SPEEDING UP PERMUTATION BASED INDEXING WITH INDEXING

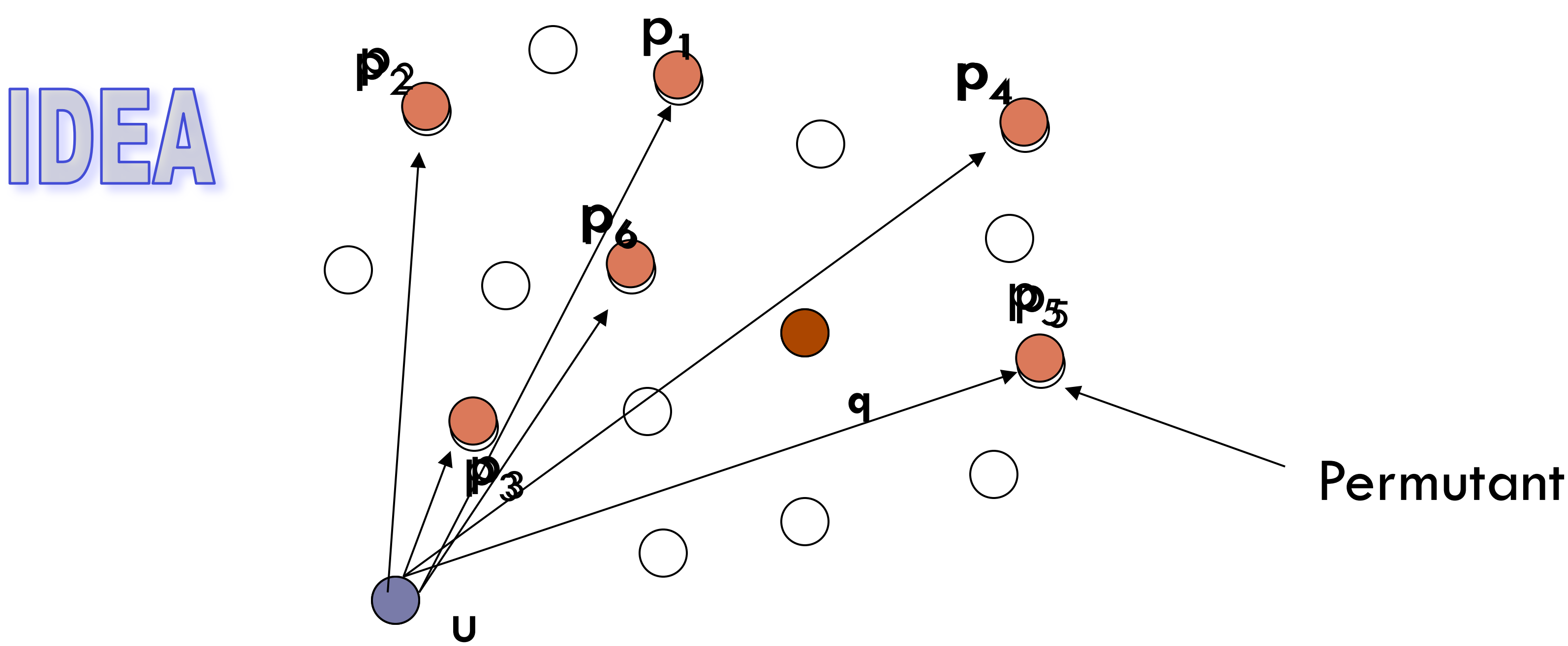
Karina Figueroa
Universidad Michoacana
México

Kimmo Frediksson
University of Kuopio
Finland

Content

- Permutation-based algorithm
- Indexing Permutations
- Experimental Results
- Conclusions

Permutation based algorithms



$$\Pi_u = p_3, p_6, p_2, p_1, p_5, p_4$$
$$\Pi_q = p_6, p_5, p_4, p_1, p_3, p_2$$

