EGNAT: A Fully Dynamic Metric Access Method for Secondary Memory

Roberto Uribe-Paredes [1,2] Gonzalo Navarro [3]

[1]Depto. de Ingeniería en Computación,
Universidad de Magallanes, Chile

[2]Grupo de Bases de Datos - UART,
Universidad Nacional de la Patagonia Austral, Río Turbio, Argentina

[3]Dept. of Computer Science,
University of Chile, Chile
E-mail: roberto.uribeparedes@gmail.com, gnavarro@dcc.uchile.cl

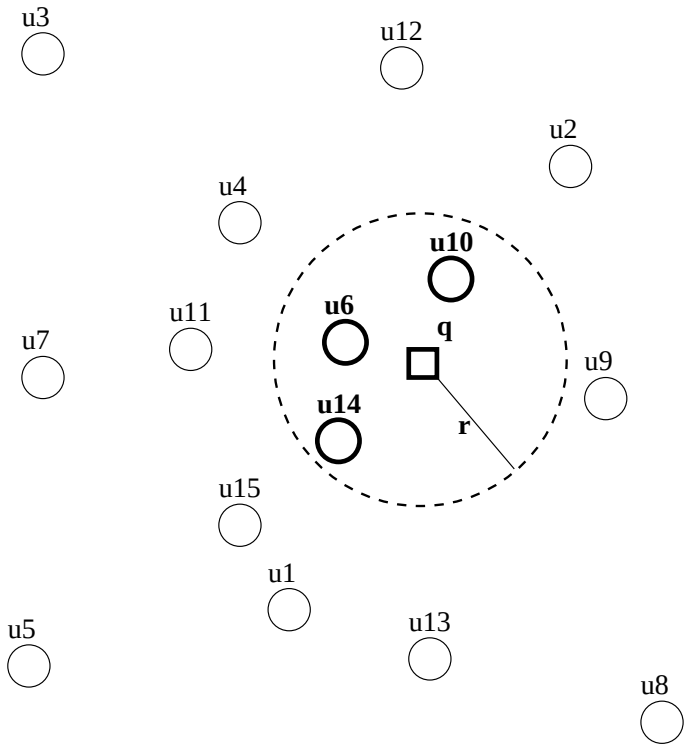
September 4, 2009

Abstract

- We introduce a novel metric space search data structure called *EGNAT*, which is fully dynamic and designed for secondary memory. The *EGNAT* implements deletions using a novel technique dubbed *Ghost Hyperplanes*.

Similarity Search

- Range search
- The k nearest neighbors



Metric Structures

Construction Methods

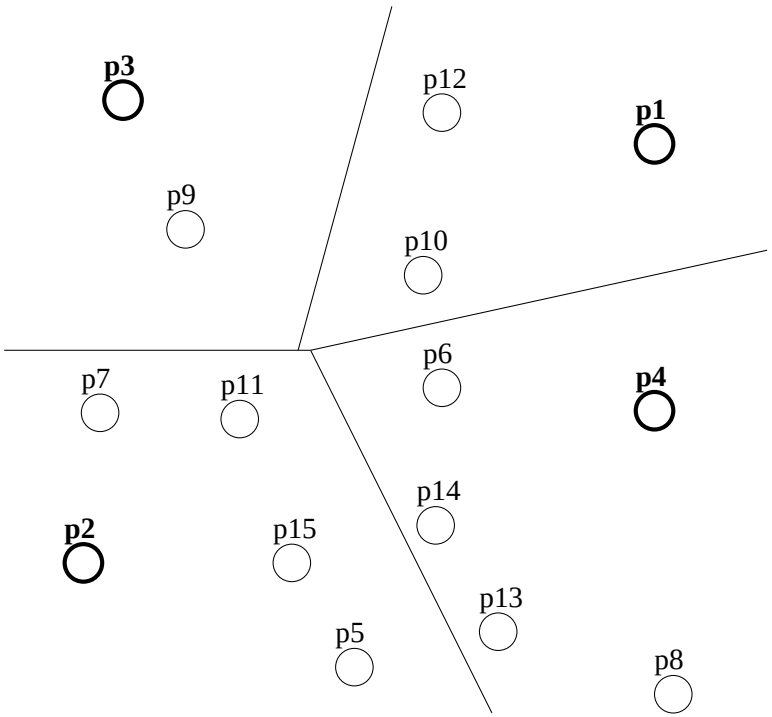
- Based on Pivots
  $|d(q, p_i) - d(x, p_i)| > r \Rightarrow d(q, x) > r$
- Based on Clustering or compact partitioning
  - Voronoi Partitioning Criterion (or Hyperplanes)
    $d(q, c_j) > d(q, c_i) + 2r$
  - Covering Radius Criterion $d(q, c_i) - r > rc(c_i)$

