Temporal Biased Streaming Submodular Optimization

Junzhou Zhao   Pinghui Wang   Chao Deng   Jing Tao

Xi’an Jiaotong University

Data Streams

- Big data:
  - online social networks
  - Internet of things
  - mobile devices
  - ...

- Data streams: "3V" challenges
  - Volume: in TBs even PBs
  - Velocity: K/s (≈9500 tweets/sec)
  - Variety: texts, images, video, numerical data, etc.

information overload

Data Stream Summarization

- **Goal**: use a carefully chosen subset of items to represent the stream at any time point
- **Challenges**:
  - data items arrives at a very fast speed
  - each data item can only be visited once
  - random access to the entire data is not allowed
  - only a small fraction of the data can be loaded into memory

surveillance camera

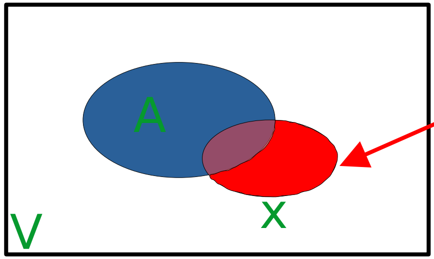
summary

Submodularity

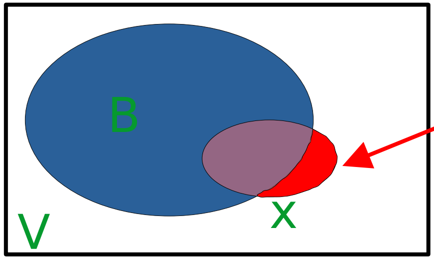
- Submodularity is a natural model for
  - representativeness, informativeness, diversity, coverage
- A set function $f: 2^V \mapsto \mathbb{R}_{\geq 0}$ is submodular if

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$$

for all $A \subseteq B \subseteq V$ and $x \in V \setminus B$.



gain of adding
x to A



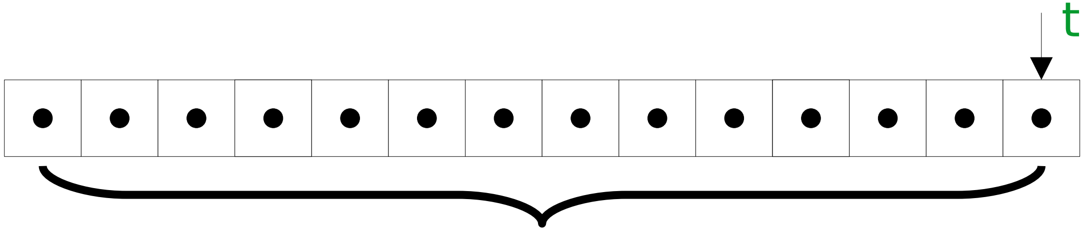
gain of adding
x to B⊇A

smaller gain!

- Captures the diminishing returns property

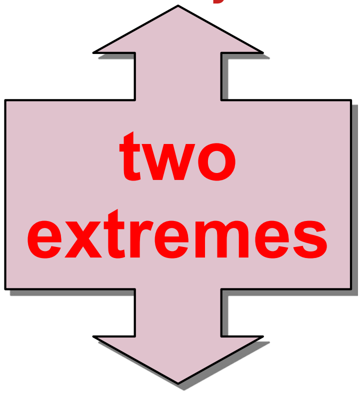
Streaming Submodular Optimization (SSO)

- SSO for insertion-only streams [KDD’14]

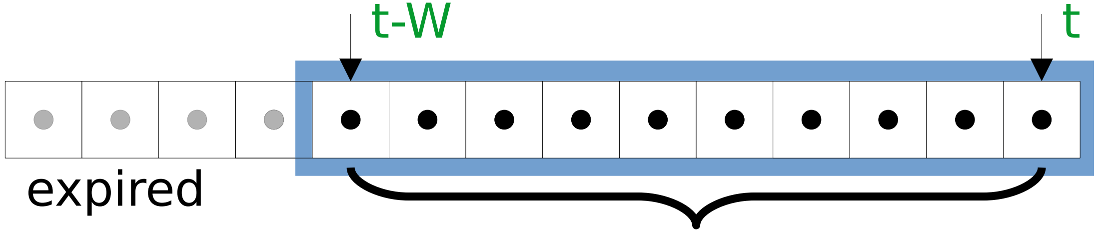


all items participate in the analysis

cons:
- all items are treated equally regardless of how outdated they are



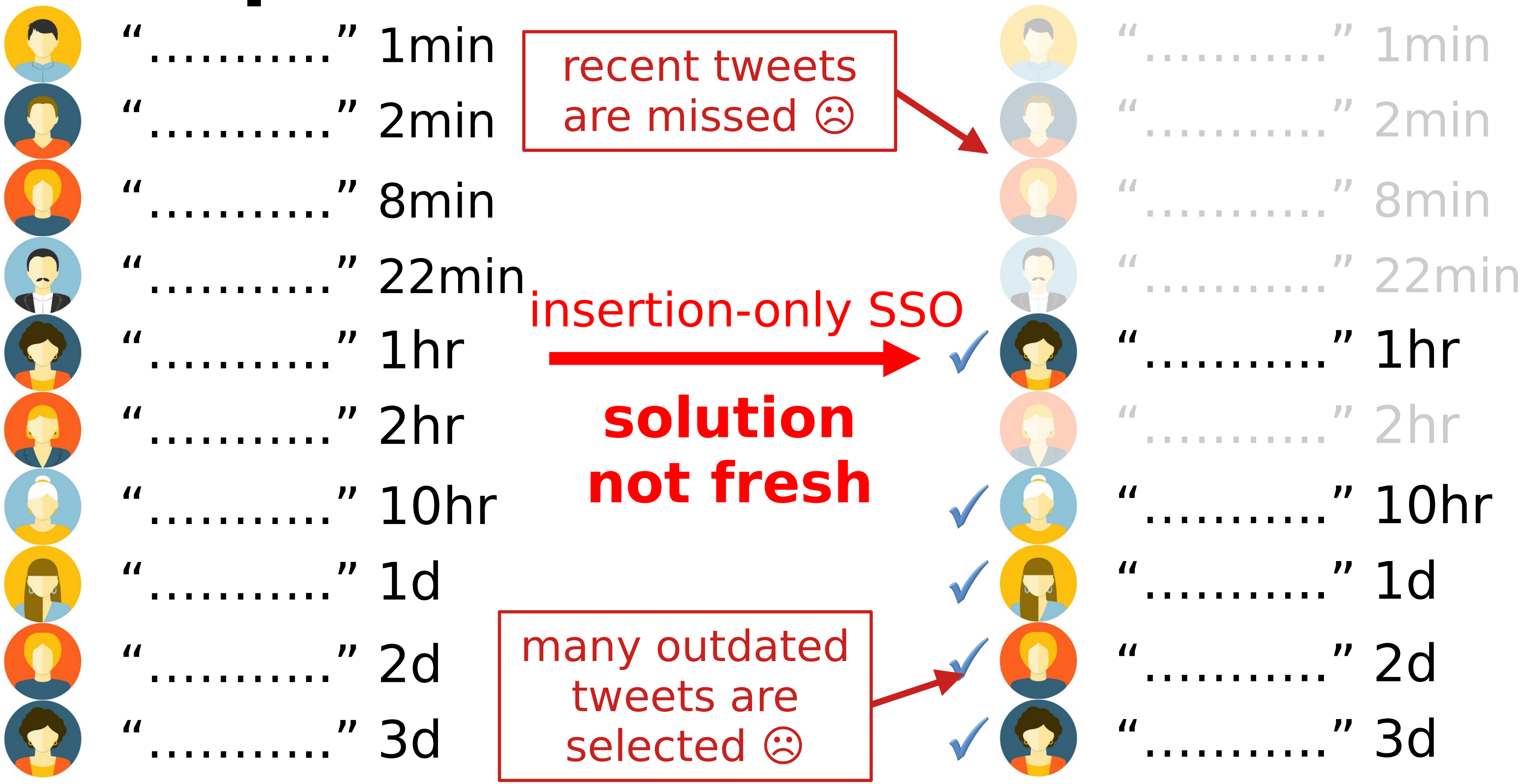
- SSO for sliding-window streams [WWW’17]