Spearman Footrule metric

$$\Pi_q = p_1, p_2, p_3, p_4, p_5, p_6$$

$$\Pi_u = p_3, p_6, p_2, p_1, p_5, p_4$$

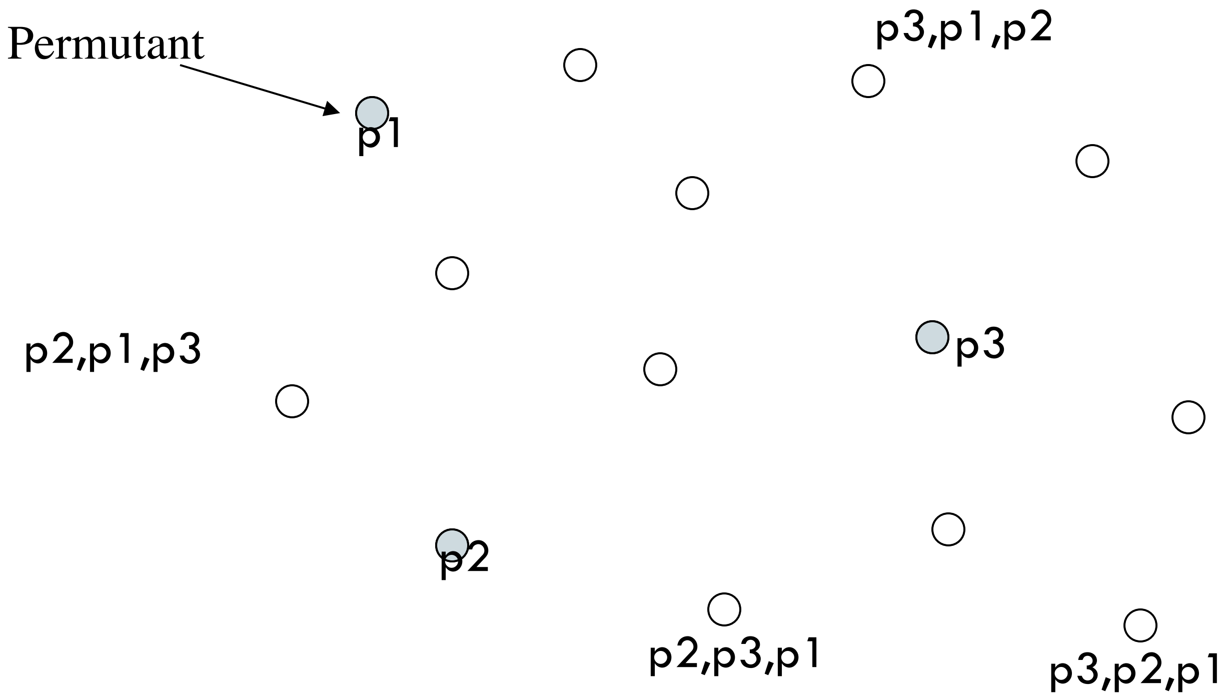
3-1, 6 - 2, 3-2, 4-1, 5-5, 6-4

Differences between positions

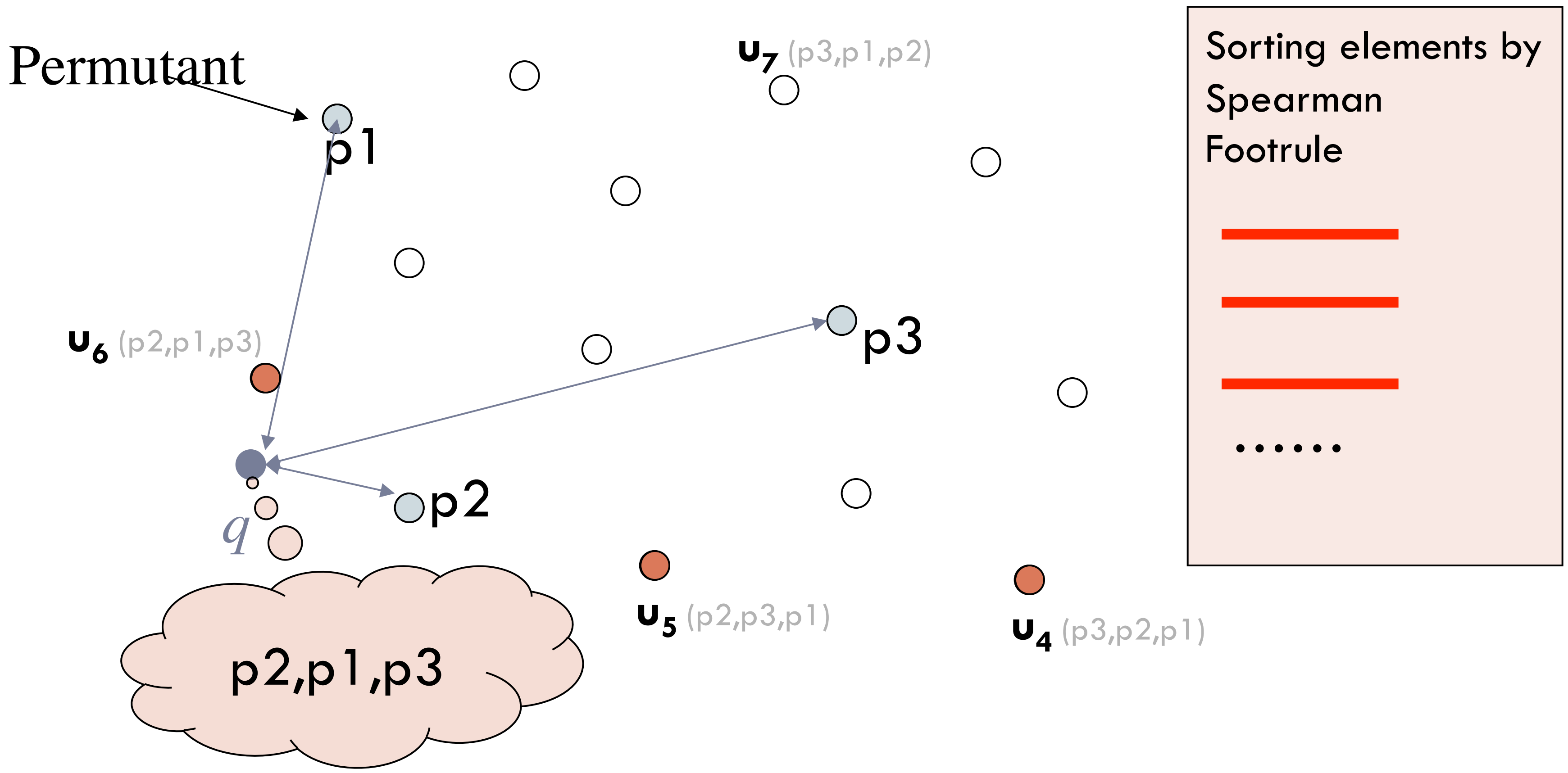
$$F(\Pi_q, \Pi_u) = |1-3| + |2-6| + |3-2| + |4-1| + |5-5| + |6-4| = 12$$

Permutation-based algorithm

Preprocessing



Search Process



Permutations based algorithm

Algorithm 1 Sequential-Range-Query(q, r, f)

INPUT: q is a query and r its radius, f is the fraction of the database to traverse.
OUTPUT: Reports a subset of those $u \in \mathbb{U}$ that are at distance at most r to q.
Let $A[1, n]$ be an array of tuples and $\mathbb{U} = \{u_1, \ldots, u_n\}$
Compute Π_q^{-1}
for $i \leftarrow 1$ to n **do**
  $A[i] \leftarrow \langle u_i, S_\rho(\Pi_{u_i}, \Pi_q) \rangle$
end for
SortIncreasing(A) // by second component of tuples
for $i \leftarrow 1$ to $f \cdot n$ **do**
  Let $\langle u, s \rangle = A[i]$
  if $d(q, u) \leq r$ **then**
    Report u
  end if
end for

$O(kn + n \log n)$

Distance evaluations.