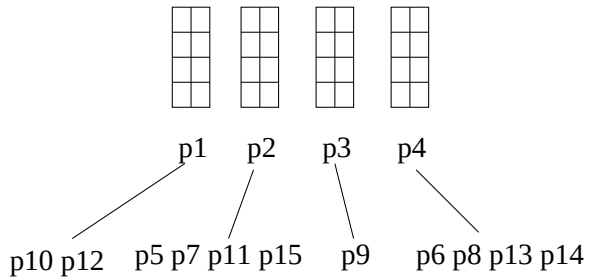
General Information

- This is a new method based on the GNAT [Brin95], that is based on the GHT (Generalized Hyperplane Tree) [Uhlmann91].
- It is based on both clustering and used the Hyperplanes and Covering radius criteria.
- It is dynamic and enhanced for Secondary Memory.

Construction

- Each child p_i maintains a table of distance ranges toward the rest of the sets D_{p_j},
  $range_{i,j} = (\min\{d(p_i, x), x \in D_{p_j}\}, \max\{d(p_i, x), x \in D_{p_j}\})$.

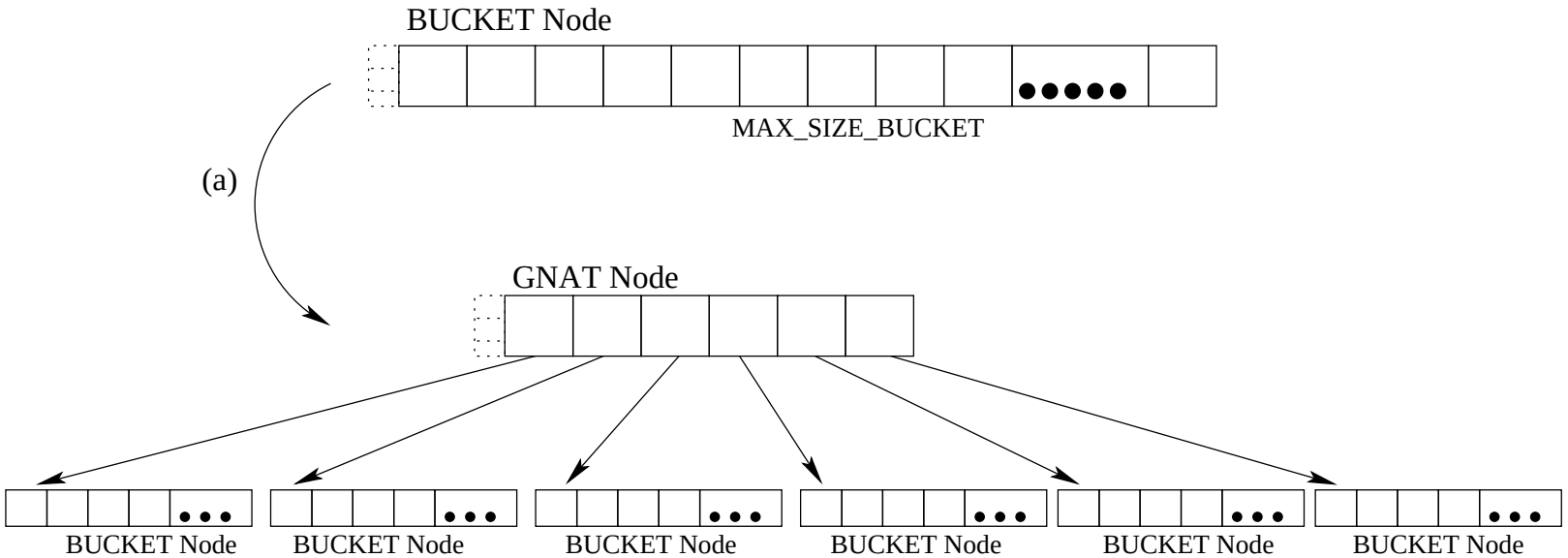




Construction

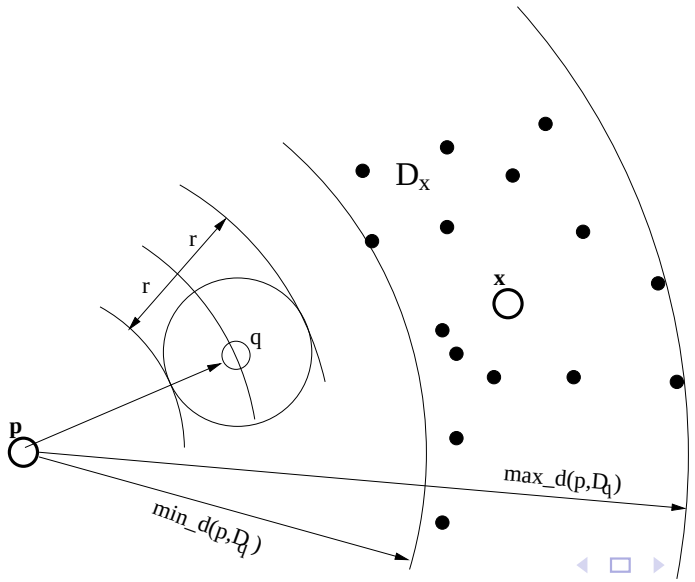
Two types of nodes:

- *buckets* (leaves)
- *gnats* (internal nodes)



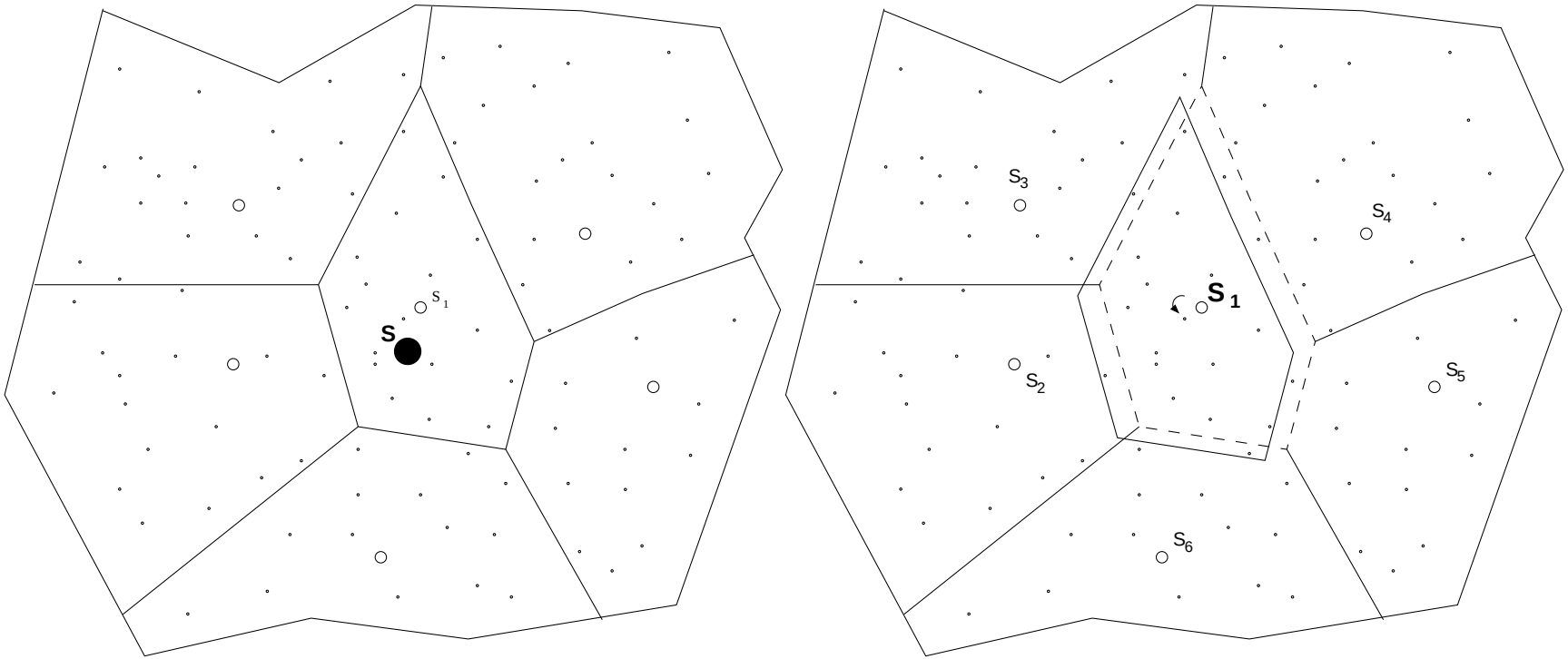
Search

- *buckets* (leaves)
  If $|d(x,p) - d(q,p)| > r$ or
- *gnats* (internal nodes)
  $range(p_0, q) \cap range(p_0, D_{p_x}) = \emptyset$
  then we know that $d(x,q) > r$ without computing that distance



