Pseu滑动窗口算法的伪随机生成器

Augusto Modanese 阿尔
托大学
augusto.modanese@aalto.fi

arXiv:2301.07384v1 [cs.CC] 18 Jan 2023

摘要

窗口大小为 t 的滑动窗口算法，是指其当前操作仅取决于最近读取的 t 个符号的算法。我们为能够访问二进制随机源的低空间随机化滑动窗口算法构造伪随机生成器（PRG）。更具体地说，我们将这些算法提升至非一致性设定下的分支程序，并将其作为分支程序的一个子类进行研究，称之为滑动窗口分支程序（SWBP）。对于一般的 SWBP，给定一个基础伪随机生成器 G_{base}，其种子长度为 d_{base}，且能够 $\varepsilon_{\text{base}}$-欺骗宽度为 w、长度为 t 的（一般）分支程序，我们给出两种伪随机生成器构造，用于欺骗任意具有相同宽度、长度为 n 且窗口大小为 t 的 SWBP（其中假设 $w \geq n$）。第一种构造额外使用 $d_{\text{base}} + O(\log(n/t)\log(1/\varepsilon_{\text{base}}))$ 个随机比特；第二种构造的种子长度为 $O((d_{\text{base}} + \log\log(n/t) + \log(1/\varepsilon_{\text{base}}))\log(d_{\text{base}} + \log(1/\varepsilon_{\text{base}})))$。两种伪随机生成器在误差参数上仅产生 $(n/2t)^{O(1)}$ 倍的乘性损失。我们还给出一种命中集生成器（HSG），在 $\log w$ 与 $\log(n/\varepsilon)$ 非常接近的情形下可实现略短的种子长度（例如，$\log w = (\log(n/\varepsilon))^{1.01}$）。

作为一项应用，我们展示如何用较小的空间判定亚线性时间概率细胞自动机的语言。更具体地说，这些结果针对的是概率交替细胞自动机模型；当且仅当所有元胞同时处于接受状态时，该模型才接受。对于（亚线性）$T(n) = \Omega(\log n)^{1.01}$，我们证明，任何可由$T$时间单侧错误概率细胞自动机（PACA，相当于 RP）接受的语言，都可以仅用 $O(T)$空间判定。另一方面，若不再要求满足先前对 T的要求，则我们证明，对于 T时间~双侧错误概率细胞自动机（PACA，相当于 BPP），使用 $O(T) + O(\log n)$空间也可得到相同结论~（其中 O 记号仅隐藏了 polylog(T) 因子）。

1 引言

在现代社会，尽可能少地使用内存资源来处理长数据流，是一种至关重要的计算范式。以流式算法的形式提出以来，这一主题一直是理论计算机科学领域深入研究的对象。然而，在处理快速过时的数据时，滑动窗口算法这一子类似乎能更准确地刻画这一过程。这类算法会在数据流中维护一个仅包含最近若干符号的窗口（因此得名），用于模拟只将最新数据视为准确或相关信息的过程。这种策略的典型应用包括天气预报和社交媒体分析。

滑动窗口算法的一个显著特征是自同步：如果两台机器在同一数据流上执行相同的算法，窗口大小为 t，并且它们在处理过程中某个时刻收到错误且不一致的数据，那么在读取至多 t 个相同符号后，两台机器将恢复到状态完全一致（即实现同步）。事实证明，这种行为不仅在实践中可能值得追求，从理论角度来看，它本身也可作为这类算法的定义性特征。在本文中，我们将采用这一观点

并研究具有这一性质的低空间算法，这些算法还可以访问一个（二进制）随机源。我们将尝试回答以下核心问题：具有这一性质的算法类需要多少（如果需要的话）随机性，才能完成其所需执行的任务?

随机滑动窗口算法。直观地说，只需让控制单元按照随机过程运行，我们就能得到滑动窗口算法的随机化版本。当然，这立即引出了一个问题：应如何将滑动窗口要求调整到这一新模型中。其定义性特征（在确定性情形下）是机器当前的操作仅取决于最近的 t 个已读取符号，因此，可以合理地要求算法的行为不仅取决于最近的 t 个符号，还取决于读取这些比特时所做的随机选择。由于在我们的设定中，真正的随机化算法预期每读取一个新符号至少使用一比特随机性，因此算法的窗口似乎很大一部分（或至少相当一部分）将包含来自随机源而非输入流的比特。实际上，在将滑动窗口性质推广到随机化算法时，我们将完全摒弃后者；也就是说，我们将仅用随机源来表述该性质，并利用非一致性，完全不提输入流。这种方法有几个优点:

1. 这不仅简化了表述（因为这样一来，算法只需在单个输入源上滑动窗口，而不是在两个输入源上滑动），还使该模型更适合采用分析低空间算法时常用的复杂性理论方法。

2. 由此得到的机器类实际上比滑动窗口性质同时适用于数据流和随机源的模型更强。因此，我们的结果不仅适用于后一种情况，也适用于更一般的情况。

如上文所述，我们将借助复杂性理论中的经典方法，分析去随机化背景下的低空间算法，具体而言，采用非一致的分支程序模型。

1.1 分支程序

分支程序可视为确定性有限自动机的非一致变体，在低空间算法的去随机化中有着广泛应用。

定义1. 一个（有序、只读一次）分支程序 P，长度为 n、宽度为 w，由一组状态 Q、$|Q| = w$ 构成，其中子集 $Q_{\text{acc}} \subseteq Q$ 为接受状态，并配有 n 个转移函数$P_1, \ldots, P_n$: $Q \times \{0,1\} \to Q$。程序 P从一个初始状态 $q_0 \in Q$开始运行，然后从左向右处理其输入 $x = x_1 \cdots x_n \in \{0,1\}^n$，并在第 i步根据 P_i更新状态。也就是说，如果 P在读取了输入中的 $i-1$ 个比特后处于状态 q_{i-1}，那么它接下来读取 x_i，并将当前状态变为 $P_i(q_{i-1}, x_i)$。如果 $q_n \in Q_{\text{acc}}$，我们就称 P 接受 x；其中 q_n 是 P 处理完最后一个输入符号 x_n后的状态。

尽管 P 在处理 x的过程中始终重复使用同一组状态，但将P视为具有 $n+1$ 层的有向无环图也很自然，其中第 i 层 Q_i 包含 Q 的每个节点，只要该节点恰好经过 i 步即可到达。（我们将 Q_i 既称为节点集合，也称为 Q 的子集。）于是，节点 $v \in Q_{i-1}$ 与 Q_i 中的节点 $u_0 = P_i(v, 0)$ 和 $u_1 = P_i(v, 1)$ 相连，并分别将对应的边标记为 0 和 1。（如果 $u_0 = u_1$，那么从 v 到 u_0之间就有一条重边。）

我们用 $P(x)$ 表示 P接受 x 的指示函数 $\{0,1\}^n \to \{0,1\}$（即，如果 P接受 x，则 $P(x) = 1$；否则 $P(x) = 0$）。对于满足 $|y| \le n$ 的 $y \in \{0,1\}^*$，我们用 $P_0(y)$ 表示 P 从其初始状态 q_0开始运行并读取 y 后的状态。令 λ 表示空字，则可通过递归定义如下：令 $P_0(\lambda) = q_0$，并令 $P_0(y'z) = P_{|y'|+1}(P_0(y'), z)$，其中 $y = y'z$且 $z \in \{0,1\}$。我们以自然的方式将 P_i的定义域从 $Q \times \{0,1\}$ 扩展到 $Q \times \{0,1\}^{\le n-i}$，如下所示： $P_i(q, \lambda) = q$，并且对于满足$y \in \{0,1\}$的 $yy' \in \{0,1\}^{\le n-i}$，$P_i(q, yy') = P_{i+1}(P_i(q, y), y')$。

一致性程序 [5] 是分支程序的一种推广：当且仅当计算过程中每个状态都是接受状态时，它才接受（而非只要求最终状态是接受状态）。形式上，这意味着我们将每个第 i 层的一个子集 $Q^i_{\text{acc}} \subseteq Q_i$ 标记为接受状态，并称一致性程序U 接受 $x \in \{0,1\}^n$，当且仅当对于 x 的每个前缀 $y \neq \lambda$，都有 $U_0(y) \in Q^{|y|}_{\text{acc}}$。（注意，这确实是一种推广，因为对每个 $i < n$ 设定 $Q^i_{\text{acc}} = Q$，就会得到一个标准分支程序。）容易看出，宽度为 w 的一致性程序可以由宽度为 $(w+1)$ 的标准分支程序模拟（例如，可以添加一个“失败”状态，以表示一致性程序在某个时刻没有接受。）

伪随机生成器。连接分支程序与去随机化这一主题的关键概念是伪随机生成器。它们是通用函数，接收一个（概念上）较小的输入子集，并将其“散布”到整个取值范围中，使其对于它们旨在欺骗的那类过程“看起来是随机的”。在下面的定义中，U_n表示一个在 $\{0,1\}^n$ 上均匀分布的随机变量。

定义 2。设 $n \in \mathbb{N}_+$和 $\varepsilon > 0$，并设 $\mathcal{F}$是函数类 $f\colon \{0,1\}^n \to \{0,1\}$。若函数 $G\colon \{0,1\}^d \to \{0,1\}^n$是一个伪随机生成器（PRG），且它对 $\mathcal{F}$具有 ε-欺骗性质，则称其满足以下条件：对每个 $f \in \mathcal{F}$，

$$|\Pr[f(G(U_d)) = 1] - \Pr[f(U_n) = 1]| \le \varepsilon.$$

此外，若存在一个（统一的）线性空间算法，输入 n（以二进制编码）和 $s \in \{0,1\}^d$后，输出 $G(s)$，则称 G 是显式的。

上述内容与低空间算法之间的联系如下：假设我们有一个 s-空间算法 A，它接收字符串 x 以及由随机抛硬币产生的数据流 $r \in \{0,1\}^n$ 作为输入。若我们取一个对宽度为 2^s、长度为 n的分支程序具有 ε-欺骗性质的伪随机生成器 G，那么$A(x, G(U_d))$与 $A(x, U_n)$必然相差不超过 ε。原因是，我们可以将 $A(x, \cdot)$转换为一个分支程序 P_x：将 A 的配置空间作为 P_x 的状态集，并以非一致的方式将 x 中被 A 读取的比特硬编码进 P_x的转移函数。由此得到的程序 P_x以 A 的随机流作为输入（尽管这或许有些违反直觉，但确实能得到所需的归约）。

研究分支程序伪随机生成器的终极目标之一，是解决 L 是否等于 BPL 这一问题。非构造性地，可以证明存在一种伪随机生成器，其最优种子长度为 $O(\log(w/\varepsilon))$，并且能够欺骗任意长度的宽度为 w 的分支程序，误差为 ε。如果我们能够证明存在一个具有这些参数且显式的伪随机生成器，那么这将意味着 L = BPL。目前已知的最佳 L 去随机化方法（忽略次对数因子）使用 $O(\log n)^{3/2}$ 空间 [34]。在这一背景下，我们的结果颇具意义，因为对于滑动窗口算法这一特殊情况，我们得到了种子长度最优或近乎最优（即除次对数因子外最优）的伪随机生成器。

滑动窗口分支程序。在本研究中，我们的目标是构造针对满足滑动窗口性质的一致性程序这一受限类别的伪随机生成器。正如前文所

讨论的，其定义性特征是自同步。

定义 3. 若一个滑动窗口分支程序 (SWBP) 是满足以下性质的一致性程序 S：存在一个数 $t \in \mathbb{N}_+$，称为 S 的窗口大小，使得对于每个$i \leq n-t$、每个 $y \in \{0,1\}^t$以及每一对状态 $q, q' \in Q_i$，都有 $S_i(q,y) = S_i(q',y)$。

换句话说，S 的当前状态完全取决于最近读入的 t 比特。从这个角度来看，不难看出，要使滑动窗口性质具有实际意义，确实有必要将一致性程序推广到更一般的情形。事实上，如果一个 SWBP S 是标准分支程序（也就是说，不只是一个一致性程序），那么它的判定只取决于其输入中最后的 t 比特。

SWBP 与滑动窗口算法之间的关系，类似于定义 2 之后所述的一般分支程序与低空间算法之间的关系。关于滑动窗口算法的已有文献，请参阅下文第1.4节。

为了说明动机，我们举一个非常自然的例子，展示具有这一性质的一类随机化算法。（由于该方案涵盖了整整一类算法，因此我们以相当抽象的方式介绍它。）假设给定输入 x，并且可以执行一个低空间随机化过程 T，用于测试 x 的某种性质（可能以某些辅助输入为条件）。那么，基于 T，有一种自然的滑动窗口算法 $A = A_T$：它只需在 x 上执行 T 指定次数 $m = m(|x|)$（每次执行都使用独立的抛硬币），并且当且仅当每次执行 T 都成功时才接受。滑动窗口性质源于这样一个事实：T 的每次执行彼此独立，而 A 只关心 T 是否返回肯定答案（也就是说，A 不会根据先前测试的结果调整下一次测试）。

1.2 我们的结果

我们的核心结果是为滑动窗口分支程序模型构造最优或近乎最优的伪随机生成器。作为一项应用，我们对某种亚线性时间概率细胞自动机模型所接受的语言进行去随机化。此外，我们还给出了滑动窗口分支程序的结构刻画，下面将对此展开讨论。

