K Nearest Neighbor Queries and KNN-Joins in Large Relational Databases (Almost) for Free

Bin Yao, Feifei Li, Piyush Kumar

Florida State University

Introduction

- KNN queries and KNN-Joins:
  spatial databases, pattern recognition, DNA sequencing.

Introduction

- KNN queries and KNN-Joins:
  spatial databases, pattern recognition, DNA sequencing.

- Our goal: design *relational algorithms* for KNN and KNN-Joins.

Introduction

- KNN queries and KNN-Joins:
  spatial databases, pattern recognition, DNA sequencing.

- Our goal: design *relational algorithms* for KNN and KNN-Joins.
  - Readily applied on relational databases without updating the engine.
  - Augmented with ad-hoc query conditions and optimized by the query optimizer.
  - Do it in SQL!

Challenge and benefit in designing relational algorithms

- The main challenge:
  A query optimizer cannot optimize user-defined functions (UDF).

Challenge and benefit in designing relational algorithms

- The main challenge:
  A query optimizer cannot optimize user-defined functions (UDF).

  SELECT TOP k * FROM Address A, Restaurant R
  WHERE R.Type='Italian' AND R.Wine='French'
  ORDER BY Euclidean (A.X, A.Y, R.X, R.Y)

Challenge and benefit in designing relational algorithms

- The main challenge:
  A query optimizer cannot optimize user-defined functions (UDF).

  SELECT TOP k * FROM Address A, Restaurant R
  WHERE R.Type='Italian' AND R.Wine='French'
  ORDER BY Euclidean (A.X, A.Y, R.X, R.Y)

Previous work on kNN and kNN-Join

- Exact kNN solution:
  - R-tree for low dimensions
  - iDistance for high dimensions.

Previous work on kNN and kNN-Join

- Exact kNN solution:
  - R-tree for low dimensions
  - iDistance for high dimensions.
- Approximate kNN solution:
  - Balanced box decomposition tree
  - Locality sensitive hashing
  - Medrank
  - LSB-tree

Previous work on kNN and kNN-Join

- Exact kNN solution:
  - R-tree for low dimensions
  - iDistance for high dimensions.
- Approximate kNN solution:
  - Balanced box decomposition tree
  - Locality sensitive hashing
  - Medrank
  - LSB-tree
- kNN-Join solution:
  - the iJoin algorithm
  - the Gorder algorithm

Problem formulation

- Data set P stored in table R_P: $\{pid, Y_1, \cdots, Y_d, A_1, \cdots, A_h\}$.
  Query set Q stored in table R_Q: $\{qid, X_1, \cdots, X_d, B_1, \cdots, B_g\}$.

Problem formulation

- Data set P stored in table R_P: $\{pid, Y_1, \cdots, Y_d, A_1, \cdots, A_h\}$.
  Query set Q stored in table R_Q: $\{qid, X_1, \cdots, X_d, B_1, \cdots, B_g\}$.
- KNN queries: let $A = kNN(q, \mathsf{R}_P)$,

$$(A \subseteq \mathsf{R}_P) \wedge (|A| = k) \wedge (\forall a \in A, \forall r \in \mathsf{R}_P - A, |a, q| \leq |r, q|).$$

Problem formulation

- Data set P stored in table R_P: $\{pid, Y_1, \cdots, Y_d, A_1, \cdots, A_h\}$.
  Query set Q stored in table R_Q: $\{qid, X_1, \cdots, X_d, B_1, \cdots, B_g\}$.
- KNN queries: let $A = kNN(q, \mathsf{R}_P)$,

$$(A \subseteq \mathsf{R}_P) \wedge (|A| = k) \wedge (\forall a \in A, \forall r \in \mathsf{R}_P - A, |a, q| \leq |r, q|).$$

- KNN-Join:
  for $\forall s \in Q$, produce k pairs (s, r), for $\forall r \in k$NN(s, R_p).

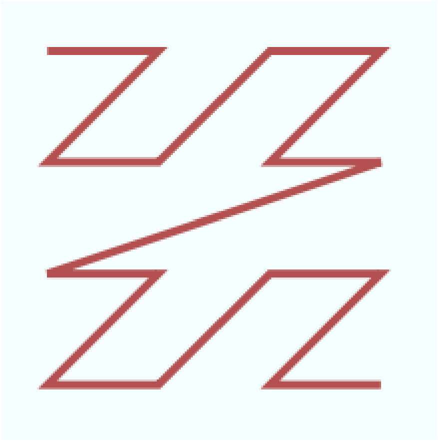
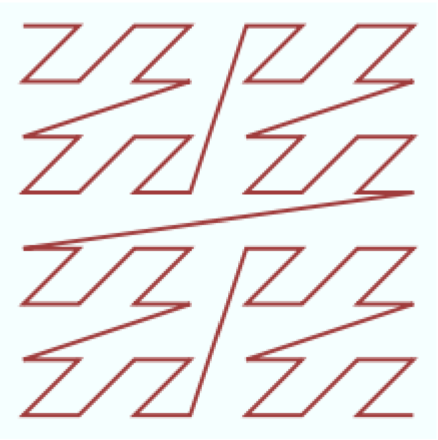
Problem formulation

- Data set P stored in table R_P: $\{pid, Y_1, \cdots, Y_d, A_1, \cdots, A_h\}$.
  Query set Q stored in table R_Q: $\{qid, X_1, \cdots, X_d, B_1, \cdots, B_g\}$.
- KNN queries: let $A = kNN(q, \mathsf{R}_P)$,

  $$(A \subseteq \mathsf{R}_P) \wedge (|A| = k) \wedge (\forall a \in A, \forall r \in \mathsf{R}_P - A, |a, q| \leq |r, q|).$$
- KNN-Join:
  for $\forall s \in Q$, produce k pairs (s, r), for $\forall r \in k$NN(s, R_p).
- Approximate k nearest neighbors:
  Suppose q's kth nn from P is p^* and $r^* = |q, p^*|$,
  p be the kth NN of q for some kNN algorithm A and $r^p = |q, p|$,
  $(p, r^p) \in \mathbb{R}^d \times \mathbb{R}$ is $(1 + \epsilon)$-approximate solution of kNN if
  $r^* \leq r^p \leq (1 + \epsilon) r^*$.

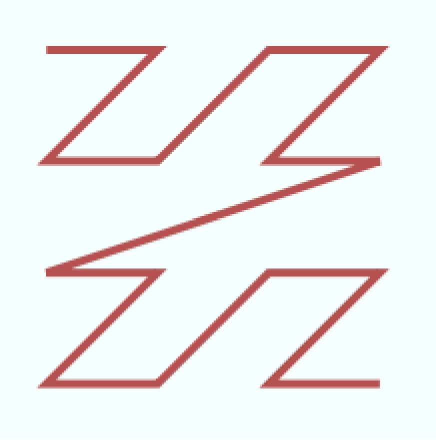
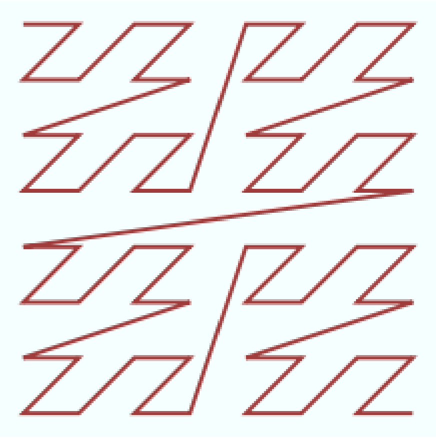
Z-value and Z-order curve

- z-value of a point:
  For point $(2, 6)$, binary representation is $(010, 110)$, z-value is $011100 = 28$.

Z-value and Z-order curve

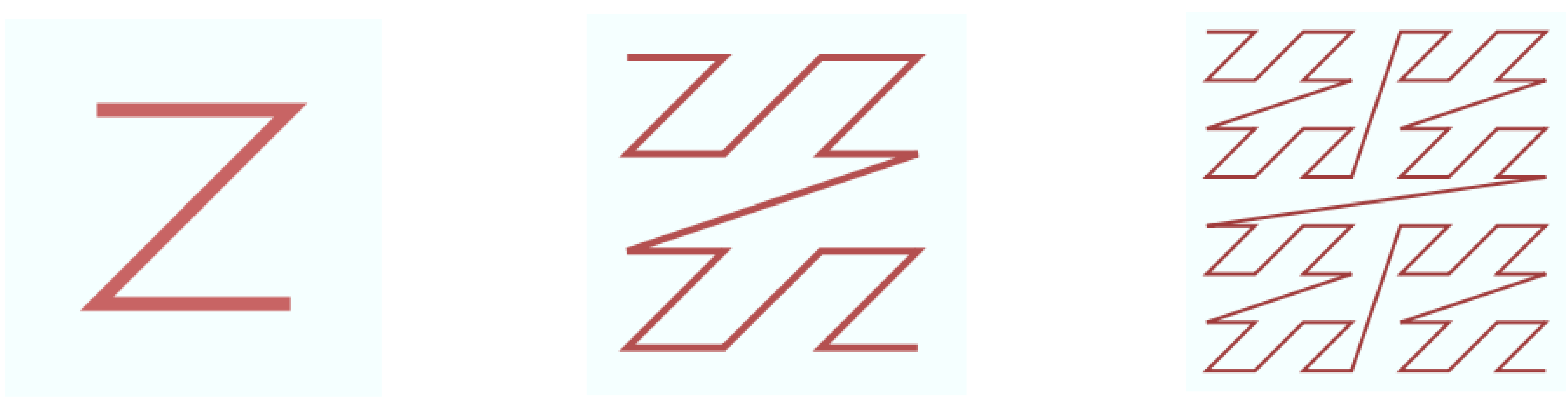
- z-value of a point:
  For point $(2, 6)$, binary representation is $(010, 110)$, z-value is $011100 = 28$.

- A well-known approach:

Z-value and Z-order curve

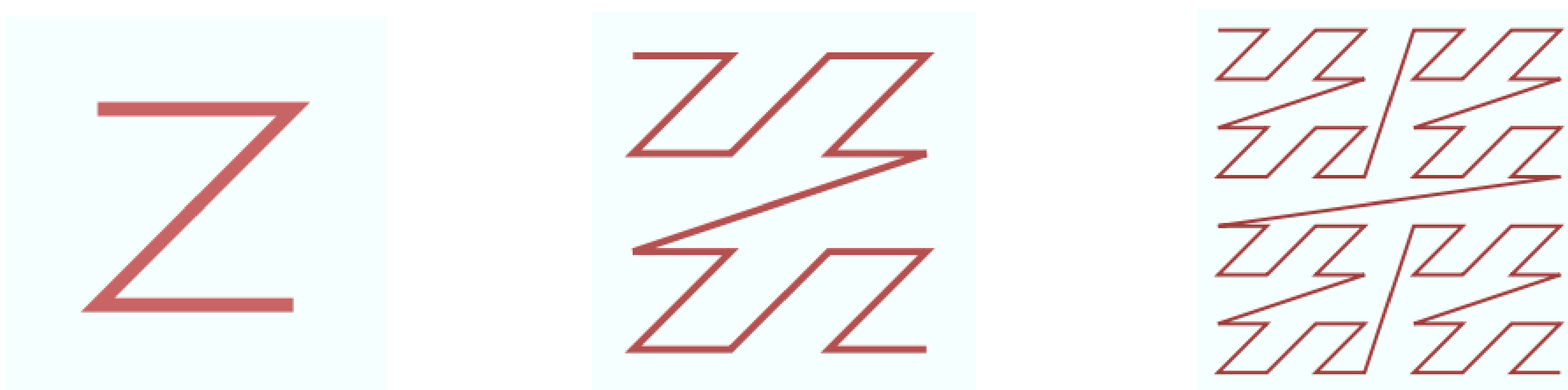
- z-value of a point:
  For point $(2, 6)$, binary representation is $(010, 110)$, z-value is $011100 = 28$.



- A well-known approach:
  - Map points in a multi-dimensional space into one dimension by using z-values.

Z-value and Z-order curve

- z-value of a point:
  For point $(2,6)$, binary representation is $(010, 110)$, z-value is $011100 = 28$.



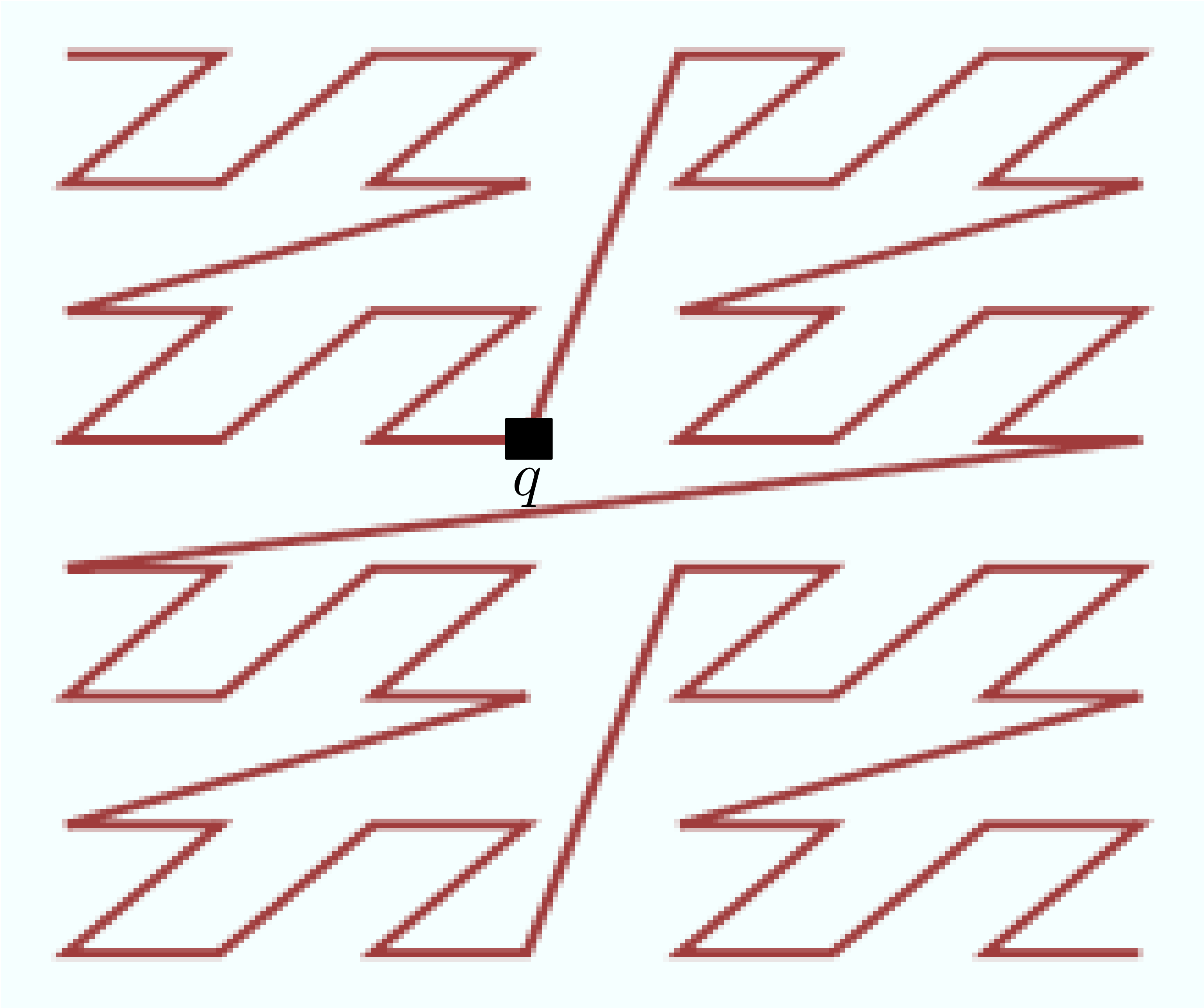
- A well-known approach:
  - Map points in a multi-dimensional space into one dimension by using z-values.
  - Translate the kNN search into one dimensional range search on the z-values.

Approximation by random shifts

- Z-values preserve the spatial locality, but not always the case.

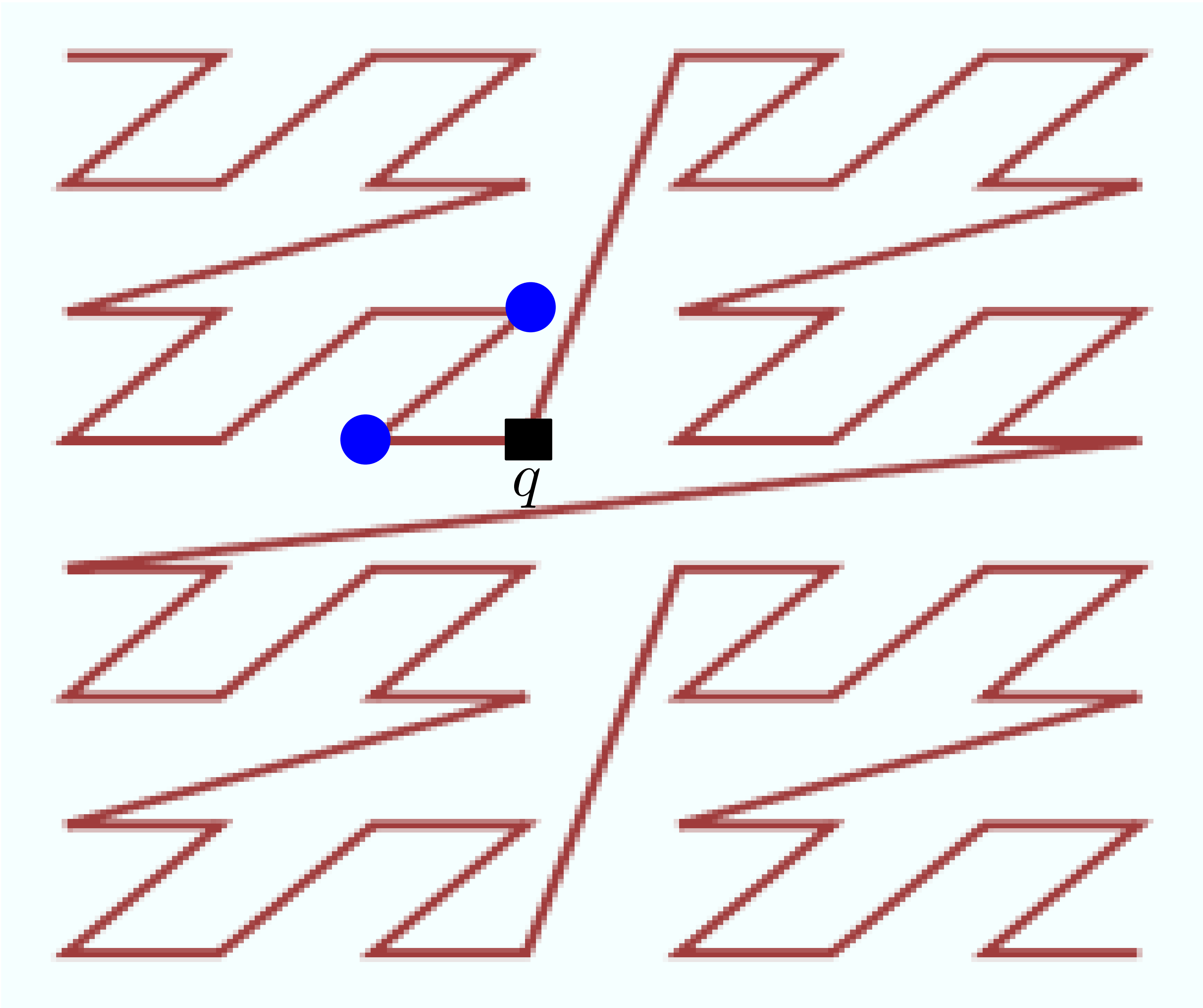
Approximation by random shifts

- Z-values preserve the spatial locality, but not always the case.