Ghost Hyperplanes

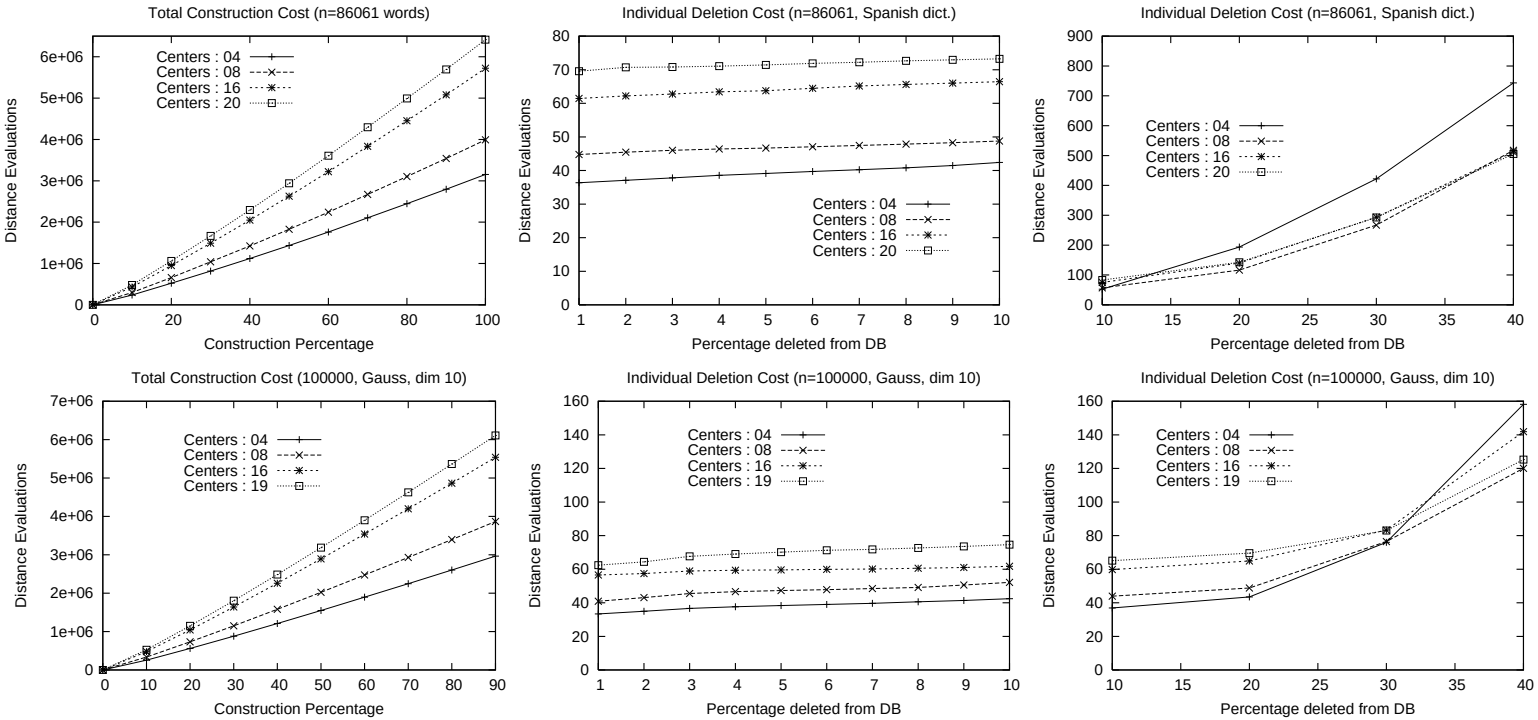
- Partition of the space before and after a deletion.



