CHAPTER 22

WEIGHTED RESERVOIR SAMPLING: RANDOMLY SAMPLING STREAMS

Chris Wyman
NVIDIA

ABSTRACT

Reservoir sampling is a family of algorithms that, given a stream of N elements, randomly select a K-element subset in a single pass. Usually, K is defined as a small constant, but N need not be known in advance.

22.1 INTRODUCTION

We describe *weighted reservoir sampling*, a well-studied algorithm [2, 8] largely unknown to the graphics community. It efficiently samples random elements from a data stream. As real-time ray tracing becomes ubiquitous, GPUs may shift from processing streams of rays or triangles toward streaming random samples, paths, or other stochastic data. In that case, weighted reservoir sampling becomes a vital tool.

After decades of research, reservoir sampling variants exist that optimize different, sometimes conflicting, properties. Input elements can have uniform or nonuniform weights. Outputs can be chosen with or without replacement (i.e., potential duplicates). With various assumptions, we can fast-forward the stream for sublinear running time, minimize the required random entropy, choose elements with greater than 100% probability, or achieve the optimal running time of $O(K + K \log(N/K))$.

Complex variants are beyond the scope of this chapter, but realize that reservoir sampling provides a flexible menu of options depending on your required properties. This rich history exists because many domains process streaming data. Consider managing modern internet traffic, where most participants lack resources to store more than a tiny percent of the stream. Or, for a more historical application, in the 1980s input data was often streamed sequentially off reel-to-reel tape drives.

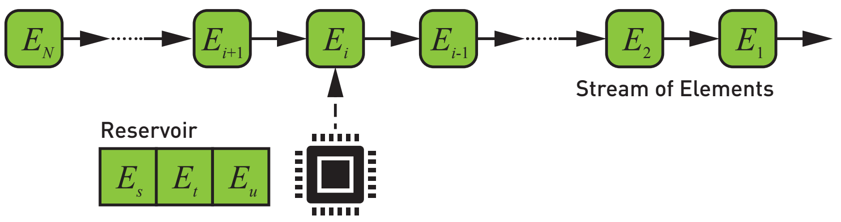

Figure 22-1. *Weighted reservoir sampling processes a stream of elements E_i and incrementally selects a subset weighted proportional to a provided set of weights w_i. Algorithmic variants can sample with or without replacement (i.e., if we guarantee $s \neq t$), can minimize the required random entropy, and can even skip processing all N elements.*

22.2 USAGE IN COMPUTER GRAPHICS

Reservoir sampling has been repeatedly rediscovered in real-time graphics. For instance, it underlies single-pass variations of stochastic transparency [5, 10], which selects random subsets of fragments from a stream of translucent triangles. Lin and Yuksel [6] use a simple variant to trace shadow rays with their desired distribution. And Bitterli et al. [1] use reservoir sampling to stream statistics for spatiotemporal importance resampling.

22.3 PROBLEM DESCRIPTION

Imagine a stream of elements E_i (for $1 \leq i \leq N$), as in Figure 22-1. At a given time, a streaming algorithm processes element i. Elements $j < i$ were previously considered and no longer reside in memory, unless copied to local temporaries. Elements $j > i$ have not yet reached the processor.

Weighted reservoir sampling incrementally builds a reservoir $\mathcal{R}$. This is a randomly chosen K-element subset of previously seen elements (i.e., E_1 to E_{i-1}). For each stream element, E_i is either discarded or inserted into $\mathcal{R}$, potentially replacing existing entries. Replaced samples are forgotten, as if they had never been in the reservoir.

Generally, if E_j has a selection weight w_j, then (for $j < i$) the chance $E_j \in \mathcal{R}$ is proportional to its relative weight $w_j / \sum_{k<i} w_k$. After finishing a stream with total weight $W = \sum_{k \leq N} w_k$, the probability $E_j \in \mathcal{R}$ is proportional to w_j/W. If $K = 1$, the probability is exactly w_j/W.

22.4 RESERVOIR SAMPLING WITH OR WITHOUT REPLACEMENT

For rendering, we generally want independent and identically distributed (abbreviated i.i.d.) samples to ensure unbiasedness. In reservoir sampling, this means that the chance of selecting E_t (in Figure 22-1) cannot vary based on E_s; if we enforce $s \neq t$, they may no longer be independent. In the reservoir sampling terminology, sampling with replacement clearly gives independence, as any input may occur multiple times in the reservoir.

Fortunately, reservoir sampling with replacement is easier. Delving into the theory literature, most papers thus optimize reservoir sampling without replacement. We leave exploring this literature to the interested reader.

When sampling with replacement, we first focus on understanding the algorithm with reservoir size $K = 1$. Using replacement, samples are i.i.d., so those given by running the algorithm three times with $K = 1$ are distributed identically to those produced by running it once with $K = 3$.

22.5 SIMPLE ALGORITHM FOR SAMPLING WITH REPLACEMENT

For $K = 1$, the reservoir is a tuple $\mathcal{R} = \{R, w_s\}$ containing the currently selected element R and the sum of weights for all previously seen elements $w_s = \sum_{k<i} w_k$. It gets initialized to $\{\emptyset, 0\}$ before processing the stream.