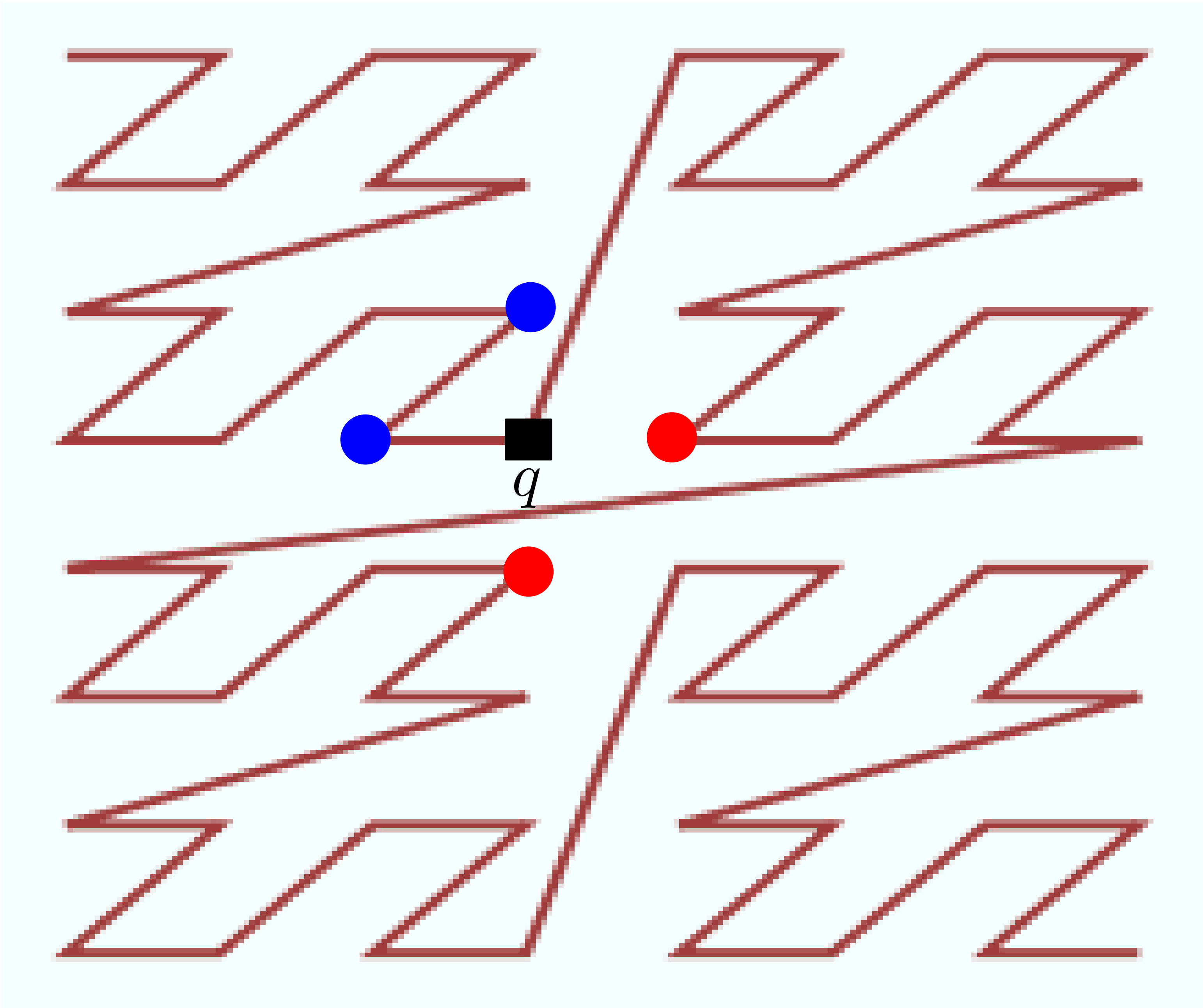
Approximation by random shifts

- Z-values preserve the spatial locality, but not always the case.



Approximation by random shifts

- Z-values preserve the spatial locality, but not always the case.



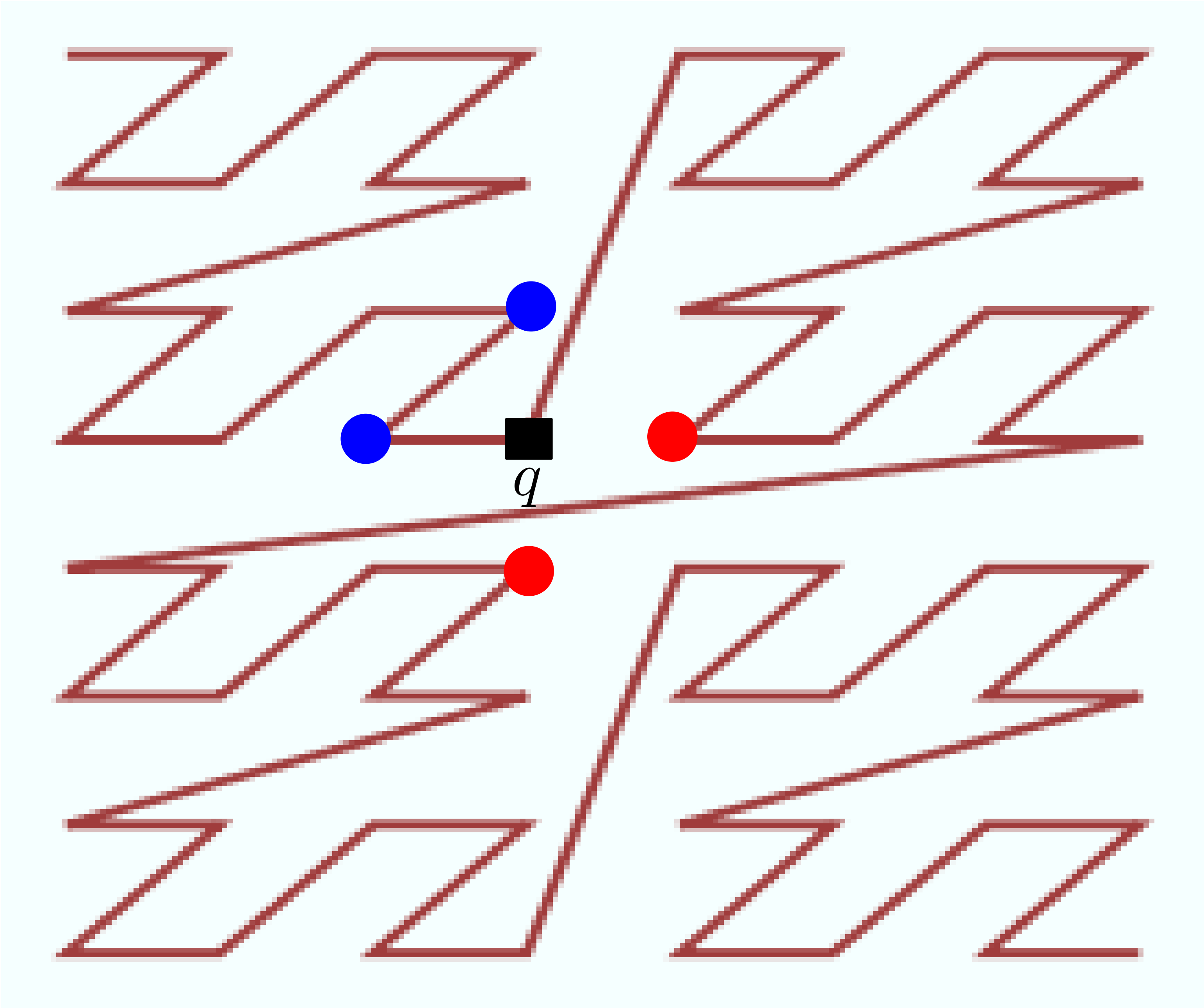
Approximation by random shifts

- Z-values preserve the spatial locality, but not always the case.

- Our idea: produce α randomly shifted copies of the input data set $(P^0, \ldots, P^\alpha)$ and repeat the one dimensional range search ($\gamma = O(k)$ points up and down next to the q) for each copy.

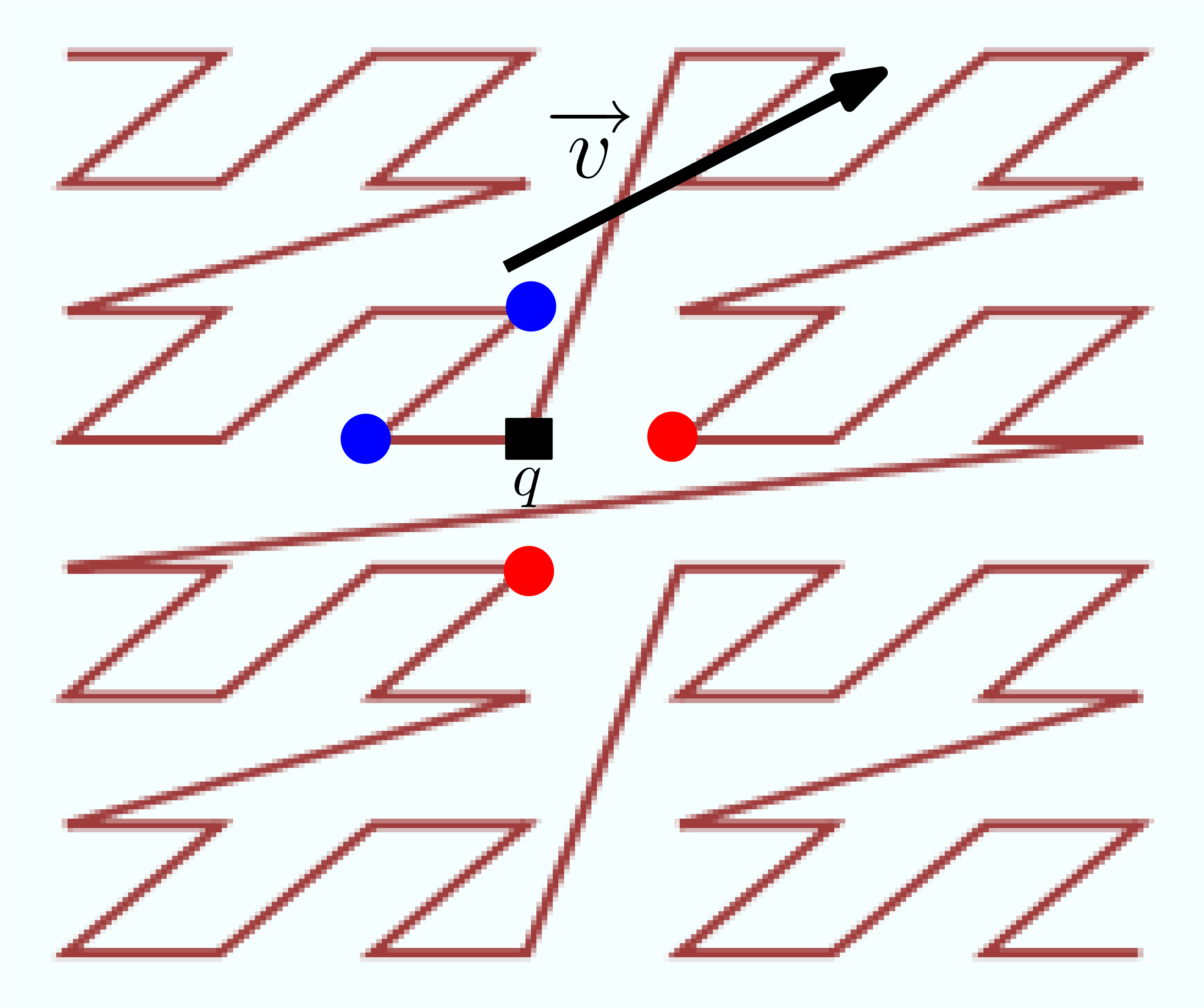
Approximation by random shifts

- Z-values preserve the spatial locality, but not always the case.



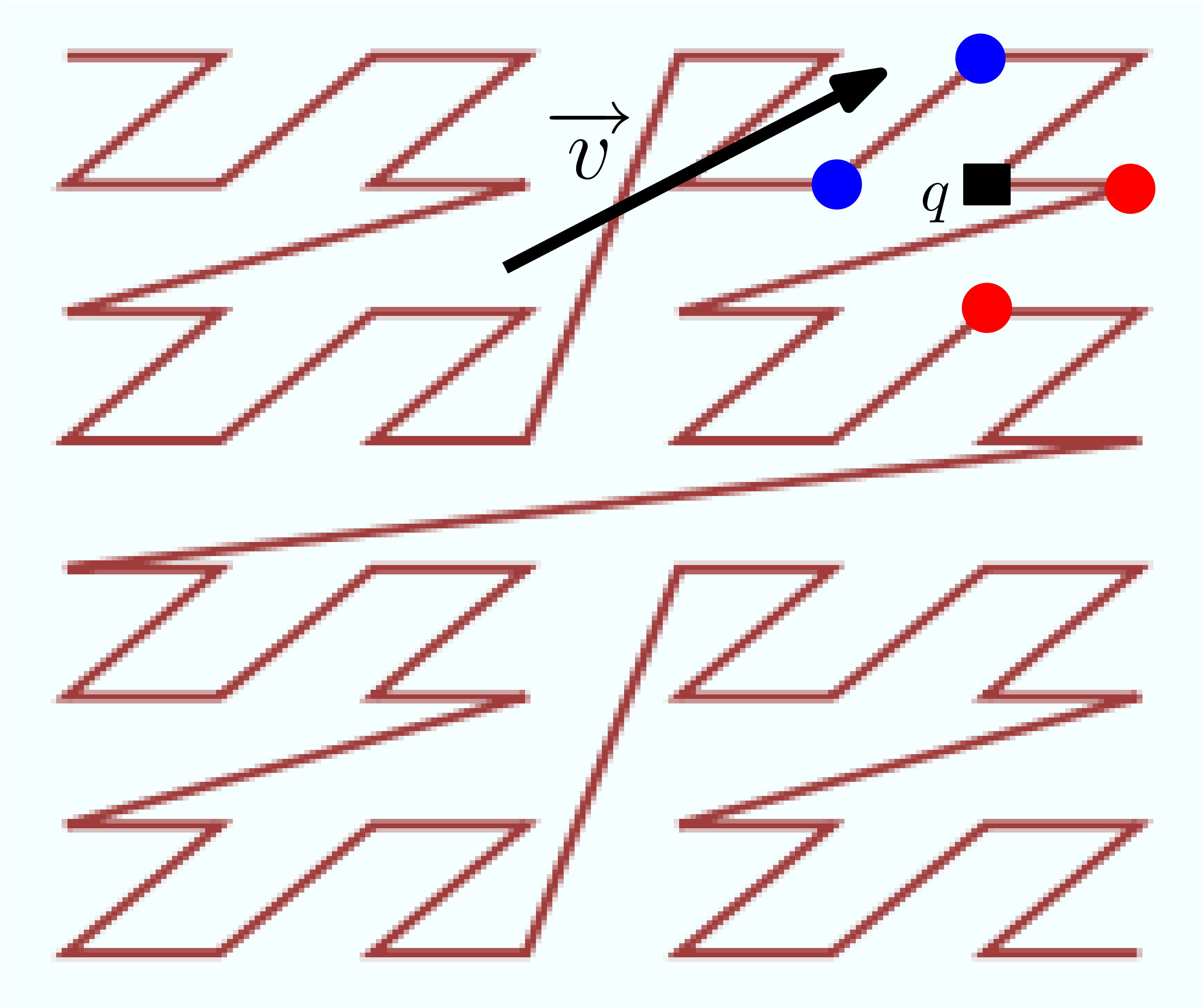
Approximation by random shifts

- Z-values preserve the spatial locality, but not always the case.



