Finding Frequent Items in Probabilistic Data

Qin Zhang, Hong Kong University of Science & Technology

Feifei Li, Florida State University

Ke Yi, Hong Kong University of Science & Technology

SIGMOD 2008

June 12, 2008

Motivation

- Identifying frequent items is important
  - network traffic monitoring
  - answering iceberg queries
  - association rule mining

  ......

Motivation

- Identifying frequent items is important
  - network traffic monitoring
  - answering iceberg queries
  - association rule mining

  ......
- Also, processing uncertain data
  - sensor reading
  - fuzzy data integration

  ......

Motivation

- Identifying frequent items is important
  - network traffic monitoring
  - answering iceberg queries
  - association rule mining

  ......
- Also, processing uncertain data
  - sensor reading
  - fuzzy data integration

  ......
- This paper: find frequent items in uncertain data

  (heavy hitters)

Previous work on heavy hitters in certain data

For a parameter ϕ, an item t is the ϕ-heavy hitter of a bag W if $m_t^W > \phi \cdot |W|$.

Previous work on heavy hitters in certain data

For a parameter ϕ, an item t is the ϕ-heavy hitter of a bag W if $m_t^W > \phi \cdot |W|$.

Approximate version heavy hitters

- return all the ϕ-heavy hitters
- not return those t with $m_t^W < (\phi - \epsilon) \cdot |W|$
- items in between: arbitrary

Previous work on heavy hitters in certain data

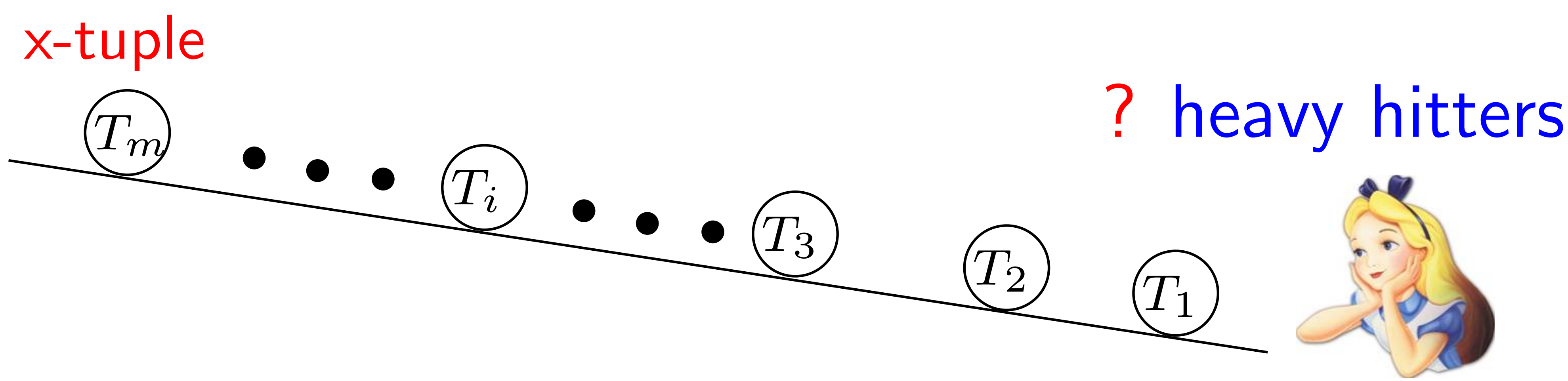
For a parameter ϕ, an item t is the ϕ-heavy hitter of a bag W if $m_t^W > \phi \cdot |W|$.

Approximate version heavy hitters

- return all the ϕ-heavy hitters
- not return those t with $m_t^W < (\phi - \epsilon) \cdot |W|$
- items in between: arbitrary

- Misra and Gries (Sci. Comput. Programming 1982)
- Demaine et. al. (ESA 2002)
- Manku & Motwani (VLDB 2002)
- Karp et. al. (TODS 2003)
- Cormode & Muthukrishnan (VLDB 2002)
- Cormode et. al. (SIGMOD 2004)
- Manjhi et. al. (ICDE 2005)
- Lee & Ting (PODS 2006)
- Metwally et. al. (TODS 2006)

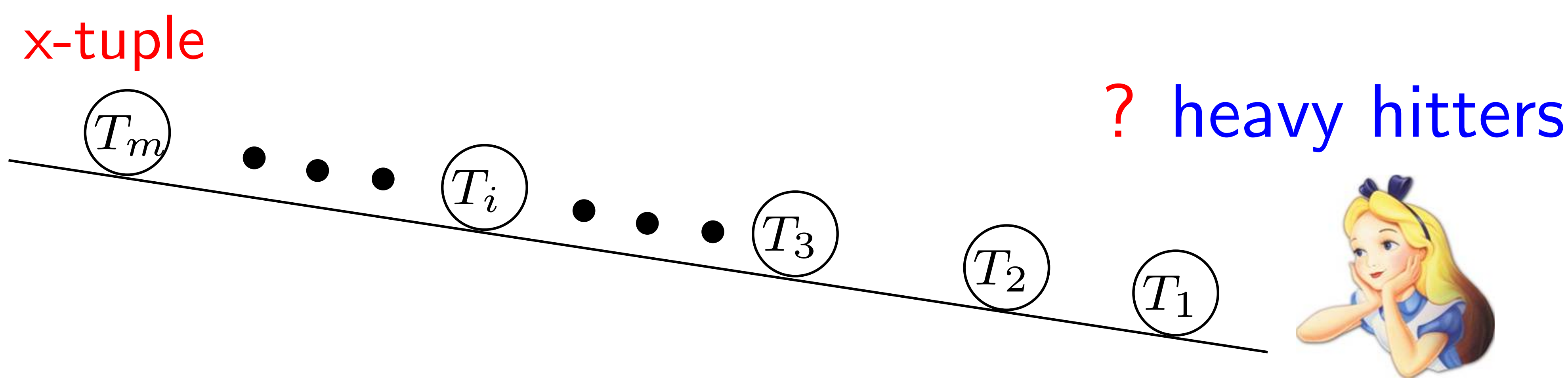
The Probabilistic Model



The x-tuple model (proposed in the TRIO system)

T_1	$\{(a, p(a)), (b, p(b))\}$
T_2	$\{(a, p'(a))\}$

The Probabilistic Model

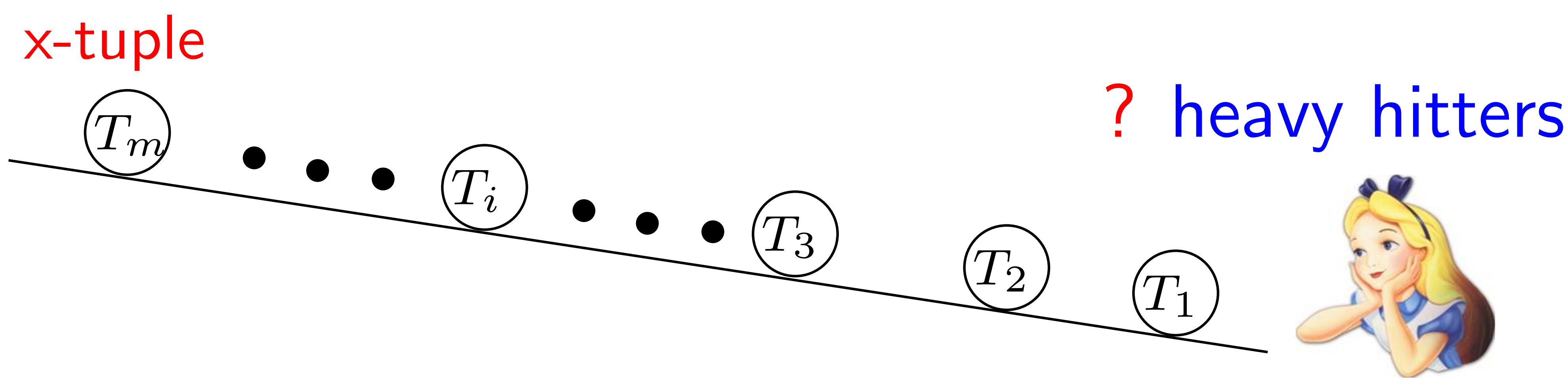


The x-tuple model (proposed in the TRIO system)

T_1	$\{(a, p(a)), (b, p(b))\}$
T_2	$\{(a, p'(a))\}$

a occurs with Pr $p(a)$, b occurs with Pr $p(b)$, nothing occurs with Pr $1 - p(a) - p(b)$.

The Probabilistic Model



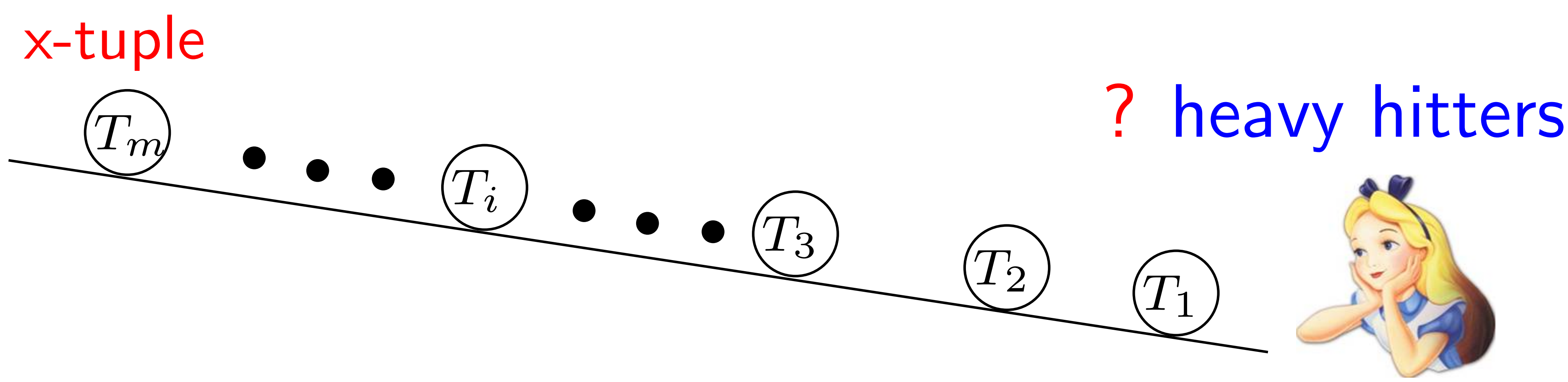
The x-tuple model (proposed in the TRIO system)

T_1	$\{(a, p(a)), (b, p(b))\}$
T_2	$\{(a, p'(a))\}$

$\mathcal{D}$: the uncertain database, consists of T_1 and T_2
W: a possible world of $\mathcal{D}$

W	$\Pr[W]$
$\emptyset$	$(1 - p(a) - p(b))(1 - p'(a))$
$\{a\}$	$p(a)(1 - p'(a))$ $+(1 - p(a) - p(b))p'(a)$
$\{b\}$	$p(b)(1 - p'(a))$
$\{aa\}$	$p(a)p'(a)$
$\{ab\}$	$p(b)p'(a)$

The Probabilistic Model



The x-tuple model (proposed in the TRIO system)

T_1	$\{(a, p(a)), (b, p(b))\}$
T_2	$\{(a, p'(a))\}$

$\mathcal{D}$: the uncertain database, consists of T_1 and T_2

W: a possible world of $\mathcal{D}$

Let R denote a random possible world

$|R|$: the number of items in R.

m_t^R: the frenquency of item t in R.

