arXiv:2603.08937v1 [cs.DS] 9 Mar 2026

随机顺序数据流中的单位区间选择

Cezar-Mihail Alexandru* Adithya Diddapur‡ MagnúsM. Halldórsson†
Christian Konrad‡ Kheeran K. Naidu§

摘要

我们研究单遍随机顺序流模型中的单位区间选择问题。在这种模型中，算法会逐个接收一系列位于直线上的 n 单位长度区间，这些区间以均匀随机顺序到达；目标是使用与最优解大小呈线性关系的空间，输出一个最大的不相交区间集合（或其近似解）。以往的研究只考虑对抗性排序数据流，并证明在这些空间限制下，此类数据流中的近似因子可达到（$2/3$）；这是最优的，因为若要达到更好的近似因子，就需要 $\Omega(n)$ 空间 [Emek 等人，TALG'16] 。

在这项工作中，我们证明，如果输入流的顺序是均匀随机的，则可以达到更优的期望近似因子，其中期望是对流顺序取的。具体而言，我们给出一种单遍流式算法，其期望近似因子为 0.7401，使用空间 $O(|OPT|)$，其中 OPT 表示一个最优解。我们还证明，期望近似因子优于 $8/9$ 的随机顺序算法需要 $\Omega(n)$ 空间；此外，以高于 $2/3$ 的概率计算出优于 $2/3$ 近似解的算法也需要 $\Omega(n)$ 空间。

在技术层面上，我们针对受限域 $[0, \Delta)$ 设计了一种算法，其中 Δ 为某个常数；并使用标准技术获得适用于非受限定义域的算法。对于受限域 $[0, \Delta)$，我们运行 $O(\Delta)$ 个算法的递归实例，每个实例针对最优解中的某个特定区间最先到达的情形。我们证明了该算法具有一个有趣的性质：当输入流仅由一组独立区间组成时，其表现最差。接下来只需分析算法在这些简单实例上的表现。

我们的下界通过通信复杂度论证得出，其思路与 [Chakrabarti 等人发表于 TheoryComput.2016] 的鲁棒通信下界类似。

*sorin_cezar@yahoo.com，独立研究者，英国布里斯托尔
†mmh@ru.is，ICE-TCS 与雷克雅未克大学计算机科学系，冰岛。部分研究经费由冰岛研究基金项目 2511609 资助。

‡{adi.diddapur,christian.konrad}@bristol.ac.uk，布里斯托大学计算机科学学院，英国布里斯托尔。
§kheeran.naidu@gmail.com，Qworky Research，英国伦敦

1 引言

在单遍流计算模型中，算法接收由输入项组成的数据流，这些输入项预先以任意顺序到达。算法的任务是在保持内存大小相对于输入流长度为次线性的同时，计算出一个解。

我们研究流模型中的单位区间选择问题。在该问题中，输入流由一系列 n 闭单位长度区间 $\mathcal{S}$ 组成，这些区间以任意顺序呈现给算法，目标是计算一个基数最大的 $\mathcal{I} \subseteq \mathcal{S}$ 两两不相交子集。

Emek 等人 [12] 率先研究了单遍流模型中的单位区间选择问题。他们提出了一种 $2/3$近似算法，使用 $O(|OPT|)$字[1]的空间，其中 OPT表示最优解；他们还证明，对于近似保证哪怕略有提升的算法，至少需要 $\Omega(n)$ 位空间，这使 [7]他们的算法在空间上达到了最优。随后，Cabello 和 Pérez-Lantero [7]给出了一个保证相同但更简单的算法。我们强调，这些结果适用于输入流按对抗性／任意顺序排列的情形。

鉴于对抗性顺序设定已得到充分研究，本文探讨：如果输入流按均匀随机顺序排列，是否可能设计出更好的算法。事实上，随机顺序流已受到广泛关注；对于许多问题，人们已知可以取得超越针对对抗性顺序所得下界的改进，例如分位数估计[13], 最大匹配 [15, 5, 2], 有界度性质测试 [17], 以及边到达集合覆盖 [16]。随机顺序流的下界研究由 Chakrabarti 等人 [9],开创，后续工作 [8] 进一步发展了这些思路。此后， [8] 中的思路已应用于各种随机顺序流下界证明，包括 [2] 和 [3]。随机顺序设定假设实例为最坏情况，但到达顺序是随机的；相较之下，任意顺序设定同时假设实例和顺序均为最坏情况，因此前者不那么悲观，也更符合从业者的实际需求。

1.1 我们的结果

作为主要结果，我们证明，在考虑均匀随机到达顺序时，可以突破对抗顺序流中的 $2/3$-近似屏障。

定理 1. 对于随机顺序流上的单位区间选择，存在一种确定性的单遍 $O(|OPT|)$ 字空间流式算法，其期望近似因子为0.7401，其中期望是对数据流的随机顺序取的。

随后，我们通过以下限制性结果补充我们的算法：

定理 2. 设 $\epsilon > 0$ 是任意小的常数。那么：

1. 对于随机顺序单位区间选择，任何期望近似因子为 $8/9 + \epsilon$ 的随机化单遍流算法都必须使用$\Omega(n)$ 位空间。*2.* 对于任意 $\delta > 0$，任何近似因子为 $2/3 + \delta$、且对任意输入实例以至少 $2/3 + \epsilon$ 的概率成功的随机化单遍流算法，都必须使用$\Omega(n)$位空间。

式 *(1)* 中的期望和式 *(2)* 中的概率，均针对数据流的流顺序和算法的随机选择。

我们注意到，在前一个定理中，(1) 可由 (2) 直接推出；然而，(1) 中陈述的结果更适合与我们的算法结果进行比较。此外，(2) 中的结果解释了为什么我们的算法只能突破对抗顺序流中成立的 $2/3$-近似障碍（在期望意义下），而无法以高常数概率突破这一障碍。

[1]一个字是存储单个区间所需的空间量。

我们的算法结果与下界结果共同表明，可实现的最优近似比位于区间 $[0.7401, 0.8]$ 内；我们将进一步缩小这一差距留作一个重要的开放问题。

1.2 技术

现在我们将讨论我们的算法和下界背后的主要思想 d.

该算法。 我们采用一种成熟的技术：给定一个针对单位长度区间的单遍流式算法，这些区间都包含在受限域 $[0, \Delta)$ 中，且对于某个常数整数 Δ，该算法的近似因子为 α；该技术可以生成一个适用于非受限定义域的算法，其空间仅增加一个常数因子，而近似因子则增加 $(\Delta - 1)/\Delta$ 倍 [11]。因此，我们只需为受限域 $[0, \Delta)$ 设计算法，其中 Δ 是一个足够大的常数，以尽可能减小因子 $(\Delta - 1)/\Delta$ 对最终近似因子的影响。

为此，作为我们算法的关键构件，我们采用以下策略：维护数据流中迄今为止观察到的最左侧区间 I_L，并将位于 I_L 右侧的区间子流输入我们算法的一个递归实例。这个构件随后会返回区间 I_L，以及递归调用得到的解。为了说明这种方法的优势，记 OPT 为一个任意但固定的最优解。此外，假设最左侧区间 $opt_1 \in OPT$ 先于 OPT 中的其他所有区间到达。在这种情况下，算法要么会选取 opt_1 作为区间 I_L，要么选取一个更靠左的区间作为 opt_1；而输入算法递归调用的子流包含 $OPT \setminus opt_1$，即一个只少一个元素的独立集。换言之，迄今为止算法的行为是最优的：它选取了一个区间 I_L，并创建了一个包含大小为 $|OPT| - 1$ 的独立集的子实例，其解可以与所选区间 I_L 结合。

遗憾的是， opt_1 在 OPT 的区间中最先出现这一事件的概率只有 $1/|OPT|$（按数据流的随机顺序）。然而，我们的目标是让这种方法奏效，即使 OPT中的任何其他区间先于 OPT中的其他区间到达也是如此。首先，根据对称性可知，当 OPT 中最右侧的区间先到达时，上述方法同样有效。因此，现假设对于某个 $1 < i < |OPT|$，第 i个（从左向右计数）最优区间 $opt_i \in OPT$先于其他最优区间到达。此外，设 ℓ 为严格小于 opt_i左边界的最大整数，设r 为严格大于 opt_i右边界的最小整数。然后，我们按如下方式构造两个解，并选取其中较大的一个。对于解 1，我们合并算法两次独立运行的输出，一次运行于定义域 $[0, \ell)$，另一次运行于定义域 $[\ell, \Delta)$。虽然这两次运行会忽略与点 ℓ 相交的区间，因此可能漏掉与该点相交的关键最优区间，但现在我们可以利用这样一个事实：区间opt_i 是定义域 $[\ell, \Delta)$ 中最靠左的最优区间。因此，如上所述，在将区间 opt_i 选作最左侧区间（或将 $[\ell, \Delta$中一个位置比 opt_i）更靠左的区间）纳入最终解时，运行于 $[\ell, \Delta)$ 上的算法不会出错。对于解 2，我们合并算法分别在定义域 $[0, r)$ 和 $[r, \Delta)$ 上的两次运行的输出，并利用这样一个事实： opt_i 是定义域 $[0, r)$ 中最靠右的最优区间。

最后，由于我们的算法无法知道 OPT 的哪个区间会先到达，因此我们针对每个整数 $1 \le i \le |OPT|$ 都运行上述策略。我们注意到，这仍然得到一个空间复杂度仅为常数的算法，因为定义域 $[0, \Delta)$ 的大小是常数，这进一步意味着该定义域中的最优解大小也仅为常数 $|OPT| \le \Delta - 1$。