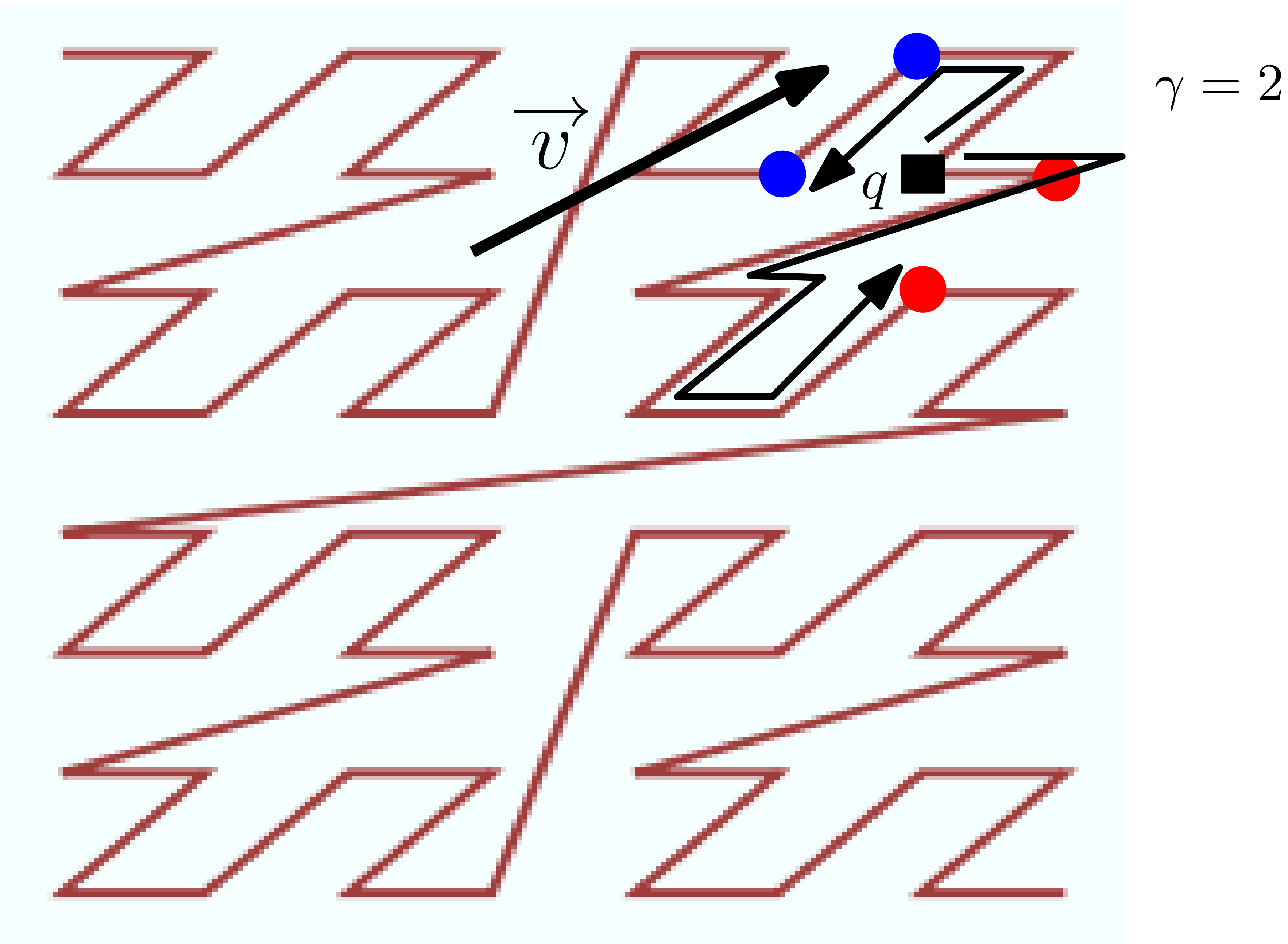
Approximation by random shifts

- Z-values preserve the spatial locality, but not always the case.



Approximation by random shifts

- Z-values preserve the spatial locality, but not always the case.



Approximation by random shifts

- Z-values preserve the spatial locality, but not always the case.
- Our idea: produce α randomly shifted copies of the input data set $(P^0, \ldots, P^\alpha)$ and repeat the one dimensional range search ($\gamma = O(k)$ points up and down next to the q) for each copy.
- Retrieve the kNN from the unioned candidates of the α copies.

Approximation by random shifts

- Z-values preserve the spatial locality, but not always the case.
- Our idea: produce α randomly shifted copies of the input data set $(P^0, \ldots, P^\alpha)$ and repeat the one dimensional range search ($\gamma = O(k)$ points up and down next to the q) for each copy.
- Retrieve the kNN from the unioned candidates of the α copies.
- Theorem 1:
  Using $\alpha = O(1)$ and $\gamma = O(k)$, z$^{\chi}$-kNN guarantees an expected constant factor approximate kNN result with $O(\log_f \frac{N}{B} + k/B)$ number of page accesses (clustered index on z-values).

Approximation algorithm

z$^\chi$-kNN (point q, point sets $\{P^0, \ldots, P^\alpha\}$)

Candidates $C = \emptyset$;
For $i = 0, \ldots, \alpha$ $\{$
Find z_p^i as the successor of z_{q+v_i} in P^i;
Let C^i be γ points up and down next to z_p^i in P^i;
For each point p in C^i, let $p = p - v_i$;
$C = C \bigcup C^i$;
$\}$
Let $A^\chi = k$NN(q, C) and output A^χ.

Approximation algorithm

z$^{\chi}$-kNN (point q, point sets $\{P^0, \ldots, P^\alpha\}$)

Candidates $C = \emptyset$;

For $i = 0, \ldots, \alpha$ $\{$

$\boxed{\text{Find } z_p^i \text{ as the successor of } z_{q+v_i} \text{ in } P^i;}$

Let C^i be γ points up and down next to z_p^i in P^i;

For each point p in C^i, let $p = p - v_i$;

$C = C \bigcup C^i$;

$\}$

Let $A^\chi = k$NN(q, C) and output A^χ.

Approximation algorithm

z$^{\chi}$-kNN (point q, point sets $\{P^0, \ldots, P^\alpha\}$)

Candidates $C = \emptyset$;
For $i = 0, \ldots, \alpha$ $\{$
Find z_p^i as the successor of z_{q+v_i} in P^i;
Let C^i be γ points up and down next to z_p^i in P^i;
For each point p in C^i, let $p = p - v_i$;
$C = C \bigcup C^i$;
$\}$
Let $A^{\chi} = k$NN(q, C) and output A^{χ}.

Approximation algorithm

z$^{\chi}$-kNN (point q, point sets $\{P^0, \ldots, P^{\alpha}\}$)

Candidates $C = \emptyset$;
For $i = 0, \ldots, \alpha$ $\{$
    Find z_p^i as the successor of z_{q+v_i} in P^i;
    Let C^i be γ points up and down next to z_p^i in P^i;
    For each point p in C^i, let $p = p - v_i$;
    $C = C \bigcup C^i$;
$\}$
Let $A^{\chi} = k$NN(q, C) and output A^{χ}.

Approximation algorithm

z$^{\chi}$-kNN (point q, point sets $\{P^0, \ldots, P^{\alpha}\}$)

Candidates $C = \emptyset$;
For $i = 0, \ldots, \alpha$ {
  Find z_p^i as the successor of z_{q+v_i} in P^i;
  Let C^i be γ points up and down next to z_p^i in P^i;
  For each point p in C^i, let $p = p - v_i$;
  $C = C \bigcup C^i$;
}
$\boxed{\text{Let } A^{\chi} = k\text{NN}(q, C) \text{ and output } A^{\chi}.}$

SQL statement for approximation algorithm

```
SELECT TOP k * FROM
        (SELECT TOP γ+1 * FROM R_P,
                (SELECT TOP 1 zval FROM R_P
                 WHERE R_P.zval ≥ q.zval
                 ORDER BY R_P.zval ASC ) AS T
         WHERE R_P.zval≥T.zval
         ORDER BY R_P.zval ASC
                 UNION
         SELECT TOP γ * FROM R_P
         WHERE R_P.zval < T.zval
         ORDER BY R_P.zval DESC ) AS C
ORDER BY Euclidean(q.X_1,q.X_2,C.Y_1,C.Y_2)
```

SQL statement for approximation algorithm

SELECT TOP k * FROM
(SELECT TOP $\gamma + 1$ * FROM R$_P$,
(SELECT TOP 1 zval FROM R$_P$
WHERE R$_P$.zval $\geq$ q.zval
ORDER BY R$_P$.zval ASC) AS T
WHERE R$_P$.zval$\geq$T.zval
ORDER BY R$_P$.zval ASC
UNION
SELECT TOP γ * FROM R$_P$
WHERE R$_P$.zval $<$ T.zval
ORDER BY R$_P$.zval DESC) AS C
ORDER BY Euclidean(q.X$_1$,q.X$_2$,C.Y$_1$,C.Y$_2$)

SQL statement for approximation algorithm

```
SELECT TOP k * FROM
        (SELECT TOP γ+1 * FROM R_P,
                (SELECT TOP 1 zval FROM R_P
                 WHERE R_P.zval ≥ q.zval
                 ORDER BY R_P.zval ASC ) AS T
         WHERE R_P.zval≥T.zval
         ORDER BY R_P.zval ASC
                 UNION
         SELECT TOP γ * FROM R_P
         WHERE R_P.zval < T.zval
         ORDER BY R_P.zval DESC ) AS C
ORDER BY Euclidean(q.X_1,q.X_2,C.Y_1,C.Y_2)
```

SQL statement for approximation algorithm

SELECT TOP k * FROM
(SELECT TOP $\gamma + 1$ * FROM R$_P$,
(SELECT TOP 1 zval FROM R$_P$
WHERE R$_P$.zval $\geq$ q.zval
ORDER BY R$_P$.zval ASC) AS T
WHERE R$_P$.zval$\geq$T.zval
ORDER BY R$_P$.zval ASC
UNION
SELECT TOP γ * FROM R$_P$
WHERE R$_P$.zval $<$ T.zval
ORDER BY R$_P$.zval DESC) AS C
ORDER BY Euclidean(q.X$_1$,q.X$_2$,C.Y$_1$,C.Y$_2$)

Exact KNN retrieval: naive solution

- The exact kNN points are enclosed by the approximate k*th nearest neighbor ball*.

Exact KNN retrieval: naive solution

- The exact kNN points are enclosed by the approximate kth *nearest neighbor ball*.

- SELECT TOP k * FROM R$_P$
  WHERE Euclidean(q.X$_1$,q.X$_2$,R$_P$.Y$_1$,R$_P$.Y$_2$)$\leq rad(p, \mathcal{A}^{\mathcal{X}})$
  ORDER BY Euclidean(q.X$_1$,q.X$_2$,R$_P$.Y$_1$,R$_P$.Y$_2$)

Exact KNN retrieval: naive solution

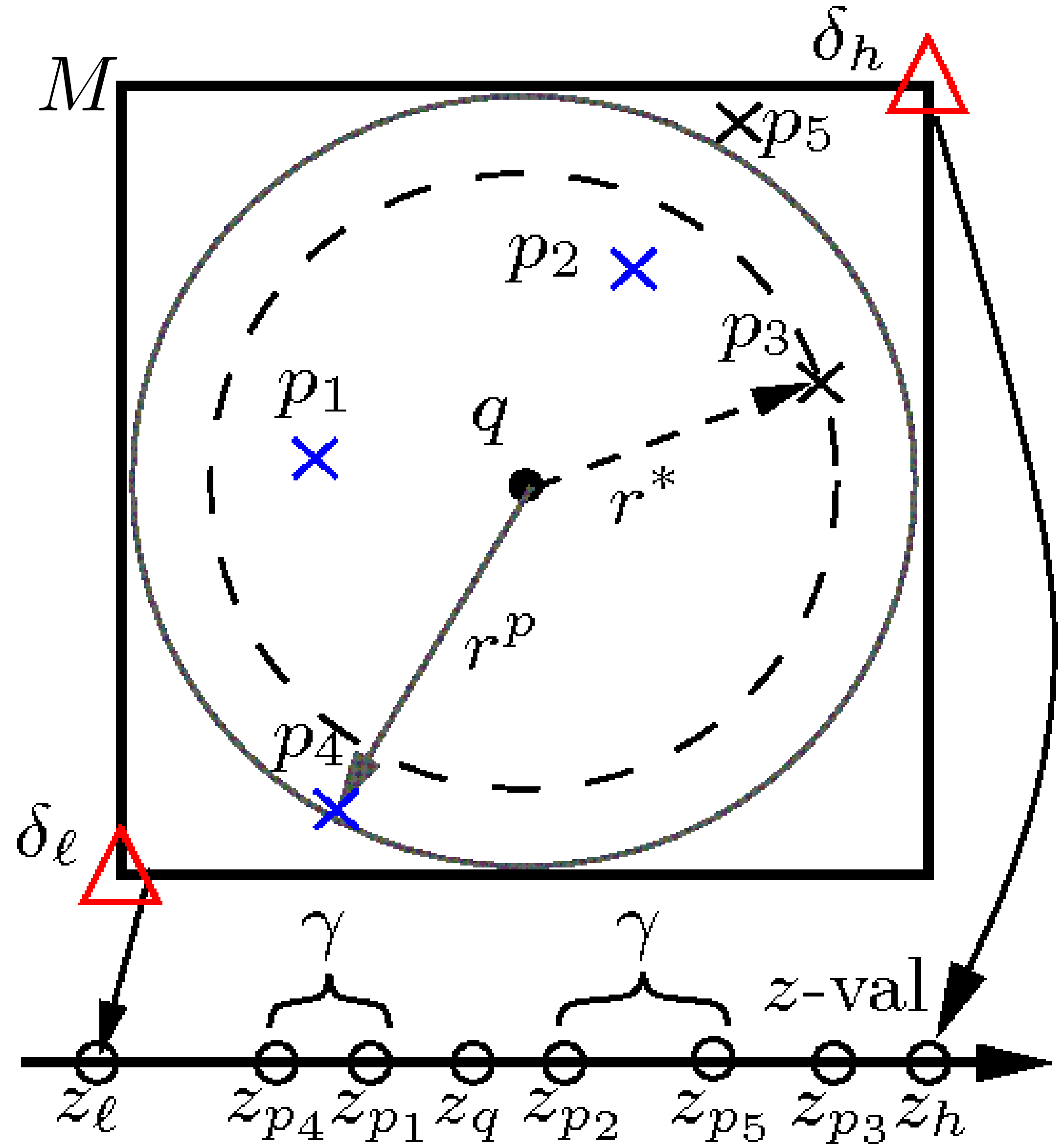
- The exact kNN points are enclosed by the approximate kth *nearest neighbor ball*.

- SELECT TOP k * FROM R$_P$
  WHERE Euclidean(q.X$_1$,q.X$_2$,R$_P$.Y$_1$,R$_P$.Y$_2$)$\leq rad(p, \mathcal{A}^\chi)$
  ORDER BY Euclidean(q.X$_1$,q.X$_2$,R$_P$.Y$_1$,R$_P$.Y$_2$)

- Can we do better?

