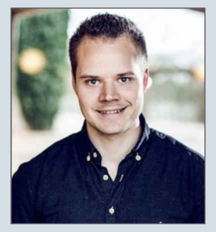
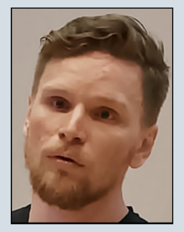
MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS


Kasper Green Larsen
Aarhus University


Morten Nygaard
Aarhus University


Kasper Eenberg
Aarhus University

Danmarks Grundforskningsfond
Danish National Research Foundation

Reservoir Sampling in a Non-Streaming Environment

What is Sampling?

Given a set S of n weights $\{w_1,\ldots,w_n\}$, select a subset Δ of S such that $\Pr[i \in \Delta] \propto w_i$, and each unique index is only sampled once per sample. Sampling generally splits into three different categories.

Solitaire The size of the sample set Δ is 1, thus, an element is sampled with probability $\frac{w_i}{\sum_S w_j}$.

Subset The size of the sample set Δ is not fixed, instead each element is sampled with probability w_i. This requires that w_i is a probability, meaning that $w_i \in [0,1]$.

Reservoir The output must be of size k, where $1 < k \leq n$.
Thus the probability of an index i to be in Δ must be calculated recursively. Here the probability of including i in an iteration, assuming that $i \notin \Delta$, is $\frac{w_i}{\sum_{S \setminus \Delta} w_j}$.

Motivation

- Sampling is used in the modelling of biological systems, Monte Carlo simulations and machine learning.
- Traditional methods have drawbacks as unpredictable running time or can be made faster.

Contribution

We present an algorithm that solves the problem of reservoir sampling in a non-streaming setting where all input weights are known beforehand and fit in memory.

Our presentation of this algorithm is accompanied by a thorough investigation of the traditional methods of doing reservoir sampling, as well comparisions with the algorithm we present.

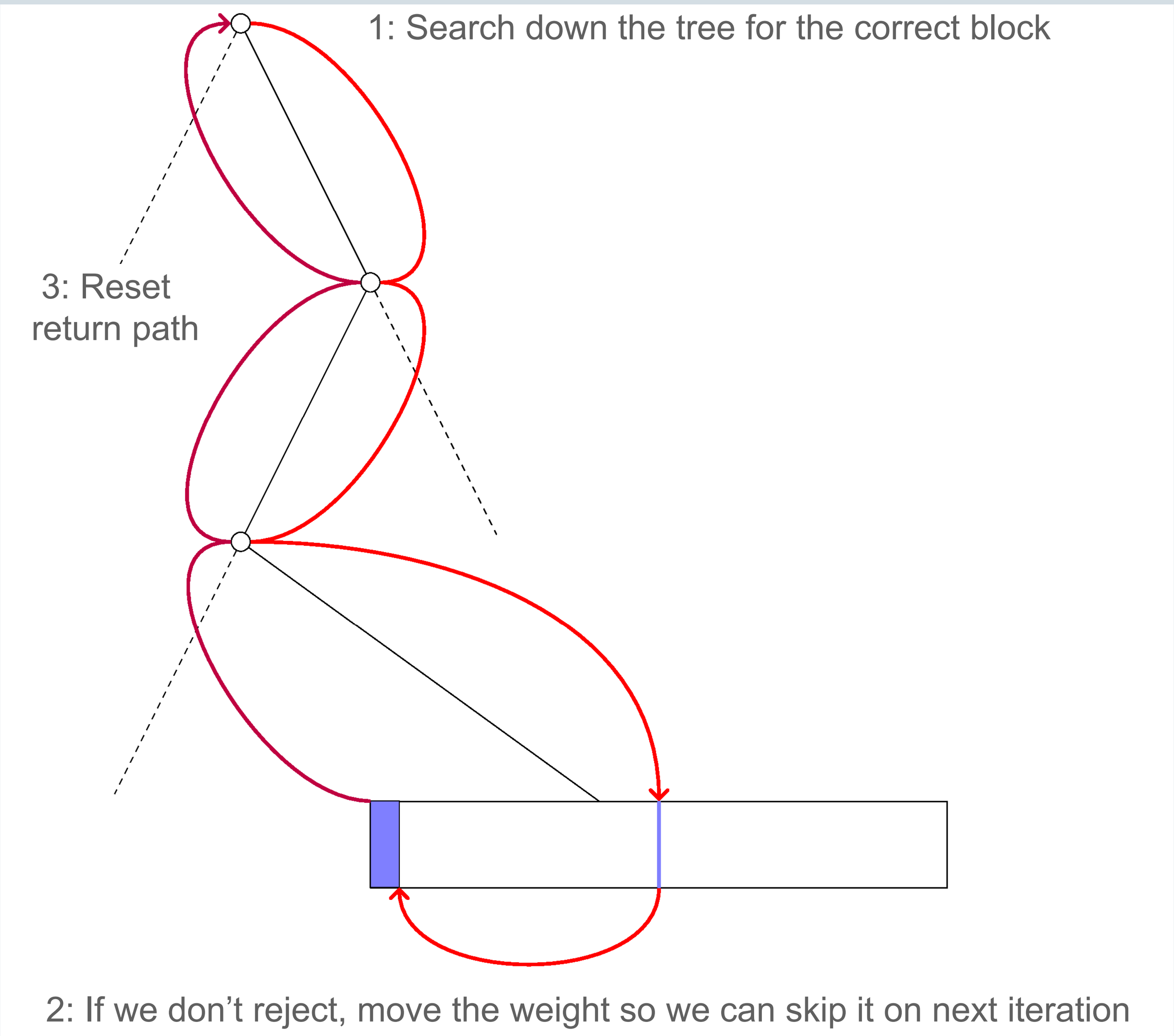
Proposed Algorithm

Preprocessing

- Normalize all the weights such that $\sum_S w_i = 1$.
- Sort indexes into buckets such that, with the exception of the last bucket, the elements in each bucket are within a factor two of each other.
- Build a tree on top of the buckets, each node containing the sum of the weights in the buckets below it.