在我们的分析中，我们证明了单调性：向输入流中添加区间绝不会降低算法产生的输出规模。换言之，我们算法的最坏情况性能出现在算法运行于一个独立

集上的情况下——当然，只需报告这些区间这一特殊情形即可得到最优解。因此，我们可以专注于分析算法在这一特殊情形下的性能。由于算法进行了多次递归调用，我们得到了所获解规模的递归公式。我们注意到，随着 Δ 增大，我们算法的近似因子 α 会降低。不过请注意，我们仍需应用上述技术，将受限域 $[0, \Delta$ 转换为非受限域，这会使近似因子进一步损失一个 $(\Delta - 1)/\Delta$ 的因子。通过计算机程序，我们看到，对于 $\Delta = 5000$，最终算法的近似因子至少为 0.7401，这一结果可以证明非常接近可能的最佳选择。 Δ 的这一具体选择将在附录 A 中进一步讨论

下界。 我们的下界是通过归约到单向两方通信复杂度设定下的 INDEX$_t$ 问题得到的：我们将 [12] 和 [7] 中已用于对抗序下界的团构件，与 Cormode等人 [8]提出的随机序下界思路结合起来。

在 INDEX$_t$ 问题中，有两位参与者，分别记为 Alice 和 Bob。Alice 持有一个由 t 个均匀独立随机比特组成的向量$X \in \{0,1\}^t$，Bob 则获得一个均匀随机索引 $A \in [t]$。Alice 向 Bob 发送一条消息，Bob 输出比特 $X[A]$。目标是构造一种通信协议，使其传输的消息大小尽可能小。已知，INDEX$_t$ 需要一条大小为 $\Omega(t)$ 的消息。由于流式算法可以用作单向两方通信协议（Alice 在自己那部分输入上运行流式算法，并将内存状态发送给 Bob；随后 Bob 在自己那部分输入上继续执行该算法），消息大小的下界可以转化为流式算法所用空间的下界。

INDEX$_t$ 问题可以如下归约为单位区间选择。Alice 利用$X \in \{0,1\}^t$中的比特，构造一个由 t 个相互重叠区间组成的团，也就是说，这是一组区间栈，其中对每个 $i > 1$，第 i个区间都比第（$i-1$）个区间稍微靠右。比特 $X[i]$ 编码在这一构造中：根据 $X[i]$的值，第 i个区间会发生极其微小的移动，但这并不影响构造背后的逻辑。关键的是，对于任意j，一个输出团区间栈中第 j个区间的算法，必须知道 $X[j]$的值。然后，Bob 根据索引 A构造两个翼区间，它们将对应于 $X[A]$的区间夹在中间，但与该区间相互独立。其关键性质是，除对应于 $X[A]$的区间外，每个团区间都必须恰好与这两个翼区间中的一个相交；换言之，唯一大小为 3 的独立集由这两个翼区间和区间栈中的第 A个区间组成。这又意味着，在这一构造上，任何计算出优于 $2/3$近似的算法都必须报告对应于 $X[A]$的区间，而该区间的位置可以用来恢复比特 $X[A]$。

将这一构造引入随机顺序设置后，若要使所得构造具有难解性，两个翼区间必须出现在第 A个团区间之后；否则，翼区间的位置会暴露索引 A，算法便可以在第 A个团区间到达时直接收集它，从而获得区间选择问题的最优解，并解决底层的 INDEX$_t$ 实例。不过，我们注意到，在均匀随机到达顺序下，两个翼区间确实以概率 $1/3$出现在第 A个团区间之后，并证明这种情况下问题仍然难以求解。我们得到的期望近似因子下界 $8/9$可由此解释：以概率 $1/3$，算法只能获得大小为 2 的解；以概率 $2/3$，算法可以获得大小为 3 的最优解，因此期望解规模为 $1/3 \cdot 2 + 2/3 \cdot 3 = 8/3$，即近似因子为 $(8/3)/3 = 8/9$。

在技术层面上，我们的方法是通过使用共享公共随机性进行归约，其思路源自 [8] 中的论证。Alice 和 Bob 对困难实例中所使用区间的共享随机排列进行采样。Alice 根据 X 中的比特构造每个先于翼区间到达的团区间，然后 Bob 通过

A，并通过采样公共随机变量构造其余的团区间。这样得到的构造会在与 $X[A]$ 对应的区间先于任一翼区间到达时，恰好反映输入到 INDEX$_t$ 中的比特 $X[A]$ ，而这一事件以 $1/3$ 的概率发生，正如所需，从而得到我们的下界结果。

1.3 其他相关工作

除了单位长度区间，也有人研究任意长度区间。Emek 等人 [12] 证明，在对抗顺序流中，使用空间 $O(|OPT|)$ 可以为任意长度区间实现 $1/2$-近似，并且还证明，具有更优近似因子的算法必须使用 $\Omega(n)$ 的空间。Cabello 和 Pérez-Lantero [7]针对这一问题给出了更简单的算法。最近，Dark等人 [11] 研究了加权区间流以及允许删除的数据流上的算法。另一类研究则考虑了估计最大不相交区间集的大小 [7, 4]。除了区间，其他几何对象，例如正方形和矩形 [10], ，以及线段和 2-区间 [6], ，也得到了研究。

区间选择也在流式滑动窗口模型中得到了研究，其目标是维护最近的 L 个区间上的解。Alexandru 和 Konrad[1]证明，与单遍流式设定相比，区间选择在滑动窗口模型中更难；对于单位长度区间和任意长度区间，使用空间$O(|OPT|)$ 的算法无法达到流式设定下相同的近似保证。

1.4 概述

我们在预备知识部分（第 2 节）中给出定义，并陈述我们用于下界结果的 INDEX$_t$ 通信问题的难度。随后，在第 3 节中，我们提出并分析我们的算法；在第 4 节中，我们给出下界结果。最后，我们在第 5 节中进行总结。

我们在附录 A 中进一步讨论所达到的精确近似因子。

2 预备知识

我们使用 n 表示输入流的长度， $[k]$ 表示集合 $\{0, \ldots, k-1\}$.

2.1 区间选择

在本文中，我们考虑闭区间 $I = [a, b]$，其中 $a < b$。我们主要关注单位长度的区间，即当 $b = a + 1$ 时。我们称 a 是 I 的左端点，而 b 是 I 的右端点。如果区间 $I = [a_I, b_I]$ 的左端点严格小于区间 $J = [a_J, b_J]$ 的左端点，即 $a_I < a_J$ 成立，我们称 $I = [a_I, b_I]$ 位于 $J = [a_J, b_J]$左侧更远处。类似地，如果 I 的右端点严格大于 J 的右端点，即 $b_I > b_J$ 成立，我们称 I 位于 J 右侧更远处。

如果 $I \cap J = \varnothing$，则两个区间 I， J 是相互独立的；如果一组区间 $\mathcal{I}$ 中的区间两两独立，则称其为独立集。单位区间选择问题要求找到大小最大的区间独立集，并且我们还假设所有区间的长度均为 1。

我们用 OPT 表示一个任意但固定的最大大小独立集，并记$\alpha = \alpha(\cdot)$为最大独立集的大小，其中传递给 $\alpha(\cdot)$的参数是数据流、集合或区间。对于参数 $0 < \beta \leq 1$，若 $|\mathcal{I}| \geq \beta \cdot |OPT|$，我们称独立集 $\mathcal{I}$是一个 β-近似。

我们还假设任意区间都可以存储在单个字的空间中。在端点可以用 $O(\log\ n)$空间位数编码这一合理假设下，这将产生 $O(\log n)$的空间开销。

2.2 通信复杂度

我们采用标准的单向两方通信复杂度模型，该模型最初由[19]提出。随后，我们将通过归约到著名的 INDEX$_t$ 问题来导出下界。关于通信复杂度的全面概述，请参阅 [18] 。

在这一模型中，我们有两方，分别记为 Alice 和 Bob。他们各自持有输入的一部分，分别为 X_A 和 X_B，目标是根据各自的输入计算/近似函数 $f(X_A,\ X_B)$。为此，Alice 向 Bob 发送一条消息 $M(X_A)$；Bob 收到消息后，输出 $f(X_A,\ X_B)$ 或其适当的近似值。于是，Alice 所发送消息的大小就是相关的复杂度度量，而目标是尽可能减少消息的大小。Alice 和 Bob 可以使用无限长的私有随机比特序列以及公共/共享随机比特序列。

定义 1. 对于整数 t，*INDEX*$_t$ 是一个单向两方通信博弈，其中 *Alice* 持有一个向量 $X \in \{0,1\}^t$，*Bob* 持有一个索引 $A \in [t]$。*Alice* 向 *Bob* 发送一条消息，随后 *Bob* 输出值 $X[A]$。

令 μ 表示 INDEX$_t$ 实例上的均匀分布。该分布的定义是：从 $\{0,1\}^t$ 中均匀随机采样 X，并独立地从 X 中采样 A，且 A 在其支撑集 $[t]$ 上均匀随机分布。

众所周知，即使从 μ 中选择输入，并且无论使用何种公共随机性，在分布意义下解决 INDEX$_t$ 仍需要大小为 $\Omega(t)$ 的消息。

定理 3 (INDEX$_t$ 困难性（例如，[18]）。令 $\delta > 0$ 为任意常数。令 P 为一个用于 *INDEX*$_t$ 的单向通信协议。令 R 表示 P 使用的任意公共随机性。若

$$\mathbb{P}_{\mu,R}(P\ outputs\ the\ correct\ answer) \geq 1/2 + \delta,$$

则 P 必须发送大小为 $\Omega(t)$ 位的消息。

3 随机顺序单位区间选择算法

现在我们证明定理 1。为此，我们首先介绍我们的算法 $\mathcal{A}$ （见算法 1），然后对其进行分析。

我们的算法按如下方式进行。它由两个整数 a, b 参数化，且满足 $a < b$，并在如下假设下运行：输入该算法的所有区间都完全包含在定义域 $\mathcal{D} = [a, b)$ 内。我们的总体目标是设计一个在定义域$[0, \Delta)$ 上运行的算法，其中 Δ 为某个整数。为适应这一点，我们稍后将看到，出于技术原因，需要以参数 $a = -1$ 和 $b = \Delta + 1$ 调用我们的算法。