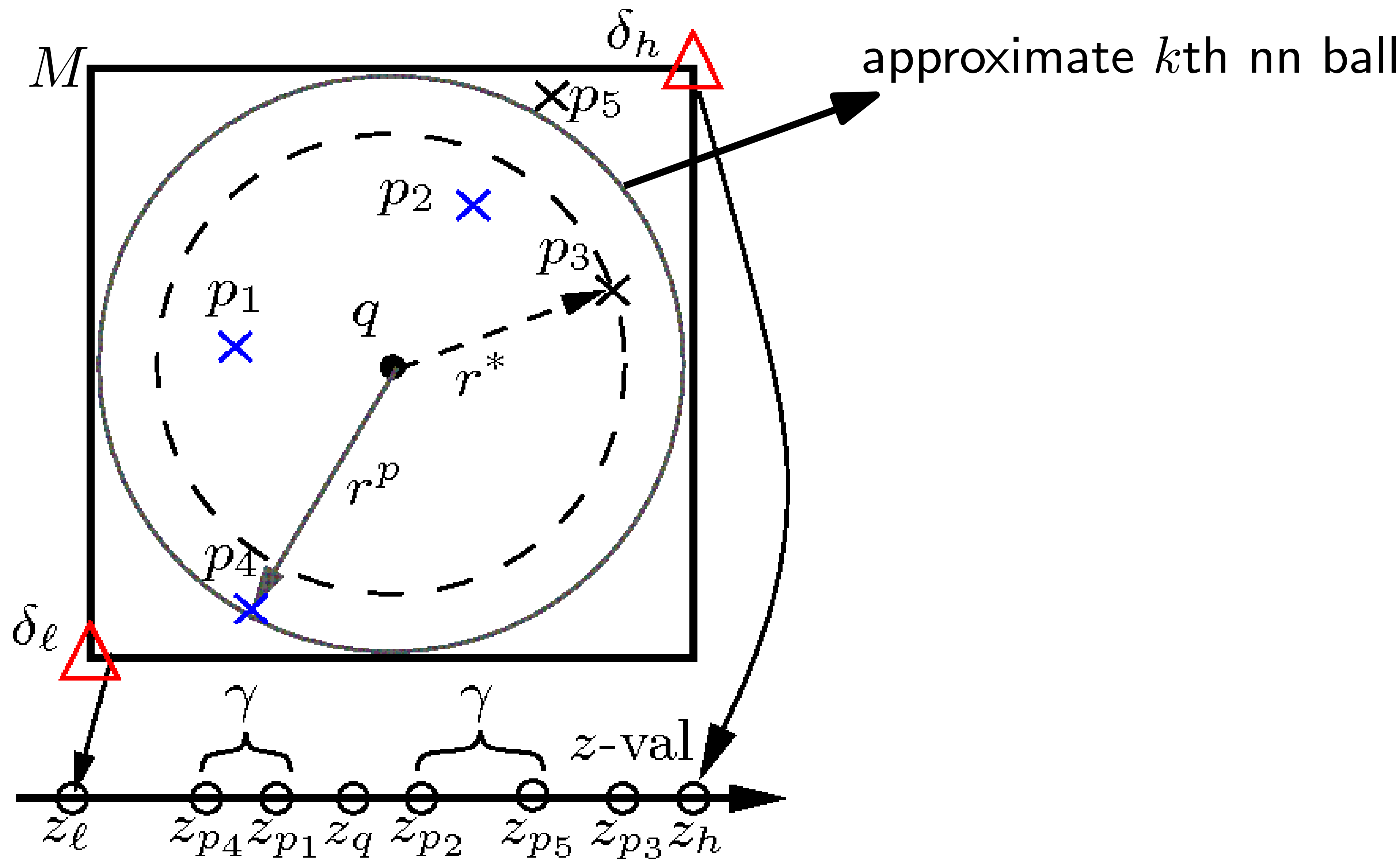
Exact KNN retrieval

$k = 3$



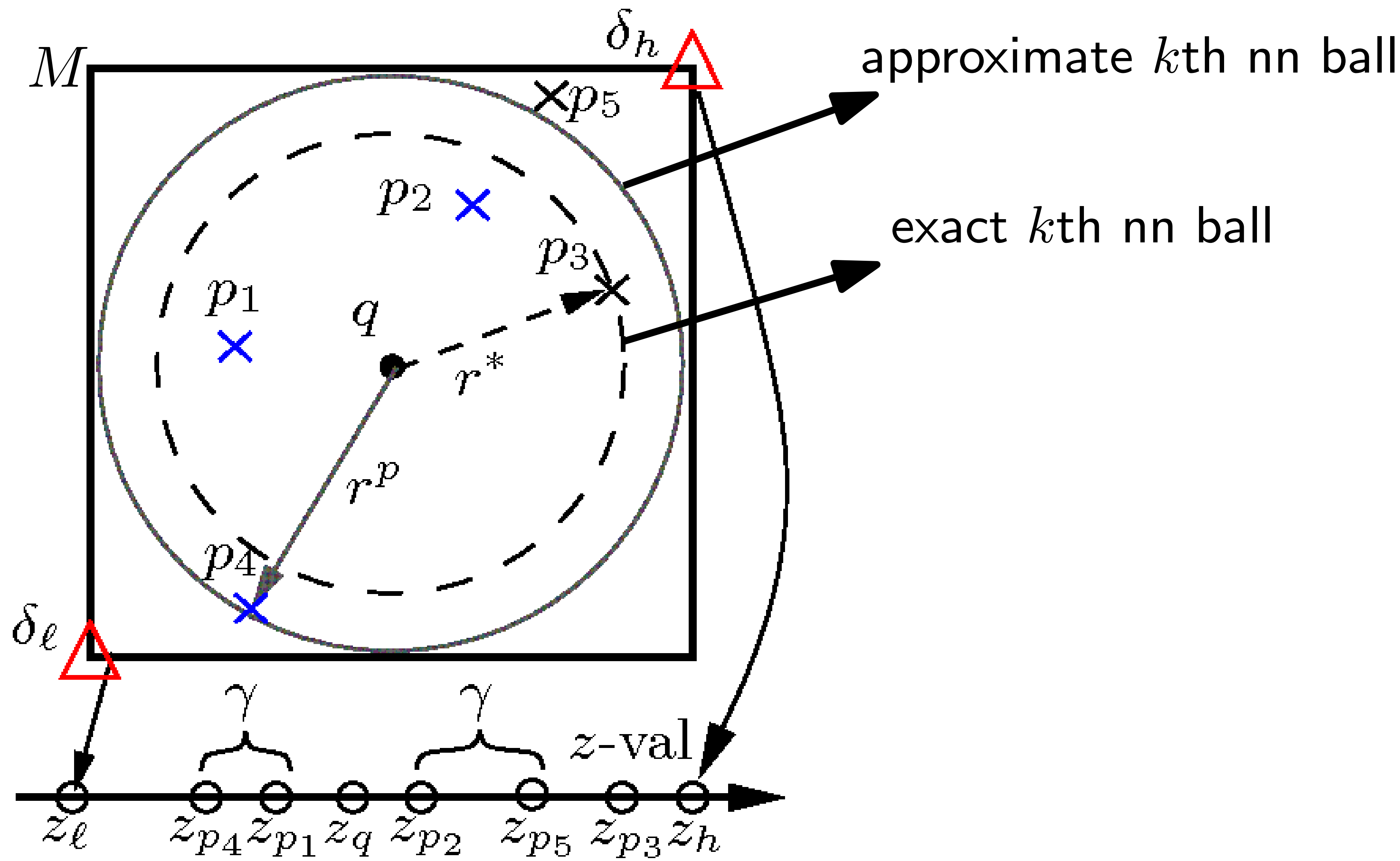
Exact KNN retrieval

$k = 3$



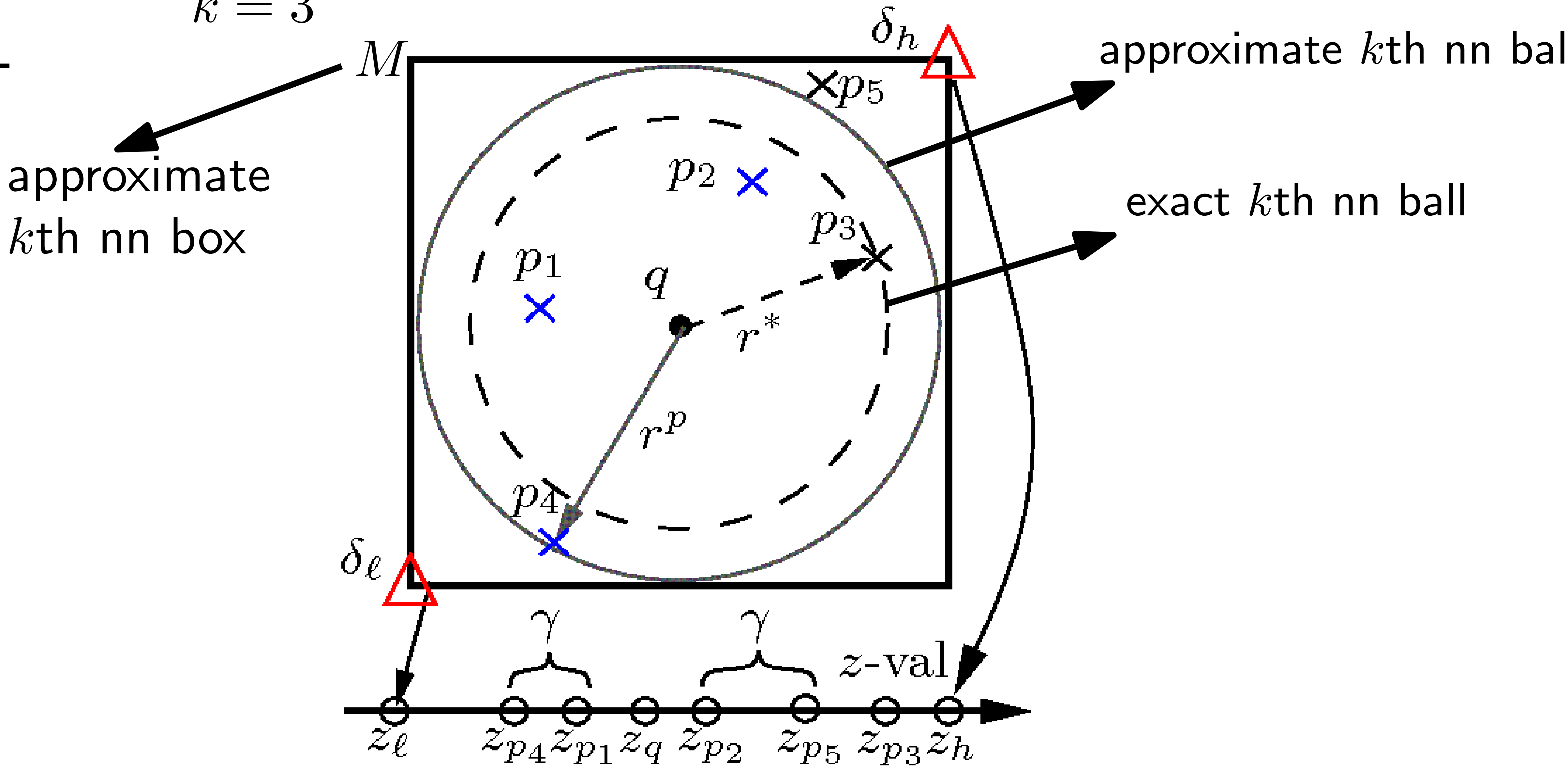
Exact KNN retrieval

$k = 3$

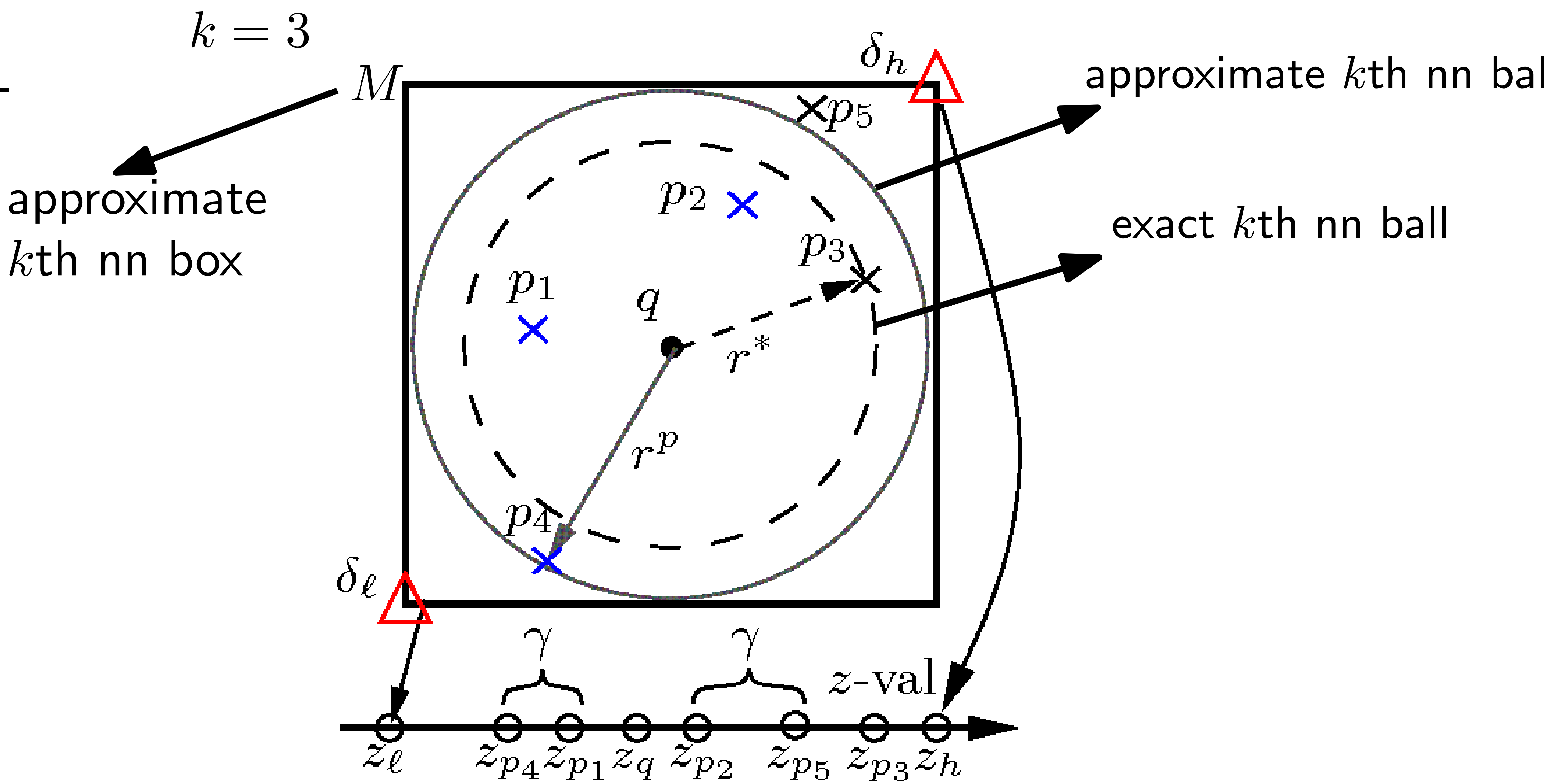


Exact KNN retrieval

$k = 3$



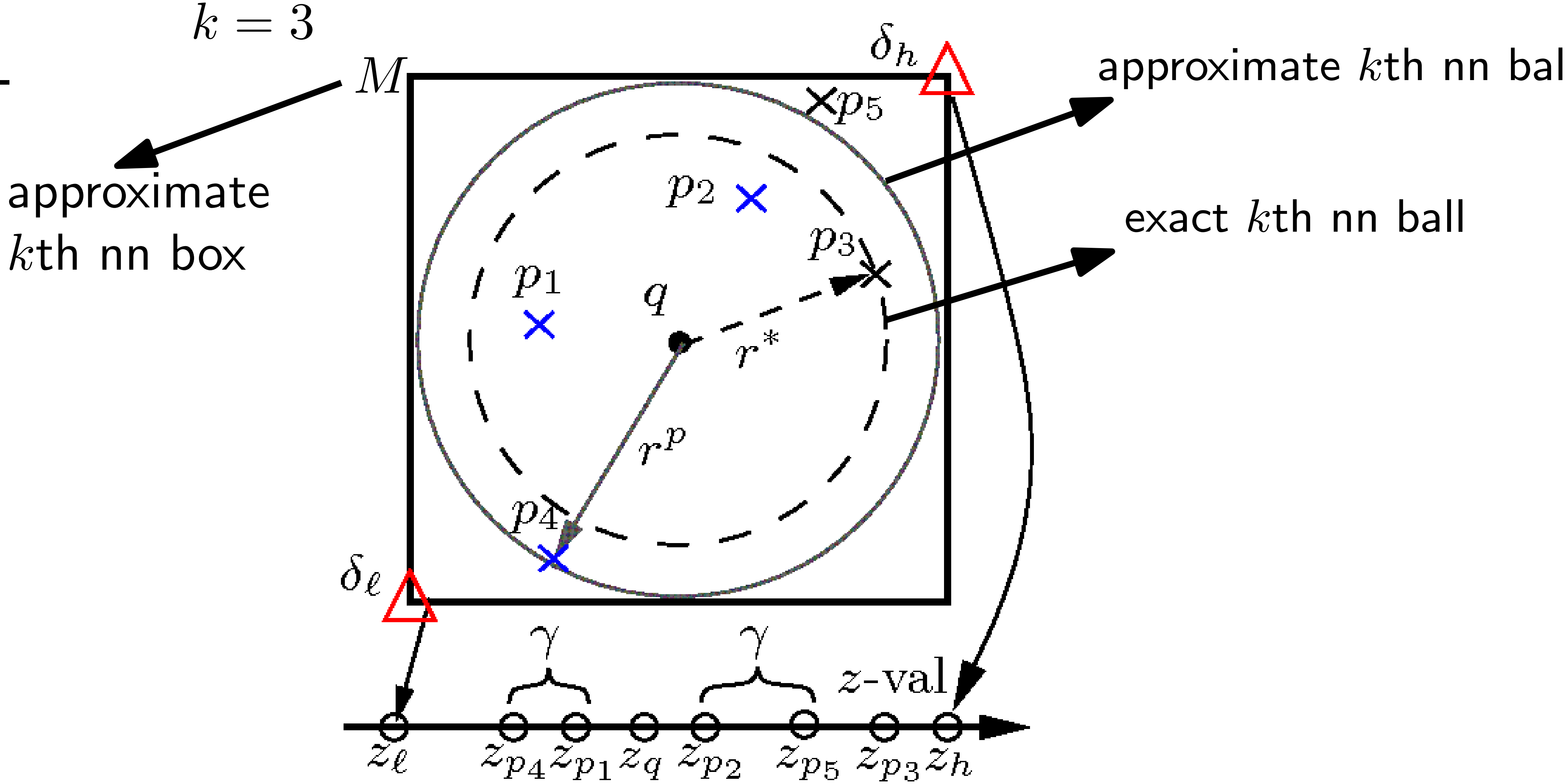
Exact KNN retrieval



Lemma 4: For a rectangular box M and its lower-left and upper-right corner points δ_ℓ, δ_h, $\forall p \in M$, $z_p \in [z_\ell, z_h]$, where z_p stands for the z-value of a point p and z_ℓ, z_h correspond to the z-values of δ_ℓ and δ_h respectively.

Exact KNN retrieval

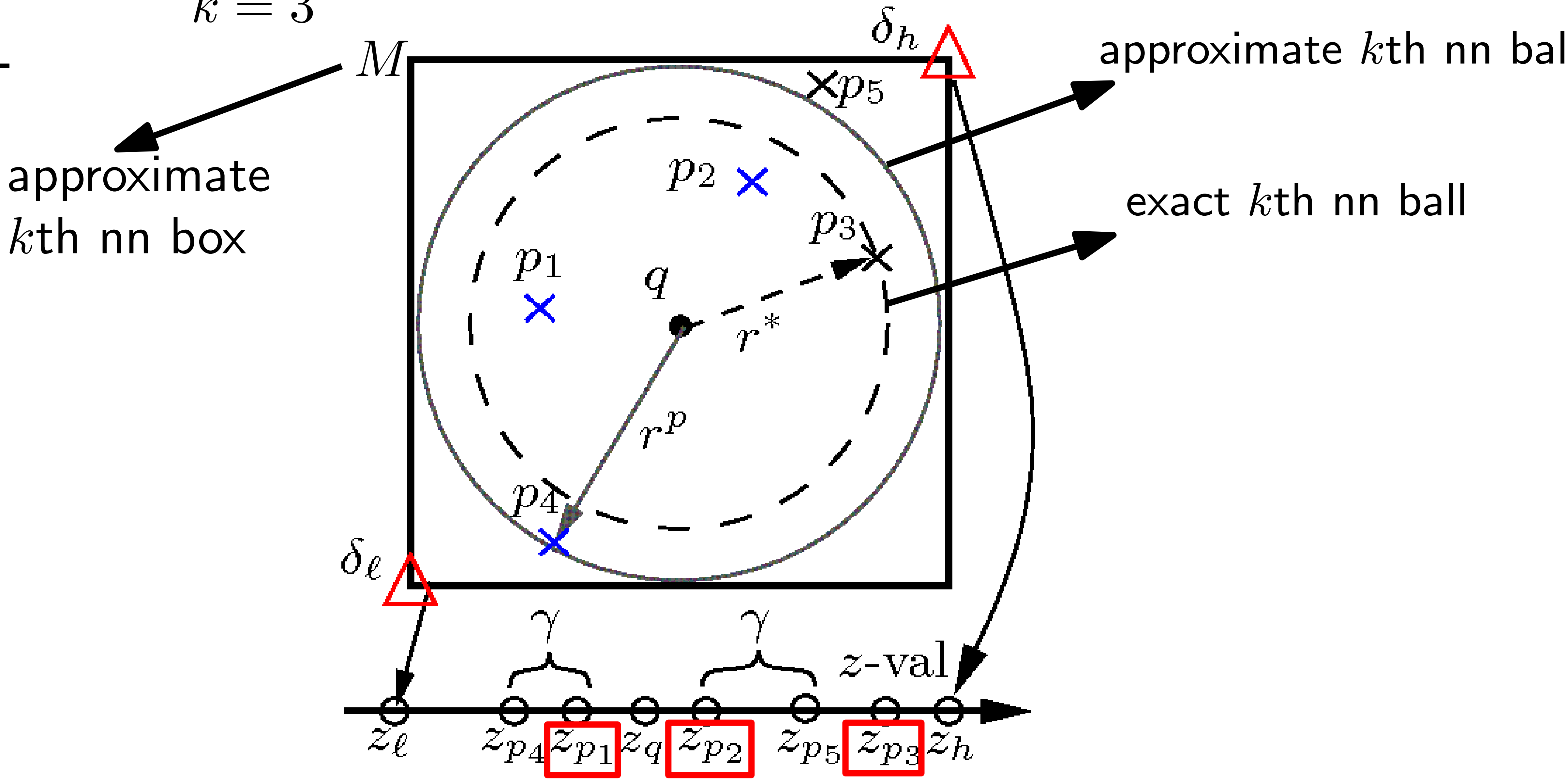
$k = 3$



Corollary 1: Let z_ℓ and z_h be the z-values of δ_ℓ and δ_h points of $M(A^\chi)$. For all $p \in A$, $z_p \in [z_\ell, z_h]$.

