Online Randomness Extraction

Simulating Barely Random Algorithms in the Random Order Arrival Model

Allan Borodin, Chris Karavasilis, David Zhang

September 28, 2025

Overview

1. **Preliminaries**
2. **Motivation**
3. **Extracting Randomness**
4. **Applications**

Randomized Online Algorithms

- Randomness from the choices that the algorithm makes, not from the input sequence, i.e., $\mathbb{E}[ALG(\sigma)]$ is over the random decisions
- Oblivious adversary, i.e., they know the distributions that the algorithm samples but they do not know the outcome of the coinflips
- Input sequence is adversarially-defined, entirely constructed before the algorithm runs
- 1-bit barely random algorithms use only 1 initial, unbiased bit of randomness. The algorithms that we study pick one of two deterministic algorithms to execute with a coin flip, before any inputs arrive
- "If one deterministic algorithm makes a mistake, then the other will do OK. In expectation we are OK"

Deterministic Random-Order Model (ROM) Algorithms

- Adversary specifies a multiset of n items and the input is a uniform random permutation (i.e., Pr[sequence σ] $= \frac{1}{n!}$)
- Randomness from the input sequence, none from the algorithm's choices, i.e., $\mathbb{E}[ALG(\sigma)]$ is over the permutations
- Removes power from the adversary
- Any algorithm that is ρ-competitive in the ROM is ρ-competitive when each input item is drawn independently from the same distribution (i.e., i.i.d.)

Motivation

- Deterministic ROM algorithms achieve better (or at least the same) competitive ratios than randomized online algorithms for all problems that we know
- Can any randomized online algorithm do better than a deterministic ROM algorithm (for any existing or contrived problem)?
- For the 1-bit barely random algorithms that we know of, we demonstrate how to construct a deterministic ROM algorithm

Extracting Randomness

- We define the *bias* $\beta \geq \frac{1}{2}$ of a bit b to mean $\Pr[b = 1] \in [\frac{1}{2}, \beta]$. An unbiased bit is 0 or 1 with equal probability
- Input is stochastic so we should be able to say something about the probability of an event occurring with respect to the order that items arrive in
- Extract a bit $b = 1$ if an event occurs and $b = 0$ otherwise
- If every input item is identical then every permutation is identical – no randomness[1]
- So assume that there are at least two distinct and comparable items
- Assign subscripts to input items in the order that they arrive, $x_1, x_2, \ldots, x_n$. Does the first $x_k \neq x_1$ arrive when k is even or odd?
- $\Pr[k \text{ even}] \in (\frac{1}{2}, \frac{2}{3}]$, i.e., $b = 1$ when k even $\rightarrow \Pr[b = 1] \in (\frac{1}{2}, \frac{2}{3}]$. We extract a bit with a worst-case bias of $\frac{2}{3}$

[1] Identical items trivialize many of the problems that we consider.

Extracting Randomness

- Strong assumption: what if every input item is distinct?
- Then there is a strict ordering on the input
- Does $x_1 < x_2$? $\Pr[x_1 < x_2] = \frac{1}{2}$
- $b = 1$ when $x_1 < x_2 \rightarrow \Pr[b = 1] = \frac{1}{2}$. We extract an unbiased bit
- More generally, we can extract $\frac{n}{2}$ unbiased bits from a sequence of n distinct items

Extracting Randomness

- Combine! Assume that there are at least two distinct items
- If the first two input items are distinct then extract an unbiased bit. Otherwise, wait to see if k is even
- $\Pr[x_1 \neq x_2] \cdot \Pr[x_1 < x_2 | x_1 \neq x_2] + \Pr[x_1 = x_2] \cdot \Pr[k \text{ is even} | x_1 = x_2] \in (\frac{1}{2}, 2 - \sqrt{2}]$
- We extract a bit with a worst-case bias of $2 - \sqrt{2} \approx 0.586$

Binary String Guessing [Böckenhauer et al., 2014]

