155     for _ in range(search.max_localize_steps):
156         if self._virtual_cap_reached():
157             return False
158         cert = self._best_certificate(channel)
159         if cert is not None and cert.certified and channel in self._may_clear:
160             if self._attempt_clear(channel, cert.center, cert):
161                 self._clear_failed.discard(channel)
162                 return True
163             self._clear_failed.add(channel)
164             self._may_clear.discard(channel)
165             continue
166     point = self._select_point(channel)

```

```

167         if point is None:
168             region = self.belief.candidate_set(channel)
169             if region is None:
170                 return False
171             point = select_measurement_point(
172                 region, self.client.state.position, self.config).position
173             outcome = self.client.measure(point, channel)
174             self._record_measure(channel, point, outcome)
175             self._may_clear.add(channel)
176             if outcome.result == NO_SIGNAL:
177                 region = self.belief.candidate_set(channel)
178                 if region is not None and safe_disk_radius(region, self.config) >= 0
179                     and math.dist(point, region.mec()[0])
180                     <= safe_disk_radius(region, self.config):
181                     self._anomaly("no_signal_in_safe_disk", channel, position=point)
182             cert = self._best_certificate(channel)
183             if cert is not None and cert.certified:
184                 if self._attempt_clear(channel, cert.center, cert):
185                     self._clear_failed.discard(channel)
186                     return True
187                 self._clear_failed.add(channel)
188                 self._may_clear.discard(channel)
189             return self.belief.channels[channel].status is ChannelStatus.CLEARED

```

问题四：朝向候选、可见性选点与网格遍历清除

每个幸存格附朝向档掩码构成联合候选集，可见性权重用于探点排序；定位停滞时以 20 m 方格遍历清除兜底。

JointBelief：单频道的位置与朝向联合候选集；朝向分 24 档，负观测只在整格必落在最小接收半径内时才约束朝向。

```

1 class JointBelief:
2     """单频道（位置 × 朝向）联合可行集。"""
3
4     def is_absent(self) -> bool:
5         """联合集空 该频道不存在源（真源保留定理的二维推广）。"""
6         return not self.position.alive
7
8     def sample(self, size: int = 256) -> list[int]:
9         """均匀抽样幸存格（评分用；证书始终用完整集合）。"""
10        cached = self._sample_cache
11        if cached is not None and cached[0] == self.position.version:
12            return cached[1]
13        alive = sorted(self.position.alive)
14        if len(alive) > size:
15            step = len(alive) / size
16            alive = [alive[int(k * step) % len(alive)] for k in range(size)]
17        self._sample_cache = (self.position.version, alive)
18        return alive
19
20    def expected_posterior_radius(self, reader: Vec2, size: int | None = None) -> float:
21        """一次测量后的期望后验 MEC 半径（米）；D1 值函数。
22
23        与 :meth:`ChannelParticles.expected_posterior_radius` 同族，差别只在
24        no_signal 分支概率：定向源不仅会因距离 > r_eff 静默，还会因朝向遮挡
25        静默，因此
26
27        
$$p_{\text{vis}}(p) = P(r_{\text{eff}} \mid p-q) \times \text{cover\_ratio}(p, q)$$

28
29        其中 ``cover_ratio`` 是该格朝向掩码中"可能覆盖 q 方位"的档位占比。
30        分支半径语义：
31
32        - direction（可见，非 near）：读数为 q 处的方位 ±，后验收缩到该方位
33          楔形与当前可行集之交，半径取幸存子采样的质心最大距离；
34        - no_signal：定向源的静默**不删空间候选**，位置后验半径保持当前值；
35        - near：后验为 5m 盘。
36
37        返回值只用于**选点排序**，不参与任何安全判定；证书仍由凸/网格口径给出。
38        """
39        pf = self.position
40        if not pf.alive:
41            return 0.0
42        if size is None:
43            size = self.config.bayes.score_subsample
44        indices = self.sample(max(2, size))

```

