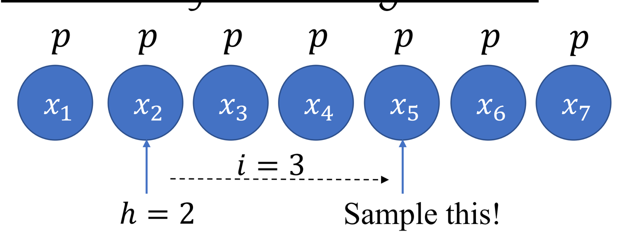
Optimal Dynamic Subset Sampling: Theory and Applications

Lu Yi, Hanzhi Wang, Zhewei Wei Contact: zhewei@ruc.edu.cn

RENMIN UNIVERSITY OF CHINA

Technique 1: Group Partition

Let's start from a simple case!



- The index of the first sample: $i \sim p(1-p)^{i-1}$
- The geometric distribution is memoryless
- The query time = # of the sampled events

Try a more complicated case!

- $2^{-j} < p(x_i) \le 2^{-j+1}$
- Let $p = 2^{-j+1}$ be the upper bound
- First sample each event with p as a candidate, then accept it with $\frac{p(x_i)}{p}$
- Each event is sampled with probability $p \cdot \frac{p(x_i)}{p} = p(x_i)$
- The expect number of candidates $= np \le 2\mu \rightarrow$ It costs $O(1+\mu)$ time

Why Group Partition?

GeoSS

Step 0. Let p be the upper bound

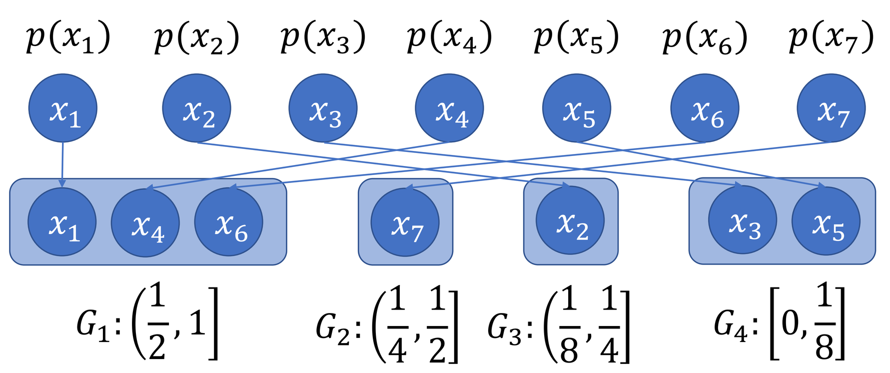
Step 1. Currently at the h-th event, $h = 0$ initially

Step 2. Generate $i \sim p(1-p)^{i-1}$

Step 3. The next candidate: $(i+h)$-th event, accept it with $\frac{p(x_i)}{p}$

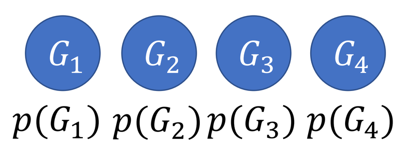
Step 4. $h = i + h$, repeat Step 2 to 4 until $h > n$

domain set S



- Create $(\lceil \log n \rceil + 1)$ groups: $G_1, G_2, \dots, G_K (K = \lceil \log n \rceil + 1)$
- $G_j = \{x_i | 2^{-j} < p(x_i) \le 2^{-j+1}\}$, $1 \le j \le K-1$
- $G_j = \{x_i | p(x_i) \le 2^{-j+1}\}, j = K$
- Use *GeoSS* within each group
- Totally costs $O(1 + \mu + \log n)$ time

How to $O(1+\mu+\log n) \rightarrow O(1+\mu)$?