1.2.1 滑动窗口分支程序的结构刻画

设 $n \in \mathbb{N}_0$。回顾一下，维数为 n 的德布鲁因图 是有向图 $B_n = (V_n, E_n)$，其顶点集为 $V_n = \{0,1\}^n$，边集为

$$E_n = \{(xw, wy) \mid w \in \{0,1\}^{n-1} \text{ and } x, y \in \{0,1\}\}.$$

类似地，维数为 n 的前缀树 是图 $T_n = (V'_n, E'_n)$，其 $V'_n = \{0,1\}^{\leq n}$ 和

$$E'_n = \{(w, wx) \mid w \in \{0,1\}^{\leq n-1} \text{ and } x \in \{0,1\}\}.$$

作为我们的首项贡献，我们注意到，对于每个固定的窗口大小 t 和输入长度 n，都存在一个“典型的”滑动窗口分支程序 Π，其拓扑结构满足：前 t 层与前缀树 T_t 同构，后续各层则按照德布鲁因图 B_t（的带标记版本）连接。因此，每个窗口大小为 t 的滑动窗口分支程序 S，都可以通过合并同一层中的节点，从这个典型滑动窗口分支程序 Π得到。（请注意，为确保对于每个 i，S_i 确实是一个函数，这一过程还可能需要合并后续各层中的节点。）

定理 4. 一致性程序 S 是一个窗口大小为 t 的滑动窗口分支程序，当且仅当存在函数α_0、……、 α_n: $\{0,1\}^{k_i} \to Q$，满足

$$k_i = \begin{cases} i, & i < t; \\ t, & i \geq t \end{cases}$$

并且对每个 x、 $y \in \{0,1\}$ 和 $w \in \{0,1\}^{k_i - 1}$，以下条件均成立 :

1. 对每个 $i < t(\alpha_i(w)\ y)\ = \alpha_{i+1}(wy)$。

2. 对每个 $t \leq i < n(\alpha_i(xw)\ y)\ = \alpha_{i+1}(wy)$。

由这一刻画可立即得出，只要 S 没有冗余状态，就有 $w \leq 2^t$。

1.2.2 用于滑动窗口分支程序的伪随机生成器

我们给出两种欺骗滑动窗口分支程序的伪随机生成器构造，具体采用哪一种可能取决于当前参数范围。

定理 5。 给定 n、 w、 $t \in \mathbb{N}_+$，其中 $n \leq w$ 且 $\varepsilon > 0$，并设 G_{base}: $\{0,1\}^{d_{\text{base}}} \to \{0,1\}^t$ 是一个能以 $\varepsilon_{\text{base}}$ 伪装宽度为 w、长度为 t 的一致性程序的伪随机生成器。那么有如下结论:

1. 存在一个显式伪随机生成器 G: $\{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d = 2d_{\text{base}} + O(\log(n/t)\log(1/\varepsilon_{\text{base}}))$$

能以 ε_G 伪装任意窗口大小为 t、宽度为 w、长度为 n 的滑动窗口分支程序，其中 $\varepsilon_G = \varepsilon_{\text{base}} \cdot (n/t)^{O(1)}$.

2. 存在一个显式伪随机生成器 G: $\{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d' = O(d_{\text{base}} + \log(n/t) + (\log(1/\varepsilon_{\text{base}}))^{3/2})$$

能以 ε_G 能欺骗任意窗口大小为 t、宽度为 w、长度为 n 的滑动窗口分支程序，其中 $\varepsilon_G = O(\varepsilon_{\text{base}} \cdot n/t)$。

3. 存在一个显式伪随机生成器 G: $\{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d' = O((d_{\text{base}} + \log\log(n/t) + \log(1/\varepsilon_{\text{base}}))\log(d_{\text{base}} + \log(1/\varepsilon_{\text{base}})))$$

能以 ε_G 能欺骗任意窗口大小为 t、宽度为 w、长度为 n 的滑动窗口分支程序，其中 $\varepsilon_G = O(\varepsilon_{\text{base}} \cdot n/t)$。

为了直观了解，与能欺骗一般分支程序的伪随机生成器相比，这种方法有何改进，我们先考虑 G_{base} 是 Nisan [31] 的经典生成器这一情况。假设 $n \leq w$，该生成器能以 $\varepsilon_{\text{base}}$-欺骗任意宽度为 w、长度为 n 的分支程序，所用种子长度为

$$d_{\text{base}} = O(\log(w/\varepsilon_{\text{base}})\log n).$$

还假设 $t = o(n)$，这通常是最有趣的情形。于是，定理 5 的第二项和第三项分别给出了种子长度为

$$d_2 = O(\log(w/\varepsilon)\log t + (\log(n/\varepsilon))^{3/2})$$

and

$$d_3 = O(\log(w/\varepsilon)\log\log(w/\varepsilon)\log t)$$

任意所需 $\varepsilon > 0$ 的伪随机生成器。（为使讨论简洁，我们暂且不考虑定理 5 的第一项，但请注意，与 d_2 相比，它在 ε 较小时给出了更好的依赖关系。）对于满足 $\log(w/\varepsilon) = \Omega(\log(n/\varepsilon))^{3/2}$ 的范围，这意味着 d_{base} 中的 $\log n$ 因子变成了 d_2 中的 $\log t$ 因子。对于大多数应用而言，这是一个显著的改进，因为窗口大小 t 很可能远小于输入长度 n。与此同时， d_3 与 n 独立，这表明它更适合用于欺骗宽度非常接近 n 的程序；这对应于空间复杂度较低的算法（即接近 $\log n$）。

目前已知用于欺骗一般分支程序的最佳生成器是 Armoni [1], 提出的显式生成器 G_{A}: $\{0,1\}^{d_{\text{A}}} \to \{0,1\}^n$（结合 Kane、Nelson 和 Woodruff [23]的改进） ε_{A}-欺骗任意宽度为 w、长度为 n 的分支程序，其中 $\varepsilon_{\text{A}} > 0$ 和n， $w \in \mathbb{N}_+$ 均可任意选取（特别地，它也能 ε_{A}-欺骗任意宽度为 $(w-1)$、长度为 n 的一致性程序）。同样假设 $n \le w$, G_{A} 的种子长度为

$$d_{\text{A}} = O\left(\frac{\log(w/\varepsilon_{\text{A}})\log n}{\max\{1, \log\log w - \log\log(n/\varepsilon_{\text{A}})\}}\right).$$

给定 $\varepsilon > 0$，在定理 5 中代入 $G_{\text{base}} = G_{\text{A}}$ 和 $\varepsilon_{\text{base}} = \varepsilon_{\text{A}} = \varepsilon \cdot (2t/n)^{\Omega(1)}$，可得：

推论 6。 设给定 $n, w, t \in \mathbb{N}_+$，其中 $n \le w$ 且 $\varepsilon > 0$。此外，设

$$d' = O\left(\frac{\log(w/\varepsilon)\log t}{\max\{1, \log\log w - \log\log(n/\varepsilon)\}}\right).$$

则有以下结论：

1. 存在一个显式伪随机生成器 G: $\{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d = O(d' + \log(n/t)\log(n/t\varepsilon))$$

它能欺骗任意宽度为 w、长度为 n 、窗口大小为 t 的滑动窗口分支程序 ε。

2. 存在一个显式伪随机生成器 G: $\{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d = O(d' + (\log(n/t\varepsilon))^{3/2})$$

它能欺骗任意宽度为 w、长度为 n 、窗口大小为 t 的滑动窗口分支程序 ε。

3. 存在一个显式伪随机生成器 G: $\{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d = O((d' + \log(n/t\varepsilon))\log(d' + \log(n/t\varepsilon)))$$

它能欺骗任意宽度为 w、长度为 n 、窗口大小为 t 的滑动窗口分支程序 ε。

如果我们采用第二项，并考虑这样的情形：$\log w = \Omega(\log(n/\varepsilon))^{3/2}$，且窗口大小 t 为 $\text{polylog}(w)$，那么得到的种子长度为 $O(\log(w/\varepsilon))$，这基本上是最优的。同时，第一项中的伪随机生成器更适合类似但 ε 值较小的情形（例如 $\varepsilon = 2^{-\Omega(\log n)^2}$）。而第三项中的伪随机生成器对于 $\log w$ 介于 $\Omega(\log(n/\varepsilon))$ 和 $O(\log(n/\varepsilon))^{3/2}$之间的情形近乎最优，这使我们几乎完全弥合了空间复杂度谱系低端算法所遗留的差距。

滑动窗口分支程序的命中集生成器。对上述两种构造中的第二种稍作调整后，如果将目标从伪随机生成器放宽为仅需一个命中集生成器（HSG），我们便能缩短种子长度。与伪随机生成器不同，对于我们希望欺骗的函数类 $\mathcal{F}$，命中集生成器不必在其整个取值范围内都表现得随机；它只需保证函数类 $\mathcal{F}$中的函数无法完全避开其取值范围（同样允许一定的误差余量 ε）。

定义 7. 设 $n \in \mathbb{N}_+$和 $\varepsilon > 0$，且 $\mathcal{F}$是函数类 $f\colon \{0,1\}^n \to \{0,1\}$。如果对每个 $f \in \mathcal{F}$都满足以下条件，我们称函数 $G\colon \{0,1\}^d \to \{0,1\}^n$是 $\mathcal{F}$的 ε-命中集生成器：

$$\Pr[f(U_n) = 1] \geq \varepsilon \implies \exists x \in \{0,1\}^d : f(G(x)) = 1.$$

同样，如果存在一个（统一的）线性空间算法，在输入 n （以二进制编码）和 $s \in \{0,1\}^d$后，输出 $G(s)$，则称 G 是显式的。

定理 8. 给定 n、w、$t \in \mathbb{N}_+$，其中 $n \leq w$，且 $\varepsilon > 0$ 已给定，并设 $G_{\text{base}}\colon \{0,1\}^{d_{\text{base}}} \to \{0,1\}^t$是一个对宽度为 w、长度为 t 的一致性程序进行 $\varepsilon_{\text{base}}$-伪随机化的伪随机生成器。则存在一个显式的 ε_G-命中集生成器 $G\colon \{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d = O(d_{\text{base}} + \log(n/t) + \log(1/\varepsilon_{\text{base}}))$$

对于窗口大小为 t 的宽度-w、长度-n 滑动窗口分支程序，其中 $\varepsilon_G = O(\varepsilon_{\text{base}} \cdot (n/t))$。

像之前一样代入 Armoni 的伪随机生成器，我们得到：

推论 9。给定 n、w、$t \in \mathbb{N}_+$，其中 $n \leq w$ 且 $\varepsilon > 0$。此外，设

$$d' = O\left(\frac{\log(w/\varepsilon)\log t}{\max\{1, \log\log w - \log\log(n/\varepsilon)\}}\right).$$

那么存在一个显式的 ε-命中集生成器 $G\colon \{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d = O(d' + \log(n/t) + \log(1/\varepsilon))$$

对于窗口大小为 t 的宽度-w、长度-n 滑动窗口分支程序。

由于推论 6 已经给出了一个种子长度相近的伪随机生成器（相比之下，得到命中集生成器并不那么理想），这里我们将重点讨论 $\log w = o(\log(n/\varepsilon))^{3/2}$ 的情形。与推论 6 后的讨论类似，如果 $t = \text{polylog}(w)$，那么我们就能得到一个种子长度为 $O(\log(w/\varepsilon))$ 的命中集生成器，其种子长度基本达到最优。区别在于，现在即使在 $\log w = \Omega(\log(n/\varepsilon))^{1+\delta}$ 对某个 $\delta > 0$ 成立的情形下（而不仅仅像之前那样在 $\log w = \Omega(\log(n/\varepsilon))^{3/2}$ 时），结论也成立。

1.2.3 概率细胞自动机的应用

作为一项应用，我们得到了用于判定亚线性时间概率细胞自动机所接受语言的空间高效算法。更具体地说，我们的结果针对 PACA（概率 ACA）模型 [30],：这类自动机具有两个局部转移函数 δ_0 和 δ_1，并且在每一步中，每个单元格都会抛掷一枚公平硬币 $c \in \{0,1\}$，然后根据 δ_c更新其状态。其接受条件与 ACA 的相同，也是最常见的一种 [21, 24, 29,35]，允许进行非平凡的亚线性时间计算：当且仅当到达某个所有单元格都处于接受状态的配置时，计算才接受。我们的结果适用于单边错误 和双边错误 PACA；从某种意义上说，它们分别是经典复杂性类 RP 和 BPP 在 PACA 中的对应形式。（定义请参阅第6.1节。）

这些结果很有意思，因为在高效判定 PACA 的语言这一相关情形中，若能得到类似的正面结果（此处的效率按时间 复杂度衡量），就会对图灵机的去随机化产生出人意料的影响（例如 P =RP） [30]。

单边错误概率可访问电路自动机。 在本研究之前，已知每种由确定性 ACA 在时间复杂度为 T 的条件下接受的语言，都可以用 $O(T)$ 空间判定 [29]。对于一般的单侧错误概率细胞自动机，我们得到以下结果：

定理 10。 设 $\varepsilon > 0$ 任意，且 $T\colon\ \mathbb{N}_+ \to \mathbb{N}_+$ 是一个（可构造的）函数。那么，对于任何识别其语言 $L(C)$ 、运行时间至多为 $T = T(n)$ 的单边错误 ε- 错误 *PACA* C，都存在一个确定性算法，用于 $L(C)$，其空间复杂度为

$$O\left(\frac{(T + \log(1/\varepsilon))\log T}{\max\{1, \log T - \log\log(n/\varepsilon)\}} + \log(1/\varepsilon) + \log n\right).$$

因此，对于（例如） $T(n) \geq (\log\ n)^{1.01}$ 且 ε 为常数的情况，可知使用 $O(T)$ 空间，我们也能判定任何由运行时间为 T 的（单边错误）概率 ACA 所接受的语言。这里可以将 ε 指定为任意值（尤其允许其依赖于 n），这是一个很大的优点，因为我们目前只知道将 PACA 的错误率降低一个常数 因子 [30] 的直接构造方法。

