NUEVAS PROPUESTAS PARA BÚSQUEDAS POR SIMILITUD EN BASES DE DATOS MÉTRICAS

Karina Figueroa

Nora Reyes

Verónica Ludueña

Patricia Roggero

Contenido del curso (1/2)

- Conceptos Fundamentales de Espacios Métricos
  - Introducción y motivación
  - Definición de espacios métricos.
  - Funciones de Distancia: propiedades.
  - Tipos de búsquedas pos similitud más comunes.
  - Maldición de la dimensión.
- Índices para Bases de Datos Métricas
  - Taxonomía de los índices
  - Principales referentes de índices basados en particiones compactas.
  - Principales referentes de índices basados en pivotes.
  - Principales referentes de índices basados en permutaciones
  - Índices estáticos y dinámicos. Ejemplos.
  - Índices para memoria secundaria. Ejemplos.
- Algoritmos Exactos y Aproximados
  - Algoritmos Exactos.
  - Algoritmos Aproximados.
  - Medidas de evaluación de calidad de respuesta.

Contenido del curso (2/2)

- Otras operaciones de Interés sobre Bases de Datos Métricas:
  - Join por Similitud, variantes.
  - Algoritmos para Join: con índices y sin índices.
  - Ejemplos de soluciones existentes.
  - Medidas de evaluación de la dimensionalidad.

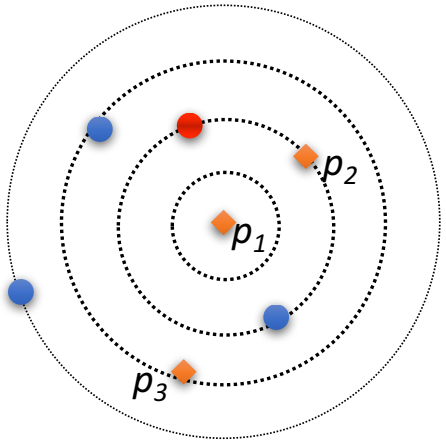
Índices

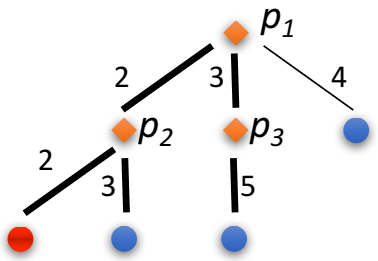
- ¿Bases de datos estáticas o dinámicas?

- Analicemos los índices basados en pivotes:
    - BKT: Burkhard Keller Tree
    - FQT: Fixed Query Tree
    - VPT: Vantage Point Tree
    - AESA: Approximating and Eliminating Searching Algorithm

BKT

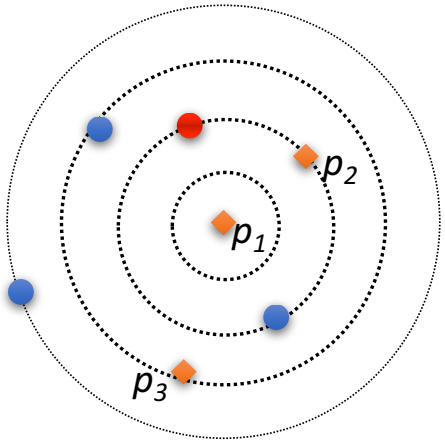
¿Qué ocurriría si inserto un elemento?

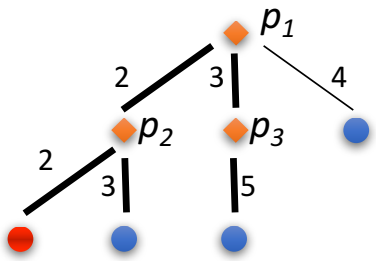




BKT

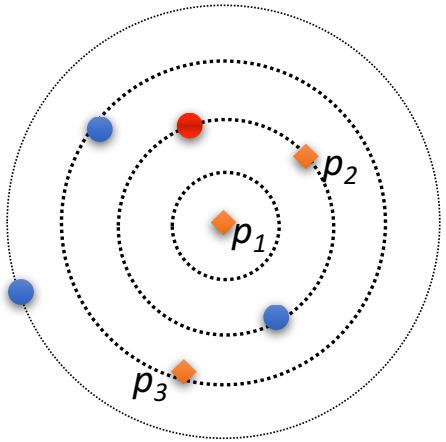
¿Y si elimino un elemento?