- Guess an n-bit binary string, one bit at a time. The true value of a bit is revealed after each guess. Goal: maximize #correct guesses
- OPT guesses all n bits correctly
- If we guess the bit that appears more than half the time $\rightarrow$ 2-competitive
- Guess all 0 or all 1 $k+1$ times until we make a mistake. Then extract a bit b and guess it for the rest of the sequence
- $\mathbb{E}[\#\text{correct guesses}] \geq k + (\sqrt{2}-1)(n-k-1) \geq (\sqrt{2}-1)(n-1)$
- $\rightarrow \frac{OPT}{\mathbb{E}[\#\text{correct guesses}]} \leq \frac{n}{(\sqrt{2}-1)(n-1)}$
- $\frac{1}{\sqrt{2}-1} \approx 2.41$-competitive. No deterministic algorithm in the adversarial model is competitive

Knapsack [Han et al., 2015]

- Each item has a weight w_i and value v_i. In the proportional case, $v_i = w_i$. Knapsack has unit capacity
- Knapsack ratios are typically stated as less than 1
- $\frac{1}{2}$-competitive 1-bit barely random algorithms for general knapsack, and proportional knapsack with no revoking
- $\frac{7}{10}$-competitive 1-bit barely random algorithm for proportional knapsack with revoking

General Knapsack

- *MAX* keeps the item with highest value and *GREEDY* keeps the items with highest *value-density* $\frac{v_i}{w_i}$, revoking the least dense items when the knapsack exceeds capacity
- Han et al. run *MAX* or *GREEDY* with probability $\frac{1}{2}$ each
- De-randomization: Pack initial identical items with *GREEDY*. If the bit selects *MAX* then discard all but the max-value item
- We produce either of their two knapsacks with at least $\sqrt{2}-1$ probability each
- $\rightarrow \sqrt{2}-1$-competitive. No deterministic algorithm in the adversarial model is competitive
- Needs revoking (allowed in the original model)

Proportional Knapsack

- Maintain two buckets, each with unit capacity. The algorithm attempts to fill the first bucket and places into the second bucket items too large to fit into the first one.
- Only BIN_1 contains items $\rightarrow BIN_1 = OPT$
- BIN_1 and BIN_2 both contain items $\rightarrow BIN_1 + BIN_2 > 1 \geq OPT$
- De-randomization: similar to previous. Once a bit is extracted, revoke items to match the corresponding knapsack that Han et al. would have packed
- Pick between the two bins each with $\sqrt{2} - 1$ probability $\rightarrow \sqrt{2} - 1$-competitive
- No deterministic algorithm in the adversarial model is competitive
- Needs revoking (**not** used in the original randomized algorithm)

Proportional Knapsack

- $\frac{7}{10}$-competitive algorithm is also 1-bit barely random
- An item i is *small* if $w_i \leq \frac{3}{10}$, *medium* if $\frac{3}{10} < w_i < \frac{7}{10}$, and *large* if $w_i \geq \frac{7}{10}$
- Maintain two buckets, each with unit capacity. One algorithm is aggressive and keeps the heaviest medium items while the other is balanced and keeps lighter medium (and heavier smaller) items, revoking other items as necessary
- Both pack the same identical items, so once a distinct item arrives we extract a bit and continue the corresponding deterministic algorithm
- The only loss in the competitive ratio is from the bias in the bit
- Case-by-case analysis: 0.676-competitive in the ROM. We do best when we pick the aggressive strategy with higher probability
- Needs revoking (allowed in the original model). Beats the tight deterministic bound of $\frac{1}{\phi} \approx 0.618$

Interval Selection [Fung et al., 2014]

- Each interval has a release time r_i, processing time p_i, deadline $d_i = r_i + p_i$, and weight w_i
- We consider real-time problems in the *real-time with random order model*
- $J_i = (r_i, p_{\pi(i)}, r_i + p_{\pi(i)}, w_{\pi(i)})$ for a uniform random permutation π
- Arbitrary weights: Single-length ($p_i = p$ is fixed), Monotone ($r_i < r_j \to d_i \leq d_j$)
- Weights as functions of interval length: C-benevolent (w_i is a convex increasing function of p_i), D-benevolent (w_i is a decreasing function of p_i)
- Problem: We cannot extract randomness from release times... Leads to *pseudo-identical* intervals with the same p_i and w_i

Interval Selection [Fung et al., 2014]