Our proposal

- Define a new metric space
  - The permutations of the database
  - The metric between permutations
- In this new metric space
  - We can use any algorithm for metric spaces
  - Finds the most similar permutations to the query
    - Avoids making comparisons through the whole database
    - Reduces the searching time

Indexing Permutations

Algorithm 2 Indexed-Range-Query(q, r, f)

```
INPUT: q is a query and r its radius, f is the fraction
       of the database to traverse.
OUTPUT: Reports a subset of those u ∈ 𝕌 that are at
       distance at most r to q.
Let A[1, f · n] be an array of elements and W an index
       of permutations
Compute Π_q^{-1}
A ← NearestNeighbors(W, Π_q, f · n)
for i ← 1 to f · n do
    if d(q, A[i]) ≤ r then
        Report u
    end if
end for
```

Nearest Neighbor query

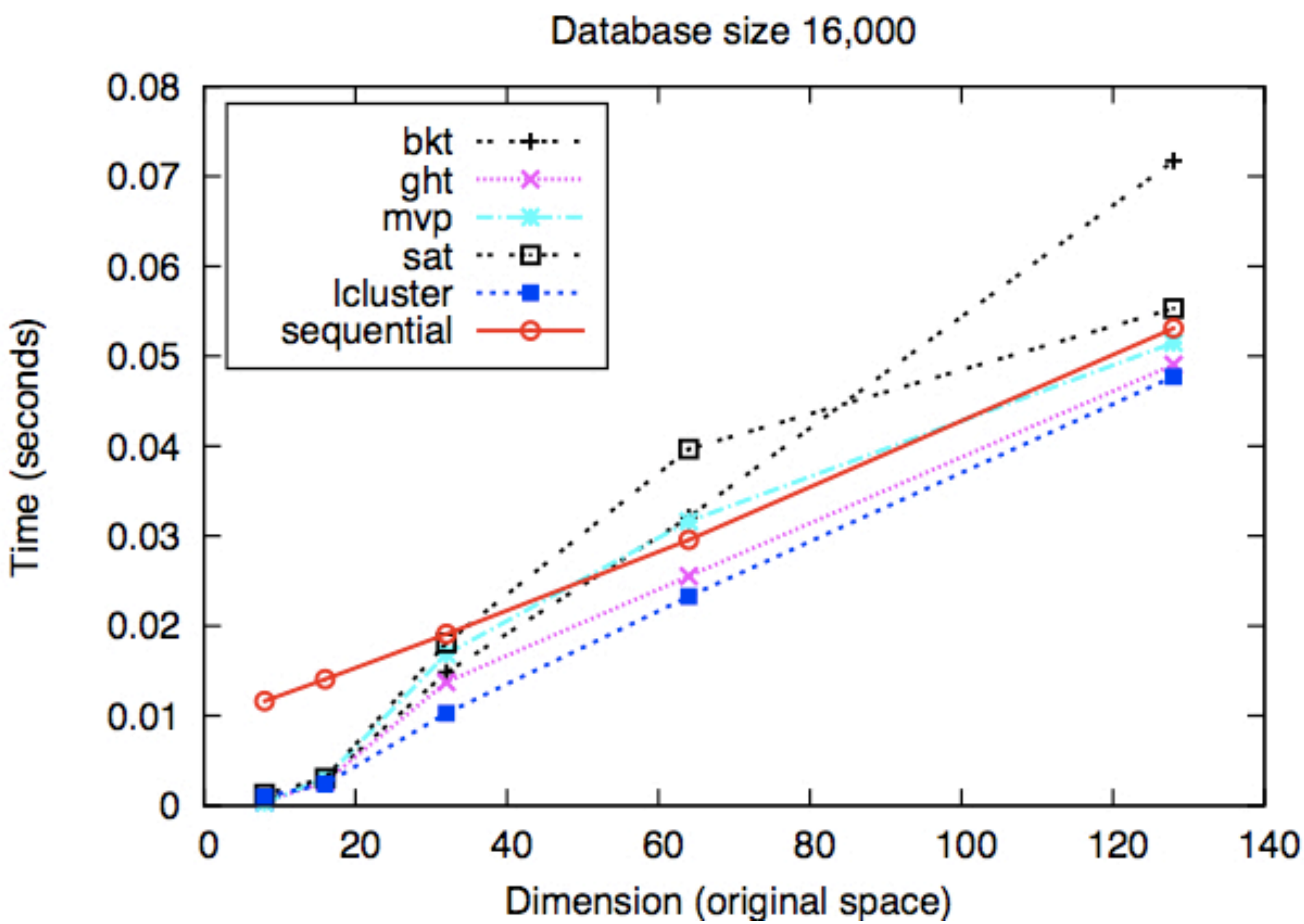
- What Algorithm can we use?
  - Any for metric spaces
    - Because it is a metric space
  - LClusters
  - GHT
  - BKT
  - MVP
  - SAT
  - Also, permutation-based (again and again, etc)