接下来，我们的算法将 $\{a + 1, \ldots, b - 1\}$ 中的所有整数视为分割点。对于每一个这样的分割点 $i \in \{a+1, \ldots, b-1\}$，该算法在变量 L_i和 R_i 中分别维护位于 i 左侧且最接近 i 的区间，以及位于 i右侧且最接近 i 的区间。它还会执行与 i相关联的算法的四次递归调用。递归调用 $\mathcal{T}_i^L$和$\mathcal{T}_i^R$分别在定义域 $[a, i)$ 和 $[i, b)$ 上实例化，并且落入这些定义域的所有区间都会被输入这些算法。递归调用 $\mathcal{A}_i^L$和 $\mathcal{A}_i^R$同样在定义域 $[a, i)$ 和 $[i, b)$ 上实例化，但只有位于 L_i 左侧（且在定义域$[a, i)$ 内）的区间会被输入 $\mathcal{A}_i^L$，而位于 R_i 右侧（且在定义域 $[i, b)$ 内）的区间会被输入$\mathcal{A}_i^R$。

对于每个分割点 i，需要考虑两种可能的输出：由 $\mathcal{T}_i^L$ 生成的输出，与 $\mathcal{A}_i^R$ 的输出以及区间 R_i 相结合；以及由 $\mathcal{A}_i^L$ 生成的输出，

算法 1算法 $\mathcal{A}$

输入： 一个定义域 $\mathcal{D} = [a, b)$，其中 $a,\ b \in \mathbb{Z}$，以及 $a < b$

初始化：

// 对应于每个分割点的递归实例
对于每个整数 $i \in \{a+1, \ldots, b-1\}$，**执行**
    // 初始化最接近每个整数位置的最左和最右区间
    $L_i \leftarrow \emptyset$
    $R_i \leftarrow \emptyset$
    // 初始化每个整数位置左侧的递归实例
    $\mathcal{A}_i^L \leftarrow$ 新建实例 $\mathcal{A}([a, i))$
    $\mathcal{T}_i^L \leftarrow$ 新建实例 $\mathcal{A}([a, i))$
    // 初始化每个整数位置右侧的递归实例
    $\mathcal{A}_i^R \leftarrow$ 新建实例 $\mathcal{A}([i, b))$
    $\mathcal{T}_i^R \leftarrow$ 新建实例 $\mathcal{A}([i, b))$

数据流期间：

输入： 由单位长度区间组成的数据流 $\mathcal{S}$，其中每个区间都完全包含在 $[a,$ 中 $b)$.

对每个区间 $I \in \mathcal{S}$ **执行**
    对每个整数 $i \in \{a+1, \ldots, b-1\}$ **执行**
        // 更新 R_i，并将 I 送入对应的右侧递归实例
        如果 $I \subseteq [i, b)$ **则**
            将 I 送入 $\mathcal{T}^{R}{}_{i}$
            如果 $R_i = \emptyset$ **或(**$R_i \neq \emptyset$ 并且 I 比 R_i更靠左**) 则**
                $R_i \leftarrow I$
            如果 $R_i \neq \emptyset$，I 与 R_i无关，并且 I 比 R_i 更靠右**则**
                将 I 送入 $\mathcal{A}^{R}{}_{i}$
        // 更新 L_i 并将 I 送入相应的左递归实例
        如果 $I \subseteq [a, i)$ **则**
            将 I 送入 $\mathcal{T}^{L}{}_{i}$
            如果 $L_i = \emptyset$ 或者（$L_i \neq \emptyset$ 且 I 位于 L_i的更右侧）**那么**
                $L_i \leftarrow I$
            如果 $L_i \neq \emptyset$，I 独立于 L_i，且 I 位于 R_i 的更左侧**那么**
                将 I 输入到 $\mathcal{A}_i^L$

输出：

其中最大的集合

$$OUT\left(\mathcal{T}_i^L\right) \cup R_i \cup OUT\left(\mathcal{A}_i^R\right) \quad \text{and} \quad OUT\left(\mathcal{A}_i^L\right) \cup L_i \cup OUT\left(\mathcal{T}_i^R\right)$$

对于每个整数 $i \in \{a+1, \ldots, b-1\}$

与 $\mathcal{T}_i^R$ 的输出以及区间 L_i 相结合。我们注意到，根据构造，这两个潜在输出都构成独立集。然后，该算法的输出是在考虑所有分割点时，此类输出中最大的一个。

下文中，我们用 $OUT(\mathcal{A}(\mathcal{S}))$ 表示算法 $\mathcal{A}$ 在输入流 $\mathcal{S}$ 上运行时产生的输出。

3.1 单调性

我们首先证明，无论数据流的顺序如何，从输入流中移除任意区间都不会增大输出的大小。

引理 1. 设 $\mathcal{A}$ 是一个由整数 $a < b$ 实例化的算法 *1* 实例。设 $\mathcal{S}$ 是任意输入流（不一定是随机顺序），其区间均完全包含于 $[a, b)$；设 $\mathcal{S}'$ 是 $\mathcal{S}$ 的任意子流。那么 $|OUT(\mathcal{A}(\mathcal{S}'))| \leq |OUT(\mathcal{A}(\mathcal{S}))|$。

证明。 我们的证明采用对量 $b - a$ 进行归纳法。我们首先证明递归树的大小始终有限。

断言 1. 对于 $b - a > 0$ 的任意取值， $\mathcal{A}$ 的递归树大小有限。

证明。 每个递归子问题都会在任何区间到达之前实例化。因此，递归树的大小只取决于 $b - a$，而不取决于输入流。每个递归实例的定义域都比其父实例的定义域小 1；定义域大小至多为 1 的实例不会生成子问题。 □

现在我们进行归纳证明。

基本情形： 设 $\mathcal{A}$ 是算法 1 的一个实例，且满足 $b-a = 1$。那么 $|OUT(\mathcal{A}(\mathcal{S}'))| \trianglelefteq OUT(\mathcal{A}(\mathcal{S}))|$。

证明。 如果 $b - a = 1$，那么没有单位长度区间可以放入 $[a, b)$。因此， $\mathcal{S}$ 和 $\mathcal{S}'$ 都必须是空数据流，所以 $|OUT(\mathcal{A}(\mathcal{S}'))| = |OUT(\mathcal{A}(\mathcal{S}))| = 0$。 □

归纳假设： 设该引理对所有满足 $b-a \leq k-1$ 的 $\mathcal{A}$ 实例均成立。现在考虑 $b - a = k$ 的情形。

归纳步骤： 根据构造， $\mathcal{A}$ 的输出由其递归子节点的输出组合而成，并附加 L_i 或 R_i 之一，其中某个 i。我们将先说明，如果任何 L_i 或 R_i 在 $\mathcal{S}'$ 下存储了一个区间，那么它也必定在 $\mathcal{S}$ 下存储一个区间。接着，我们将说明，与 $\mathcal{S}'$ 相比，所有递归子节点在 $\mathcal{S}$ 下输出的解都至少同样大。这足以证明该结论。

首先，不失一般性，假设某个 L_i 在 $\mathcal{S}'$ 下非空。那么， $\mathcal{S}'$ 中必定存在一个区间，该区间包含于定义域 $[a, i)$ 。接着，这个区间也必定包含于 $\mathcal{S}$ （$\mathcal{S}$ 是 $\mathcal{S}'$ 的超流），因此 L_i 在 $\mathcal{S}$ 下也必定非空。此论证适用于所有相关整数 i，以及每个 R_i。

现在我们转向输入到 $\mathcal{A}$ 的递归子节点中的子流，并从实例 $\mathcal{T}_i^L$ 和 $\mathcal{T}_i^R$ 开始。不失一般性，考虑某个相关整数 i 对应的 $\mathcal{T}_i^L$。那么，根据构造， $\mathcal{T}_i^L$ 的定义域大小至多为 $k - 1$。接下来，令 $\mathcal{S}_i'$ 表示在将 $\mathcal{S}'$ 输入到 $\mathcal{A}$ 时，输入到 $\mathcal{T}_i^L$ 的区间子流；令 $\mathcal{S}_i$ 表示在将 $\mathcal{S}$ 输入到 $\mathcal{A}$ 时输入的子流。那么， $\mathcal{S}_i'$ 恰好是 $\mathcal{S}'$ 中完全包含在 $[a, i)$ 内的区间所构成的子流。于是， $\mathcal{S}_i'$ 中的每个区间也包含在 $\mathcal{S}'$ 中，因此 $\mathcal{S}$；故 $\mathcal{S}_i'$ 中的每个区间也包含在 $\mathcal{S}_i$ 中。由此可知， $\mathcal{S}_i'$ 是 $\mathcal{S}_i$ 的子流。现在我们可以应用归纳假设，得到

$$\left|OUT\left(\mathcal{T}_i^L(\mathcal{S}_i')\right)\right| \leq \left|OUT\left(\mathcal{T}_i^L(\mathcal{S}_i)\right)\right|.$$

同样的论证适用于每个相关整数 i，也适用于每个 $\mathcal{T}_i^R$。

最后，我们现在考虑实例 $\mathcal{A}_i^L$ 和 $\mathcal{A}_i^R$。不失一般性，考虑某个相关整数 i 对应的 $\mathcal{A}_i^L$。于是，根据构造，$\mathcal{A}_i^L$ 的定义域大小至多为 $k-1$。接下来，令 $\mathcal{S}_i'$ 表示将 $\mathcal{S}'$ 输入 $\mathcal{A}$ 时输入 $\mathcal{A}_i^L$ 的区间子流，并令 $\mathcal{S}_i$ 表示将 $\mathcal{S}$ 输入 $\mathcal{A}$ 时输入的子流。那么，这些分别是 $\mathcal{S}'$ 和 $\mathcal{S}$ 中完全包含在 $[a,i)$ 内的区间所组成的子流；在每个区间到达时，它们也都位于 L_i 的更左侧，且与之相互独立。现在我们注意到，每当 $\mathcal{S}_i'$ 中的一个区间到达 $\mathcal{S}$ 时，根据算法的构造，L_i 中存储的区间必定位于至少与同一区间到达 $\mathcal{S}'$ 时一样靠右的位置。因此，$\mathcal{S}_i'$ 中的每个区间也都包含在 $\mathcal{S}_i$ 中，这说明 $\mathcal{S}_i'$ 是 $\mathcal{S}_i$ 的一个子流。现在我们可以应用归纳假设，得到

$$\left|OUT\left(\mathcal{A}_i^L(\mathcal{S}_i')\right)\right| \le \left|OUT\left(\mathcal{A}_i^L(\mathcal{S}_i)\right)\right|.$$