Then, for every stream element E_i with weight $w_i \geq 0$, the reservoir is updated as follows, given a uniform random variate ξ:

```
update($E_i$,$w_i$)
    $w_s \leftarrow w_s + w_i$
    $\xi \leftarrow$rand()$\in [0...1]$
    if ($\xi < w_i/w_s$)
        $R \leftarrow E_i$
```

When $w_i = 0$, this should leave the reservoir unmodified. This happens naturally, due to IEEE-754 NaN (not a number) behavior, but explicitly checking may be needed to guarantee this behavior for $w_1 = 0$ on non-conformant hardware.

That this update algorithm selects E_i with probability $w_i/\sum_{k\leq N} w_k$ after processing N elements can be easily shown by induction. If $N = 1$, we select E_1 with probability $w_1/w_1 = 1$ (or with zero probability if $w_1 = 0$).

Before processing element E_i at step i, assume that the reservoir contains $\{E_j, \sum_{k<i} w_k\}$, where E_j has been selected with probability $w_j/\sum_{k<i} w_k$. By design, the `update()` function selects E_i with probability

$$\frac{w_i}{w_i + \sum_{k<i} w_k} = \frac{w_i}{\sum_{k\leq i} w_k}. \tag{22.1}$$

Alternatively, it leaves the prior selection, E_j, in the reservoir with probability

$$1 - \frac{w_i}{w_i + \sum_{k<i} w_k} = \frac{\left(\sum_{k\leq i} w_k\right) - w_i}{\sum_{k\leq i} w_k} = \frac{\sum_{k<i} w_k}{\sum_{k\leq i} w_k}. \tag{22.2}$$

As E_j was previously selected with probability $w_j / \sum_{k<i} w_k$, its final weighting is

$$\left(\frac{w_j}{\sum_{k<i} w_k}\right)\left(\frac{\sum_{k<i} w_k}{\sum_{k\leq i} w_k}\right) = \frac{w_j}{\sum_{k\leq i} w_k}. \tag{22.3}$$

That leaves either sample E_i or E_j in the reservoir with the desired probability.

22.6 WEIGHTED RESERVOIR SAMPLING FOR $K > 1$

Extending the algorithm from the previous section for a reservoir with more than one entry (with replacement) is very straightforward. Now the reservoir is $\{\{R_1, \ldots, R_K\}, w_s\}$ and gets initialized to $\{\{\emptyset, \ldots, \emptyset\}, 0\}$. The stream update becomes:

```
update($E_i$, $w_i$)
    $w_s \leftarrow w_s + w_i$
    for ($k \in 1...K$)
        $\xi_k \leftarrow$rand() $\in [0...1)$
        if ($\xi_k < w_i/w_s$)
            $R_k \leftarrow E_i$
```

22.7 AN INTERESTING PROPERTY

A key property of reservoir sampling is that one can combine multiple independent reservoirs without reprocessing their input streams. This is vital in some streaming contexts where it is impossible to replay the streams for a second look.

To combine two reservoirs $\{R_1, w_1\}$ and $\{R_2, w_2\}$, you get $\{R, w_1 + w_2\}$, where $R = R_1$ with probability $w_1/(w_1 + w_2)$; otherwise $R = R_2$.

22.8 ADDITIONAL READING

The Wikipedia page for reservoir sampling [9] is a fine starting point for further reading. The algorithm given here is a simplified "A-Chao" [2] from Wikipedia. Other variants avoid storing the sum w_s, but are less intuitive. Because renderers often need sum-of-weights normalization, storing this sum seems useful. However, the variants that exponentiate random variates may prove useful for volumetric transport.

Much prior research on reservoir sampling has occurred in biostatistics (e.g., Chao [2]) leaving them unaccessible behind paywalls and containing domain-specific jargon. We found Tillé [7] to be a good statistics reference covering reservoir sampling. We highly recommend the work of Efraimidis and Spirakis, which includes several comparisons and surveys of reservoir variants [3, 4] and is largely comprehensible to non-statisticians.

REFERENCES

[1] Bitterli, B., Wyman, C., Pharr, M., Shirley, P., Lefohn, A., and Jarosz, W. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics*, 39(4):148:1–148:17, 2020. DOI: 10.1145/3386569.3392481.

[2] Chao, M. T. A general purpose unequal probability sampling plan. *Biometrika*, 69(3):653–656, 1982. DOI: 10.2307/2336002.

[3] Efraimidis, P. Weighted random sampling over data streams. *Computing Research Repository (CoRR)*, arXiv, https://arxiv.org/abs/1012.0256, 2010.

[4] Efraimidis, P. and Spirakis, P. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(3):181–185, 2006. DOI: 10.1016/j.ipl.2005.11.003.

[5] Enderton, E., Sintorn, E., Shirley, P., and Lubke, D. Stochastic transparency. In *Symposium on Interactive 3D Graphics and Games*, pages 157–164, 2010. DOI: 10.1145/1730804.1730830.

[6] Lin, D. and Yuksel, C. Real-time rendering with lighting grid hierarchy. In *Symposium on Interactive 3D Graphics and Games*, 8:1–8:10, 2019. DOI: 10.1145/3321361.

[7] Tillé, Y. *Sampling Algorithms*. Springer-Verlog, 2006. DOI: 10.1007/0-387-34240-0.

[8] Vitter, J. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985. DOI: 10.1145/3147.3165.

[9] Wikipedia. Reservoir sampling. https://en.wikipedia.org/wiki/Reservoir_sampling. Accessed January 6, 2021.

[10] Wyman, C. Exploring and expanding the continuum of oit algorithms. In *High Performance Graphics*, pages 1–11, 2016. DOI: 10.2312/hpg.20161187.