Sampling Items are sampled with weight of maximal element in each bucket, let m be the maximal weight in the bucket.

- For $1,\ldots,k$, select a uniformly random number, and follow the path down the tree, this corresponds to an index in the bucket. Reject the selection with probability $1 - \frac{w_i}{m}$.
  If we selected the index, update the path up the tree to reflect that an element was removed.
- Reset the entire tree to an equivalent state when a sample has finished.



Sample iteration

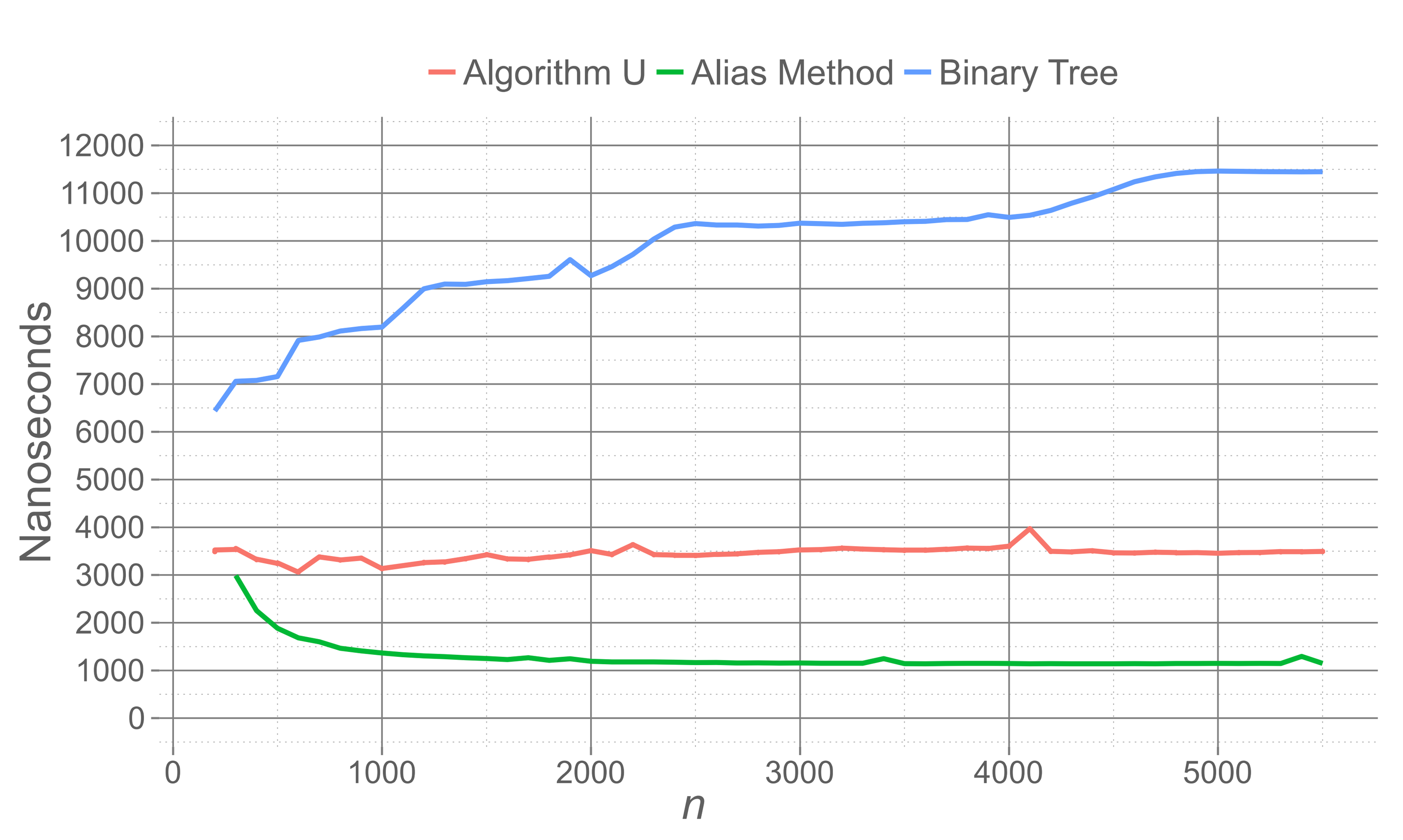
Traditional Methods

Some traditional approaches to solve the reservoir sampling problem have been:

- Using a method for solitaire sampling, swapping selected indexes out after individual draw.
  $O(k \cdot n)$ query time, k random number generations.
- Using a method for solitaire sampling (e.g. the Alias method), rejecting indices already selected.
  Uses k random numbers, sampling time heavily dependent on distribution of data, but $\Omega(k)$.
- Building a binary tree on top of the weights, updating the internal nodes of the tree as elements are selected, like in the proposed algorithm, but without blocking.
  Sampling time $O(k \cdot \log n)$, k random numbers used.
- In the streaming setting, methods exist using $O(n)$ sampling time, and $O(n)$ or $O(k \cdot \log \frac{n}{k})$ random numbers.

Experiments

The figure shows the sampling time for 100 indices (k = 100) with uniformly random weights.



These results are promising as the Alias method slows down significantly for non-uniform data or when $k \to n$.