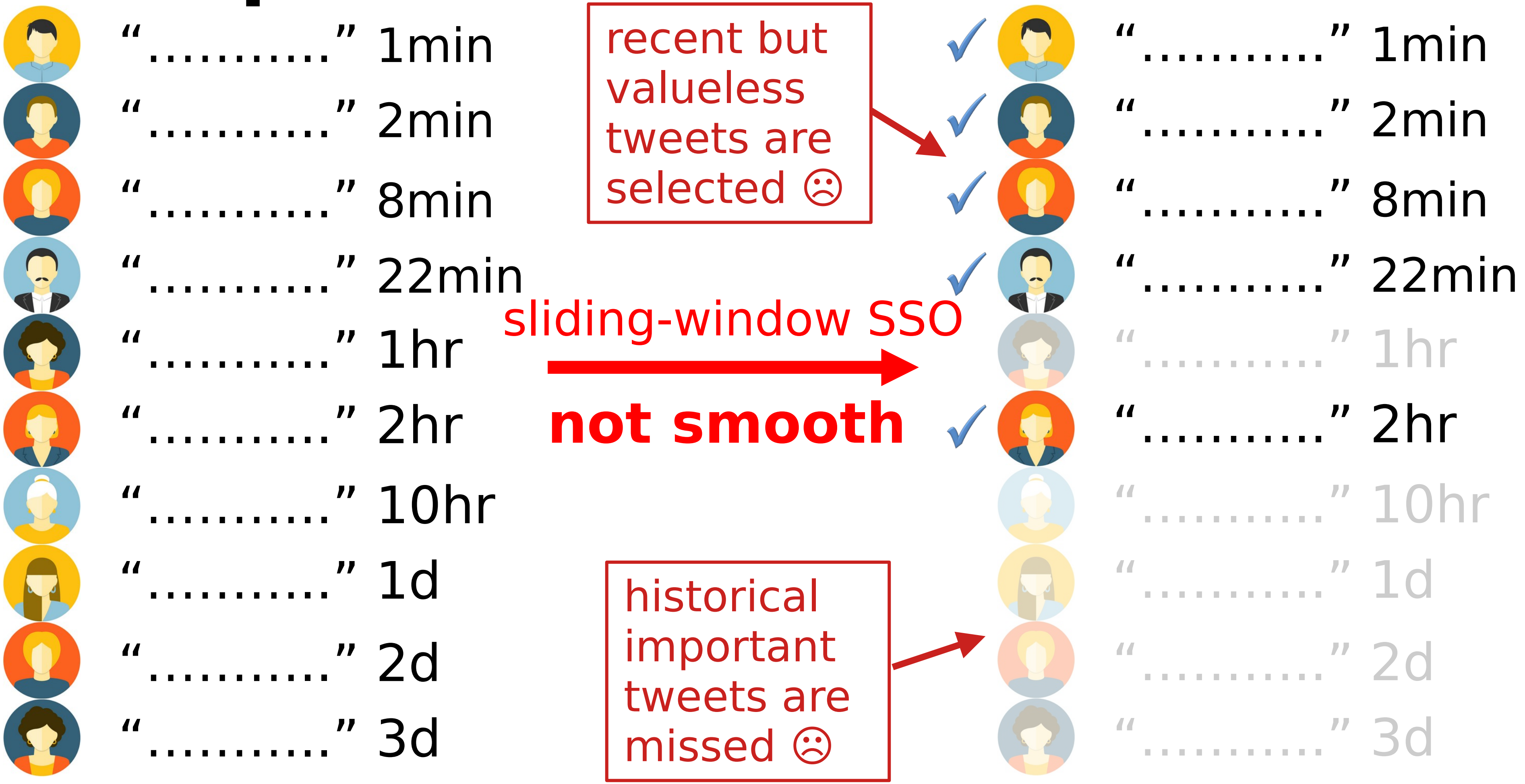
only the most recent items participate in the analysis

cons:
- abruptly forgets all past data which may be still important

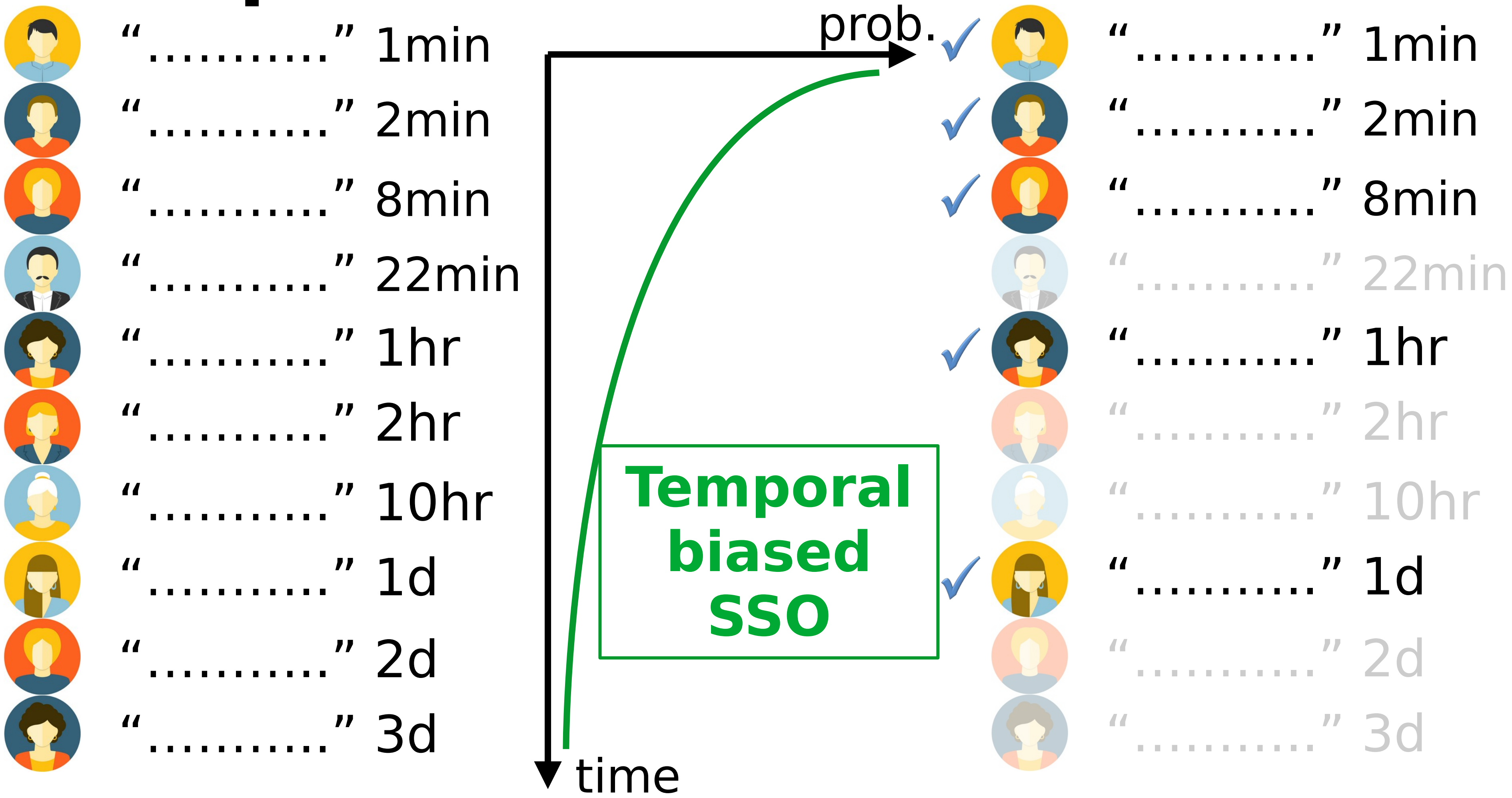
Example: Tweets Recommendation



Example: Tweets Recommendation



Example: Tweets Recommendation

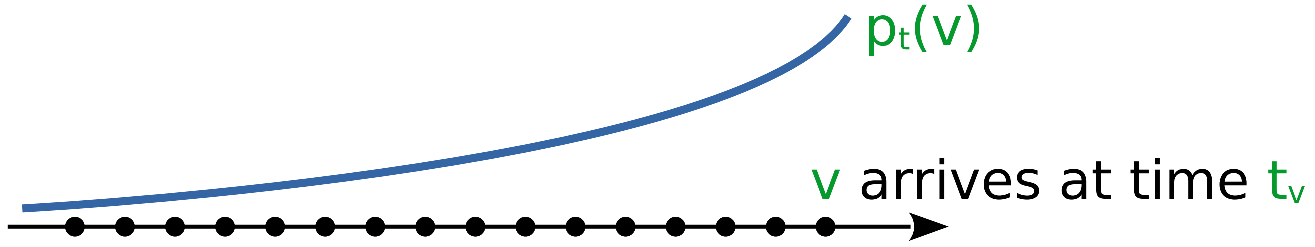


Outline

- Background & Motivation
- **TBSSO Problem Formulation**
- Algorithms
- Experiments
- Conclusion

Temporal Biased Stream Model

- Each item v participates in the analysis with a probability $p_t(v)$ decreasing over time.



- and $p_t(v) \triangleq h(t - t_v)$ where $h: Z \mapsto [0,1]$ assigns an item of age x a participation probability $h(x)$, called the decay function, e.g.,
  - $h(x) = p_0 e^{-\lambda x}$, i.e., an exponential decay function

Temporal Biased SSO Problem

TBSSO Problem formulation:

- **Given** a stream of items $\mathbb{S}_t = \{v \in V: t_v \leq t\}$ with decay function $h(x)$,
- **Want** to find k items $S \subseteq V$ that maximize $\mathbb{E}_h[f(S|\mathbb{S}_t)]$ at any query time t.