双边错误 PACA。 对于双边错误 PACA，我们距离实现与上述相同的空间复杂度仅差一步。

定理 11。 设 $T\colon\ \mathbb{N}_+ \to \mathbb{N}_+$ 是一个函数，且 $\varepsilon > 0$ 任意。对于任何识别其语言 $L(C)$ 、运行时间至多为 $T = T(n)$ 的双边$(1/2 - \varepsilon)$错误 *PACA* C，都存在一个用于 $L(C)$ 的确定性算法，其空间复杂度为 $O((d' + \log\ n)\log(d' + \log\ n))$，其中$d' = O((T + \log(1/\varepsilon))\log\ T)$。

因此，我们得出：任何能被运行时间为 T、具有双边错误的概率细胞自动机（即错误概率为常数 ε）接受的语言，都可以用 $O(T(\log\ T)^2)$ 的空间确定性地判定。事实上，即使我们只有 $\varepsilon = 2^{-O(T)}$，相同的空间复杂度上界仍然成立。作为比较，如果直接使用 Armoni 的伪随机生成器，那么所得空间复杂度（对于常数 ε）为 $O(T\log n)$；因此，对于运行时间为 $2^{O(\sqrt{\log n})}$ 的概率细胞自动机，我们的方法有所改进。

1.3 技术概述

适用于一般滑动窗口分支程序的生成器。我们的构造依赖于两个主要思想，其中第一个特别适合利用滑动窗口性质。借助这项我们称为交错的技术，我们能够有效地将滑动窗口分支程序“打散”为一组短程序。为了说明这一思想，设有一个窗口大小为 t 的滑动窗口分支程序 S，并考虑它在输入 S 上的行为

$$X \parallel Y = X_1Y_1X_2Y_2 \cdots X_nY_n$$

其中 X_i 和 Y_i 的长度都为 t 和 $X = X_1 \cdots X_n$，且 $Y = Y_1 \cdots Y_n$ 彼此独立地选取（但可能服从相同的分布）。关键在于，由于 S 在读取 X_i 和 X_{i+1}之间必须读取 Y_i 的 t 比特，因此当它开始处理 X_{i+1} 时，实际上已经“忘记”了有关 X_i的所有信息。此外，由于 X 和 Y 彼此独立地选取， Y_i 完全不包含关于 X_{i+1} 的信息； Y_i 与 X_{i+1} 有关的唯一作用，是决定处理 X_{i+1} 从哪个状态开始。因此，当我们的生成器的输出形式为 $X \parallel Y$时，我们只需（比如说）将 Y设为某个固定字符串 y，然后将 S作为由 n 个长度为 t 的一致性程序组成的集合来分析，其中每个程序都接收一个 X_i 作为输入。

因此，我们已将原始任务归约为同时愚弄一组 n 程序；也就是说，给定宽度和长度相同的一致性程序 $P_1, \ldots, P_n$，我们希望生成

伪随机输入 $X_1, \ldots, X_n$，使得

$$\left| \Pr[\forall i \in [n] : P_i(X_i) = 1] - \prod_{i=1}^{n} \Pr[P_i(U) = 1] \right|$$

很小，其中 U 是 P_i所有可能输入上的均匀分布。事实证明，我们现在可以采用两 种方法，在不同的参数范围内，各有其优势。

如 INW 生成器那样进行同时欺骗。 我们给出的第一个构造采用了 Impagliazzo、Nisan 和 Wigderson 的生成器 [22]背后的一项关键思想：假设我们有一个生成器 G，它能使用长度为 d的随机种子 s 同时欺骗 $n/2$ 许多程序，那么我们就使用一个合适的提取器 （定理 15），用“仅多出的少量”随机比特生成一个新种子 s'，然后输出拼接结果 $G(s)G(s')$。（定义参见第 2 节。）在我们的情形中，这一策略实际上达到了尽可能好的效果。这是因为前 $n/2$ 个程序处理完 $G(s)$后，只泄露了极少的熵；也就是说，关于 s，我们只能获知 P_1、……、$P_{n/2}$ 都接受了，这仅包含常数量的信息。

通过归纳法，我们可以同时欺骗任意数量 $n \geq 2$ 的一致性程序。对于基例 $n = 1$，我们代入任意一个自己选定的、用于一致性程序的生成器 G_{base}；关键在于，G_{base} 只需欺骗长度与原始滑动窗口分支程序的窗口大小 t 相同的程序 S （而不必欺骗长度与其自身 S 相同的程序）。

使用用于组合矩形的伪随机生成器实现同时欺骗。定理 5 的第二项需要采用不同的方法。总体思路让人想起 Hoza 和 Zuckerman 的一项工作[20], ，不过此前的文献中很可能也有类似的构造。我们不再像之前那样试图一遍又一遍地“回收”种子，而是改为选取 m 多个独立种子 s_1，……，s_n 并输出它们的拼接 $G_{\text{base}}(s_1) \cdots G_{\text{base}}(s_n)$。根据构造，这样便能保证欺骗这些程序 n （代价是误差参数中一个非常温和的因子 n ）。接下来关键的一步是，通过另一种构造生成种子本身来缩短种子长度；这种构造能够欺骗另一类组合对象，即所谓的组合矩形。

定义 12。 设 m，$n \in \mathbb{N}_+$。一个 (n, m)-组合矩形 是一个函数 $f\colon (\{0,1\}^m)^n \to \{0,1\}$，它由 n 个布尔函数 $f_i\colon \{0,1\}^m \to \{0,1\}$ 的乘积复合而成；也就是说，

$$f(x_1, \ldots, x_n) = \prod_{i=1}^{n} f_i(x_i).$$

这有两个原因：首先，这些对象非常契合我们同时欺骗的概念。若将每个 f_i 与一个程序 P_i关联起来，那么 $f(x_1, \ldots, x_n) = 1$ 恰好对应于同时接受所有各自输入 x_i 的程序集合；此外，与我们的 P_i一样，每个 f_i 的输出彼此完全独立。其次，针对这些对象的伪随机构造文献相对丰富；与一般的分支程序不同，已有渐近最优或近乎最优的结果（分别参见第 4 节和第 5 节中的定理 19 和 23）。

最后，为了得到定理 8 中的命中集生成器，我们将上述针对组合矩形的伪随机生成器替换为 Linial 等人提出的（渐近）最优命中集生成器构造 [25]。分析过程类似，但重新适配仍需要一些工作。详情请参见第 5 节。

1.4 相关工作

分支程序。在复杂性理论中，对低空间算法进行去随机化时，标准做法是将其提升到更一般的（非一致）分支程序模型，因为该模型更适合进行组合分析。这种方法至少可以追溯到 30 年前 Nisan 的开创性工作 [31]。此后，针对不同情形下的分支程序去随机化研究取得了进展，例如，每一层的转移函数都是置换的分支程序 [19] ，或以某种固定但未知的顺序读取输入的分支程序 [10]。Bogdanov 等人最近提出了一致性程序 [5]。据我们所知，我们是首次研究具有滑动窗口性质或任何类似性质的分支程序。

滑动窗口算法。 滑动窗口范式是一种自然的流处理形式，已在数据库管理系统、 [4], 网络监控[7], 和强化学习 [15]等背景下得到研究。自 Datar 等人的工作 [8],以来，滑动窗口模型也在数据流统计数据维护的背景下得到了广泛研究。（例如，参见 [6] 中的相关综述。）

Ganardi 等人 [11, 12, 13, 14] 也在语言识别等领域研究了滑动窗口算法。我们指出，本文考虑的模型与他们的模型之间存在几个根本区别：

- 他们的结果也适用于非一致情形，但参数由窗口大小决定。具体而言，他们的模型允许同一数据流因窗口大小不同而表现出截然不同的行为。而在我们的情形中，非一致性由输入大小（即数据流长度）决定。

- Ganardi 等人的工作所采用的底层概率模型是 Rabin 的概率自动机模型 [33], ，该模型允许按照任意分布进行状态转移。相比之下，我们的模型从二进制源获取随机性，而且最重要的是，滑动窗口性质适用于随机输入（而 Ganardi 等人研究的模型中，该性质仅适用于数据流）。

总之，Ganardi 等人关注的是一种模型：它会验证（或计算某个量），对象是其数据流中每个固定大小的窗口；而我们关注的模型则验证整个数据流。

我们还提及 Pacut 等人最近的一篇论文 [32] ，该论文指出了分布式算法与滑动窗口算法之间的联系。我们对概率细胞自动机的应用，可以视为这一联系的直接推论。

概率细胞自动机。我们对概率细胞自动机的研究，为近期一系列旨在研究亚线性时间细胞自动机的成果 [28, 29, 30]增添了新的关联。如 [29], 所述，细胞自动机领域普遍似乎忽视了这一主题；据我们所知，在这一系列近期论文之前，关于该主题的理论成果仅有 [21, 24, 35]。Arrighi、Schabanel 和 Theyssier 此前提出过一种与我们所研究的概率细胞自动机类似的概率模型 [3], ，但[30] 是首项探讨亚线性时间情形的工作。

最后请注意，我们遵循 [30] 的用法，使用“概率”一词（因为它们与概率图灵机相似）来指称这些自动机，并将它们与更一般的随机细胞自动机 区别开来。在后者中，局部转移函数可以服从任意分布（与前文提及的 Rabin [33] 的工作思路相同）。遗憾的是，对于

文献中这两个术语之间的区别尚未达成共识，而且二者一直被交替使用。关于随机细胞自动机的综述，参见 [27]。

1.5 组织结构

本文其余部分的组织结构如下：第 2 节回顾所需的基本定义和结果。后续各节分别介绍一组结果：第3节讨论滑动窗口分支程序的结构性结果。第 4 节和第 5 节分别给出定理5和定理8的证明。最后，第6节介绍其在概率细胞自动机中的应用。

2 预备知识

假定读者熟悉计算复杂性理论和伪随机性的基本概念（例如，参见标准参考文献 [2, 16, 36]）。

所有对数均以 2 为底。整数集记为 $\mathbb{Z}$，非负整数集记为 $\mathbb{N}_0$，正整数集记为 $\mathbb{N}_+$。对于集合 S 和 n，$m \in \mathbb{N}_+$，$S^{n\times m}$ 表示元素取自 S 的 n 行、m 列矩阵集合。对于 $n \in \mathbb{N}_+$，$[n] = \{i \in \mathbb{N}_0 \mid i < n\}$ 表示前 n 个非负整数构成的集合。

单词中的符号从一开始编号。单词的第 i 个符号 w 记作w_i。对于字母表 Σ 和 $n \in \mathbb{N}_0$，$\Sigma^{\le n}$ 包含满足 $|w| \le n$的单词 $w \in \Sigma^*$。不失一般性，空字不属于我们所考虑的任何语言。

我们用 U_n（分别用 $U_{n\times m}$）表示在 $\{0,1\}^n$（分别在$\{0,1\}^{n\times m}$）上均匀分布的随机变量。对于随机变量 X，Supp(X) 表示 X 的支撑集。我们将需要以下 Chernoff 界的变体（例如参见 [36]）：

定理13。设 X_1、…、X_n 是独立同分布的伯努利随机变量，且$\mu = \mathrm{E}[X_i]$。则存在常数 $c > 0$，使得对每个 $\varepsilon > 0$，以下结论成立：

$$\Pr\left[\left|\frac{\sum_{i=1}^{n} X_i}{n} - \mu\right| > \varepsilon\right] < 2^{-cn\varepsilon^2}.$$

哈希函数。对于 N，$M \in \mathbb{N}_+$，若函数族 $H = \{h\colon [N] \to [M]\}$ 满足以下条件，则称其为两两独立：对 x_1，$x_2 \in [N]$，其中 $x_1 \ne x_2$ 和 h 从 H 中均匀选取时，随机变量$h(x_1)$和 $h(x_2)$相互独立且服从均匀分布。等价地，若对于任意 y_1，$y_2 \in [M]$ 均有以下结论，则称 H 两两独立：

$$\Pr\left[h(x_1) = y_1 \wedge h(x_2) = y_2\right] = \frac{1}{M^2}.$$

众所周知（例如，参见 [36]），存在一个两两独立函数族 H，可以用 $O(\log N + \log M)$ 比特从 H 中均匀采样一个函数 h。

提取器。设 $n \in \mathbb{N}_+$，且 X 和 Y 是取值于 $\{0,1\}^n$ 的随机变量。那么 X 与 Y 之间的统计距离 为

$$\Delta(X,Y) = \frac{1}{2}\sum_{w} |\Pr[X = w] - \Pr[Y = w]|.$$

X 的最小熵 $\mathrm{H}_\infty(X)$定义为

$$\mathrm{H}_\infty(X) = \min_{w \in \mathrm{Supp}(X)} \log \frac{1}{\Pr[X = w]}.$$

对于 $k \le n$，如果 $\mathrm{H}_\infty(X) \ge k$，则称 X 为一个 k-源。对于 d， $m \in \mathbb{N}_+$ 和 $\varepsilon > 0$，(k， ε)-提取器 是一个函数 $\mathrm{Ext}: \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$，满足对于每个 k-源 X，都有$\Delta(\mathrm{Ext}(X, U_d), U_m) \le \varepsilon$。在此语境下，Ext 的第二个参数是其种子，相应地，d 是其种子长度。

3 德布鲁因图完全刻画滑动窗口分支程序

在本节中，我们证明：

定理 4. 一致性程序 S 是窗口大小为 t 的滑动窗口分支程序，当且仅当存在函数$\alpha_0, \ldots, \alpha_n$: $\{0,1\}^{k_i} \to Q$ 满足

$$k_i = \begin{cases} i, & i < t; \\ t, & i \ge t \end{cases}$$

并且对每个 x、 $y \in \{0,1\}$ 和 $w \in \{0,1\}^{k_i - 1}$，以下条件均成立 ：

1. 对每个 $i < t$、 $S_i(\alpha_i(w), y) = \alpha_{i+1}(wy)$。
2. 对每个 $t \le i < n$、 $S_i(\alpha_i(xw), y) = \alpha_{i+1}(wy)$。