对于每个相关的整数 i，同样的论证均适用，对于每个 $\mathcal{A}_i^R$ 也是如此。

引理的证明至此完成。 □

3.2 界定期望近似因子

现在，我们推导算法 1 所达到的期望近似因子的一个界。为此，令 Δ 为任意但固定的整数，并考察当所有区间均位于定义域 $[0,\Delta)$ 内时算法的性能。出于技术上的原因，我们在定义域 $[-1,\Delta+1)$ 上运行算法 $\mathcal{A}$，并分析该算法实例的性能。

对于构成输入实例的区间集合 $\mathcal{I}$，我们将 $\Pi(\mathcal{I})$ 记为 $\mathcal{I}$ 的排列集合。注意，此时输入流构成 $\Pi(\mathcal{I})$ 的一个均匀随机元素。

我们关注算法生成的解规模作为 $\alpha(\mathcal{I})$（即最优解大小）的函数。为此，对于任意整数 $x \ge 0$，我们定义如下量：

$$out(x) = \min_{\text{instance } \mathcal{I} \text{ with } \alpha(\mathcal{I})=x} \mathbb{E}_{S\sim\Pi(\mathcal{I})} \left|OUT(\mathcal{A}(S))\right|, \tag{1}$$

即，最优解大小为 x 的实例上算法最坏情况下的期望性能。对于 $x<0$，定义 $opt(x)=0$。

我们首先注意到，只需分析算法在仅由相互独立的区间组成的数据流上的性能。

观察3.1. 设 $\mathcal{I}$ 为任意输入实例，并令 $OPT \subseteq \mathcal{I}$ 表示 $\mathcal{I}$ 中的最优区间。那么，

$$\mathbb{E}_{S_{OPT}\sim\Pi(OPT)} \left|OUT(\mathcal{A}(S_{OPT}))\right| \le \mathbb{E}_{S_{\mathcal{I}}\sim\Pi(\mathcal{I})} \left|OUT(\mathcal{A}(S_{\mathcal{I}}))\right|.$$

证明。 对于区间 $\mathcal{I}$ 的一个排列 $S_{\mathcal{I}}$，我们用 $OPT(S_{\mathcal{I}})$ 表示最优区间的子流。于是，由于 $OPT(S_{\mathcal{I}})$ 是 $S_{\mathcal{I}}$ 的一个子流，我们可以使用引理 1 得到：

$$\mathbb{E}_{S_{\mathcal{I}}\sim\Pi(\mathcal{I})} \left|OUT(\mathcal{A}(OPT(S_{\mathcal{I}})))\right| \le \mathbb{E}_{S_{\mathcal{I}}\sim\Pi(\mathcal{I})} \left|OUT(\mathcal{A}(S_{\mathcal{I}}))\right|.$$

由 S_{OPT} 与 $OPT(S_{\mathcal{I}})$ 同分布这一观察即可得到该结论。

□

因此，从现在起我们假设输入实例 $\mathcal{I}$ 是由一组不相交区间构成的。在本小节余下部分，我们将输入流取为 $\mathcal{S}\sim\Pi(I)$，并省略说明 $\mathcal{S}$ 是 $\mathcal{A}$ 的输入。

接下来，我们将针对每个 $x \ge 0$ 建立 $out(x)$ 的下界。为此，我们显式处理 $x\in\{0,1,2\}$ 的情形，然后对较大的 x 给出递推关系。

引理 2. 对每个 $x \in \{0,1,2\}$， $out(x) = x$。

证明。 首先， $\alpha = 0$ 意味着输入流中没有区间，这是显然的。其次，如果 $\alpha = 1$，那么输入流由一个区间组成。由于R_1 构成定义域 $[0,\Delta)$ 中最左侧的区间，因此该区间会被 R_1 选中。

现在假设 $\alpha = 2$，这意味着输入由两个区间 opt_1 和opt_2 组成，且 opt_1 位于 opt_2的更左侧。如果 opt_1 先于 opt_2到达，那么 R_1 将存储 opt_1，而 opt_2将被传入 $\mathcal{A}_1^R$ 并由这次递归调用返回。随后，区间 opt_2 将作为传入 $\mathcal{A}_1^R$的最右侧区间存储，从而确保 $|OUT（\mathcal{A}_1^R）| \geq 1$。因此我们得到$|R_0 \cup OUT（\mathcal{A}_0^R）| = 2$。然后，根据算法的构造，我们得到

$$|OUT(\mathcal{A})| \geq |\{R_1\} \cup OUT(\mathcal{A}_1^R)| \geq 1 + 1 = 2.$$

类似地，如果 I_2 先到达，那么 $L_{\Delta+1}$ ，且$\mathcal{A}_{\Delta+1}^L$ 的输出构成一个大小为 2 的候选解。

□

接下来，我们考虑 $\alpha \geq 3$ 的情形。记

$$OPT = \{opt_1, \ldots, opt_\alpha\},$$

为按从左到右的顺序排列的最优区间，并考虑 OPT 中哪个区间最先到达的情形。接着，我们固定某个 $i \in \{1,\ldots,\alpha\}$，并假设 opt_i 在 OPT 中其他任何区间之前到达输入流。设 $opt_i = [f, f+1]$，其中 f 为某个实数，并令

$$r = \lfloor f + 2 \rfloor, \text{ and } l = \lfloor f \rfloor$$

分别表示大于 $f+1$ 且与 opt_i 不相交的最小整数，以及小于或等于 f 且可能仅与 opt_i 的左端点相交的最大整数。注意，当 f 为整数时，我们有 $l = f$，但 $r = f + 2$。如此选择是因为，我们算法的每次递归实例化都在一个左端点闭合、右端点开放的定义域上运行。

引理 3. 在 $opt_i = [f, f+1]$ 最先到达的条件下，以下两点均成立：

1. $\mathbb{E}\left[\left|R_l \cup OUT\left(\mathcal{A}_l^R\right)\right| \mid opt_i \text{ arrives first}\right] = 1 + out(\alpha - i)$, *and*
2. $\mathbb{E}\left[\left|L_r \cup OUT\left(\mathcal{A}_r^L\right)\right| \mid opt_i \text{ arrives first}\right] = 1 + out(i-1)$.

证明。 我们只证明 1)——2) 的证明类似，故省略。

根据定义，区间 opt_i 位于更右侧，且与 l相互独立，因此会被存储在R_l中。接下来注意，每个区间 opt_{i+1}、.....、 opt_α 都以均匀随机的顺序输入 $\mathcal{A}_l^R$。于是，根据 out(。)的定义，可得 $\mathbb{E}|OUT(\mathcal{A}_l^R)| \geq out(\alpha - i)$。最后，通过期望的线性性合并$R_l$ 和 $\mathcal{A}_R^l$ ，即可得到所需的界。

注记： 我们注意到，每个区间都包含在定义域 $[0,\Delta)$ 内，这意味着 $0 \leq l$且 $r \leq \Delta$。回想一下，我们在定义域 $[-1, \Delta+1)$ 上运行算法 $\mathcal{A}$，这意味着实例 $\mathcal{A}_l^R$和 $\mathcal{A}_r^L$ 都必定存在。

□

下一个引理可通过类似的证明得到。

引理 4。 在给定 $opt_i = [f, f+1]$ 最先到达的条件下，以下两条均成立：

1. $\mathbb{E}\left[\left|OUT\left(\mathcal{T}_r^R\right)\right| \mid opt_i \text{最先到达}\right] \geq out（\alpha - i - 1）$，并且
2. $\mathbb{E}\left[\left|OUT\left(\mathcal{T}_l^L\right)\right| \mid opt_i \text{首先到达}\right] \geq out（i-2）$。

证明。 我们只证明 2)——1) 的证明类似，故略去。

根据构造，输入到 $\mathcal{T}_l^L$ 中的子流 $\mathcal{S}'$ 由 $\{opt_1, \ldots, opt_{i-1}\}$ 中与点 l 不重叠的区间组成。由于每个区间的长度均为单位长度，且 $f - l \leq 1$ 成立，因此 $\mathcal{S}'$ 包含 OPT 中的 $i-1$ 或 $i-2$ 最左区间，而这些区间以均匀随机顺序输入到 $\mathcal{T}_l^L$ 中。由此得到 $\mathbb{E}|OUT(\mathcal{T}_l^L)| \geq \min\{out(i-1)\ out(i-2)\}$。最后，根据引理 1， $out(\cdot)$ 是非递减函数，这意味着 $out(i-1) \geq out(i-2)$ 成立，从而完成证明。 □

现在我们结合这些引理，为 $out(\cdot)$ 构造一个递推式。下文中，为方便起见，我们将最大值函数分成多行书写。

引理 5. 在条件 opt_i 最先到达的情况下，以下结论成立：

$$\mathbb{E}\left[|OUT(\mathcal{A})| \mid opt_i \ arrives \ first|\right] \geq 1 + \max\left\{ \begin{array}{l} out(i-1) + out(\alpha - i - 1) \\ out(\alpha - i) + out(i-2) \end{array} \right\}.$$