W	$\Pr[W]$
$\emptyset$	$(1-p(a)-p(b))(1-p'(a))$
$\{a\}$	$p(a)(1-p'(a))$ $+(1-p(a)-p(b))p'(a)$
$\{b\}$	$p(b)(1-p'(a))$
$\{aa\}$	$p(a)p'(a)$
$\{ab\}$	$p(b)p'(a)$

EHH and PHH

- An intuitive definition
  t is a *ϕ-expected heavy hitter* (EHH) of $\mathcal{D}$ if

$$E[m_t^R] > \phi \cdot E[|R|]$$

EHH and PHH

- An intuitive definition
  t is a *ϕ-expected heavy hitter* (EHH) of $\mathcal{D}$ if

  $$E[m_t^R] > \phi \cdot E[|R|]$$

- Problems with EHH (finding 0.5-heavy hitters.)

  - $\mathcal{D}_1 = \{\ \{(a, 0.9), (b, 0.1)\}, \{(c, 1)\}\ \}$.
    with Pr. 0.9 $R = \{a, c\}$
    with Pr. 0.1 $R = \{b, c\}$
    a is not a 0.5-expected heavy hitter.
    But, a has a 90% chance of being a 0.5-heavy hitter!

EHH and PHH

- An intuitive definition
  t is a *ϕ-expected heavy hitter* (EHH) of $\mathcal{D}$ if

  $$E[m_t^R] > \phi \cdot E[\|R\|]$$

- Problems with EHH (finding 0.5-heavy hitters.)
  - $\mathcal{D}_1 = \{ \{(a, 0.9), (b, 0.1)\}, \{(c, 1)\} \}$.
    with Pr. 0.9 $R = \{a, c\}$
    with Pr. 0.1 $R = \{b, c\}$
    a is not a 0.5-expected heavy hitter.
    But, a has a 90% chance of being a 0.5-heavy hitter!
  - $\mathcal{D}_2 = \{\{(a, 0.5)\}, \{(b, 0.5)\}\}$.
    a is a 0.5-expected heavy hitter,
    but only has a 50% chance of being a 0.5-heavy hitter.

EHH and PHH

- An intuitive definition
  t is a *ϕ-expected heavy hitter* (EHH) of $\mathcal{D}$ if

  $$E[m_t^R] > \phi \cdot E[|R|]$$

- A more rigorous definition
  t is a *(ϕ, τ)-probabilistic heavy hitter* (PHH) of $\mathcal{D}$ if

  $$\Pr[m_t^R > \phi|R|] > \tau$$

  Follow "probabilistic thresholding" framework
  (Dalvi and Suciu VLDB 2004)

EHH and PHH

- An intuitive definition
  t is a *ϕ-expected heavy hitter* (EHH) of $\mathcal{D}$ if

  $$E[m_t^R] > \phi \cdot E[|R|]$$

- A more rigorous definition
  t is a *(ϕ, τ)-probabilistic heavy hitter* (PHH) of $\mathcal{D}$ if

  $$\Pr[m_t^R > \phi|R|] > \tau$$

  Follow "probabilistic thresholding" framework
  (Dalvi and Suciu VLDB 2004)

Summary of main results

1. Give low degree polynomial-time algorithms for computing the exact PHH for offline data.

2. Design both space and time-efficient algorithms to compute the approximate PHH for streaming data, with theoretically guaranteed accuracy and space/time bounds.

3. Establish a tradeoff between the accuracy and the per-tuple processing time of the proposed approximation algorithms.

Algorithm for offline data

- For a single item t, dynamic programming(DP).

  m: the number of x-tuples, n: the number of distinct items

  The running time of DP $O(m^3)$.

  - Main idea: calculate Pr[item t appears i times and items other than t appear j times in the first k x-tuples of $\mathcal{D}$] for all i, j, k.

- Thus, if we do this for every item, the running time would be $O(nm^3)$

Algorithm for offline data

- For a single item t, dynamic programming(DP).

  m: the number of x-tuples, n: the number of distinct items

  The running time of DP $O(m^3)$.

  - Main idea: calculate Pr[item t appears i times and items other than t appear j times in the first k x-tuples of $\mathcal{D}$] for all i, j, k.

- Thus, if we do this for every item, the running time would be $O(nm^3)$

- However, we can reduce the running time by almost a factor of n using the pruning lemma (next page).

The Prunning Lemma

- The following lemma gives an upper bound on $\Pr[m_t^R > \phi|R|]$ depending on $E[m_t^R]/E[|R|]$.

$$Pr[m_t^R > \phi|R|] \leq \frac{2}{\phi}\frac{E[m_t^R]}{E[|R|]} + e^{-\frac{1}{8}E[|R|]}$$

(small)

The Prunning Lemma

- The following lemma gives an upper bound on $\Pr[m_t^R > \phi|R|]$ depending on $E[m_t^R]/E[|R|]$.

$$Pr[m_t^R > \phi|R|] \leq \frac{2}{\phi}\frac{E[m_t^R]}{E[|R|]} + e^{-\frac{1}{8}E[|R|]}$$

(small)

If $\phi = 0.1$, $\tau = 0.6$

$\frac{E[m_t^R]}{E[|R|]} < 0.02 \rightarrow Pr[m_t^R > \phi|R|] < 0.6$

since $\sum_t E(m_t^R) = E(|R|)$, checking 50 items is enough!

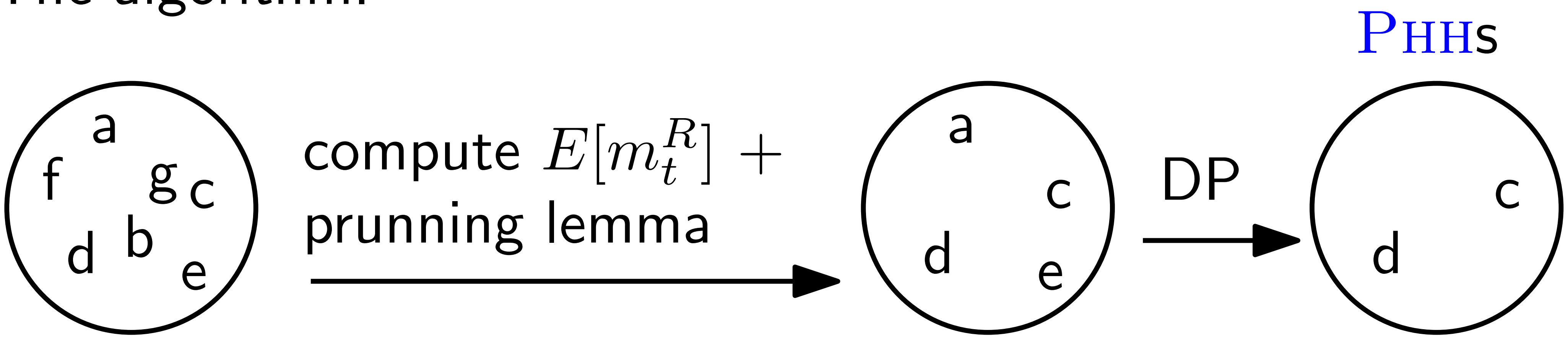
The Prunning Lemma

- The following lemma gives an upper bound on $\Pr[m_t^R > \phi|R|]$ depending on $E[m_t^R]/E[|R|]$.

$$Pr[m_t^R > \phi|R|] \leq \frac{2}{\phi}\frac{E[m_t^R]}{E[|R|]} + e^{-\frac{1}{8}E[|R|]}$$

(small)

- The algorithm.



Now running time is $O(\frac{1}{\phi\tau}m^3)$.

Approximation algorithms for streaming data

- An item t is an approximate PHH if
  $Pr[m_t^R > \phi|R|] > \tau$, and not an approximate PHH if
  $Pr[m_t^R > (\phi - \epsilon)|R|] < (1 - \theta)\tau$.

  Approximation is necessary since calculating the exact frequency of heavy hitters require $\Omega(n)$ memory (Alon et. al.)

Approximation algorithms for streaming data

- An item t is an approximate $\textsc{Phh}$ if $Pr[m_t^R > \phi|R|] > \tau$, and not an approximate $\textsc{Phh}$ if $Pr[m_t^R > (\phi - \epsilon)|R|] < (1-\theta)\tau$.

  Approximation is necessary since calculating the exact frequency of heavy hitters require $\Omega(n)$ memory (Alon et. al.)

- We propose algorithms with the following guarantees.

  - finds all approximate (ϕ, τ)-$\textsc{Phh}$ with probability at least $1 - \delta$.
  - space $O(\frac{1}{\epsilon\theta^2\tau} \log(\frac{1}{\delta\phi\tau}))$
  - processing time: $O(\frac{1}{\theta^2\tau} \log(\frac{1}{\delta\phi\tau}) + \log(1/\epsilon))$

Approximation algorithms for streaming data

- An item t is an approximate $\textsc{Phh}$ if $Pr[m_t^R > \phi|R|] > \tau$, and not an approximate $\textsc{Phh}$ if $Pr[m_t^R > (\phi - \epsilon)|R|] < (1-\theta)\tau$.

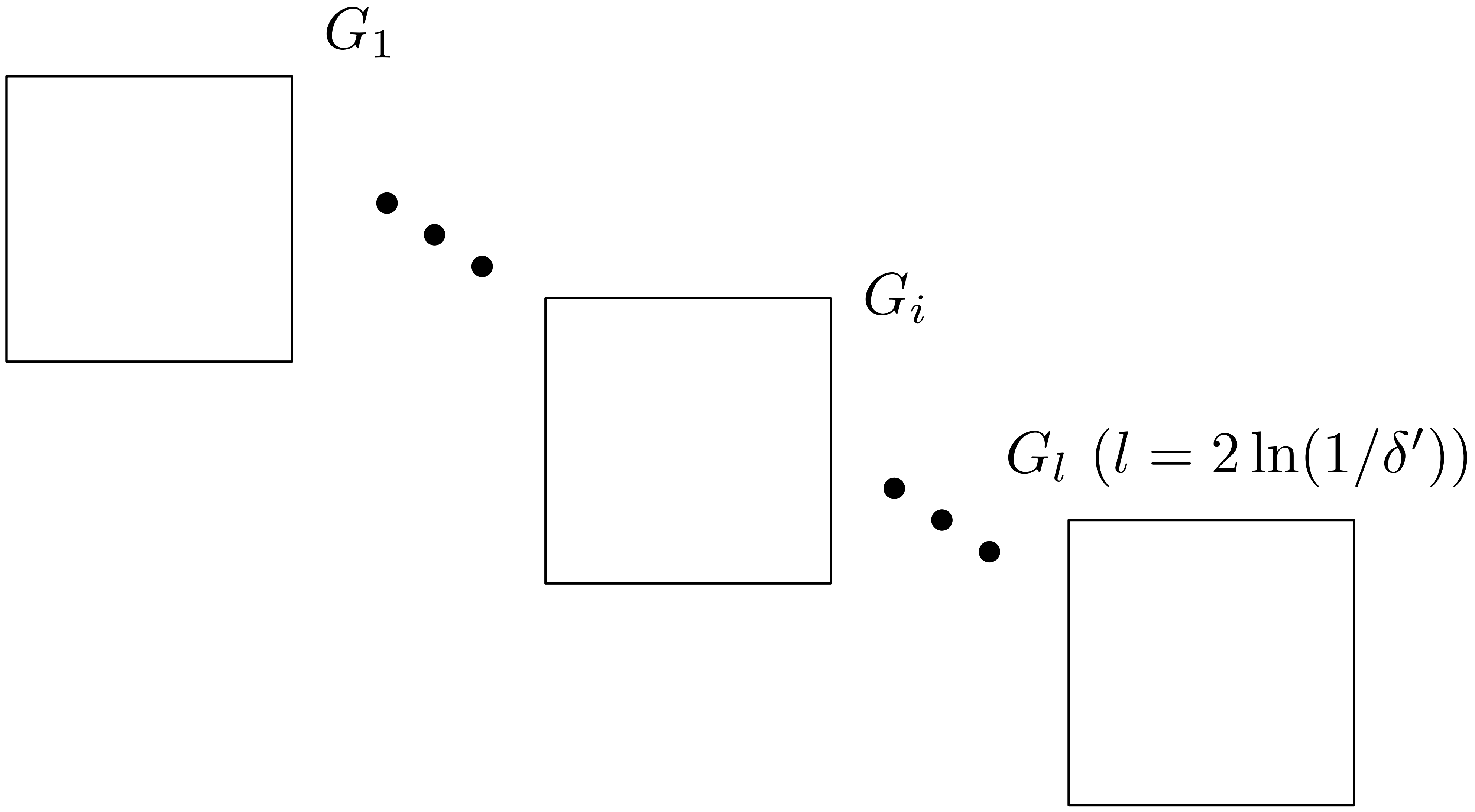
  Approximation is necessary since calculating the exact frequency of heavy hitters require $\Omega(n)$ memory (Alon et. al.)

