Approximate String Search in Spatial Databases

Bin Yao
Florida State University

Feifei Li
Florida State University

Marios Hadjieleftheriou
AT&T Labs Research

Kun Hou
Florida State University

Introduction

- Approximate string search is important
  - data cleaning
  - data integration
  - online search engine

Introduction

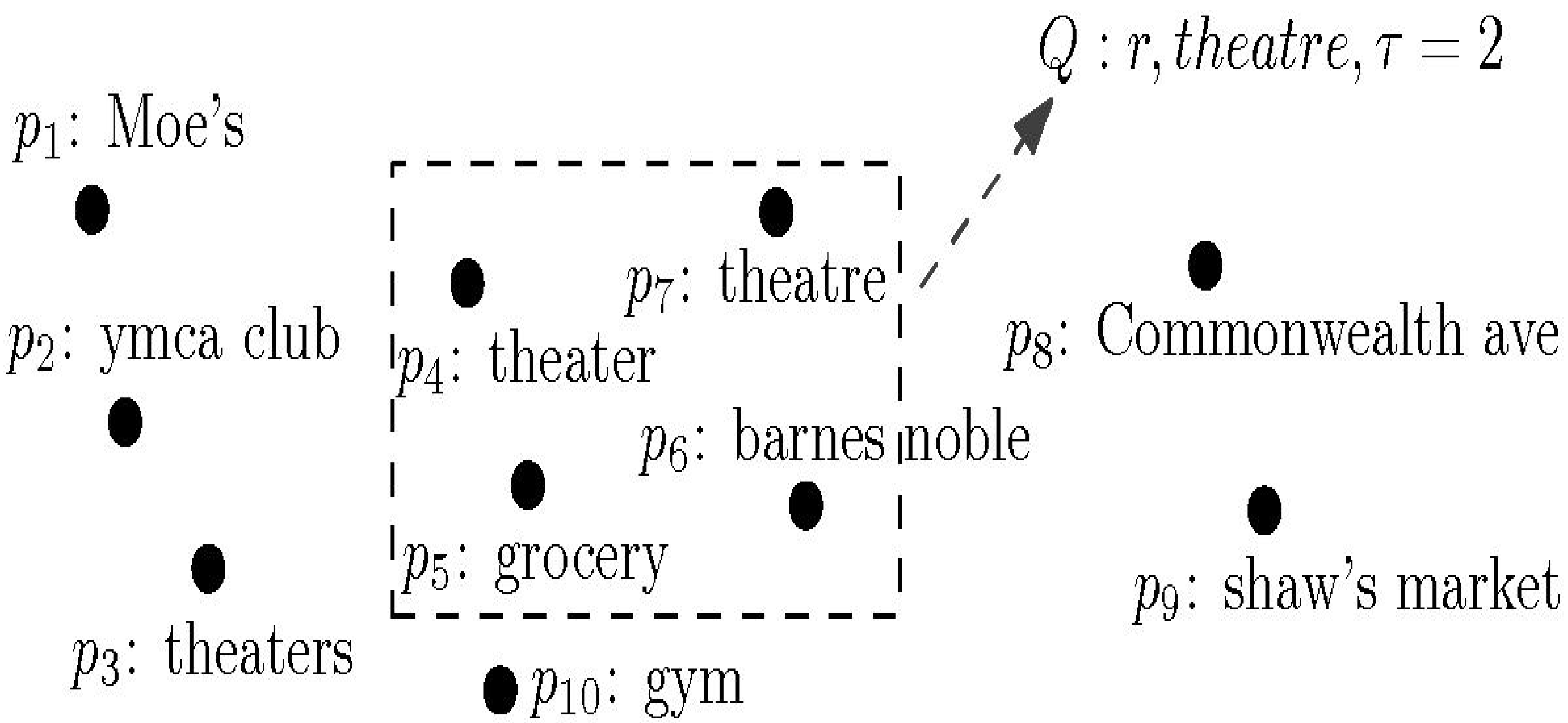
- Approximate string search is important
  - data cleaning
  - data integration
  - online search engine
- Also, spatial database
  - map service
  - strategic planning of resources

Introduction

- Approximate string search is important
  - data cleaning
  - data integration
  - online search engine
- Also, spatial database
  - map service
  - strategic planning of resources

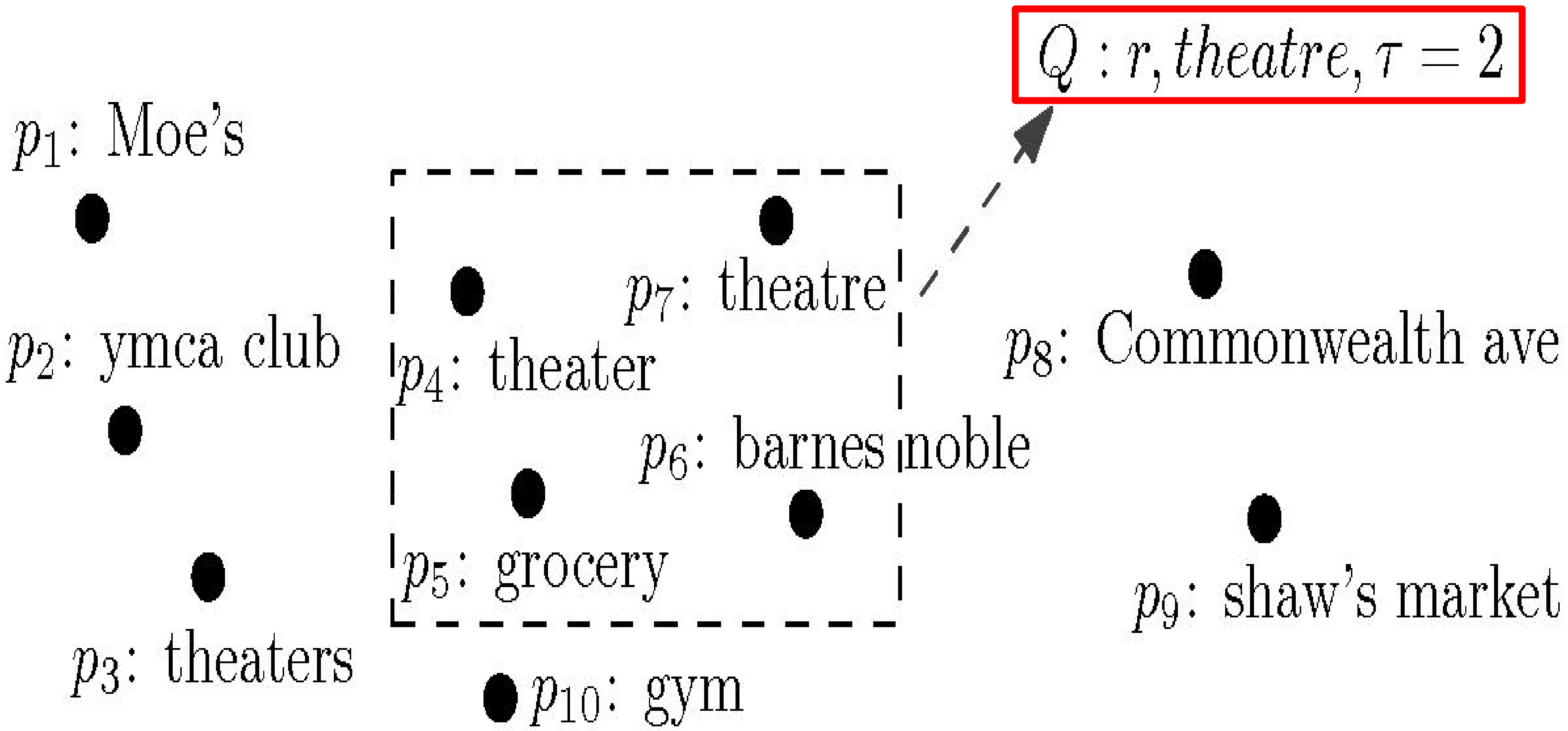
- Our work: the approximate string search in spatial database (*Spatial Approximate String* (SAS) queries).

Example of a SAS range query



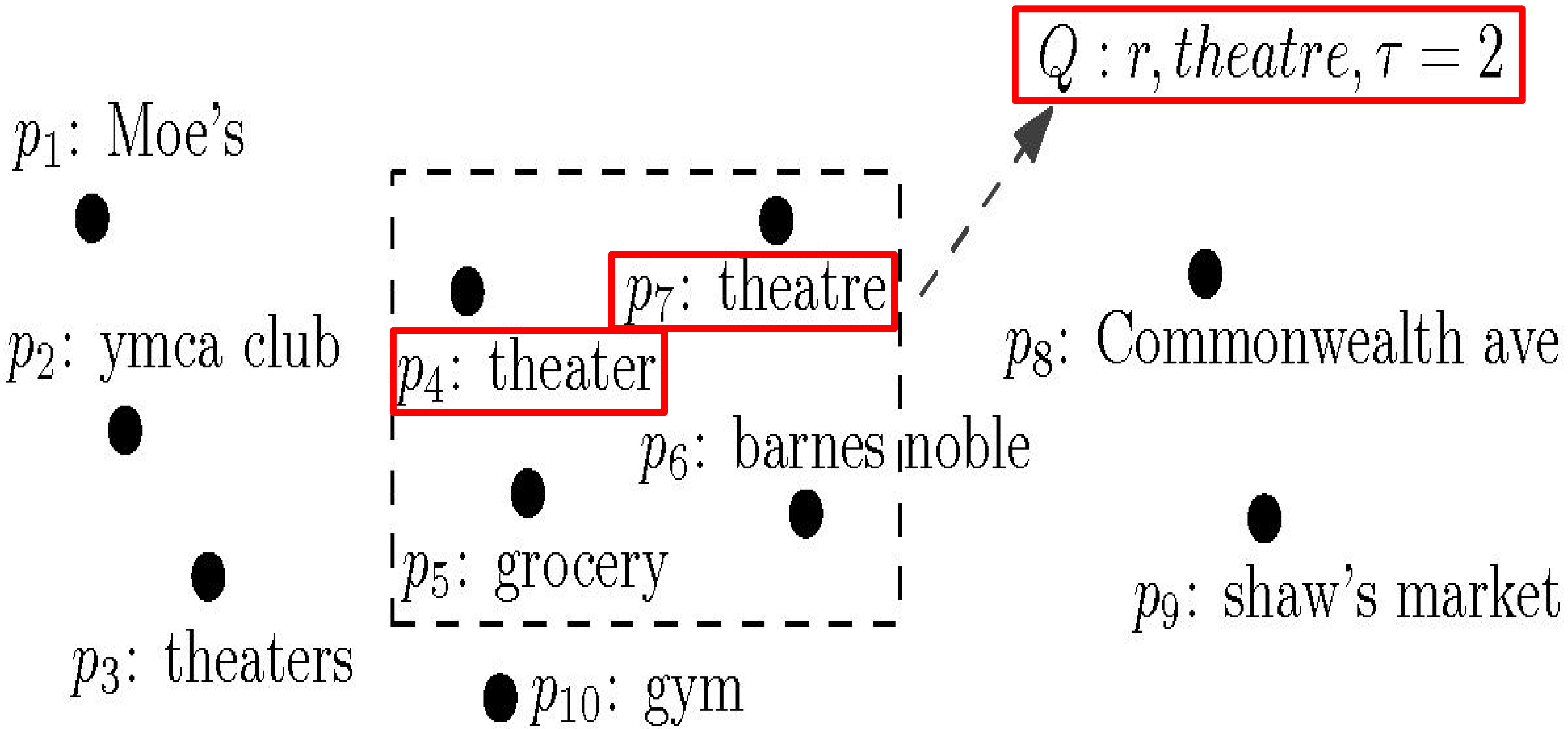
String similarity metric: edit distance with threshold τ.

Example of a SAS range query



String similarity metric: edit distance with threshold τ.

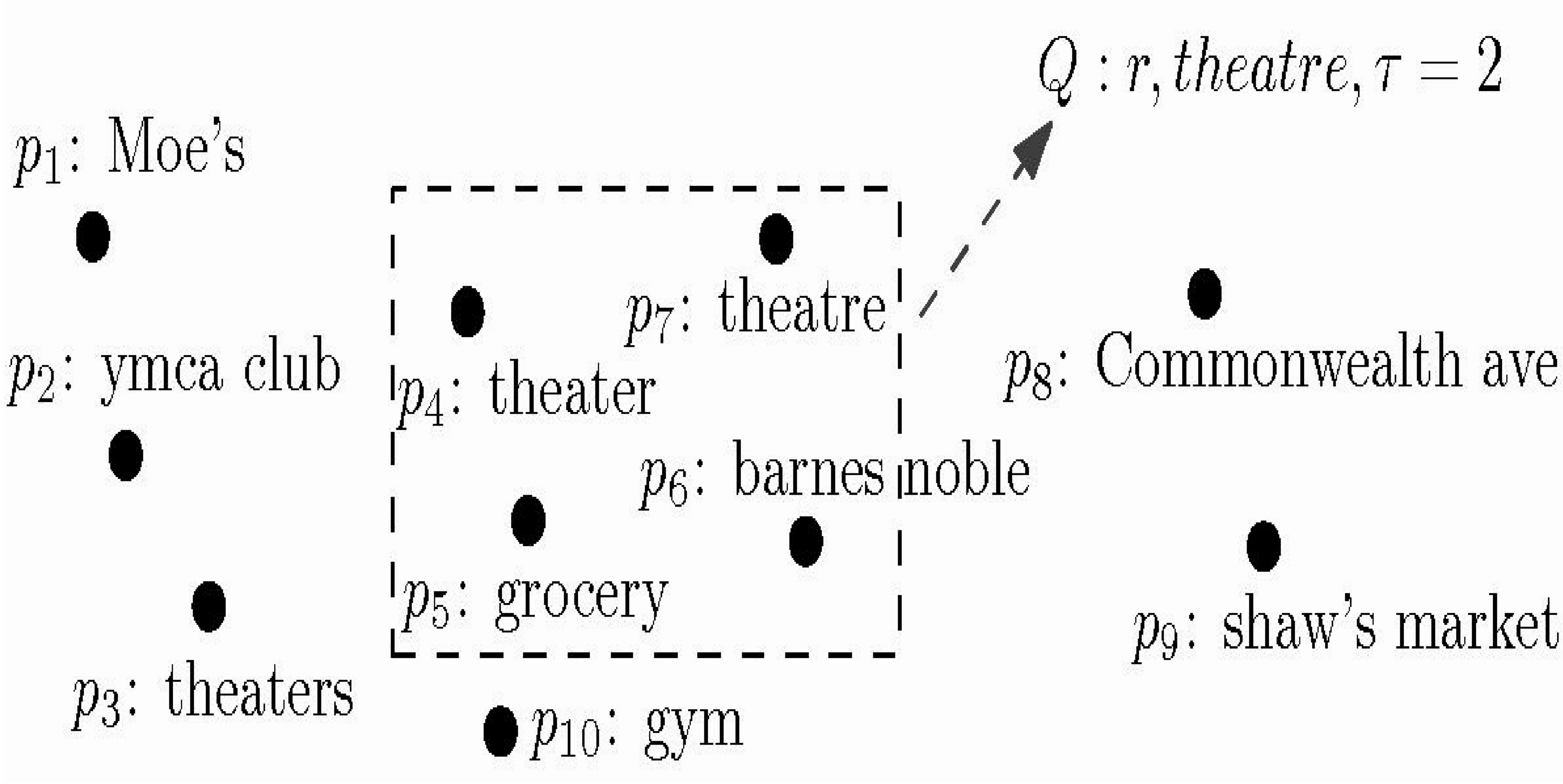
Example of a SAS range query



String similarity metric: edit distance with threshold τ.

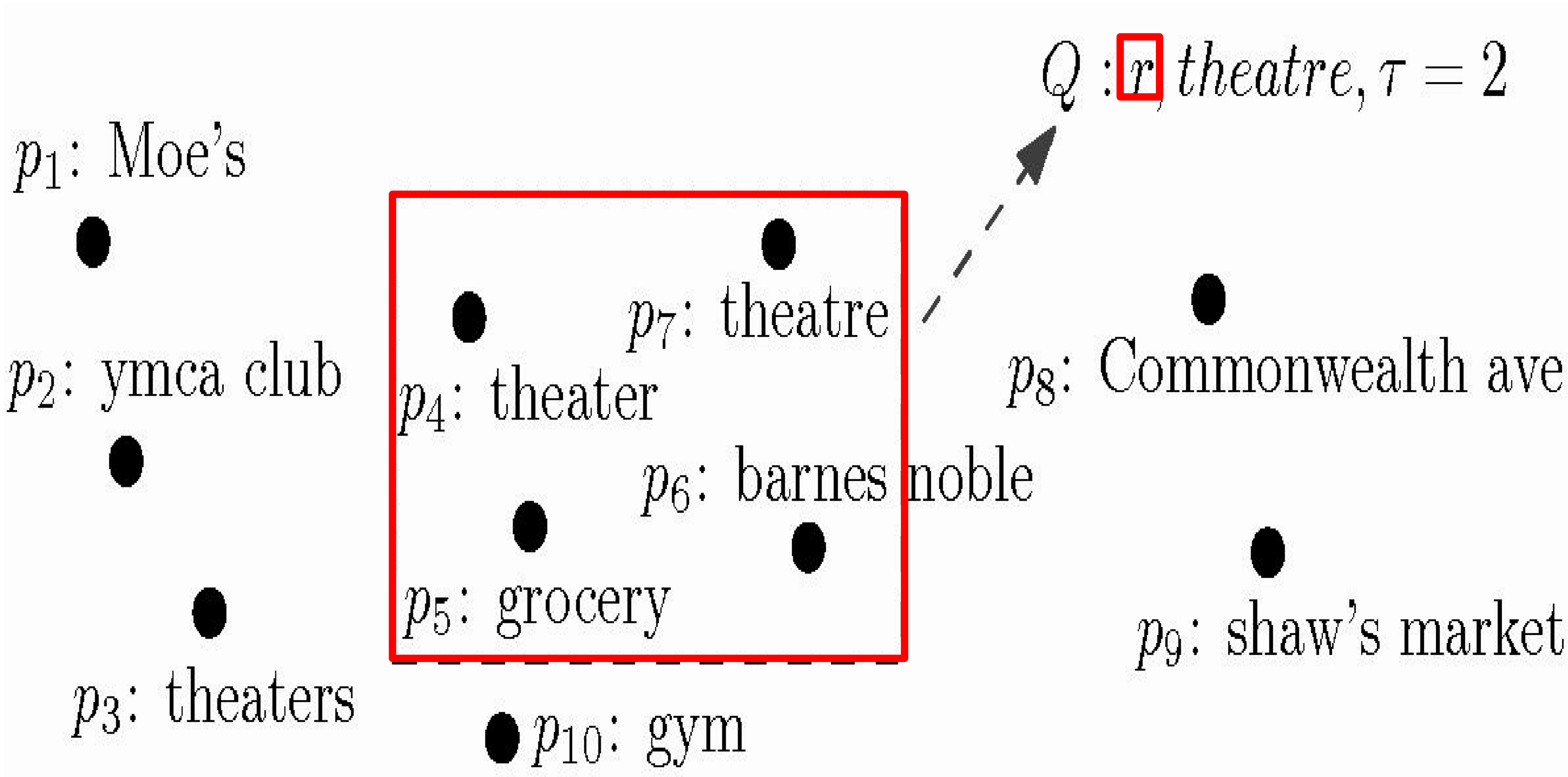
A straightforward solution

R-tree solution:



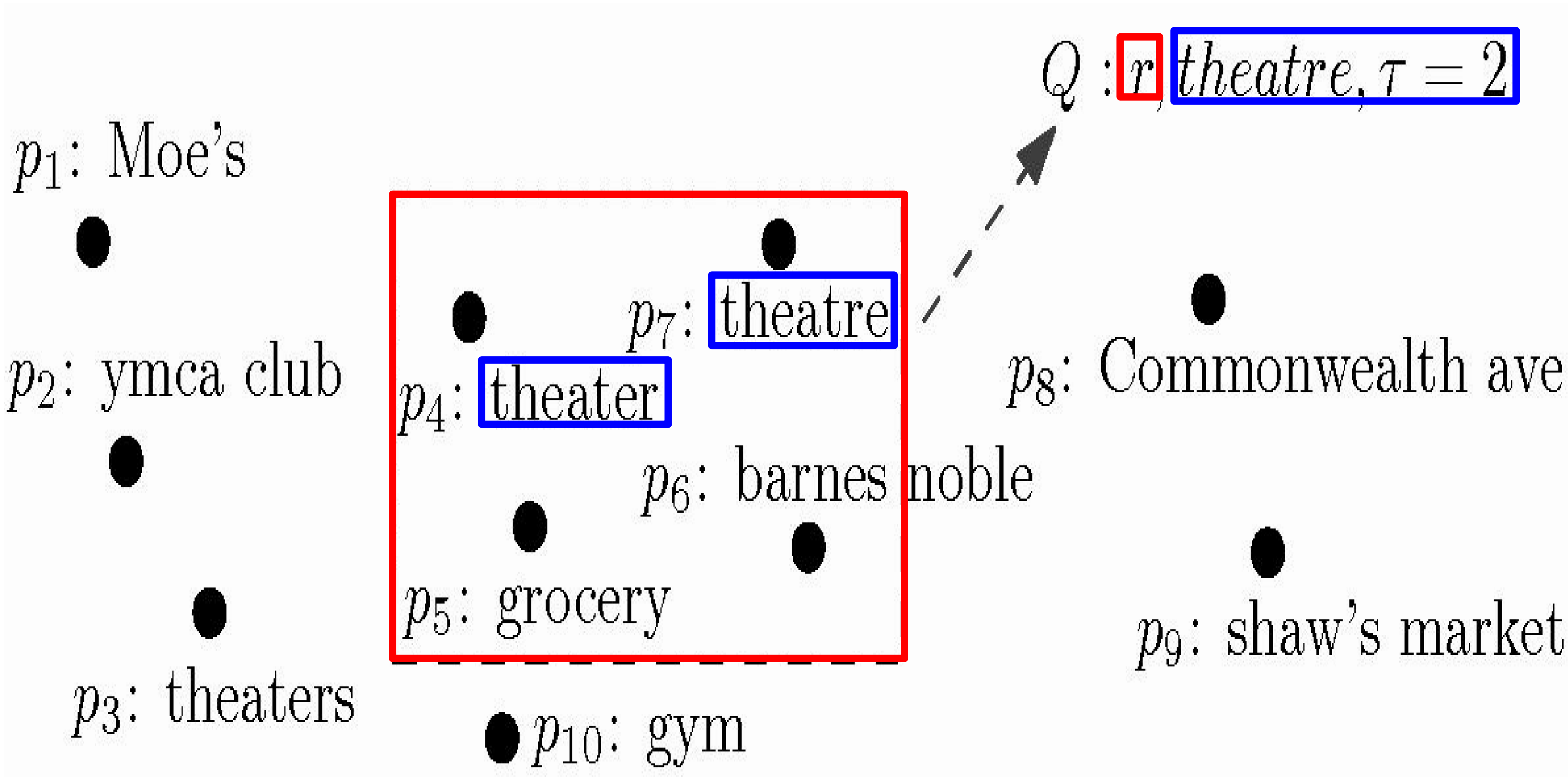
A straightforward solution

R-tree solution:



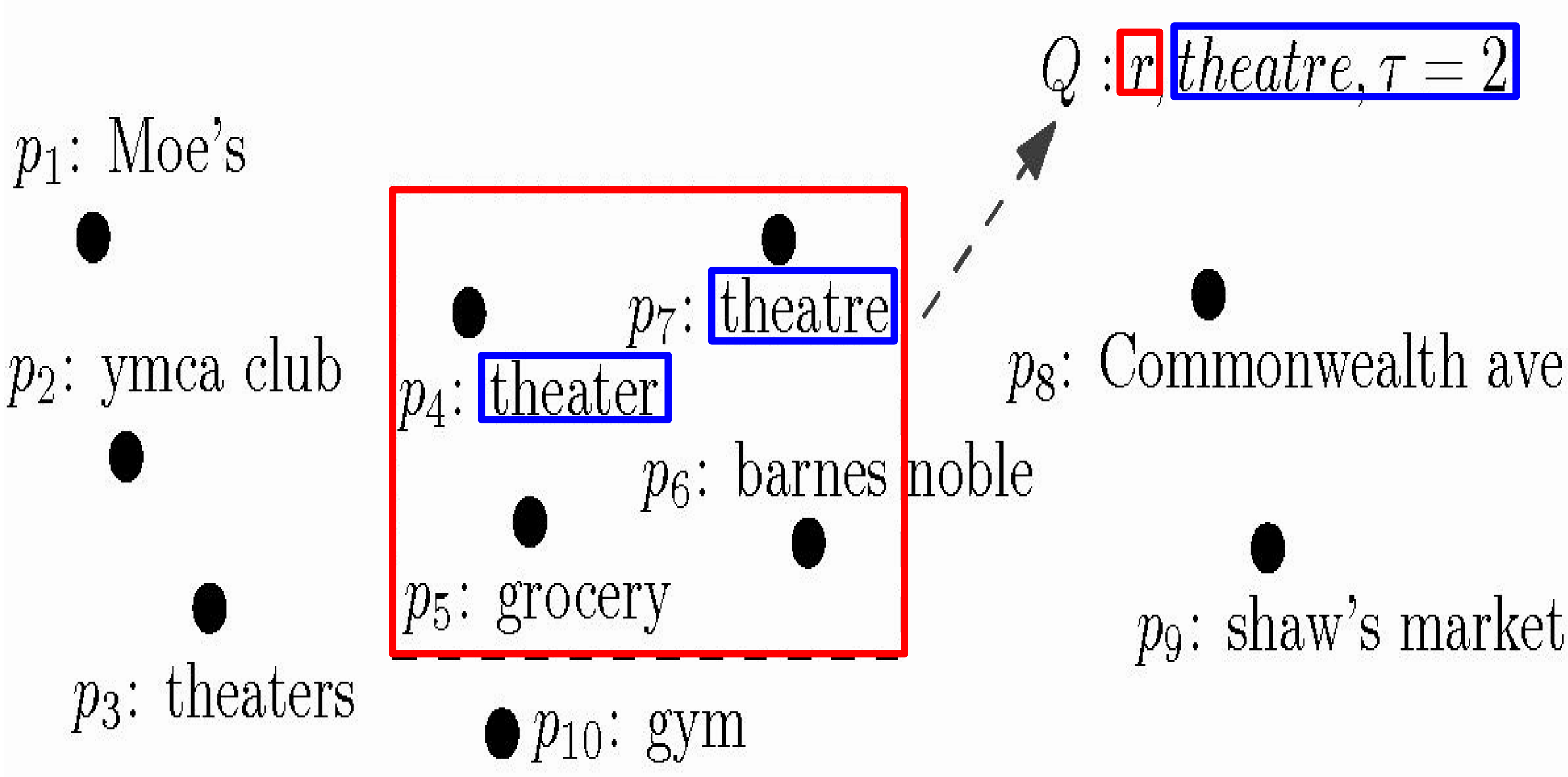
A straightforward solution

R-tree solution:



A straightforward solution

R-tree solution:



Problem: only utilize the spatial dimension for the pruning.

Background: q-grams based edit distance pruning

- Lemma: For strings σ_1 and σ_2 of length $|\sigma_1|$ and $|\sigma_2|$, if $\varepsilon(\sigma_1, \sigma_2) = \tau$, then $|G_{\sigma_1} \cap G_{\sigma_2}| \geq \max(|\sigma_1|, |\sigma_2|) - 1 - (\tau - 1) * q$.

Background: q-grams based edit distance pruning

- Lemma: For strings σ_1 and σ_2 of length $|\sigma_1|$ and $|\sigma_2|$, if $\varepsilon(\sigma_1, \sigma_2) = \tau$, then $|G_{\sigma_1} \cap G_{\sigma_2}| \geq \max(|\sigma_1|, |\sigma_2|) - 1 - (\tau - 1) * q$.

- Example: $q = 2, \tau = 2$
  $\sigma_1 : theatre$; $\sigma_2 : theater$.

Background: q-grams based edit distance pruning

- Lemma: For strings σ_1 and σ_2 of length $|\sigma_1|$ and $|\sigma_2|$, if $\varepsilon(\sigma_1, \sigma_2) = \tau$, then $|G_{\sigma_1} \cap G_{\sigma_2}| \geq \max(|\sigma_1|, |\sigma_2|) - 1 - (\tau - 1) * q$.

- Example: $q = 2, \tau = 2$
  $\sigma_1 : theatre$; $\sigma_2 : theater$.
  $G_{\sigma_1}\{\#t, th, he, ea, at, tr, re, e\$\}$,
  $G_{\sigma_2}\{\#t, th, he, ea, at, te, er, r\$\}$.

Background: q-grams based edit distance pruning

- Lemma: For strings σ_1 and σ_2 of length $|\sigma_1|$ and $|\sigma_2|$, if $\varepsilon(\sigma_1, \sigma_2) = \tau$, then $|G_{\sigma_1} \cap G_{\sigma_2}| \geq \max(|\sigma_1|, |\sigma_2|) - 1 - (\tau - 1) * q$.

- Example: $q = 2, \tau = 2$
  $\sigma_1 : theatre$; $\sigma_2 : theater$.
  $G_{\sigma_1}\{\#t, th, he, ea, at, tr, re, e\$\}$,
  $G_{\sigma_2}\{\#t, th, he, ea, at, te, er, r\$\}$.