证明。 由引理3和引理4几乎可以立即得出证明：

$$\begin{aligned}
\mathbb{E}|OUT(\mathcal{A}) \mid opt_i \text{ arrives first}| &\geq \mathbb{E}\left[\max\left\{ \begin{array}{l} |OUT\left(\mathcal{A}_r^L\right) \cup L_r \cup OUT\left(\mathcal{T}_r^R\right)| \\ |OUT\left(\mathcal{T}_l^L\right) \cup R_l \cup OUT\left(\mathcal{A}_l^R\right)| \end{array} \right\}\right] \\
&\qquad \text{(Construction of the algorithm.)} \\
&\geq \max\left\{ \begin{array}{l} \mathbb{E}|OUT\left(\mathcal{A}_r^L\right) \cup L_r \cup OUT\left(\mathcal{T}_r^R\right)| \\ \mathbb{E}|OUT\left(\mathcal{T}_l^L\right) \cup R_l \cup OUT\left(\mathcal{A}_l^R\right)| \end{array} \right\} \\
&\qquad \text{(Jensen's inequality.)} \\
&\geq \max\left\{ \begin{array}{l} 1 + out(i-1) + out(\alpha - i - 1) \\ 1 + out(\alpha - i) + out(i-2) \end{array} \right\}. \\
&\qquad \text{(Lemmas 3 and 4.)} \\
&\geq 1 + \max\left\{ \begin{array}{l} out(i-1) + out(\alpha - i - 1) \\ out(\alpha - i) + out(i-2) \end{array} \right\}
\end{aligned}$$

□

在以下引理中，我们给出算法的期望近似因子的一个下界。

引理6。 $\mathcal{A}$在定义域$[0, \Delta)$ 内各区间上达到的期望近似因子有如下下界

$$\begin{aligned}
&\mathbb{E}[approximation \ factor] \\
&\geq \min_{\alpha \in \{1, \ldots, \Delta - 1\}} \left\{ \frac{1}{\alpha} + \frac{1}{\alpha^2} \cdot \sum_{i=1}^{\alpha} \max\left\{ \begin{array}{l} out(i-1) + out(\alpha - i - 1) \\ out(\alpha - i) + out(i-2) \end{array} \right\} \right\},
\end{aligned}$$

其中 $out(2) = 2$, $out(1) = 1$, 且 $out(y) = 0$ 对所有 $y \leq 0$均成立。

证明。引理 t $\alpha \in \{1, \ldots, \Delta - 1\}$ 并考虑一个输入实例 $\mathcal{I}$，它确立了最小值 m

在 $out(\alpha)$ 的定义中。我们考虑 $\mathcal{A}$ 在 $\mathcal{I}$ 上的一次运行。于是:

$$\begin{aligned}
out(\alpha) &= \mathbb{E}\left[|OUT(\mathcal{A})|\right] \\
&= \sum_{i=1}^{\alpha} \mathbb{P}(opt_i \text{ arrives first}) \cdot \mathbb{E}\left[|OUT(\mathcal{A})| \mid opt_i \text{ arrives first}\right] \quad \text{(Law of total expectation.)} \\
&= \sum_{i=1}^{\alpha} \frac{1}{\alpha} \cdot \mathbb{E}\left[|OUT(\mathcal{A})| \mid opt_i \text{ arrives first}\right] \quad \text{(Uniform random arrival order.)} \\
&\geq \frac{1}{\alpha} \cdot \sum_{i=1}^{\alpha} \left[1 + \max\left\{ \begin{array}{l} out(i-1) + out(\alpha - i - 1) \\ out(\alpha - i) + out(i-2) \end{array} \right\}\right] \quad \text{(By Lemma 5.)} \\
&= 1 + \frac{1}{\alpha} \cdot \sum_{i=1}^{\alpha} \max\left\{ \begin{array}{l} out(i-1) + out(\alpha - i - 1) \\ out(\alpha - i) + out(i-2) \end{array} \right\}
\end{aligned}$$

特别地，这为 $out(\cdot)$提供了一个递推关系，我们稍后将用它来数值计算。期望近似因子就是将该量乘以 $1/\alpha$。

最后，当我们的算法在定义域 $[0, \Delta)$ 内的区间上运行时，我们只能知道$\alpha \in \{1, \ldots, \Delta - 1\}$。因此，为了界定算法在此类实例上的近似因子，我们考虑 $\alpha \in \{1, \ldots, \Delta - 1\}$ 的所有可能取值，并选取这些情况下的最小近似因子，由此得证该引理。 □

3.3 界定所用空间

本节中，我们将为算法所用的空间给出一个界。

引理 7. 设 $\mathcal{A}$ 是一个实例，在定义域边界 $a < b$ 上运行算法 *1*，且满足 $b - a = k$。那么， $\mathcal{A}$ 使用 $O(4^k k^k)$ 字空间。

证明。 根据构造， $\mathcal{A}$ 由以下部分组成:

- $4k - 4$ 个算法的递归实例，每个实例的输入定义域长度至多为 $k - 1$,

至多 $2k - 2$ 个存储区间（在每个 L_i 和 R_i中）。

然后，令 $S(k)$ 表示算法 1 在长度为k的定义域上的一个实例所使用的空间，我们得到递推式

$$S(k) \leq (4k - 4) \cdot S(k - 1) + 2k - 2 \ ,$$

这意味着 $S(k) = O(4^k k^k)$。 □

3.4 转向非受限定义域

现在我们采用滑动窗口思想，该方法最初由 Hochbaum 和 Mass 提出 [14], ，后来由 [7]和 [11],应用，从而能够从受限域转向非受限定义域。

下面的引理是将 [11] 中针对对抗顺序流给出的相应引理直接改写为适用于随机顺序流的形式。关键区别在于，我们现在还利用期望的线性性来处理数据流的随机顺序，而此前考虑的是对抗顺序流。除此之外，证明相当相似。

引理 8. 设 $\Delta \geq 2$ 为一个整数。设 $\mathcal{A}$ 为任意一个用于受限域 $[0, \Delta)$ 上的随机顺序单位长度区间选择的（确定性）算法，该算法能够以期望（针对数据流的随机顺序）计算 α-近似，并使用空间 $O(s)$。那么，存在一个用于非受限域上的随机顺序单位长度区间选择的（确定性）算法，该算法能够以期望（针对数据流的随机顺序）计算 $\alpha \cdot (\Delta - 1)/\Delta$-近似，使用空间 $O(s \cdot \Delta \cdot |OPT|)$，并将 $\mathcal{A}$用作黑盒。

证明。设 $\mathcal{A}$ 为任意一个用于受限域 $[0,\Delta)$ 上的随机顺序单位长度区间选择的确定性算法，该算法能够以期望（针对数据流的随机顺序）计算 α-近似，并使用空间 $O(s)$。我们将 $\mathcal{A}$ 用作黑盒，以给出一个用于非受限域的算法。

为此，对于所有 $i \in \mathbb{Z}$，我们首先定义窗口 $W_i^\Delta = [i, i+\Delta)$。初始时，我们将每个窗口 $\{W_i^\Delta\colon\ i \in \mathbb{Z}\}$ 标记为非活跃。对于数据流中到达的每个区间 I，令 $\mathcal{W}_I = \{W_i^\Delta\colon\ I \subseteq W_i^\Delta\}$ 为完全包含 I 的窗口集合。对于每个 $W \in \mathcal{W}_I$，若 W 被标记为非活跃，则为对应于 W 的 $\mathcal{A}$ 创建一个新实例，并将 I 输入 $\mathcal{A}$。然后将 W 标记为活跃。否则，若 W 已被标记为活跃，则已有一个对应于 W 的 $\mathcal{A}$ 实例在运行。此时，我们将区间 I 输入 $\mathcal{A}$ 的实例。需要注意的是，作为一项技术处理，每当要将区间 $I = [a,b]$ 输入对应于窗口 W_i^Δ 的实例 $\mathcal{A}$ 时，就向 $\mathcal{A}$ 提供变换后的区间 $I' = [a-i, b-i]$，使得 $\mathcal{A}$ 的输入包含在受限域 $[0,\Delta)$ 内。

对于每个窗口 W，令 $OUT(W)$ 表示数据流结束时对应于 W 的 $\mathcal{A}$ 实例的输出。我们隐式地将 $OUT(W)$ 中的每个区间转换回去，使其对应于非受限域上的区间。在数据流结束时，输出一个由以下区间组成的最大独立集

$$\bigcup_{\substack{i\in\mathbb{Z}\\ W_i^\Delta \text{ is active}}} OUT(W_i^\Delta)$$

我们通过仅存储标记为活动状态的窗口集合来优化算法所使用的空间；也就是说，在某个窗口被明确标记为活动状态之前，我们都隐式地将其视为非活动状态。因此，标记为非活动状态的窗口不占用任何空间。

算法使用的空间恰好是标记为活动状态的窗口所使用的空间，而每个窗口使用 $O(s+1) = O(s)$ 空间（用于存储 $\mathcal{A}$ 并将其标记为活动状态）。对于每个窗口 W_i^Δ，定义 $W_i^\Delta(L) = [i-1, i-1+\Delta)$ 和 $W_i^\Delta(R) = [i+1, i+1\Delta)$ 分别表示向左和向右移动一个位置后的窗口。接着定义

$$\begin{aligned} W(OPT) &= \{W_i^\Delta : \exists I \in OPT \text{ s.t. } I \subseteq W_i^\Delta\},\\ W_L(OPT) &= \{W(L) : W \in W(OPT)\},\\ W_R(OPT) &= \{W(R) : W \in W(OPT)\}. \end{aligned}$$

现在定义

$$\mathcal{W}' = W(OPT) \cup W_L(OPT) \cup W_R(OPT).$$

接下来，对于每个区间 I，恰好存在 $\Delta - 1$ 个 j 的值，使得 $I \subseteq W_j^\Delta$。因此，$|\mathcal{W}'| \le 3 \cdot \Delta \cdot |OPT|$。此外，被标记为活动的窗口集合必须是 $\mathcal{W}'$ 的子集。否则，就存在一个被标记为活动的窗口，它与 OPT 中的任何区间都不相交，因此 $\mathcal{S}$ 中存在一个与 OPT 中任何区间都不相交的区间。这与最大独立集的极大性相矛盾。由此可知，该算法使用的总空间受以下值所界定：

$$O(s) \cdot O(\Delta \cdot |OPT|) = O(s \cdot \Delta \cdot |OPT|).$$

对于任意窗口 W，包含于 W 中的 $\mathcal{S}$ 内的区间集合与 $\mathcal{S}$ 的顺序无关。因此，W 上的 $\mathcal{S}$ 子流也是一个[2]随机顺序流。