- We propose algorithms with the following guarantees.
  - finds all approximate (ϕ, τ)-$\textsc{Phh}$ with probability at least $1-\delta$.
  - space $O(\frac{1}{\epsilon\theta^2\tau}\log(\frac{1}{\delta\phi\tau}))$
  - processing time: $O(\frac{1}{\theta^2\tau}\log(\frac{1}{\delta\phi\tau}) + \log(1/\epsilon))$

    further improve to : $O(\log(\frac{1}{\delta\phi\tau\epsilon}))$

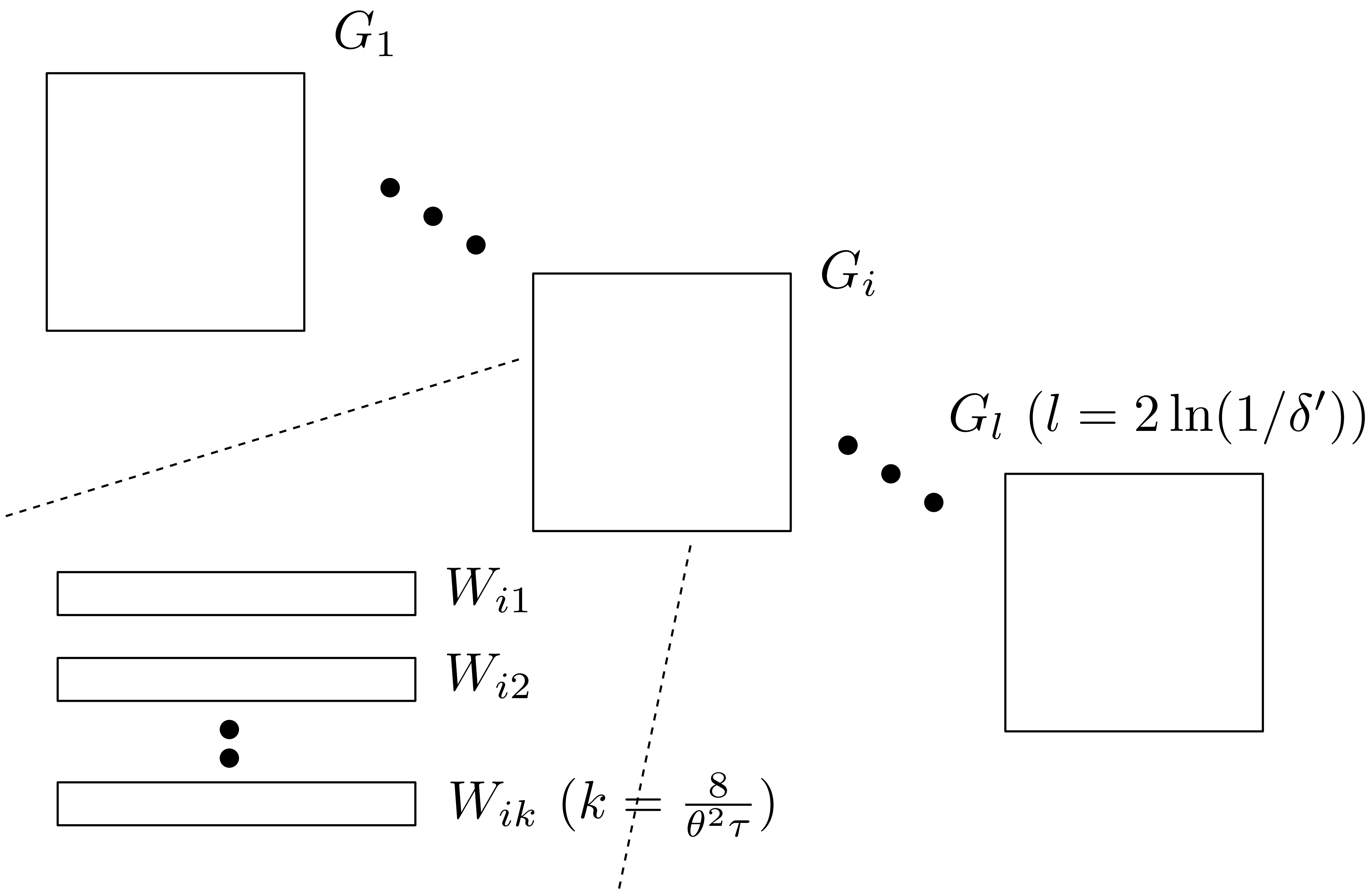
The basic sampling algorithm

- The idea follows from Alon et. al. (JCSS 99) Average-Median



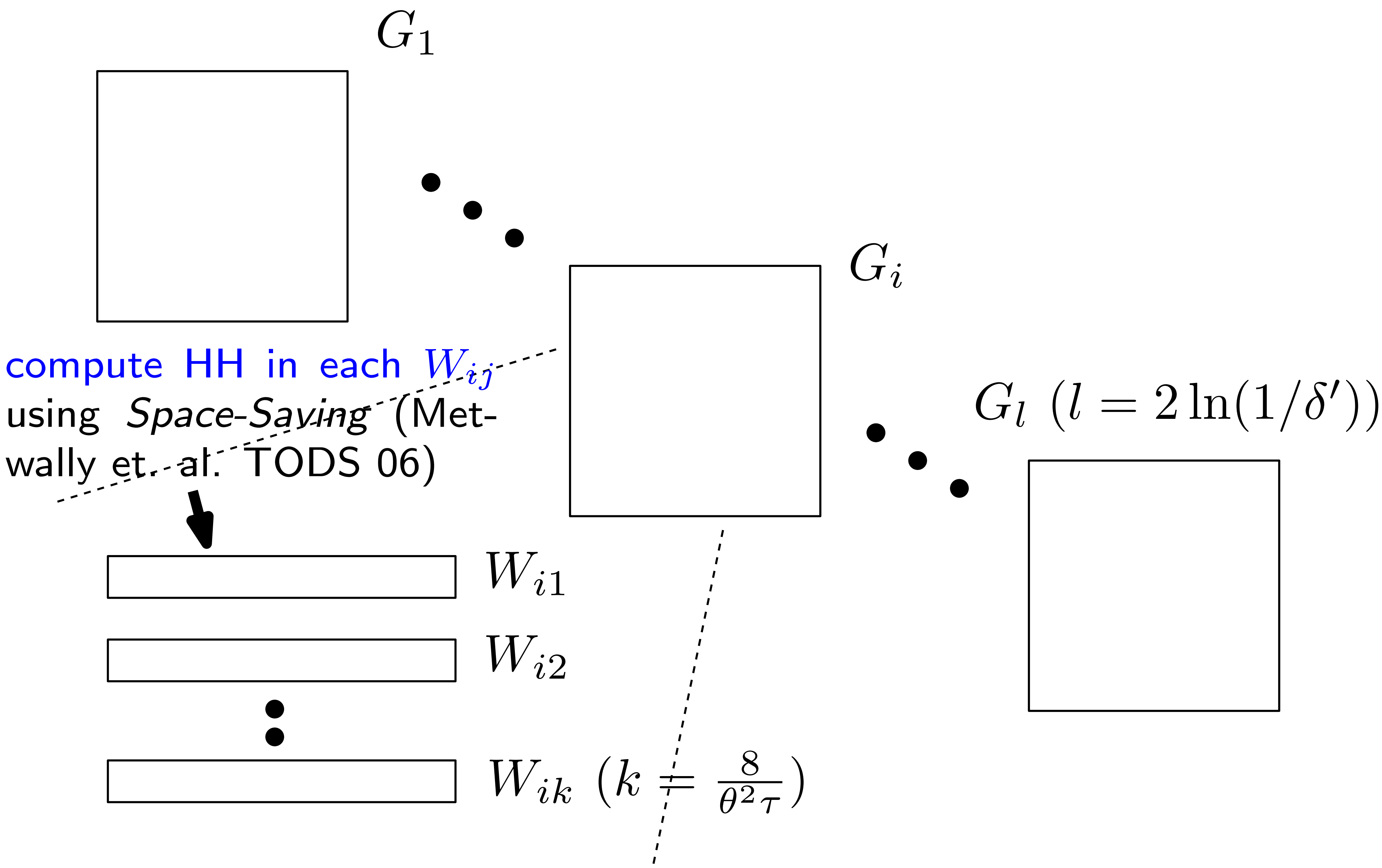
The basic sampling algorithm

- The idea follows from Alon et. al. (JCSS 99) Average-Median



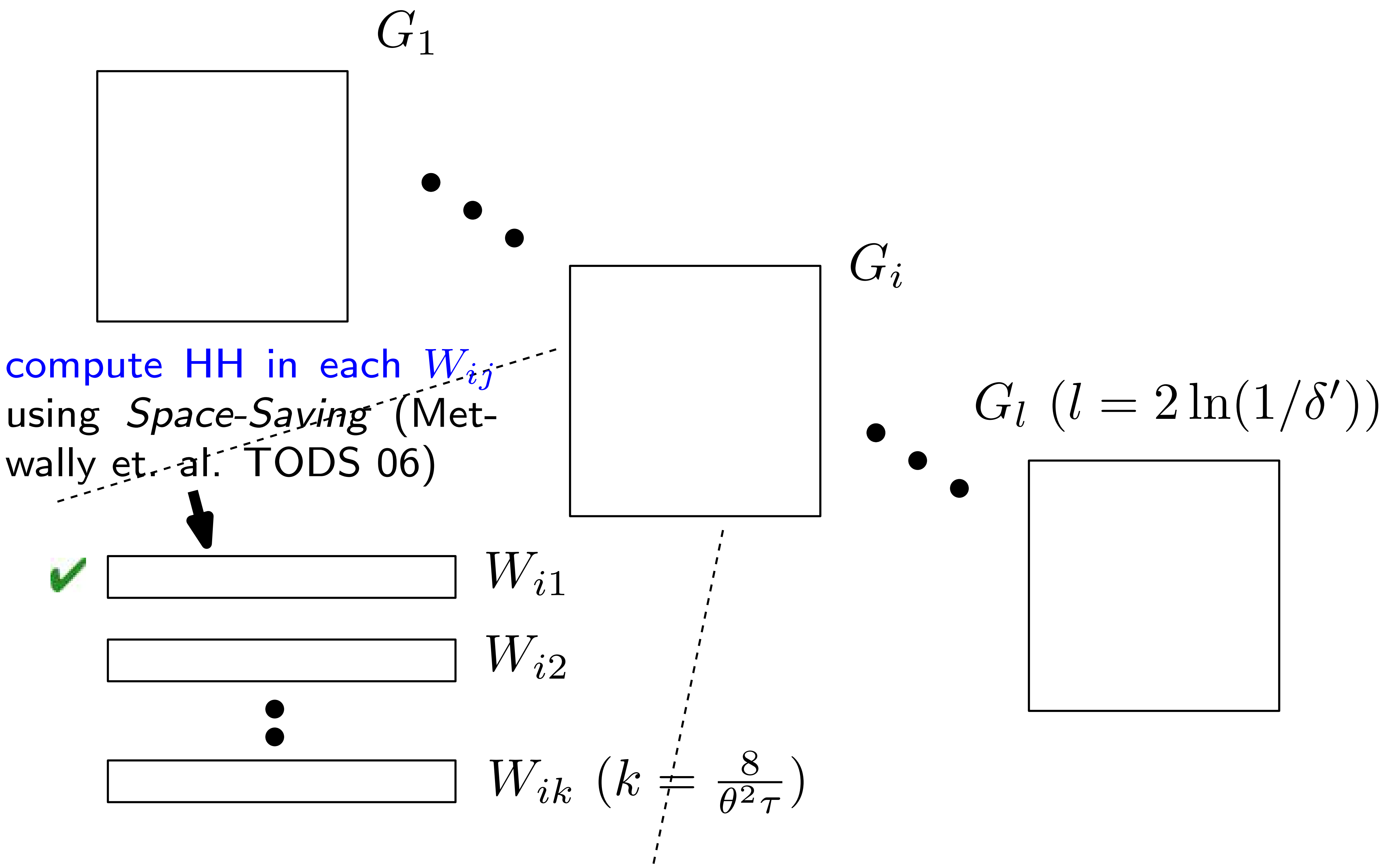
The basic sampling algorithm

- The idea follows from Alon et. al. (JCSS 99) Average-Median



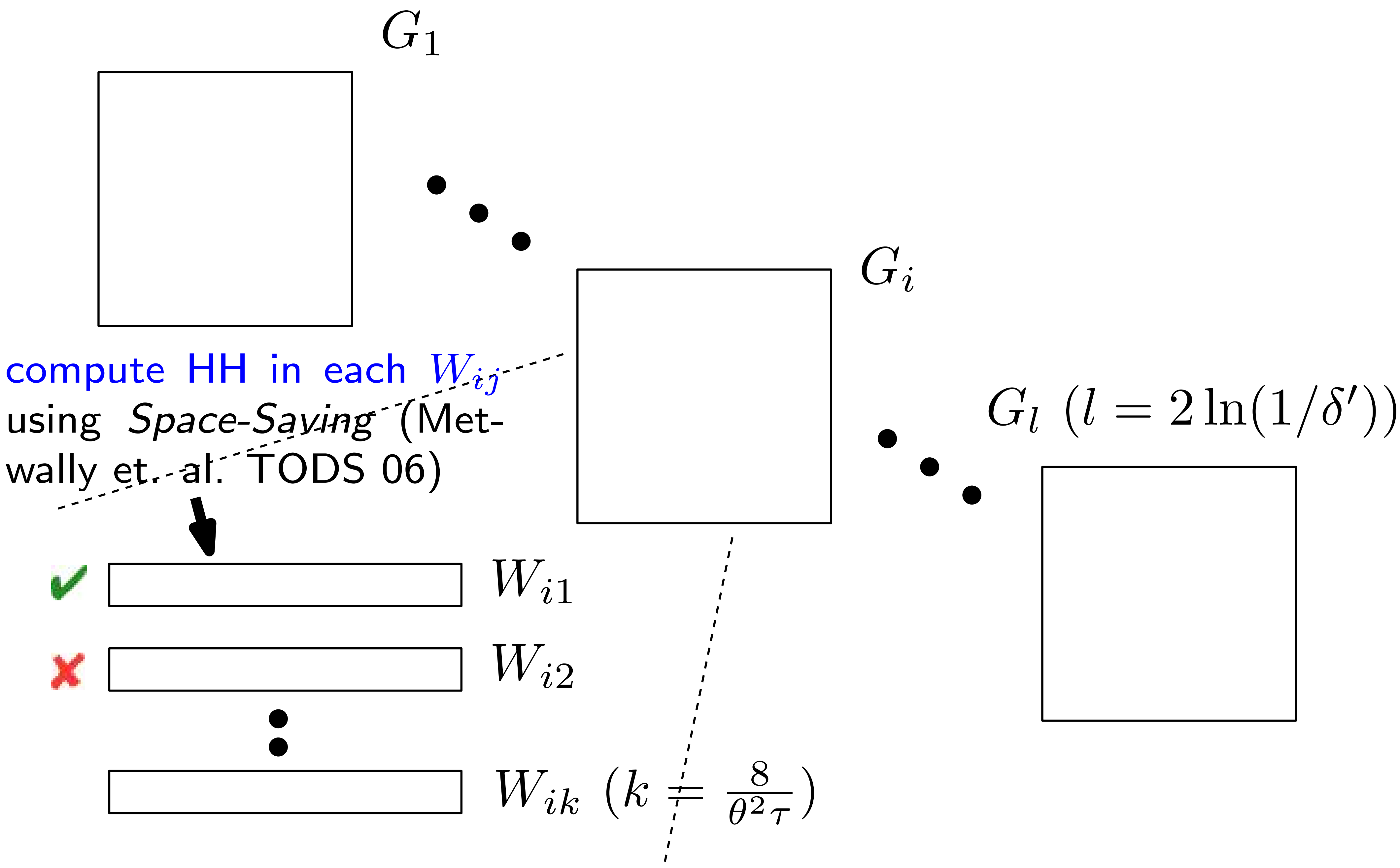
The basic sampling algorithm