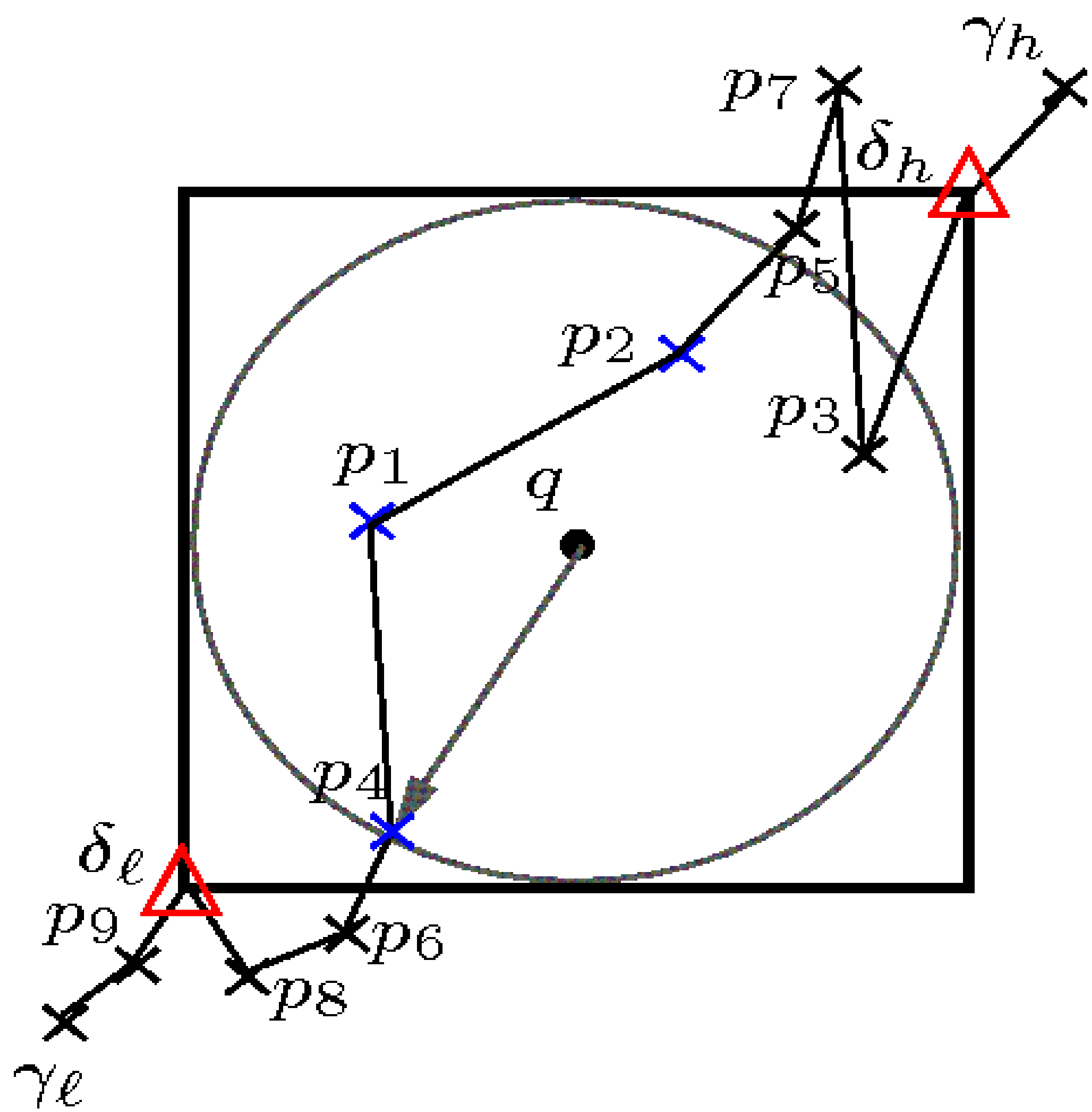
Exact KNN retrieval

$k = 3$



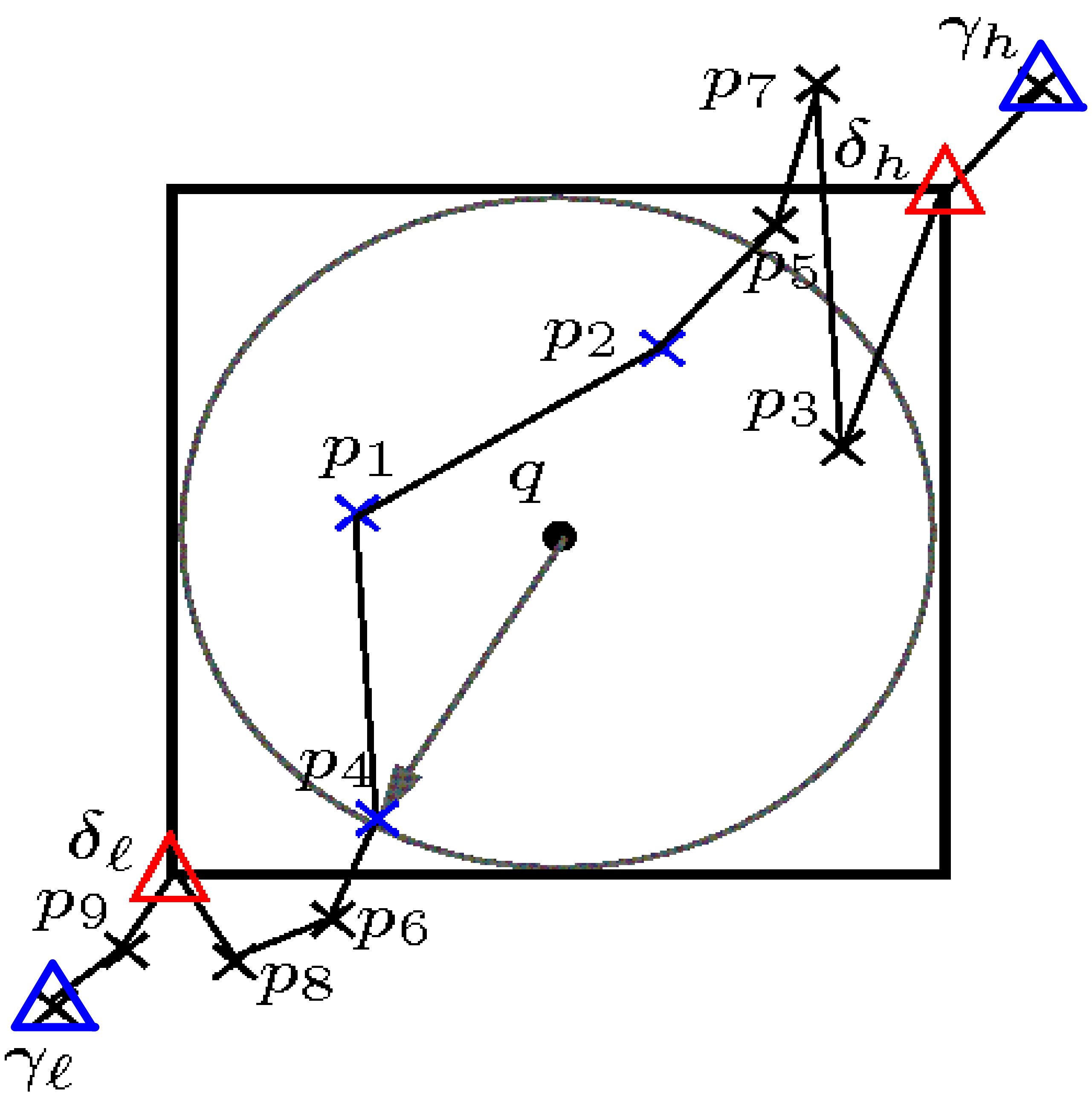
Corollary 1: Let z_ℓ and z_h be the z-values of δ_ℓ and δ_h points of $M(A^\chi)$. For all $p \in A$, $z_p \in [z_\ell, z_h]$.

Exact KNN retrieval



Let γ_ℓ and γ_h denote the left and right γ-th points close to the query point, if $z_{\gamma_\ell} \leq z_\ell$ and $z_{\gamma_h} \geq z_h$ in *at least* one of the α tables, $A^\chi = A$

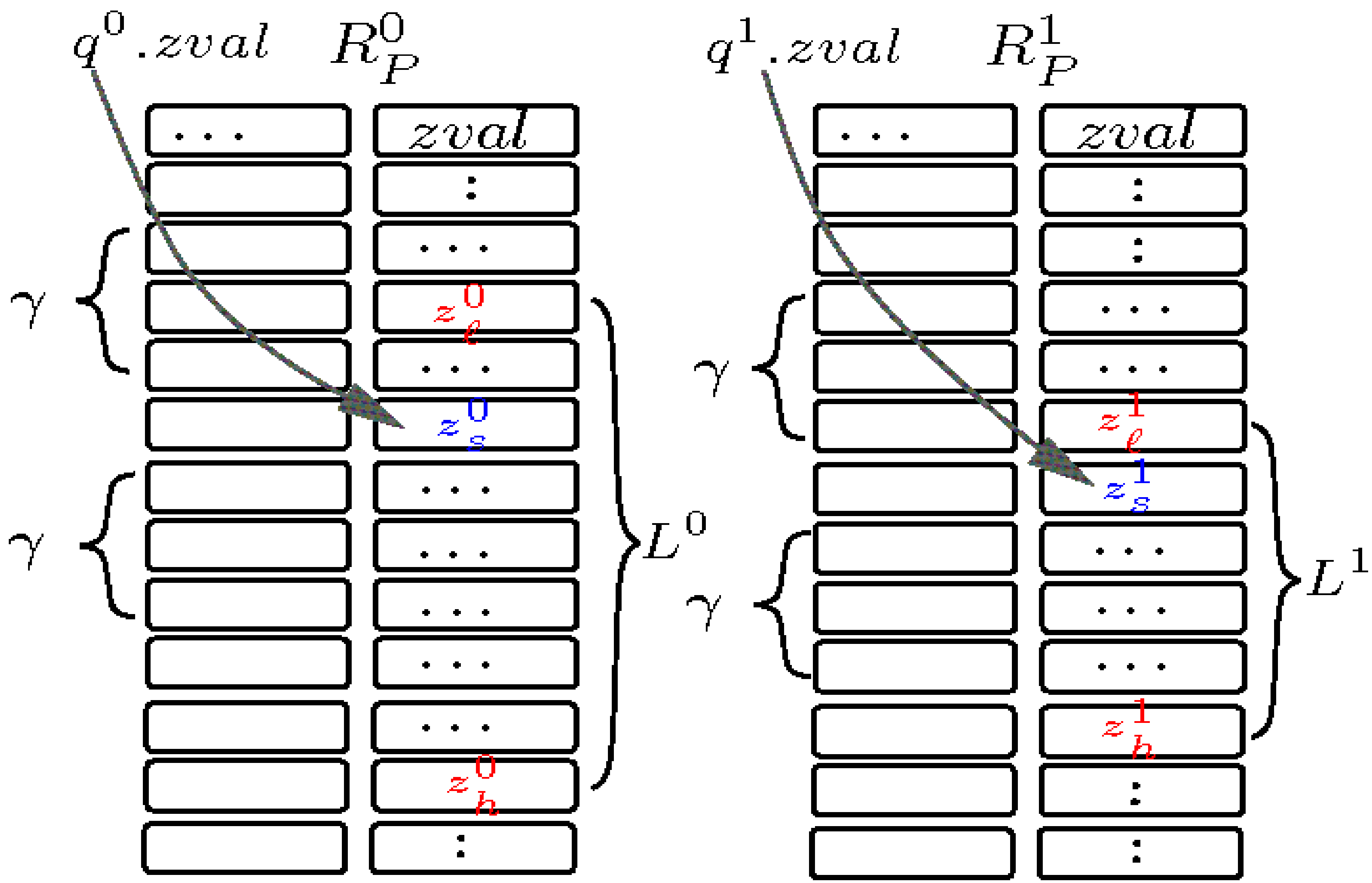
Exact KNN retrieval



Let γ_ℓ and γ_h denote the left and right γ-th points close to the query point, if $z_{\gamma_\ell} \leq z_\ell$ and $z_{\gamma_h} \geq z_h$ in *at least* one of the α tables, $A^\chi = A$

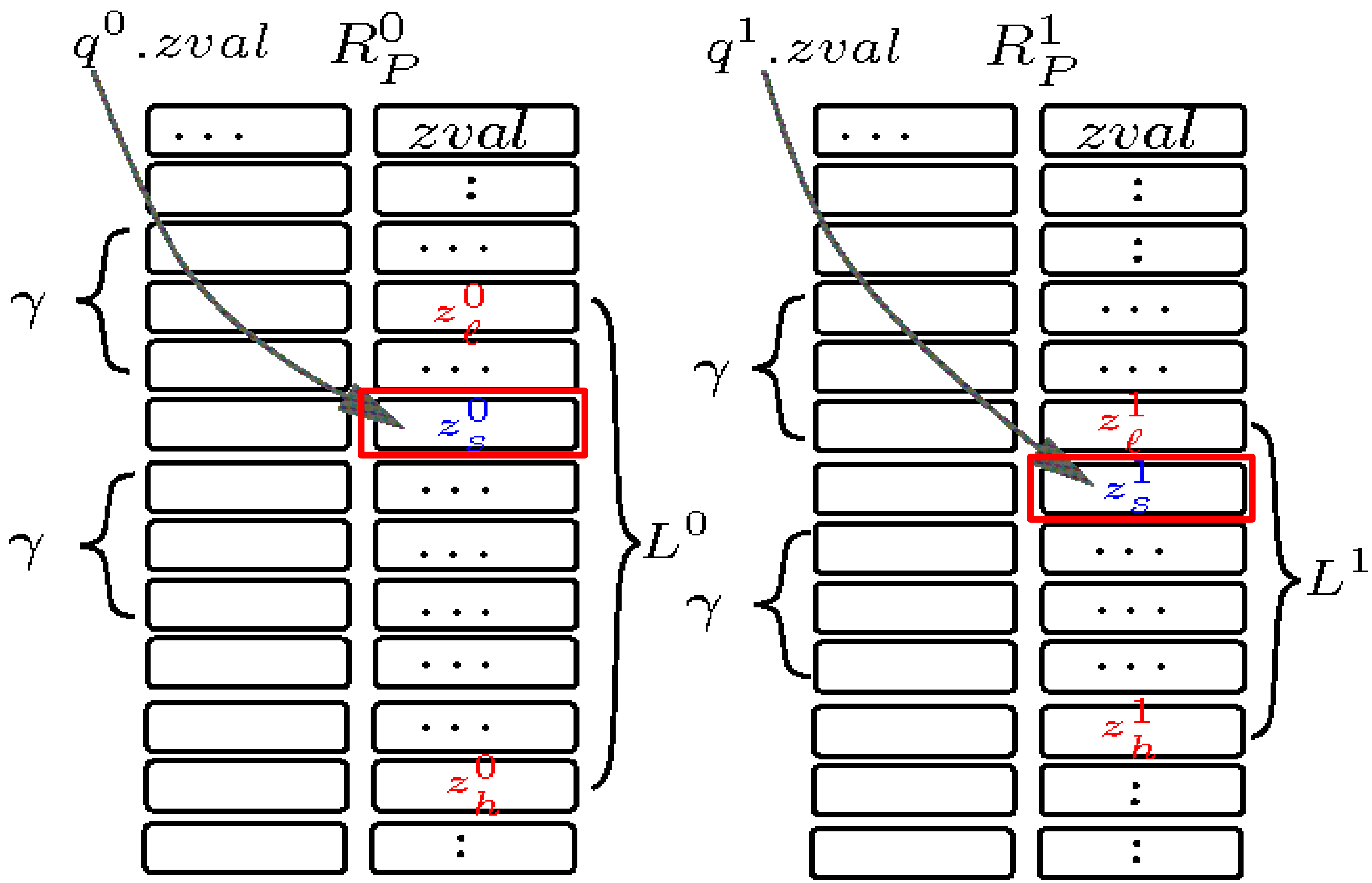
Exact KNN retrieval

If not, we can find A by doing a range query with $[z_\ell^j, z_h^j]$ on any of the α tables. Ideally, we use the table with smallest $[z_\ell^j, z_h^j]$.