Databases

- Synthetic vectors in an unitary cube
  - Dimensions from 8 to 128
  - 16,000 and 40,000 objects
- Gaussians vectors
  - 10,000 object
- Spanish Language dictionary
  - 86,000 words
- Faces (762 dimensions)
    - FERET
    - 761 images

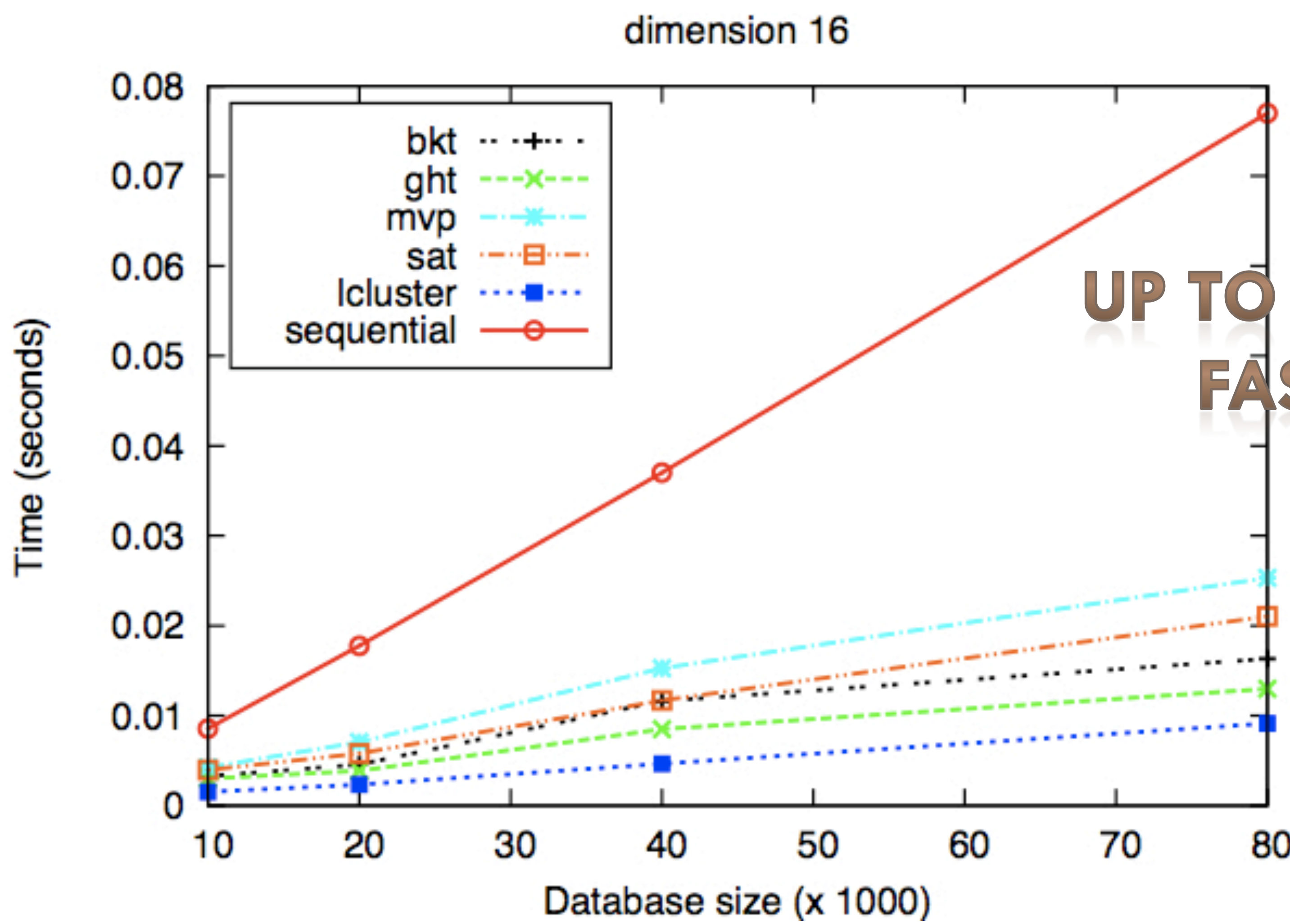
Experimental Results

Synthetic vectors in a Unitary cube



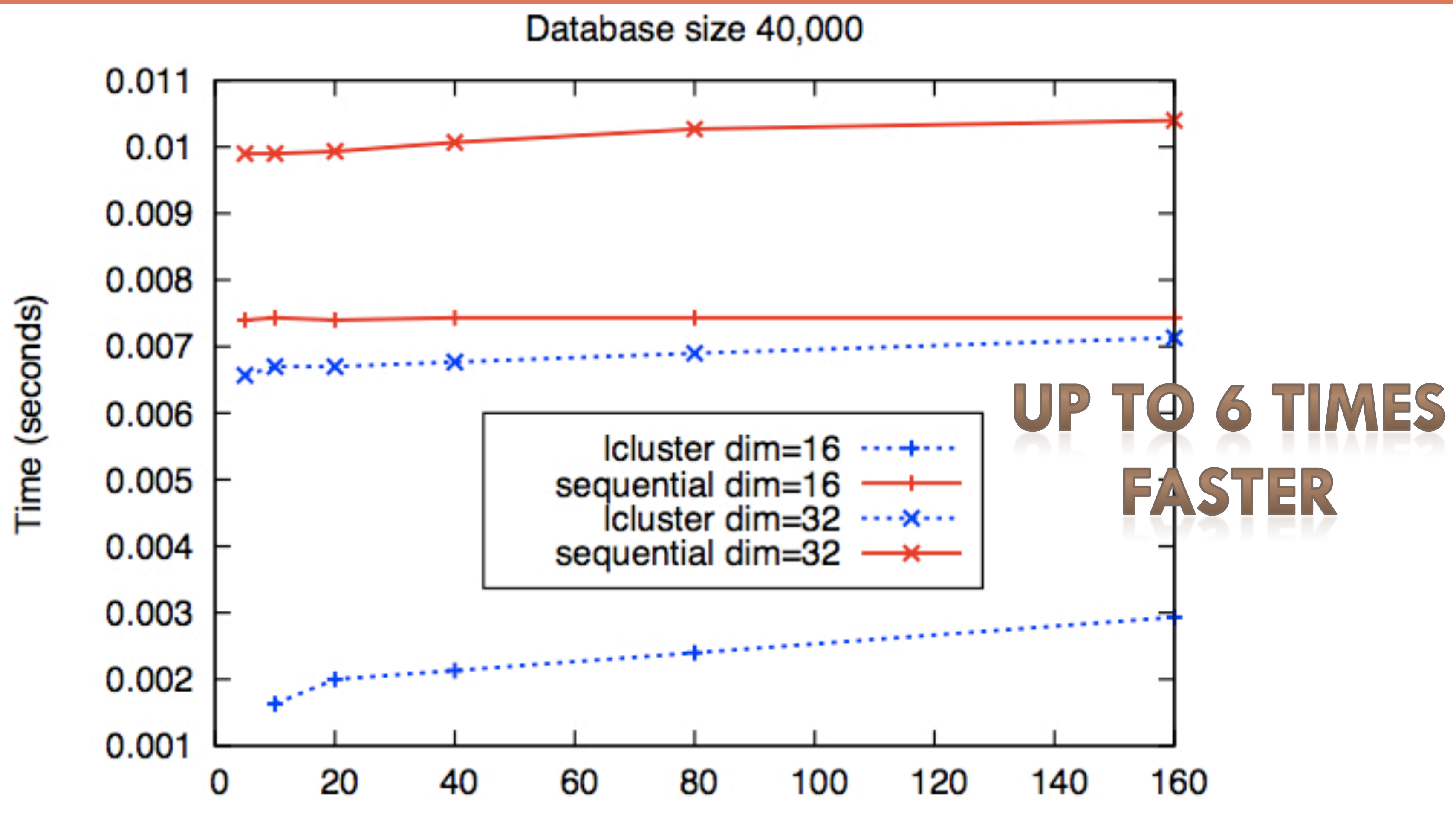
Experimental Results

Synthetic vectors in a Unitary cube



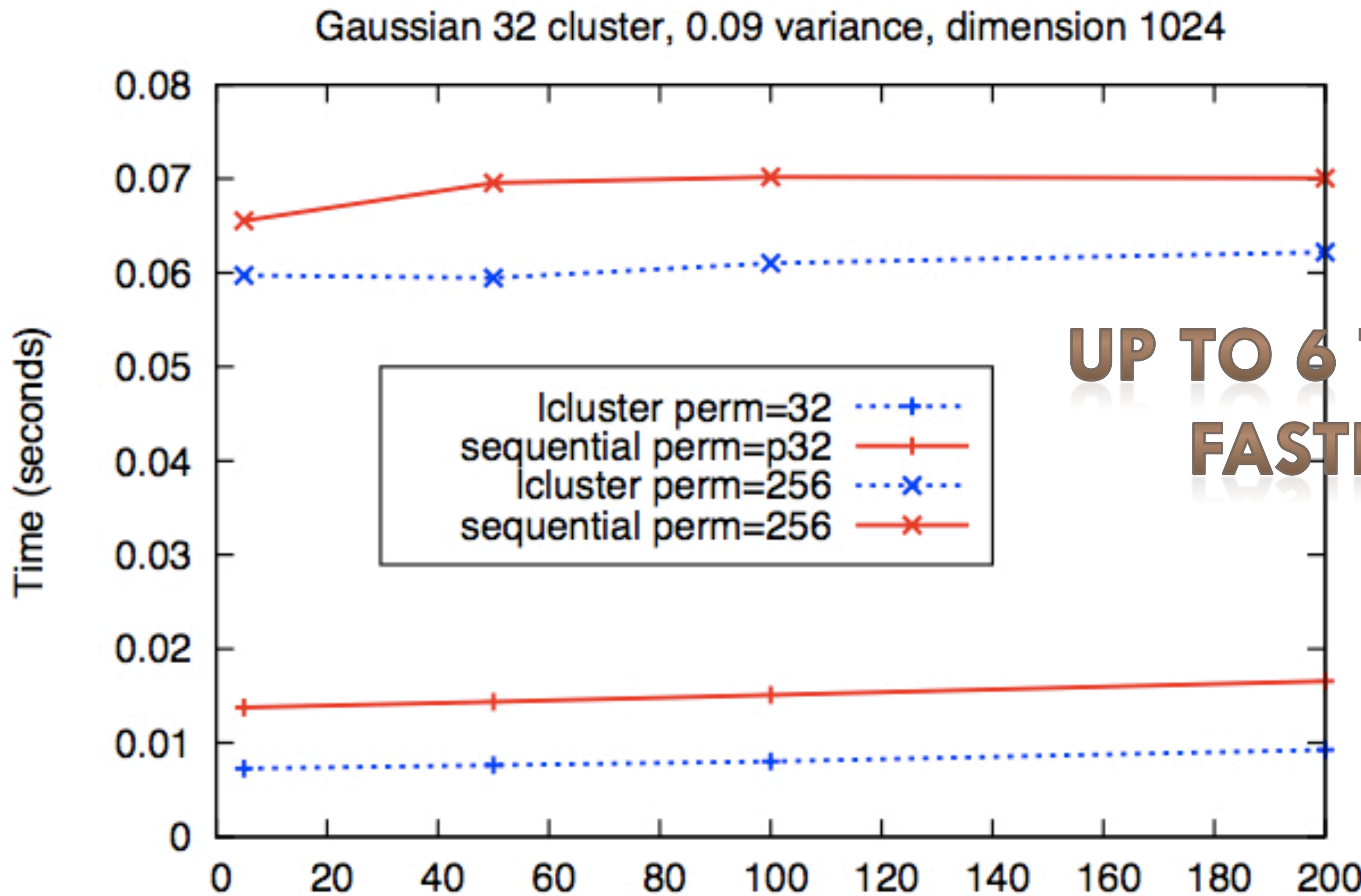
UP TO 7 TIMES FASTER