Background: estimating set resemblance

- Set resemblance: two sets A and B

$$\rho(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Background: estimating set resemblance

- Set resemblance: two sets A and B
$$\rho(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$
- The min-wise signature:
Any set X, any $x \in X$,$\pi \in F$(a family of min-wise independent permutations),
$$\Pr(\min\{\pi(X)\}) = \pi(x)) = \frac{1}{|X|}.$$

Background: estimating set resemblance

- Set resemblance: two sets A and B
$$\rho(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$
- The min-wise signature:
Any set X, any $x \in X$,$\pi \in F$(a family of min-wise independent permutations),
$$\Pr(\min\{\pi(X)\}) = \pi(x)) = \frac{1}{|X|}.$$
$s(A) = \{\min\{\pi_1(A)\}, \ldots, \min\{\pi_\ell(A)\}\}$,
$s(A \cup B) = \{\min\{\pi_1(A), \pi_1(B)\}, \ldots, \min\{\pi_\ell(A), \pi_\ell(B)\}\}$.

Background: estimating set resemblance

- Set resemblance: two sets A and B

$$\rho(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

- The min-wise signature:

  Any set X, any $x \in X$, $\pi \in F$(a family of min-wise independent permutations),

$$\Pr(\min\{\pi(X)\}) = \pi(x)) = \frac{1}{|X|}.$$

$$s(A) = \{\min\{\pi_1(A)\}, \ldots, \min\{\pi_\ell(A)\}\},$$
$$s(A \cup B) = \{\min\{\pi_1(A), \pi_1(B)\}, \ldots, \min\{\pi_\ell(A), \pi_\ell(B)\}\}.$$

- Unbiased estimator for $\rho(A,B)$:

$$\widehat{\rho}(A,B) = \Pr(\min\{\pi(A)\} = \min\{\pi(B)\}).$$

Background: estimating set resemblance

- Set resemblance: two sets A and B
$$\rho(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$
- The min-wise signature:
Any set X, any $x \in X$,$\pi \in F$(a family of min-wise independent permutations),
$$\Pr(\min\{\pi(X)\}) = \pi(x)) = \frac{1}{|X|}.$$
$s(A) = \{\min\{\pi_1(A)\}, \ldots, \min\{\pi_\ell(A)\}\}$,
$s(A \cup B) = \{\min\{\pi_1(A), \pi_1(B)\}, \ldots, \min\{\pi_\ell(A), \pi_\ell(B)\}\}$.
- Unbiased estimator for $\rho(A,B)$:
$$\widehat{\rho}(A,B) = \Pr(\min\{\pi(A)\} = \min\{\pi(B)\}).$$
- $$\widehat{\rho}(A,B) = \frac{|\{i | \min\{\pi_i(A)\} = \min\{\pi_i(B)\}\}|}{\ell}.$$

Background: estimating set resemblance

Set $A = \{1, 2, 4\}$

hashes	1	2	4
h_1	1	3	4
h_2	3	1	2
h_3	2	4	3
h_4	4	2	1

Background: estimating set resemblance

Set $A = \{1, 2, 4\}$

hashes	1	2	4
h_1	(1)	3	4
h_2	3	(1)	2
h_3	(2)	4	3
h_4	4	2	(1)

signature of A
1
1
2
1

Background: estimating set resemblance

Set $A = \{1, 2, 4\}$

hashes	1	2	4
h_1	1	3	4
h_2	3	1	2
h_3	2	4	3
h_4	4	2	1

signature of A
1
1
2
1

Set $B = \{2, 3\}$

hashes	2	3
h_1	3	2
h_2	1	4
h_3	4	1
h_4	2	3

signature of B
2
1
1
2

Background: estimating set resemblance

Set $A = \{1, 2, 4\}$

hashes	1	2	4
h_1	1	3	4
h_2	3	1	2
h_3	2	4	3
h_4	4	2	1

signature of A
1
1
2
1

Set $B = \{2, 3\}$

hashes	2	3
h_1	3	2
h_2	1	4
h_3	4	1
h_4	2	3

signature of B
2
1
1
2

Background: estimating set resemblance

Set $A = \{1, 2, 4\}$

hashes	1	2	4
h_1	(1)	3	4
h_2	3	(1)	2
h_3	(2)	4	3
h_4	4	2	(1)

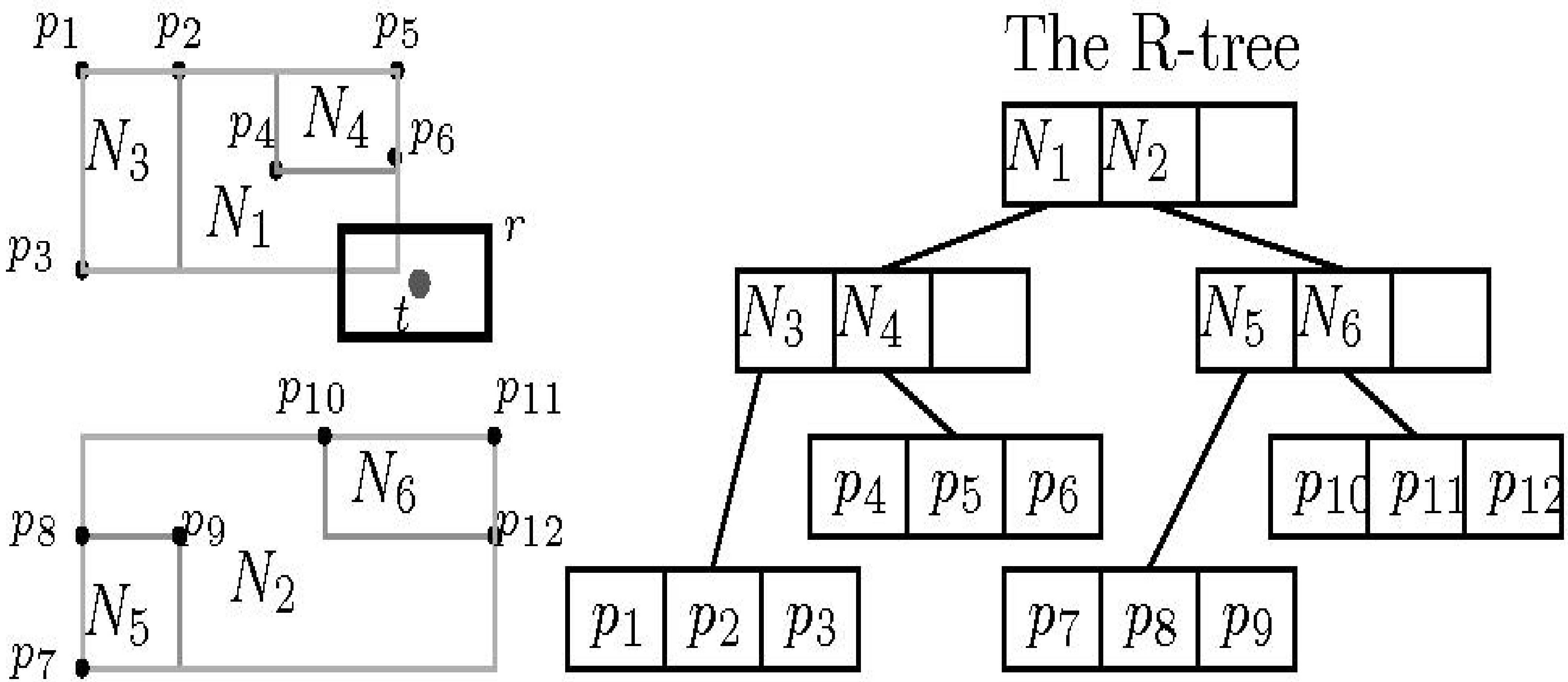
Set $B = \{2, 3\}$

hashes	2	3
h_1	3	(2)
h_2	(1)	4
h_3	4	(1)
h_4	(2)	3

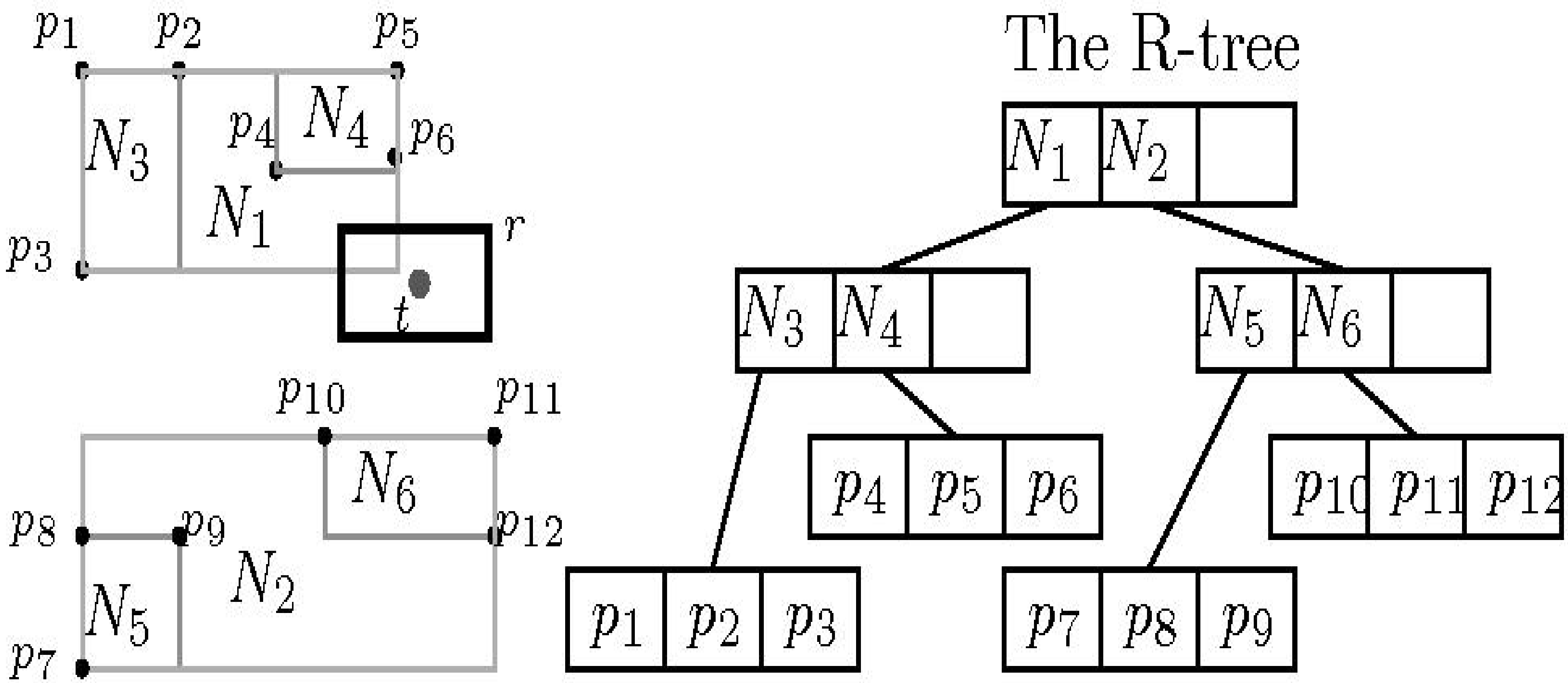
signature of A	signature of B
1	2
1 ⟷	1
2	1
1	2

$$\widehat{\rho}(A, B) = 1/4$$

The MHR-tree: basic idea

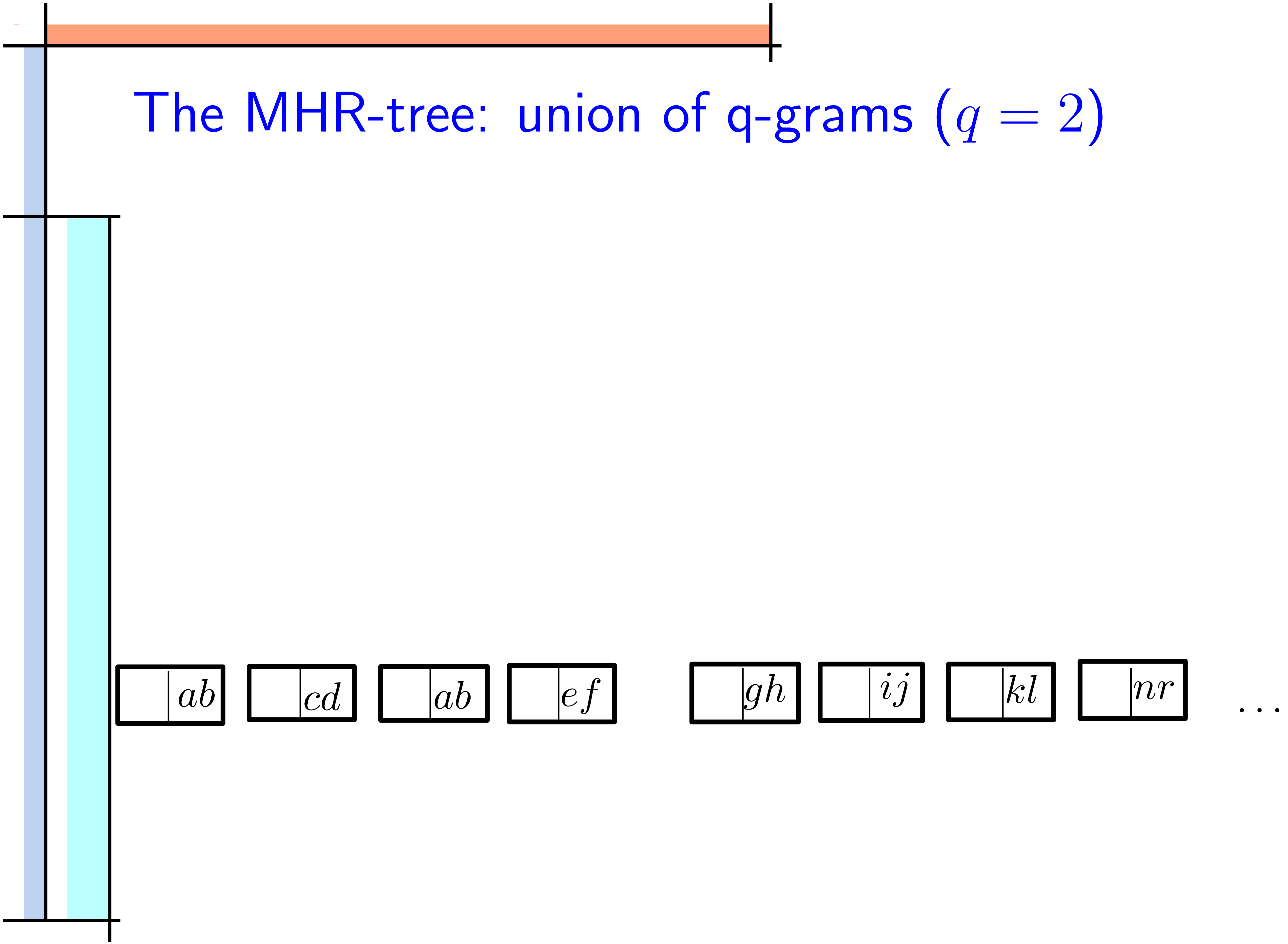


The MHR-tree: basic idea

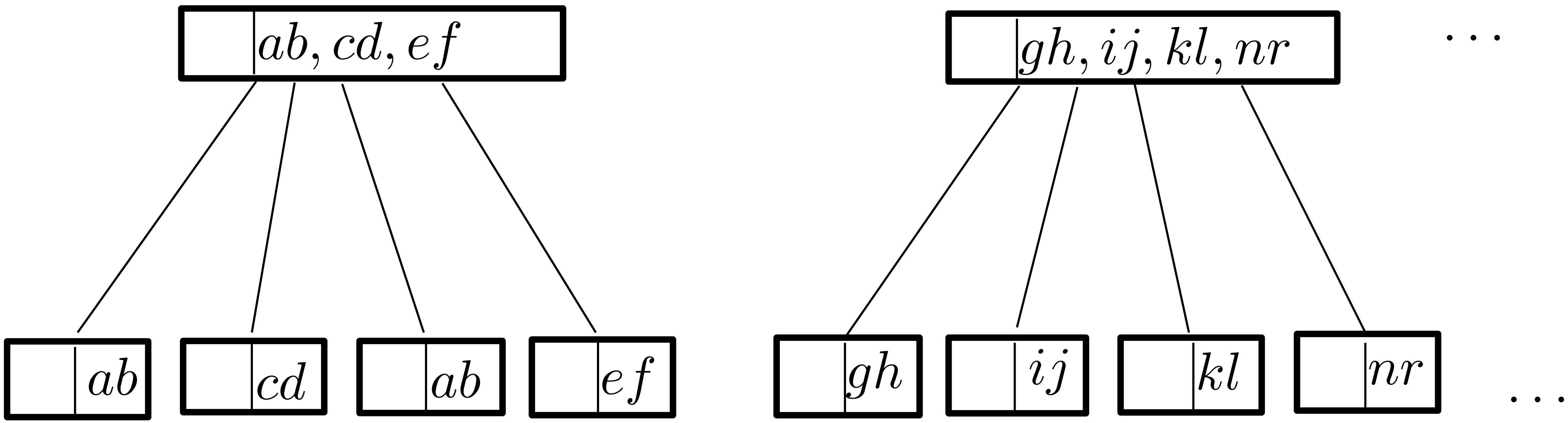


Lemma 2 Let G_u be the set for the union of q-grams of strings in the subtree of node u. For a SAS query (r, σ, τ), if $|G_u \cap G_\sigma| < |\sigma| - 1 - (\tau - 1) * q$, then the subtree of node u does not contain any element from A_s.

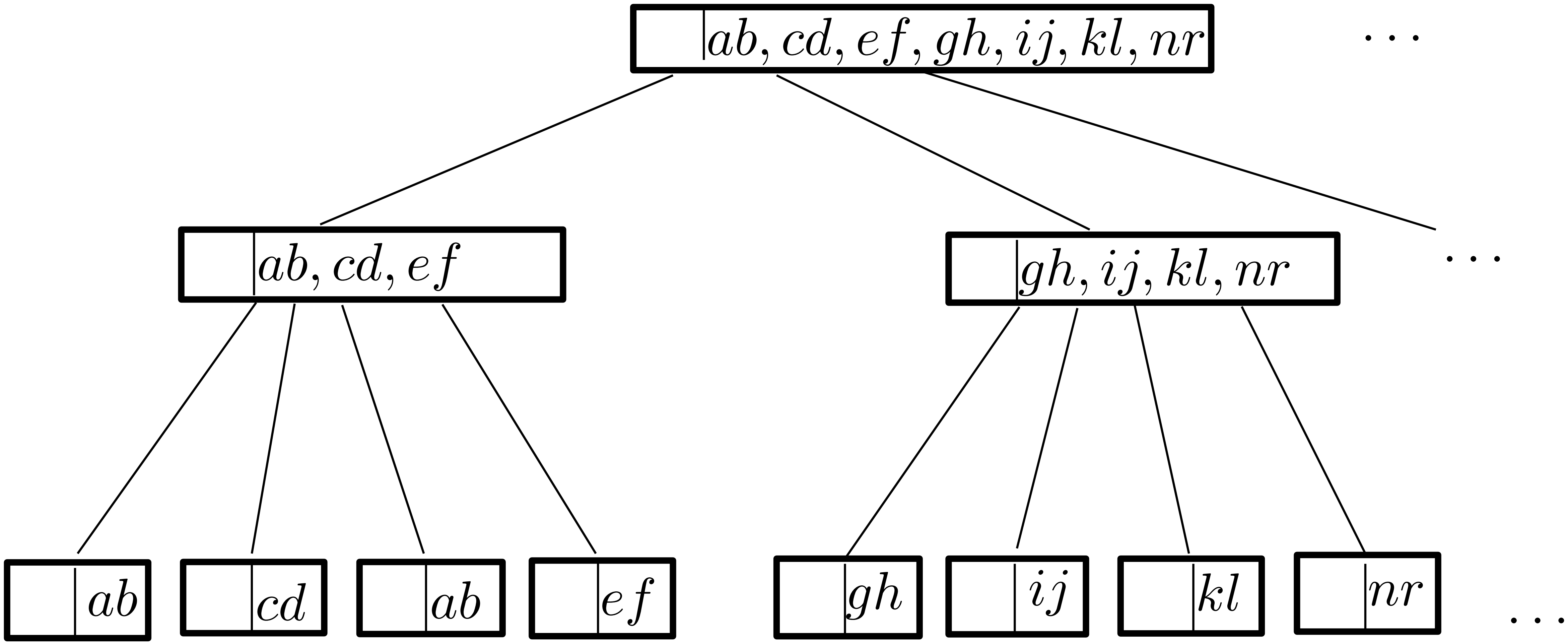
The MHR-tree: union of q-grams ($q = 2$)



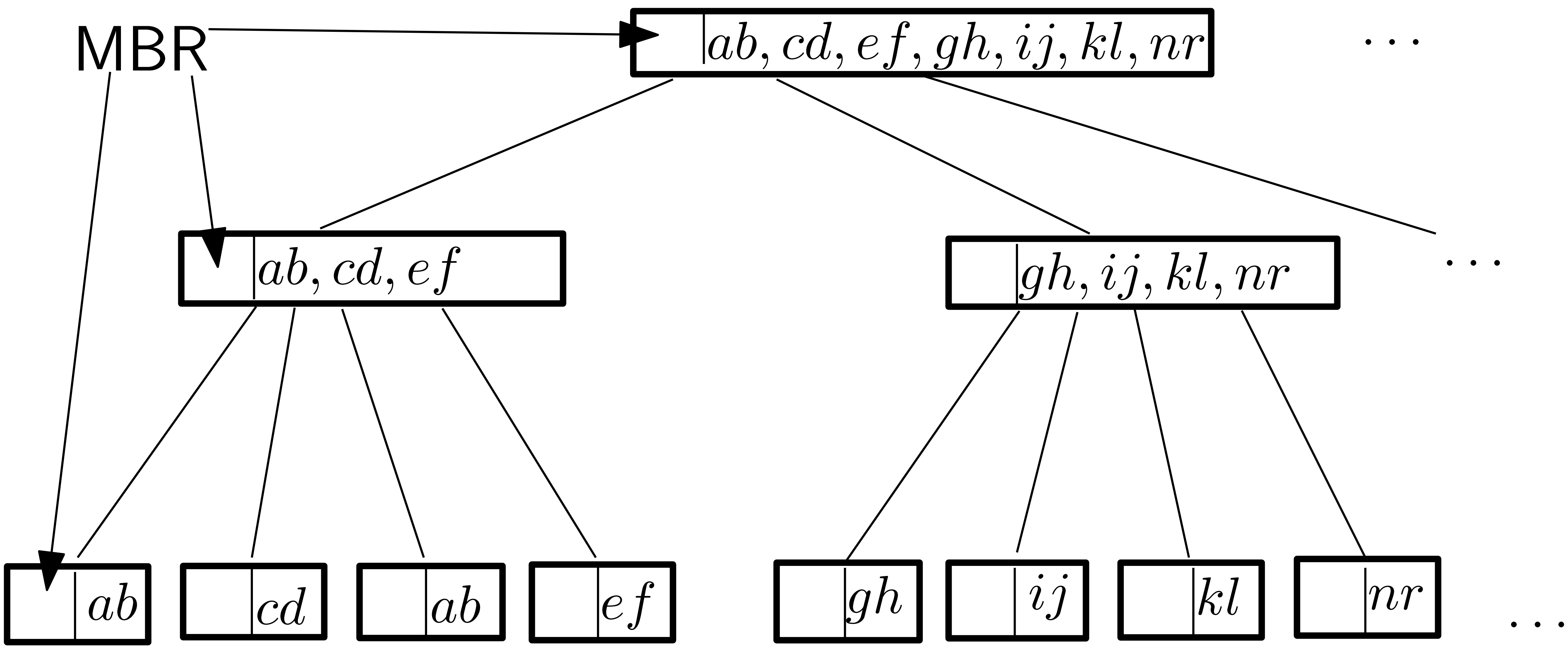
The MHR-tree: union of q-grams ($q = 2$)



The MHR-tree: union of q-grams ($q = 2$)



The MHR-tree: union of q-grams ($q = 2$)



The MHR-tree

Min-wise signature with linear hashing R-tree (MHR-tree)

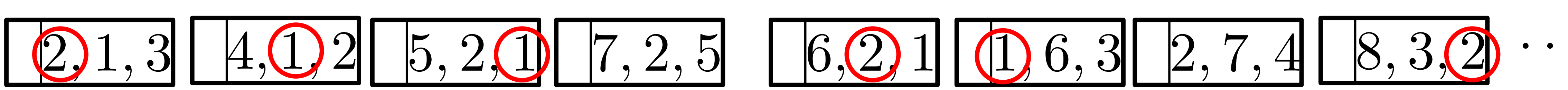
The MHR-tree

Min-wise signature with linear hashing R-tree (MHR-tree)

$2, 1, 3$	$4, 1, 2$	$5, 2, 1$	$7, 2, 5$	$6, 2, 1$	$1, 6, 3$	$2, 7, 4$	$8, 3, 2$	$\cdots$

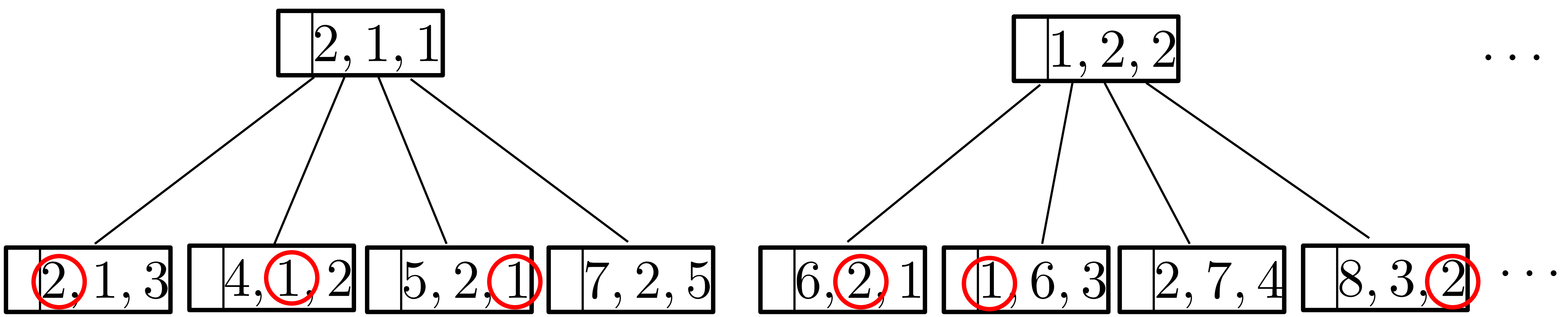
The MHR-tree

Min-wise signature with linear hashing R-tree (MHR-tree)



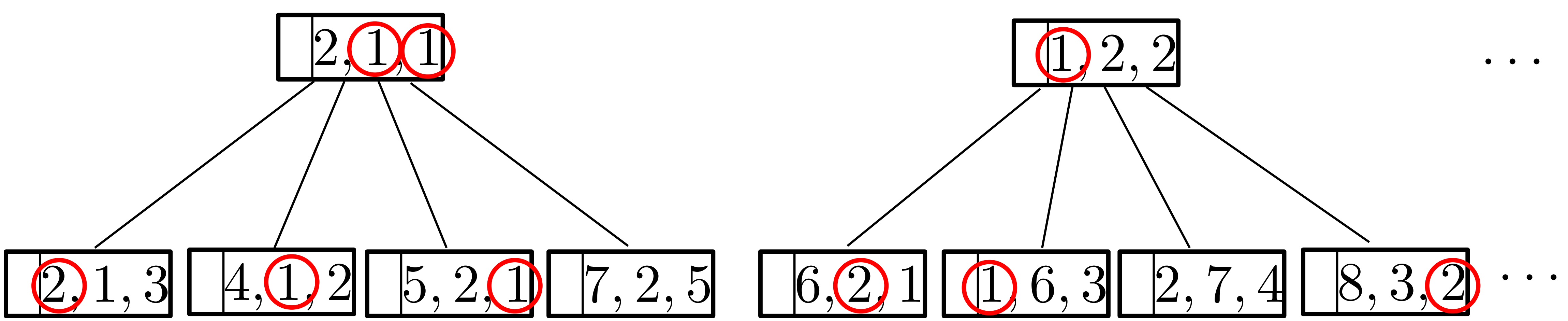
The MHR-tree

Min-wise signature with linear hashing R-tree (MHR-tree)