- **Only sample the groups with at least one candidate!**
- The probability that G_j contains at least one candidate:

$$p(G_j) = 1 - (1 - 2^{-j+1})^{|G_j|}$$

- First sample among the groups with $p(G_j)$, then sample within the sampled groups

```
Algorithm 1: SampleWithinGroup
Input: a group G_k
Output: a drawn sample T
n_k ← |G_k|, T ← ∅, h ← 0;
Let G_k[i] be the i-th element of G_k;
Generate a random r s.t. Pr[r = j] = 2^{-k+1}(1-2^{-k+1})^{j-1} / p(G_k), j ∈ {1,...,n_k};
while r + h ≤ n_k do
    h ← r + h;
    if rand() < p(G_k[h])/2^{-k+1} then
        T ← T ∪ {G_k[h]};
    Generate a random r ~ Geo(2^{-k+1});
return T
```

Tips for updates

Example: **inserting an event x**

S0. Insert x to a group $G_k^{(0)}$ based on $p(x)$

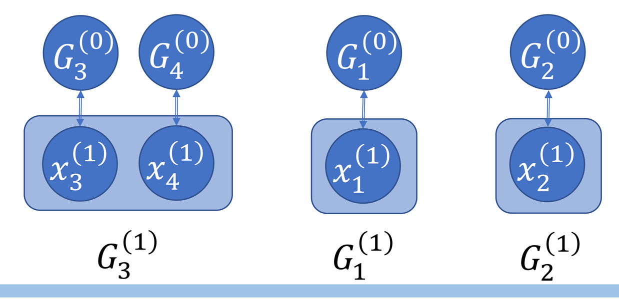
S1. Recalculate the prob. $p(G_k^{(0)})$

S2. Transfer $x_k^{(1)}$ from one group to another according to $p(x_k^{(1)})$ (also $p(G_k^{(0)})$)

S3. Recalculate the prob. of the modified groups at **S2**

How to sample among the groups?

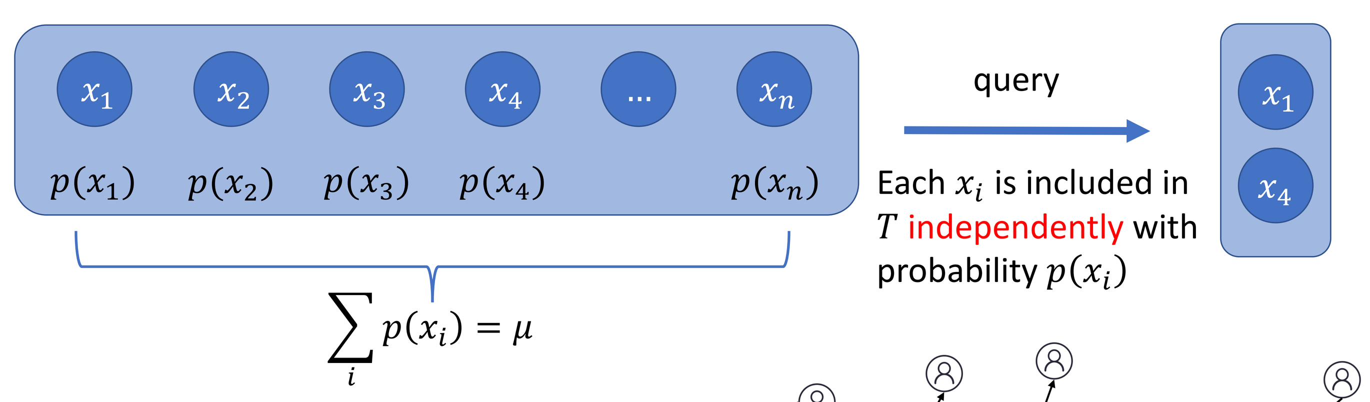
- **Partition again!**
- We add level index to distinguish various subset sampling problems
- Use **Technique 2** to sample the groups at level 1, only $O(\log \log n)$ events



Overview

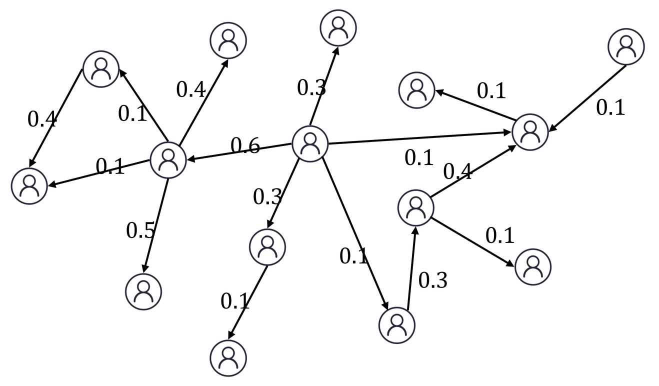
Subset Sampling Problem

- Given a set of n distinct events $S = \{x_1, \dots, x_n\}$, in which each event x_i has an associated probability $p(x_i)$, a query for the subset sampling problem returns a subset $T \subseteq S$, such that every x_i is independently included in T with probability $p(x_i)$.