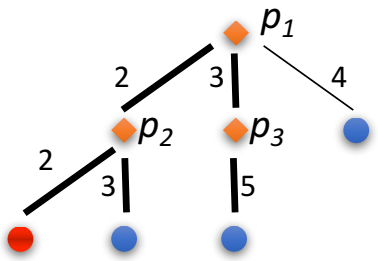




BKT

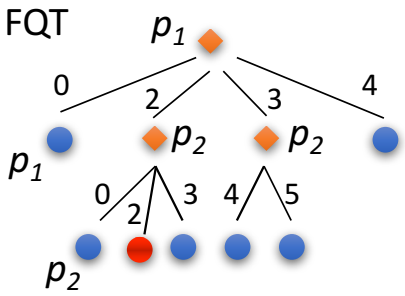
¿Y si elimino un pivote?

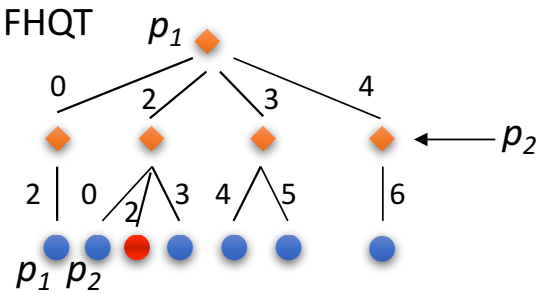


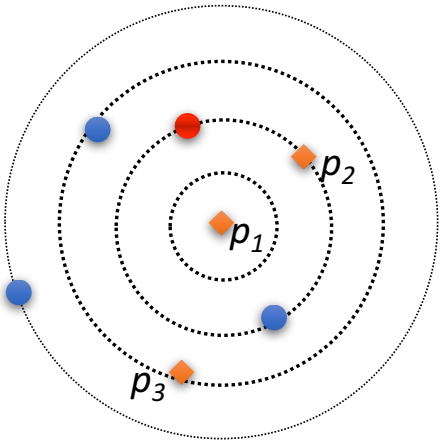


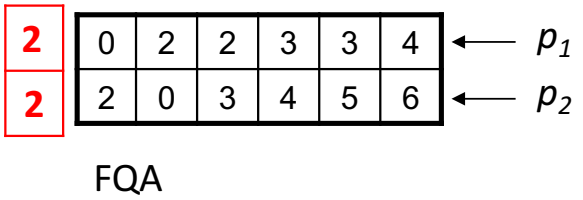
FQT, FHQT, FQA

- ¿Qué pasa si les insertamos un objeto?