接下来，对于每个 $j \in [\Delta]$，定义不相交窗口集合 $\mathbb{W}_j = \{W_{j+a\Delta}^\Delta :\ a \in \mathbb{Z}\}$，并定义 $OPT_j = \{I \in OPT\colon \text{满足 } I \subseteq W\}$ 的 $\exists W \in \mathbb{W}_j$。然后，利用 OPT 中的每个区间恰好包含于 $\Delta - 1$ 个窗口中，且对于任意给定的 $\mathbb{W}_j$ 仅有一个窗口，

$$\sum_{j=1}^{\Delta} |OPT_j| = (\Delta - 1)|OPT|.$$

[2]这正是我们的论证在 [11] 提出的类似论证基础上进一步推进之处。

因此，存在某个 $k \in [\Delta]$ ，使得

$$|OPT_k| \geq \frac{\Delta - 1}{\Delta} \cdot |OPT|. \tag{2}$$

现在考虑窗口集合 $\mathbb{W}_k$。 $\mathbb{W}_k$ 中的每个窗口两两不相交，因此 $\bigcup_{W \in \mathbb{W}_k} OUT(W)$ 是 $\mathcal{S}$ 中区间的一个有效独立集。然后，对数据流的随机顺序取期望，

$$\begin{aligned}
\mathbb{E}[|OUT|] &\geq \mathbb{E}\left[\sum_{W \in \mathbb{W}_k} |OUT(W)|\right] && (\textstyle\bigcup_{W \in \mathbb{W}_k} OUT(W) \text{ is a valid independent set.}) \\
&= \sum_{W \in \mathbb{W}_k} \mathbb{E}[|OUT(W)|] \\
&\quad (\text{Linearity of Expectation and the substream of } \mathcal{S} \text{ on } W \text{ is also random-order.}) \\
&\geq \sum_{W \in \mathbb{W}_k} \alpha \cdot |\{I \in OPT : I \subseteq W\}| \\
&\quad (\mathcal{A} \text{ computes an } \alpha\text{-approximation in expectation.}) \\
&= \alpha \cdot \sum_{W \in \mathbb{W}_k} |\{I \in OPT : I \subseteq W\}| \\
&= \alpha \cdot \left| \bigcup_{W \in \mathbb{W}_k} \{I \in OPT : I \subseteq W\} \right| \\
&\quad (\text{Each } W \in \mathbb{W}_k \text{ is disjoint, so each } I \text{ is contained in exactly one } W \in \mathbb{W}_k.) \\
&= \alpha \cdot |\{I \in OPT : \exists W \in \mathbb{W}_k \text{ s.t. } I \subseteq W\}| \\
&= \alpha \cdot |OPT_k| && (\text{Definition of } OPT_k.) \\
&\geq \alpha \cdot \frac{\Delta - 1}{\Delta} \cdot |OPT|. && (\text{Equation 2.})
\end{aligned}$$

这个适用于非受限定义域的元算法是确定性的。因此，当且仅当 $\mathcal{A}$ 是确定性的，适用于非受限定义域的最终算法才是确定性的。 □

3.5 完成分析

现在，我们将算法 1 与引理 8 结合起来，以完成定理 1 的证明。

定理 1. 存在一种确定性的单遍($O(|OPT|)$ 字空间流式算法，可在随机顺序流上解决单位区间选择问题，其期望近似因子为0.7401，其中期望是对数据流的随机顺序取的。

证明。 将算法 1 与引理 8 结合，可得到一种适用于随机顺序、区间长度为单位长度且输入域不受任何限制的数据流的算法。我们首先考察这种方法所需的空间。接着，我们用一个仅取决于整数窗口长度 Δ （来自引理 8）的函数，给出非受限定义域上所能达到的期望近似因子的下界，最后对该函数进行数值优化。

根据第 3.2 小节，在长度为 Δ 的窗口上，我们运行一个域大小为 $\Delta + 2$ 的算法。然后，由引理 7 和引理 8，这给出了一个用于非受限定义域的算法，该算法使用

$$\begin{aligned}
O(S(\Delta + 2) \cdot \Delta \cdot |OPT|) &= O\left(4^{\Delta+2} \cdot (\Delta + 2)^{\Delta+2} \cdot \Delta \cdot |OPT|\right) \\
&= O\left(4^{\Delta} \cdot (\Delta + 2)^{\Delta+3} \cdot |OPT|\right),
\end{aligned}$$

字空间，其中 $S(k)$ 表示在长度为 k 的定义域上实例化的算法实例所使用的空间（来自引理 7）。然后，当 $\Delta = O(1)$ 时，该空间收缩为$O(|OPT|)$。

我们现在考虑不受限算法的近似因子。注意到窗口 $[0, \Delta)$ 中最大独立集的大小至多为 $\Delta - 1$。因此，结合引理 6 和引理 8，对于任意 Δ，存在一种用于非受限定义域上随机顺序单位长度区间选择的算法，其期望近似因子为

$$\frac{\Delta-1}{\Delta} \cdot \min_{\alpha \in \{1,\ldots,\Delta-1\}} \left\{ \frac{1}{\alpha} + \frac{1}{\alpha^2} \cdot \sum_{i=1}^{\alpha} \max \left\{ \begin{array}{l} out(i-1) + out(\alpha - i - 1) \\ out(\alpha - i) + out(i-2) \end{array} \right\} \right\}. \tag{3}$$

接下来需要选取一个合适的常数 Δ。使用 $\Delta = 5{,}000$ 数值计算公式（3）（利用引理6 中 $out(\cdot)$ 的递推关系），可得期望近似因子的下界略高于 0.7401，从而完成证明。我们将在附录 A 中进一步讨论 Δ的这一选择。

□

4 下界

现在我们证明定理 2 中所述的下界结果。

给定一个随机顺序单位区间选择算法 $\mathcal{A}$，我们定义协议 Q，该协议使用 $\mathcal{A}$来解决 Index$_t$ 问题:

用于 INDEX$_t$的协议：

输入： 设 Alice 和 Bob 是持有来自均匀分布 μ的一个 INDEX$_t$ 实例（X, A）的两位参与者。设 $\mathcal{A}$是随机顺序单位区间选择的任意算法。对于每个 $i \in [t]$，定义二元随机变量 $Y_i \in \{0, 1\}$，它们服从均匀分布且相互独立。此外，定义服从均匀分布的随机变量 $B_L = B_R \in [t]$。令 T 表示包含这些随机变量的集合，即 $T = \{Y_1, \ldots, Y_t, B_L, B_R\}$。

协议：

1. Alice 和 Bob 使用共享公共随机性对 T 中的每个变量进行采样。他们还对一个双射 $\sigma : T \to [|T|]$ 进行采样。

2. 设 $j' = \min\{\sigma(B_L)\ \sigma(B_R)\}$。

3. 对于每个满足 $\sigma(Y_i) < j'$ 的 $i \in [t]$，Alice 构造区间

$$I[i] = \left[\frac{i}{t+1} + \frac{X[i]}{t^2}, 1 + \frac{i}{t+1} + \frac{X[i]}{t^2} \right].$$

4. Alice 使用排列 σ 在这些构造出的区间上运行 $\mathcal{A}$。注意，σ 通过将每个 $I[i]$ 与 $Y[i]$ 关联，为这些区间定义了一个排列。随后，Alice 将 $\mathcal{A}$ 的状态作为消息发送给 Bob。

5. 对于每个满足 $\sigma(Y[i]) > j'$ 的 $i \in [t]$，Bob 构造区间

$$I[i] = \left[\frac{i}{t+1} + \frac{Y[i]}{t^2}, 1 + \frac{i}{t+1} + \frac{Y[i]}{t^2} \right].$$

Bob 同样构造这些区间

$$J_L = \left[\frac{A}{t+1} - \frac{1}{t^3} - 1, \frac{A}{t+1} - \frac{1}{t^3} \right]$$

and

$$J_R = \left[1 + \frac{A}{t+1} + \frac{1}{t^2} + \frac{1}{t^3}, 2 + \frac{A}{t+1} + \frac{1}{t^2} + \frac{1}{t^3} \right].$$

6. Bob 使用来自 Alice 的消息，继续在这些区间上运行 $\mathcal{A}$，使用排列 σ，将 J_L 与 B_L关联，并将 J_R 与 B_R关联，且在 σ下进行。

输出： 如果 $\mathcal{A}$ 产生的输出大小至多为 2，或者 $\sigma(Y[A]) > j'$，那么 Bob 输出一个均匀随机比特。否则，$\mathcal{A}$ 必定输出一个大小为 3 的解，且 $\sigma(Y[A]) < j'$。如下文所述，这个解必定包含区间 $I[A]$。接着，如果 $I[A]$ 位于位置 $[\frac{A}{t+1}, 1+\frac{A}{t+1}]$，那么 Bob 输出比特 $x=0$；否则，该区间必定位于位置 $[\frac{A}{t+1}+\frac{1}{t^2}, 1+\frac{A}{t+1}+\frac{1}{t^2}]$，Bob 输出比特 $x=1$。

协议 Q 如下进行。给定一个均匀随机输入 (X, A) 到 INDEX$_t$，Alice 和 Bob 首先构造一个随机区间实例，该实例由均匀随机比特 $Y_1, \ldots\ldots, Y_t$以及来自 INDEX$_t$ 输入的索引 A 完全描述：每个比特 Y_i 对应一个区间 $I[i]$，使得区间 $I[1], \ldots\ldots, I[t]$ 像图 1 中那样彼此相交。比特 Y_i的作用如下：如果 $Y_i = 1$，那么区间 $I[i]$ 会略微向右移动；而如果 $Y_i = 0$，则不会发生这种情况。然后，索引 A 转换为另外两个区间 J_L 和 J_R，它们将区间 $I[A]$ 包围起来，使得 $\{J_L, J_R, I[A]\}$ 构成唯一的大小为 3 的独立集。