Choosing the Replacement

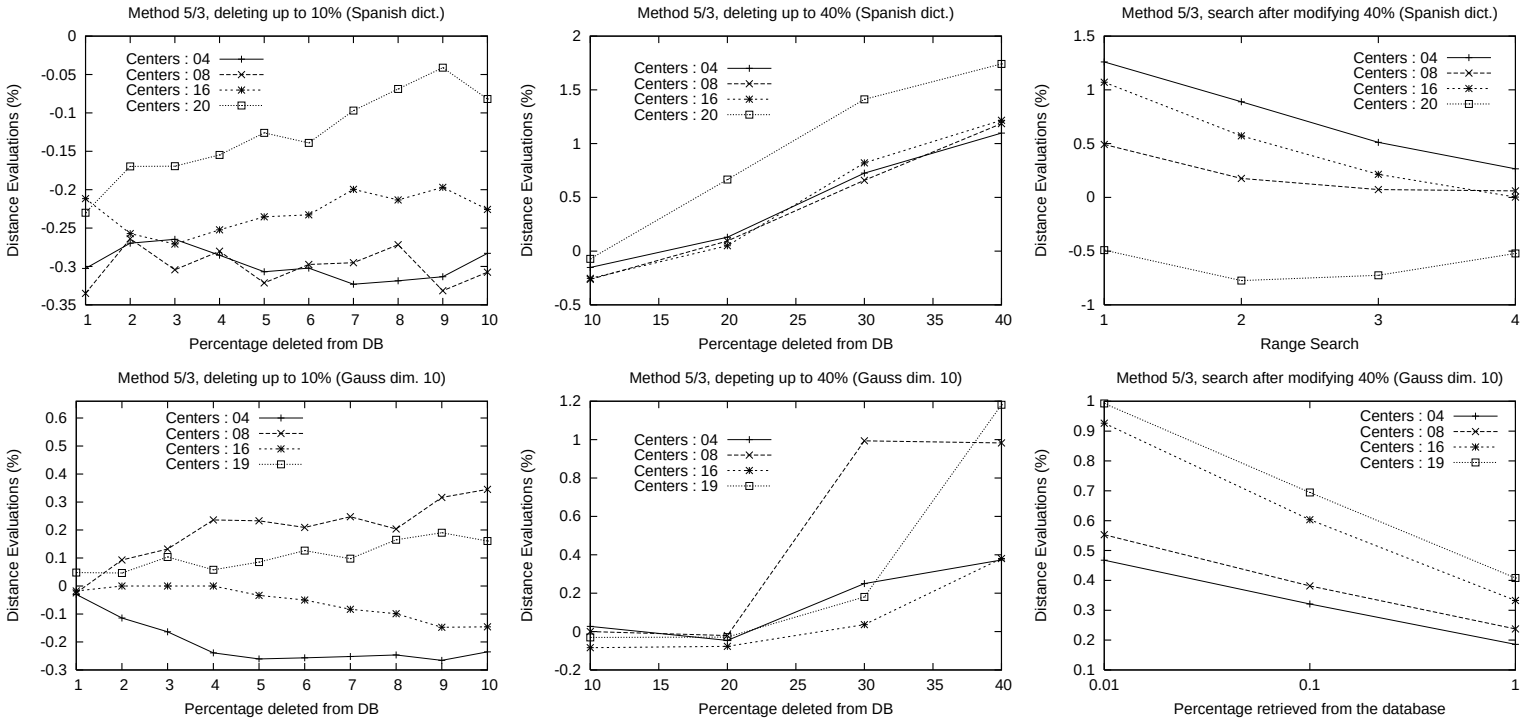
- The nearest neighbor.
- The nearest descendant.
- The nearest descendent located in a leaf.
- A promising descendant leaf.
- An arbitrary descendant leaf.