证明。 我们先证明正向蕴含。设 S 是一个窗口大小为 t的滑动窗口分支程序。 α_i 递归定义，其中基础情形为 $\alpha_0(\lambda)$，它被设为 S的初始状态。定义了α_i （ $i < n$）后，我们设定 α_{i+1} ，使得

$$\alpha_{i+1}(wy) = \begin{cases} S_i(\alpha_i(w), y), & i < t; \\ S_i(\alpha_i(xw), y), & i \ge t \end{cases}$$

对于 x， $y \in \{0,1\}$ 和 $w \in \{0,1\}^{k_i - 1}$ （因此自动满足定理陈述中的要求）。

还需证明 α_i 定义良好，我们将用归纳法证明这一点。归纳基础 $i = 0$ 是显然的，而 $i < t$ 的归纳步骤通过应用归纳假设以及 S_i 是一个函数即可轻松得出。因此，设 $i \ge t$。令 $w \in \{0,1\}^{t-1}$ 和 $y \in \{0,1\}$，我们将证明 $S_i(\alpha_i(0w), y) = S_i(\alpha_i(1w), y)$。根据归纳假设和 α_i的性质，存在 S 的第（$i - t + 1$）层中的状态 q_0 和 q_1，使得对于$x \in \{0,1\}$， $S_{i-t+1}(q_x, w) = \alpha_i(xw)$。因此，根据 S 的滑动窗口性质（并再次根据 α_i的性质），

$$S_i(\alpha_i(0w), y) = S_{i-t+1}(q_0, wy) = S_{i-t+1}(q_1, wy) = S_i(\alpha_i(1w), y).$$

对于逆向蕴含，设给定陈述中的 α_i 。接下来我们说明，对于任意 i 和 $z \in \{0,1\}^t$ ，以及在 S 的第 i 层可达的任意状态 q 和 q' ，都有 $S_i(q, z) = S_i(q', z)$。只要 q 和 q' 属于 α_i 的像，这一点通过归纳法便可简单得出。为证明确实如此，注意到，如果输入 $wz \in \{0,1\}^i$ 满足 $|z| = t$，且 q 可通过该输入在第 i 层可达，那么 $q = \alpha_i(z)$（同样，这是由滑动窗口性质导致的）；或者，在 $|z| = 0$ 且 $i < t$ 的情况下， $q = \alpha_i(w)$。两种情况下，结论均成立。 □

4 滑动窗口分支程序的伪随机生成器

在本节中，我们回顾并证明：

定理 5. 设给定 n、w、$t \in \mathbb{N}_+$，其中 $n \le w$ 且 $\varepsilon > 0$，并设 $G_{\text{base}}\colon \{0,1\}^{d_{\text{base}}} \to \{0,1\}^t$ 是一个伪随机生成器，能够 $\varepsilon_{\text{base}}$-欺骗宽度为 w、长度为 t 的一致性程序。则有以下结论：

1. 存在一个显式伪随机生成器 $G\colon \{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d = 2d_{\text{base}} + O(\log(n/t)\log(1/\varepsilon_{\text{base}}))$$

它能以 ε_G 的误差欺骗任意宽度为 w、长度为 n、窗口大小为 t 的滑动窗口分支程序，其中 $\varepsilon_G = \varepsilon_{\text{base}} \cdot (n/t)^{O(1)}$。

2. 存在一个显式伪随机生成器 $G\colon \{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d' = O(d_{\text{base}} + \log(n/t) + (\log(1/\varepsilon_{\text{base}}))^{3/2})$$

它能以 ε_G 的误差欺骗任意宽度为 w、长度为 n、窗口大小为 t 的滑动窗口分支程序，其中 $\varepsilon_G = O(\varepsilon_{\text{base}} \cdot n/t)$。

3. 存在一个显式伪随机生成器 $G\colon \{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d' = O((d_{\text{base}} + \log\log(n/t) + \log(1/\varepsilon_{\text{base}}))\log(d_{\text{base}} + \log(1/\varepsilon_{\text{base}})))$$

它能以 ε_G 的误差欺骗任意宽度为 w、长度为 n、窗口大小为 t 的滑动窗口分支程序，其中 $\varepsilon_G = O(\varepsilon_{\text{base}} \cdot n/t)$。

如第 1.3 节所述，在我们的构造中，我们将首先使用 G_{base} 来获得一个伪随机生成器，使其能够同时欺骗尽可能多的一致性程序。然后，我们使用交错将所得构造转换为适用于滑动窗口分支程序的伪随机生成器。我们将按此顺序处理这两个步骤。

4.1 同时欺骗

定义 14。 设 $m, t \in \mathbb{N}_+$ 和 $\varepsilon > 0$，并设 $\mathcal{F}$ 为函数类 $f\colon \{0,1\}^t \to \{0,1\}$。若分布 $X = (X_1, \ldots, X_m)$ 在 $(\{0,1\}^t)^m$ m-同时 ε-欺骗 $\mathcal{F}$，则称其满足以下条件，对每个 $f_1, \ldots, f_m \in \mathcal{F}$ 均成立：

$$\left| \Pr[\forall i \in [m] : f_i(X_i) = 1] - \prod_{i=1}^{m} \Pr[f_i(U_t) = 1] \right| \le \varepsilon.$$

类似地，若函数 $G\colon \{0,1\}^d \to (\{0,1\}^t)^m$ m-同时 ε-欺骗 $\mathcal{F}$，则称 $G(U_d)$ 欺骗 $\mathcal{F}$。

如第 1.3 节中所述，我们将介绍两种方法，利用基础伪随机生成器 G_{base} 构造一个能同时欺骗一致性程序的伪随机生成器。第一种方法基于 INW 生成器 [22]，的核心思想：使用具有适当参数的提取器，反复为 G_{base} 生成新的种子，从而扩展其输出。第二种方法则依靠能欺骗组合矩形的伪随机生成器来生成多个优质种子，并将它们直接代入 G_{base}。

4.1.1 使用 INW 生成器

以下结果源自 Goldreich 和 Wigderson [17] 的工作（另见 [36]）：

定理 15 ([17])**.** 对于每个 n、$k \in \mathbb{N}_+$ 和 $\varepsilon > 0$，都存在一个 (k, ε) 提取器 $\text{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n$，其种子长度为 $d = O(n - k + \log(1/\varepsilon))$，且可在 $O(n + d)$ 空间内计算。

我们按照第 1.3 节中所述的方法继续。从给定的生成器 G 开始，我们使用提取器生成一个新的种子供 G 使用，只需再用“几个”随机比特，然后将两个输出直接拼接起来。

引理 16. 设 G: $\{0,1\}^d \to (\{0,1\}^t)^m$ 是一个函数，能够同时以 m 程度愚弄宽度为 w、长度为 t 的一致性程序，误差为 ε。那么存在一个函数 G': $\{0,1\}^{d'} \to (\{0,1\}^t)^{2m}$，其 $d' = d + O(\log(1/\varepsilon))$，能够同时以 3ε 程度愚弄宽度为 w、长度为 t 的一致性程序，误差为 $2m$。此外，如果 G 是显式的，那么 G' 也是显式的。

证明。令 Ext : $\{0,1\}^d \times \{0,1\}^{d_{\text{Ext}}} \to \{0,1\}^d$ 为定理 15 中的 (k, ε) 提取器，其中 $k = d - \log(1/\varepsilon)$ 且 $d_{\text{Ext}} = O(\log(1/\varepsilon))$。我们定义

$$G'(s_G, s_{\text{Ext}}) = G(s_G)G(\text{Ext}(s_G, s_{\text{Ext}}))$$

并将 G' 输出中的第 i 个分量记为 $G'(\cdot)_i$。现在，令 P_1, ..., P_{2m} 为宽度为 w、长度为 t 的一致性程序。对于 $j \in \{1,2\}$，令

$$A_j = \{x_1 \cdots x_m \in \{0,1\}^{mt} \mid x_1, \ldots, x_m \in \{0,1\}^t \text{ and } \forall i \in [m] : P_{(j-1)m+i}(x_i) = 1\}$$

且 $\mu_j = \mu(A_j)$。注意，$\mu_1\mu_2 = \prod_{i=1}^{2m} \Pr[P_i(U_t) = 1]$。若 $\mu_1 < 2\varepsilon$，则我们立即得到 $\mu_1\mu_2 < 2\varepsilon$ 与 .....之间的距离至多为 3ε

$$\Pr[\forall i \in [2m] : P_i(G'(U_d)_i) = 1] \le \Pr[G(U_d) \in A_1] \le \mu_1 + \varepsilon < 3\varepsilon.$$

因此，假设 $\mu_1 \ge 2\varepsilon$。注意，根据关于 G 的假设，这意味着

$$\Pr[U_d = x \mid G(U_d) \in A_1] \le \frac{\Pr[U_d = x]}{\Pr[G(U_d) \in A_1]} \le \frac{2^{-d}}{\mu_1 - \varepsilon} \le \frac{2^{-d}}{\varepsilon}$$

对于 $x \in \{0,1\}^d$。特别地，这意味着，如果 Z 是一个服从 $\Pr[Z = x] = \Pr[U_d = x \mid G(U_d) \in A_1]$ 分布的随机变量，那么 $H_\infty(Z) \ge k$。因此，根据提取性质，

$$\begin{aligned}
\left|\Pr[\forall i \in [2m] : P_i(G'_i(U_d)) = 1] - \mu_1\mu_2\right| &= |\Pr[G(U_d) \in A_1 \wedge G(\text{Ext}(U_d, U_{d_{\text{Ext}}})) \in A_2] - \mu_1\mu_2| \\
&\le |\Pr[G(\text{Ext}(U_d, U_{d_{\text{Ext}}})) \in A_2 \mid G(U_d) \in A_1] - \mu_2| + \varepsilon \\
&\le |\Pr[G(U_d) \in A_2] - \mu_2| + 2\varepsilon \\
&\le 3\varepsilon.
\end{aligned}$$

□

现在，从一个能够欺骗（单个）宽度为 w、长度为 t 的一致性程序的伪随机生成器出发，并重复进行引理 16 的构造r 次，我们得到：

引理 17。令 G: $\{0,1\}^d \to \{0,1\}^t$ 为一个函数，能够以 ε-欺骗宽度为 w、长度为 t 的一致性程序。对于每个 $r > 0$，存在一个函数 G': $\{0,1\}^{d'} \to (\{0,1\}^t)^{2^r}$，其满足 $d' = d + O(r\log(1/\varepsilon))$，并且能够同时以 2^r-欺骗宽度为 w、长度为 t 的一致性程序。此外，如果 G 是显式的，那么 G'也是显式的。

4.1.2 使用伪随机生成器构造组合矩形

关于构造能够欺骗组合矩形的伪随机生成器，已有大量研究。为了获得尽可能好的参数，我们将使用两种不同的构造。第一种来自 Lu 的一项经典工作：

定理18 ([26])。对于每个 n、$m \in \mathbb{N}_+$ 和 $\varepsilon_{\text{CR}} > 0$，都存在一个显式伪随机生成器 G_{CR}: $\{0,1\}^{d_{\text{CR}}} \to (\{0,1\}^m)^n$，它能以 ε_{CR} 的误差欺骗任意 (n, m) 组合矩形，且其种子长度为

$$d_{\text{CR}} = O(m + \log n + (\log(1/\varepsilon_{\text{CR}}))^{3/2}).$$

该种子长度对 m 的依赖是最优的，但当误差参数 $\varepsilon_{\mathrm{CR}}$ 较小时，表现就不够理想。（对 n 的依赖已经满足我们的需求。）因此，我们还采用了 Gopalan 和 Yehudayoff 最近提出的一种构造，它以增加一个多对数因子为代价，改善了对 $\varepsilon_{\mathrm{CR}}$ 的依赖：

定理 19 ([18])。对于每个 n、$m \in \mathbb{N}_+$ 和 $\varepsilon_{\mathrm{CR}} > 0$，都存在一个显式伪随机生成器 $G_{\mathrm{CR}}\colon \{0,1\}^{d_{\mathrm{CR}}} \to (\{0,1\}^m)^n$，它能以 $\varepsilon_{\mathrm{CR}}$ 的误差欺骗任意 (n,m)组合矩形，且其种子长度为

$$d_{\mathrm{CR}} = O((m + \log\log n + \log(1/\varepsilon_{\mathrm{CR}}))\log(m + \log(1/\varepsilon_{\mathrm{CR}}))).$$

引理 20。设 $G\colon \{0,1\}^d \to \{0,1\}^t$ 是一个能以 ε 的误差欺骗宽度为 w、长度为 t 的一致性程序的函数。对于每个 $r \in \mathbb{N}_+$，都存在一个函数 $G'\colon \{0,1\}^{d'} \to (\{0,1\}^t)^r$，满足以下任一条件

$$d' = O(d + \log r + (\log(1/\varepsilon_{\mathrm{CR}}))^{3/2})$$

or

$$d' = O((d + \log\log r + \log(1/\varepsilon'))\log(d + \log(1/\varepsilon')))$$

它能以 r 的误差同时欺骗宽度为 w、长度为 t 的一致性程序，其中 $\varepsilon' = O(\varepsilon r)$。此外，若 G 是显式的，那么 G' 也是显式的。

证明。我们分两步构造 G'。首先，按照 G 生成 r 个字符串，并使用完全随机（相互独立）的种子。然后，通过应用定理 18 和 19 中的生成器 G_{CR}，将种子长度从 dr 降至 d'。正确性源于：同时欺骗 r 个一致性程序是欺骗组合矩形这一情形的特例。详情如下。