我们注意到，迄今为止构造出的实例是一个均匀随机实例，因为所有变量Y_1、……、Y_t、A 都是相互独立的均匀随机变量。

接下来，Alice 和 Bob 对该实例中的区间进行均匀随机排列 σ。我们强调，根据构造，σ 与已构造的区间相互独立，也就是说，它与各个量Y_1、……、Y_t 以及 A相互独立。

Alice 现在按如下方式将一些比特 Y_i 替换为值 X_i：如果在均匀随机排列 σ 中，区间 $I[i]$ 先于 J_L 和 J_R到达，那么 Y_i 就更新为值 X_i。从分布的角度来看，这一过程将均匀随机比特 Y_i 替换为不同的均匀随机比特 X_i（回想一下，INDEX$_t$ 实例的输入是一个均匀随机实例）。因此，所得区间实例服从均匀分布；最重要的是，区间的顺序是随机的，并且与实例相互独立。因此，该归约构造出了一个随机顺序流。

我们注意到，Alice 只会更新在区间 J_L 和 J_R 之前到达的区间，换句话说，更新哪些区间取决于流顺序。因此，人们可能会错误地得出结论，认为由此产生的实例可能与到达顺序 σ 相关，这意味着所得到的数据流并非均匀随机顺序。然而，我们强调，事实并非如此，因为这一过程只是将均匀随机比特替换为另一组均匀随机比特。因此，不可能建立这种相关性。

接下来我们看到，每当 Alice 将正确的比特 $X[A]$ 编码到区间 $I[A]$ 中时——正如我们将要看到的，这以概率 $1/3$ 发生——并且算法 $\mathcal{A}$ 同时成功时，Alice 和 Bob 就能够解决底层的 INDEX$_t$ 实例。否则，Bob 输出一个均匀随机比特。我们将看到，如果 $\mathcal{A}$ 以大于 $2/3$ 的概率成功，那么协议 Q 的成功概率将严格大于 $1/2$。定理 3 中所述的 INDEX$_t$ 下界意味着，Q 必须通信 $\Omega(t)$ 个比特，因此，算法 $\mathcal{A}$ 也必须至少使用同样多的空间。

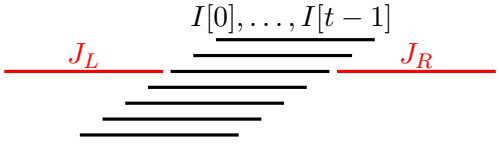


图 1：由协议 Q构造的区间示例。黑色区间对应每个 $I[i]$，红色区间表示 J_L 和 J_R。每个 $I[i]$ 区间是通过 $X[j]$ 还是公共随机比特构造的，取决于采样得到的排列 σ。

定理 2. 设 $\epsilon > 0$ 为任意一个较小的常数。那么：

1. 任何用于随机顺序单位区间选择、期望近似因子为 $8/9 + \epsilon$ 的随机化单遍流算法都必须使用 $\Omega(n)$ 空间位数。*2.* 对于任意 $\delta > 0$，任何用于随机顺序单位区间选择、近似因子为 $2/3 + \delta$ 且在任意输入实例上以至少 $2/3 + \epsilon$的概率成功的随机化单遍流算法都必须使用 $\Omega(n)$空间位数。

第（*1*）项中的期望和第（*2*）项中的概率均同时取自流顺序和算法的随机选择。

证明。设 $\mathcal{A}$ 是一个用于单位区间选择的随机化算法，其近似因子为$2/3 + \delta$（对任意小常数 δ），并且对某个$\epsilon > 0$，在每个输入实例上以至少 $2/3 + \epsilon$的概率成功，其中概率同时取自流顺序和算法的随机硬币翻转。我们将首先证明 $\mathcal{A}$ 必须使用 $\Omega(n)$ 空间，即我们确立结果 2。然后，我们论证结果 2 直接蕴含结果 1。

为了证明结果 2 成立，我们首先注意到，每当算法 $\mathcal{A}$ 成功时，它必定输出一个大小为 3 的解，因为它的近似因子严格大于 $2/3$ ，而且构造的每个实例都包含一个大小为 3 的最优解。其次，如上所述，无论构造出何种输入实例，它都会以均匀随机顺序呈现给 $\mathcal{A}$ ，这意味着该算法的成功概率确实为 $2/3 + \epsilon$。第三，我们看到，以概率 $1/3$，区间 $I[A]$ 由 Alice 创建，因此能够正确反映比特 $X[A]$。因此，我们关心的是 $I[A]$ 反映 $X[A]$ 且算法 $\mathcal{A}$ 在由此产生的实例上成功的概率。

利用事件 $\mathcal{E}_1 =$ “Alice 创建 $I[A]$”和 $\mathcal{E}_2 =$“$\mathcal{A}$ 成功”，我们得到：

$$\Pr[\mathcal{E}_1 \text{ and } \mathcal{E}_2] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] = \frac{1}{3} \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \ . \tag{4}$$

接下来，我们利用 $\mathcal{A}$ 的成功概率至少为 $2/3 + \epsilon$ 这一事实。因此，

$$\begin{aligned} 2/3 + \epsilon \le \Pr[\mathcal{E}_2] &= \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] + (1 - \Pr[\mathcal{E}_1]) \cdot \Pr[\mathcal{E}_2 \mid \overline{\mathcal{E}_1}] \\ &= \frac{1}{3} \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] + \frac{2}{3} \cdot \Pr[\mathcal{E}_2 \mid \overline{\mathcal{E}_1}] \\ &\le \frac{1}{3} \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] + \frac{2}{3} \cdot 1 \ , \end{aligned}$$

这意味着 $\epsilon \le \Pr[\mathcal{E}_2 \mid \mathcal{E}_1]$. 将此界代入等式 4，可得 $\Pr[\mathcal{E}_1$ 和 $\mathcal{E}_2] \ge \frac{1}{3} \cdot \epsilon$。换言之，以概率 $\frac{1}{3} \cdot \epsilon$，Bob 能够解出底层的 Index_t 实例。

最后，在所有其他情况下，即以概率 $1 - \epsilon/3$，Bob 输出一个均匀随机比特。因此，协议 Q 解决 Index_t 实例的概率为 $\epsilon/3 + (1 - \epsilon/3) \cdot \frac{1}{2} = \frac{1}{2} + \epsilon/6$。于是，根据定理 3，所得协议必须使用大小为 $\Omega(t)$ 比特的消息；由于消息大小与我们的算法所用空间相同，因此同一下界也适用于 $\mathcal{A}$的空间使用量。最后，由于 $n = t + 2$ 在我们的构造中，结果 2 得证。

为证明结果 1 成立，考虑一个用于单位区间选择的算法 $\mathcal{A}$ ，其期望近似因子为 $\frac{8}{9} + \epsilon$，其中 $\epsilon > 0$；这里的期望是对算法的随机硬币翻转以及数据流顺序取的。我们现在将论证， $\mathcal{A}$ 以至少 $2/3 + \epsilon$ 的近似因子达到该结果的概率为 $2/3 + \epsilon$。于是，根据结果 2 可知， $\mathcal{A}$ 必须使用 $\Omega(n)$ 比特的空间，从而证明结果 1。

我们计算：

$$\begin{aligned} \frac{8}{9} + \epsilon \le \mathbb{E}[\text{approx factor}] \\ &\le \Pr[\text{approx factor} \le 2/3 + \epsilon] \cdot (2/3 + \epsilon) + \Pr[\text{approx factor} > 2/3 + \epsilon] \cdot 1 \\ &= (1 - \Pr[\text{approx factor} > 2/3 + \epsilon]) \cdot (2/3 + \epsilon) + \Pr[\text{approx factor} > 2/3 + \epsilon] \ , \end{aligned}$$

由此可得，$\Pr$[近似因子$> 2/3 + \epsilon] \ge \frac{2}{3} + \epsilon$ ，符合预期。 □

5 结论

本文针对单位区间选择提出了一种单遍随机顺序流式算法，其期望近似因子为 0.7401，所需空间为 $O(|OPT|)$，其中期望是对数据流的顺序取的。我们还证明，在此空间限制下，没有算法能够实现超过 $8/9$ 的期望近似因子，也没有算法能够以高于 $2/3$ 的概率达到优于 $2/3$ 的近似因子。

我们以两个尚未解决的问题作为结尾。

1. 我们的结果并非紧确。能否设计出具有更优期望近似保证的算法，或者证明更强的下界?

2. 我们尚未探讨任意长度的区间。对于任意长度的区间，是否存在一种算法，其区间选择的期望近似因子高于 $1/2$?