Constructions Costs and Deletion Costs



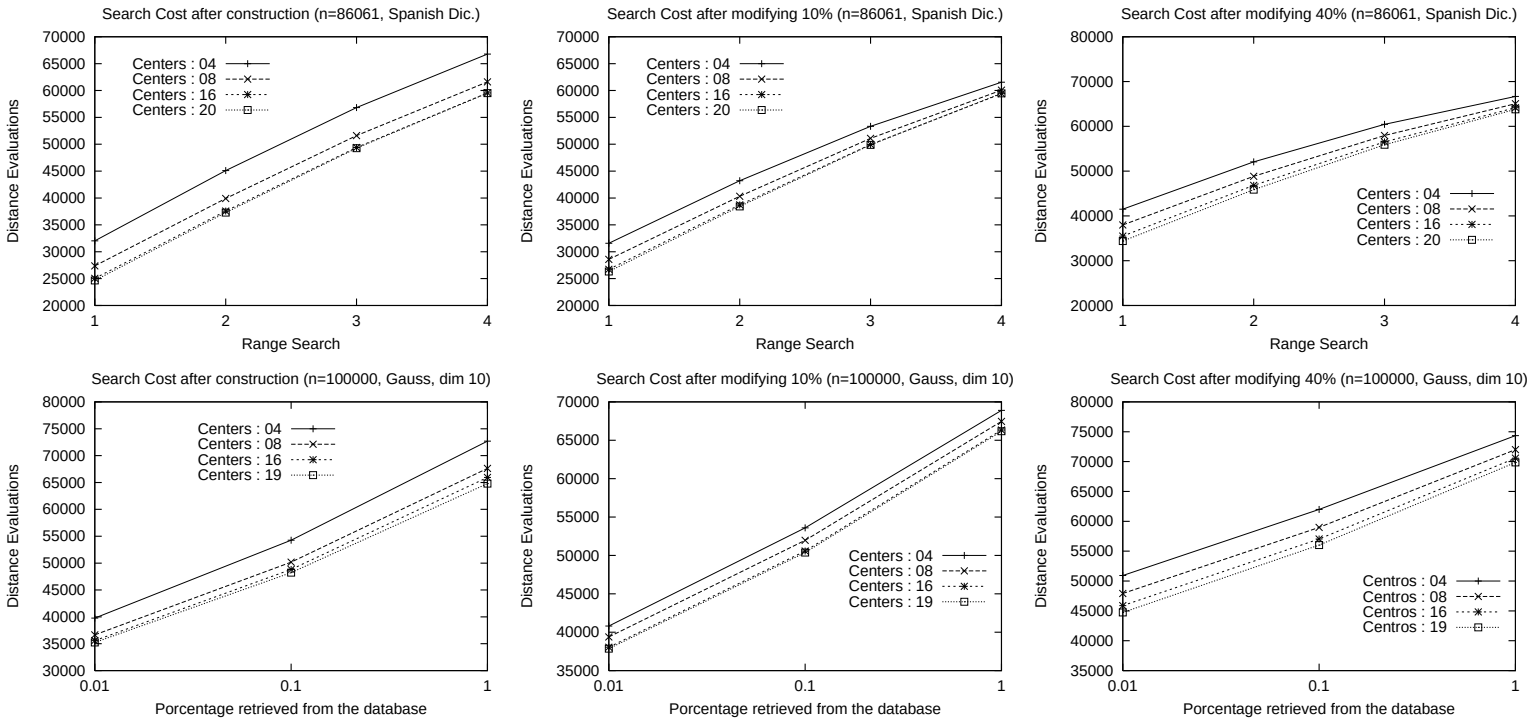
Aggregated construction costs (left) and individual deletion costs when deleting the first 10% (middle) and 40% (right) of the DB.