- The idea follows from Alon et. al. (JCSS 99) Average-Median



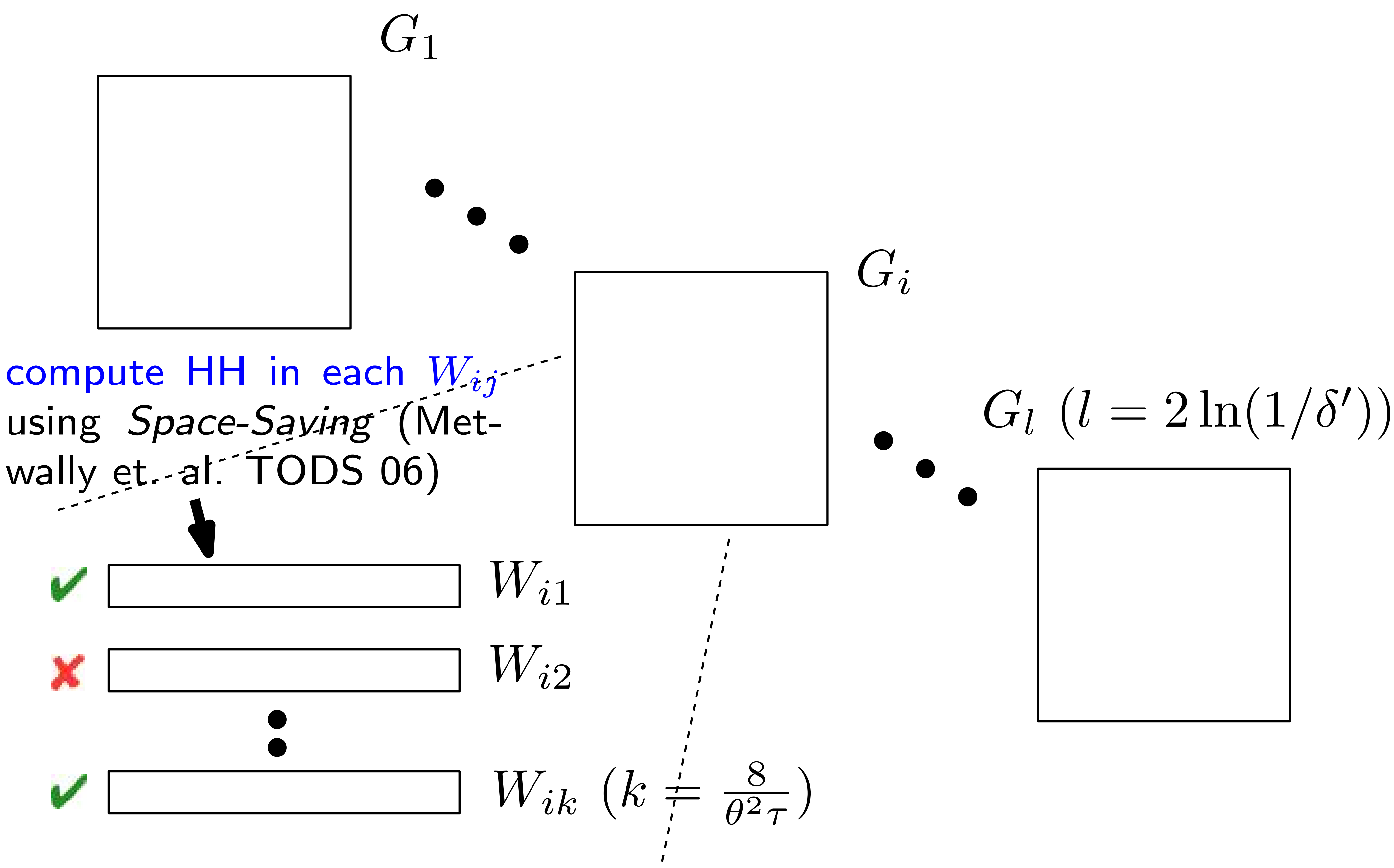
The basic sampling algorithm

- The idea follows from Alon et. al. (JCSS 99) Average-Median



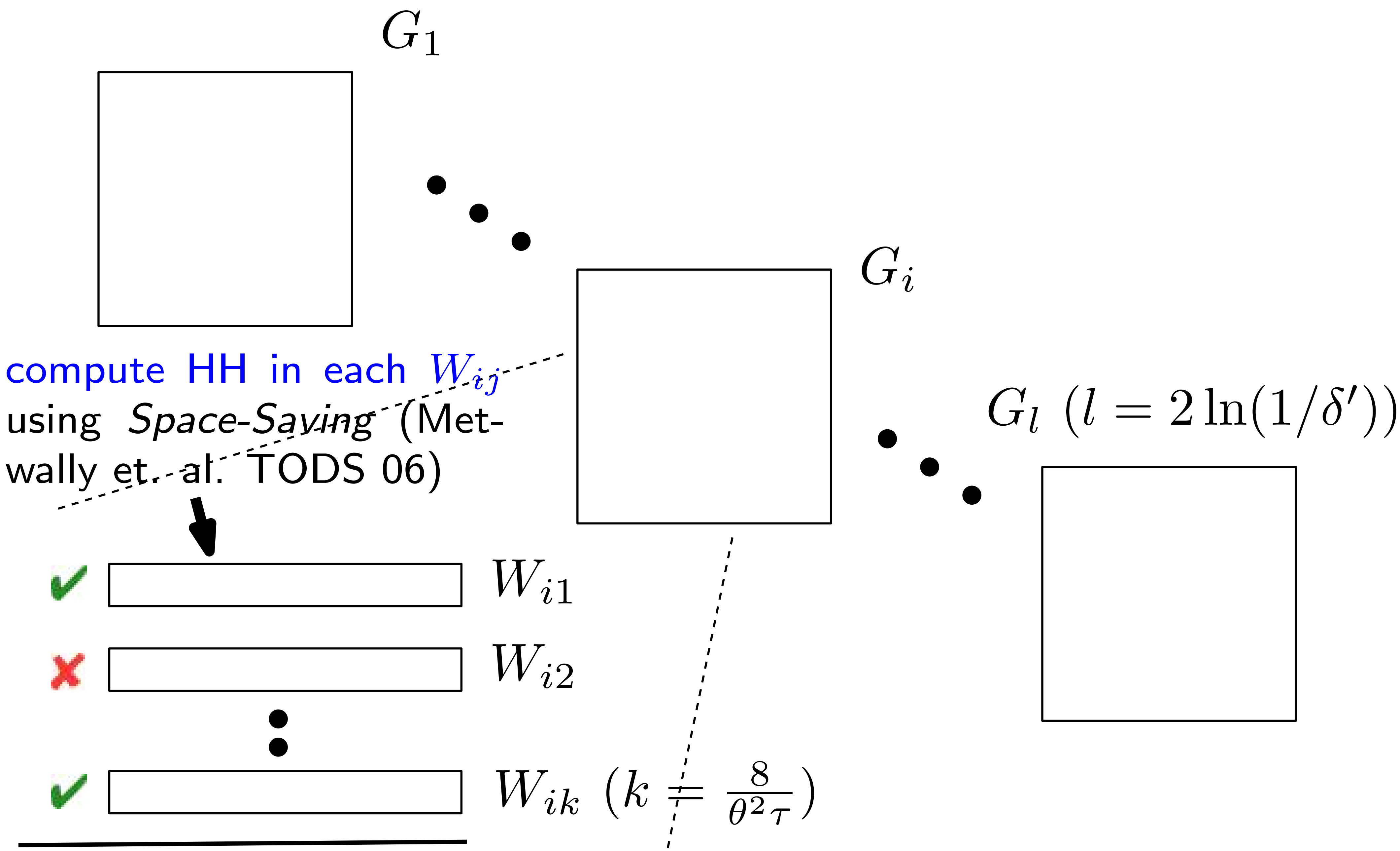
The basic sampling algorithm

- The idea follows from Alon et. al. (JCSS 99) Average-Median



The basic sampling algorithm

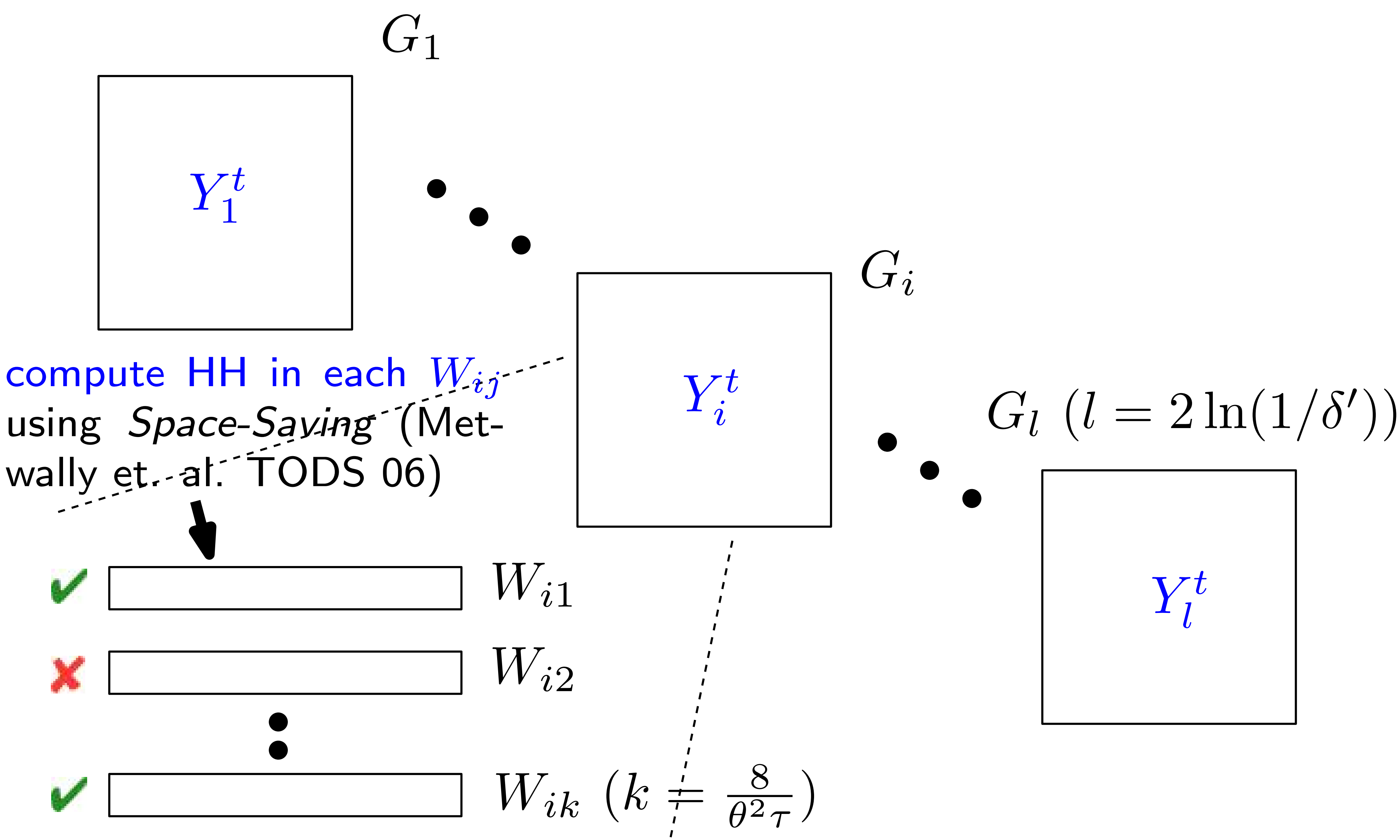
- The idea follows from Alon et. al. (JCSS 99) Average-Median



Y_i^t = # possible worlds in which t is a heavy hitter $/k$

The basic sampling algorithm

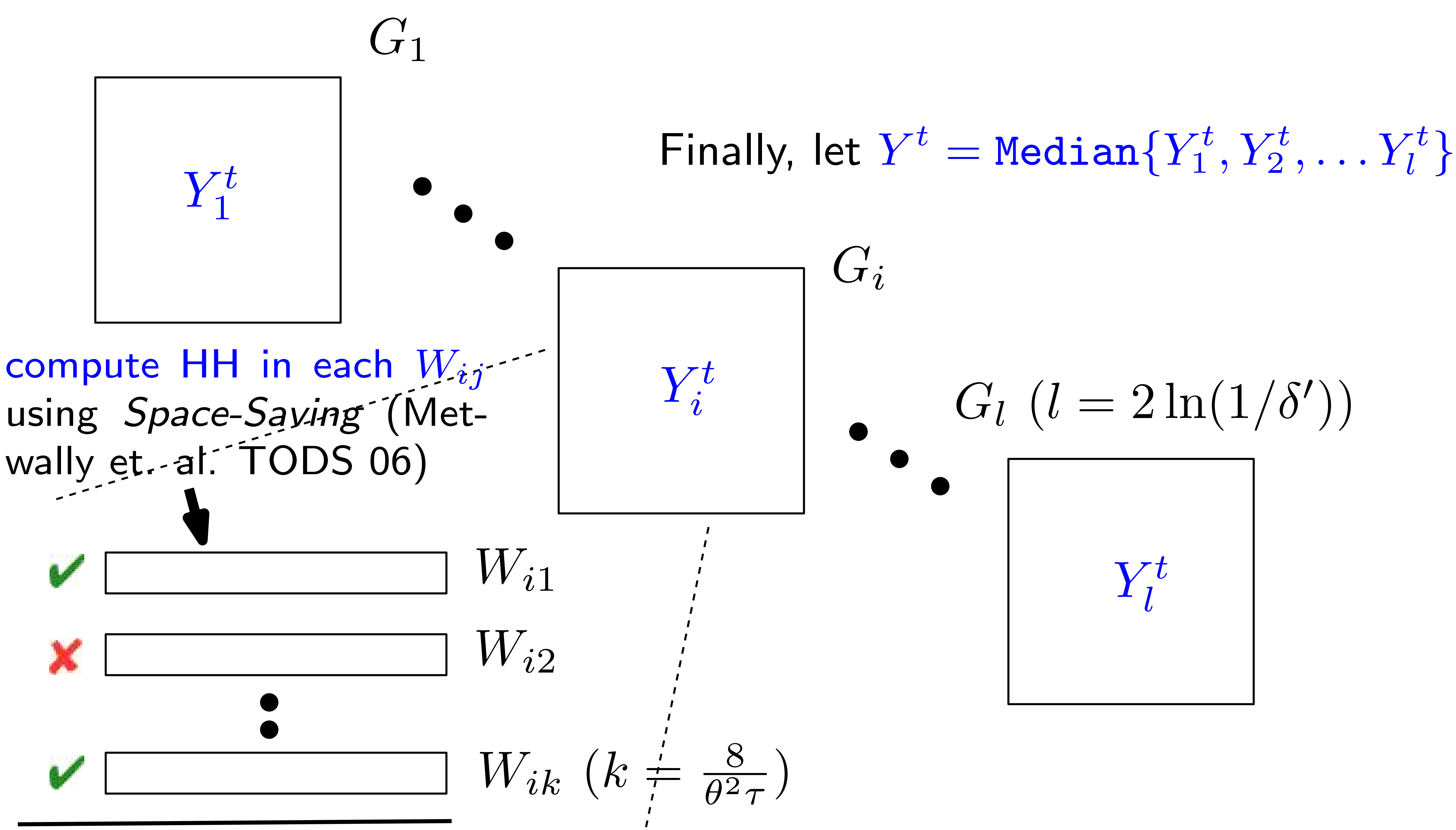
- The idea follows from Alon et. al. (JCSS 99) Average-Median



Y_i^t = # possible worlds in which t is a heavy hitter $/k$

The basic sampling algorithm

- The idea follows from Alon et. al. (JCSS 99) Average-Median



Y_i^t = # possible worlds in which t is a heavy hitter $/k$

The basic sampling algorithm

- $Y^t > (1-\theta/2)\tau \longrightarrow t$ is a (ϕ, τ)-PHH.
  $Y^t \leq (1-\theta/2)\tau \longrightarrow t$ is not a PHH.

The basic sampling algorithm

- $Y^t > (1 - \theta/2)\tau \longrightarrow t$ is a (ϕ, τ)-$\textsc{Phh}$.
  $Y^t \leq (1 - \theta/2)\tau \longrightarrow t$ is not a $\textsc{Phh}$.

- Correct with probability at least $1 - \delta'$ for any particular item t.

The basic sampling algorithm

- $Y^t > (1 - \theta/2)\tau \longrightarrow t$ is a (ϕ, τ)-$\textsc{Phh}$.
  $Y^t \leq (1 - \theta/2)\tau \longrightarrow t$ is not a $\textsc{Phh}$.

- Correct with probability at least $1 - \delta'$ for any particular item t.

- Setting $\delta' = \frac{\phi\tau}{4}\delta$ is enough since we only need to consider at most $\frac{3}{\phi\tau}$ candidates $\textsc{Phh}$, by the Prunning Lemma.