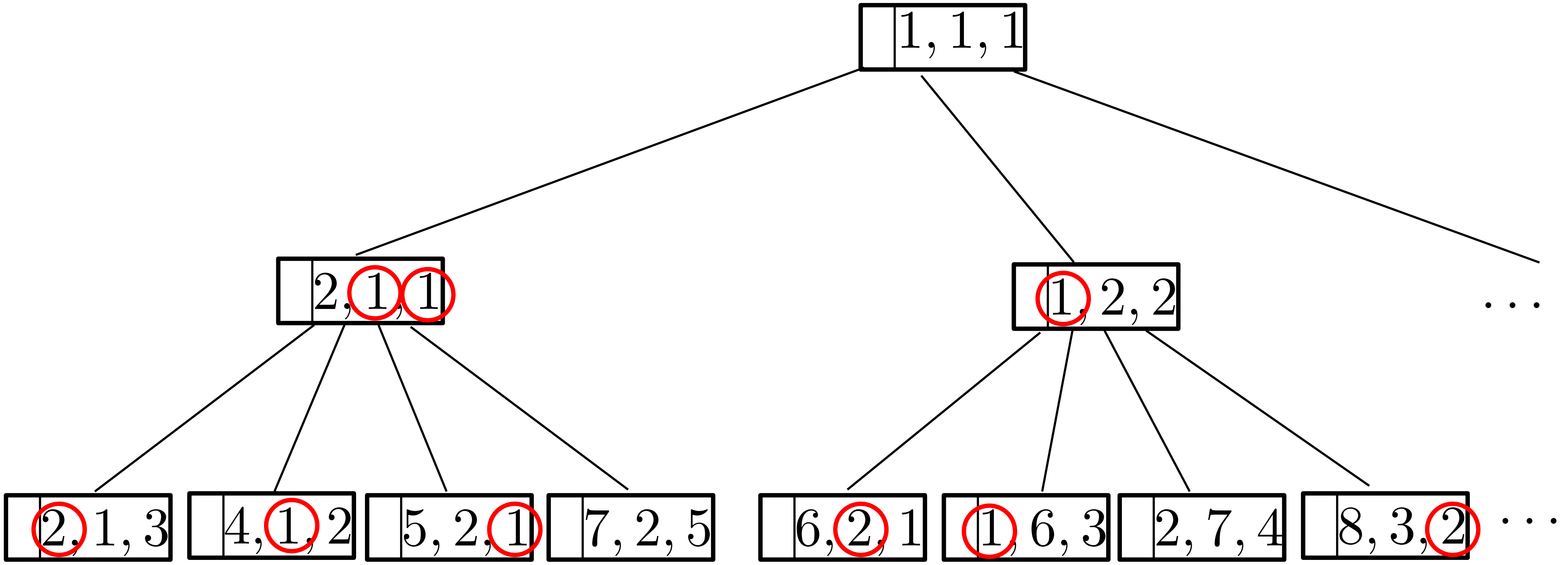
The MHR-tree

Min-wise signature with linear hashing R-tree (MHR-tree)



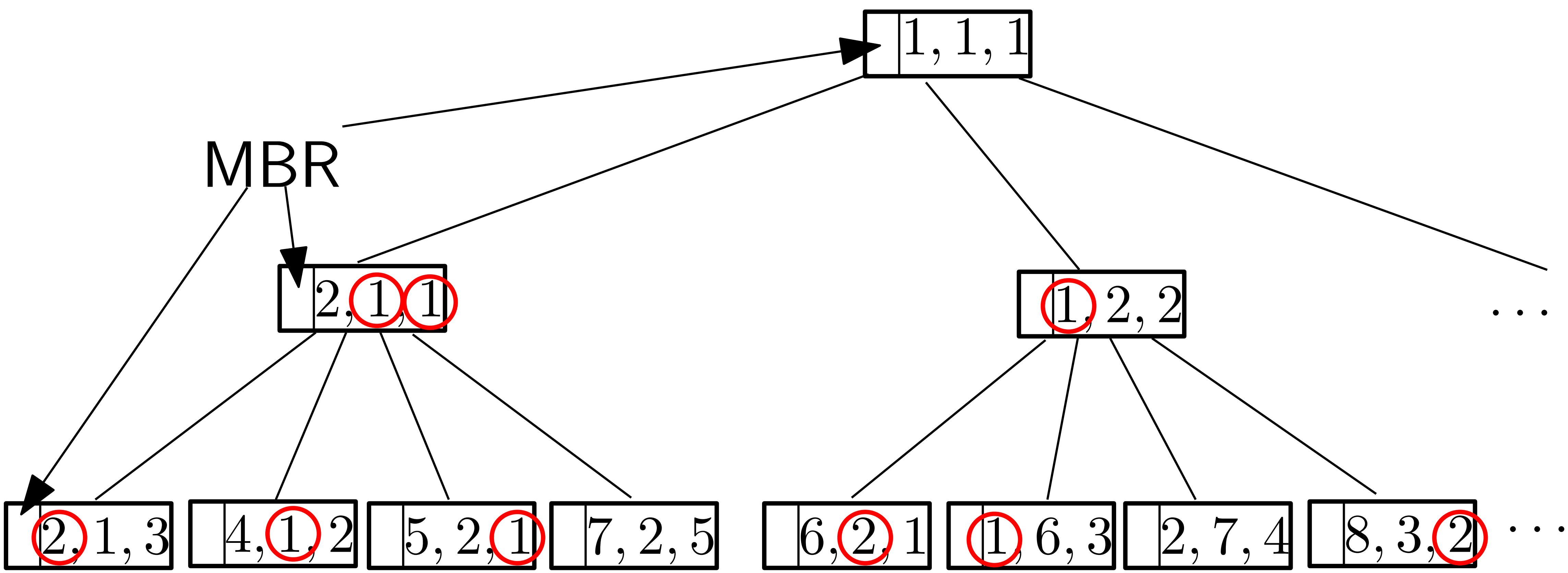
The MHR-tree

Min-wise signature with linear hashing R-tree (MHR-tree)



The MHR-tree

Min-wise signature with linear hashing R-tree (MHR-tree)



Query algorithms for the MHR-tree

RANGE-MHR(MHR-tree R, Range r, String σ, int τ)

Follow the range query algorithm on R-tree,

If u is a leaf node

For every point $p \in \mathbf{u}_p$

If p is contained in r and $|G_p \cap G_\sigma| \geq \max(|\sigma_p|, |\sigma|) - 1 - (\tau - 1) * q$ and $\varepsilon(\sigma_p, \sigma) < \tau$ then Insert p in A;

Else

For every child node w_i of u

If r and MBR(w_i) intersect, and $|\widehat{G_{w_i} \cap G_\sigma}| \geq |\sigma| - 1 - (\tau - 1) * q$

insert w_i into queue;

Return A

Query algorithms for the MHR-tree

RANGE-MHR(MHR-tree R, Range r, String σ, int τ)

Follow the range query algorithm on R-tree,

If u is a leaf node

For every point $p \in \mathbf{u}_p$

If p is contained in r and $|G_p \cap G_\sigma| \geq \max(|\sigma_p|, |\sigma|) - 1 - (\tau - 1) * q$ and $\varepsilon(\sigma_p, \sigma) < \tau$ then Insert p in A;

Else

For every child node w_i of u

If r and MBR(w_i) intersect, and $|\widehat{G_{w_i} \cap G_\sigma}| \geq |\sigma| - 1 - (\tau - 1) * q$

insert w_i into queue;

Return A

Query algorithms for the MHR-tree

RANGE-MHR(MHR-tree R, Range r, String σ, int τ)

Follow the range query algorithm on R-tree,

If u is a leaf node

For every point $p \in \mathbf{u}_p$

If p is contained in r and $|G_p \cap G_\sigma| \geq \max(|\sigma_p|, |\sigma|) - 1 - (\tau - 1) * q$ and $\varepsilon(\sigma_p, \sigma) < \tau$ then Insert p in A;

Else

For every child node w_i of u

If r and MBR(w_i) intersect, and $|\widehat{G_{w_i} \cap G_\sigma}| \geq |\sigma| - 1 - (\tau - 1) * q$

insert w_i into queue;

Return A

Query algorithms for the MHR-tree

$\textsc{Range-MHR}$(MHR-tree R, Range r, String σ, int τ)

Follow the range query algorithm on R-tree,

If u is a leaf node

For every point $p \in \mathbf{u}_p$

If p is contained in r and $|G_p \cap G_\sigma| \geq \max(|\sigma_p|, |\sigma|) - 1 - (\tau - 1) * q$ and $\varepsilon(\sigma_p, \sigma) < \tau$ then Insert p in A;

Else

For every child node w_i of u

If r and MBR(w_i) intersect, and $|\widehat{G_{w_i} \cap G_\sigma}| \geq |\sigma| - 1 - (\tau - 1) * q$

insert w_i into queue;

Return A

Query algorithms for the MHR-tree

$\textsc{Range-MHR}$(MHR-tree R, Range r, String σ, int τ)

Follow the range query algorithm on R-tree,

If u is a leaf node

For every point $p \in \mathbf{u}_p$

If p is contained in r and $|G_p \cap G_\sigma| \geq \max(|\sigma_p|, |\sigma|) - 1 - (\tau - 1) * q$ and $\varepsilon(\sigma_p, \sigma) < \tau$ then Insert p in A;

Else

For every child node w_i of u

If r and MBR(w_i) intersect, and $|\widehat{G_{w_i} \cap G_\sigma}| \geq |\sigma| - 1 - (\tau - 1) * q$

insert w_i into queue;

Return A

Query algorithms for the MHR-tree

- Key issue: estimating $|G_u \cap G_\sigma|$ using $s(G_u)$ and σ.

Query algorithms for the MHR-tree

- Key issue: estimating $|G_u \cap G_\sigma|$ using $s(G_u)$ and σ.

σ: crab G_σ: $\#$c,cr,ra,ab,b$

$s(G_\sigma)$: $3, 1, 4, 5$ $s(G_u)$: $3, 1, 3, 5$

Query algorithms for the MHR-tree

- Key issue: estimating $|G_u \cap G_\sigma|$ using $s(G_u)$ and σ.

σ: crab G_σ: #c,cr,ra,ab,b$

$s(G_\sigma)$: $3, 1, 4, 5$ $s(G_u)$: $3, 1, 3, 5$

Let $G = G_u \cup G_\sigma$, $s(G) = s(G_u \cup G_\sigma) = 3, 1, 3, 5$

Query algorithms for the MHR-tree

- Key issue: estimating $|G_u \cap G_\sigma|$ using $s(G_u)$ and σ.

σ: crab G_σ: #c,cr,ra,ab,b$

$s(G_\sigma)$: $3, 1, 4, 5$ $s(G_u)$: $3, 1, 3, 5$

Let $G = G_u \cup G_\sigma$, $s(G) = s(G_u \cup G_\sigma) = 3, 1, 3, 5$

$\widehat{\rho}(G, G_\sigma) = \frac{|\{i|\min\{h_i(G)\}=\min\{h_i(G_\sigma)\}\}|}{\ell} = 3/4.$

Query algorithms for the MHR-tree

- Key issue: estimating $|G_u \cap G_\sigma|$ using $s(G_u)$ and σ.

σ: crab G_σ: #c,cr,ra,ab,b$

$s(G_\sigma)$: $3, 1, 4, 5$ $s(G_u)$: $3, 1, 3, 5$

Let $G = G_u \cup G_\sigma$, $s(G) = s(G_u \cup G_\sigma) = 3, 1, 3, 5$

$\widehat{\rho}(G, G_\sigma) = \frac{|\{i| \min\{h_i(G)\}=\min\{h_i(G_\sigma)\}\}|}{\ell} = 3/4.$

$\rho(G, G_\sigma) = \frac{|G \cap G_\sigma|}{|G \cup G_\sigma|} = \frac{|(G_u \cup G_\sigma) \cap G_\sigma|}{|(G_u \cup G_\sigma) \cup G_\sigma|} = \frac{|G_\sigma|}{|G_u \cup G_\sigma|}.$

Query algorithms for the MHR-tree

- Key issue: estimating $|G_u \cap G_\sigma|$ using $s(G_u)$ and σ.

σ: crab G_σ: #c,cr,ra,ab,b$

$s(G_\sigma)$: $3, 1, 4, 5$ $s(G_u)$: $3, 1, 3, 5$

Let $G = G_u \cup G_\sigma$, $s(G) = s(G_u \cup G_\sigma) = 3, 1, 3, 5$

$$\widehat{\rho}(G, G_\sigma) = \frac{|\{i|\min\{h_i(G)\} = \min\{h_i(G_\sigma)\}\}|}{\ell} = 3/4.$$

$$\rho(G, G_\sigma) = \frac{|G \cap G_\sigma|}{|G \cup G_\sigma|} = \boxed{\frac{|(G_u \cup G_\sigma) \cap G_\sigma|}{|(G_u \cup G_\sigma) \cup G_\sigma|}} = \frac{|G_\sigma|}{|G_u \cup G_\sigma|}.$$

Query algorithms for the MHR-tree

- Key issue: estimating $|G_u \cap G_\sigma|$ using $s(G_u)$ and σ.

σ: crab G_σ: #c,cr,ra,ab,b$

$s(G_\sigma)$: $3, 1, 4, 5$ $s(G_u)$: $3, 1, 3, 5$

Let $G = G_u \cup G_\sigma$, $s(G) = s(G_u \cup G_\sigma) = 3, 1, 3, 5$

$$\widehat{\rho}(G, G_\sigma) = \frac{|\{i| \min\{h_i(G)\} = \min\{h_i(G_\sigma)\}\}|}{\ell} = 3/4.$$

$$\rho(G, G_\sigma) = \frac{|G \cap G_\sigma|}{|G \cup G_\sigma|} = \frac{|(G_u \cup G_\sigma) \cap G_\sigma|}{|(G_u \cup G_\sigma) \cup G_\sigma|} = \frac{|G_\sigma|}{|G_u \cup G_\sigma|}.$$

$$|\widehat{G_u \cup G_\sigma}| = \frac{|G_\sigma|}{\widehat{\rho}(G, G_\sigma)} = 5/(3/4) = 20/3.$$

Query algorithms for the MHR-tree

- Key issue: estimating $|G_u \cap G_\sigma|$ using $s(G_u)$ and σ.

σ: crab G_σ: #c,cr,ra,ab,b$

$s(G_\sigma)$: $3, 1, 4, 5$ $s(G_u)$: $3, 1, 3, 5$

Let $G = G_u \cup G_\sigma$, $s(G) = s(G_u \cup G_\sigma) = 3, 1, 3, 5$

$$\widehat{\rho}(G, G_\sigma) = \frac{|\{i| \min\{h_i(G)\} = \min\{h_i(G_\sigma)\}\}|}{\ell} = 3/4.$$

$$\rho(G, G_\sigma) = \frac{|G \cap G_\sigma|}{|G \cup G_\sigma|} = \frac{|(G_u \cup G_\sigma) \cap G_\sigma|}{|(G_u \cup G_\sigma) \cup G_\sigma|} = \frac{|G_\sigma|}{|G_u \cup G_\sigma|}.$$

$$|\widehat{G_u \cup G_\sigma}| = \frac{|G_\sigma|}{\widehat{\rho}(G, G_\sigma)} = 5/(3/4) = 20/3.$$

$$\widehat{\rho}(G_u, G_\sigma) = \frac{|\{i| \min\{h_i(G_u)\} = \min\{h_i(G_\sigma)\}\}|}{\ell} = 3/4.$$

Query algorithms for the MHR-tree

- Key issue: estimating $|G_u \cap G_\sigma|$ using $s(G_u)$ and σ.

σ: crab G_σ: #c,cr,ra,ab,b$

$s(G_\sigma)$: $3, 1, 4, 5$ $s(G_u)$: $3, 1, 3, 5$

Let $G = G_u \cup G_\sigma$, $s(G) = s(G_u \cup G_\sigma) = 3, 1, 3, 5$

$$\widehat{\rho}(G, G_\sigma) = \frac{|\{i|\min\{h_i(G)\}=\min\{h_i(G_\sigma)\}\}|}{\ell} = 3/4.$$

$$\rho(G, G_\sigma) = \frac{|G \cap G_\sigma|}{|G \cup G_\sigma|} = \frac{|(G_u \cup G_\sigma) \cap G_\sigma|}{|(G_u \cup G_\sigma) \cup G_\sigma|} = \frac{|G_\sigma|}{|G_u \cup G_\sigma|}.$$

$$|\widehat{G_u \cup G_\sigma}| = \frac{|G_\sigma|}{\widehat{\rho}(G, G_\sigma)} = 5/(3/4) = 20/3.$$

$$\widehat{\rho}(G_u, G_\sigma) = \frac{|\{i|\min\{h_i(G_u)\}=\min\{h_i(G_\sigma)\}\}|}{\ell} = 3/4.$$

$$|\widehat{G_u \cap G_\sigma}| = \widehat{\rho}(G_u, G_\sigma) * |\widehat{G_u \cup G_\sigma}| = 3/4 * 20/3 = 5.$$

Duplicate q-grams in strings

- Issue 1: duplicate q-grams in one string (for both query and data).

Duplicate q-grams in strings

- Issue 1: duplicate q-grams in one string (for both query and data).

  example:

  s: aabbaabb, {1#a,1aa, 1ab, 1bb, 1ba, 2aa, 2ab, 2bb, 1b$}

  q: aabcaa, {1#a, 1aa, 1ab, 1bc, 1ca, 2aa, 1a$}

Duplicate q-grams in strings

- Issue 1: duplicate q-grams in one string (for both query and data).

  example:

  s: aabbaabb, {1#a,1aa, 1ab, 1bb, 1ba, 2aa, 2ab, 2bb, 1b$}

  q: aabcaa, {1#a, 1aa, 1ab, 1bc, 1ca, 2aa, 1a$}

Duplicate q-grams in strings

- Issue 1: duplicate q-grams in one string (for both query and data).

  example:
  s: aabbaabb, {1#a, 1aa, 1ab, 1bb, 1ba, 2aa, 2ab, 2bb, 1b$}
  q: aabcaa, {1#a, 1aa, 1ab, 1bc, 1ca, 2aa, 1a$}

Duplicate q-grams in strings

- Issue 1: duplicate q-grams in one string (for both query and data).

  example:
  s: aabbaabb, {1#a, 1aa, 1ab, 1bb, 1ba, 2aa, 2ab, 2bb, 1b$}
  q: aabcaa, {1#a, 1aa, 1ab, 1bc, 1ca, 2aa, 1a$}

- Issue 2: duplicate q-grams between strings.

Duplicate q-grams in strings

- Issue 1: duplicate q-grams in one string (for both query and data).

  example:
  s: aabbaabb, {1#a, 1aa, 1ab, 1bb, 1ba, 2aa, 2ab, 2bb, 1b$}
  q: aabcaa, {1#a, 1aa, 1ab, 1bc, 1ca, 2aa, 1a$}

- Issue 2: duplicate q-grams between strings.

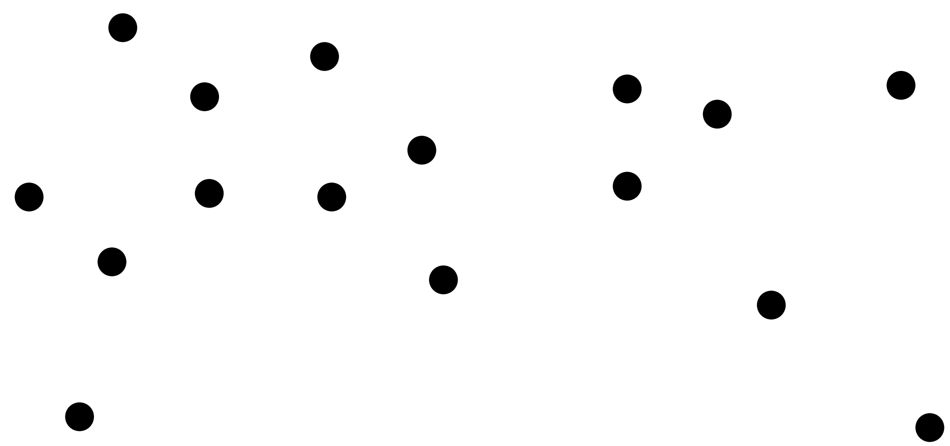
  Do not distinguish q-grams from different nodes.
  node 1: pizz (#p, pi, iz, zz, z$);
  node 2: zza (#z, zz, za, a$)
  parent node 3:
  union of signatures corresponding to (#p, pi, iz, zz, z$, #z, za, a$)

Selectivity estimation for SAS range queries

- Combine the range query selectivity estimator with the string selectivity estimator ($VSol$ [Mazeika et al.2007] based on min-wise signatures of inverted lists of q-grams).

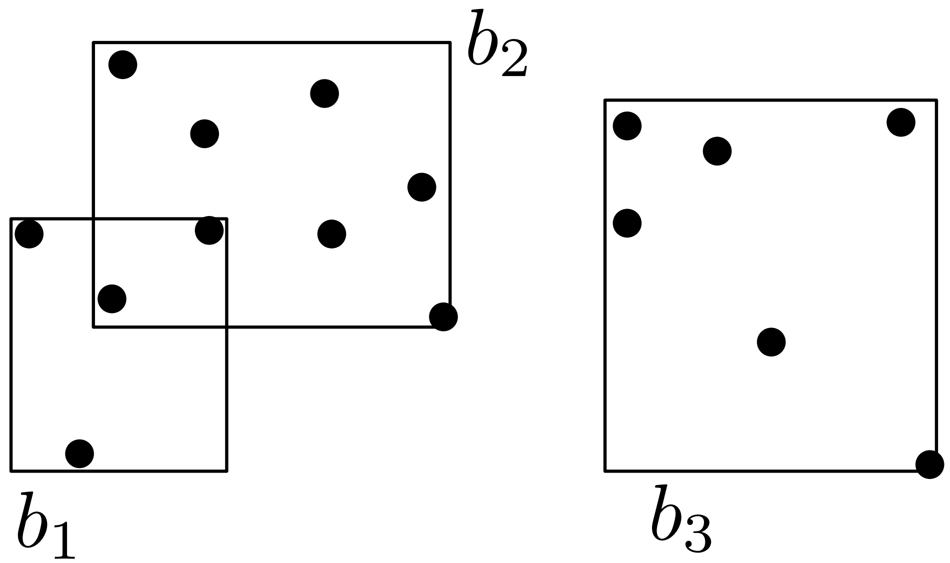
Selectivity estimation for SAS range queries

- Combine the range query selectivity estimator with the string selectivity estimator ($VSol$ [Mazeika et al.2007] based on min-wise signatures of inverted lists of q-grams).



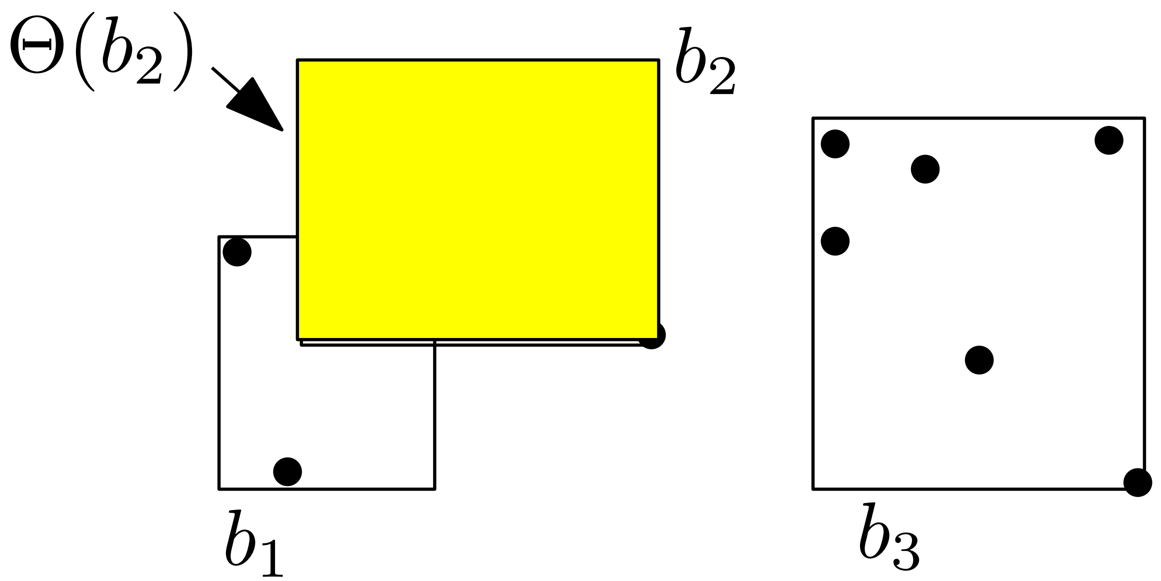
Selectivity estimation for SAS range queries

- Combine the range query selectivity estimator with the string selectivity estimator ($VSol$ [Mazeika et al.2007] based on min-wise signatures of inverted lists of q-grams).



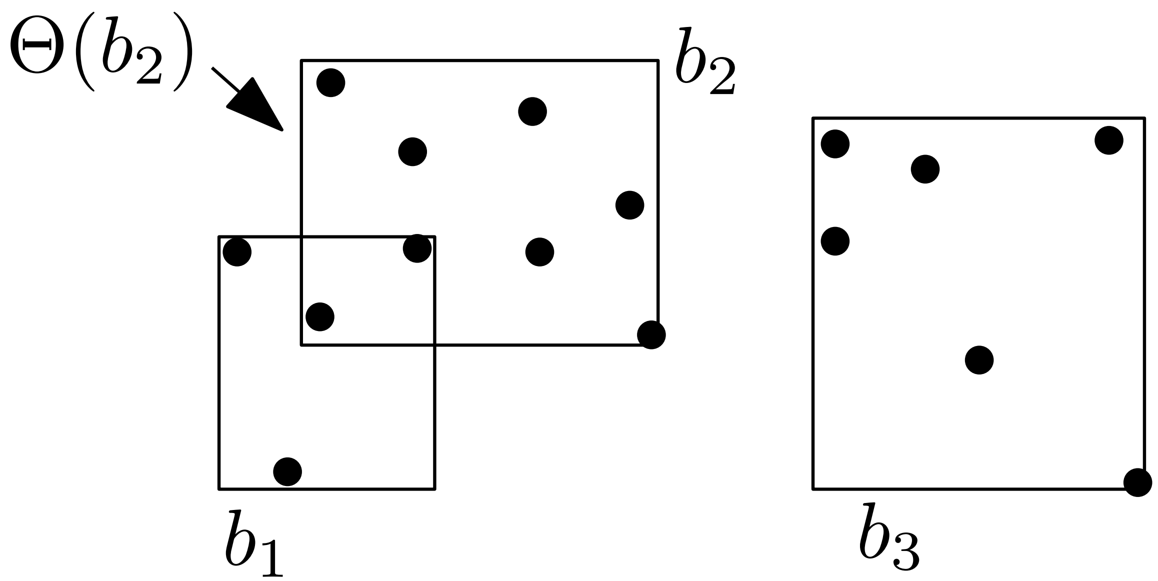
Selectivity estimation for SAS range queries

- Combine the range query selectivity estimator with the string selectivity estimator ($VSol$ [Mazeika et al.2007] based on min-wise signatures of inverted lists of q-grams).



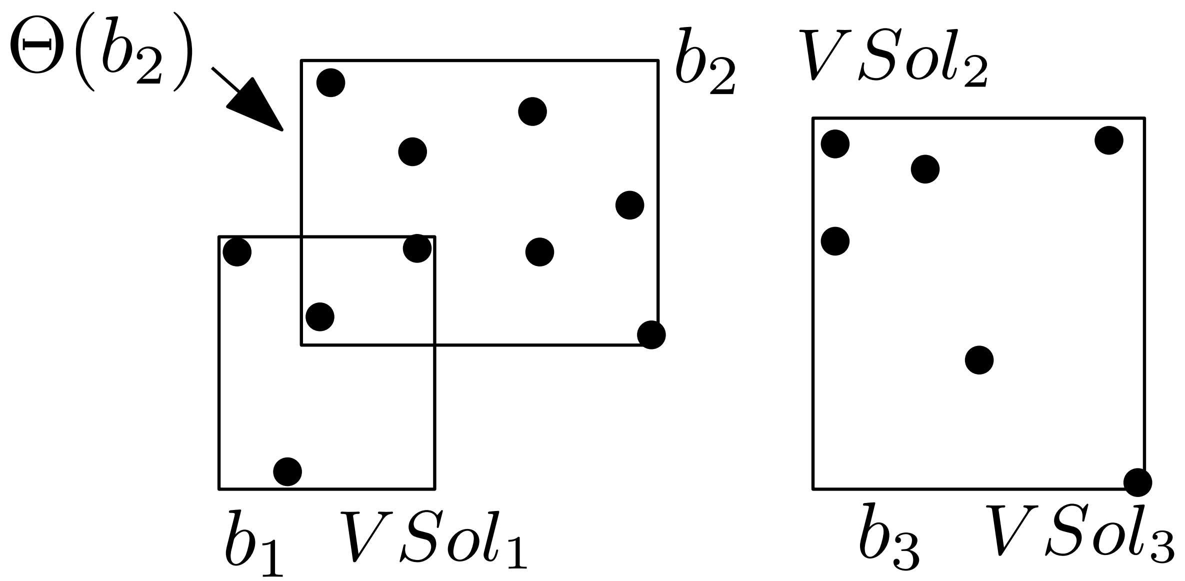
Selectivity estimation for SAS range queries

- Combine the range query selectivity estimator with the string selectivity estimator ($VSol$ [Mazeika et al.2007] based on min-wise signatures of inverted lists of q-grams).