Consi首先，中间伪随机生成器 $G_{\mathrm{int}}\colon (\{0,1\}^d)^r \to (\{0,1\}^t)^r$ 定义为

$$G_{\mathrm{int}}(s_1, \ldots, s_r) = (G(s_1), \ldots, G(s_r)).$$

然后我们有，G_{int} 必须同时（r）欺骗任意一组宽度为 w、长度为 t 的一致性程序。这是因为 G 的副本使用独立种子，这意味着

$$\begin{aligned}
&\left|\Pr[\forall i \in [r] : P_i(G(s_i)) = 1] - \prod_{i=1}^{r} \Pr[P_i(U_t) = 1]\right| \\
&\quad = \left|\prod_{i=1}^{r} \Pr[P_i(G(s_i)) = 1] - \prod_{i=1}^{r} \Pr[P_i(U_t) = 1]\right| \\
&\quad = r \max_{i \in [r]} |\Pr[P_i(G(s_i)) = 1] - \Pr[P_i(U_t) = 1]| + O(\varepsilon^2),
\end{aligned}$$

其中右侧显然有界于 $O(r\varepsilon)$（当然假设 $\varepsilon < 1$，这是有趣的情形）。

现在，我们使用定理 18 或定理 19 中的任意一个，实例化用于 (r,d)-组合矩形的伪随机生成器 G_{CR}，并令其与 $\varepsilon_{\mathrm{CR}} = \varepsilon r$ 结合。将 G_{CR} 与 G_{int} 组合得到 $G' = G_{\mathrm{int}} \circ G_{\mathrm{CR}}\colon \{0,1\}^{d_{\mathrm{CR}}} \to (\{0,1\}^t)^r$。为说明其正确性，设 $P_1, \ldots, P_r$ 是任意宽度为 w、长度为 t 的一致性程序，并考虑组合 f_i 所得到的组合矩形 $f\colon \{0,1\}^d \to \{0,1\}$ 与 $f_i(s) = P_i(G(s))$ 组合。于是我们有

$$\Pr[\forall i \in [r] : P_i(G'(U_{d_{\mathrm{CR}}})_i) = 1] = \Pr[\forall i \in [r] : f_i(G_{\mathrm{CR}}(U_{d_{\mathrm{CR}}})_i) = 1]$$

（其中 $G'(U_{d_{\mathrm{CR}}})_i$ 和 $G_{\mathrm{CR}}(U_{d_{\mathrm{CR}}})_i$ 分别表示 $G'(U_{d_{\mathrm{CR}}})$和 $G_{\mathrm{CR}}(U_{d_{\mathrm{CR}}})$中的第 i个分量）与$\varepsilon_{\mathrm{CR}}$接近

$$\Pr[\forall i \in [r] : f_i(U_d) = 1] = \Pr[\forall i \in [r] : P_i(G(U_d)) = 1]$$

（其中，对于每个 i，U_d 都是来自同一分布的独立副本）。 □

4.2 获得实际的伪随机生成器

我们将证明，要将引理17和引理20中的伪随机生成器转换为适用于滑动窗口分支程序的伪随机生成器，只需交错相应伪随机生成器 G' 的两个独立副本。为此，对于 m， $t \in \mathbb{N}_+$以及 $(\{0,1\}^t)^m$ 中的元组 $x = (x_1, \ldots\ldots, x_m)$ 和 $y = (y_1, \ldots\ldots, y_m)$，令

$$x \parallel y = x_1 y_1 x_2 y_2 \cdots x_m y_m \in \{0,1\}^{2tm}.$$

引理21。 设 m、 $t \in \mathbb{N}_+$ 和 $\varepsilon > 0$ 是任意的。则，对于任意（独立的）随机变量$X = (X_1, \ldots, X_m)$ 和 $Y = (Y_1, \ldots, Y_m)$，其取值于 $(\{0,1\}^t)^m$，并且满足 X （分别为 Y） m- 同时 ε- 欺骗宽度为 w、长度为t 的一致性程序，则 $X \parallel Y$（即取值为 $x \parallel y$ 的随机变量，其中 x 和 y 分别从 X 和 Y中抽取） 2ε-欺骗窗口大小为 t、宽度为w、长度为$2tm$ 的滑动窗口分支程序。

证明。 设 S 是一个窗口大小为 t、宽度为w、长度为$2tm$ 的滑动窗口分支程序。我们证明，$\mathbf{Pr}[S(X \parallel Y) = 1]$ 与$\mathbf{Pr}[S(X \parallel U) = 1]$相差至多$\varepsilon$，其中 U是一个在$(\{0,1\}^t)^m$上均匀分布的随机变量。通过类似的论证，我们还有$\mathbf{Pr}[S(X \parallel U) = 1]$与$\mathbf{Pr}[S(U \parallel U) = 1]$相差至多 ε（其中 U 的两次出现表示同一随机变量的独立副本）。这便得出该引理的结论。

固定某个 $x = (x_1、\ldots\ldots、 x_m) \in \mathbf{Supp}(X)$，并令 S_x^i 为如下定义的宽度为 w、长度为 t 的一致性程序：

- S_x^i 与 S具有相同的状态。
- S_x^i 第 j 层的状态转移函数与 S第 $(k_i + t + j)$ 层的状态转移函数相同，其中 $k_i = 2(i-1)t$。
- 初始状态为 $S_{k_i}(x_i)$。（注意，由于 S 的窗口大小为 t 且$|x_i| = t$，这一定义是良好的。）
- 对于 $j < t$，状态 s 在 S_x^i 的第 j 层接受，当且仅当它在 S的第 $(k_i + t + j)$ 层接受。状态 s 在 S_x^i 的第 t 层接受，当且仅当它也在 S 的第 $(k_i + 2t)$ 层接受，并且还满足 $S_{k_i+2t}(x_{i+1}) = 1$。（如果 $i = m$，则最后这个条件显然成立。）此外， S_1 的初始状态只有在$S_0(x_1) = 1$时才接受。

注意，根据构造， S 在读取任意输入 $x \parallel y$的 y 部分时所处的状态，与 S_x^i中对应的状态相同。因此，我们观察到，对于每个 i， $S(x \parallel y) = 1$ 当且仅当$S_x^i(y_i) = 1$：

- 如果 $S(x \parallel y) = 0$，那么对于某个 $r \in [n]$，输入 $x \parallel y$ 会经过 S 第 r层中的一个拒绝状态。– 如果 r 对应于输入的 y 部分，即对于某个 i 和 $j \in [t]$， $r = k_i + t + j$，那么（根据构造） y_i 会经过 S_x^i第 j层中的同一个状态。– 反过来，如果 r 对应于输入的 x 部分，那么对于某个 i 和 $j \in [t]$， $r = k_i + j$。如果 $i > 1$，那么 $S_x^{i-1}(y_{i-1}) = 0$，因为 S_x^{i-1}最后一层的状态会拒绝；否则， $i = 1$，于是 $S_x^1(y_1) = 0$，因为 S_x^1 的初始状态是拒绝状态。

在任一情况下，我们都能找到某个 i，使得 $S_x^i(y_i) = 0$。

- 如果对于某个 i， $S_x^i(y_i) = 0$，那么存在 $j \in [t]$，使得 y_i 经过 S_x^i 的第 j 层中的一个拒绝状态。如果 S 的第 $(k_i + j)$ 层中的对应状态也为拒绝状态，那么 $S(x \,\|\, y) = 0$，因此不妨假设情况并非如此。于是，要么 $j = t$，此时 $S_{k+i+2t}(x_{i+1}) = 0$；要么 $j = 1$且 $i = 1$，此时 $S_0(x_1) = 0$。在任一情况下，均可推出 $S(x \,\|\, y) = 0$。

由上述内容并利用 X 和 Y 相互独立，可得

$$\begin{aligned}
&|\Pr[S(X \,\|\, Y) = 1] - \Pr[S(X \,\|\, U) = 1]| \\
&\quad = \left| \sum_x \Pr[X = x] \left(\Pr[S(x \,\|\, Y) = 1] - \Pr[S(x \,\|\, U) = 1] \right) \right| \\
&\quad \le \sum_x \Pr[X = x] \left| \Pr[\forall i \in [m] : S_x^i(Y_i) = 1] - \prod_{i=1}^{m} \Pr[S_x^i(U_t) = 1] \right| \\
&\quad \le \varepsilon.
\end{aligned}$$

□

现在我们汇总上述思路，以证明定理 5。

定理 *5* 的证明。 我们使用独立种子实例化两个生成器 G_{base}，将每个生成器扩展至至少 $n/2$ 比特（使用引理17和引理20中的任一个），然后通过交错排列每块 t 比特的分块将它们组合起来。具体如下。

为简单起见，我们假设 n 是 $2t$ 的倍数。对于定理中的第一项，在引理 17 中代入G_{base}和 $r = \log(n/2t)$，得到一个生成器 $G'\colon \{0,1\}^{d'} \to (\{0,1\}^t)^{n/2t}$，满足

$$d' = d_{\text{base}} + O(\log(n/t) \log(1/\varepsilon_{\text{base}}))$$

使得它能同时以 $(n/2t)$-误差欺骗宽度为 w、长度为 t 的一致性程序，其中 $\varepsilon' = \varepsilon_{\text{base}} \cdot (n/2t)^{\log 3}$。对于第二项，我们再次在引理 20 中代入 G_{base}，但现在令 $r = n/2t$。这得到一个生成器 $G'\colon \{0,1\}^{d'} \to (\{0,1\}^t)^{n/2t}$，满足以下任一条件

$$d' = O(d_{\text{base}} + \log(n/t) + (\log(1/\varepsilon_{\text{base}}))^{3/2})$$

or

$$d' = O((d_{\text{base}} + \log\log(n/t) + \log(1/\varepsilon_{\text{base}})) \log(d_{\text{base}} + \log(1/\varepsilon_{\text{base}})))$$

它对宽度为 w、长度为 t 的一致性程序同时实现 ε'-欺骗，且 $(n/2t)$，其中 $\varepsilon' = O(\varepsilon_{\text{base}} \cdot n/t)$。对于生成器 G 的实际构造：根据该断言，我们将使用两份 G'（使用独立种子），为方便起见，分别记为 G_1 和 G_2。进而，对于 $i \in \{1,2\}$， $G_i(\cdot)$ 输出中的分量记为 $G_i(\cdot)_1$，……， $G_i(\cdot)_{n/2}$。接着，G 只需将 G_1 和 G_2 的输出交错排列；也就是说，

$$G(s_1, s_2) = G_1(s_1)_1 G_2(s_2)_1 G_1(s_1)_2 G_2(s_2)_2 \cdots G_1(s_1)_{n/2t} G_2(s_2)_{n/2t}$$

对于 s_1， $s_2 \in \{0,1\}^{d'}$。因此， G 的种子长度为 $2d'$，且显然 G 是显式的。G 的正确性在两种情况下都可由引理 25 结合引理 17 或引理 20 直接推出。 □

5 滑动窗口分支程序的命中集生成器

在本节中，我们证明：

定理 8. 给定 n、w、$t \in \mathbb{N}_+$，其中 $n \le w$ 和 $\varepsilon > 0$，并设 $G_{\text{base}}\colon \{0,1\}^{d_{\text{base}}} \to \{0,1\}^t$ 是一个能够欺骗宽度为 w、长度为 t 的一致性程序的 $\varepsilon_{\text{base}}$-*PRG*。那么存在一个显式的 ε_G-命中集生成器$G\colon \{0,1\}^d \to \{0,1\}^n$，其种子长度为

$$d = O(d_{\text{base}} + \log(n/t) + \log(1/\varepsilon_{\text{base}}))$$

对于窗口大小为 t 的宽度为 w、长度为 n 的滑动窗口分支程序，其中 $\varepsilon_G = O(\varepsilon_{\text{base}} \cdot (n/t))$。

定理 8 的证明只是对定理 10 第二项证明的直接改编。为避免重复，我们仅陈述与命中集生成器情形特别相关的事实，并省略论证中与第 4 节已涵盖内容相同的部分。

我们需要放宽的第一个概念是同时欺骗。

定义 22。 设 $m, t \in \mathbb{N}_+$ 和 $\varepsilon > 0$，且 $\mathcal{F}$ 是函数类 $f\colon \{0,1\}^t \to \{0,1\}$。如果分布 $X = (X_1, \ldots, X_m)$ 在 $(\{0,1\}^t)^m$ m-同时 ε-命中 $\mathcal{F}$，则对每个 $f_1, \ldots, f_m \in \mathcal{F}$，以下条件均成立：

$$\Pr[\forall i \in [m] : f_i(U_t) = 1] \ge \varepsilon \implies \exists (x_1, \ldots, x_m) \in \operatorname{Supp}(X) : \forall i \in [m] : f_i(x_i) = 1$$

（其中，对每个 i，U_t 都是同一随机变量的一个独立副本）。与之前一样，如果 $G(U_d)$ 欺骗 $\mathcal{F}$，则称函数 $G\colon \{0,1\}^d \to (\{0,1\}^t)^m$ m-同时 ε-命中 $\mathcal{F}$。

我们将采用 Linial 等人的命中集生成器，而不是定理 19 中的伪随机生成器。如第 1.3 节所述，它的种子长度在渐近意义下是最优的。

定理 23 ([25])。对于每个 n、$m \in \mathbb{N}_+$ 和 $\varepsilon_{\text{CR}} > 0$，都存在一个显式的 ε_{CR}-命中集生成器 $G_{\text{CR}}\colon \{0,1\}^{d_{\text{CR}}} \to (\{0,1\}^m)^n$，适用于 $(n,\ m)$-组合矩形，其种子长度为

$$d_{\text{CR}} = O(m + \log\log n + \log(1/\varepsilon_{\text{CR}})).$$

现在，引理 20 的对应结论如下：

引理 24。 设 $G\colon \{0,1\}^d \to \{0,1\}^t$ 是一个能够以 ε 的误差欺骗宽度为 w、长度为 t 的一致性程序的函数。对于每个 $r \in \mathbb{N}_+$，都存在一个函数 $G'\colon \{0,1\}^{d'} \to (\{0,1\}^t)^r$，满足

$$d' = O(d + \log\log r + \log(1/\varepsilon'))$$

该函数能同时以 r 的误差命中宽度为 w、长度为 t 的一致性程序，其中 $\varepsilon' = O(\varepsilon r)$。此外，如果 G 是显式的，那么 G' 也是显式的。