- The TBSSO problem generalizes previous settings:
  - If $h(x) = 1, \forall x$, then becomes insertion-only SSO.
  - If $h(x) = 1$ for $x \leq W$, and $h(x) = 0$ otherwise, then becomes sliding-window SSO.

Outline

- Background & Motivation
- TBSSO Problem Formulation
- **Algorithms**
- Experiments
- Conclusion

How to Calculate $\mathbb{E}_h[f(S|\mathbb{S}_t)]$?

Example:

Suppose $S=\{a,b\}$, then

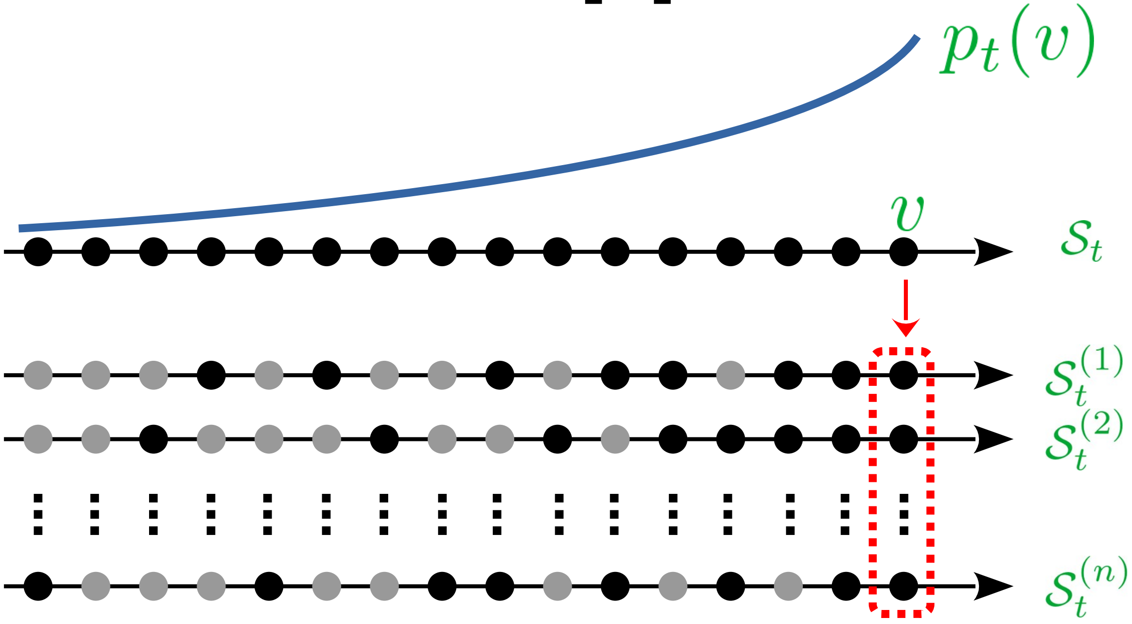
$$\mathbb{E}_h[f(\{a,b\}|\mathbb{S}_t)] = \boxed{p_t(a)p_t(b)f(\{a,b\})}$$ both a and b participate in the analysis

$$+ \boxed{p_t(a)(1 - p_t(b))f(\{a\})}$$ only a participates in the analysis

$$+ \boxed{(1 - p_t(a))p_t(b)f(\{b\})}$$ only b participates in the analysis

- Exactly calculating $\mathbb{E}_h[f(S|\mathbb{S}_t)]$ needs $O(2^{|S|})$ oracle calls,
  - one oracle call refers to one evaluation of f.
- **Too expensive!**

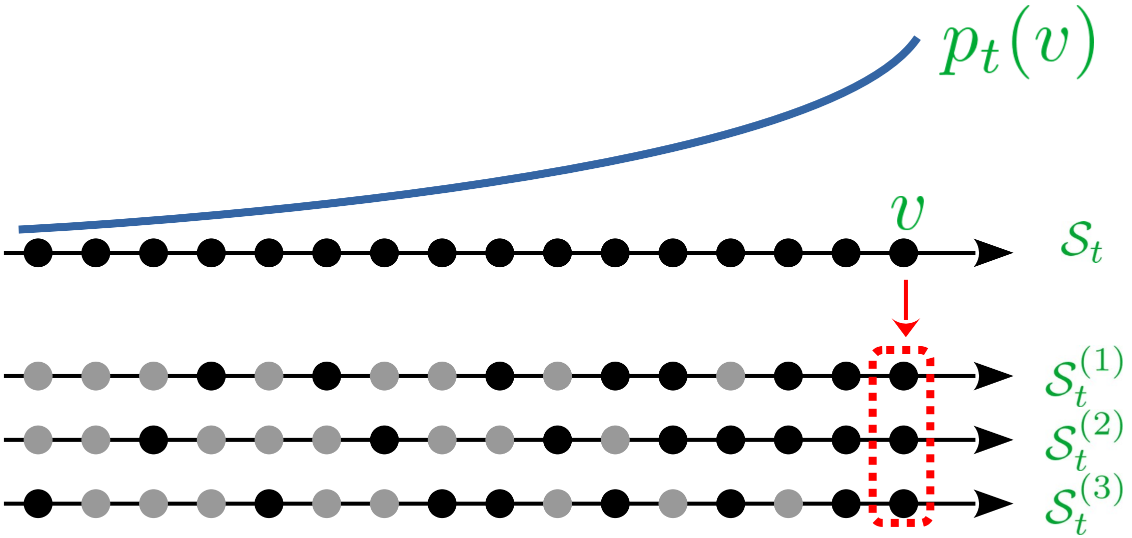
Monte-Carlo Approximation



$$F(S) \triangleq \frac{1}{n}\sum_{i=1}^{n} f(S \cap \mathcal{S}_t^{(i)}) \xrightarrow{a.s.} \mathbb{E}_h[f(S|\mathcal{S}_t)]$$ as $n \to \infty$

require n oracle calls, n << $2^{|S|}$

Bernoulli Set (B-Set)

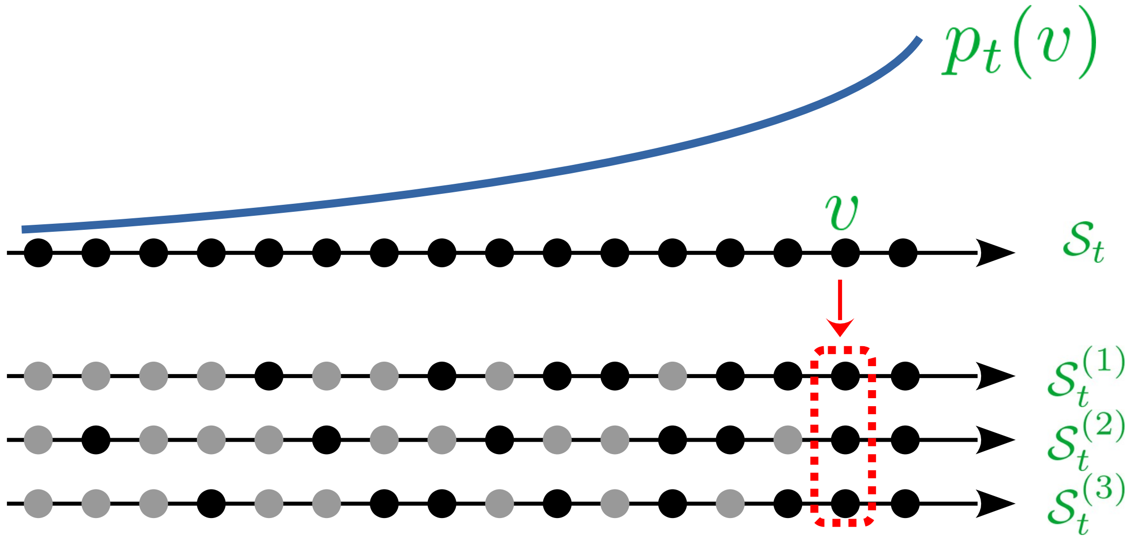


$$I_0(v) = \{1, 2, 3\}$$

- B-set of an item v at time $t_v + l$:

$$I_l(v) \triangleq \{i \colon v \in \mathcal{S}_t^{(i)} \wedge t_v + l = t\}$$

Bernoulli Set (B-Set)