- Single-length algorithm (works in the real-time and online models): Slots $s_1, s_2, s_3 \ldots$ are $[0, p), [p, 2p), [2p, 3p), \ldots$
- No algorithm can schedule two intervals that release in the same slot without conflict
- Fung et al. initially pick even-indexed or odd-indexed slots with $\frac{1}{2}$ probability each and then take the heaviest intervals that release during slots with the chosen parity
- Generalizes to monotone, C-benevolent and D-benevolent instances but slots must be redefined adaptively online
- De-randomization: Schedule a newly-released pseudo-identical interval if it does not conflict. When a distinct interval arrives, extract a bit and define slots from the current time forward
- Fung et al. are 2-competitive in all instances – we are $\frac{1}{\sqrt{2}-1} \approx 2.41$-competitive in all instances

Throughput

- Each job has a release time r_i, processing time p_i, deadline d_i, and weight w_i. Define slack $s_i = d_i - r_i - p_i$
- Real time with random-order: $J_i = (r_i, p_{\pi(i)}, s_{\pi(i)}, w_{\pi(i)})$ for a uniform random permutation π
- [Kalyanasundaram and Pruhs, 2003]: Unweighted single-processor throughput: > 100000-competitive 1-bit barely random algorithm (no det. alg is competitive)
  - Both deterministic algorithms behave identically on pseudo-identical intervals $\rightarrow$ $O(1)$-competitive de-randomization by choosing between them once a distinct interval arrives
- [Chrobak et al., 2007]: Unweighted single-processor single-length throughput: $\frac{5}{3}$-competitive 1-bit barely random algorithm (det. lower bound of 2)
  - De-randomize by greedily taking pseudo-identical intervals and beginning their (synchronized) deterministic algorithms once a distinct interval releases. Fair bit $\rightarrow \frac{5}{3}$-competitive, (worst case) biased bit $\rightarrow 1.77$-competitive

Job Shop [Kimbrel and Saia, 2000]

- Scheduling on 2 machines. Goal: minimize makespan
- Each job has some number of operations. Only the first operation is known initially and a subsequent operation is revealed on the completion of its predecessor
- Processing times per operation are not known until they complete
- If a job has operations $o_1, o_2, o_3, \ldots$, odd-indexed operations are requests to machine 1 and even-indexed operations are requests to machine 2; o_1 can be a dummy entry if the job requests machine 2 first
- An operation may be preempted and resumed later
- Assign priorities to jobs arbitrarily. One deterministic algorithm runs jobs in priority order on machine 1 and in the reverse order on machine 2. The other deterministic algorithm runs jobs in the reverse order on machine 1 and in priority order on machine 2
- De-randomization is difficult because we need to reverse priorities on the machines

Makespan [Albers, 2002]

- Each job has a processing requirement p_j, jobs revealed sequentially
- 1-bit barely random is 1.916-competitive by picking between two strategies – one aggressive (keeping more machines lightly loaded) and one passive (distributing jobs more evenly)
- Strategies behave very differently on identical jobs (simulations suggest that $\Omega(m)$ job relocations are required to move from one state to the other)
- De-randomization: Competitive ratio of at least 2 if we uniformly spread identical jobs before we can extract a bit (using no recourse)
- 1.8478-competitive deterministic ROM algorithm due to [Albers and Janke, 2021]

References I


Albers, S. (2002).
On randomized online scheduling.
In Reif, J. H., editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 134–143. ACM.


Albers, S. and Janke, M. (2021).
Scheduling in the random-order model.
Algorithmica, 83(9):2803–2832.


Böckenhauer, H., Hromkovic, J., Komm, D., Krug, S., Smula, J., and Sprock, A. (2014).
The string guessing problem as a method to prove lower bounds on the advice complexity.
Theor. Comput. Sci., 554:95–108.


Chrobak, M., Jawor, W., Sgall, J., and Tichỳ, T. (2007).
Online scheduling of equal-length jobs: Randomization and restarts help.
SIAM Journal on Computing, 36(6):1709–1728.

References II


Fung, S. P., Poon, C. K., and Zheng, F. (2014).
Improved randomized online scheduling of intervals and jobs.
Theory of Computing Systems, 55(1):202–228.


Han, X., Kawase, Y., and Makino, K. (2015).
Randomized algorithms for online knapsack problems.
Theoretical Computer Science, 562:395–405.


Kalyanasundaram, B. and Pruhs, K. R. (2003).
Maximizing job completions online.
Journal of Algorithms, 49(1):63–85.


Kimbrel, T. and Saia, J. (2000).
On-line and off-line preemptive two-machine job shop scheduling.
Journal of Scheduling, 3:335–364.