Comparative Methods



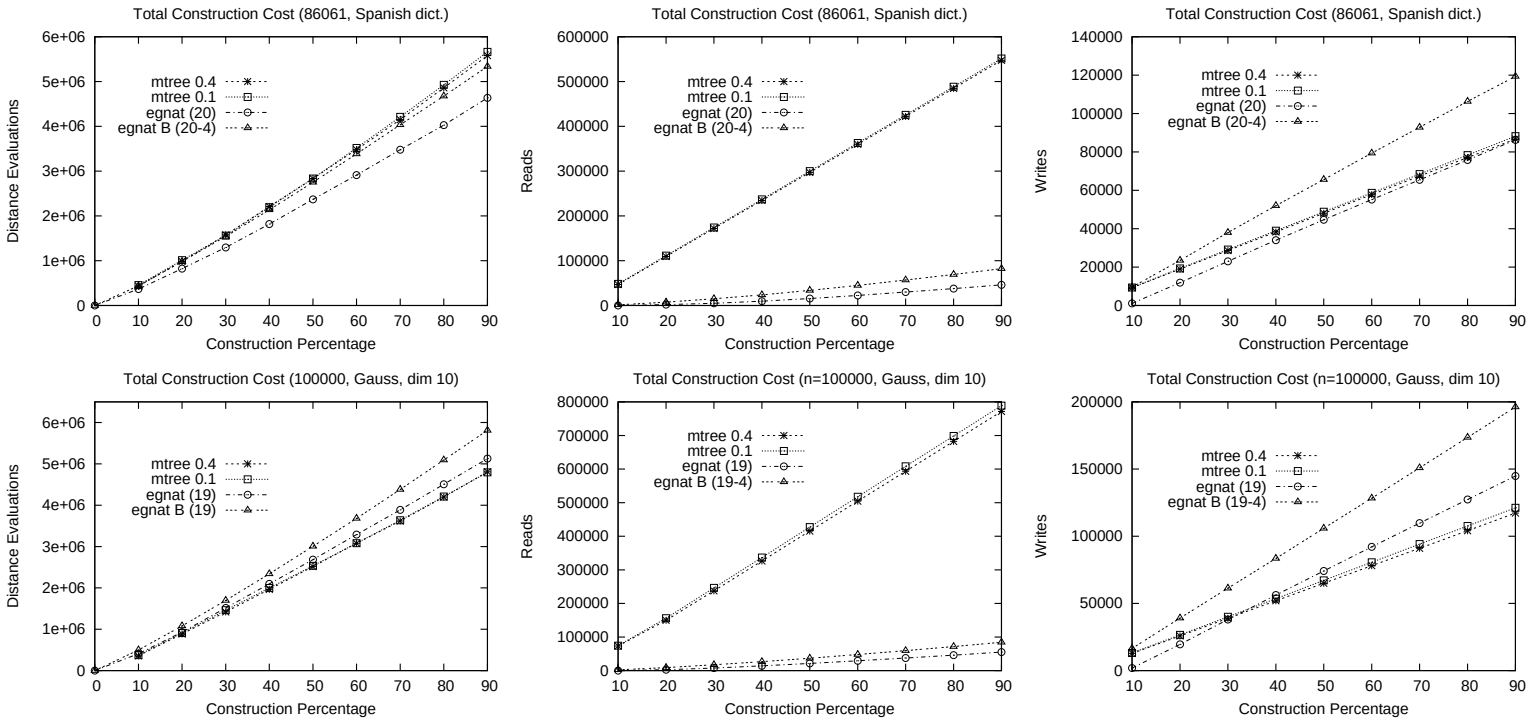
Ratio between methods, in deletion (left, middle) and search (right) costs.

Search Costs



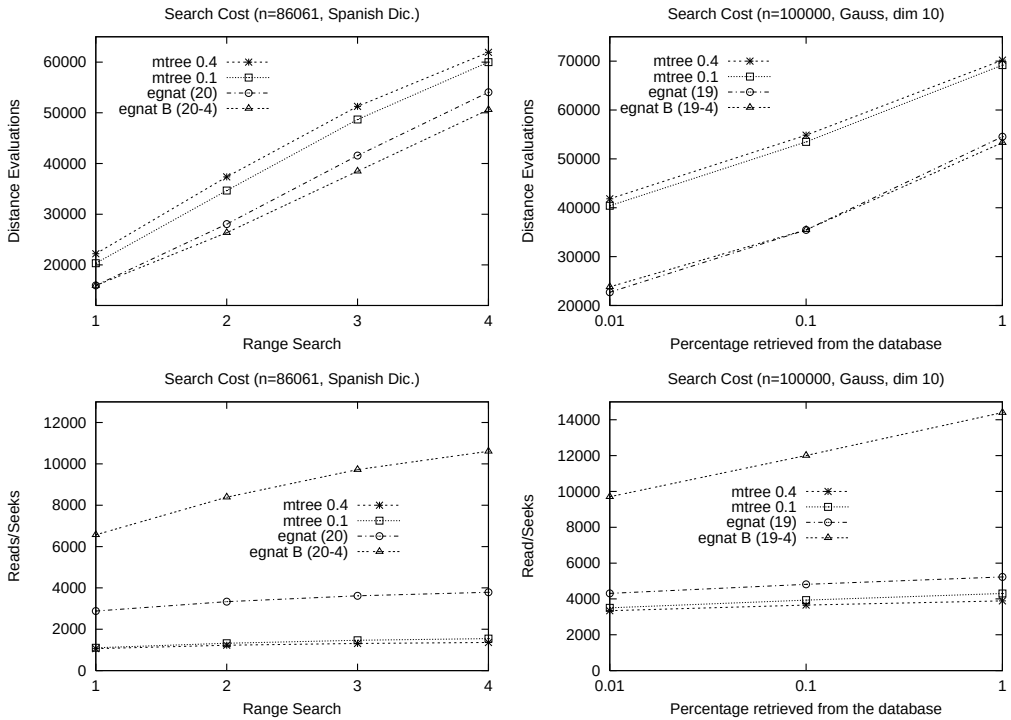
Search costs after construction and after deleting and reinserting part of the database.