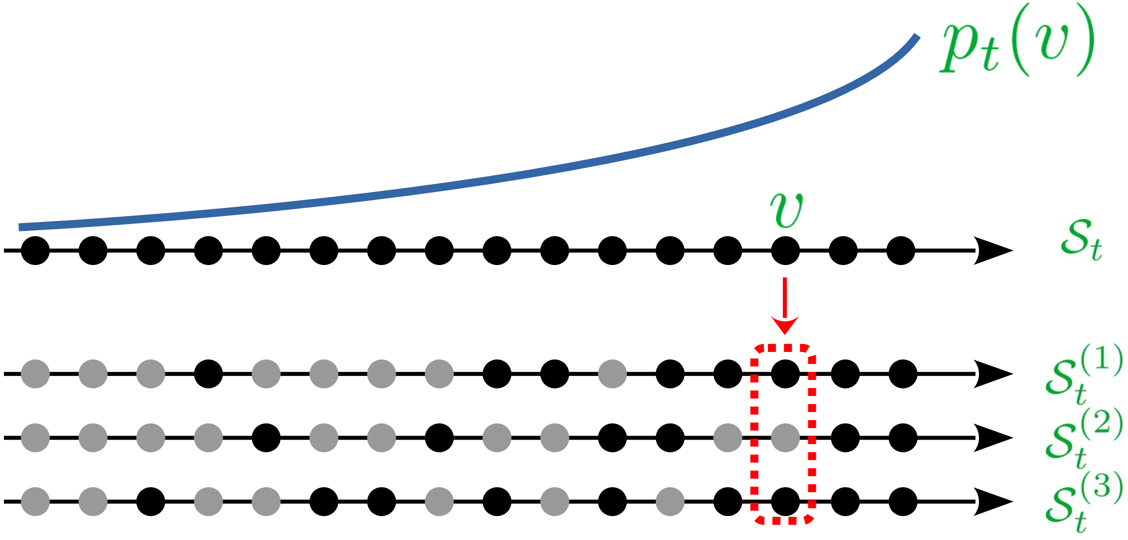


$$I_1(v) = \{1, 2, 3\}$$

- B-set of an item v at time $t_v + l$:

$$I_l(v) \triangleq \{i \colon v \in \mathcal{S}_t^{(i)} \wedge t_v + l = t\}$$

Bernoulli Set (B-Set)

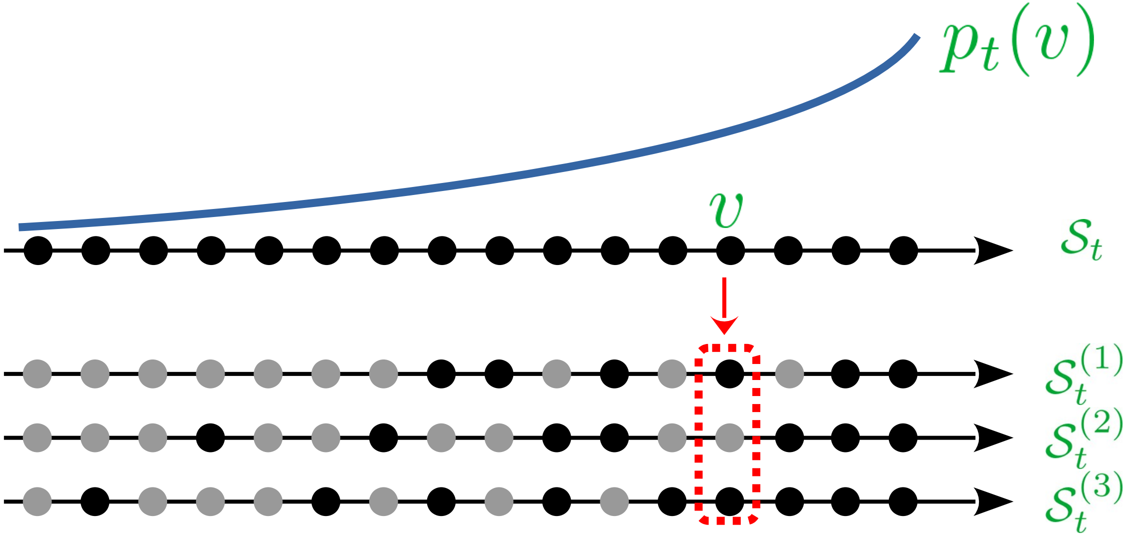


$I_2(v) = \{1, 3\}$

- B-set of an item v at time $t_v + l$:

$$I_l(v) \triangleq \{i \colon v \in \mathcal{S}_t^{(i)} \wedge t_v + l = t\}$$

Bernoulli Set (B-Set)

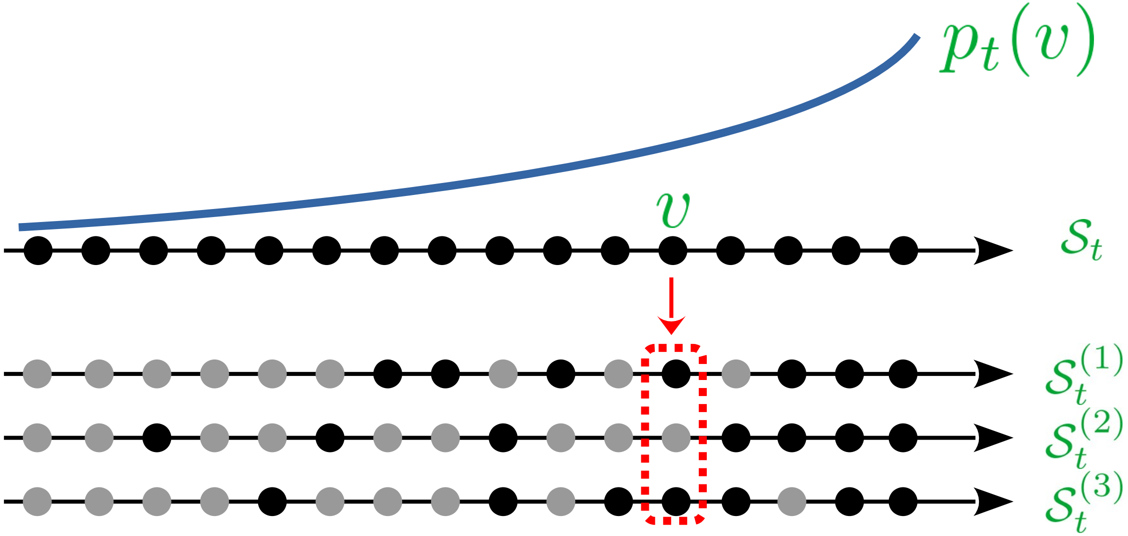


$I_3(v) = \{1, 3\}$

- B-set of an item v at time $t_v + l$:

$$I_l(v) \triangleq \{i : v \in \mathcal{S}_t^{(i)} \wedge t_v + l = t\}$$

Bernoulli Set (B-Set)

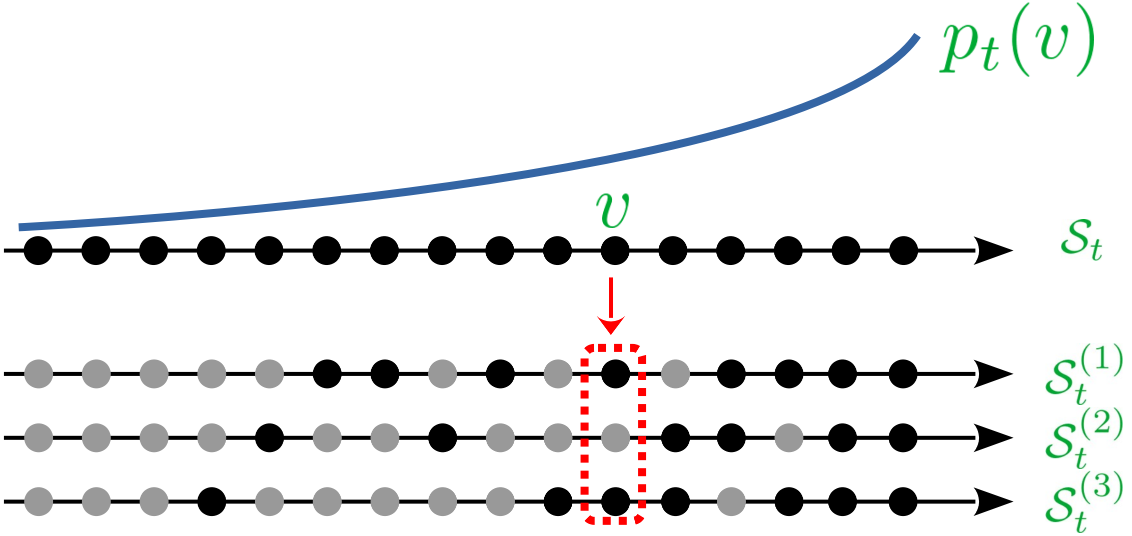


$$I_4(v) = \{1,3\}$$

- B-set of an item v at time $t_v + l$:

$$I_l(v) \triangleq \{i\colon v \in \mathcal{S}_t^{(i)} \wedge t_v + l = t\}$$

Bernoulli Set (B-Set)