证明。令 G' 和 G_{int} 与引理 20 的证明中相同，但我们使用定理 23 中的 HSG G_{CR}，而不是定理 19 中的 PRG。更确切地说，我们要求 G_{CR} 是针对 $(d,\ r)$-组合矩形的 (εr)-HSG。由于构造的相应部分完全相同，我们仍然有：G_{int} 必须同时对任意一组宽度为 w、长度为 t 的一致性程序进行 (εr)-欺骗。因此，我们只需证明：当用 G_{CR} 生成的种子替换 G_{int} 的种子时，G' 仍然正确。

同样，令 $P_1, \ldots\ldots, P_r$ 为任意宽度为 w、长度为 t 的一致性程序，并按照引理 20 的证明定义 f_i；假设 $\Pr[\forall i \in [r] : P_i(U_t) = 1] \ge 2\varepsilon r$（其中 U_t 是来自同一分布的独立副本）。根据上文所述的 G_{int} 的性质，这意味着

$$\Pr[\forall i \in [r] : f_i(U_d) = 1] \ge \varepsilon r.$$

该结论可由 G_{CR} 是一个 (εr)-命中集生成器直接推出。 □

引理 25 的对应结论如下：

引理 25. 设 m、$t \in \mathbb{N}_+$ 和 $\varepsilon > 0$ 任意。则对于取值于 $(\{0,1\}^t)^m$ 的任意（独立）随机变量$X = (X_1、\ldots\ldots、X_m)$和 $Y = (Y_1、\ldots\ldots、Y_m)$，若 X（分别为 Y）m 同时命中宽度为 w、长度为 t 的一致性程序，则 $X \parallel Y$（即取值为 $x \parallel y$ 的随机变量，其中 x 和 y 分别从 X 和 Y 中抽取）会命中宽度为 w、长度为 $2tm$ 、窗口大小为 t 的滑动窗口分支程序。

证明。从某种意义上说，证明与引理 25 的证明类似。令 S 和 U 如上述证明中所述，并假设 $\Pr[S(U \parallel U) = 1] \geq 2\varepsilon$。我们采用相同的归约（到 S_x^i），但正确性需要不同的论证，因为现在我们必须证明，对于某些 $x \in \in \mathrm{Supp}(X)$ 和 $y \in \in \mathrm{Supp}(Y)$，$S(x \parallel y) = 1$（而不只是论证分布 $S(X \parallel Y)$ 接近 $S(U \parallel U)$）。

需要注意的关键事实是，所有 S_x^i 接受这一事件（对于任意选择的 x）都包含在 S 自身接受这一事件之中。具体而言，这意味着对于任意固定的 x，只要 $\Pr[S(x \parallel U) = 1] \geq \varepsilon$，若我们按照 U 对 x 进行采样，那么

$$\Pr_x[\forall i \in [m] : S_x^i(U_t) = 1] \geq \Pr[S(x \parallel U) = 1] \geq \varepsilon,$$

这意味着存在 $y = y(x) = (y_1, \ldots\ldots, y_m) \in \mathrm{Supp}(Y)$，使得对于每个 i，都有 $S_x^i(y_i) = 1$，因此$S(x \parallel y) = 1$。

我们称满足这一条件（即 $\Pr[S(x \parallel U) = 1] \geq \varepsilon$）的 x 为良好的。根据马尔可夫不等式，

$$\Pr_{x \leftarrow U}[\Pr[S(x \parallel U) = 0] \geq 1 - \varepsilon] \leq \frac{1 - 2\varepsilon}{1 - \varepsilon} < 1 - \varepsilon,$$

这意味着，均匀随机选取的 x 为良好的概率严格大于 ε。反过来，正如上文所述，每个这样的 x 都满足：存在 $y \in \mathrm{Supp}(Y)$，使得 $S(x \parallel y) = 1$。由此可得 $\Pr[S(U \parallel Y) = 1] > \varepsilon$。

至此，我们已完成目标的一半（即证明存在 $x \in \mathrm{Supp}(X)$ 和 $y \in \mathrm{Supp}(Y)$，使得 $S(x \parallel y) = 1$）。下一步，我们再次使用马尔可夫不等式，得到

$$\Pr_{y \leftarrow Y}[\Pr[S(U \parallel y) = 0] \geq 1 - \varepsilon] < 1.$$

因此，若按照Y选择 y，则 $\Pr[S(U \parallel y) = 1] \geq \varepsilon$ 成立的概率非零。采用与上述类似的论证可知，存在 $x = x(y) \in \mathrm{Supp}(X)$，使得$S(x \parallel y) = 1$。

□

定理 8 的证明最后部分与定理 10 的证明相同。至此，定理 8 的证明完成。

6 子线性时间概率细胞自动机的应用

对于本节的结果，我们假设读者熟悉细胞自动机理论。（例如，[9] 可参阅一篇标准参考文献。）

6.1 概率细胞自动机

如引言中所述，PACA 模型定义于 [30]。为便于读者查阅，现将定义原样重述如下。

我们仅考虑有界的一维元胞自动机。

定义 26。 元胞自动机 是一个三元组 $C = (Q,\ \$\ \delta)$，其中 Q 是有限的状态集合，$\$ \notin Q$ 是边界符号，而 δ: $Q_\$ \times Q \times Q_\$ \to Q$ 是局部转移函数，其中$Q_\$ = Q \cup \{\$\}$。 δ 的定义域中的元素是 C 的各个元胞可能具有的局部配置。对于固定宽度 $n \in \mathbb{N}_+$， C 的全局配置是 Q^n 的元素。单元格 1 和 n 是 C 的边界元胞。全局转移函数 Δ: $Q^n \to Q^n$ 通过在各处同时应用 δ 获得；也就是说，如果 $s \in Q^n$ 是 C 当前的全局配置，那么

$$\Delta(s) = \delta(\$, s_1, s_2)\, \delta(s_1, s_2, s_3) \cdots \delta(s_{n-1}, s_n, \$).$$

对于 $t \in \mathbb{N}_0$， Δ^t 表示 Δ的第 t次迭代。对于初始配置 $s \in Q^n$，序列$s = \Delta^0(s)\ \Delta(s)\ \Delta^2(s) \ldots$是 C（对于 s）的演化轨迹。将 C的演化轨迹逐行书写，即得到其时空图。最后，对于一个元胞 $i \in [n]$ 和 $r \in \mathbb{N}_0$， $[i-r, i+r] \cap [n]$ 中的元胞构成 i的r-邻域 。

定义 27。 确定性 *ACA* （DACA）是一个元胞自动机 C，它具有输入字母表$\Sigma \subseteq Q$ 以及接受状态的子集 $A \subseteq Q$ 。如果存在$t \in \mathbb{N}_0$ 使得 $\Delta^t(x) \in A^n$，我们就称 C 接受 输入 $x \in \Sigma^+$，并将所有这类 x 的集合记为 $L(C)$。此外，如果对于每个 $x \in L(C) \cap \Sigma^n$，都存在 $t < T(|x|)$ 使得 $\Delta^t(x) \in A^n$，则称 C 具有（不超过） T的时间复杂度： $\mathbb{N}_+ \to \mathbb{N}_0$ 。

定义 28。 令 Σ 为一个字母表，令 Q 为一个有限状态集，且 $\Sigma \subseteq Q$。一个概率 *ACA*（PACA） C是一个元胞自动机，具有两个局部转移函数 δ_0、δ_1: $Q^3 \to Q$。在 C的每一步中，每个元胞都会 $c \in \{0,1\}$ 抛一次公平硬币，并根据 δ_c更新其状态；也就是说，如果 C的当前配置为 $s \in Q^n$，且各元胞抛硬币的结果为 $r = r_1 \cdots r_n \in \{0,1\}^n$（其中 r_i 是第 i个元胞抛硬币的结果），那么 C 的下一个配置为

$$\Delta_r(s) = \delta_{r_1}(\$, s_1, s_2)\, \delta_{r_2}(s_1, s_2, s_3) \cdots \delta_{r_n}(s_{n-1}, s_n, \$).$$

将这一过程视为定义在 Q^n上的马尔可夫链 M，我们把全局转移函数 $\Delta = \Delta_{U_n}$重新表述为一族由当前配置 s 参数化的随机变量（$\Delta(s)$）$_{s \in Q^n}$，其中 $\Delta(s)$通过从状态 s 开始，并在 M 上执行一次转移来采样（按照 U_n进行各元胞的抛硬币）。类似地，对于 $t \in \mathbb{N}_0$， $\Delta^t(s)$通过从 s 开始，并在 M上执行 t 次转移来采样。

C 对输入 $x \in \Sigma^n$ 的计算过程是 M 中的一条路径，从 x开始。如果该路径至少访问一次 A^n，则该计算过程是接受的。此外，为了能够量化 PACA 接受输入的概率，我们还要求对于每个 PACA C，都存在一个函数 T: $\mathbb{N}_+ \to \mathbb{N}_0$，使得对于任意输入 $x \in \Sigma^n$， x 的每个接受计算过程首次访问 A^n所需的步数严格小于 $T(n)$；也就是说，如果存在 $t \in \mathbb{N}_0$ 使得 $\Delta^t(x) \in A^n$，那么$\Delta^{t_1}(x) \in A^n$ 对于某个 $t_1 < T(n)$成立。（因此， x 的每个接受计算过程都有一个初始片段，其终点位于 A^n 中，且长度严格小于 $T(n)$。）如果对任意这样的 T都满足这一条件，我们就称 C 具有由 T界定的时间复杂度。

在这一限制下，我们现在可以等价地用一个比特矩阵$R \in \{0,1\}^{T(n) \times n}$来替代 C 的抛硬币过程，该矩阵的行分别为 R_0、.....、 $R_{T(n)-1}$，并且 $R_j(i)$对应于第 j步中第 i个单元格的抛硬币结果。（如果 C在第 t步接受，则第 t、.....、T行中的抛硬币结果$n(-1)$将被忽略。）我们将 R 称为 C 的随机输入。模糊在线随机性与离线随机性之间的区别，如果 C 按照 R设置其抛硬币结果时接受 x，我们记为 $C(x,\ R) = 1$；否则记为 $C(x,\ R) = 0$ 。

定义29。设 $L \subseteq \Sigma^*$ 和 $p \in [0,1)$。对于 L，一个单侧 p-错误概率 *PACA*是一个 PACA C，其时间复杂度为 T，并且对每个 $x \in \Sigma^n$，都满足以下条件：

$$x \in L \iff \Pr[C(x, U_{T(n)\times n}) = 1] \geq 1 - p \qquad \text{and} \qquad x \notin L \iff \Pr[C(x, U_{T(n)\times n}) = 1] = 0.$$

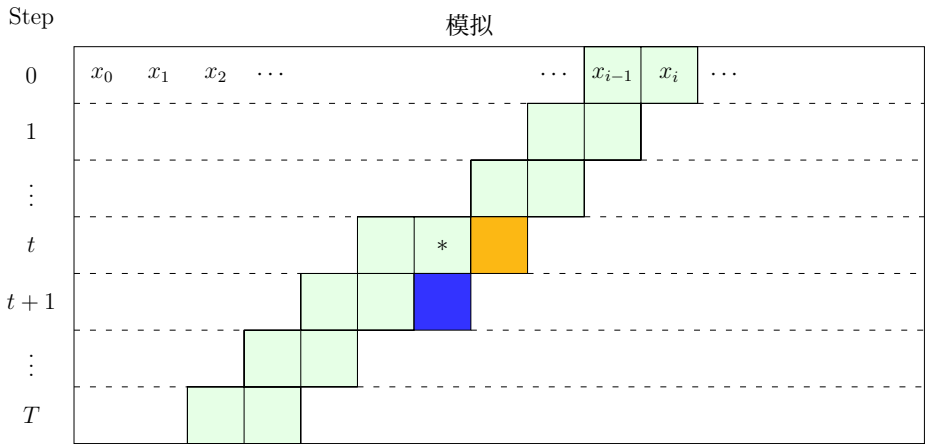


图1：通过滑动窗口算法 S 对 C 进行模拟。在图中，S 刚刚读取了输入符号 x_i，现在正在确定时间步 $t+1$ 中用星号标记的单元格状态（以蓝色显示）。浅绿色元胞是 S 在其 stateLeft 和 stateCenter 数组中维护的元胞，而橙色元胞对应于 stateRight。

如果 $p = 1/2$，那么我们就简单地称 C 为单侧错误概率 *PACA*。类似地，对于 $p < 1/2$，一个双侧p-错误概率 *PACA*是一个 PACA C，并且对每个 $x \in \Sigma^*$，都满足以下条件：

$$x \in L \iff \Pr\ [C(x, U_{T(n)\times n}) = 1] \geq 1 - p \qquad \text{and} \qquad x \notin L \iff \Pr[C(x, U_{T(n)\times n}) = 1] \leq p.$$

如果 $p = 1/3$，那么我们就称 C 为双侧错误概率细胞自动机。在这两种情况下，我们记作 $L(C) = L$，并称 C 接受 L。

6.2 使用低空间滑动窗口算法模拟 PACA

现在我们展示如何用低空间的随机滑动窗口算法模拟 PACA。只需稍作改编流式算法 [28]，即可实现这一点。（该算法实际上是为细胞自动机的另一种变体设计的，但同样的策略也适用于我们的情形。）为确保内容完整，我们将完整给出这一改编。