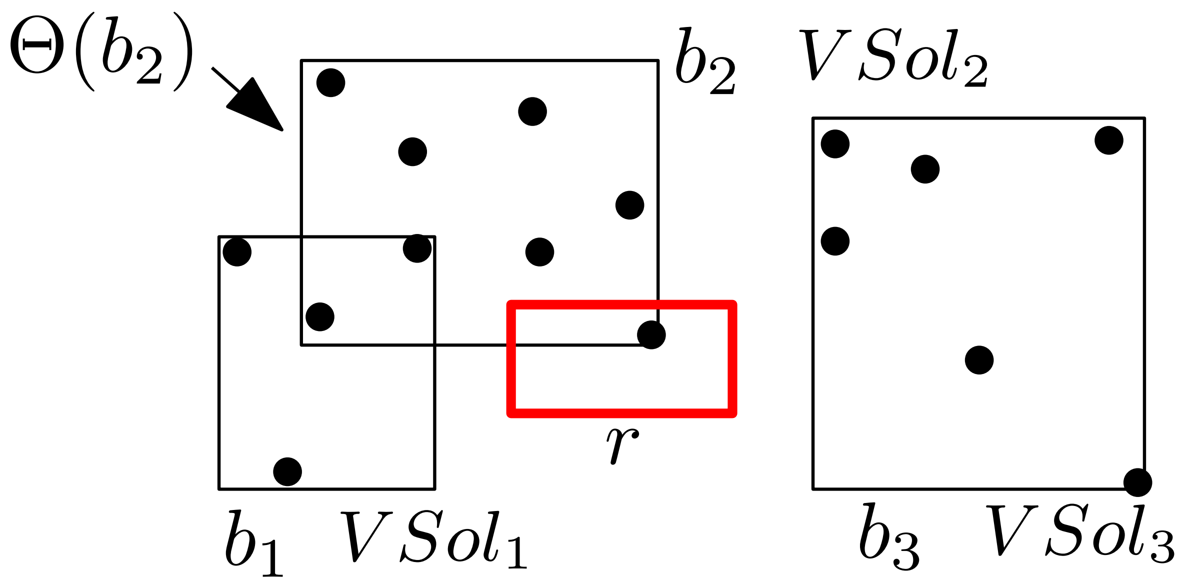
Selectivity estimation for SAS range queries

- Combine the range query selectivity estimator with the string selectivity estimator ($VSol$ [Mazeika et al.2007] based on min-wise signatures of inverted lists of q-grams).



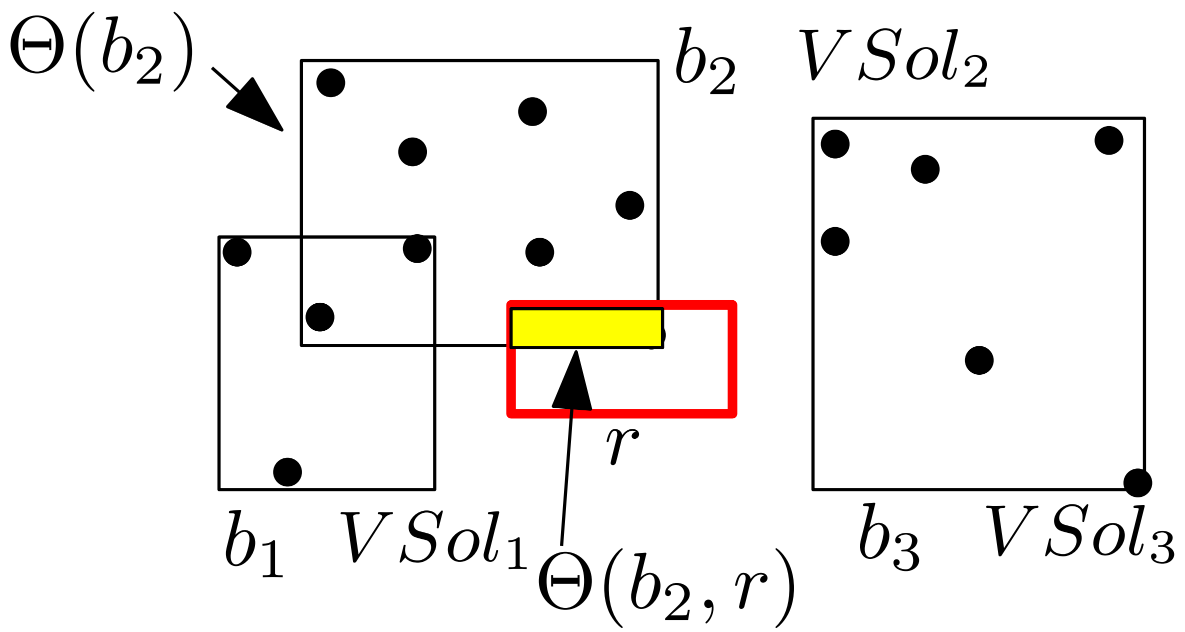
Selectivity estimation for SAS range queries

- Combine the range query selectivity estimator with the string selectivity estimator ($VSol$ [Mazeika et al.2007] based on min-wise signatures of inverted lists of q-grams).



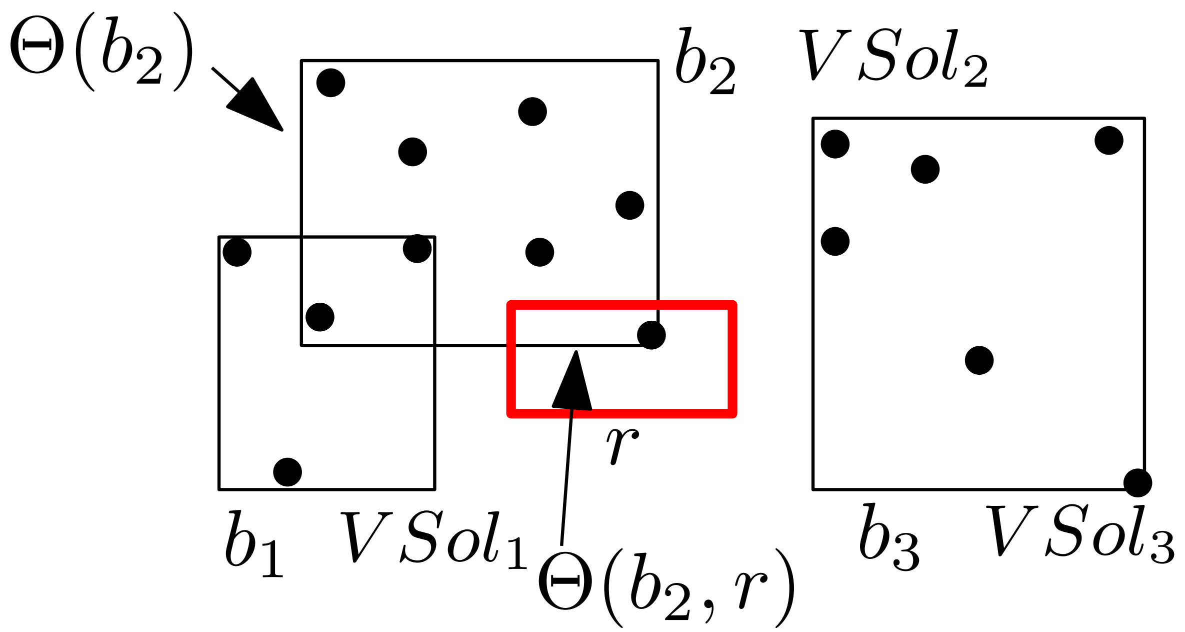
Selectivity estimation for SAS range queries

- Combine the range query selectivity estimator with the string selectivity estimator ($VSol$ [Mazeika et al.2007] based on min-wise signatures of inverted lists of q-grams).



Selectivity estimation for SAS range queries

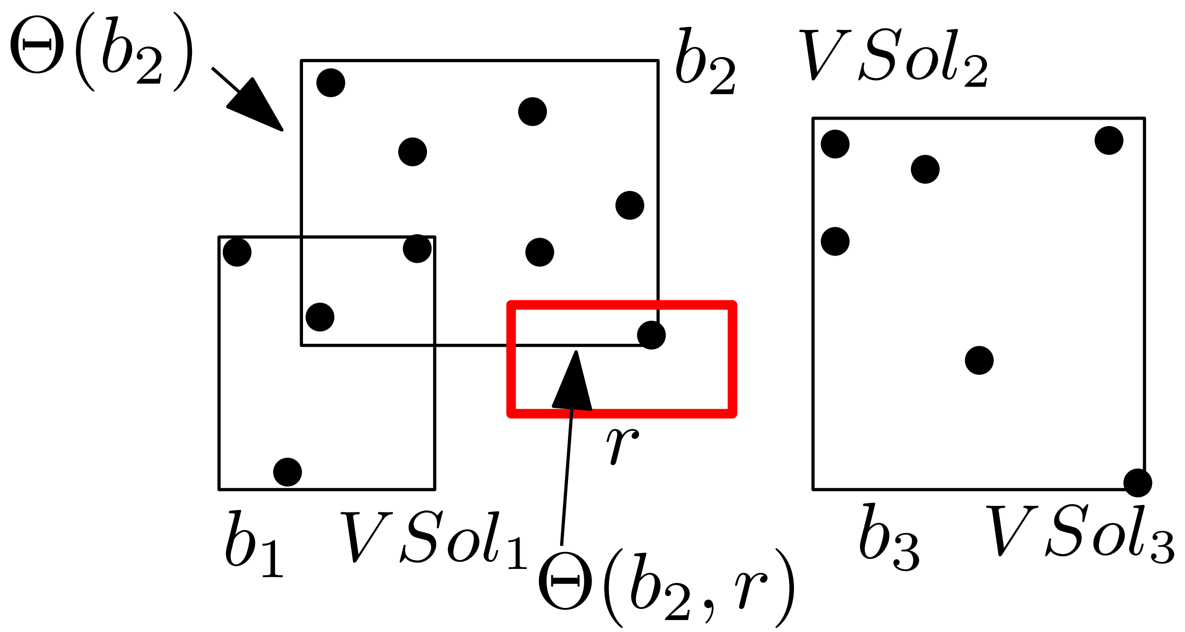
- Combine the range query selectivity estimator with the string selectivity estimator ($VSol$ [Mazeika et al.2007] based on min-wise signatures of inverted lists of q-grams).



$\widehat{|A_{b_i}|} = n_i \frac{\Theta(b_i,r)}{\Theta(b_i)} \frac{\rho^i_{LM}}{n_i} = \frac{\Theta(b_i,r)}{\Theta(b_i)} \rho^i_{LM}$. ($\rho^i_{LM}$ estimates the number of similar strings with query string in b_i.)

Selectivity estimation for SAS range queries

- Combine the range query selectivity estimator with the string selectivity estimator ($VSol$ [Mazeika et al.2007] based on min-wise signatures of inverted lists of q-grams).



$\widehat{|A_{b_i}|} = n_i \frac{\Theta(b_i,r)}{\Theta(b_i)} \frac{\rho^i_{LM}}{n_i} = \frac{\Theta(b_i,r)}{\Theta(b_i)} \rho^i_{LM}$. ($\rho^i_{LM}$ estimates the number of similar strings with query string in b_i.)

- Improvements:
  Minimum number of neighborhoods principle,
  Spatial uniformity principle.

Two improvements for selectivity estimation

- Minimum number of neighborhoods principle:

Two improvements for selectivity estimation

- Minimum number of neighborhoods principle:

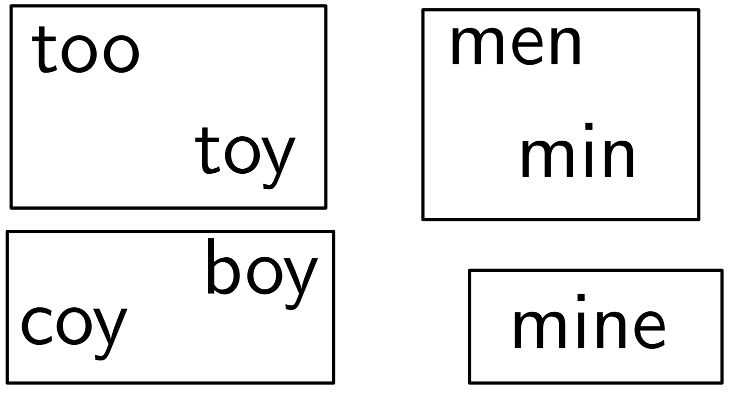
$\tau = 1$

too
men
toy
min
boy
coy
mine

Two improvements for selectivity estimation

- Minimum number of neighborhoods principle:

$\tau = 1$

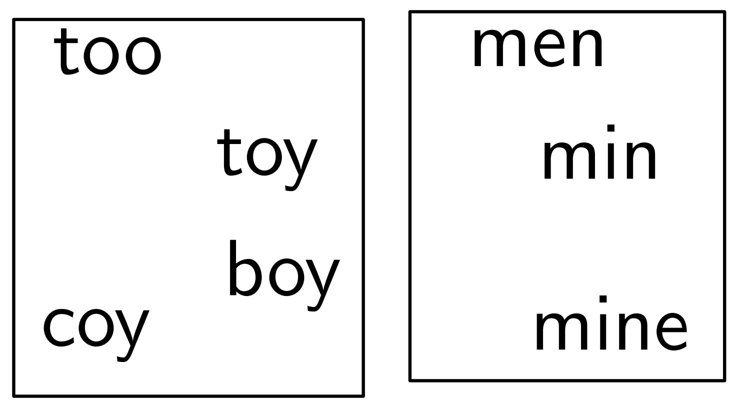


$\eta = 4$

Two improvements for selectivity estimation

- Minimum number of neighborhoods principle:

$\tau = 1$

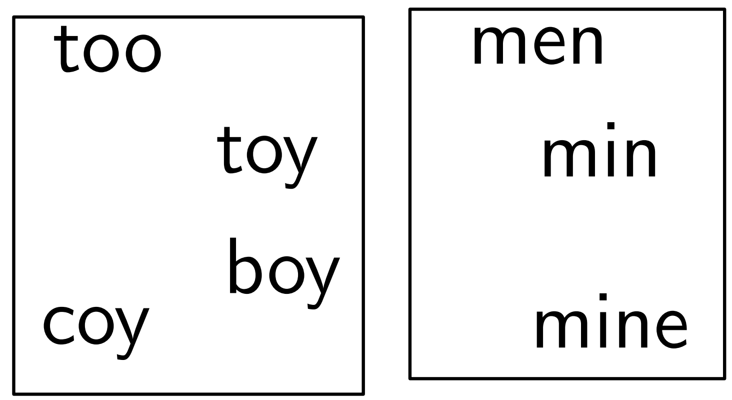


$\eta = 2$

Two improvements for selectivity estimation

- Minimum number of neighborhoods principle:

$\tau = 1$



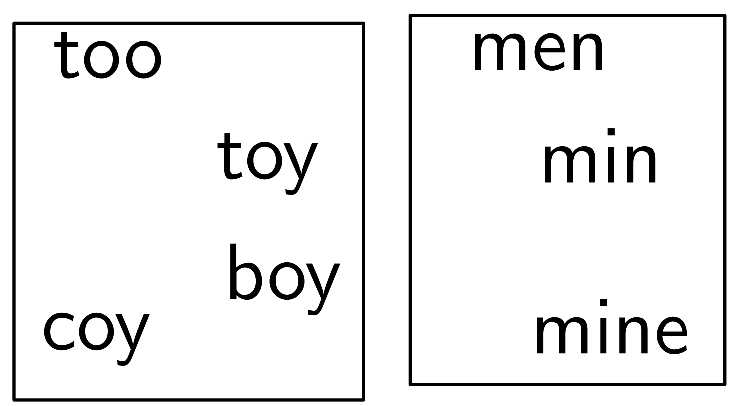
$\eta = 2$

A smaller η give a more accurate estimator.

Two improvements for selectivity estimation

- Minimum number of neighborhoods principle:

$\tau = 1$



$\eta = 2$

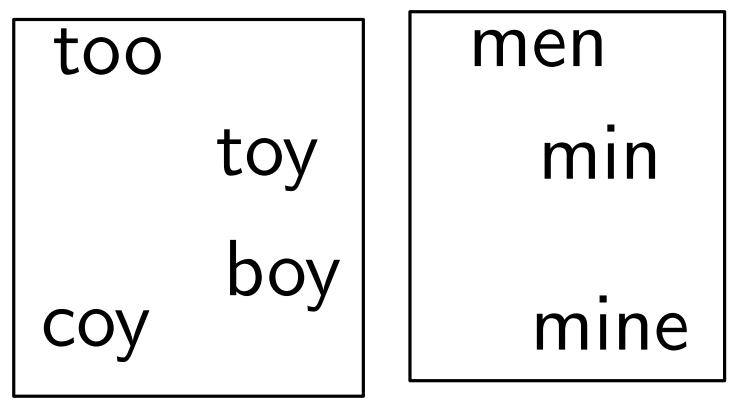
A smaller η give a more accurate estimator.

- Spatial uniformity principle:

Two improvements for selectivity estimation

- Minimum number of neighborhoods principle:

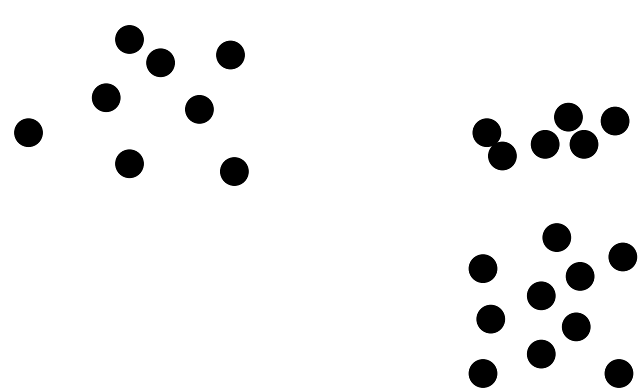
$\tau = 1$



$\eta = 2$

A smaller η give a more accurate estimator.

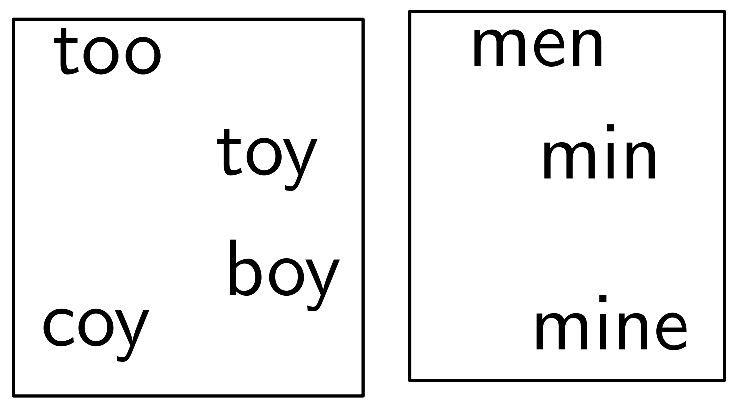
- Spatial uniformity principle:



Two improvements for selectivity estimation

- Minimum number of neighborhoods principle:

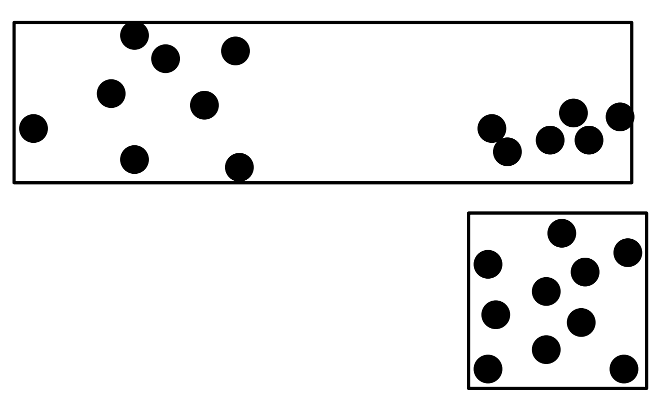
$\tau = 1$



$\eta = 2$

A smaller η give a more accurate estimator.

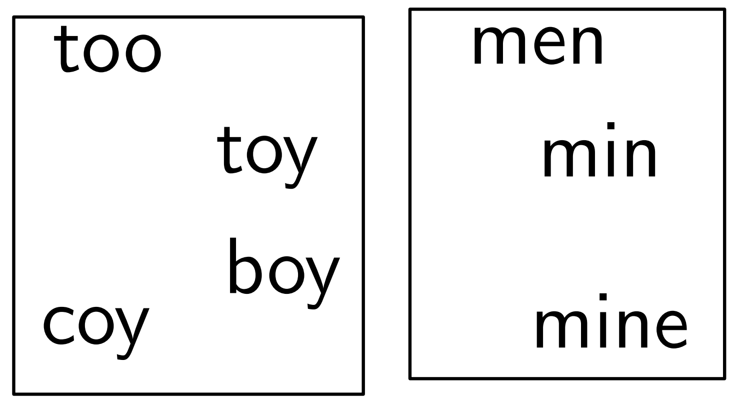
- Spatial uniformity principle:



Two improvements for selectivity estimation

- Minimum number of neighborhoods principle:

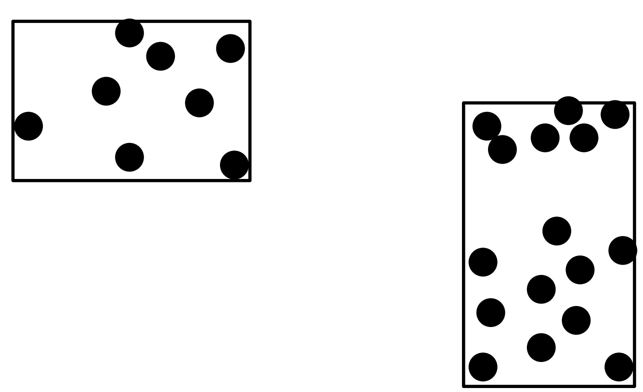
$\tau = 1$



$\eta = 2$

A smaller η give a more accurate estimator.

- Spatial uniformity principle:



The partitioning metric

- Neighborhood and uniformity quality of b:
  $\Delta(b) = \eta_b n_b \sum_{1,\ldots,d} X_i$,
  $\{X_1, \ldots, X_d\}$: the side lengths of b in each dimension.

The partitioning metric

- Neighborhood and uniformity quality of b:
  $\Delta(b) = \eta_b n_b \sum_{1,\ldots,d} X_i$,
  $\{X_1, \ldots, X_d\}$: the side lengths of b in each dimension.

- Our goal:
  Minimize $\sum_{i=1}^{k} \Delta(b_i)$, where k is the number of buckets specified by the user.

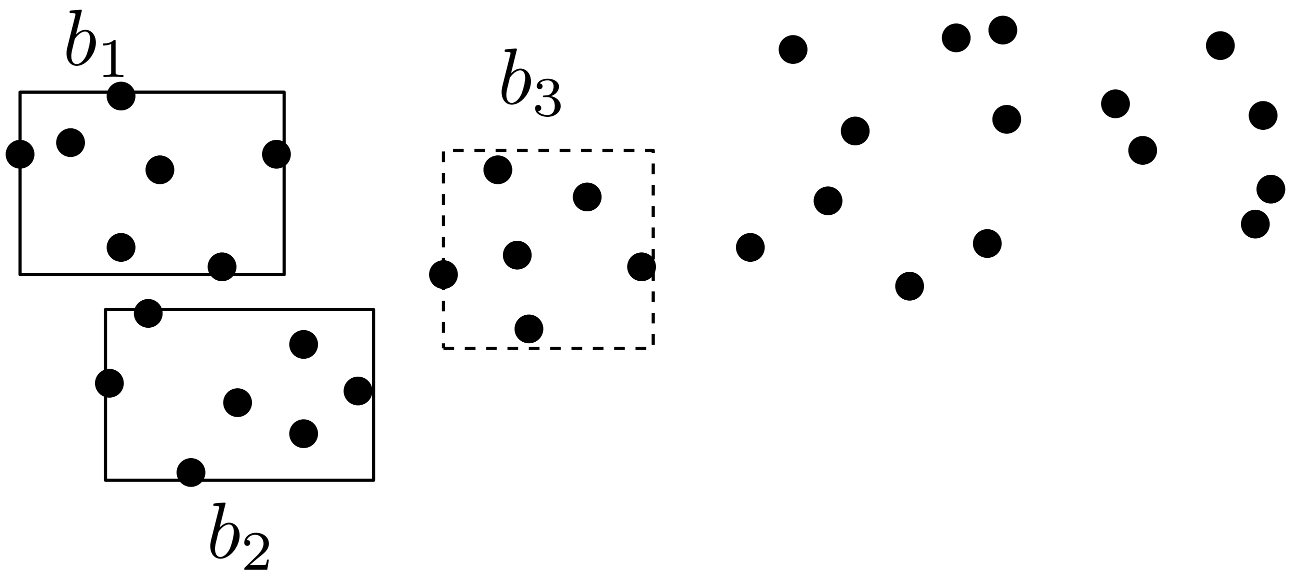
The partitioning metric

- Neighborhood and uniformity quality of b:
  $\Delta(b) = \eta_b n_b \sum_{1,\ldots,d} X_i$,
  $\{X_1, \ldots, X_d\}$: the side lengths of b in each dimension.

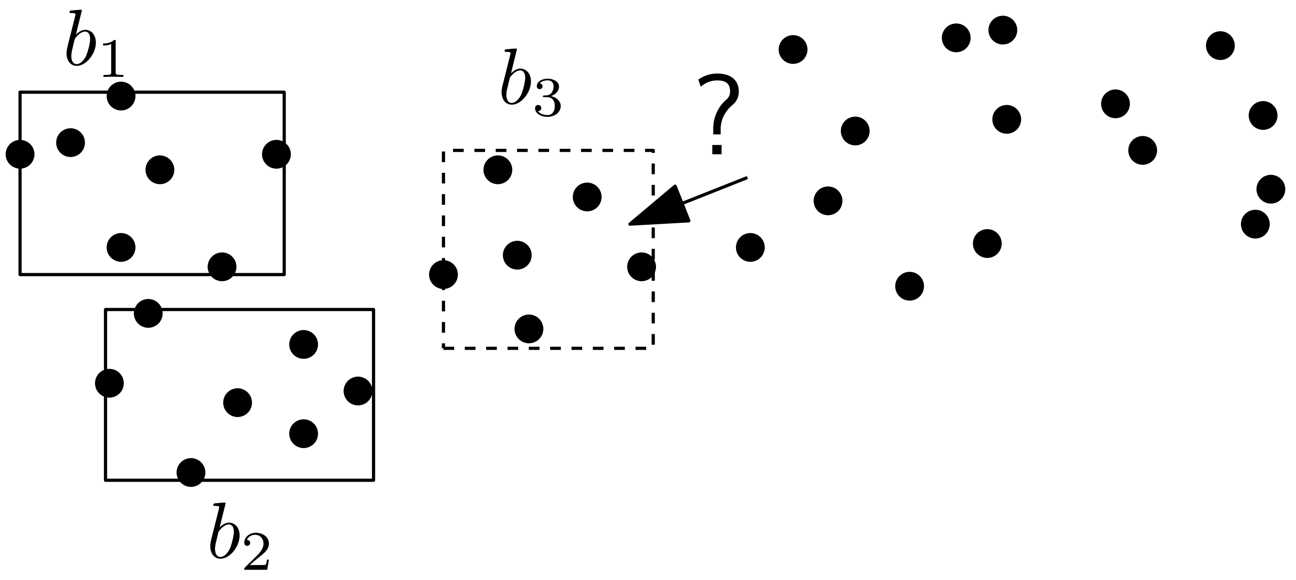
- Our goal:
  Minimize $\sum_{i=1}^{k} \Delta(b_i)$, where k is the number of buckets specified by the user.

- The greedy algorithm;
  The adaptive R-tree algorithm.

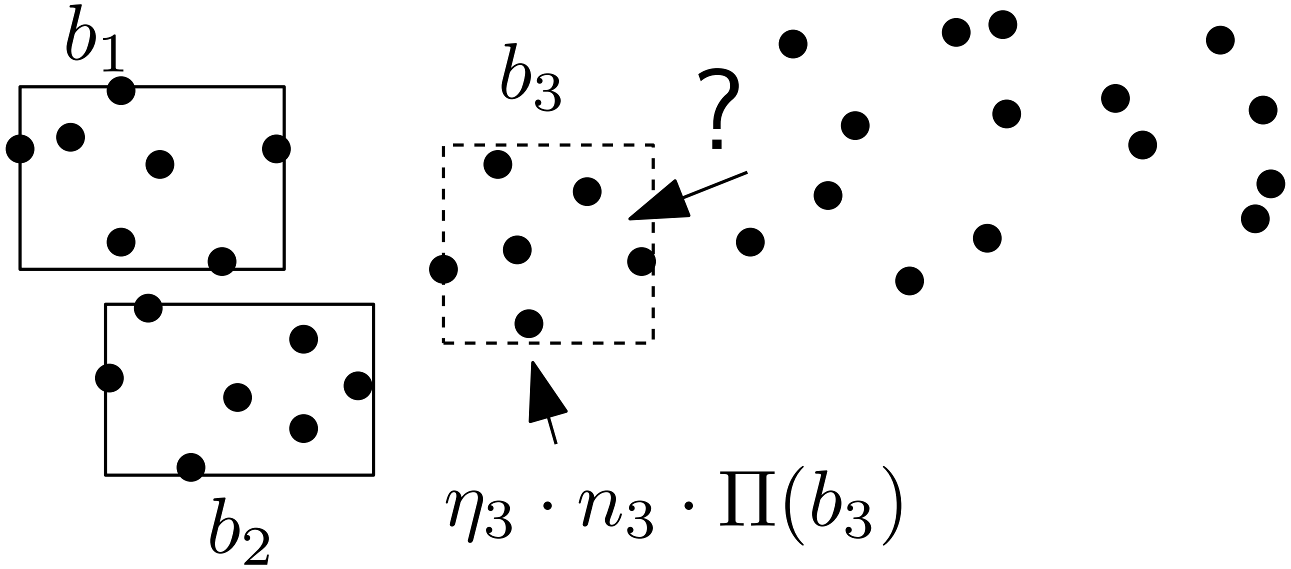
The greedy algorithm



The greedy algorithm



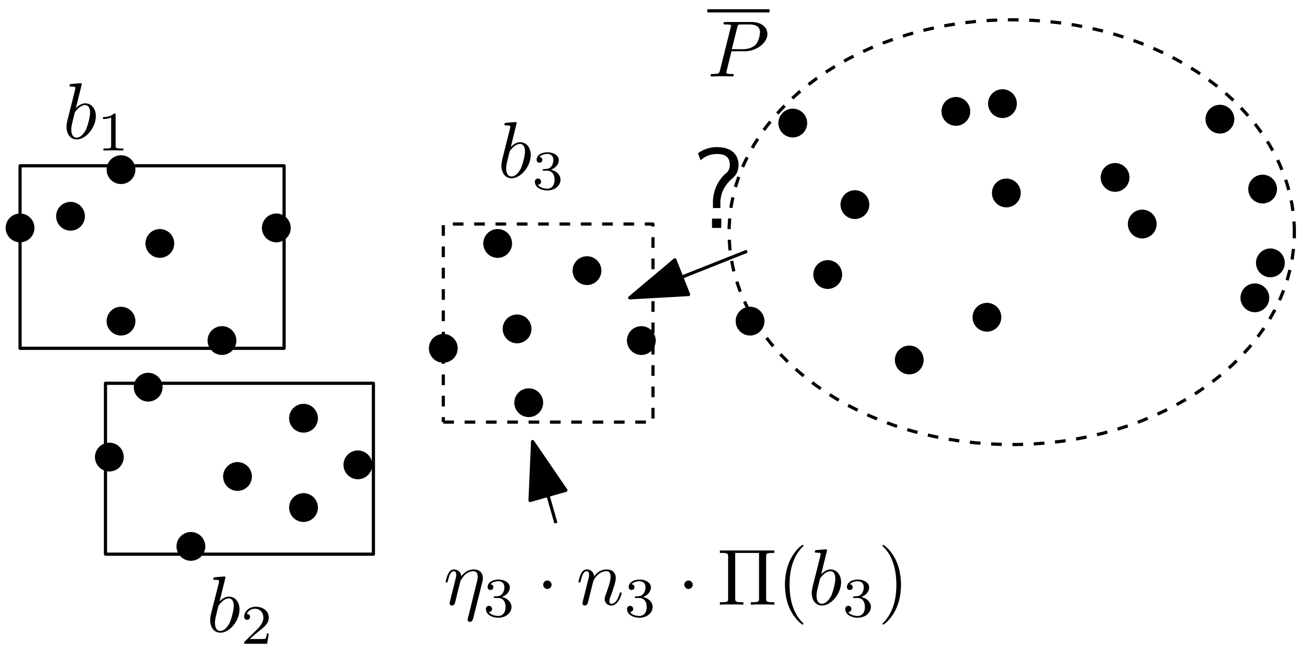
The greedy algorithm



$\Pi(b_i)$: the perimeter of b_i.