```

45     if not indices:
46         return 0.0
47     stride = max(len(indices) // max(self.config.bayes.mc_representatives, 1), 1)
48     representatives = indices[::stride]
49     if _np is not None:
50         return self._epr_vector(reader, indices, representatives)
51     return self._epr_python(reader, indices, representatives) # pragma: no cover
52
53 def prob_detect(self, reader: Vec2, size: int = 256) -> float:
54     """在 ``reader`` 处能看见任一可行（位置，朝向）的概率质量（启发式）。
55
56     r_eff ~ U[1000, 1500] 解析积分：朝向按可行档位占比计权。
57     只用于选点排序，不参与任何安全判定。
58     """
59     cells = self.sample(size)
60     if not cells:
61         return 0.0
62     total = 0.0
63     span = self.field.r_eff_max_m - self.field.r_eff_min_m
64     for index in cells:
65         gx, gy = self.position.cell_center(index)
66         distance = math.hypot(reader[0] - gx, reader[1] - gy)
67         if distance > self.field.r_eff_max_m:
68             continue
69         if distance <= self.field.r_eff_min_m:
70             p_range = 1.0
71         else:
72             p_range = (self.field.r_eff_max_m - distance) / span
73         if p_range <= 0.0:
74             continue
75         bearing = math.degrees(math.atan2(reader[1] - gy, reader[0] - gx))
76         cover = _COVER_TABLE[_table_index(bearing)]
77         bits = bin(self._mask(index) & cover).count("1")
78         if bits:
79             total += p_range * bits / HEADING_BINS
80     return total / len(cells)
81
82 def apply(self, station: Vec2, result: str,
83          svd_deg: float | None = None) -> None:
84     """按读数更新联合可行集（保守，真源（位置，朝向）永不删）。"""
85     if result == DIRECTION:
86         self.position.apply_measure(station, DIRECTION, svd_deg,
87                                    directional=True)
88         self._apply_heading(station, keep=True)
89     elif result == NEAR:
90         # near 同样要求可见（评测机先判朝向再判距离）
91         self.position.apply_measure(station, NEAR, directional=True)
92         self._apply_heading(station, keep=True)
93     elif result == NO_SIGNAL:
94         # 距离阈值必须取**保守侧**：只有"整格都必然落在 r_eff_min 圆盘内"
95         # 时，no_signal 才一定由朝向遮挡解释。格心到格内任一点 margin，
96         # 故阈值取 r_eff_min - margin；取 + margin 会把 1000~1010m 的模糊带
97         # 也当成"必须不覆盖"，从而可能删掉真源的朝向档（破坏真源保留性）。
98         self._apply_heading(station, keep=False,
99                             within_m=max(
100                                 0.0, self.field.r_eff_min_m
101                                 - self.position.margin_m))
102     else:
103         raise ValueError(f"未知测向结果: {result!r}")
104
105 def summary(self) -> dict:
106     """联合集规模快照（写入策略 summary，供审计与论文复算）。"""
107     cells = self.sample(256)
108     bins = [bin(self._mask(index)).count("1") for index in cells]
109     return {"cells": len(self.position),
110            "sampled_mean_heading_bins": (round(sum(bins) / len(bins), 2)
111                                           if bins else 0.0),
112            "heading_bins_total": HEADING_BINS}

```

DirectionalExtension: 问题四闭环：定向覆盖计划、遮挡解释与网格遍历清除兜底。

```

1 class DirectionalExtension(OmnidirectionalLoop):
2     """问题4混合源闭环（P5）：全向基线 + 定向观测处理 + 后备清除。"""
3
4     def _localize_and_clear(self, channel: int) -> bool:
5         """定位子循环：选点测量收敛候选集，拿到清除证书后清除。
6

```

```

7      与 P4 的差异: 保证接收盘内的 no_signal 说明探点被朝向遮挡
8      [观测推理], 走"对跖复核 -> 环探"升级梯子 (不计入定位步数,
9      由 max_blocked_probes 限流), 之后仍未收敛再转后备清除。
10     """
11     search = self.config.search
12     self._may_clear.add(channel)
13     steps = 0
14     while steps < search.max_localize_steps:
15         if self._virtual_cap_reached():
16             return False
17         region = self.belief.candidate_set(channel)
18         if region is None:
19             return False
20         cert = clear_certificate(region, channel, self.config)
21         if cert.certified and channel in self._may_clear:
22             if self._attempt_clear(channel, cert.center, cert):
23                 return True
24             self._may_clear.discard(channel) # 证书失败: 补测后才可重试
25             continue
26         if len(self.blocked_probes.get(channel, [])) >= search.max_blocked_probes:
27             break # 梯子用尽仍被遮挡: 定位停滞, 转后备清除
28         point = self._next_probe(region, channel)
29         within_range = (math.dist(point, region.mec()[0])
30             <= safe_disk_radius(region, self.config))
31         outcome = self.client.measure(point, channel)
32         self._observe(channel, point, outcome.result, outcome.svd_deg)
33         self._may_clear.add(channel)
34         if outcome.result == NO_SIGNAL:
35             if within_range and channel not in self.directional_observed:
36                 self.directional_observed.append(channel)
37                 self.blocked_probes.setdefault(channel, []).append(point)
38                 continue # 盘外无信号也可能是距离不足, 不据此断言定向
39             self.blocked_probes.pop(channel, None)
40             steps += 1
41         if self.belief.channels[channel].status is ChannelStatus.CLEARED:
42             return True
43         region = self.belief.candidate_set(channel)
44         if region is None:
45             return False
46         cert = clear_certificate(region, channel, self.config)
47         if cert.certified and channel in self._may_clear:
48             if self._attempt_clear(channel, cert.center, cert):
49                 return True
50             self._may_clear.discard(channel)
51         return self._fallback_clear(channel, region)
52
53     def _observe(self, channel: int, point: Vec2, result: str,
54                 svd_deg: float | None) -> None:
55         """同步更新凸候选集与 (位置 × 朝向) 联合可行集。
56
57         联合集只在频道首次被看见 (direction/near) 时建立: 从未被看见的
58         频道由覆盖证明负责缺席, 不需要朝向维度。联合集一旦为空, 直接给出
59         该频道的缺席证明 (严格, 与覆盖证明同一条件), 可提前停止对其检测。
60         """
61         self.belief.update_measure(channel, point, result, svd_deg)
62         if not self.config.search.directional_joint_belief:
63             return
64         joint = self.joint.get(channel)
65         if joint is None:
66             if result not in (DIRECTION, NEAR):
67                 return
68             joint = JointBelief(self.config)
69             self.joint[channel] = joint
70         joint.apply(point, result, svd_deg)
71         if joint.is_absent() and self.belief.channels[channel].status
72             is not ChannelStatus.CLEARED:
73             self.belief.mark_absent(channel)
74             self.absence_evidence[channel] = "joint_empty"
75             if channel not in self.joint_absent_channels:
76                 self.joint_absent_channels.append(channel)
77
78     def _next_probe(self, region: CandidateRegion, channel: int) -> Vec2:
79         """定向遮挡下的探点升级梯子 [工程]。
80
81         - 未被遮挡过: worst-posterior 选点器 (保证接收盘内);
82         - 第 1 次被遮挡: 对 MEC 圆心取对跖方向, 偏移裁进保证接收盘;
83         - 之后: 绕 MEC 圆心、半径 = 的环探, 方位自首个被遮挡点起

```