参考文献

[1]Cezar-Mihail Alexandru 和 Christian Konrad。《滑动窗口中的区间选择》。载于 Timothy Chan、Johannes Fischer、John Iacono 和 Grzegorz Herman 编，第 *32* 届欧洲算法研讨会（*ESA 2024*），第 308 卷，莱布尼茨信息学国际论文集（*LIPIcs*），第 8:1–8:17 页，德国达格斯图尔，2024 年。达格斯图尔城堡—莱布尼茨信息学中心。网址：https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ESA.2024.8，doi:10.4230/LIPIcs.ESA.2024.8。

[2]Sepehr Assadi和Soheil Behnezhad。在随机序流式匹配中突破三分之二。载于Nikhil Bansal、Emanuela Merelli和James Worrell编，第*48*届自动机、语言与程序设计国际学术会议，*ICALP 2021*，*2021*年*7*月*12*日至*16*日，苏格兰格拉斯哥（线上会议），*LIPIcs*第198卷，第19:1–19:13页。达格斯图尔城堡－莱布尼茨信息学中心，2021年。URL: https://doi.org/10. 4230/LIPIcs.ICALP.2021.19，doi:10.4230/LIPICS.ICALP.2021.19。

[3] Sepehr Assadi和Janani Sundaresan。（含噪声）间隙环计数卷土重来：连通分量及其他问题的随机顺序流下界。载于第*55*届年度*ACM*计算理论研讨会论文集，第183–195页，2023年。

[4] Ainesh Bakshi、Nadiia Chepurko 和 David P. Woodruff。旋转门流中几何对象的加权最大独立集。载于 Jarosław Byrka 和 Raghu Meka 编，近似、随机化与组合优化：算法与技术（*APPROX/RANDOM 2020*），LIPIcs 第 176 卷，页码 64:1–64:22，德国达格斯图尔，2020 年。Schloss Dagstuhl – Leibniz 信息学中心。网址：https://drops.dagstuhl.de/entities/document/10. 4230/LIPIcs.APPROX/RANDOM.2020.64，doi:10.4230/LIPIcs.APPROX/RANDOM.2020.64。

[5] Aaron Bernstein。随机顺序流中匹配的改进界。计算系统理论，68(4):758–772，2023年12月。doi:10.1007/s00224-023-10155-7。

[6] Sujoy Bhore、Fabian Klute 和 Jelle J. Oostveen。几何独立集与团的流式算法。载于 Parinya Chalermsook 和 Bundit Laekhanukit 编，《近似与在线算法——第20届国际研讨会 WAOA 2022，德国波茨坦，2022年9月8-9日，论文集》，第13538卷，计算机科学讲义，第211–224页。施普林格，2022年。doi:10.1007/978-3-031-18367-6_11。

[7] Sergio Cabello 和 Pablo Pérez-Lantero。流模型中的区间选择。理论计算机科学，702:77–96，2017年。网址：https://doi.org/10.1016/j.tcs.2017.08.015，doi:10.1016/J.TCS.2017.08.015。

[8] Amit Chakrabarti、Graham Cormode 和 Andrew McGregor。通信与流计算的鲁棒下界。载于第四十届美国计算机学会计算理论年度研讨会论文集，STOC ’08，第641–650页，美国纽约州纽约，2008年。美国计算机学会。doi:10.1145/1374376.1374470。[9] Amit Chakrabarti、T. S. Jayram 和 Mihai Pundefinedtraşcu。随机排序流中选择问题的紧下界。载于第十九届美国计算机学会—美国工业与应用数学学会离散算法研讨会论文集，SODA ’08，第720–729页，美国，2008年。美国工业与应用数学学会。[10]Graham Cormode、Jacques Dark 和 Christian Konrad。顶点到达流中的独立集。载于 Christel Baier、Ioannis Chatzigiannakis、Paola Flocchini 和 Stefano Leonardi 编，《第46届自动机、语言与编程国际研讨会，ICALP 2019，2019年7月9日至12日，希腊帕特雷》，第132卷*LIPIcs*，第45:1–45:14页。达格斯图尔城堡—莱布尼茨信息学中心，2019年。网址：https://doi.org/10.4230/LIPIcs.ICALP.2019.45，doi:10.4230/LIPICS.ICALP.2019.45。[11] Jacques Dark、Adithya Diddapur 和 Christian Konrad。数据流中的区间选择：加权区间与插入-删除设置。载于 Patricia Bouyer 和 Srikanth Srinivasan 编，第*43*届 *IARCS* 软件技术与理论计算机科学基础年度会议，*FSTTCS 2023*，*2023*年*12*月*18*日至*20*日，印度特伦甘纳邦海得拉巴 *IIIT*，第284卷*LIPIcs*，第24:1–24:17页。达格斯图尔城堡—莱布尼茨信息学中心，2023年。网址：https://doi.org/10.4230/LIPIcs.FSTTCS.2023.24，doi:10.4230/LIPICS.FSTTCS.2023.24。[12] Yuval Emek、Magnús M. Halldórsson 和 Adi Rosén。空间受限区间选择。美国计算机学会算法汇刊，12(4)，2016年9月。doi:10.1145/2886102。[13] Sudipto Guha 和 Andrew McGregor。流顺序与顺序统计量：随机顺序流中的分位数估计。*SIAM*计算杂志，38(5):2044–2059，2009年。arXiv:https://doi.org/10.1137/07069328X，doi:10.1137/07069328X。[14] Dorit S. Hochbaum 和 Wolfgang Maass。图像处理与超大规模集成电路中的覆盖和装箱问题近似方案。美国计算机学会会刊，32(1):130–136，1985年1月。doi:10.1145/ 2455.214106。[15] Michael Kapralov。流模型中匹配问题的更优界。载于 Sanjeev Khanna 编，第二十四届美国计算机学会—美国工业与应用数学学会离散算法研讨会论文集，*SODA 2013*，*2013*年*1*月*6*日至*8*日，美国路易斯安那州新奥尔良，第1679–1697页。SIAM，2013年。doi:10.1137/1.9781611973105.121。[16] Sanjeev Khanna、Christian Konrad 和 Cezar-Mihail Alexandru。单遍边到达流模型中的集合覆盖。载于 Floris Geerts、Hung Q. Ngo 和 Stavros Sintos 编，第*42*届美国计算机学会 *SIGMOD-SIGACT-SIGAI* 数据库系统原理研讨会论文集，*PODS 2023*，*2023*年*6*月*18*日至*23*日，美国华盛顿州西雅图，第127–139页。美国计算机学会，2023年。doi:10.1145/3584372.3588678。[17]Morteza Monemizadeh、S. Muthukrishnan、Pan Peng 和 Christian Sohler。可测试的有界度图性质可随机顺序流式处理。载于 Ioannis Chatzigiannakis、Piotr Indyk、Fabian Kuhn 和 Anca Muscholl 编，第*44*届自动机、语言与编程国际研讨会（*ICALP 2017*），第80卷莱布尼茨国际信息学会议录（*LIPIcs*），第131:1–131:14页，德国达格斯图尔，2017年。达格斯图尔城堡—莱布尼茨信息学中心。网址：https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2017.131，doi:10.4230/LIPIcs.ICALP.2017.131。

[18] Anup Rao 和 Amir Yehudayoff。《通信复杂度：及其应用》。剑桥大学出版社，2020 年。
[19] 姚期智。关于分布式计算的一些复杂度问题（初步报告）。载于第十一届年度 *ACM* 计算理论研讨会论文集，第 209–213 页，1979 年。

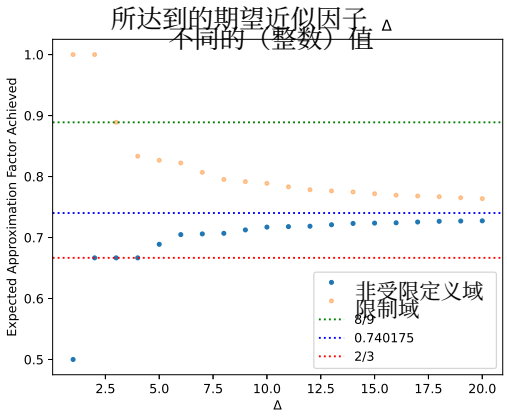


图2：每个整数 $\Delta \leq 20$ 所达到的期望近似因子。

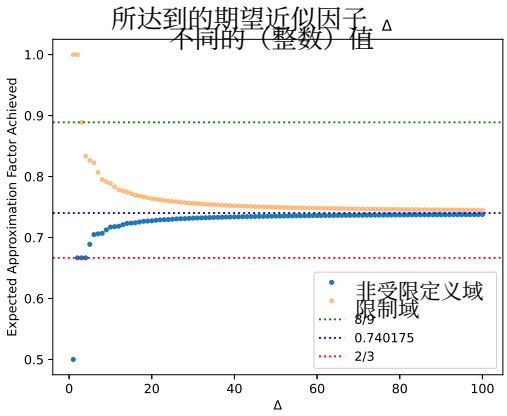


图3：每个整数 $\Delta \leq 100$ 所达到的期望近似因子。

A 不同 Δ 值下达到的期望近似因子

正如定理 1 的证明中所讨论的，我们的算法所达到的近似因子取决于 Δ 的选择。为求完整起见，我们现在讨论取不同（常数）值的 Δ 时所达到的近似因子。

我们通过数值计算来完成这一点，而这可以借助动态规划高效实现。计算结果表明，近似因子似乎是 Δ 的递增函数；当选择 $\Delta = 5,000$ 时，我们得到的期望近似因子大于 ~ 0.7401。有趣的是，任何不小于五的整数 Δ 都足以突破对抗顺序2/3障碍。这也为我们提供了一种方法：通过考察更简单的 $\Delta = 5$ 情形，来“手动证明”一个强于 $2/3$ 的结果。

因此，一个自然而然的后续问题是：通过选择更大的 Δ，还能在多大程度上改善这一结果？为回答这个问题，我们可以先将该算法的期望近似因子（公式 3）分解为两个部分。项 $(\Delta - 1)/\Delta$ 表示通过引理 8 从受限域转到非受限域所产生的“损失”，其余部分则是算法 1 在受限域内达到的近似因子。显然，$(\Delta - 1)/\Delta \leq 1$，因此整个量可以用算法 1 在受限域内达到的近似因子作为上界。

现在，我们可以通过数值计算求出这些值，得到的界为 ~ 0.7401803，对应的窗口长度为 $\Delta = 100,000$。这表明，仅仅通过选取更大的 Δ，我们恐怕无法取得太大的进一步提升。

图2和图3展示了这些思路。在两幅图中，对于不同的 Δ 值，蓝色圆点表示通用算法在非受限域上的期望近似因子（定理 1），橙色圆点表示算法 1 在受限域上达到的近似因子（引理 6）。如前所述，橙色圆点构成蓝色圆点的上界。红色虚线表示对抗顺序设置中的 $2/3$ 界限，蓝色虚线表示数值 0.740175，该值介于我们计算得到的两种近似因子之间。请注意，绘制蓝色虚线是为了辅助直观理解，它并不对应任何理论计算出的数值。绿色虚线表示使用空间 $o(n)$的算法所面临的 $8/9$-近似界限（见定理 2）。