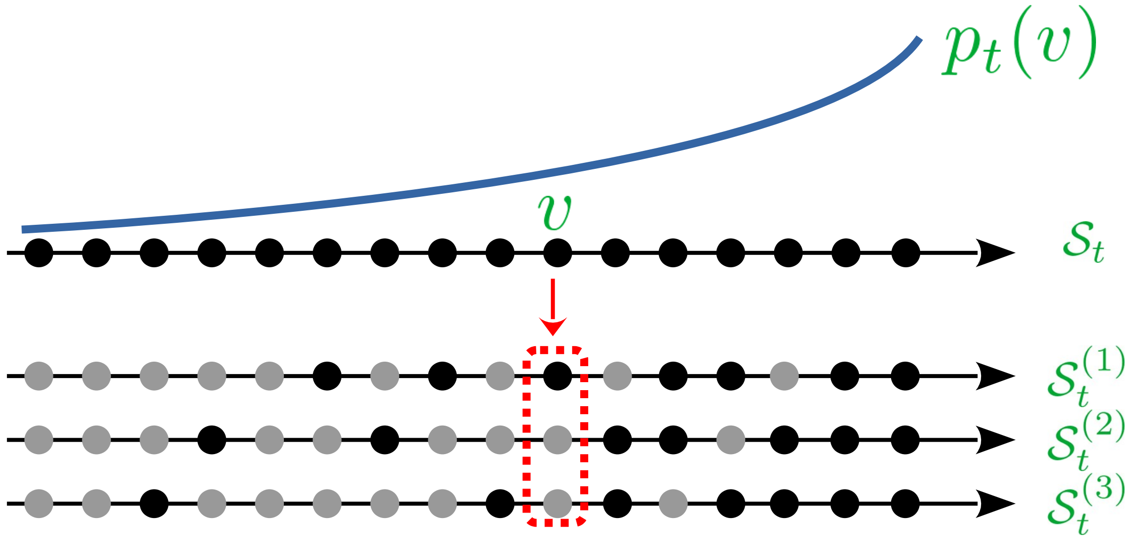


$$I_5(v) = \{1, 3\}$$

- B-set of an item v at time $t_v + l$:

$$I_l(v) \triangleq \{i : v \in \mathcal{S}_t^{(i)} \wedge t_v + l = t\}$$

Bernoulli Set (B-Set)

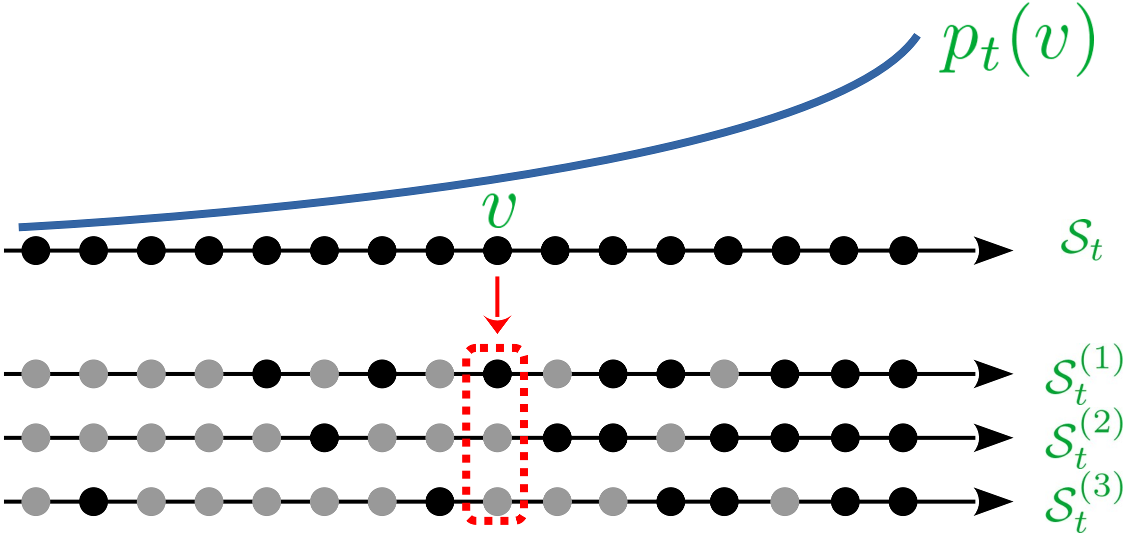


$I_6(v) = \{1\}$

- B-set of an item *v* at time $t_v + l$:

$$I_l(v) \triangleq \{i\colon v \in \mathcal{S}_t^{(i)} \wedge t_v + l = t\}$$

Bernoulli Set (B-Set)

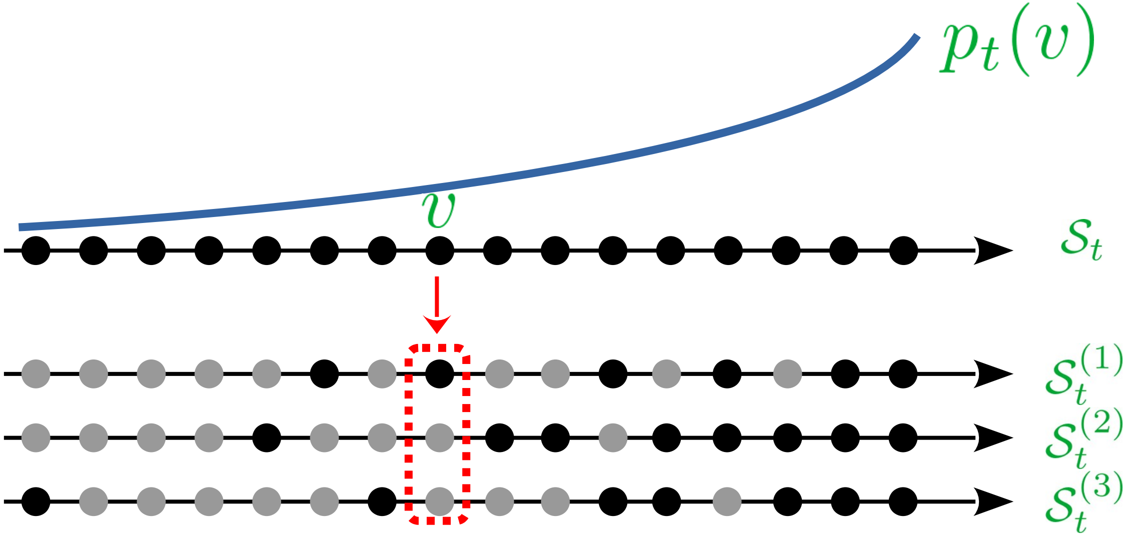


$I_7(v) = \{1\}$

- B-set of an item v at time $t_v + l$:

$$I_l(v) \triangleq \{i : v \in \mathcal{S}_t^{(i)} \wedge t_v + l = t\}$$

Bernoulli Set (B-Set)

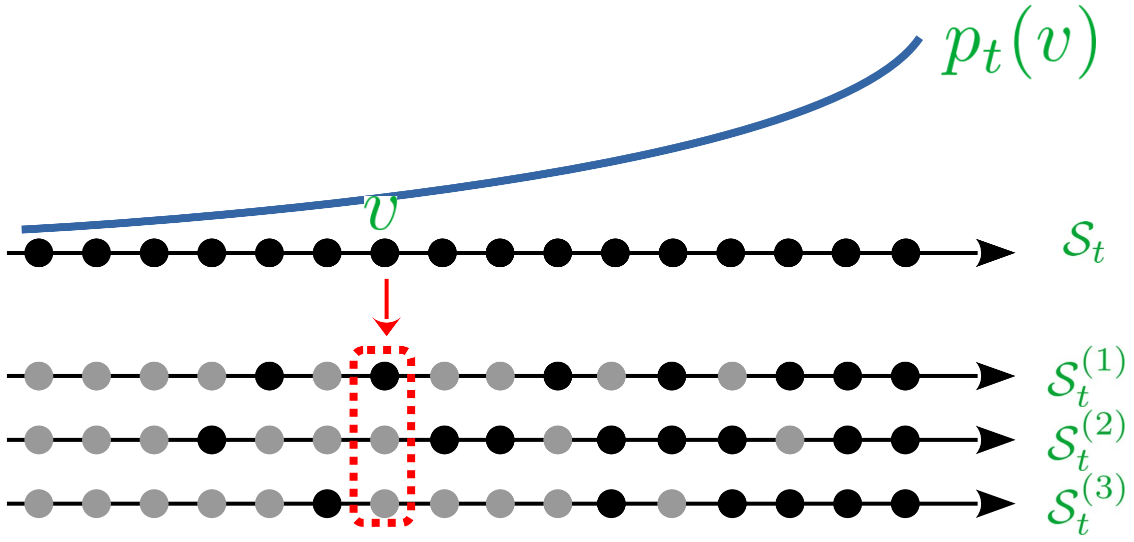


$$I_8(v) = \{1\}$$

- B-set of an item v at time $t_v + l$:

$$I_l(v) \triangleq \{i \colon v \in \mathcal{S}_t^{(i)} \wedge t_v + l = t\}$$

Bernoulli Set (B-Set)