Dynamic Subset Sampling Problem

- Insert an event
- Delete an event
- Modify the probability of an event



Influence spreading under the IC (Independent Cascade) model

Contributions

- ✓ Optimal query time: $O(1+\mu)$
- ✓ Optimal update time: $O(1)$
- ✓ Great experimental performance
- ✓ Empirical study on Influence Maximization

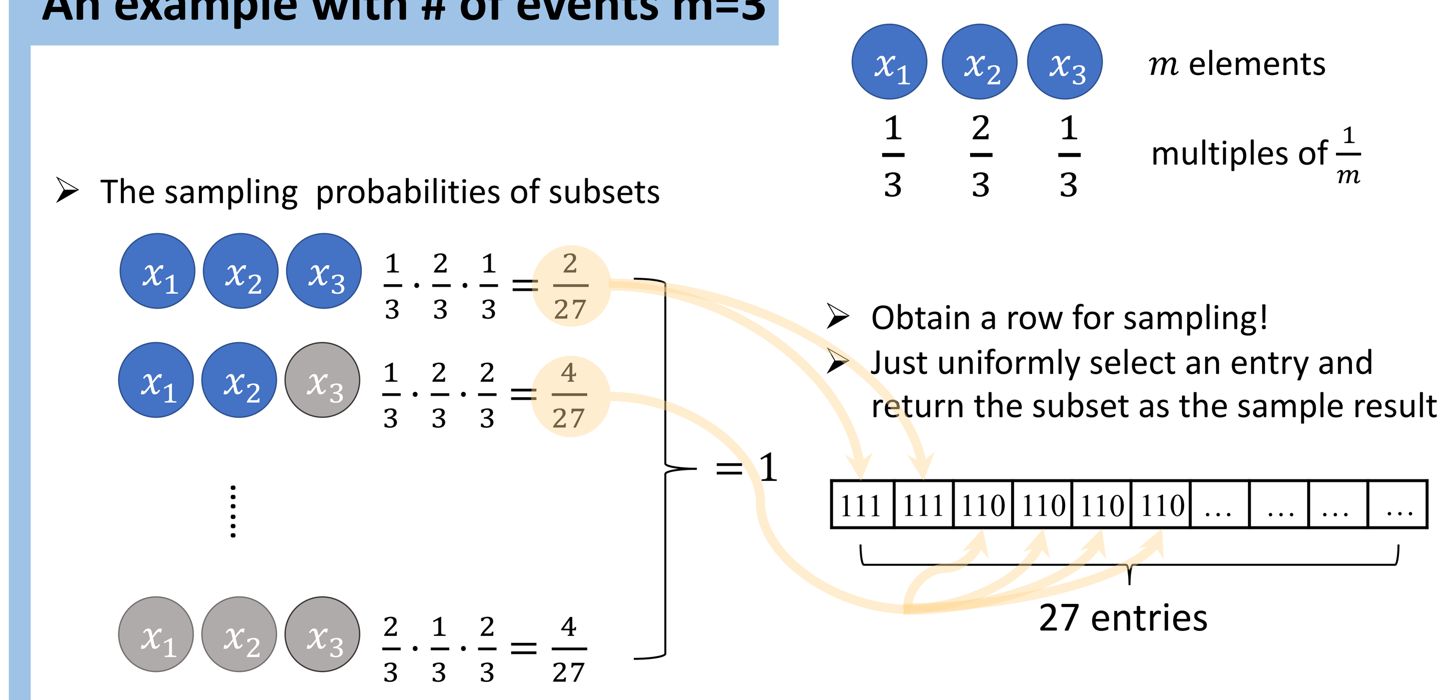
Applications

- Dynamic Influence Maximization
- Approximate Graph Propagation
- Computational Epidemiology
- Fractional (bipartite) matching