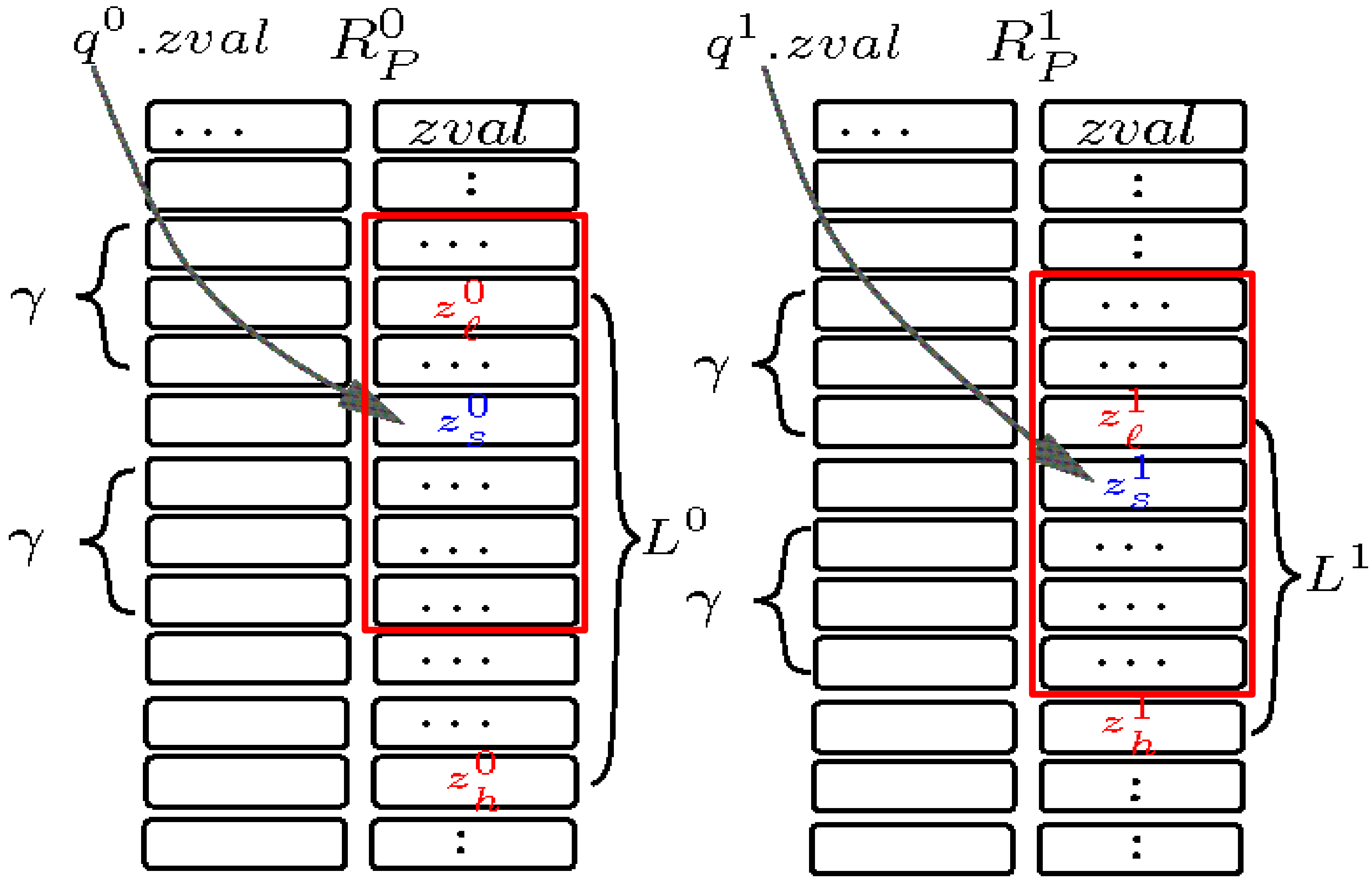
Exact KNN retrieval

If not, we can find A by doing a range query with $[z^j_\ell, z^j_h]$ on any of the α tables. Ideally, we use the table with smallest $[z^j_\ell, z^j_h]$.



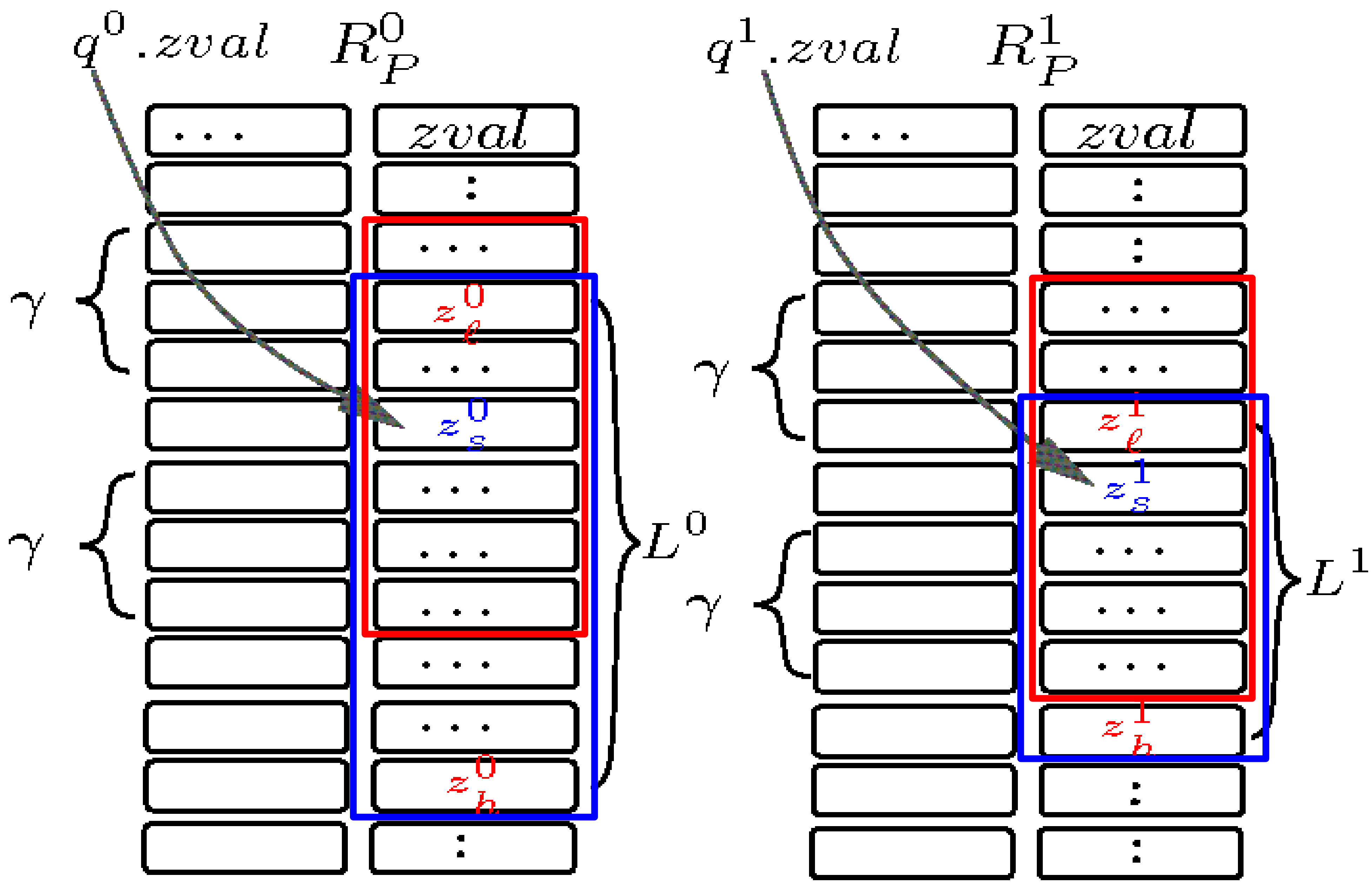
Exact KNN retrieval

If not, we can find A by doing a range query with $[z_\ell^j, z_h^j]$ on any of the α tables. Ideally, we use the table with smallest $[z_\ell^j, z_h^j]$.



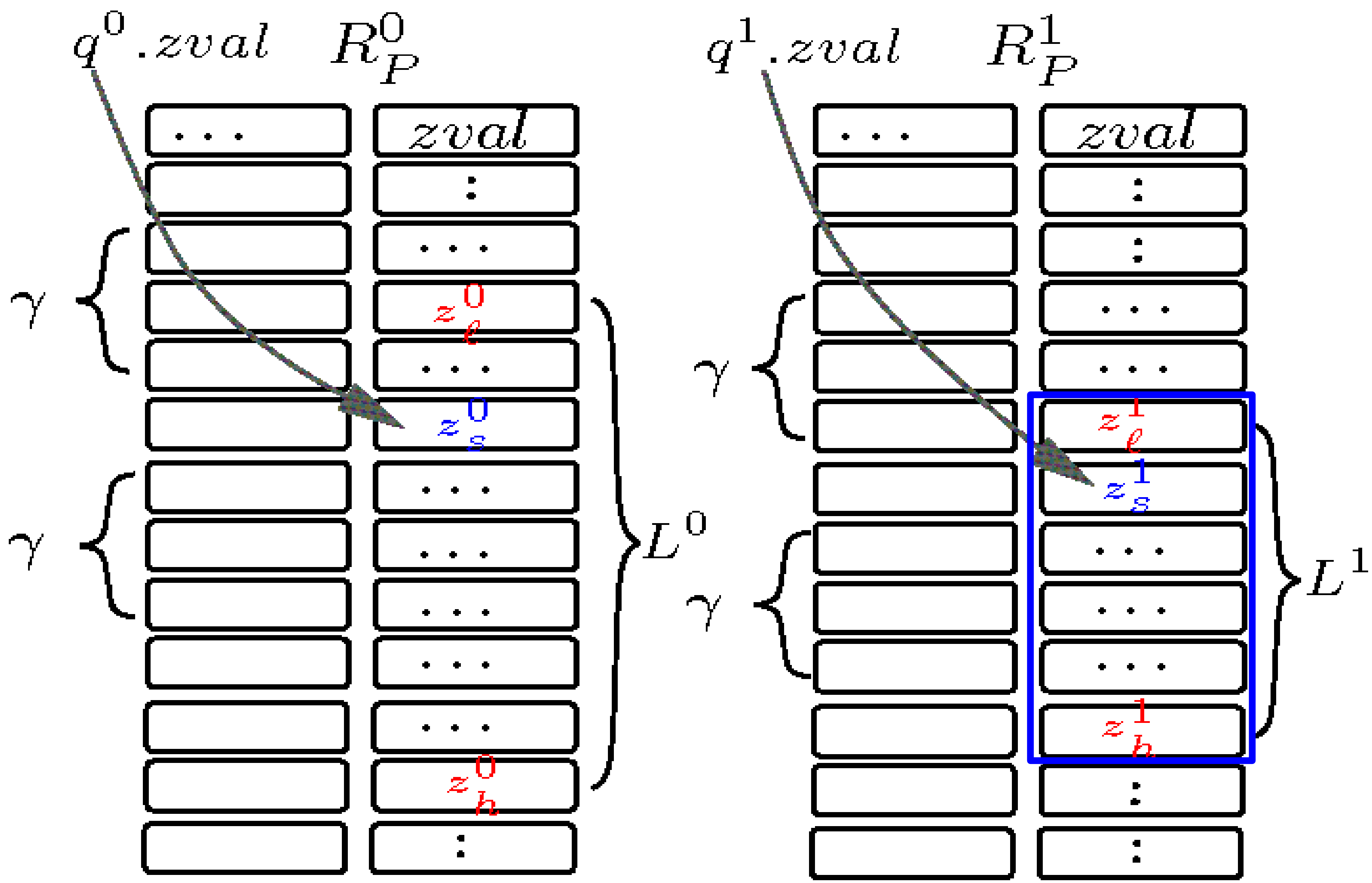
Exact KNN retrieval

If not, we can find A by doing a range query with $[z^j_\ell, z^j_h]$ on any of the α tables. Ideally, we use the table with smallest $[z^j_\ell, z^j_h]$.



Exact KNN retrieval

If not, we can find A by doing a range query with $[z_\ell^j, z_h^j]$ on any of the α tables. Ideally, we use the table with smallest $[z_\ell^j, z_h^j]$.



KNN-join, higher dimensions and updates

- Our approach can easily and efficiently support join queries.

KNN-join, higher dimensions and updates

- Our approach can easily and efficiently support join queries.

- Deal with data in any dimension: without changing the framework; for large dimensionality (say $d > 20$), using LSH-based method.

KNN-join, higher dimensions and updates

- Our approach can easily and efficiently support join queries.
- Deal with data in any dimension: without changing the framework; for large dimensionality (say $d > 20$), using LSH-based method.
- Updates: for deletion, delete record r based on its pid from all talbes $\mathsf{R}^0, \ldots, \mathsf{R}^\alpha$; for insertion, calculate the z-values of the point for all randomly shifted versions, insert them into corresponding tables.

Experiment Setup

- All algorithms are implemented in Microsoft SQL Server 2005. Experiments are conducted on an Intel Xeon CPU @ 2.33GHz. The memory of the SQL Server is set to 1.5GB.

Experiment Setup

- All algorithms are implemented in Microsoft SQL Server 2005. Experiments are conducted on an Intel Xeon CPU @ 2.33GHz. The memory of the SQL Server is set to 1.5GB.

- Real data sets: points representing the road-networks for states in USA

Experiment Setup

- All algorithms are implemented in Microsoft SQL Server 2005. Experiments are conducted on an Intel Xeon CPU @ 2.33GHz. The memory of the SQL Server is set to 1.5GB.

- Real data sets: points representing the road-networks for states in USA

- Two synthetic data sets: uniform points and random clustered points.

Experiment Setup

- All algorithms are implemented in Microsoft SQL Server 2005. Experiments are conducted on an Intel Xeon CPU @ 2.33GHz. The memory of the SQL Server is set to 1.5GB.

- Real data sets: points representing the road-networks for states in USA

- Two synthetic data sets: uniform points and random clustered points.

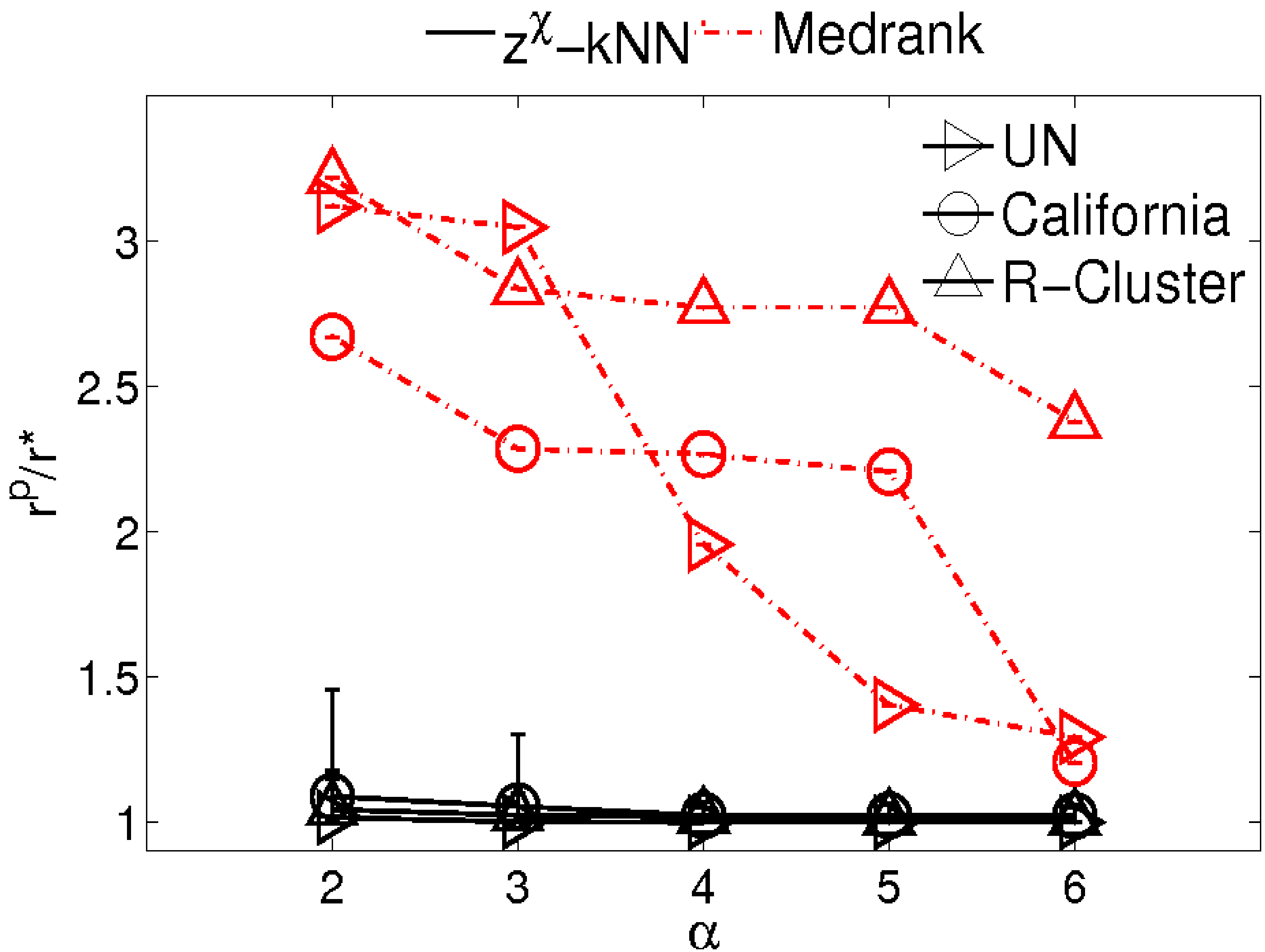
- Compare against the Medrank and iDistance algorithms (implemented by SQL statement and store precedure).