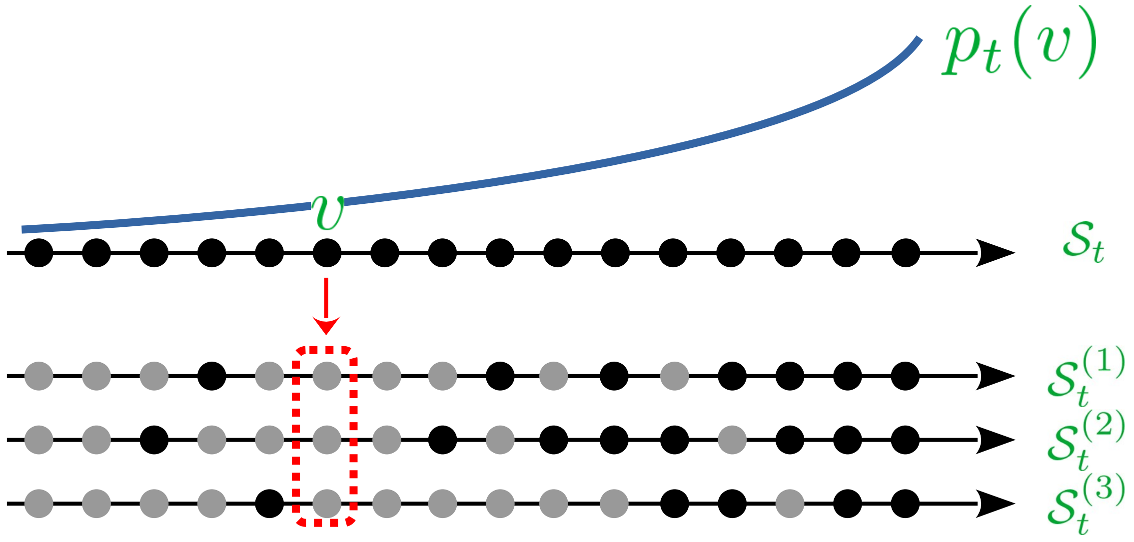


$I_9(v) = \{1\}$

- B-set of an item v at time $t_v + l$:

$$I_l(v) \triangleq \{i \colon v \in \mathcal{S}_t^{(i)} \wedge t_v + l = t\}$$

Bernoulli Set (B-Set)



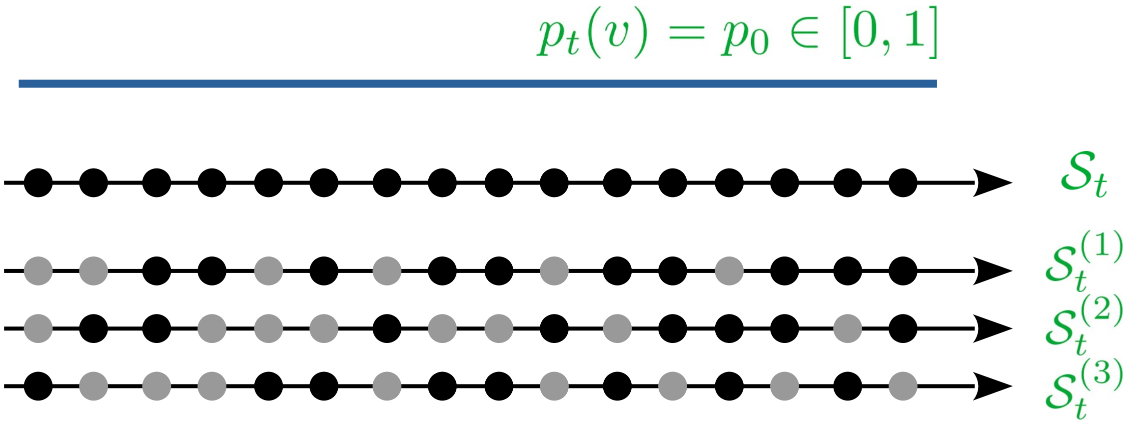
$I_{10}(v) = \emptyset$

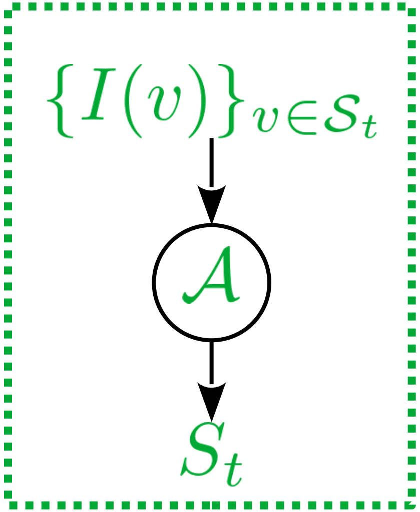
- B-set of an item v at time $t_v + l$:

$$I_l(v) \triangleq \{i : v \in \mathcal{S}_t^{(i)} \wedge t_v + l = t\}$$

- Eventually, a B-set will shrink to an empty set.
- B-set is another way to represent probabilistic decays.

The Non-Decaying Case

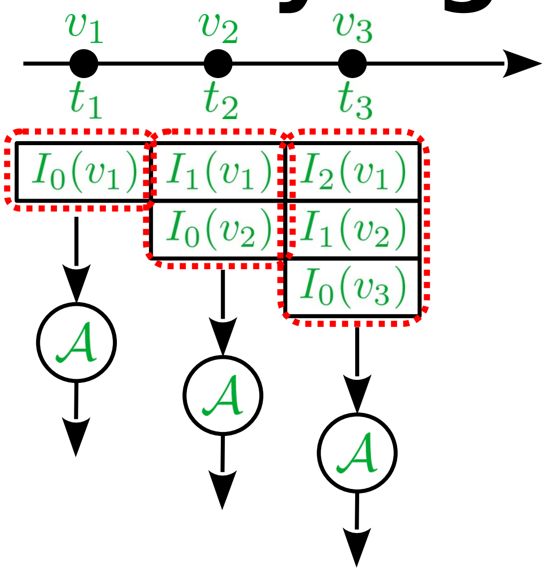




Atom

- Each item’s B-set is a constant set.
- The stream becomes an insertion-only stream, where each item in the stream is a B-set, i.e., $\{I(v) \colon v \in \mathcal{S}_t\}$
- Many insertion-only SSO algorithms can be applied.
  - denote one implementation by Atom.

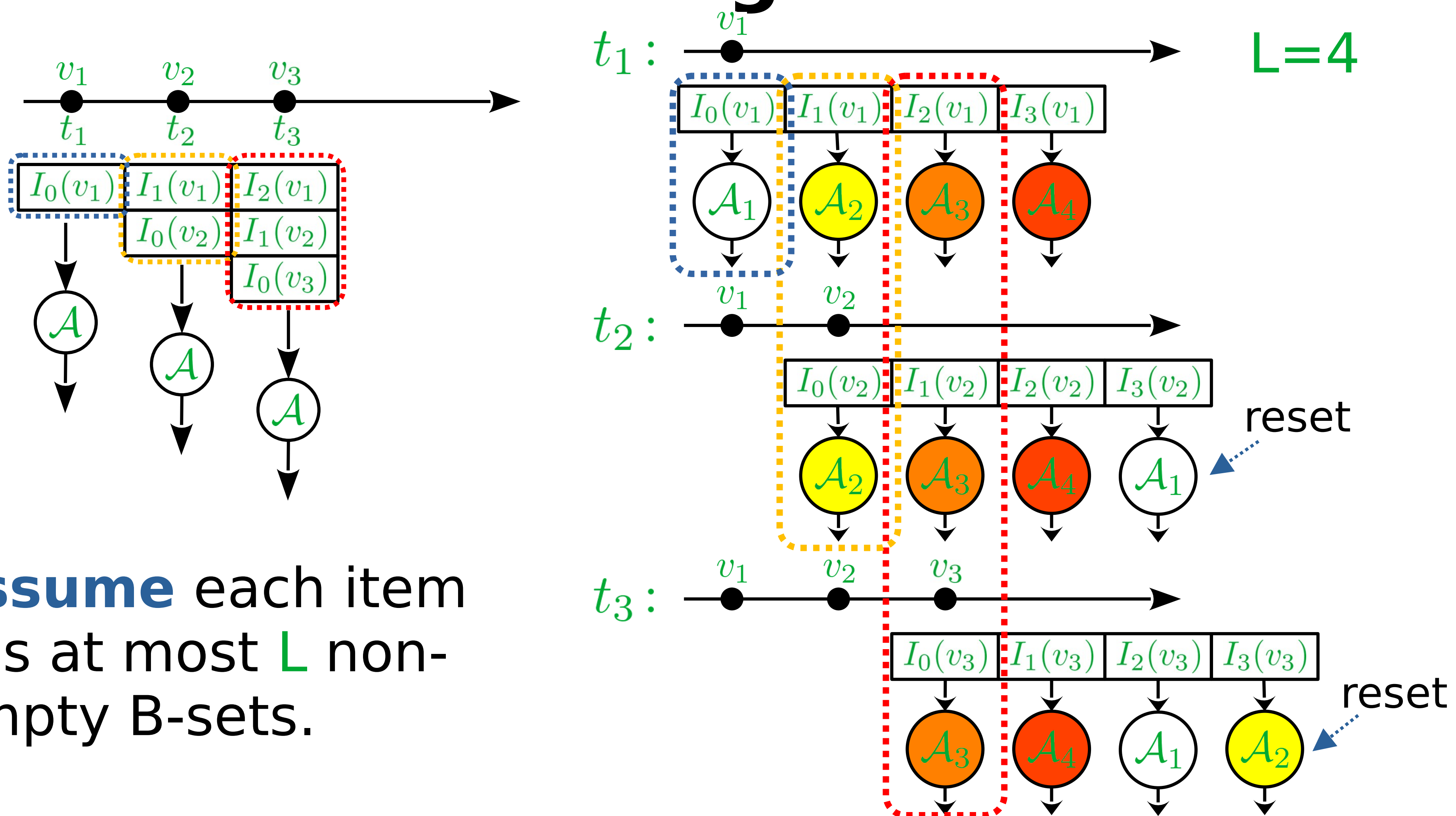
The Decaying Case



$$I_0(v_1) \supseteq I_1(v_1) \supseteq I_2(v_1) \supseteq \ldots$$
$$I_0(v_2) \supseteq I_1(v_2) \supseteq \ldots$$