```

84     每次 +45°。几何边界必须如实陈述：环上可见弧没有下界——
85     源可位于环上（|g-c| = ）且朝向沿外切线，可见集退化为单个
86     切点，固定步进环探可以全部漏过。梯子只提高提前命中概率，
87     不构成接收或命中保证；完备性完全由后备清除兜底。
88     """
89     visibility = self._visibility_probe(region, channel)
90     if visibility is not None:
91         return visibility
92     blocked = self.blocked_probes.get(channel, [])
93     center, radius = region.mec()
94     if not blocked:
95         return select_measurement_point(region, self.client.state.position,
96                                         self.config).position
97     safe = safe_disk_radius(region, self.config)
98     first = blocked[0]
99     if len(blocked) == 1 and safe >= 0:
100         dx, dy = center[0] - first[0], center[1] - first[1]
101         norm = math.hypot(dx, dy)
102         if norm <= 1e-9:
103             # 首个被遮挡探点恰在圆心：方向信息为零，沿 +x 走满安全盘。
104             return (center[0] + safe, center[1])
105             scale = min(1.0, safe / norm)
106             # 对距方向，偏移模长 = min(norm, safe)：圆心对称点裁进保证接收盘。
107             return (center[0] + dx * scale, center[1] + dy * scale)
108         base = math.atan2(first[1] - center[1], first[0] - center[0])
109         azimuth = base + math.radians(45.0 * (len(blocked) - 1))
110         return (center[0] + radius * math.cos(azimuth),
111                center[1] + radius * math.sin(azimuth))
112
113 def _fallback_clear(self, channel: int, region: CandidateRegion) -> bool:
114     """定位停滞的后备：候选集有限圆盘覆盖逐格清除（格数即请求上界）。"""
115     search, session = self.config.search, self.config.session
116     width = max(p[0] for p in region.polygon) - min(p[0] for p in region.polygon)
117     height = max(p[1] for p in region.polygon) - min(p[1] for p in region.polygon)
118     cells = fallback_clear_cells(region.polygon, self.config)
119     entry = {"channel": channel, "cells": len(cells),
120             "bound": fallback_cell_bound(width, height, self.config),
121             "posterior_radius_m": round(region.mec()[1], 3)}
122     self.fallback_log.append(entry)
123     if len(cells) > search.fallback_max_cells:
124         # 宽候选集盲扫违背预算纪律：留待定位收敛，不得硬扫 [计划S4]。
125         entry["outcome"] = "skipped_cell_cap"
126         self._anomaly("fallback_cell_cap_exceeded", channel,
127                      cells=len(cells), cap=search.fallback_max_cells)
128         return False
129     # 现实预算核对：N 次请求 × 单请求高分位估计 + 退出余量，不足不启动。
130     remaining = self.client.real_remaining_s
131     if remaining is None or len(cells) * session.real_rtt_p95_s
132         + session.reserve_exit_s > remaining:
133         entry["outcome"] = "skipped_budget"
134         return False
135     attempts = 0
136     order = self._fallback_order(cells)
137     for cell in order:
138         if self._virtual_cap_reached():
139             entry["outcome"] = "skipped_virtual_budget"
140             entry["attempts"] = attempts
141             return False
142         attempts += 1
143         outcome = self.client.clear(cell, channel)
144         if outcome.result == CLEAR_SUCCESS:
145             self.belief.mark_cleared(channel)
146             self.cleared.append(channel)
147             entry["outcome"] = "cleared"
148             entry["attempts"] = attempts
149             return True
150         # 候选集保守包含真源，格心距格内任一点 <= 10*sqrt(2) < 20m：全部
151         # 失败说明守恒性被打破，按异常留痕，不得据此删除候选。
152         entry["outcome"] = "exhausted"
153         entry["attempts"] = attempts
154         self._anomaly("fallback_exhausted_without_success", channel,
155                      cells=len(cells))
156     return False

```