Technique 2: Table Lookup

Sample each element independently ⇔ Sample one subset

An example with # of events m=3

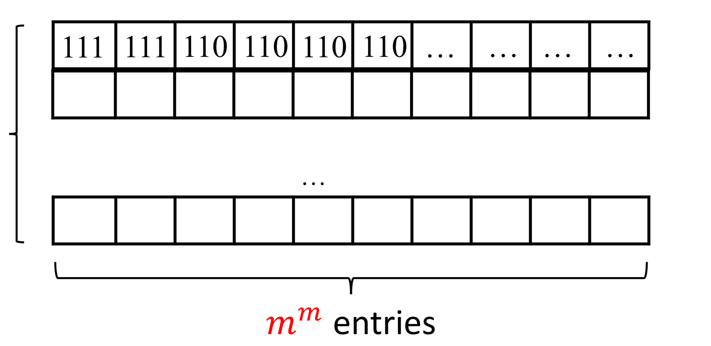


Deal with non-multiples of $1/m$

- Fill the row with respect to $\bar{p}(x_i) = \frac{\lceil mp(x_i) \rceil}{m} \in \{\frac{1}{m}, \dots, \frac{m}{m}\}$
- Accept the event with $\bar{p}(x_i)/p(x_i)$

Deal with dynamic probabilities

- Create a lookup row for each possible distribution
- Store the current row index $A(\bar{p}(x_1), \dots, \bar{p}(x_m)) = \sum_{i=1}^{m} (m\bar{p}(x_i) - 1) m^{i-1}$
- m must be small enough!



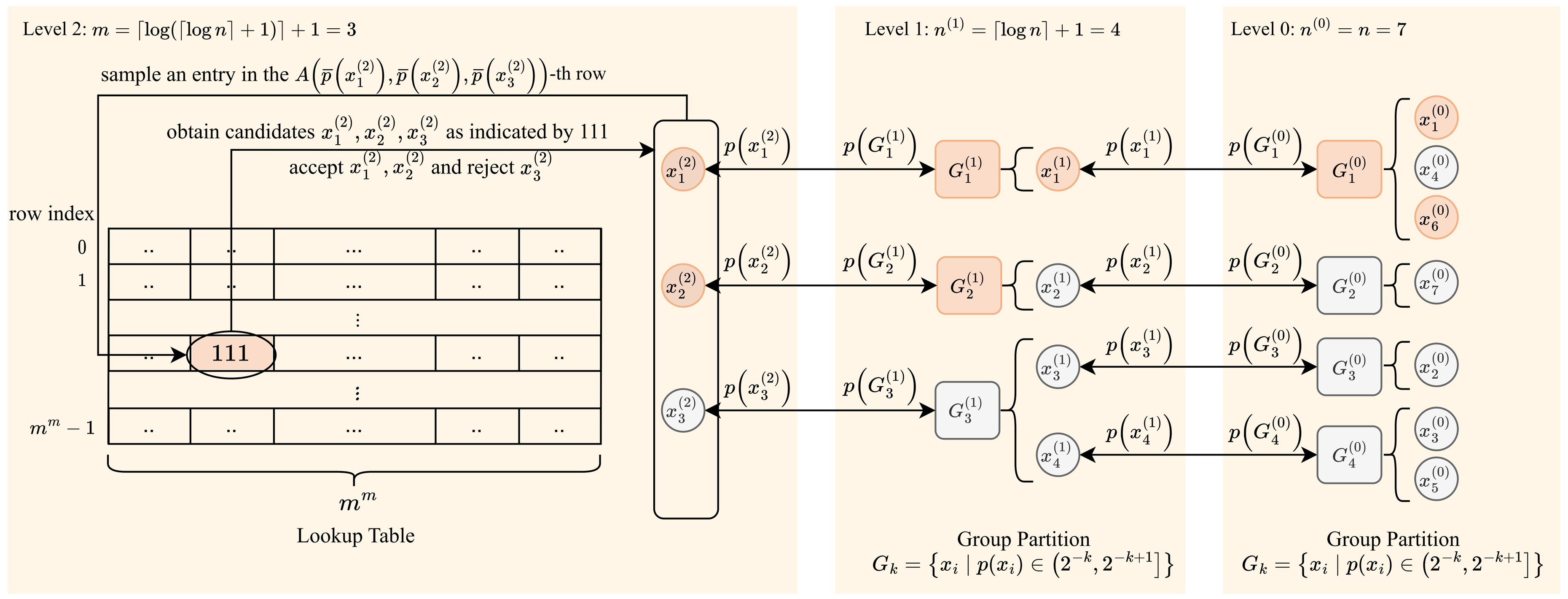
Tips for updates

Suppose: $\bar{p}(x_i) \rightarrow \bar{p}'(x_i), \bar{p}(x_j) \rightarrow \bar{p}'(x_j)$

The new row index:

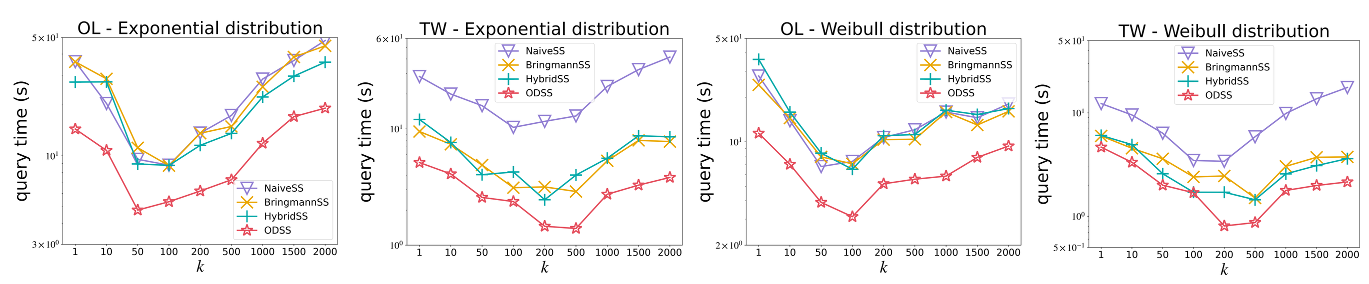
$$A'(\bar{p}(x_1), \dots, \bar{p}(x_m)) = A(\bar{p}(x_1), \dots, \bar{p}(x_m)) + (m\bar{p}'(x_i) - m\bar{p}(x_i))m^{i-1} + (m\bar{p}'(x_j) - m\bar{p}(x_j))m^{j-1}$$

General Framework

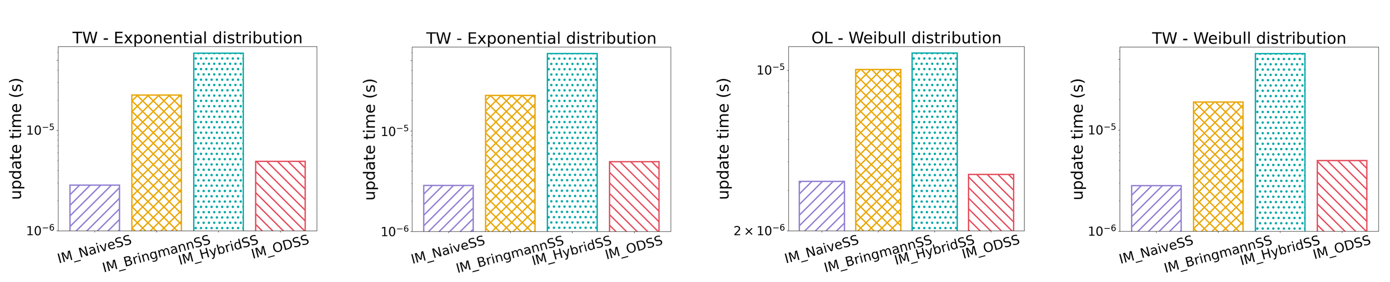


Empirical Study on Influence Maximization

- Based on the framework OPIM-C[ICMD'18], replace the subset sampling module with various dynamic subset sampling structures and thus obtain a new dynamic IM algorithm for the fully dynamic model.
- No algorithms can achieve any meaningful approximation guarantee in the fully dynamic network model. That is, re-running an IM algorithm upon each update can achieve the lower bound of the running time.



- Running time of dynamic IM algorithms based on various subset sampling structures.



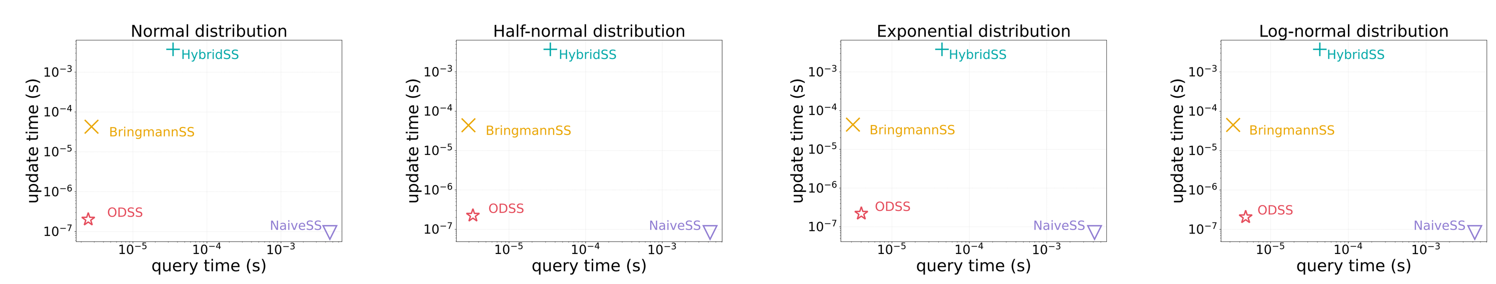
- Update time of dynamic IM algorithms based on various subset sampling structures.