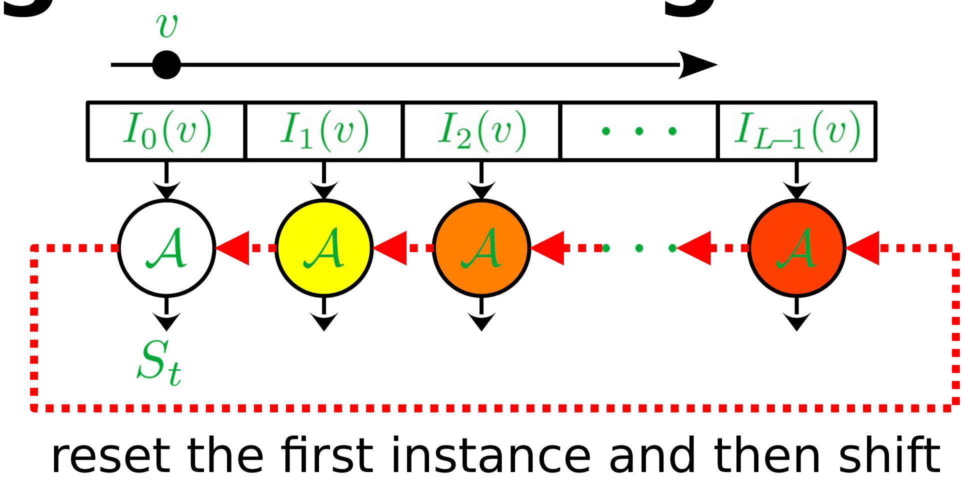
- **Idea**: If we can feed the B-sets at each time point to an Atom instance, then the instance’s output will be the solution of the TBSSO problem at the time point.
- **Challenge**: Each B-set is shrinking, how to process these evolving B-sets in a streaming fashion?

A Basic Algorithm



- **Assume** each item has at most L non-empty B-sets.

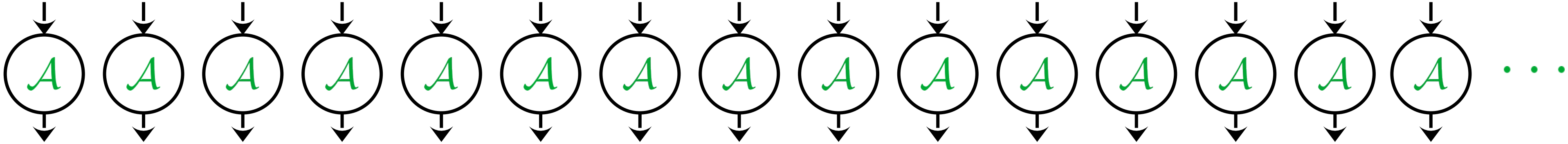
Basic Alg: Processing and Shifting



- For each item v, obtain its B-sets $\{I_l(v)\text{: } 0 \leq l < L\}$;
- Feed these B-sets to L Atom instances, respectively;
- Reset the first instance and circularly shift these instances before processing the next item;
- The first Atom instance's out is always the solution.

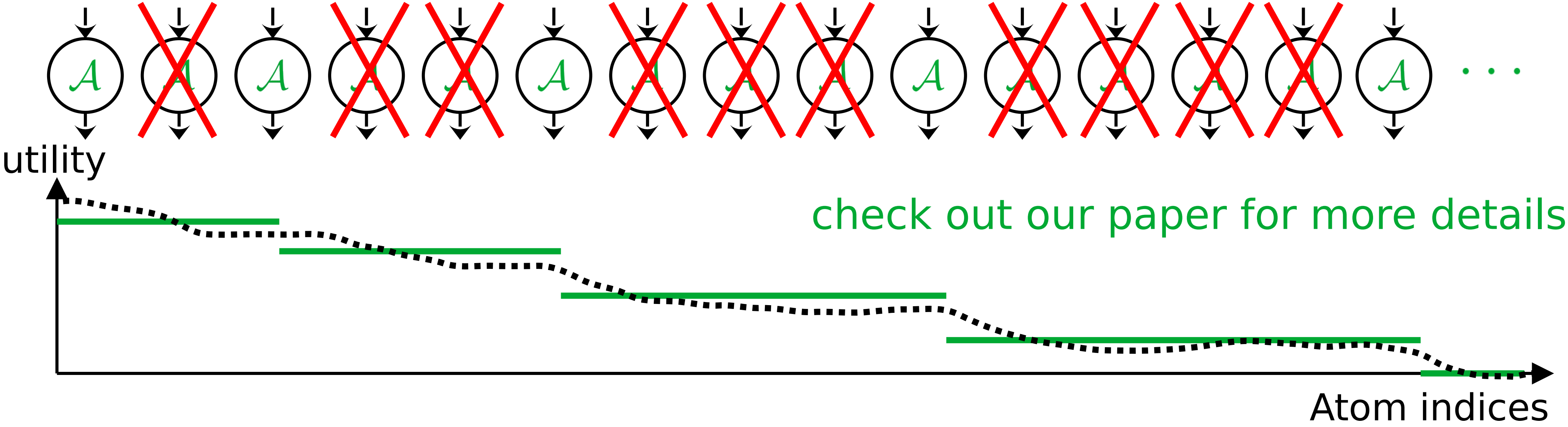
Limitations of the Basic Algorithm

- What if L is very large, e.g., $L = \infty$?
- **Weakness**: Maintaining a large number of Atom instances will incur too much RAM and CPU overloads.



A Faster Algorithm

- **Idea**: remove redundant Atom instances
  - trick 1: group same B-sets into one segment, each segment needs only one Atom instance;
  - trick 2: remove Atom instances that have close outputs
- Similar to use a histogram to approximate a curve.



Summary

Algorithm	Approx. Ratio	Update Time	Space
Atom	α	β	γ
Basic Alg.	α	$O(L\beta)$	$O(L\gamma)$
Fast Alg.	$\alpha(1-\varepsilon)/2$	$O(\beta\varepsilon^{-1}\log k)$	$O(\gamma\varepsilon^{-1}\log k)$

- For example, if Atom is implemented by adapting SieveStreaming [KDD'14], then
  - $\alpha=1/2 - \varepsilon$
  - $\beta=O(n\varepsilon^{-1}\log k)$
  - $\gamma=O(nk\varepsilon^{-1}\log k)$

Outline

- Background & Motivation
- TBSSO Problem Formulation
- Algorithms
- ☞ **Experiments**
- Conclusion

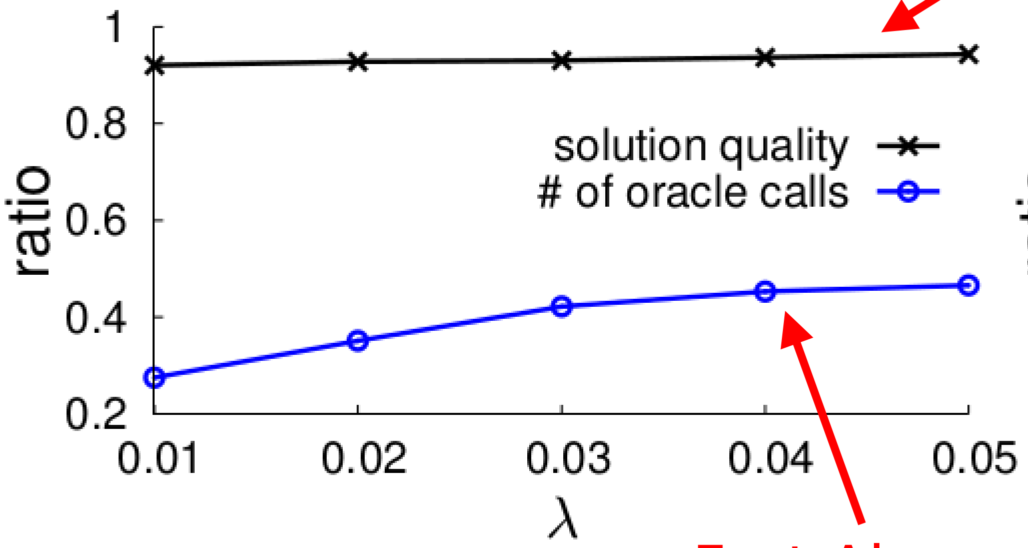
Dataset

data stream	element	length	time period
DBLP [9]	author	372K	1936 - 2018
MemeTracker [21]	article	714K	1/1/2009 - 31/1/2009
StackOverflow [26]	question	2.9M	1/1/2015 - 1/3/2016
US2020-Tweets [11]	tweet	1.7M	15/10/2020 - 8/11/2020