η_i: the number of neighborhoods of strings in b_i.
n_i: the number of points in b_i.

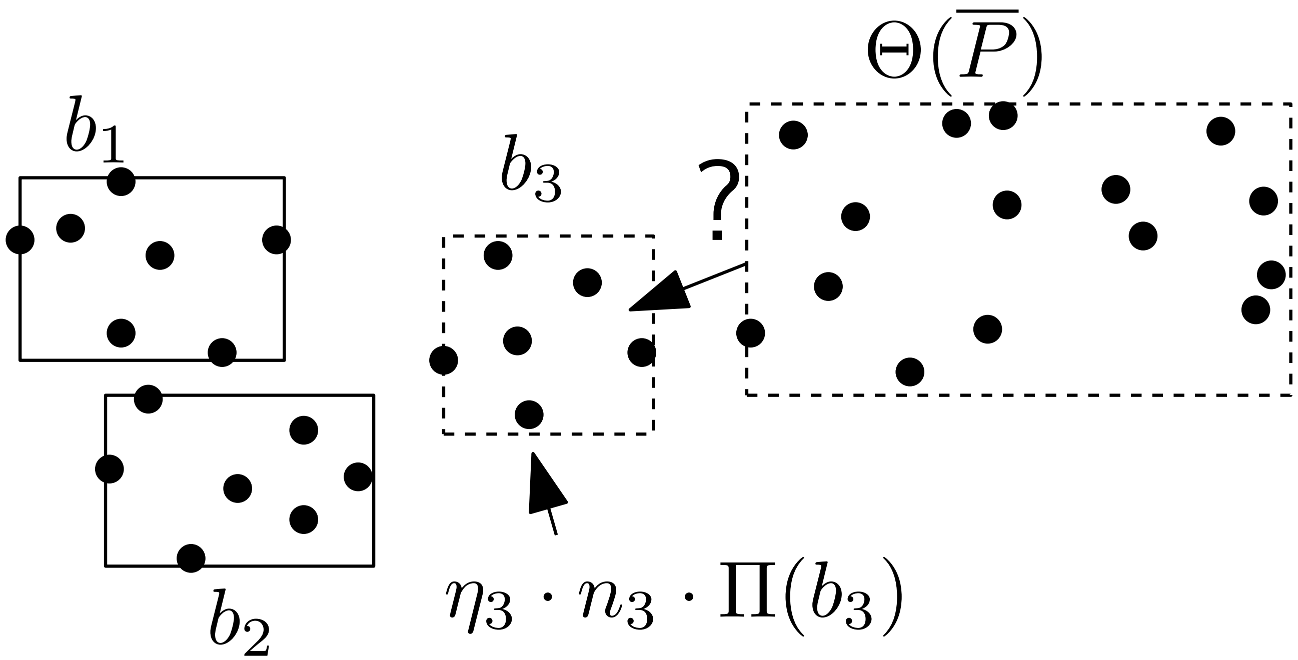
The greedy algorithm



$\Pi(b_i)$: the perimeter of b_i.

η_i: the number of neighborhoods of strings in b_i.
n_i: the number of points in b_i.

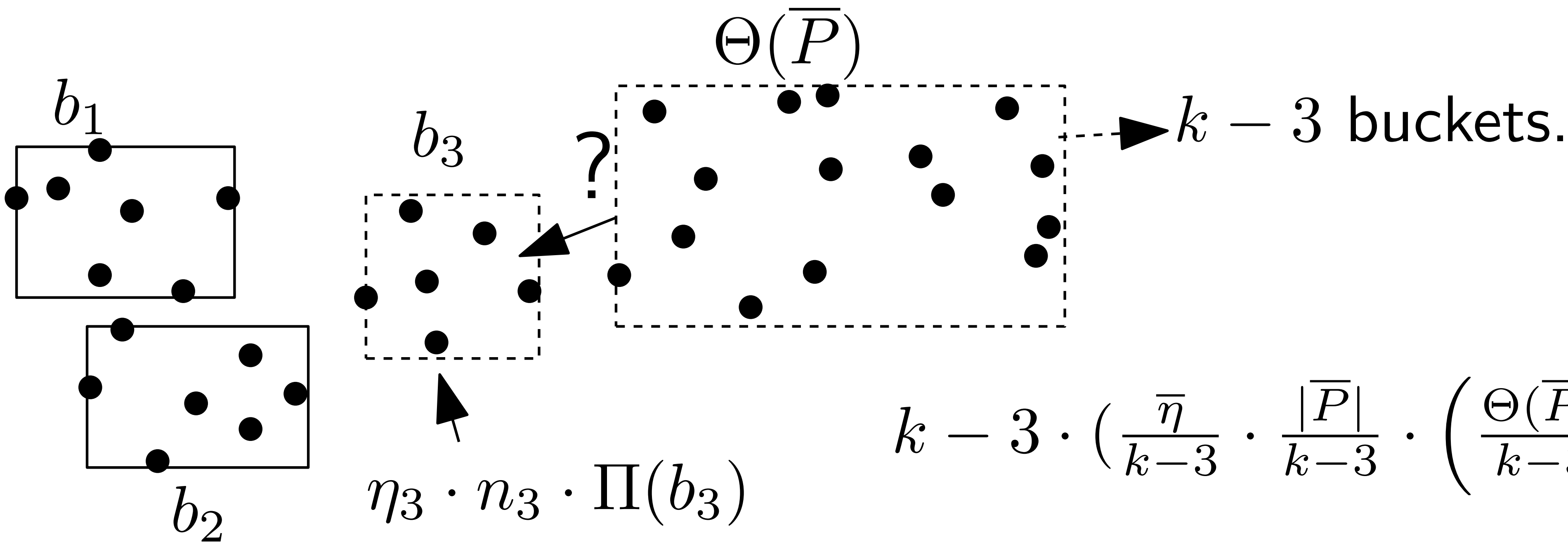
The greedy algorithm



$\Pi(b_i)$: the perimeter of b_i.

η_i: the number of neighborhoods of strings in b_i.
n_i: the number of points in b_i.

The greedy algorithm

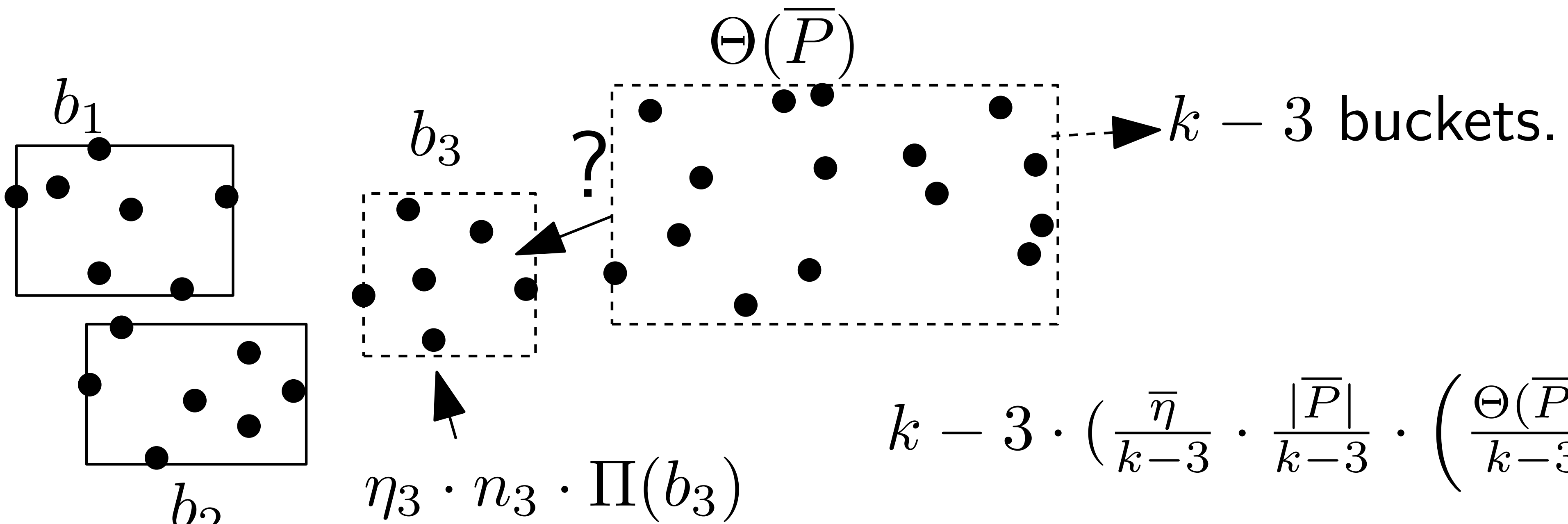


$$k - 3 \cdot \left(\frac{\overline{\eta}}{k-3} \cdot \frac{|\overline{P}|}{k-3} \cdot \left(\frac{\Theta(\overline{P})}{k-3} \right)^{1/d} \cdot d \right)$$

$\Pi(b_i)$: the perimeter of b_i.

η_i: the number of neighborhoods of strings in b_i.
n_i: the number of points in b_i.
$\overline{\eta}$: number of neighborhoods of strings for remaining points.

The greedy algorithm



$$k - 3 \cdot (\frac{\overline{\eta}}{k-3} \cdot \frac{|\overline{P}|}{k-3} \cdot \left(\frac{\Theta(\overline{P})}{k-3}\right)^{1/d} \cdot d)$$

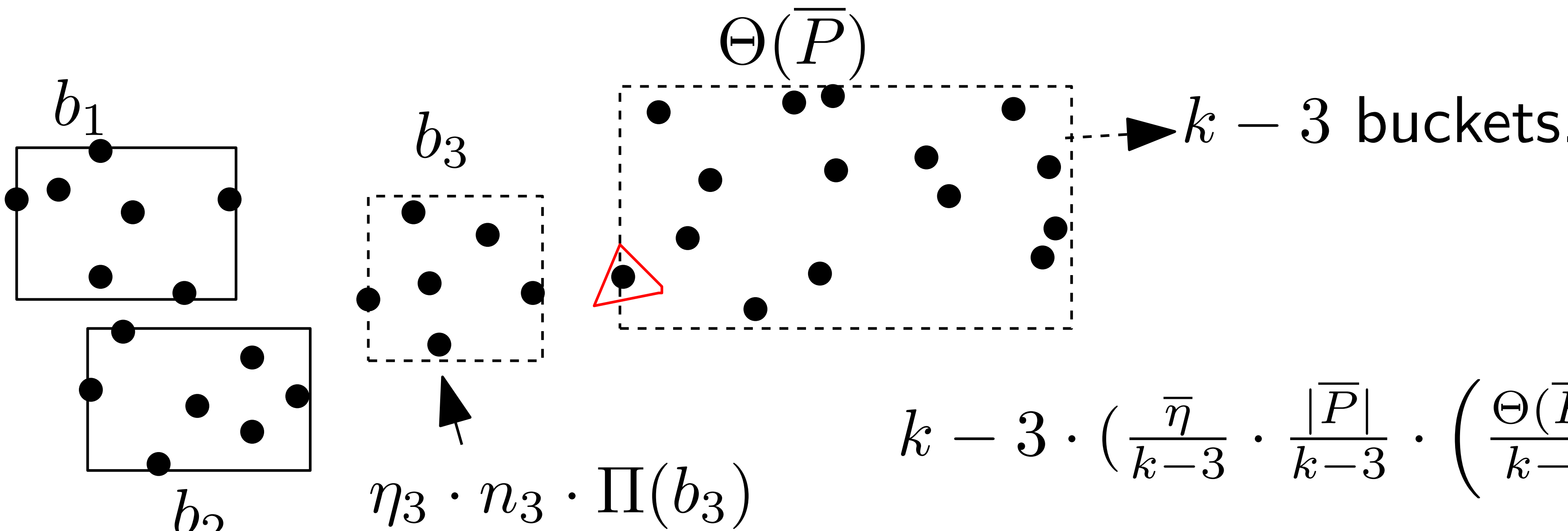
$\Pi(b_i)$: the perimeter of b_i.

η_i: the number of neighborhoods of strings in b_i.
n_i: the number of points in b_i.
$\overline{\eta}$: number of neighborhoods of strings for remaining points.

$$U(b_i) = \eta_i \cdot n_i \cdot \Pi(\mathbf{b}_i) + \frac{\overline{\eta}}{k-i} \cdot |\overline{P}| \cdot \left(\frac{\Theta(\overline{P})}{k-i}\right)^{1/d} \cdot d$$

($\Pi(b_i)$ is the perimeter of bucket b_i)

The greedy algorithm



$$k - 3 \cdot \left(\frac{\overline{\eta}}{k-3} \cdot \frac{|\overline{P}|}{k-3} \cdot \left(\frac{\Theta(\overline{P})}{k-3} \right)^{1/d} \cdot d \right)$$

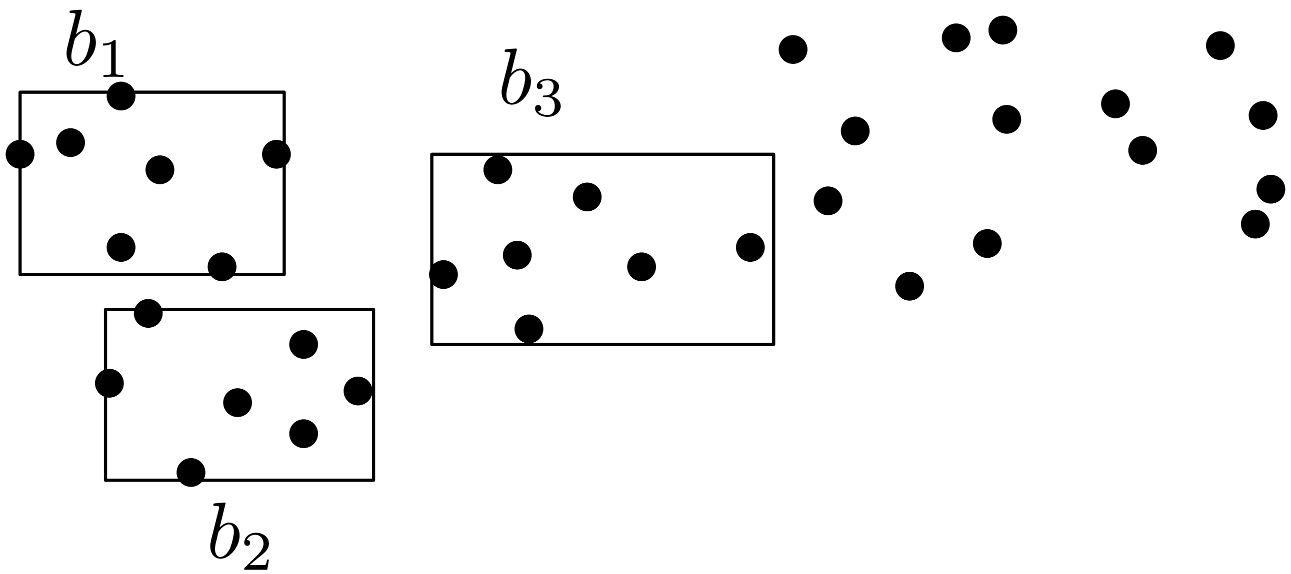
$\Pi(b_i)$: the perimeter of b_i.

η_i: the number of neighborhoods of strings in b_i.
n_i: the number of points in b_i.
$\overline{\eta}$: number of neighborhoods of strings for remaining points.

$$U(b_i) = \eta_i \cdot n_i \cdot \Pi(\mathbf{b}_i) + \frac{\overline{\eta}}{k-i} \cdot |\overline{P}| \cdot \left(\frac{\Theta(\overline{P})}{k-i} \right)^{1/d} \cdot d$$

($\Pi(b_i)$ is the perimeter of bucket b_i)

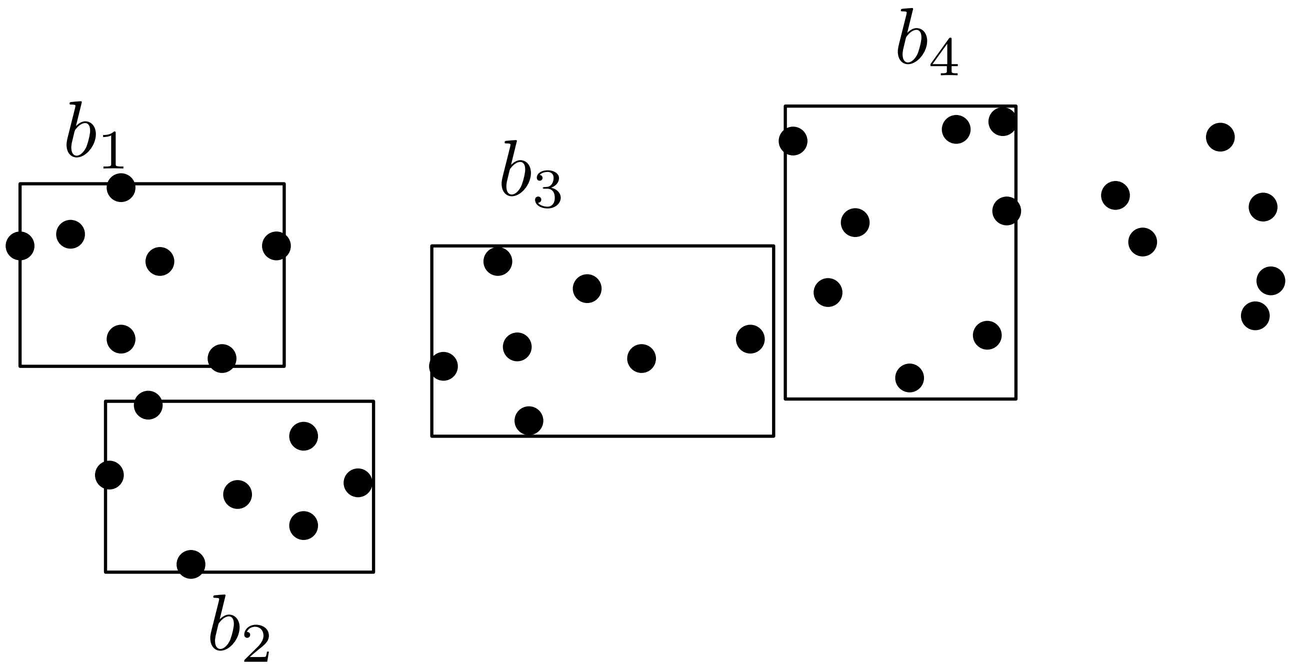
The greedy algorithm



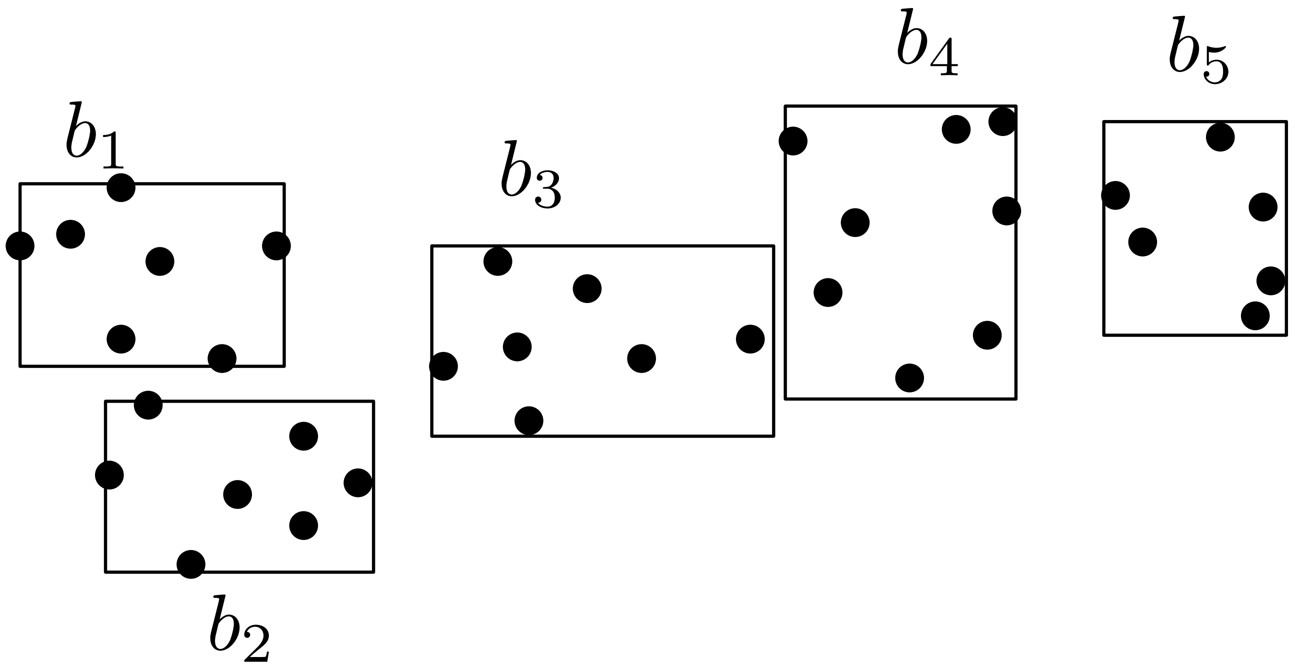
$$U(b_i) = \eta_i \cdot n_i \cdot \Pi(\mathbf{b}_i) + \frac{\overline{\eta}}{k-i} \cdot |\overline{P}| \cdot \left(\frac{\Theta(\overline{P})}{k-i}\right)^{1/d} \cdot d$$

($\Pi(b_i)$ is the perimeter of bucket b_i)