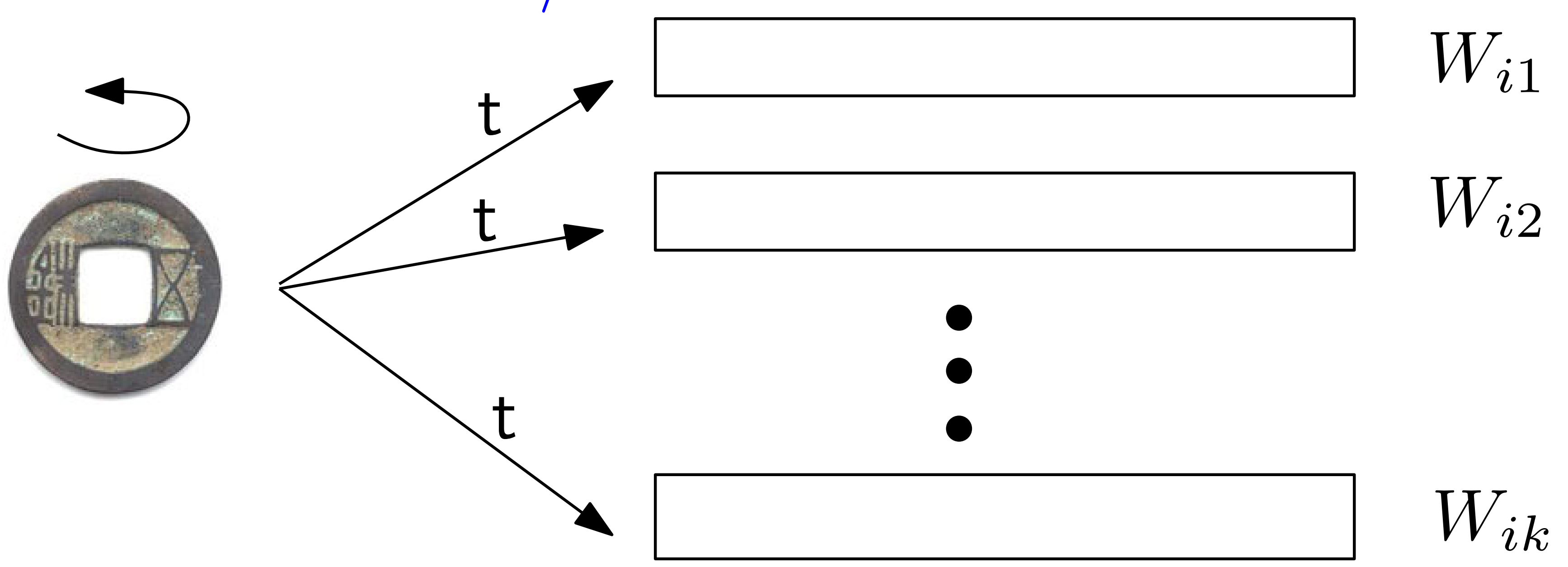
The improved sampling algorithm

- Problem of the basic algorithm: the processing time for each item is too large! $\tilde{O}(\frac{1}{\theta^2\tau})$

The improved sampling algorithm

- Problem of the basic algorithm: the processing time for each item is too large! $\tilde{O}(\frac{1}{\theta^2 \tau})$
- Solution: reduce the sampling rate!

With Pr. $1/k^2$



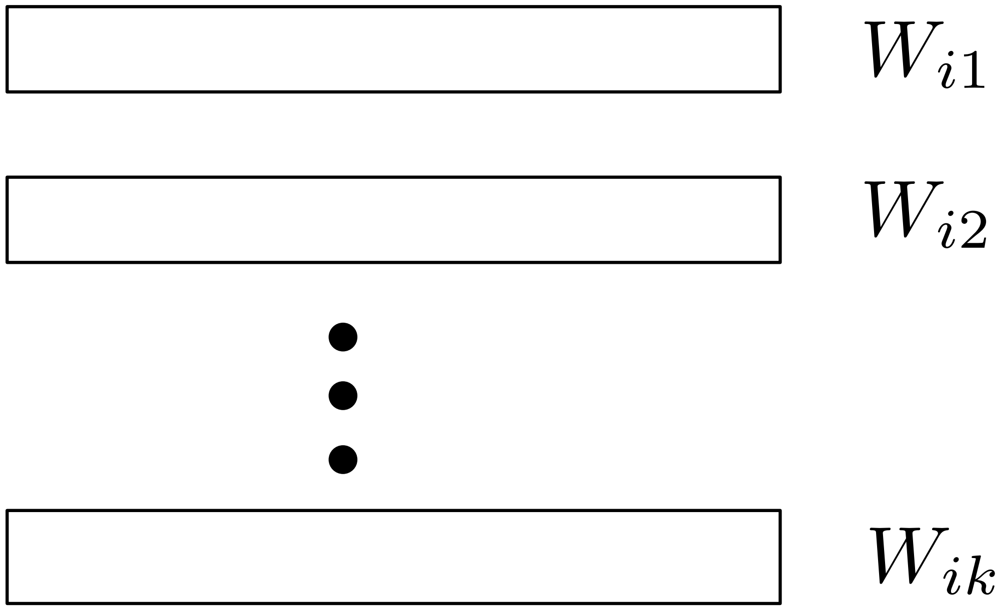
The improved sampling algorithm

- Problem of the basic algorithm: the processing time for each item is too large! $\tilde{O}(\frac{1}{\theta^2 \tau})$

- Solution: reduce the sampling rate!

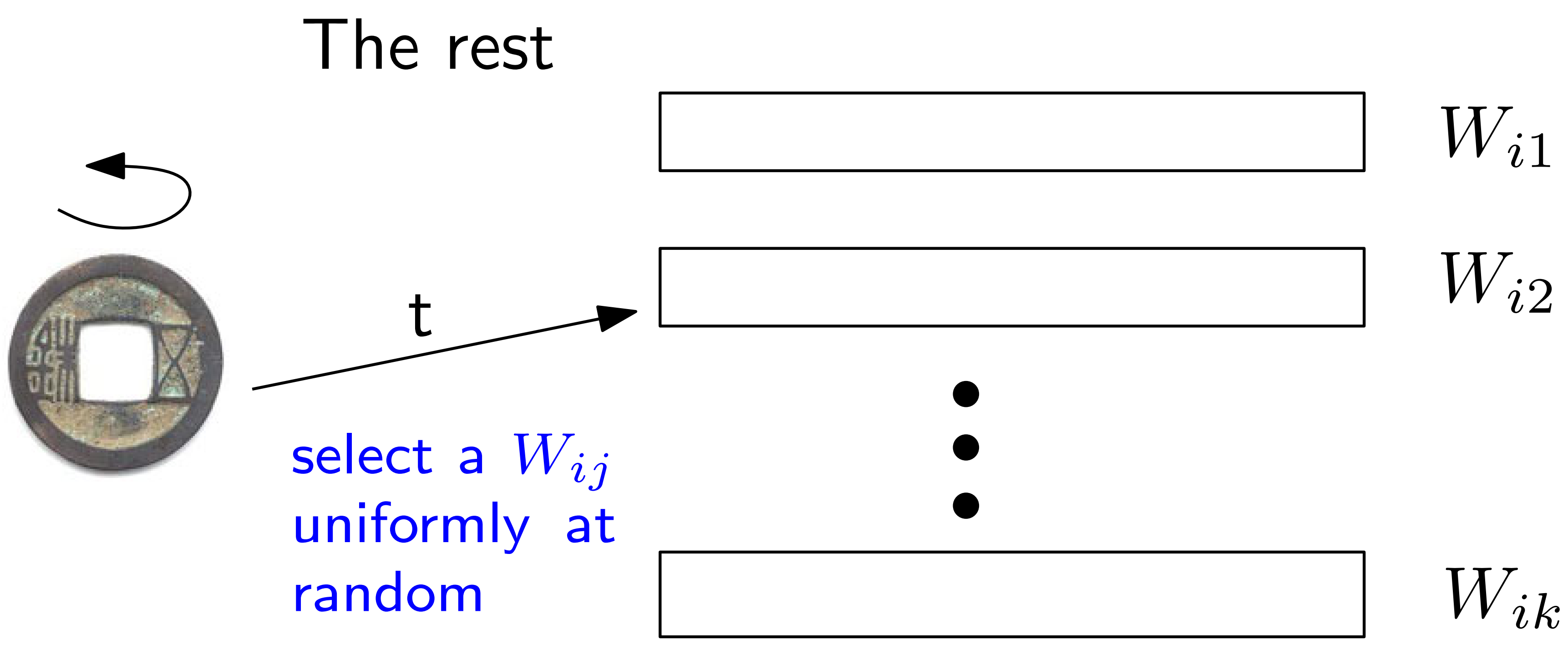
With Pr. $(k-1)/k^2$

don't send



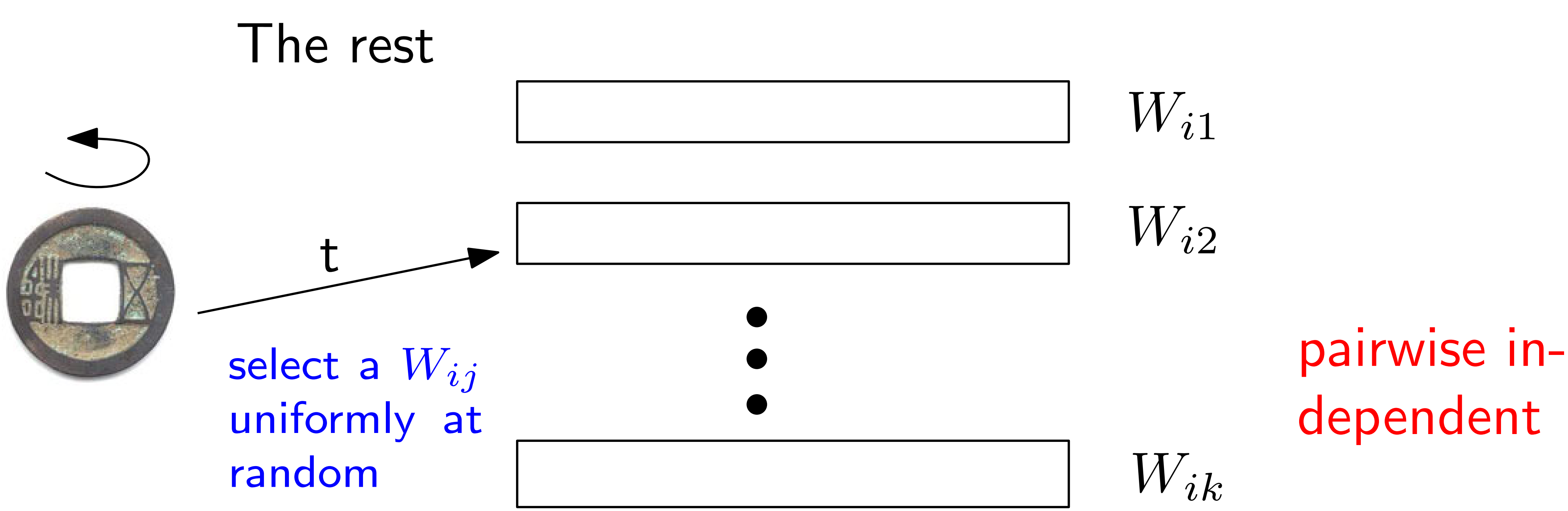
The improved sampling algorithm

- Problem of the basic algorithm: the processing time for each item is too large! $\tilde{O}(\frac{1}{\theta^2 \tau})$
- Solution: reduce the sampling rate!