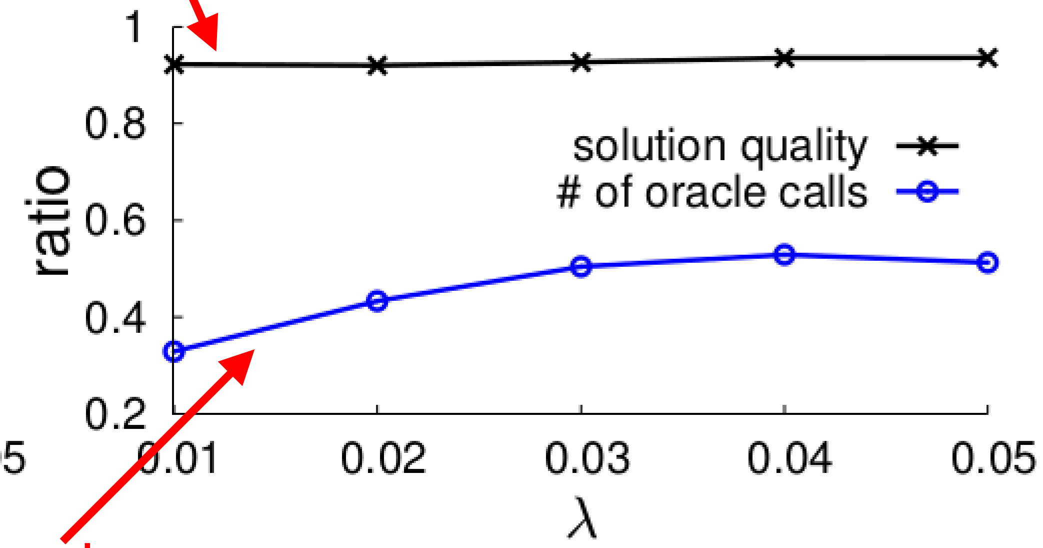
- **Goal**: maintain *k* items that have the maximum coverage, i.e., $f(S) = |\cup_{v \in S} v|$ where each item is a set.
- **Decay function**: $h(x) = e^{-\lambda x}$, $\lambda \geq 0$.
- Baselines:
  - Greedy: uses the lazy evaluation trick, non-streaming;
  - Random: randomly select *k* items.

Basic Alg. vs Fast Alg.

similar solution quality



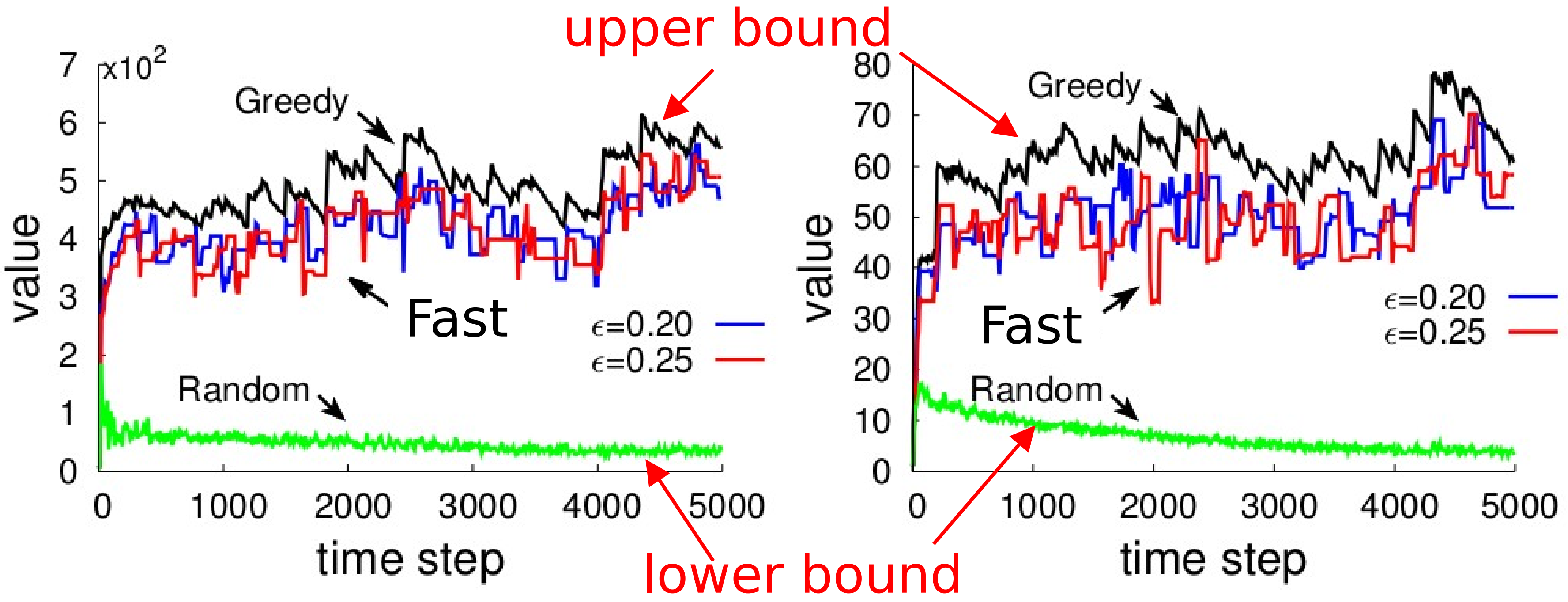
(a) DBLP



(b) MemeTracker

Fast Alg needs fewer oracle calls than Basic Alg

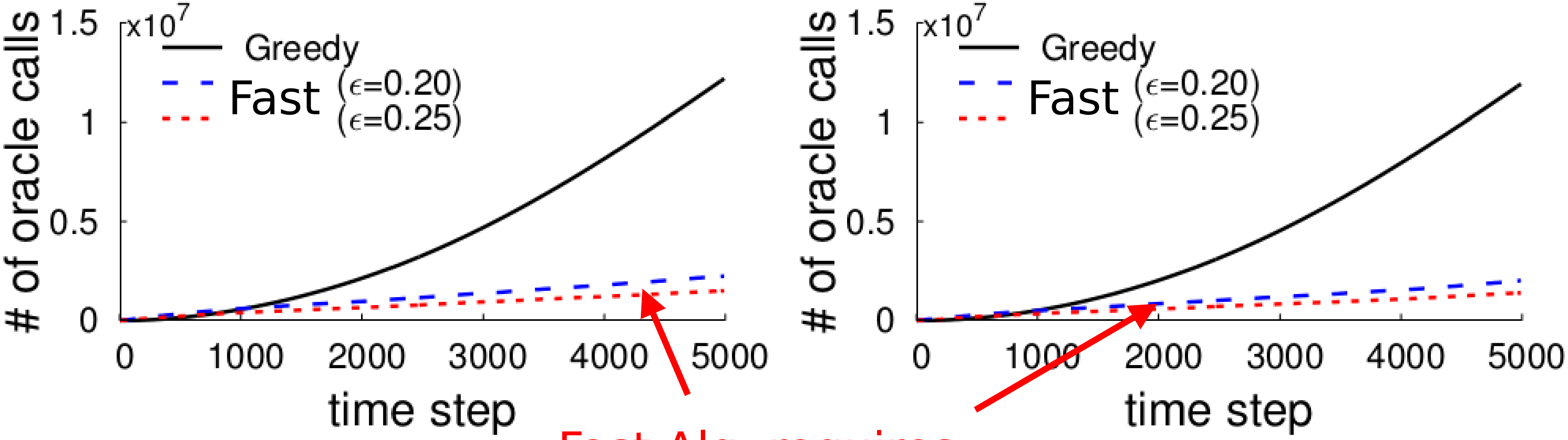
Solution Quality



(a) DBLP (b) MemeTracker

- The fast algorithm is close to the solution quality of Greedy, and is much better than the Random.

Computational Efficiency



(a) DBLP

(b) MemeTracker

Fast Alg. requires much fewer oracle calls than Greedy

Outline

- Background & Motivation
- TBSSO Problem Formulation
- Algorithms
- Experiments
- **Conclusion**

Conclusion

- A novel Temporal Biased Streaming Submodular Optimization problem.
- Can be reduced to SSO over insertion-only streams.
- The proposed algorithm can find near-optimal solutions with much fewer costs than baselines.

Algorithm	Approx. Ratio	Update Time	Space
Atom	α	β	γ
Basic Alg.	α	$O(L\beta)$	$O(L\gamma)$
Fast Alg.	$\alpha(1-\varepsilon)/2$	$O(\beta\varepsilon^{-1}\log k)$	$O(\gamma\varepsilon^{-1}\log k)$

Thanks for Listening!