Experiment Setup

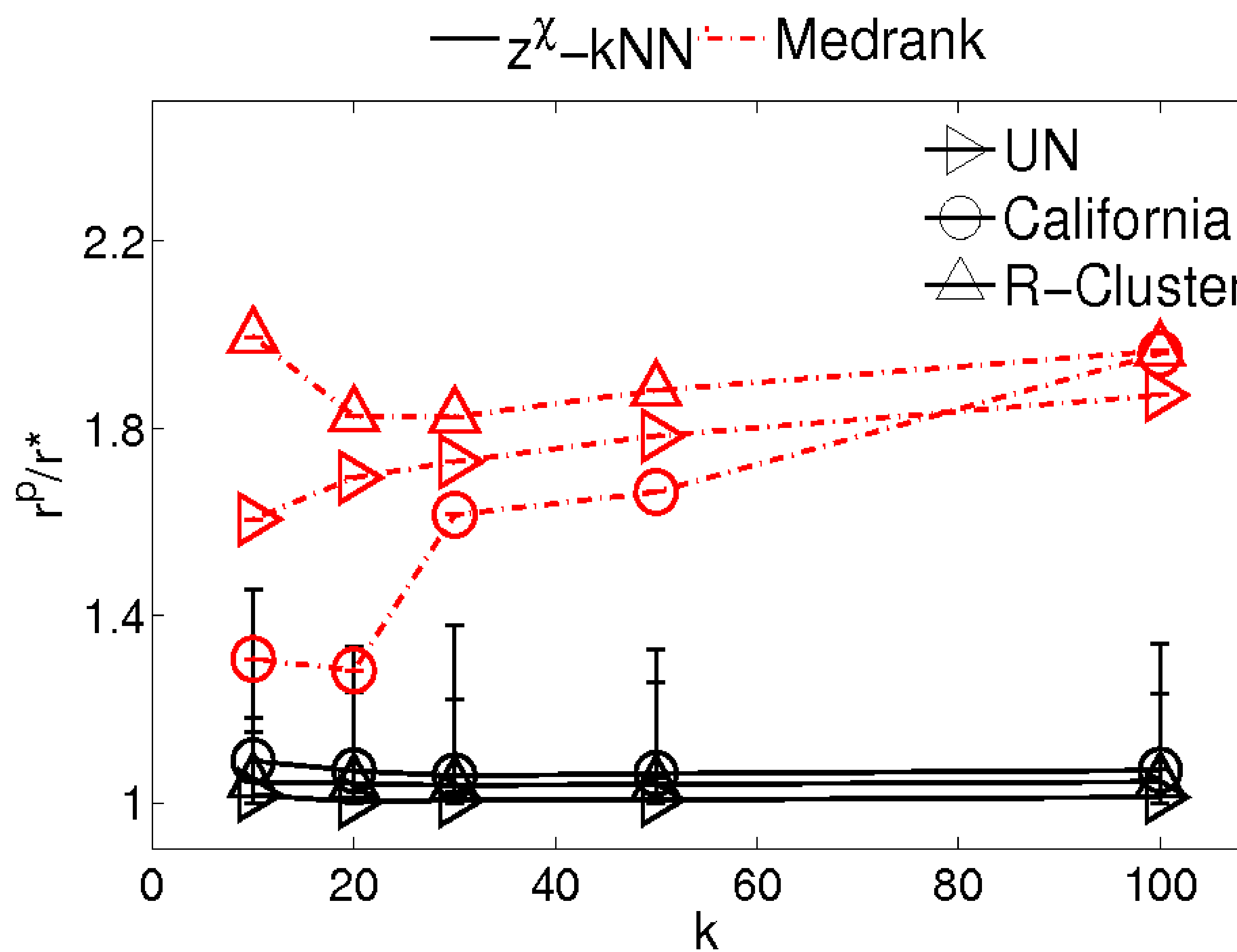
- The default experimental parameters are summarized below

Symbol	Definition	Default Value
k	number of neighbors	10
N	size of points set	1,000,000
α	randomly shifted copies	2
γ	number of points up and down	$2k$
d	dimensionality	2

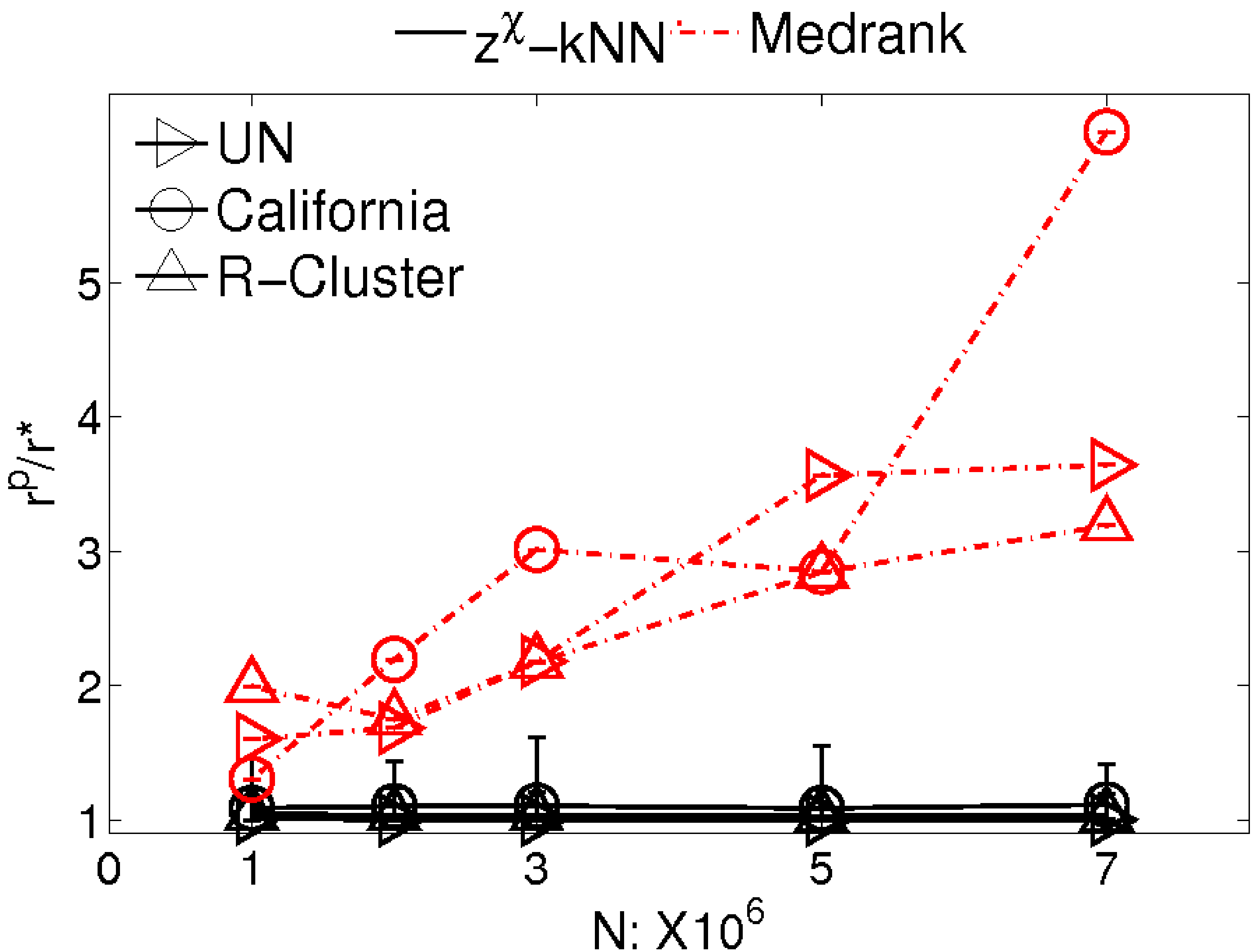
Results for the kNN query: approximation quality



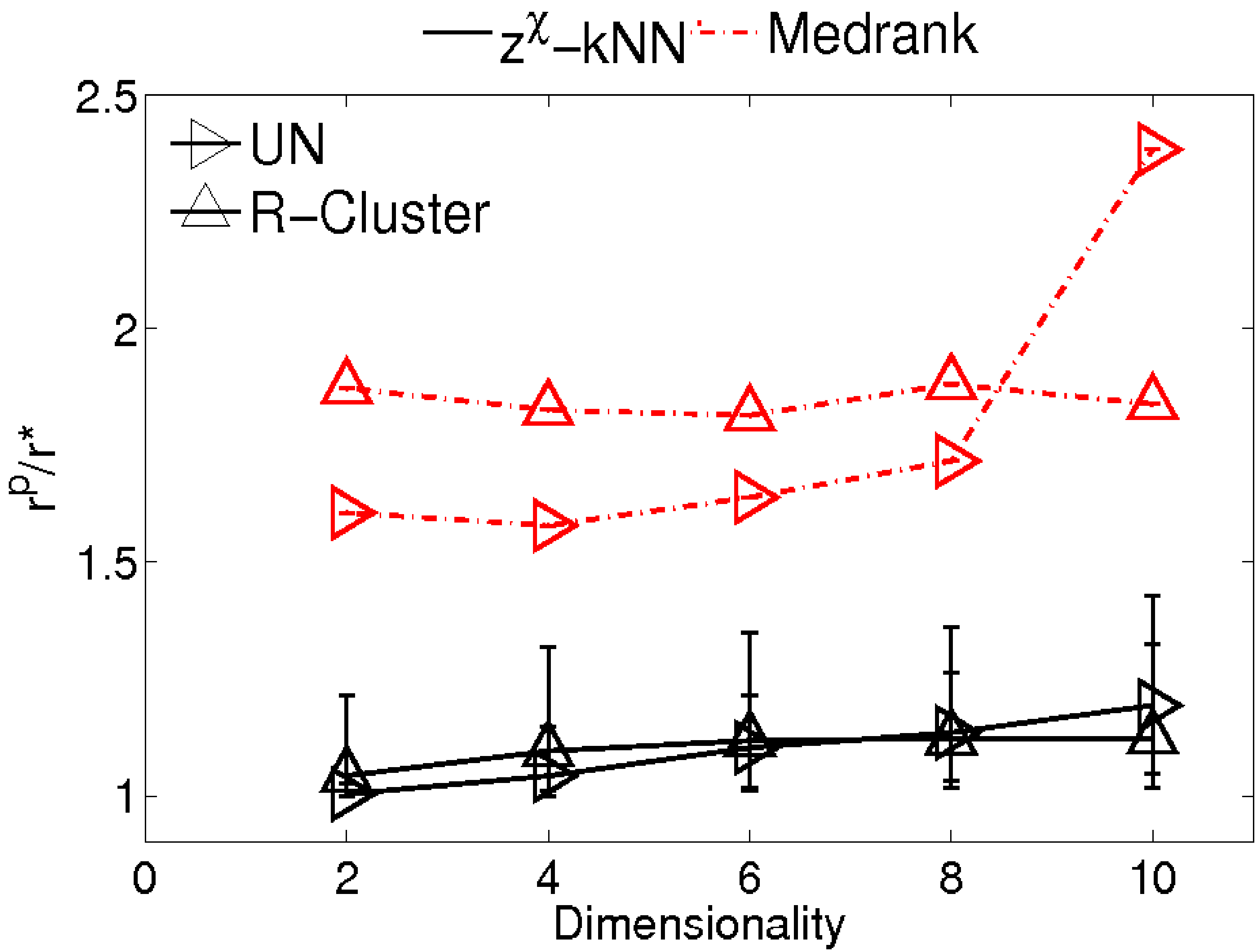
Results for the kNN query: approximation quality



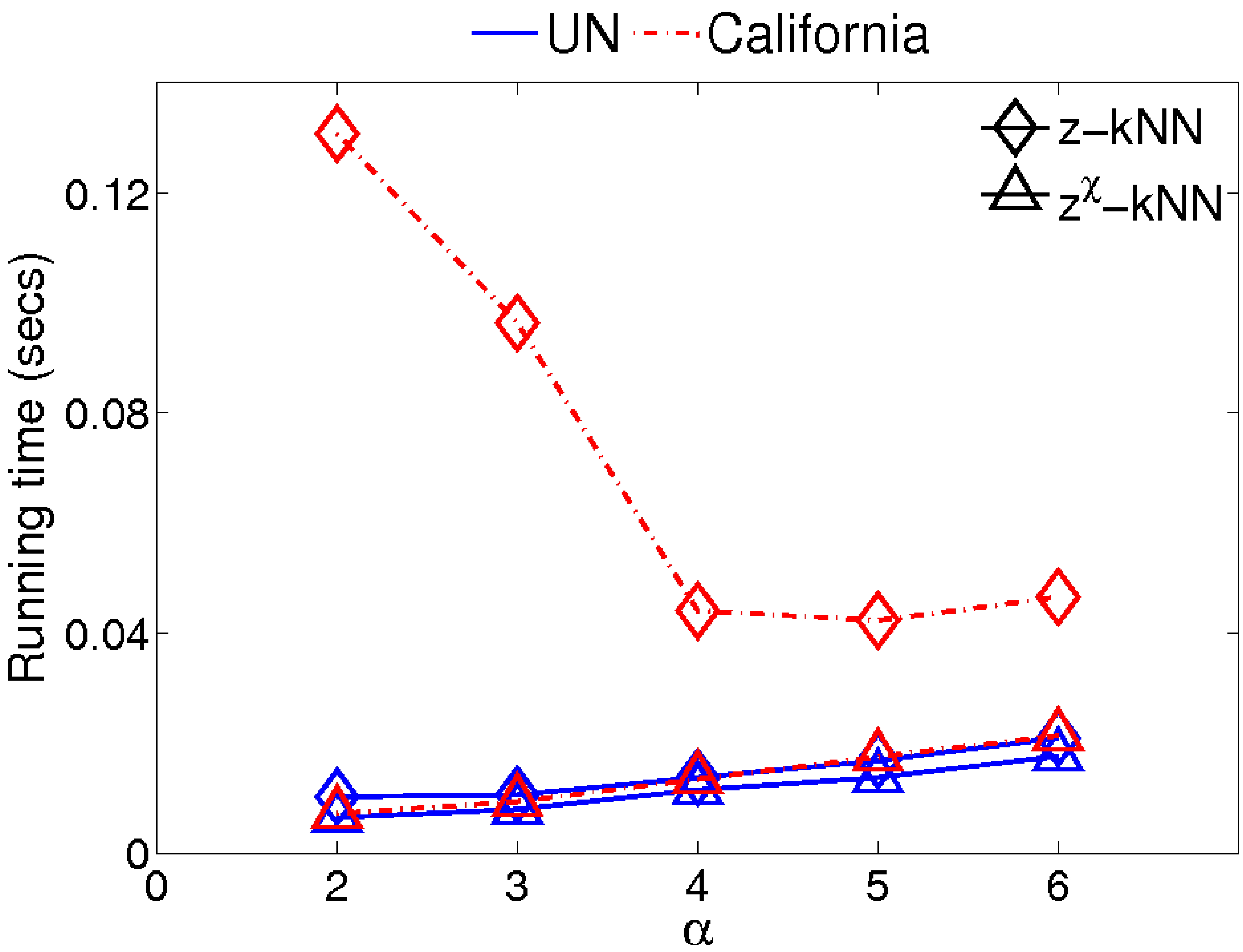
Results for the kNN query: approximation quality



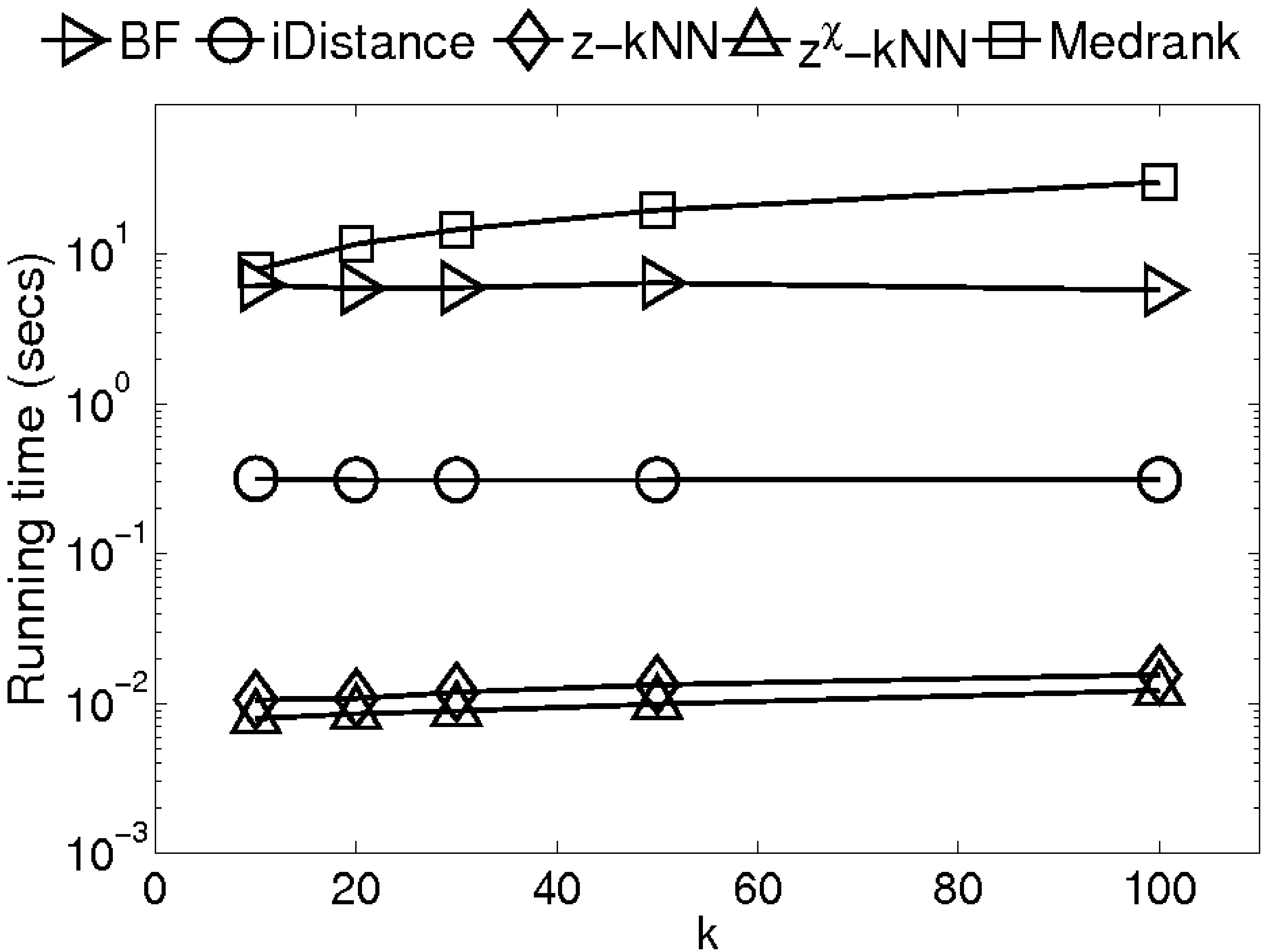
Results for the kNN query: approximation quality