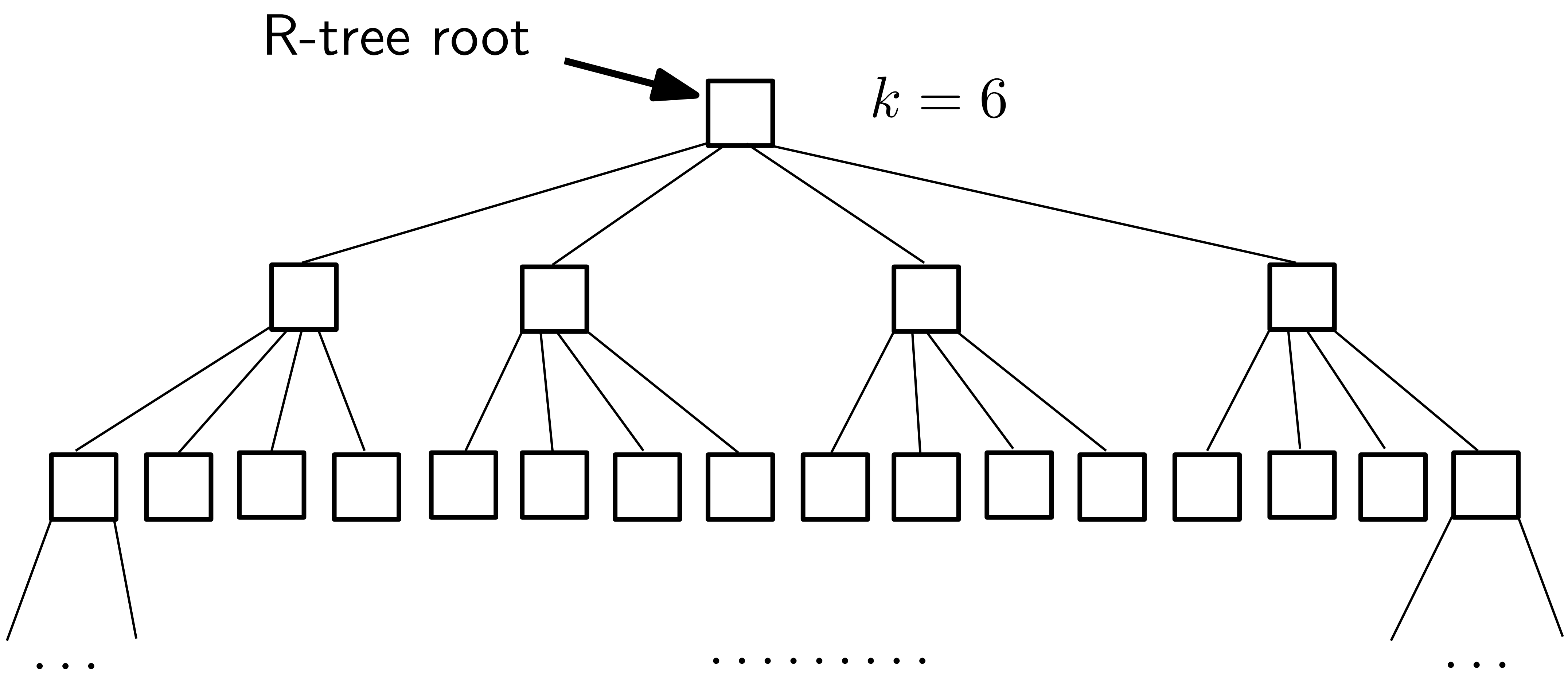
The greedy algorithm



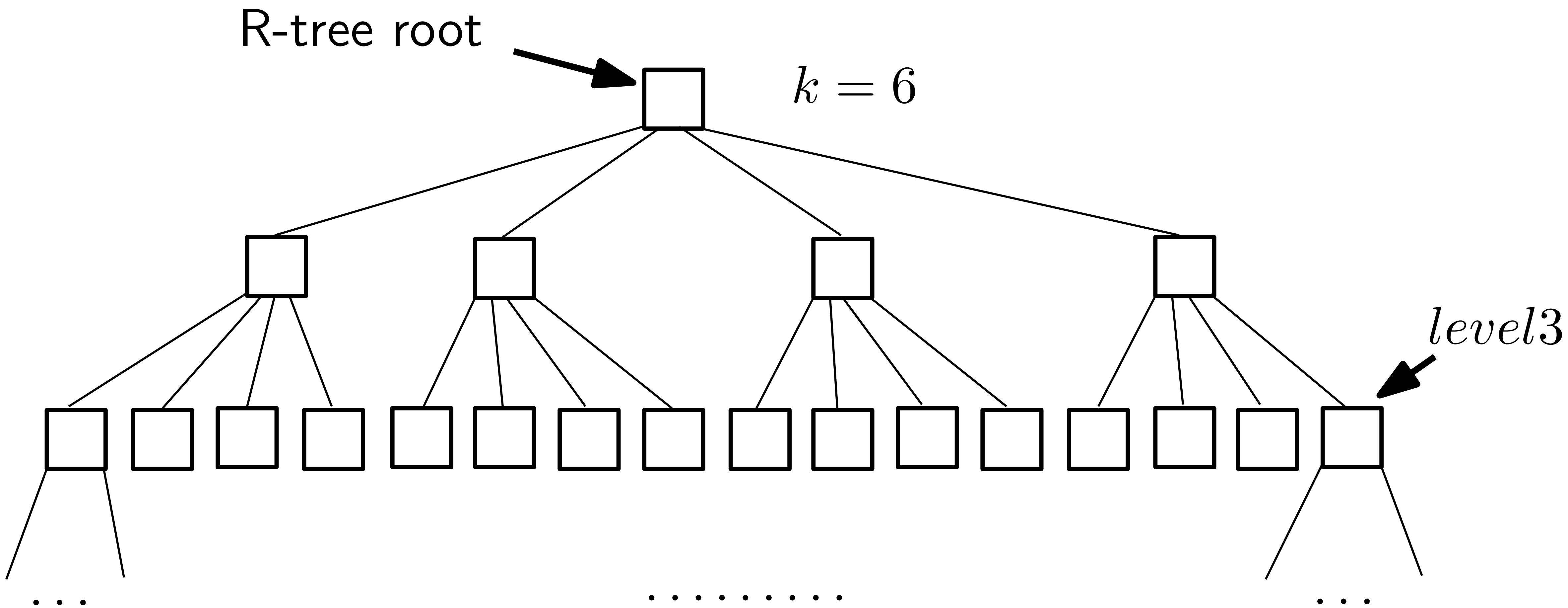
The greedy algorithm



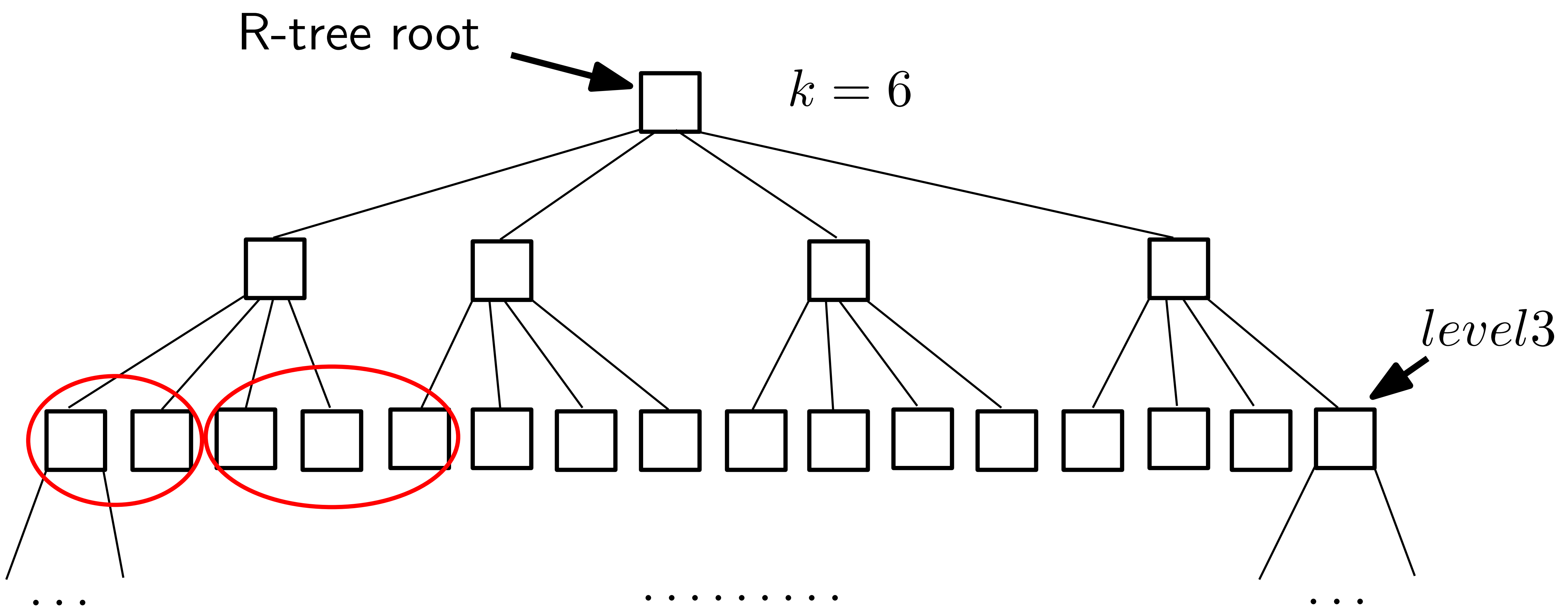
The adaptive R-tree algorithm



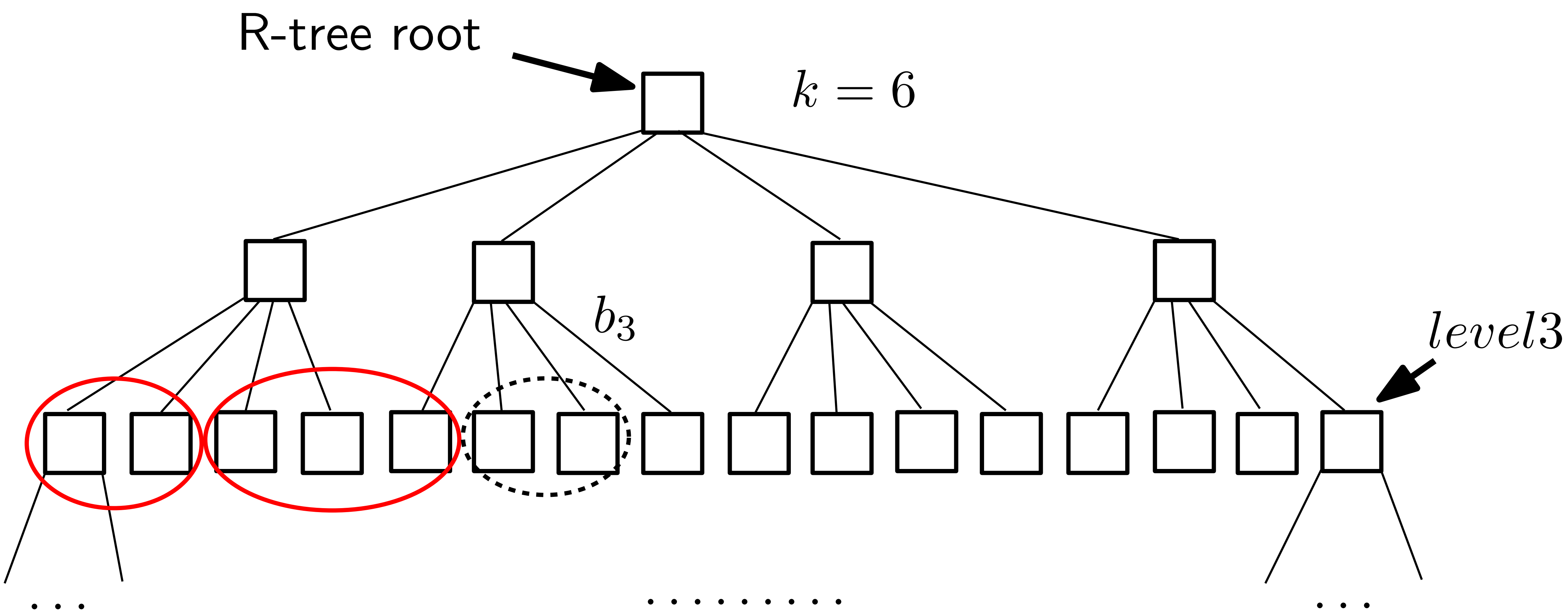
The adaptive R-tree algorithm



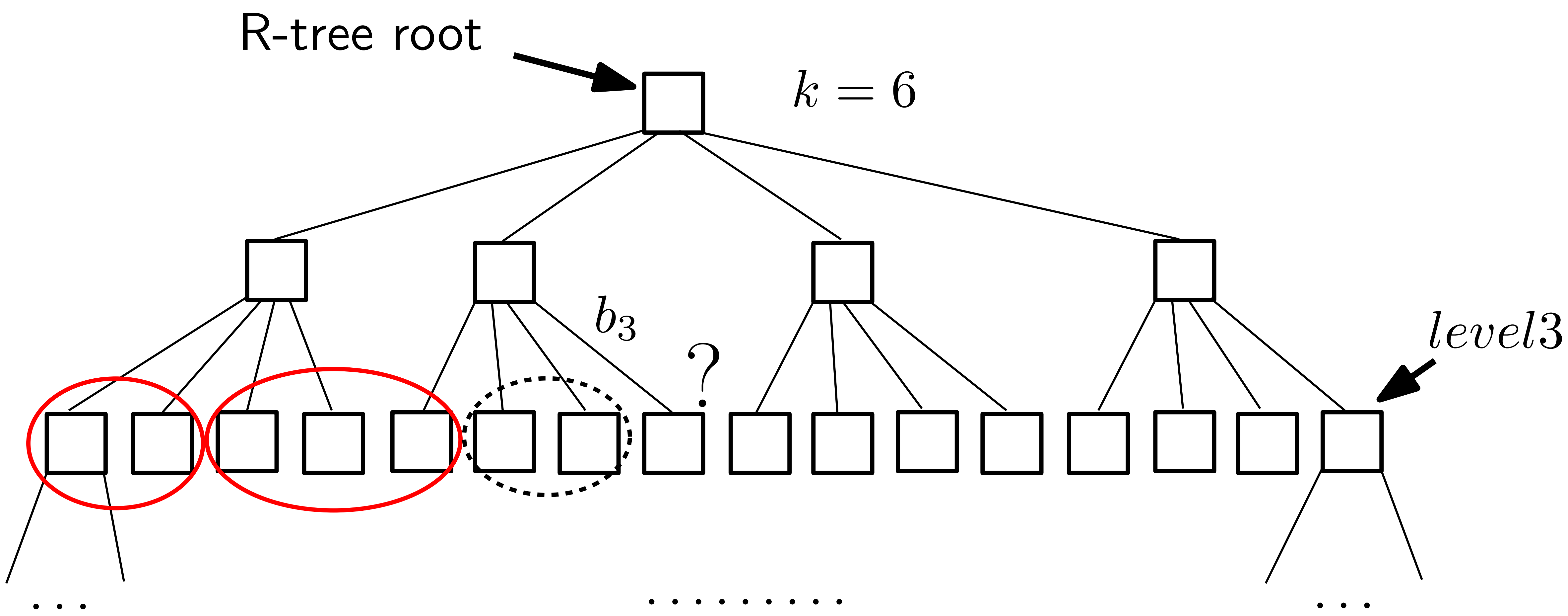
The adaptive R-tree algorithm



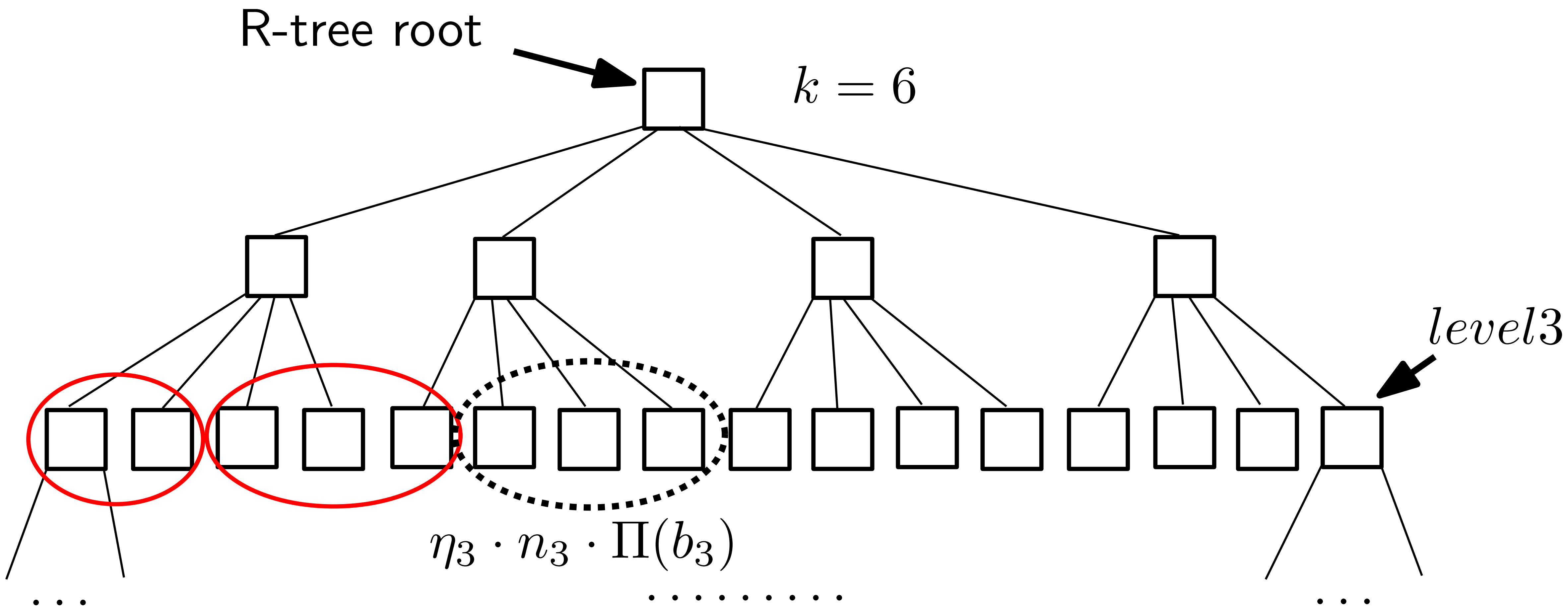
The adaptive R-tree algorithm



The adaptive R-tree algorithm

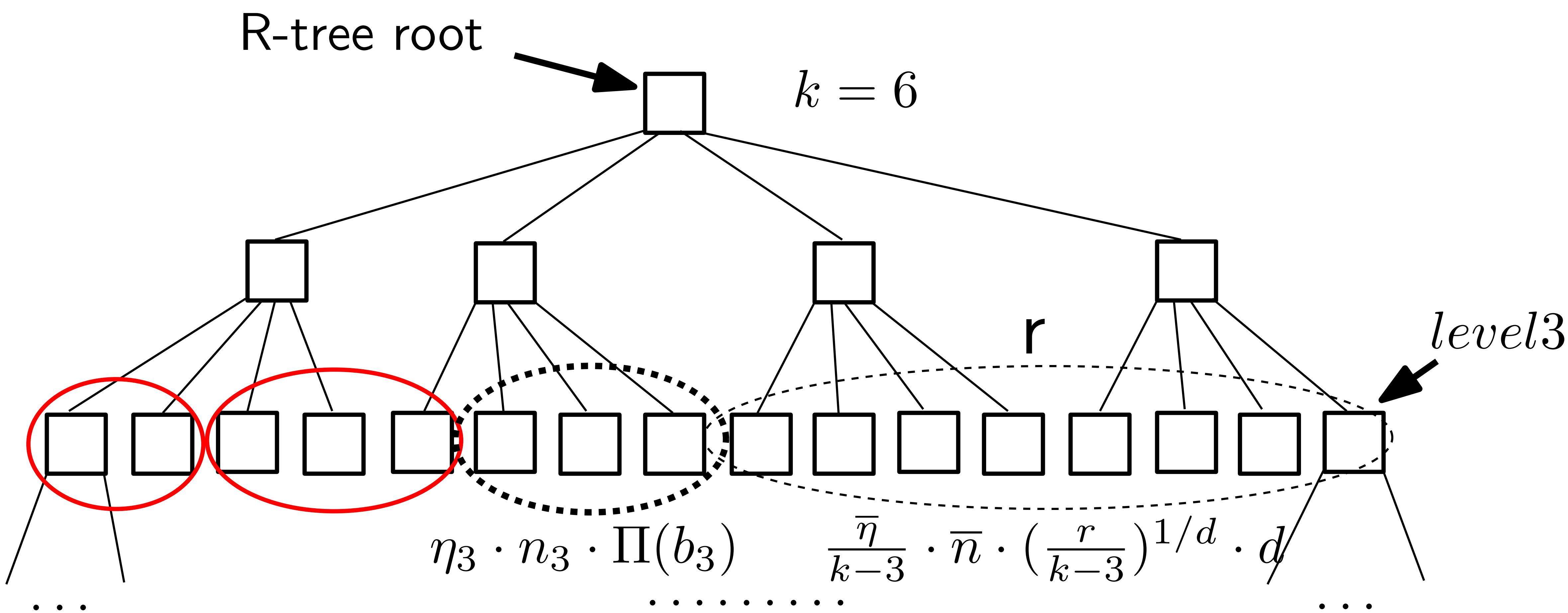


The adaptive R-tree algorithm



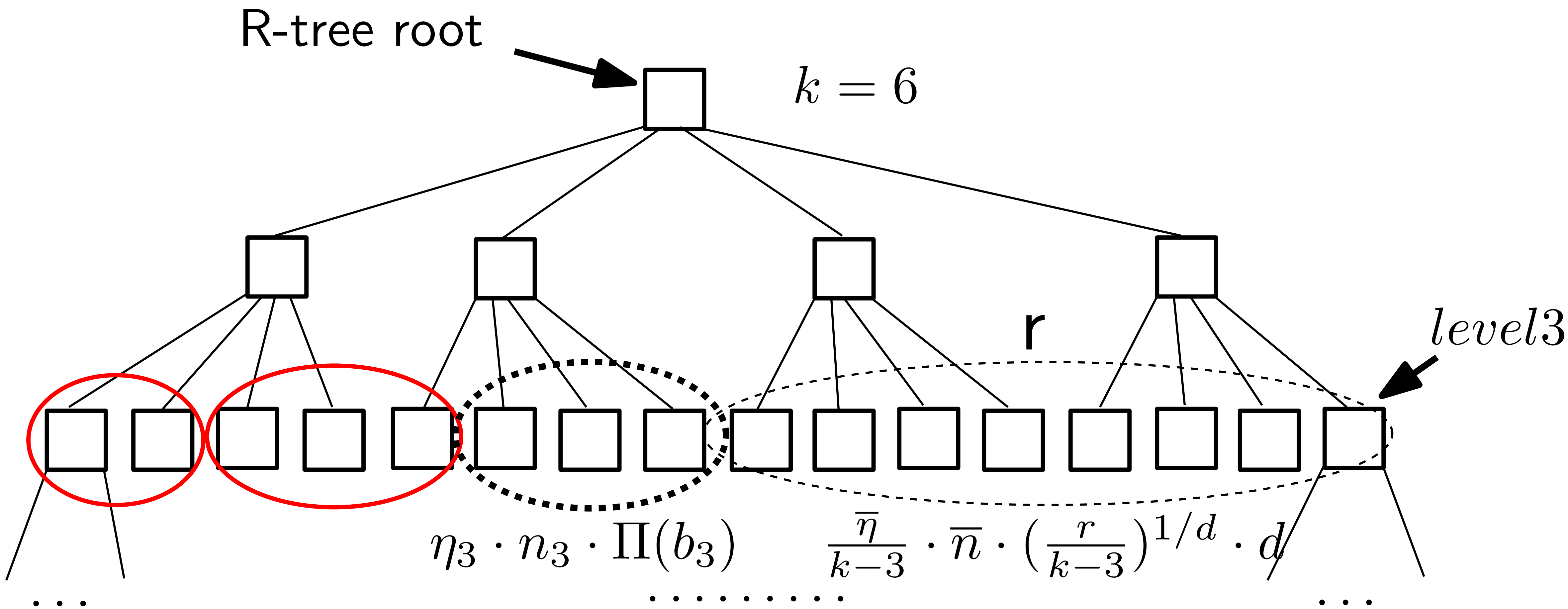
η_i: the number of neighborhoods of strings in b_i.
n_i: the number of points in b_i.
$\Pi(b_i)$: the perimeter of b_i.

The adaptive R-tree algorithm



η_i: the number of neighborhoods of strings in b_i.
n_i: the number of points in b_i.
$\Pi(b_i)$: the perimeter of b_i.
$\overline{\eta}$: number of neighborhoods of strings for remaining points.

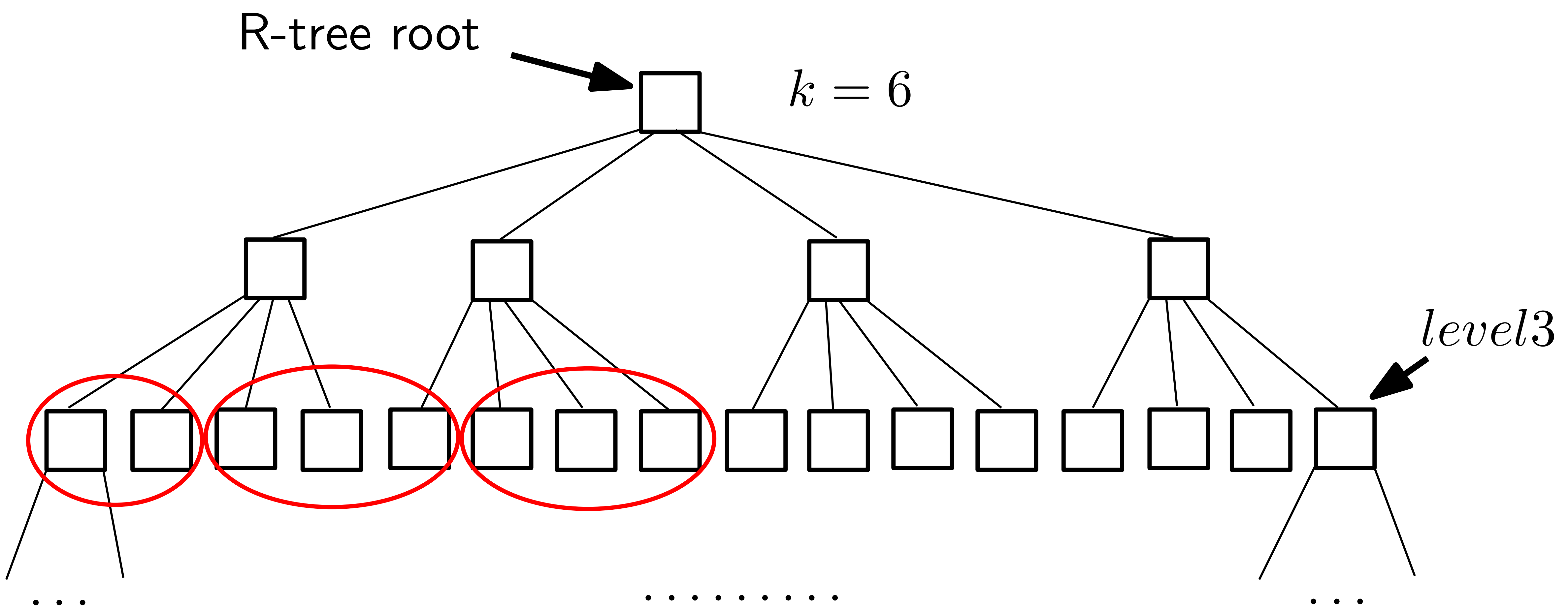
The adaptive R-tree algorithm



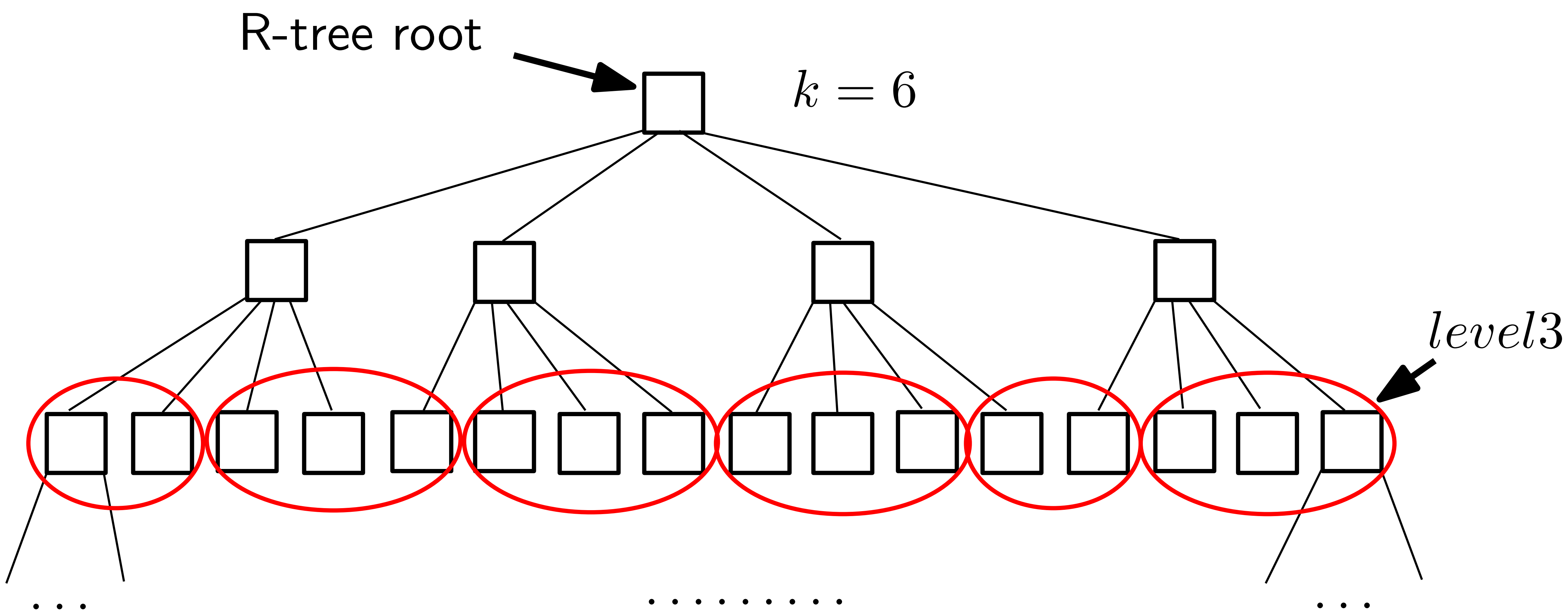
$$U'(b_i) = \eta_i \cdot n_i \cdot \Pi(\mathbf{b}_i) + \frac{\overline{\eta}}{k-i} \cdot \overline{n} \cdot (\frac{r}{k-i})^{1/d} \cdot d.$$

η_i: the number of neighborhoods of strings in b_i.
n_i: the number of points in b_i.
$\Pi(b_i)$: the perimeter of b_i.
$\overline{\eta}$: number of neighborhoods of strings for remaining points.

The adaptive R-tree algorithm



The adaptive R-tree algorithm



Experiment setup

- All experiments were executed on a Linux machine with an Intel Xeon CPU at 2GHz and 2GB of memory.

Experiment setup

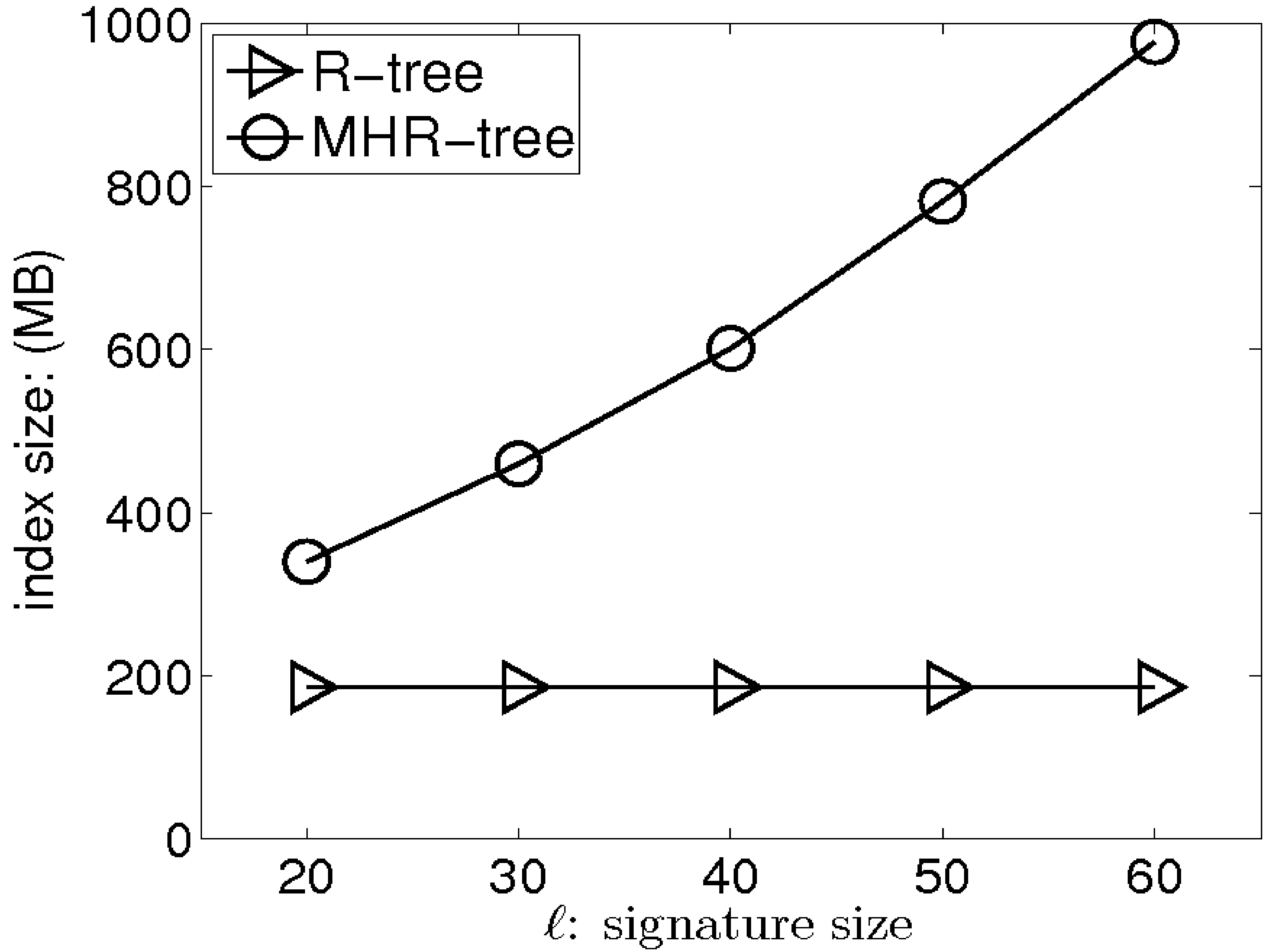
- All experiments were executed on a Linux machine with an Intel Xeon CPU at 2GHz and 2GB of memory.
- Real data sets:
  The road-networks for states (Texas, California) in USA, each point is associated with the names of the state, county and town.
  Synthetic data sets: uniform points and random clustered points. Assign strings from real data sets randomly to the point generated.