Experiments

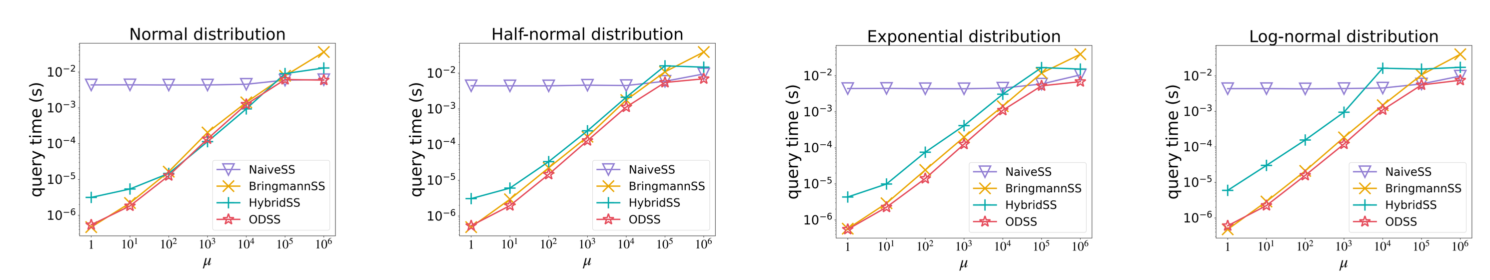
- Competitors

Algorithm	Expected Query Time	Update Time
The Naive Method	$O(n)$	$O(1)$
HybridSS[COCOON'10]	$O\left(1 + n\sqrt{\min\{\bar{p}, 1-\bar{p}\}}\right)$	$O(n)$
BringmannSS[ICALP'12]	$O(1+\mu)$	$O(\log^2 n)$
ODSS (Ours)	$O(1+\mu)$	$O(1)$

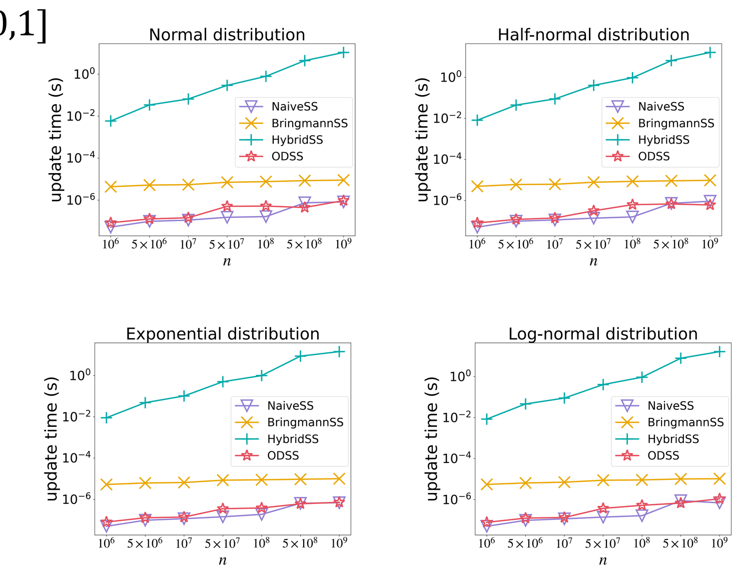
- Distributions of probabilities
  - Normal distribution (skewness as 0)
  - Half-normal distribution (skewness below 1)
  - Exponential distribution (skewness as 2)
  - Log-normal distribution (skewness as 4)
  - Re-scale the range of the random number into [0,1]



- Trade-off between query time and update time. $(n = 10^5, \mu = 1)$



- Varying μ: query time (s) on distributions with different skewnesses. $(n = 10^6)$



- Varying n: update time (s) on distributions with different skewnesses.