命题 30。 设 C 为一个状态集为 Q 且具有 $T \in \mathbb{N}_0$的（单边或双边错误）*PACA*。那么存在一个空间复杂度为 $O(T \log|Q|)$的随机化非一致滑动窗口算法 $S = S_T$，其窗口大小为 $O(T^2)$，使得

$$\Pr[S(x) = 1] = \Pr[C \text{ accepts } x \text{ in time step } T].$$

我们注意到，S 的非一致性仅用于设置 T。S 的其他所有方面都是以一致的方式实现的。

其基本思路是，为了模拟 C 在输入 x 上的行为，只需在其时空图上移动一个滑动窗口，该窗口长 T 个元胞、宽 $O(1)$ 个元胞（见图 1），并按需向元胞提供随机比特。每当读取到 x 中的一个新符号，窗口就向右移动一个元胞；若窗口移到了 C 的边界之外，则用边界符号 \$ 填充窗口相应的部分。如果 S 注意到第 T 步中的任一元胞不接受，就立即停机并拒绝；否则，它会读完整个输入，并最终接受。

算法1： 随机滑动窗口算法 S

对于 $t \leftarrow 0, \ldots, T-1$ **执行**
  stateLeft$[t] \leftarrow$\$;
  stateCenter$[t] \leftarrow$ \$;
end
$i \leftarrow 0$;
一个 **while** $i < n + T$ **do**
  如果 $i < n$ **那么**
    stateRight $\leftarrow x_i$;
  else
    stateRight $\leftarrow$ \$;
  end
  $t \leftarrow 0$;
B **当** $t < T$ **执行时**
C 从随机源中采样 $b \in \{0,1\}$;
  新状态 $\leftarrow \delta_b$(stateLeft$[t]$,stateCenter$[t]$,stateRight);
  stateLeft$[t] \leftarrow$ stateCenter$[t]$;
  stateCenter$[t] \leftarrow$stateRight;
  stateRight $\leftarrow$ 新状态;
  $t \leftarrow t + 1$;
  end
  如果 stateRight $\neq$ \$ 且 stateRight 不接受 **则**
    拒绝;
  end
  $i \leftarrow i + 1$;
end
接受;

证明。 我们证明算法1（下文称为 S）满足该主张中的性质。为简明起见，在本证明中，输入 $x = x_0 \cdots x_{n-1}$ 以及 PACAC 的元胞均从零开始编号。

我们首先证明 S 的正确性。固定一个从 $r \in \{0,1\}^{(n+T)T}$到 S 的随机输入。将 r 重新表示为矩阵 $R \in \{0,1\}^{T \times (n+T)}$ 会很方便，其中 $R(i,\ j) = r(i + jT)$；也就是说， R 的第 j 列对应于外层 **while** 循环第 j 次运行中 S 所使用的随机性，而该列的第 i 行等于内层 **while** 循环第 i 次运行中 b 的值。此外，令 $R' \in \{0,1\}^{T \times n}$ 为矩阵，其中 $R'(i,\ j) = R(i,\ i + j + 1)$。我们将看到， R' 恰好对应于 S 对 C 进行模拟时所使用的随机输入，而 R 中在 R' 中没有对应项的比特不会影响 S 的结果。

通过 $D = D_{R'}$： $\{0, \ldots, T\} \times \mathbb{Z} \to Q_{\$}$我们将 C使用来自 R' 的抛硬币时的时空图记为\$，其中 \$ 用于填充 C“边界之外”的状态；也就是说，对于 $i \in [n]$， $D(t,\ i)$ 等于在输入 x 上使用R'给出的抛硬币时， C 第 t步中第 i个单元格的状态；否则， $D(t,\ i) =$ \$。我们将证明外层循环（第 A 行）满足以下不变量：

I_1: 对于每个 $t \in \{0, \ldots, T-1\}$, stateCenter$[t] = D(t, i-t-1)$ and stateLeft$[t] = D(t, i-t-2)$

I_2: S 尚未拒绝 当且仅当对于每个 $i' < i$、 D（T， $i' - T$）要么是接受状态，要么等于 \$.

特别地， I_2 直接蕴含了 S 的正确性（如果我们还知道 S 仅依赖于 R 中在 R'里有对应条目的随机比特，而事实确实如此）。

我们通过证明内层循环（第 B 行）满足其自身的不变式 I_3 来证明 I_1 和 I_2，具体而言，stateRight $= D(t,\ i-t)$。具体来说，如果在外层循环第 i 次执行前，I_1 和 I_2 成立，并且在内层循环结束时 I_3 成立，那么 I_1 和 I_2 将按如下方式保持不变：不变式 I_1 成立，因为执行的指令确保在内层循环结束时，对每个 t，stateLeft$[t] = D(t,\ i-t-1)$ 和 stateCenter$[t] = D(t,\ i-t)$ 均成立，并且 i 在结束时递增。类似地，由于内层循环结束后 $t = T$成立，因此循环结束后 stateRight $= D(t,\ i-T)$ 成立，从而可得 I_2。

为了证明 I_3 是不变式，假设对于某个 i，I_1 成立。显然，在首次执行循环之前，stateRight $= D(0,\ i)$ 成立，因为此时若 $i < n$，则 stateRight $= x(i) = D(0,\ i)$；否则，stateRight $=$ \$ $= D(0,\ i)$。随后，在循环第 t 次执行时，若 $t+1 \le i \le n+t$，则 b 被设为 $R(t,\ i) = R'(t,\ i-t-1)$，而 stateRight 被设为

$$\begin{aligned}\delta_b(\mathsf{stateLeft}[t], \mathsf{stateCenter}[t], \mathsf{stateRight}) &= \delta_b(D(t, i-t-2), D(t, i-t-1), D(t, i-t)) \\ &= D(t+1, i-t-1).\end{aligned}$$

如果 $i < t+1$，那么 $i-t-1 < 0$，因此 stateLeft$[t] =$ stateCenter$[t] =$ \$，这意味着无论 $R(t,\ i)$ 的值是什么，stateRight 都会被设为 \$；$i > n+t$ 的情况也相同，因为此时 $i-t-1 > n-1$。因此，当 $t+1 \le i \le n+t$ 时，S 的操作仅取决于 $R(t,\ i)$，而这恰好是在 $R'(t,\ i-t-1) = R(t,\ i)$ 时的情形。由于 t 在循环末尾递增，因此可得 I_3 是一个不变量，符合预期。

S 的空间复杂度显而易见，因为它由数组 stateLeft 和 stateRight 主导，两者都包含 T 个 Q。最后，关于 S 的滑动窗口大小，我们可以根据上述不变量以及时空图 D 的性质进行论证。显然，条目 $R'(t,i)$ 只会影响 $j \in \{1,\dots,T-t\}$ 和 $k \in \mathbb{Z}$ 对应的状态 $D(t+j, i+k)$，其中 $|k| < j$（因为对于每个 t 和每个 i，$R'(t,i)$ 只会影响 $D(t+1,i)$，而 $D(t,i)$ 只会影响 D、$D(t+1,i)$ 和 $D(t+1,i+1)$）。因此，在读取 $R'(t,i)$ 后，如果 S 再读取 $2T^2$ 个随机比特，那么它的状态将独立于条目 $R'(t,i)$。这意味着 S 的窗口大小为 $O(T^2)$，符合预期。

□

6.3 使用小空间对亚线性时间概率交替细胞自动机进行去随机化

在最后这一节中，我们回顾并证明针对由亚线性时间概率交替细胞自动机接受的语言的低空间去随机化结果（即定理 10 和定理 11）。

6.3.1 单边错误概率可访问电路自动机

定理 10. 设 $\varepsilon > 0$ 为任意值，并设 $T\colon \mathbb{N}_+ \to \mathbb{N}_+$ 是一个（可构造的）函数。那么，对于任何识别其语言 $L(C)$、运行时间至多为 $T = T(n)$ 的单侧 ε-错误概率交替细胞自动机 C，存在一个针对 $L(C)$ 的确定性算法，其空间复杂度为

$$O\left(\frac{(T+\log(1/\varepsilon))\log T}{\max\{1, \log T - \log\log(n/\varepsilon)\}} + \log(1/\varepsilon) + \log n\right).$$

证明大致上是直接的：给定定理陈述中的一个 T-时间、单边 ε-错误的 PACA C，我们使用命题 30 中的构造，并结合推论 9 中的命中集生成器。由于该 PACA 可能在任意时间步 $t < T$ 接受，因此我们需要针对每个这样的时间步 t模拟命题 30 中的算法S_t；然而，根据平均论证，至少存在一个 t，使得 S_t 以至少 ε/T 的概率接受。因此，使用一个 (ε/T)-命中集生成器即可。

证明。 我们描述一个用于 $L(C)$、具有所需空间复杂度的算法 A 。令 G 为推论 6 中的伪随机生成器，它能以 ε_G-伪随机化宽度为 $2^{O(T)}$、长度为 $m=(n+T)T$ 且窗口大小为 $O(T^2)$ 的滑动窗口分支程序，其中 $\varepsilon_G=\varepsilon/T$。此外，对于 $t<T$，令 S_t 为命题 30 中的滑动窗口算法。我们的算法 A 按如下方式运行：

1. 对于每个 $t<T$，枚举每个可能的种子以 G ，并在其输出上模拟 S_t 。

2. 如果 S_t 接受，则立即接受。

3. 如果所有 S_t 都未接受，则拒绝。

注意， A 的空间复杂度由 G的种子长度主导。根据我们的参数设置，这个长度为

$$d=O\left(\frac{(T+\log(1/\varepsilon))\log T}{\max\{1,\log T-\log\log(n/\varepsilon)\}}+\log(1/\varepsilon)+\log n\right)$$

（当然大于 $O(T)$，后者是模拟 S_t每次运行所需的空间）。

为了证明 A的正确性，固定一个输入 x ，并将其输入 C。如果 $x\notin L(C)$，那么 C 接受x 的概率为零，这意味着对于每个 t，$\Pr[S_t(U_m)=1]=0$ ，因此 A 永远不会接受。否则，如果$x\in L(C)$，根据平均论证，存在 t 使得 $\Pr[S_t(U_m)=1]\geq\varepsilon/T=\varepsilon_G$。由于 G 是一个 ε_G-命中集生成器，因此存在一个种子 $s\in\{0,1\}^d$ ，使得 $S_t(G(s))$接受，所以 A 也接受 x。 □

6.3.2 双边错误概率自动元胞自动机

对于双边错误概率交替细胞自动机，我们理想情况下希望在定理 10 的证明中直接用伪随机生成器“替换”命中集生成器。毕竟，伪随机生成器基本上能让我们估计每个时间步 t 概率交替细胞自动机接受的概率。不幸的是，即使我们确实有办法精确地 确定这些概率，这种做法仍然行不通。

我们通过一个例子来说明原因。具体而言，设输入字母表为$\Sigma=\{0,1\}$。考虑按如下方式运行的 PACA C_1 和 C_2 ：除第一个单元格外，所有单元格都无条件处于接受状态。在前 8 步中，最左侧的单元格收集随机比特，组成随机字符串 $r\in\{0,1\}^8$。此后，在接下来的步骤中，该单元格按如下方式运行：

- 在 C_1中，当且仅当 r 的前两位比特均为零（即$r_1=r_2=0$）时，该单元格转为接受状态。该单元格恰好保持接受状态四步，然后变为非接受状态并保持该状态。

- 在 C_2 中，在随后四个步 $j\in[4]$ 的每一步，单元格在第 j 步转为接受状态，当且仅当 $r_{2j-1}=r_{2j}=0$。此后，单元格进入非接受状态并保持该状态。

很容易看出， C_1 和 C_2 都是双边错误概率交替细胞自动机，但它们接受的语言完全不同： C_1 接受任意输入 $w\in\{0,1\}^*$ 的概率为 $1/4$，这意味着 $L(C_1)=\varnothing$；反过来， C_2 接受任意输入的概率为 $1-(3/4)^4>2/3$，因此 $L(C_2)=\{0,1\}^*$。然而，在任意固定时间步 t （单独考虑）上，PACA 接受的概率在这两种情况下都相同 C_i。

这意味着需要采用一种新策略。一种可能的解决方案是调整命题 30 中的算法 S ，使其检查 PACA 的多个步是否接受（例如，为每个步 i 维护一个布尔变量 accept$_i$ ，初始设为 true；若在步 i 中看到任何非接受状态，则将其设为 false）。然而，目前还不清楚如何在不放弃滑动窗口性质的情况下做到这一点。尽管如此，如果我们不再只考虑单个时间步 t ，而是验证 PACA 在

属于某个子集 $t \subseteq [T]$ 的每个步都接受（该子集以非一致的方式提供给 S），那么就可以采用这一策略（而不放弃滑动窗口性质）。这确实是一种直接的调整，而且不难看出，所得算法 S_t 满足

$$\Pr[S_t(x) = 1] = \Pr[\forall i \in t : \text{on input } x,\ C \text{ is accepting in time step } i]$$

其中，第二个概率是以 C 的抛硬币结果为条件计算的。（从这个角度来看，当 $|t| = 1$ 时，原算法适用于这一特殊情况。）接下来，我们可以运用容斥原理，将 C 接受的概率改写为对 S_t 接受的概率求和（对 t 的每一种可能的非空选择）。使用参数适当的伪随机生成器后，我们便可以估计该和式中的每一项，并根据这个估计值高于还是低于 $1/2$，判断 C 是否接受。

定理 11。设 $T\colon\ \mathbb{N}_+ \to \mathbb{N}_+$ 是一个函数，$\varepsilon > 0$ 任意。对于任意识别语言 $L(C)$ 的双侧($1/2 - \varepsilon$)错误 *PACA* C，若其运行时间至多为 $T = T(n)$，则存在一个用于 $L(C)$ 的确定性算法，其空间复杂度为 $O((d' + \log n)\log(d' + \log n))$，其中$d' = O((T + \log(1/\varepsilon)) \log T)$。