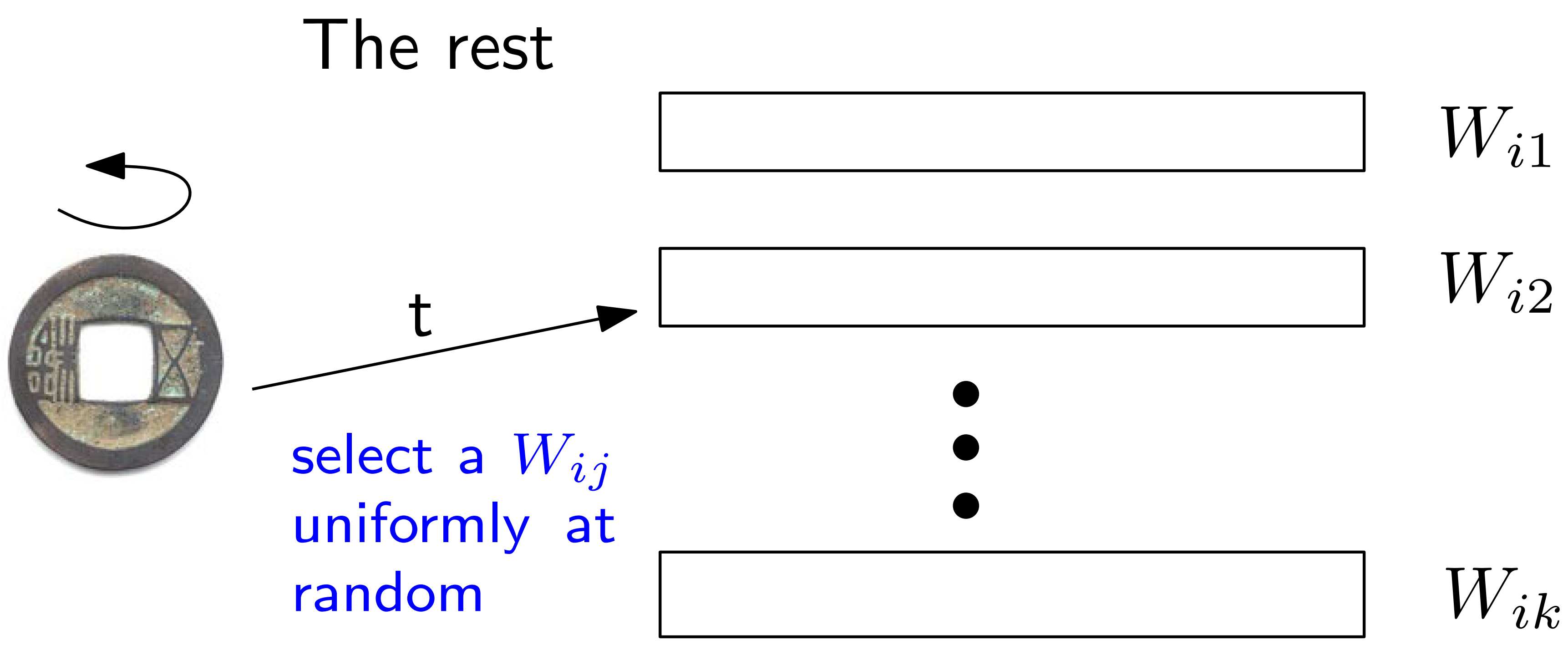
The improved sampling algorithm

- Problem of the basic algorithm: the processing time for each item is too large! $\tilde{O}(\frac{1}{\theta^2 \tau})$
- Solution: reduce the sampling rate!



The improved sampling algorithm

- Problem of the basic algorithm: the processing time for each item is too large! $\tilde{O}(\frac{1}{\theta^2 \tau})$
- Solution: reduce the sampling rate!



pairwise in-dependent

- Now processing time per x-tuple: $O(\log(\frac{1}{\delta\phi\tau\epsilon}))$.

Experiments - the data sets

- Data sets.

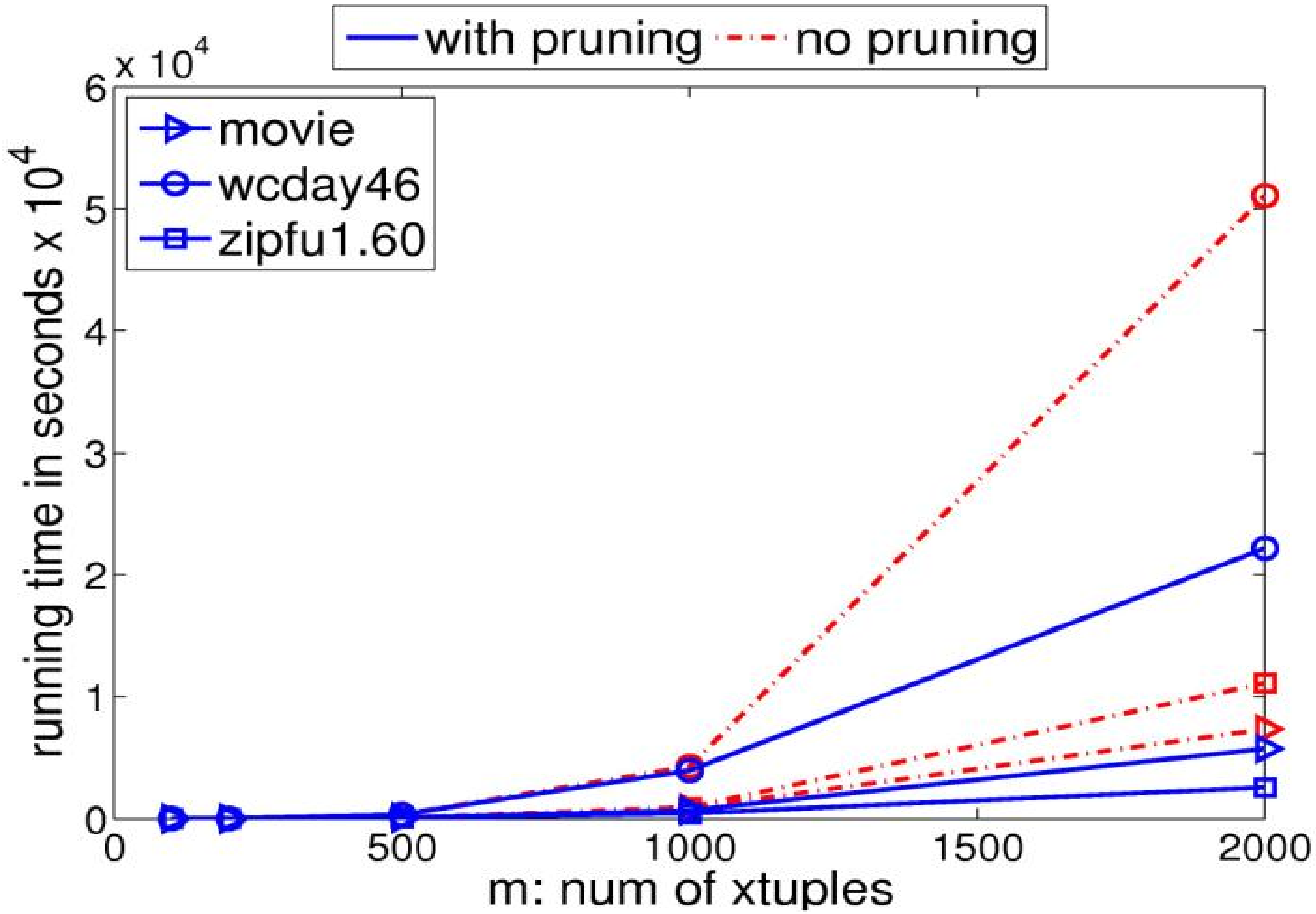
  movie from the MystiQ project; has a total of approximately $100,000$ x-tuples, most of which have only one alternative, but some have a few.

  It contains probabilistic movie records reflecting the matching probability as a result of data integration from multiple sources.

  wcday46
  zipfu1.60

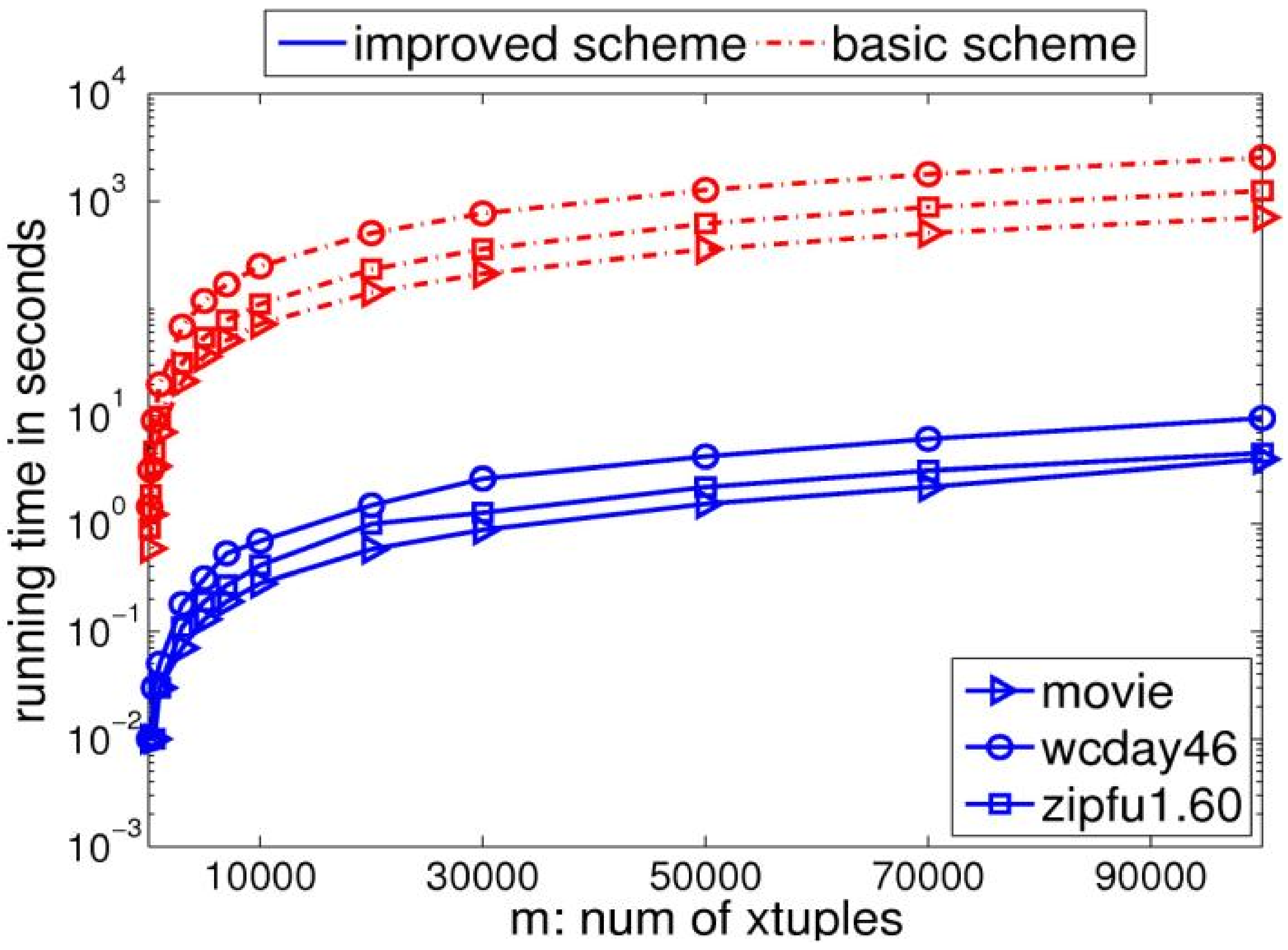
Experiments - the power of prunning

Effectiveness of the pruning lemma, where for skewed data sets, more than 90% of the items are pruned.