Experiment setup

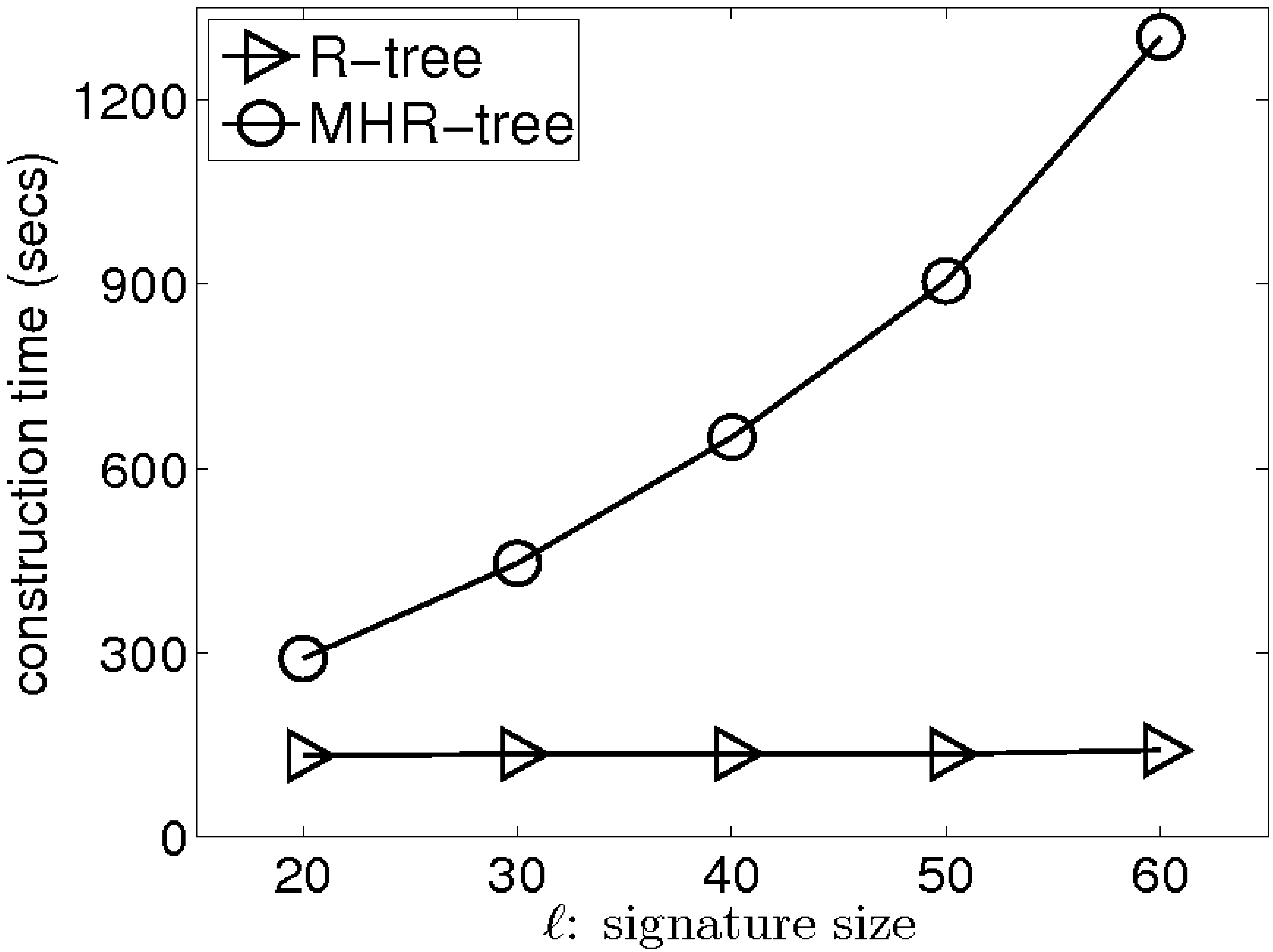
- All experiments were executed on a Linux machine with an Intel Xeon CPU at 2GHz and 2GB of memory.
- Real data sets:
  The road-networks for states (Texas, California) in USA, each point is associated with the names of the state, county and town.
  Synthetic data sets: uniform points and random clustered points.
  Assign strings from real data sets randomly to the point generated.
- The default experimental parameters are summarized below.

Symbol	Definition	Default Value
θ	query area percentage of data space	3%
N	size of points set	2,000,000
l	signature length	50
τ	edit distance threshold	2
d	dimensionality	2

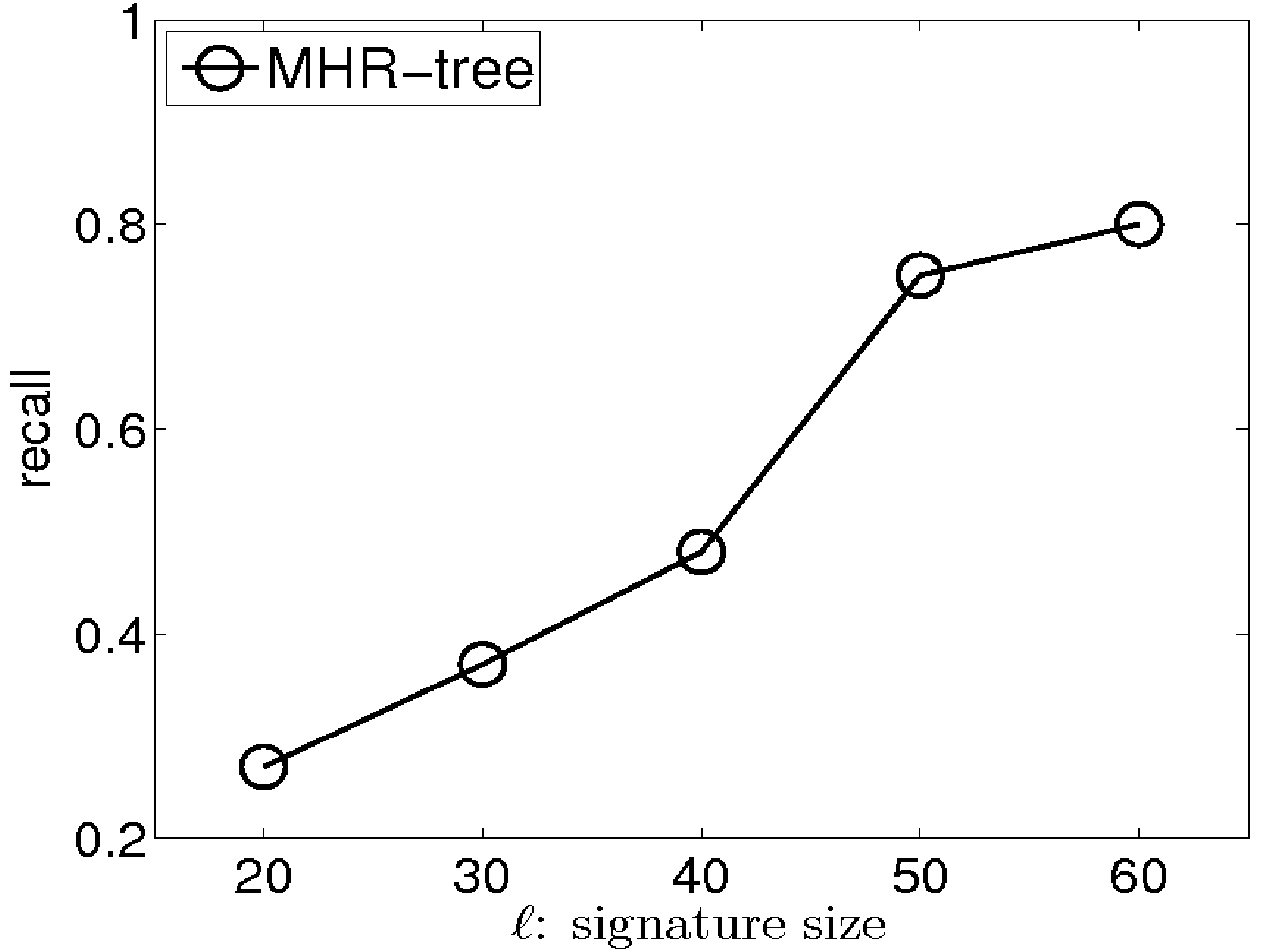
SAS range queries: impact of the signature size



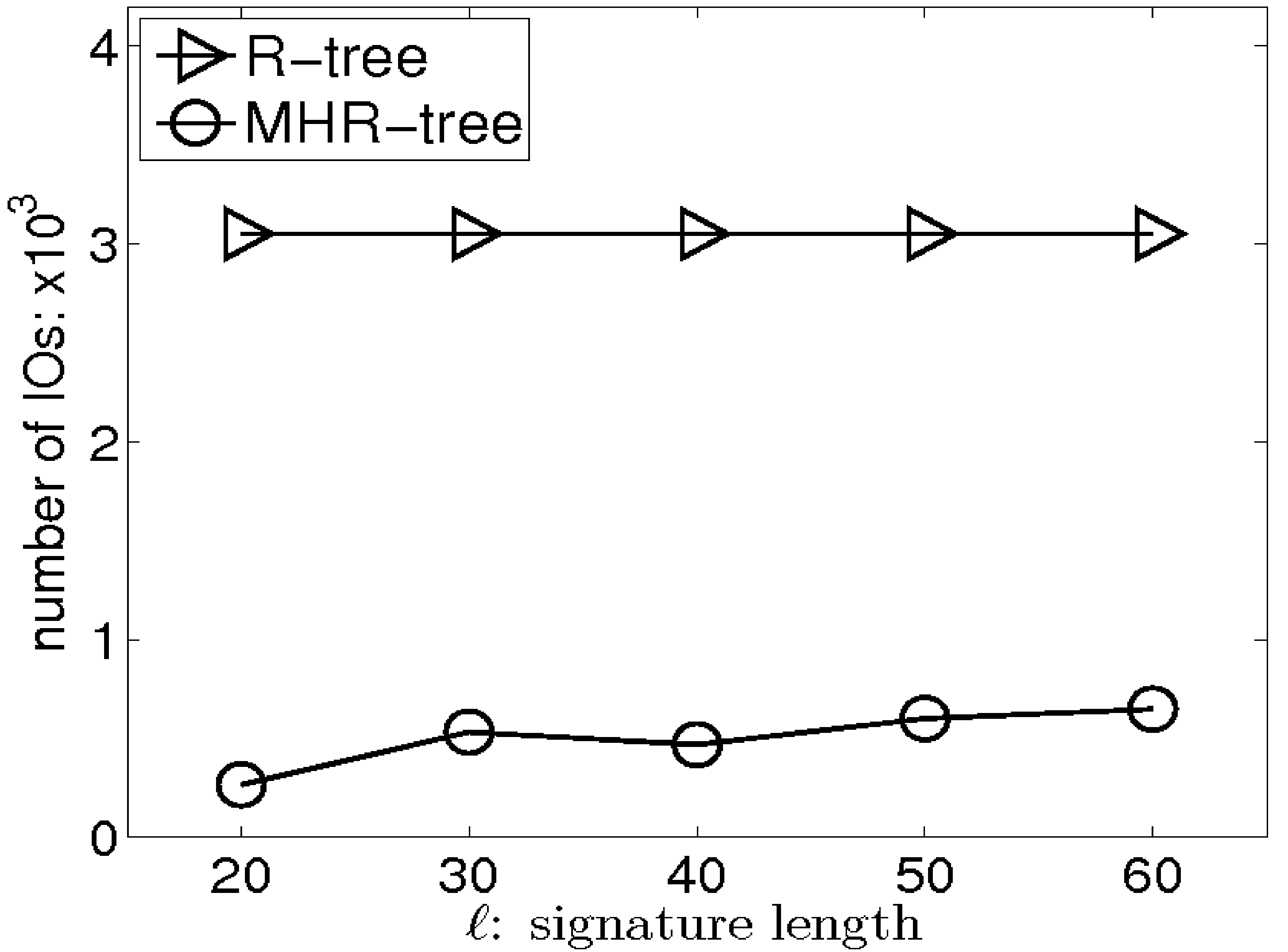
SAS range queries: impact of the signature size



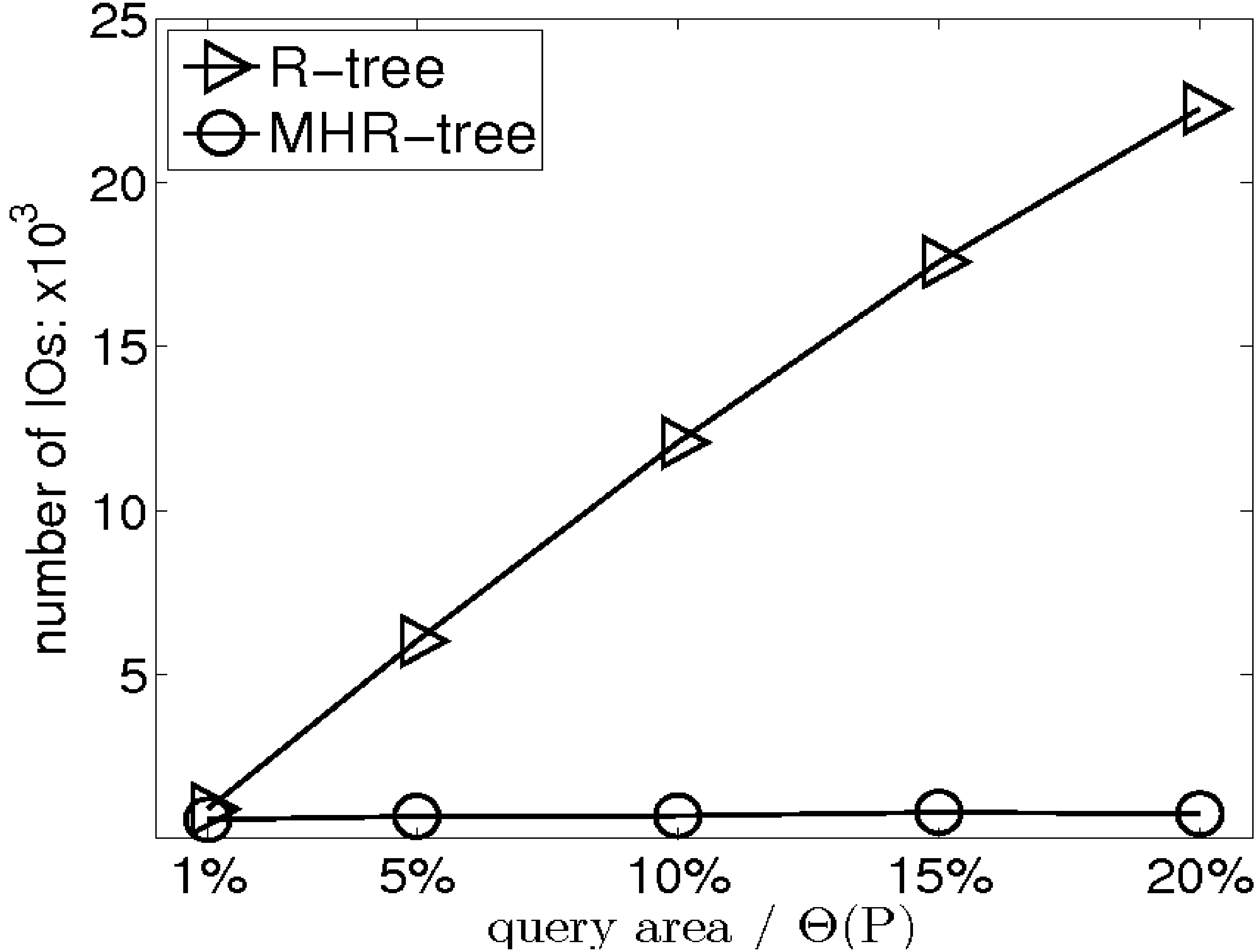
SAS range queries: impact of the signature size



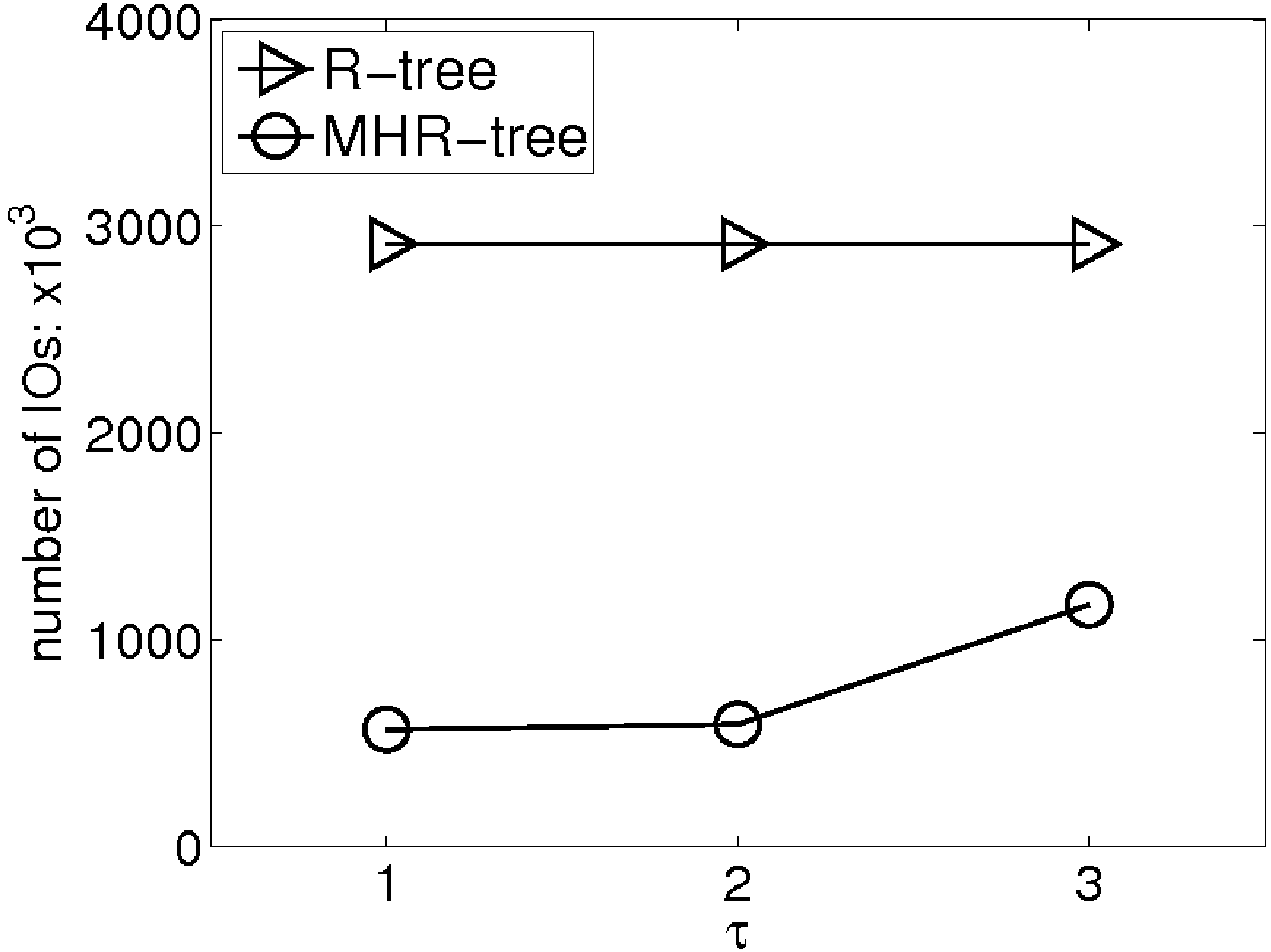
SAS range queries: impact of the signature size