证明。我们构造一个具有所需空间复杂度的算法 A for $L(C)$。令 G 为推论 6 的第 2 项中的伪随机生成器，它能以 ε_G-欺骗宽度为 $2^{O(T)}$、长度为 $m = (n + T)T$、窗口大小为 $O(T^2)$ 的滑动窗口分支程序，其中 $\varepsilon_G = \varepsilon/2^T$。（除 ε_G 外，这些参数与定理 10 的证明中的相同；此处额外因子在 T 中变得指数级小。）对于 $\varnothing \neq t \subseteq [T]$，令 S_t 为此前所述命题 30 中算法的推广。算法 A 按如下方式进行：

1. 对每个 $\varnothing \neq t \subseteq [T]$，枚举 G 的所有可能种子，并在其输出上模拟 S_t，从而得到一个估计值 $\eta_t = \Pr[S_t(G(U_d)) = 1]$，该估计值与 $\Pr[S_t(U_m) = 1]$相差不超过 ε_G。
2. 计算

$$\eta = \sum_{\substack{t \subseteq [T] \\ |t|=1}} \eta_t - \sum_{\substack{t \subseteq [T] \\ |t|=2}} \eta_t + \cdots + (-1)^{T+1} \eta_{[T]}$$

并在 $\eta > 1/2$ 时接受；否则拒绝。

根据我们的参数设置，G 的种子长度为

$$d = O((d' + \log n)\log(d' + \log n)),$$

其中

$$d' = O((T + \log(1/\varepsilon)) \log T).$$

与定理 10 的证明一样，A 的空间复杂度由 d 主导。

对于正确性，令 $Z_t(R)$ 表示以下事件：C 在使用来自 $R \in \{0,1\}^{T \times n}$ 的抛硬币结果时，在 $\varnothing \neq t \subseteq [T]$ 的每一步中都接受 x。注意，根据容斥原理，我们有

$$\begin{aligned}\Pr[C(x, U_{T\times n}) = 1] &= \Pr[\exists t \in [T] : Z_{\{t\}}(U_{T\times n})] \\ &= \sum_{\substack{t \subseteq [T] \\ |t|=1}} \Pr[Z_t(U_{T\times n})] - \sum_{\substack{t \subseteq [T] \\ |t|=2}} \Pr[Z_t(U_{T\times n})] + \cdots + (-1)^{T+1} \Pr[Z_{[T]}(U_{T\times n})].\end{aligned}$$

由于右侧总共严格少于 2^T 项，且将$U_{T\times n}$ 替换为$G(U_d)$会使每一项产生至多 ε_G 的加性偏差，因此可得

$$|\eta - \Pr[C(x, U_{T\times n} = 1)]| \leq \varepsilon_G 2^T = \varepsilon. \qquad \square$$

致谢。 我谨向 Thomas Worsch 致谢，感谢他提供的有益讨论和反馈。我也感谢 Ted Pyne 指出了一些伪随机生成器的构造方法，这些方法略微改进了结果。

参考文献

[1] Roy Armoni。《空间受限计算的去随机化》。载于：计算机科学中的随机化与近似技术：第二届国际研讨会 *RAN- DOM'98*，西班牙巴塞罗那，*1998*年*10*月*8*日至*10*日，论文集。Michael Luby、José D. P. Rolim 和 Maria J. Serna 编。第1518卷。计算机科学讲义。Springer，1998年，第47–59页。doi: 10.1007/3-540-49543-6_5.[2] Sanjeev Arora 和 Boaz Barak。《计算复杂性：现代方法》。剑桥：剑桥大学出版社，2009年。[3] Pablo Arrighi、Nicolas Schabanel 和 Guillaume Theyssier。《随机细胞自动机：相关性、可判定性与模拟》。载于：*Fundam. Informaticae* 126.2-3（2013年），第121–156页。doi: 10.3233/FI-2013-875。[4] Brian Babcock、Shivnath Babu、Mayur Datar、Rajeev Motwani 和 Jennifer Widom。《数据流系统中的模型与问题》。载于：第二十一届 *ACM SIGACT-SIGMOD-SIGART* 数据库系统原理研讨会论文集，*2002*年*6*月*3*日至*5*日，美国威斯康星州麦迪逊。Lucian Popa、Serge Abiteboul 和 Phokion G. Kolaitis 编。ACM，2002年，第1–16页。doi: 10.1145/543613.543615。[5] Andrej Bogdanov、William M. Hoza、Gautam Prakriya 和 Edward Pyne。《正则分支程序的命中集》。载于：第*37*届计算复杂性会议（*CCC 2022*），*2022*年*7*月*20*日至*23*日，美国宾夕法尼亚州费城。Shachar Lovett 编。第234卷。LIPIcs。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2022年，3:1–3:22。doi: 10.4230/LIPIcs.CCC.2022.3。[6] Vladimir Braverman。《滑动窗口算法》。载于：算法百科全书。2016年，第2006–2011页。doi: 10.1007/978-1-4939-2864-4_797.[7] Graham Cormode。《连续分布式监控模型》。载于：*SIGMOD Rec.* 42.1（2013年），第5–14页。doi: 10.1145/2481528.2481530。[8] Mayur Datar、Aristides Gionis、Piotr Indyk 和 Rajeev Motwani。《维护滑动窗口上的数据流统计信息》。载于：*SIAM J. Comput.* 31.6（2002年），第1794–1813页。doi: 10.1137/S0097539701398363。[9] Marianne Delorme 和 Jacques Mazoyer 编。《细胞自动机：并行模型》。数学及其应用丛书第460卷。多德雷赫特：Springer 荷兰，1999年。doi: 10.1007/978-94-015-9153-9。[10] Michael A. Forbes 和 Zander Kelley。《适用于任意顺序的读一次分支程序伪随机生成器》。载于：第*59*届 *IEEE* 计算机科学基础年会（*FOCS 2018*），法国巴黎，*2018*年*10*月*7*日至*9*日。MikkelThorup 编。IEEE 计算机学会，2018年，第946–955页。doi: 10.1109/FOCS.2018.00093。[11] Moses Ganardi、Danny Hucke 和 Markus Lohrey。《严格正确的滑动窗口算法去随机化*》。载于：*Theory Comput. Syst.* 65.3（2021年），第1–18页。doi: 10.1007/s00224-020-10000-1。

[12] Moses Ganardi、Danny Hucke 和 Markus Lohrey。《正则语言的随机滑动窗口算法》。载于：第*45*届国际自动机、语言与编程研讨会（*ICALP 2018*），*2018*年*7*月*9*日至*13*日，捷克共和国布拉格。Ioannis Chatzi-giannakis、Christos Kaklamanis、Dániel Marx 和 Donald Sannella 编。第107卷。LIPIcs。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2018年，127:1–127:13。doi: 10.4230/LIPIcs.ICALP.2018.127。[13] Moses Ganardi、Danny Hucke 和 Markus Lohrey。《正则语言的滑动窗口算法》。载于：语言与自动机理论及应用——第*12*届国际会议（*LATA 2018*），以色列拉马特甘，*2018*年*4*月*9*日至*11*日，论文集。Shmuel Tomi Klein、Carlos Martín-Vide 和 Dana Shapira 编。第10792卷。计算机科学讲义。Springer，2018年，第26–35页。doi: 10.1007/978-3-319-77313-1_2.[14] Moses Ganardi、Artur Jez 和 Markus Lohrey。《上下文无关语言上的滑动窗口》。载于：第*43*届国际计算机科学数学基础研讨会（*MFCS 2018*），*2018*年*8*月*27*日至*31*日，英国利物浦。Igor Potapov、Paul G. Spirakis 和 James Worrell 编。第 117 卷。LIPIcs。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2018年，15:1–15:15。doi: 10.4230/LIPIcs.MFCS.2018.15。[15] Aurélien Garivier 和 Eric Moulines。《关于切换多臂老虎机问题的上置信界策略》。载于：算法学习理论——第*22*届国际会议（*ALT 2011*），芬兰埃斯波，*2011*年*10*月*5*日至*7*日，论文集。Jyrki Kivinen、Csaba Szepesvári、Esko Ukkonen 和 Thomas Zeugmann 编。第6925卷。计算机科学讲义。Springer，2011年，第174–188页。doi: 10.1007/978-3-642-24412-4_16.[16] Oded Goldreich。《计算复杂性：概念性视角》。剑桥：剑桥大学出版社，2008年。[17] Oded Goldreich 和 Avi Wigderson。《具有随机性质的小函数族：哈希中的质量—规模权衡》。载于：*Random Struct. Algorithms* 11.4（1997年），第315–343页。doi: 10.1002/(SICI)1098-2418(199712)11:4\<315::AID-RSA3\>3.0.CO;2-1。[18] Parikshit Gopalan 和 Amir Yehudayoff。《通过初等对称多项式不等式研究有限独立性下的集中性》。载于：*Theory Comput.* 16（2020年），第1–29页。doi: 10.4086/toc.2020.v016a017。[19] William M. Hoza、EdwardPyne 和 Salil P. Vadhan。《无界宽度置换分支程序的伪随机生成器》。载于：第*12*届理论计算机科学创新会议（*ITCS 2021*），*2021*年*1*月*6*日至*8*日，线上会议。James R. Lee 编。第185卷。LIPIcs。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2021年，7:1–7:20。doi: 10.4230/LIPIcs.ITCS.2021.7。[20] William M. Hoza 和 David Zuckerman。《低成功概率RL的简单最优命中集》。载于：*SIAM J. Comput.* 49.4（2020年），第811–820页。doi: 10.1137/19M1268707。[21] Oscar H. Ibarra、Michael A. Palis 和 Sam M. Kim。《细胞自动机快速并行识别语言》。载于：*Theor. Comput. Sci.* 41（1985年），第231–246页。doi: 10.1016/0304-3975(85)90073-8。[22] Russell Impagliazzo、Noam Nisan 和 Avi Wigderson。《网络算法的伪随机性》。载于：第二十六届*ACM* 计算理论年度研讨会论文集，*1994*年*5*月*23*日至*25*日，加拿大魁北克省蒙特利尔。Frank Thomson Leighton 和 Michael T. Goodrich 编。ACM，1994年，第356–364页。doi: 10.1145/195058.195190。[23] Daniel M. Kane、Jelani Nelson 和 David P. Woodruff。《重新审视数据流中的范数估计》。载于：*CoRR* abs/0811.3648（2008年）。arXiv: 0811.3648。

[24] Sam Kim 和 Robert McCloskey. “常数时间元胞自动机计算的刻画”.载于: *Phys. D* 45.1-3(1990),第404–419页. doi: 10.1016/0167-2789(90)90198-X.[25] Nathan Linial、Michael Luby、Michael E. Saks 和 David Zuckerman. “为高维组合矩形高效构造小型命中集”. 载于: *Comb.*17.2 (1997年), 第 215–234 页. doi: 10.1007/BF01200907.[26] Chi-Jen Lu. “组合矩形的改进伪随机生成器”.载于:*Comb.*22.3 (2002), 第 417–434 页. doi: 10.1007/s004930200021.[27] Jean Mairesse 和 Irène Marcovici. “关于概率细胞自动机”. 载于: *Theor.Comput. Sci.* 559 (2014), 第 42–72 页. doi: 10.1016/j.tcs.2014.09.009.[28] Augusto Modanese. “亚线性时间收缩元胞自动机的下界与困难性放大”. 载于: 计算机科学：理论与应用——第*16*届俄罗斯国际计算机科学研讨会，*CSR 2021*，俄罗斯索契，*2021*年*6*月*28*日至*7*月*2*日，论文集. Rahul Santhanam 和 Daniil Musatov 编. 第12730卷. 计算机科学讲义. Springer, 2021年, 第296–320页. doi: 10.1007/978-3-030-79416-3_18.[29] Augusto Modanese. “一维元胞自动机的亚线性时间语言识别与判定”. 载于: *Int. J. Found. Comput. Sci.*32.6 (2021), 第 713–731 页. doi: 10.1142/S0129054121420053.[30] Augusto Modanese. “亚线性时间概率细胞自动机”. 载于: 第*40*届计算机科学理论方面国际研讨会，*STACS 2023*，*2023*年*3*月*7*日至*9*日，德国汉堡. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023年. 即将出版.[31] Noam Nisan. “空间受限计算的伪随机生成器”. 载于:*Comb.* 12.4 (1992), 第 449–461 页. doi: 10.1007/BF01305237.[32] Maciej Pacut、Mahmoud Parham、Joel Rybicki、Stefan Schmid、Jukka Suomela 和 Aleksandr Tereshchenko. “在线算法中的局部性”. 载于: *CoRR* abs/2102.09413 (2021年). arXiv: 2102.09413.[33] Michael O. Rabin. “概率自动机”. 载于: *Inf. Control.* 6.3 (1963), 第 230–245 页. doi: 10.1016/S0019-9958(63)90290-0.[34] Michael E. Saks 和 Shiyu Zhou. “BP_H SPACE(S) $\subseteq$ DSPACE($S^{3/2}$)”. 载于: *J. Comput. Syst. Sci.*58.2 (1999), 第 376–403 页. doi: 10.1006/jcss.1998.1616.[35] Rudolph Sommerhalder 和 S. Christian van Westrhenen. “细胞自动机在常数时间内的并行语言识别”. 载于: *Acta Inf.* 19 (1983), 第 397–407 页. doi: 10.1007/BF00290736.[36] Salil P. Vadhan. “伪随机性”. 载于: *Found. Trends Theor. Comput. Sci.* 7.1-3 (2012), 第 1–336 页. doi: 10.1561/0400000010.