Vectors: Unitary cube



k most similar permutations

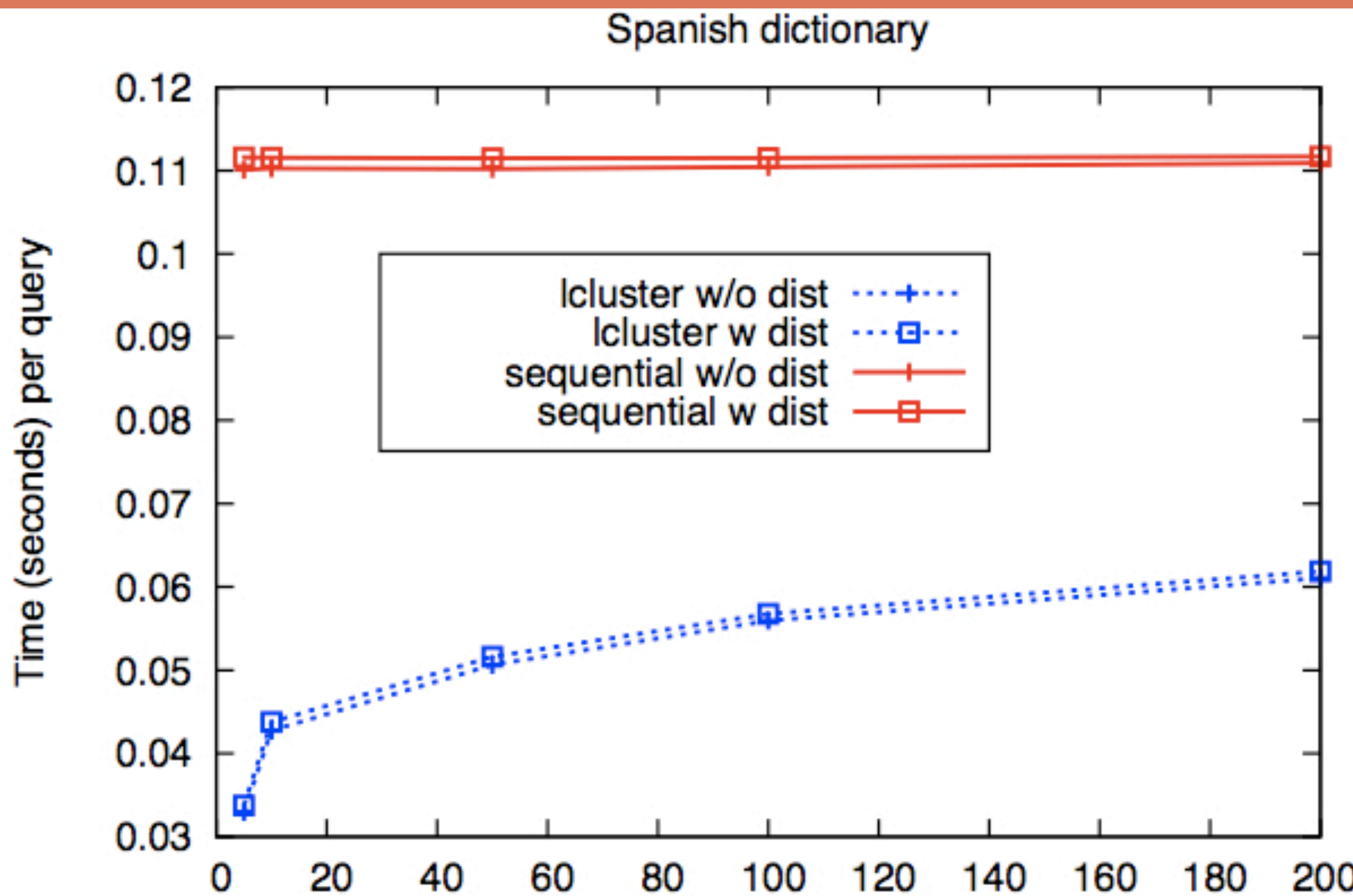
Gaussians vectors



most similar permutations

UP TO 6 TIMES FASTER

Strings: Spanish Language dictionary



k most similar permutations

Conclusions

- We have significantly reduced the CPU time of one of the best algorithms for searching in very high dimensional metric spaces

- We were able to do this by identifying another metric space search problem in the algorithms internals.

Thanks

karina@fismat.umich.mx

Kimmo.frediksson@uku.fi