Experiments - basic, improved streaming algorithm

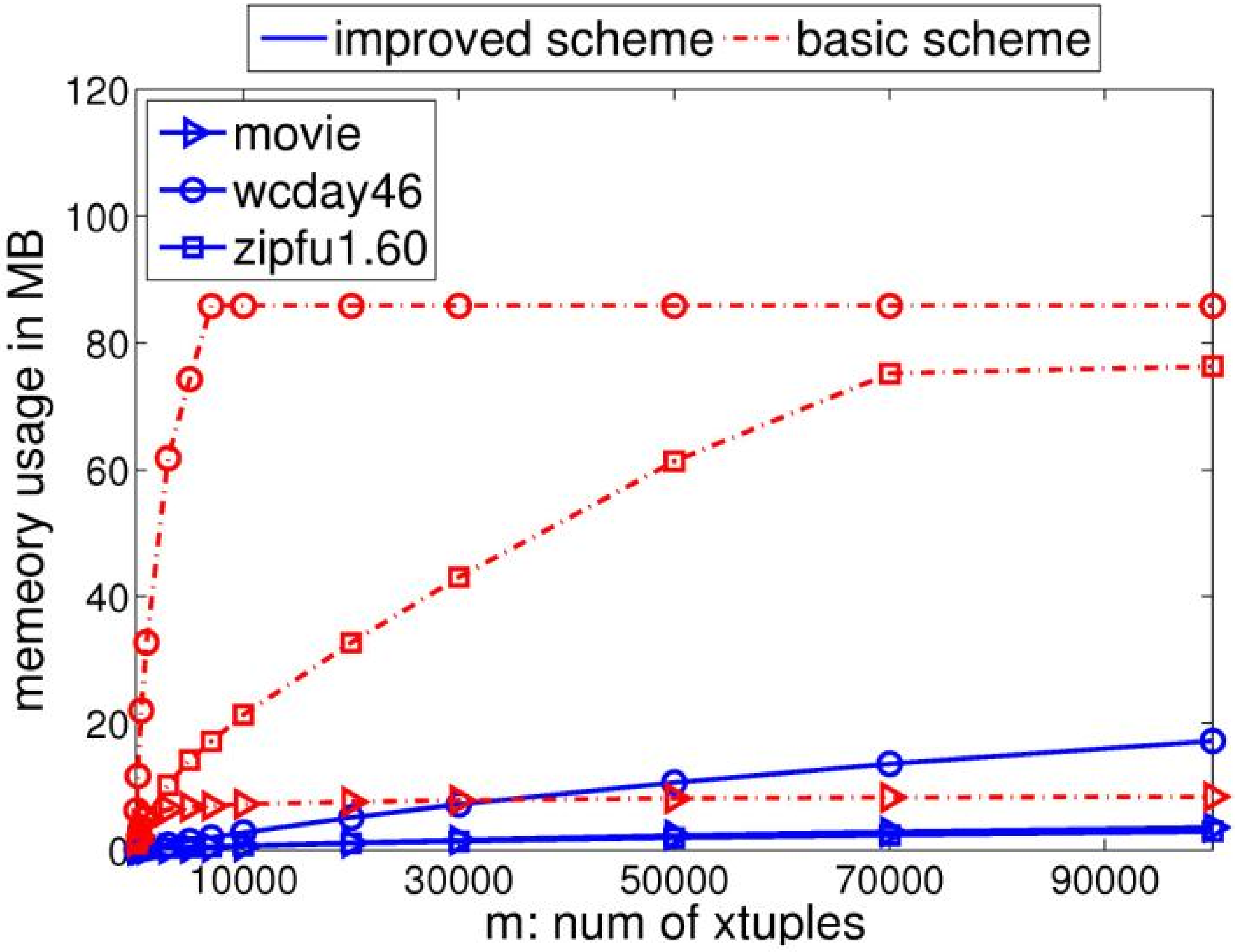
Varying m: $\phi = 0.01$, $\tau = 0.8$, $\delta = 0.05$, $\theta = 0.05$, $\epsilon = 0.001$.



running time

Experiments - basic, improved streaming algorithm

Varying m: $\phi = 0.01$, $\tau = 0.8$, $\delta = 0.05$, $\theta = 0.05$, $\epsilon = 0.001$.



memory usage

Conclusion

- We have
  - formalized the notion of probabilistic heavy hitters following the commonly adopted possible world query semantics in uncertain databases.
  - presented efficient algorithms with theoretical guarantees for both offline and streaming data, under the widely adopted x-relation model.
- Future work includes handling distributed data, and more interestingly, supporting other uncertain data models.

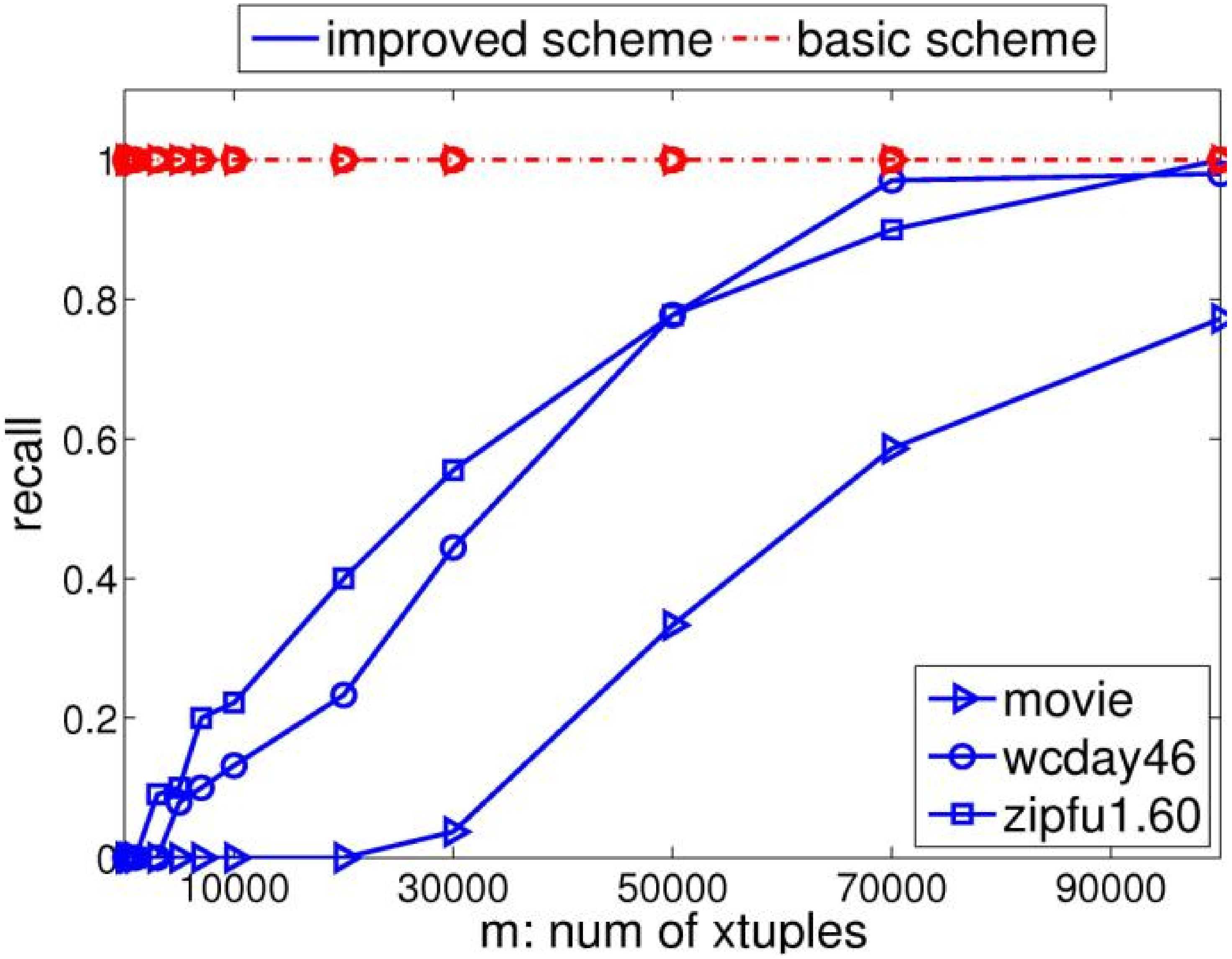
The End

THANK YOU

Q and A

Experiments - basic, improved streaming algorithm

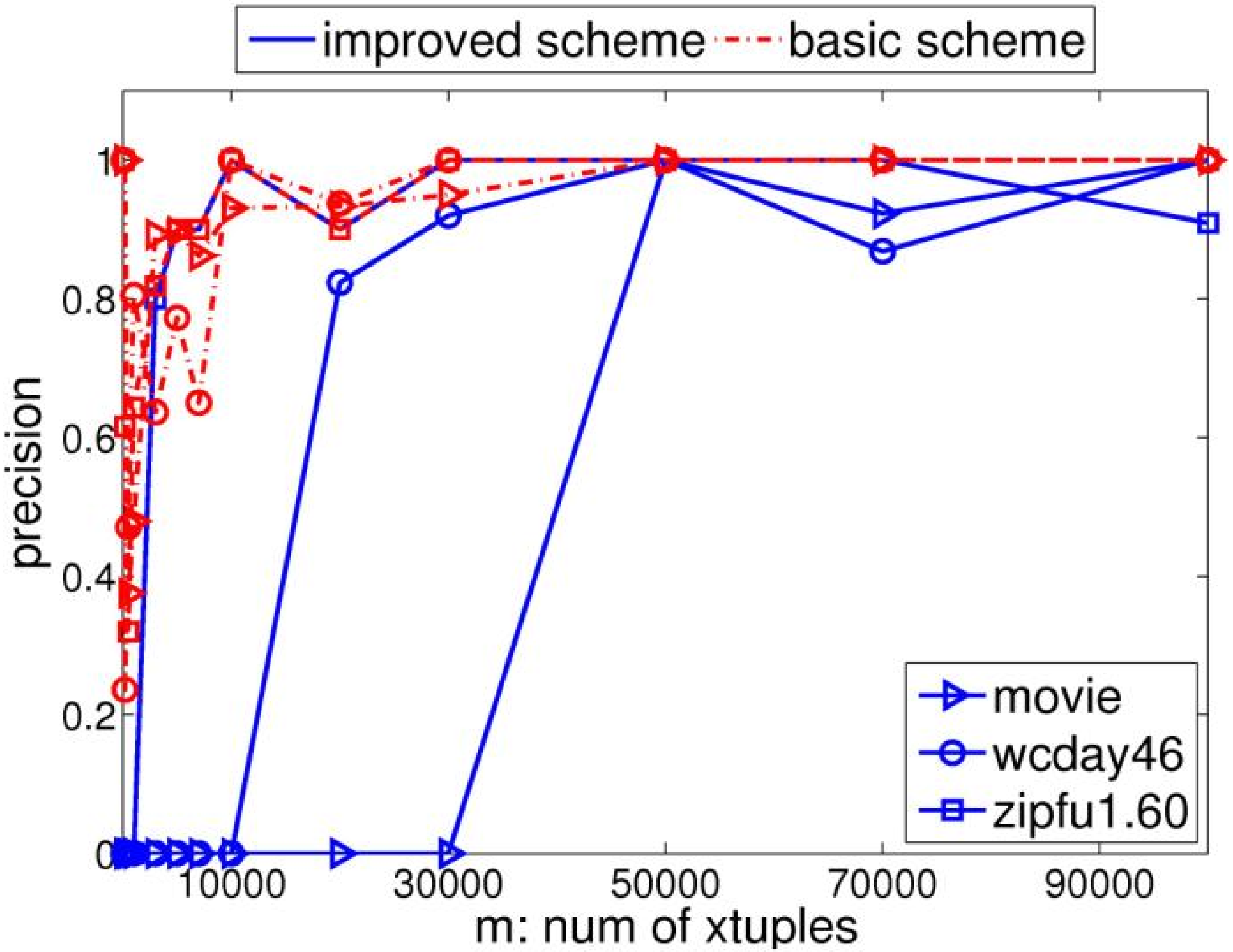
Varying m: $\phi = 0.01$, $\tau = 0.8$, $\delta = 0.05$, $\theta = 0.05$, $\epsilon = 0.001$.



recall

Experiments - basic, improved streaming algorithm

Varying m: $\phi = 0.01$, $\tau = 0.8$, $\delta = 0.05$, $\theta = 0.05$, $\epsilon = 0.001$.



precision

Experiments - generalized algorithm

Tradeoff in cost/accuracy, varying s, $\delta = 0.05$, $\theta = 0.05$, $\phi = 0.01$, $\tau = 0.8$, $\epsilon = 0.001$.

For s/k as small as 0.05, its accuracy is already very close to perfect. 20-fold speedup from the basic scheme!