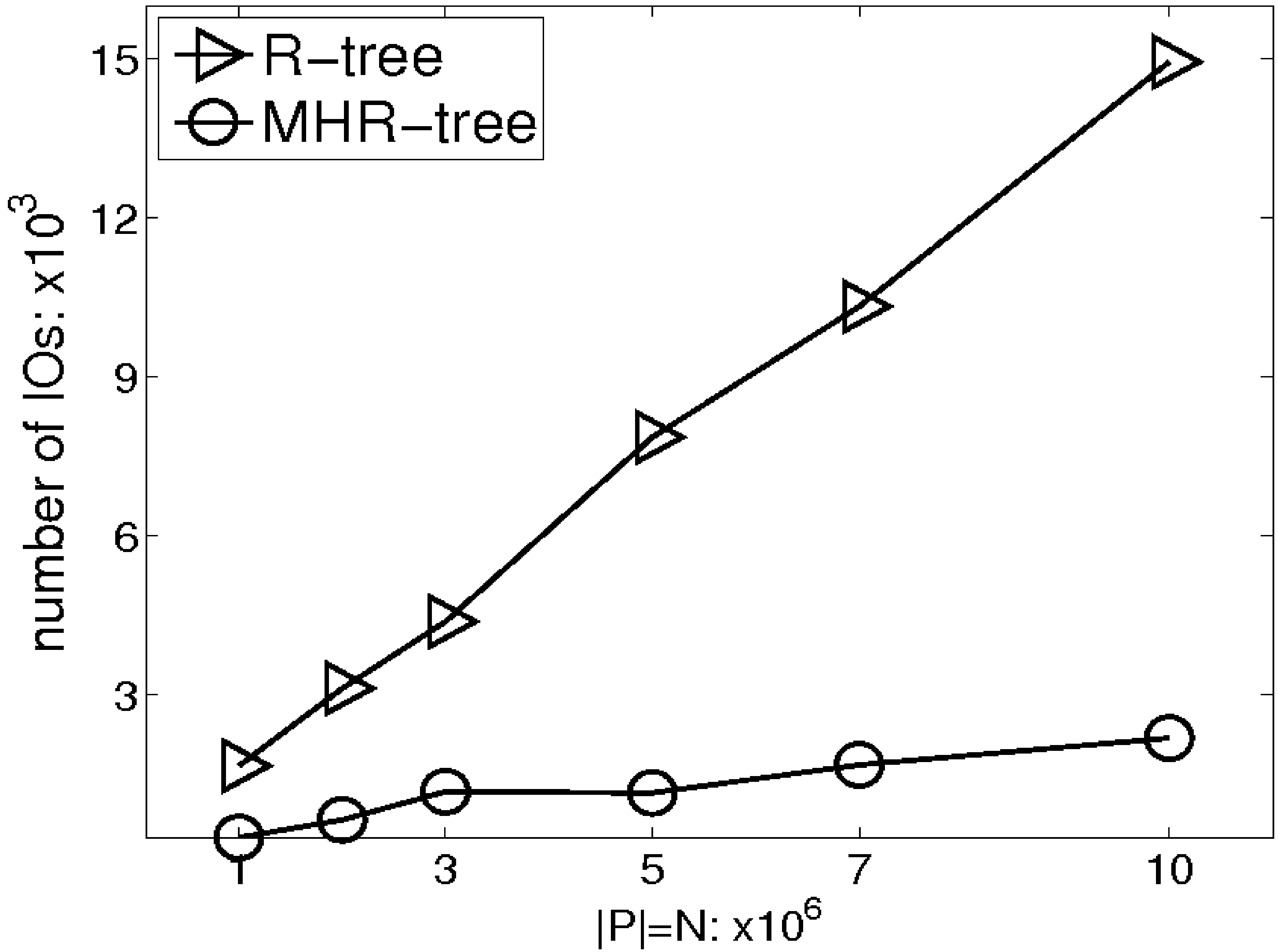
SAS range queries: query performance



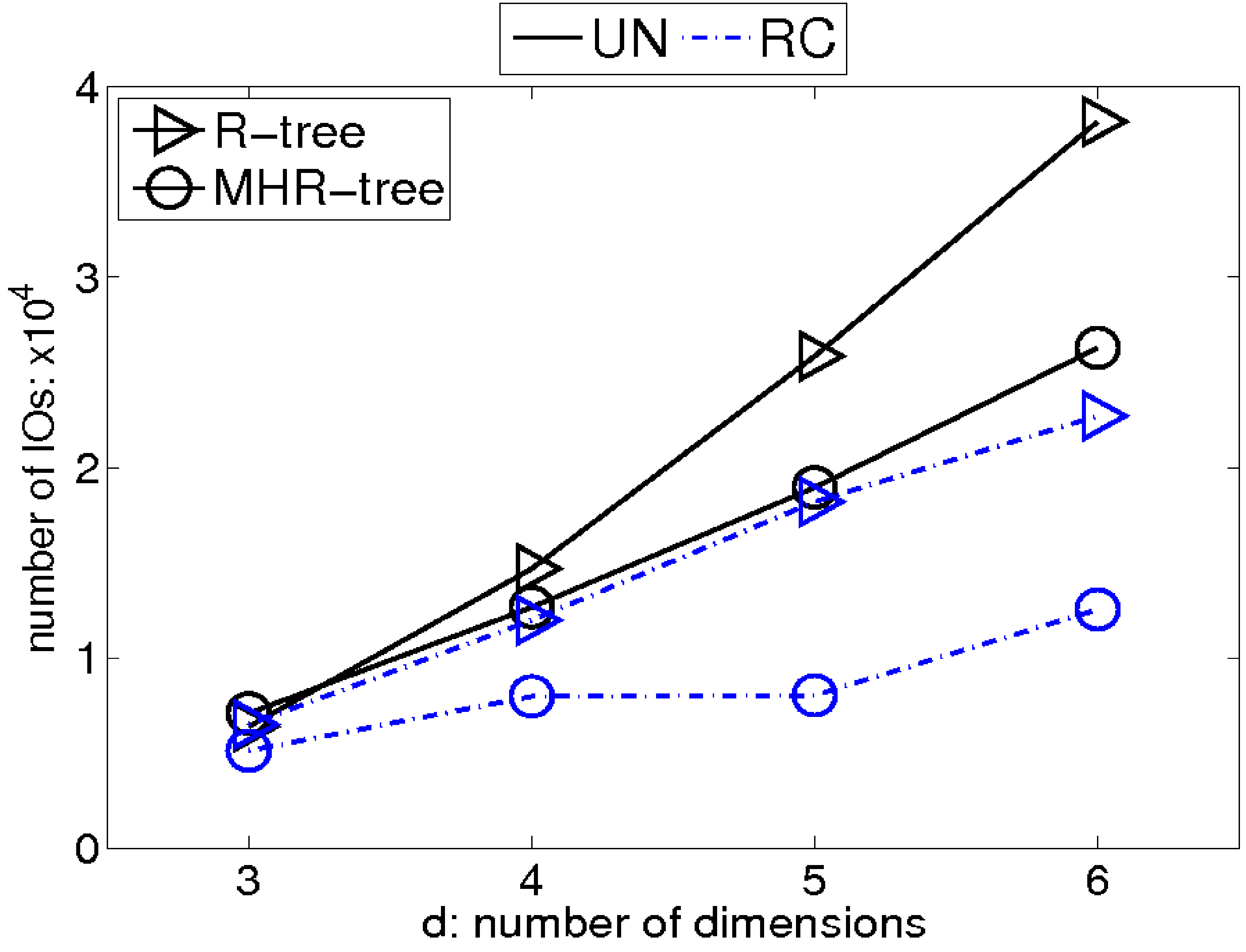
SAS range queries: query performance



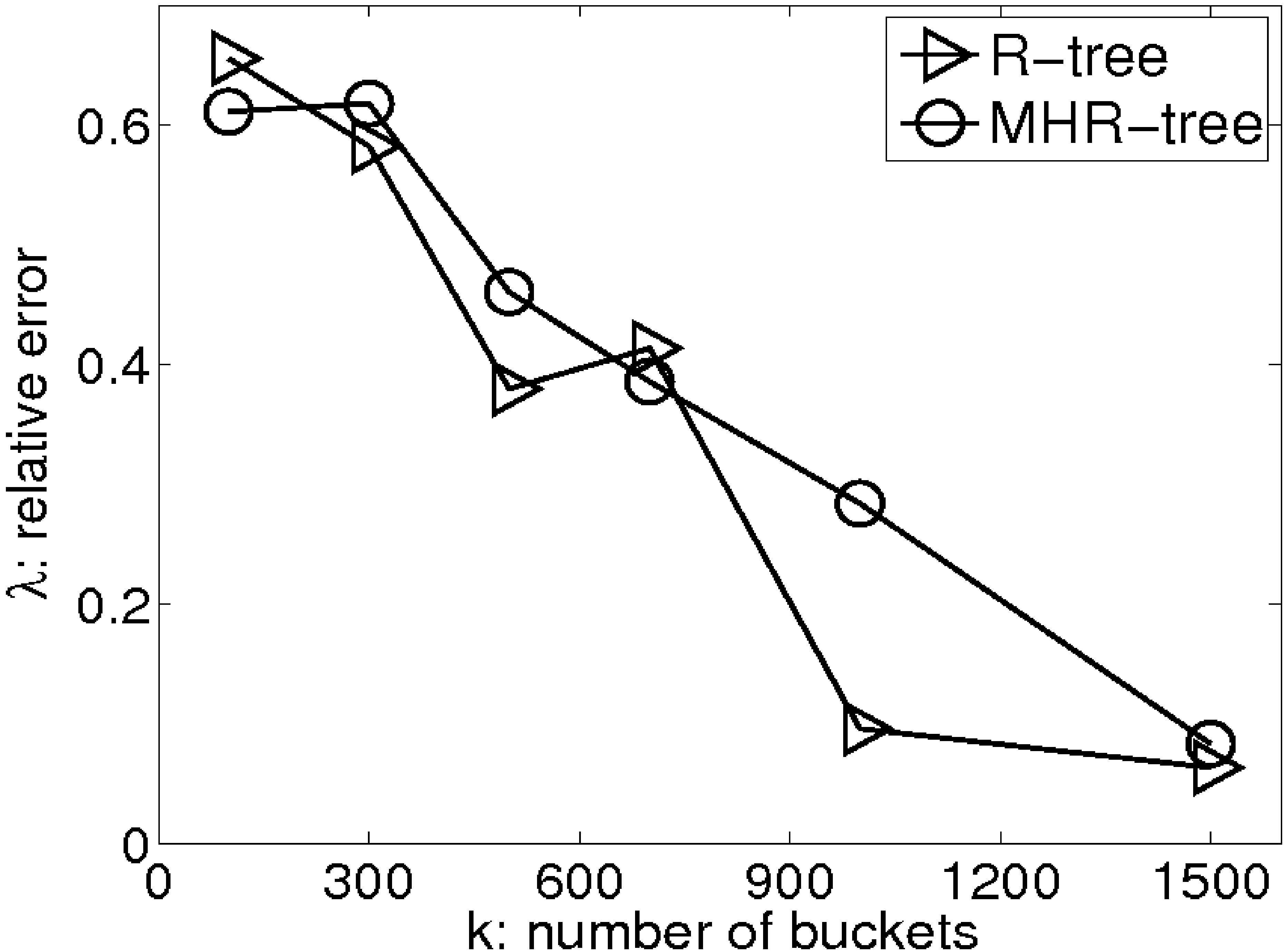
SAS range queries: query performance



SAS range queries: query performance

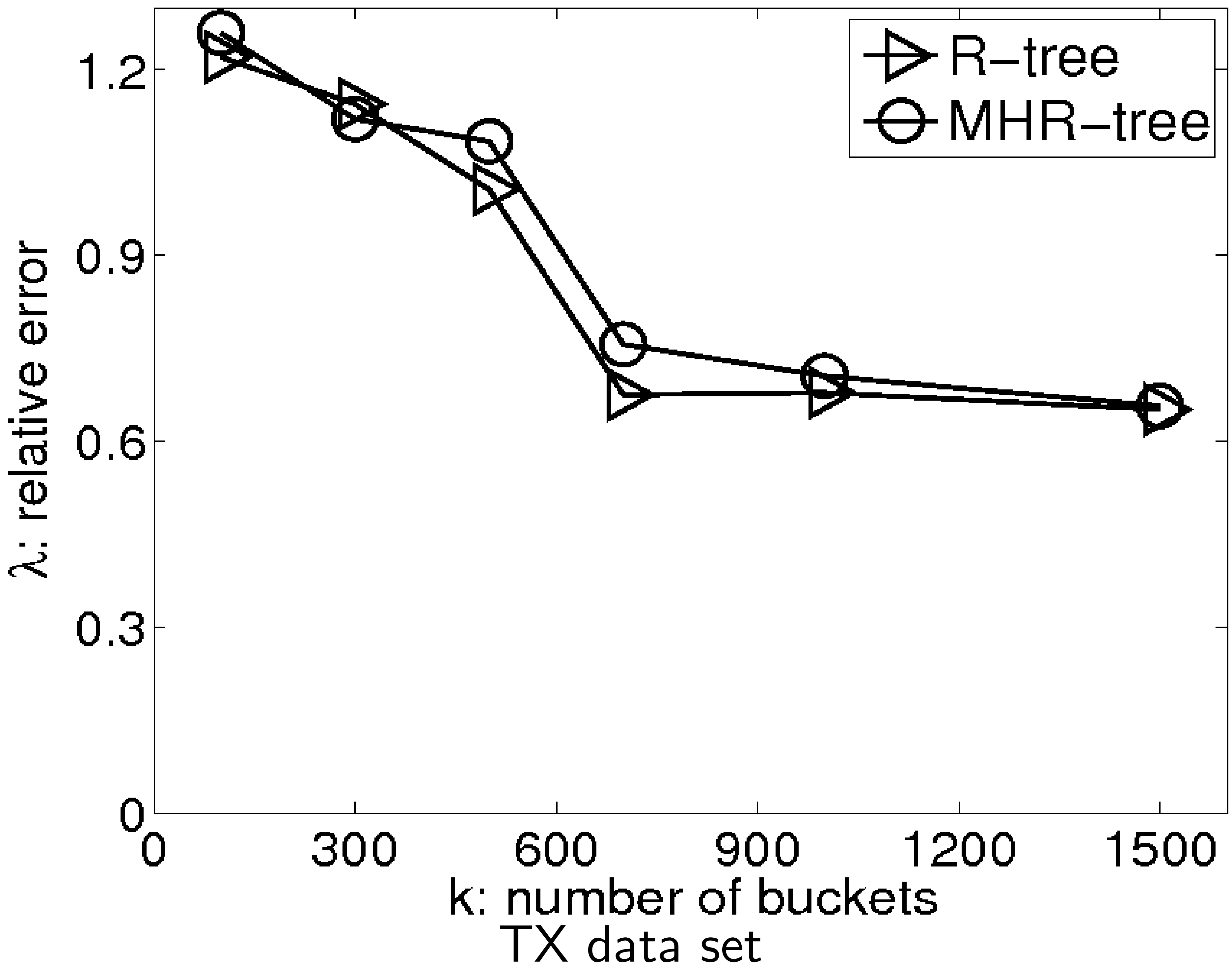


Selectivity estimation: relative errors

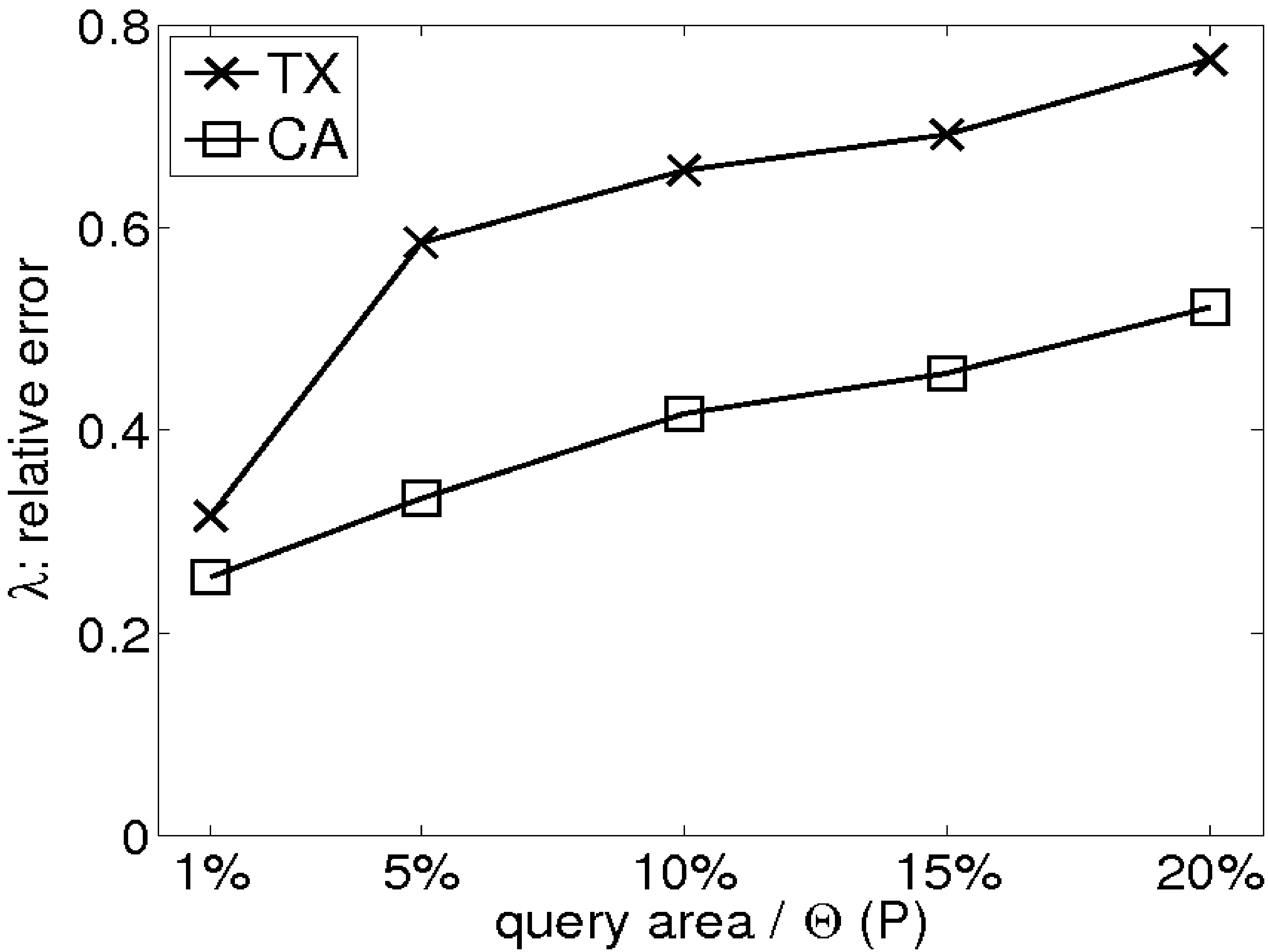


CA data set

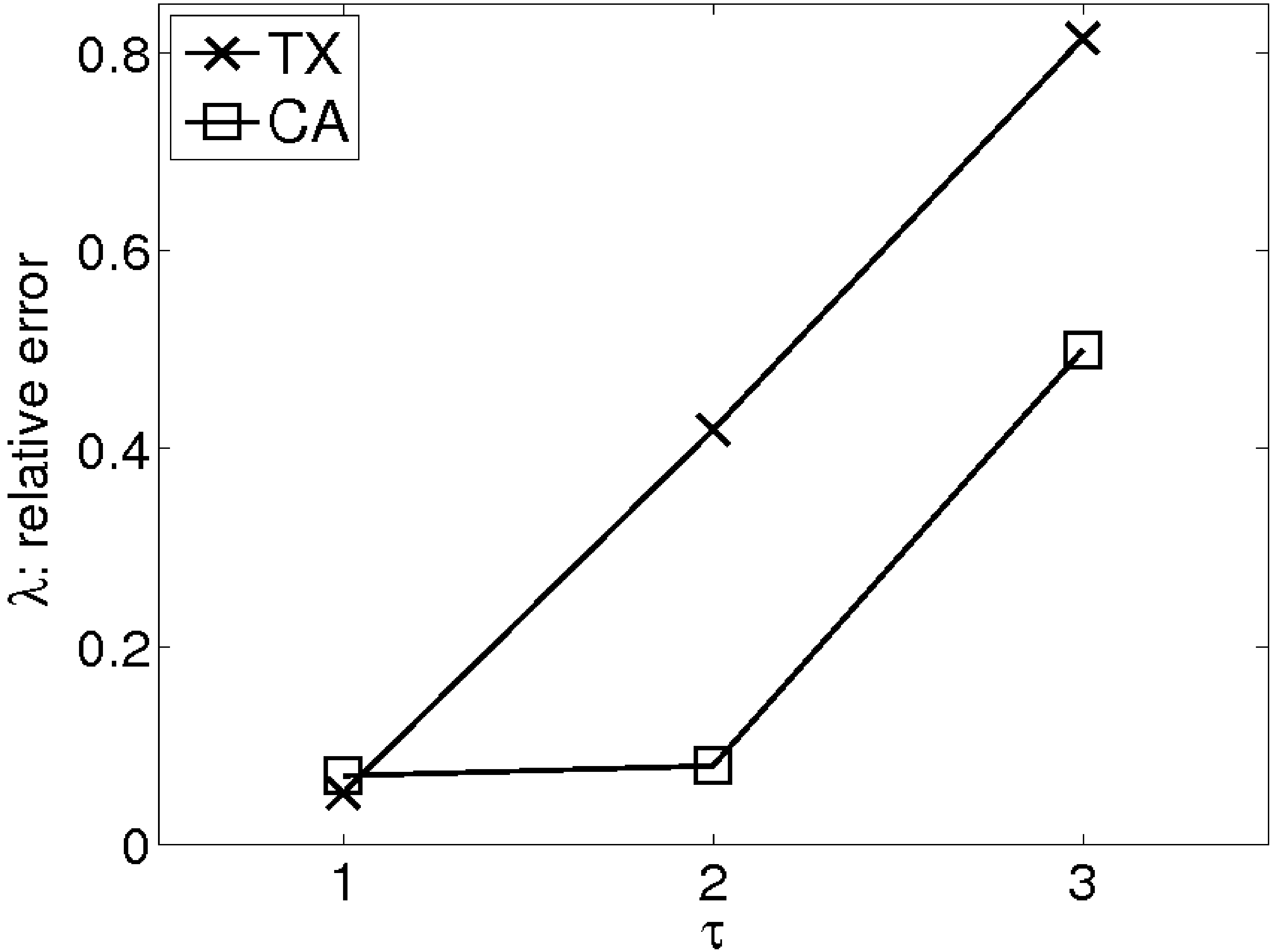
Selectivity estimation: relative errors



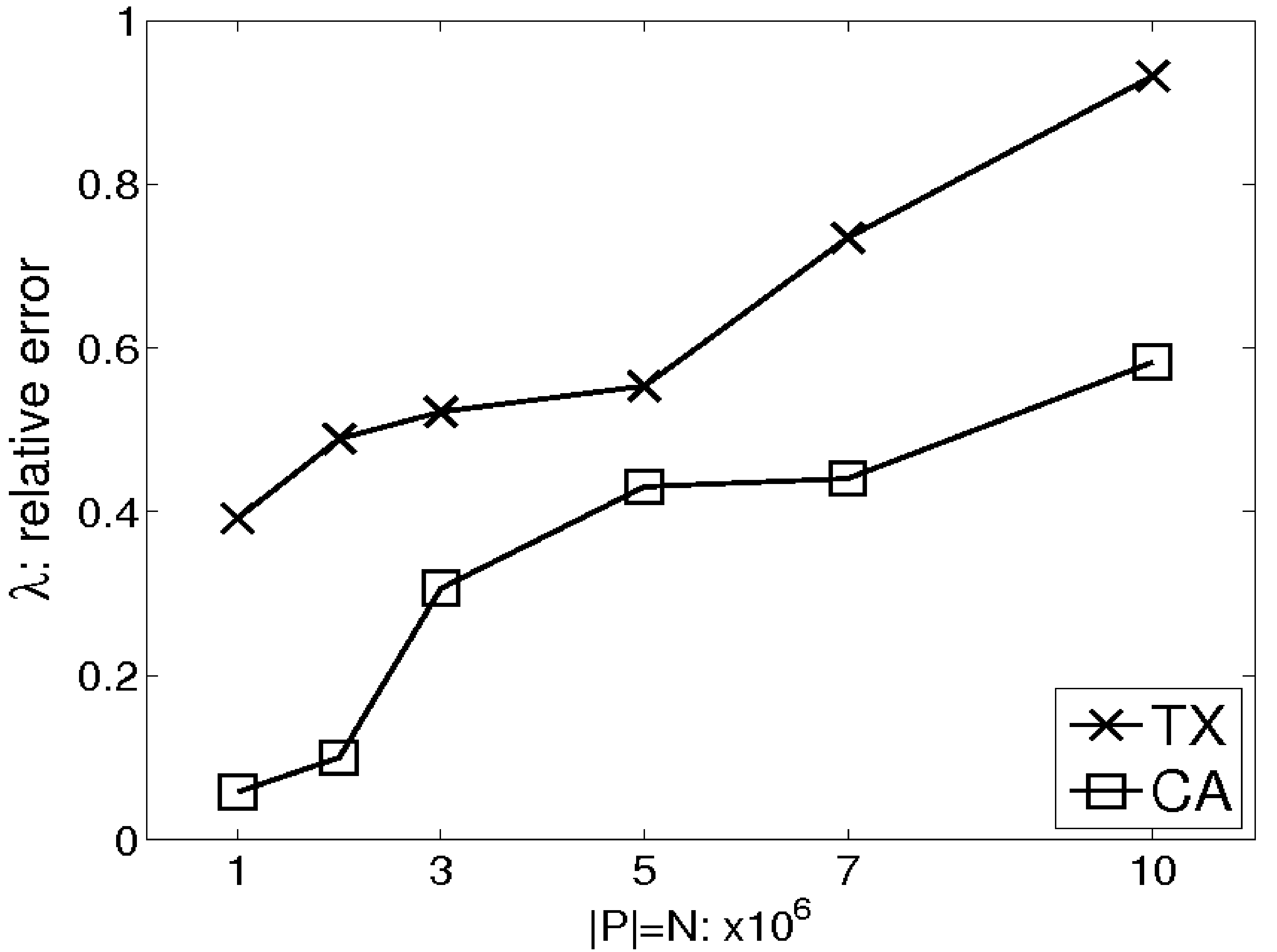
Selectivity estimation: relative errors



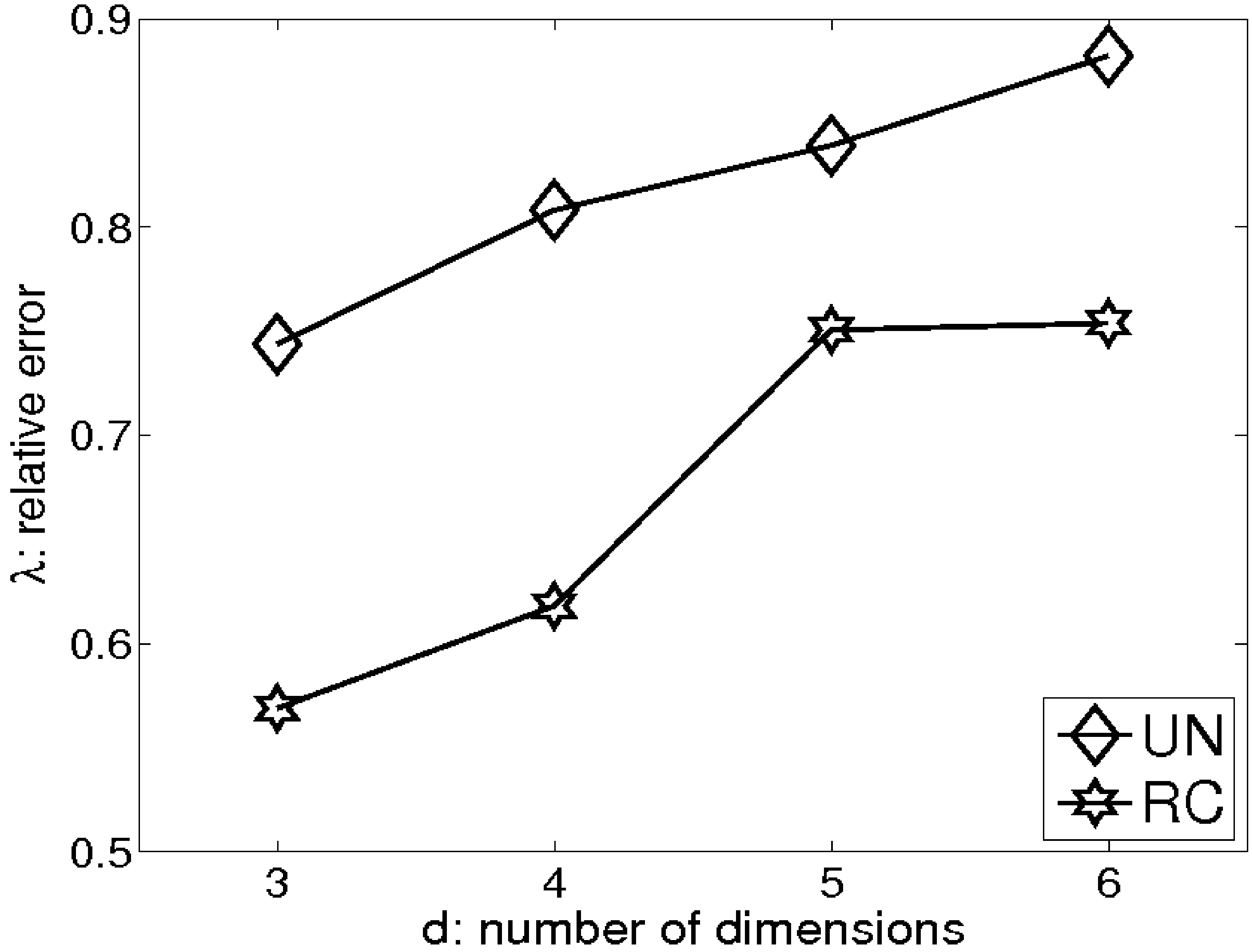
Selectivity estimation: relative errors



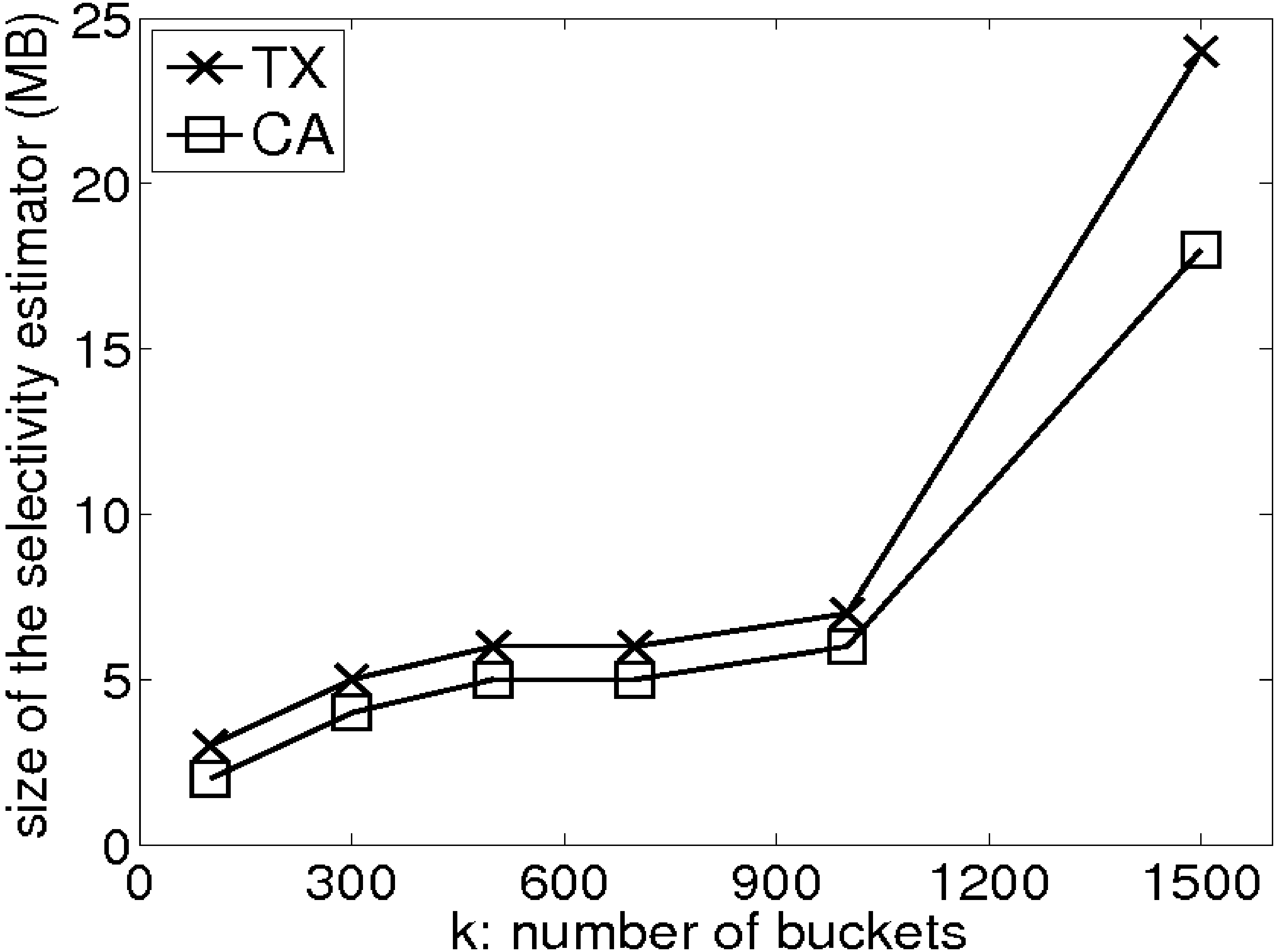
Selectivity estimation: relative errors



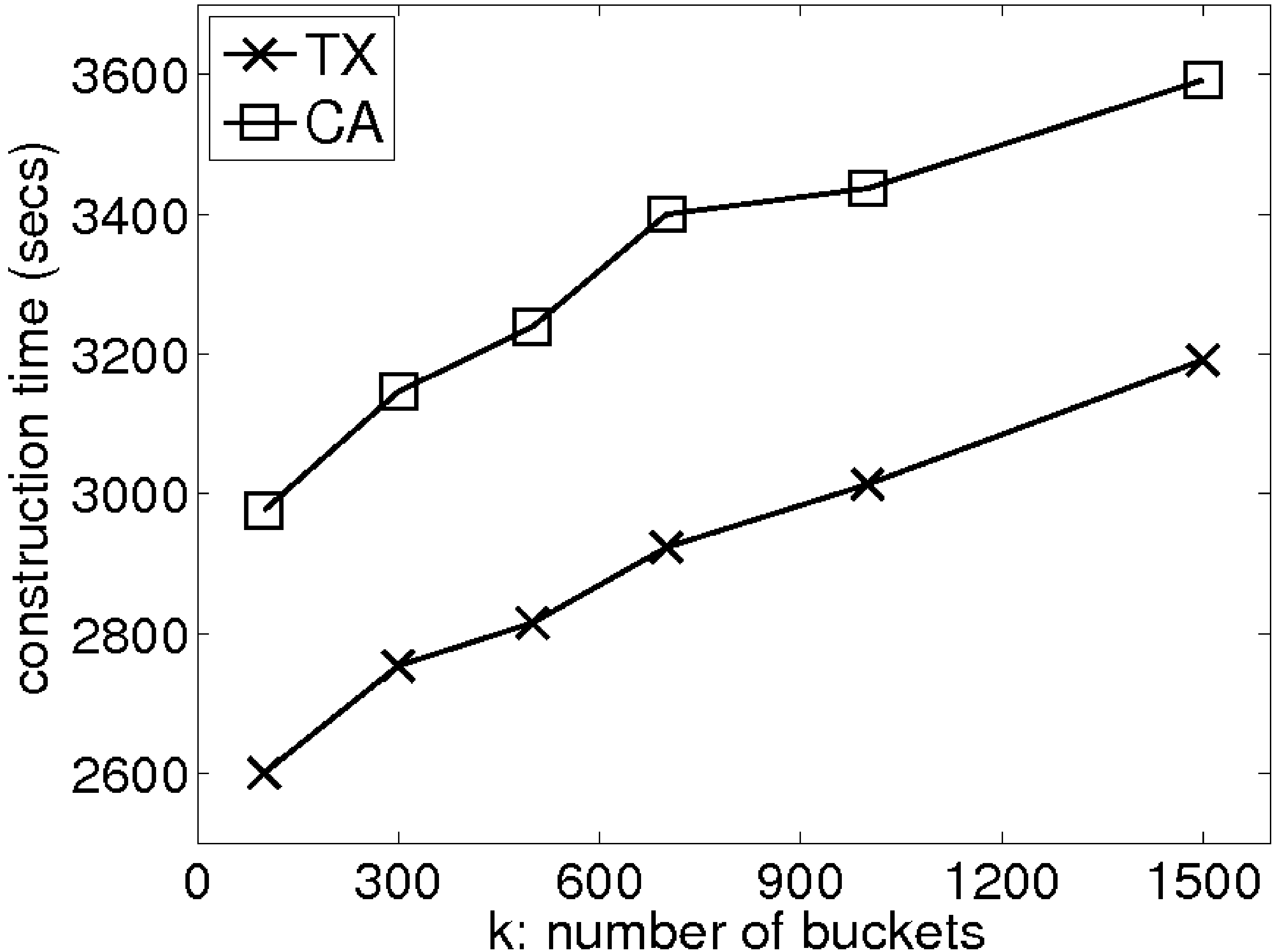
Selectivity estimation: relative errors



Selectivity estimation: cost of the adaptive estimator



Selectivity estimation: cost of the adaptive estimator



Conclusions

- We designed MHR-tree for spatial approximate string queries.

Conclusions

- We designed MHR-tree for spatial approximate string queries.
- We designed novel selectivity estimator for SAS range queries, which take into account both the spatial and string distributions.

Conclusions

- We designed MHR-tree for spatial approximate string queries.
- We designed novel selectivity estimator for SAS range queries, which take into account both the spatial and string distributions.
- Future work includes examining spatial approximate sub-string queries, and using the KMV synopsis to improve the performance.

The End

$$THANK \quad YOU$$

Q and A