Construction Costs on Secondary Memory



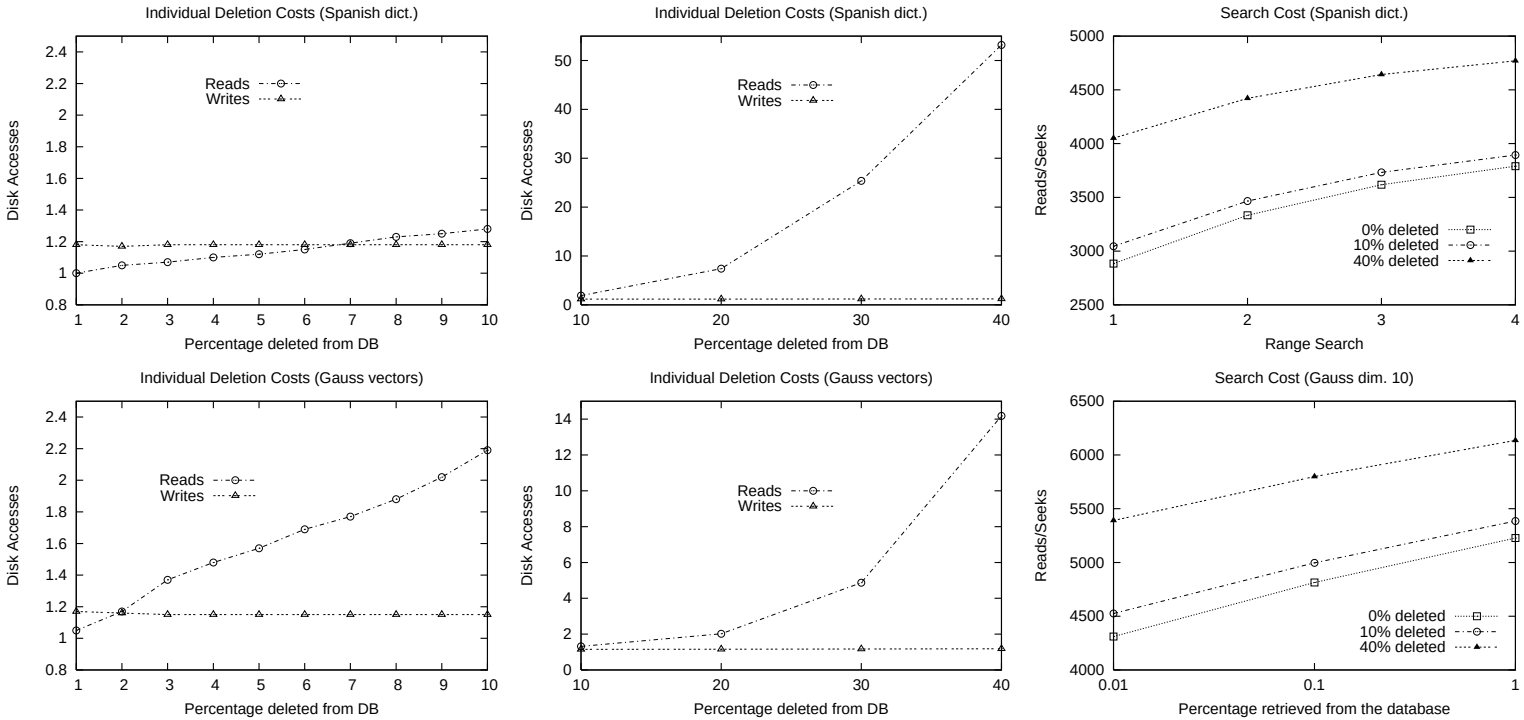
Construction costs of the secondary memory variants: distance computations, disk reads and writes.

Search Costs of the Secondary Memory Variants



Search costs of the secondary memory variants: distance computations and disk reads.

Deletion and Search Costs on Secondary Memory



Deletion costs in secondary memory, deleting 10% (left) and 40% (middle). On the right, disk reads for searching after deletions.

Conclusions

- We have presented a dynamic and secondary-memory-bound data structure based on hyperplane partitioning, *EGNAT*.
- Experimental results show that, as expected, the *M-tree* achieves better disk page usage and consequently fewer I/O operations at search time, whereas our *EGNAT* data structure carries out fewer distance computations.
- We have presented a novel mechanism to handle deletions based on *ghost hyperplanes*.
- The method of ghost hyperplanes is applicable to other similar data structures.