Results for the kNN query: running time

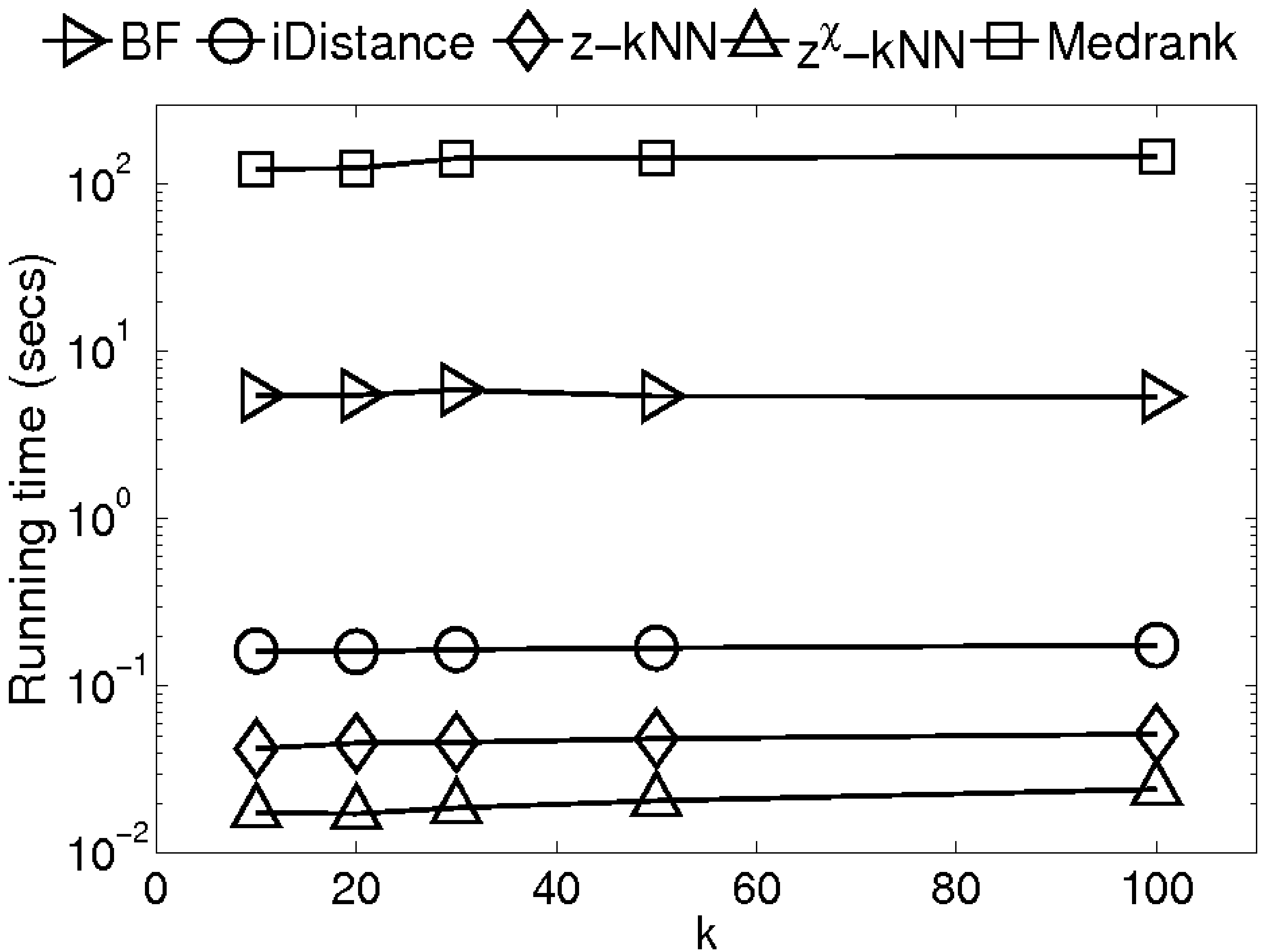


Results for the kNN query: running time



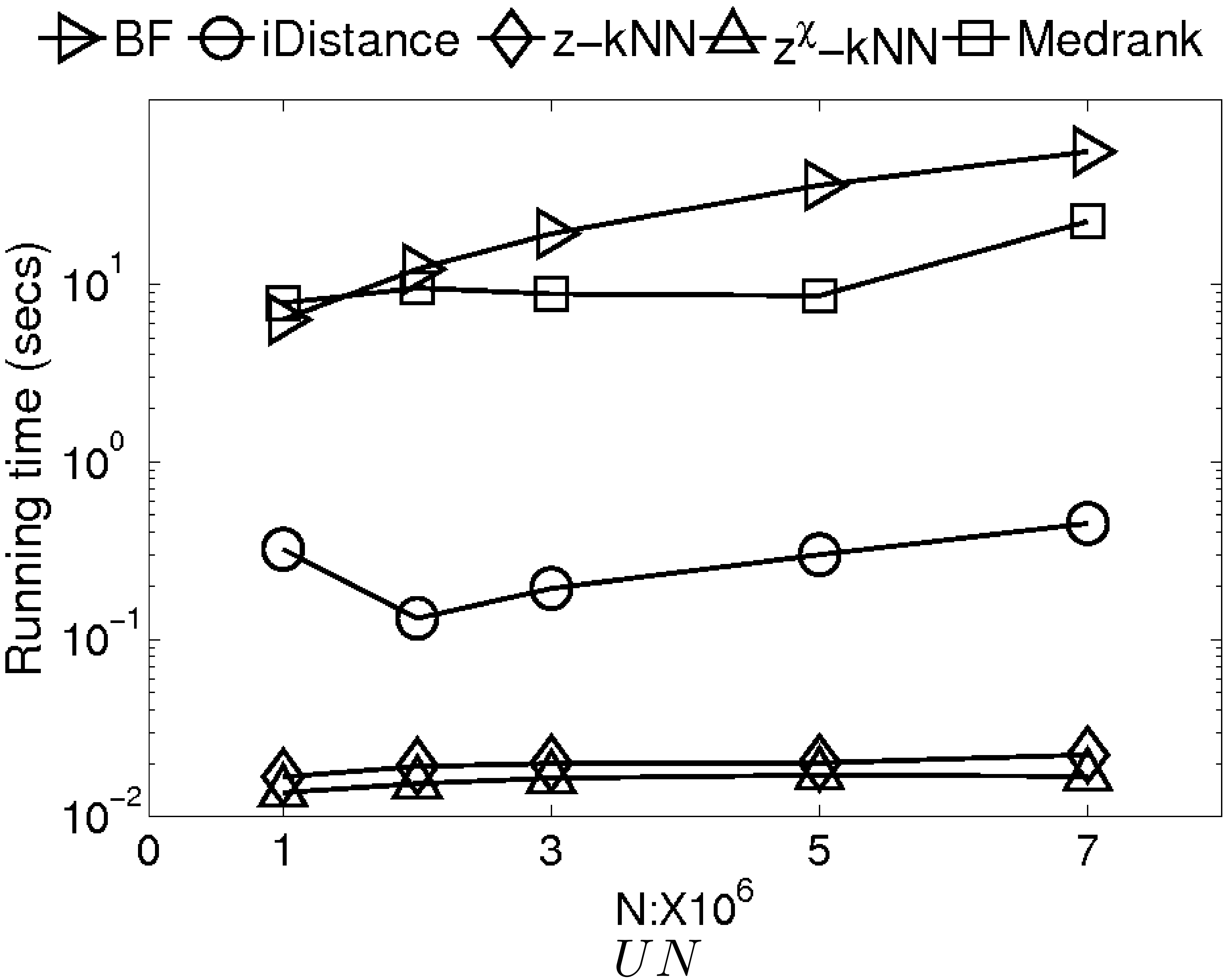
UN

Results for the kNN query: running time

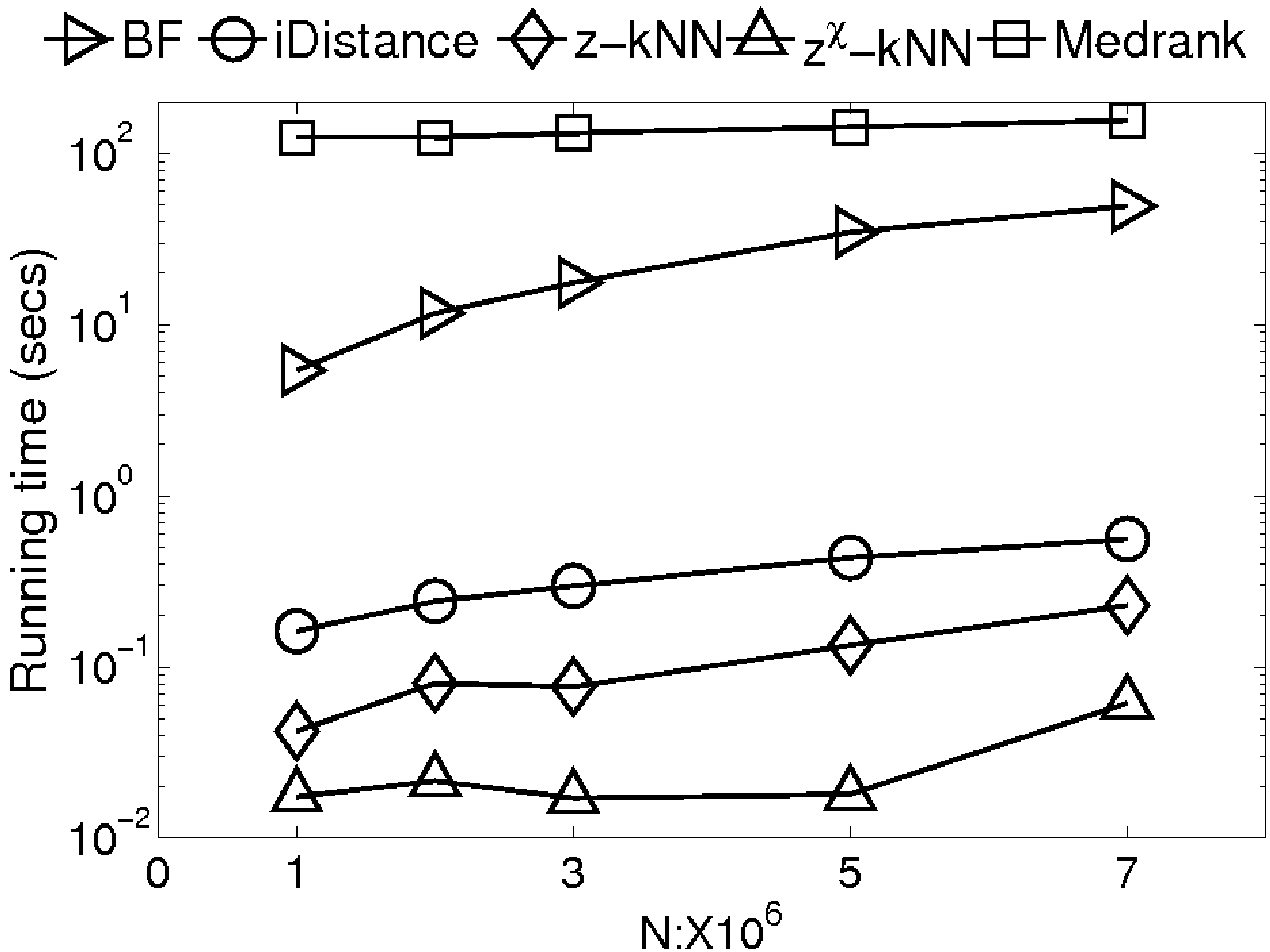


$California$

Results for the kNN query: running time

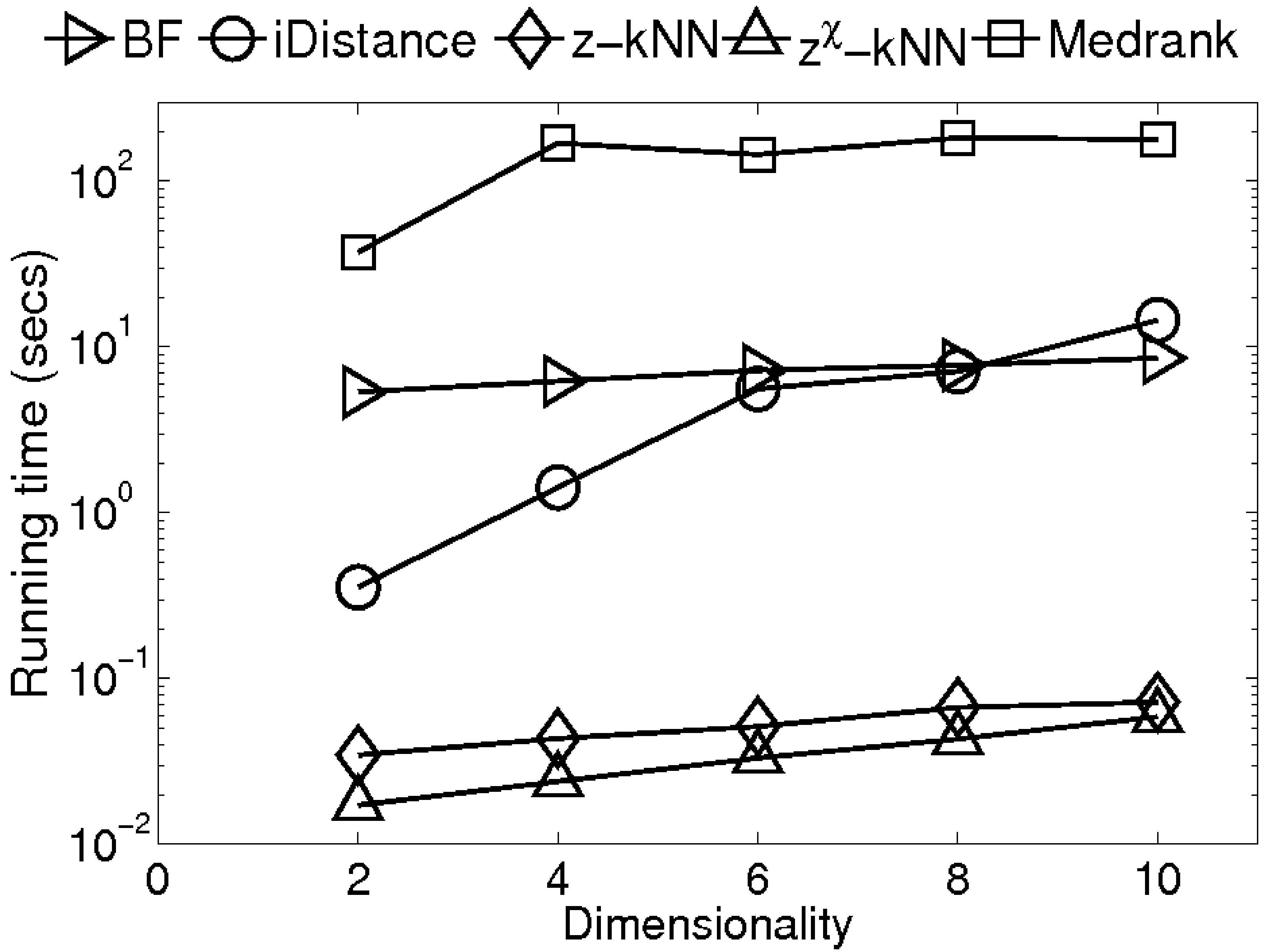


Results for the kNN query: running time



California

Results for the kNN query: running time



Conclusions

- Presented a constant approximation for the kNN query, with logarithmic page accesses in any fixed dimension and extended it to the exact solution, both using just $O(1)$ random shifts.

Conclusions

- Presented a constant approximation for the kNN query, with logarithmic page accesses in any fixed dimension and extended it to the exact solution, both using just $O(1)$ random shifts.
- All the algorithms can be implemented by SQL operators in relational databases.

Conclusions

- Presented a constant approximation for the kNN query, with logarithmic page accesses in any fixed dimension and extended it to the exact solution, both using just $O(1)$ random shifts.
- All the algorithms can be implemented by SQL operators in relational databases.
- Our approach naturally supports kNN-Joins.

Conclusions

- Presented a constant approximation for the kNN query, with logarithmic page accesses in any fixed dimension and extended it to the exact solution, both using just $O(1)$ random shifts.
- All the algorithms can be implemented by SQL operators in relational databases.
- Our approach naturally supports kNN-Joins.
- No changes are required for different dimensions, and the update is trivial.

Conclusions

- Presented a constant approximation for the kNN query, with logarithmic page accesses in any fixed dimension and extended it to the exact solution, both using just $O(1)$ random shifts.
- All the algorithms can be implemented by SQL operators in relational databases.
- Our approach naturally supports kNN-Joins.
- No changes are required for different dimensions, and the update is trivial.
- Future research:
  - Study other related, interesting queries in this framework, e.g., the reverse nearest neighbor queries.
  - Examine the relational algorithms to the data space other than the L_p-norms, such as the road networks.

The End

$$THANK \quad YOU$$

Q and A