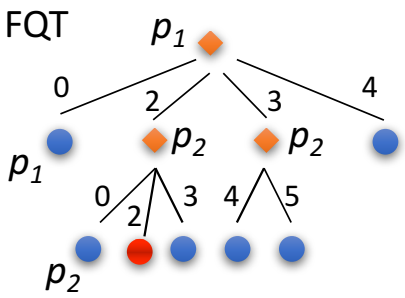


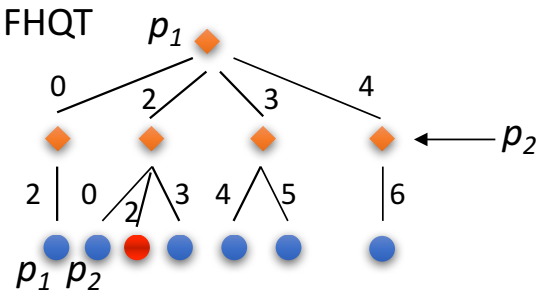


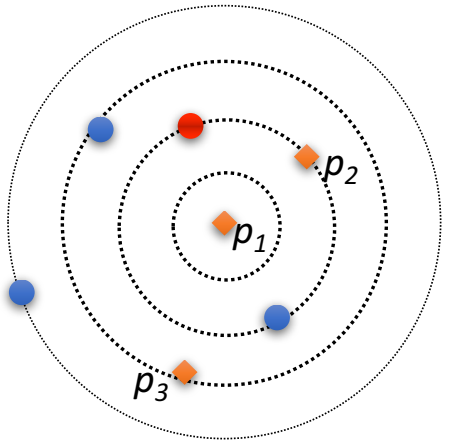


FQT, FHQT, FQA

- ¿Y si les eliminamos un objeto?





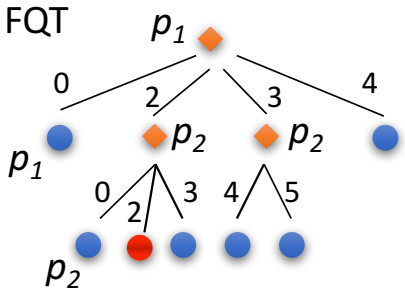


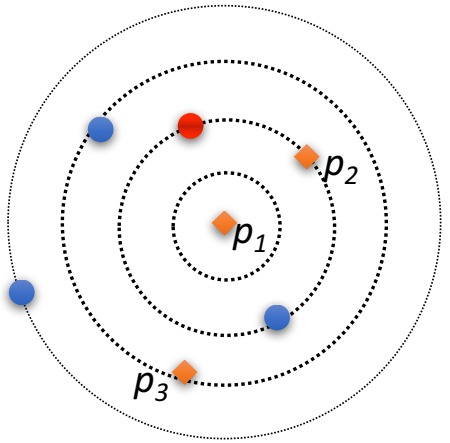
2	0	2	2	3	3	4	← p_1
2	2	0	3	4	5	6	← p_2

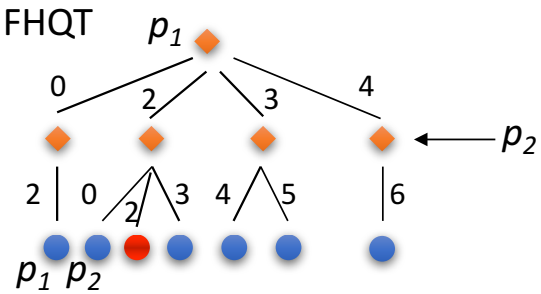
FQA

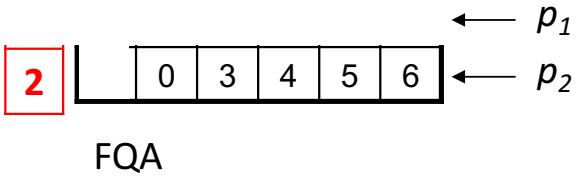
FQT, FHQT, FQA

- ¿Y si les eliminamos un pivote?



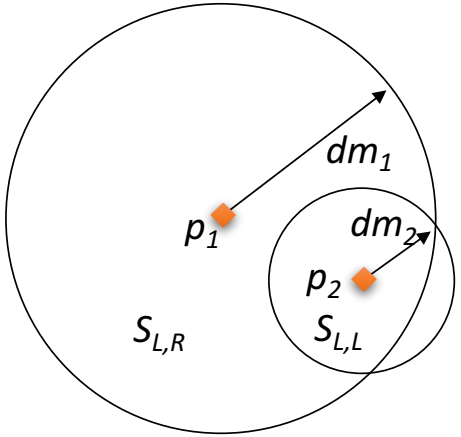


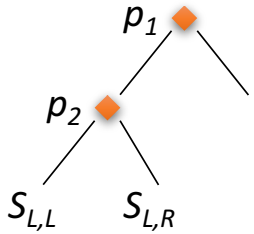


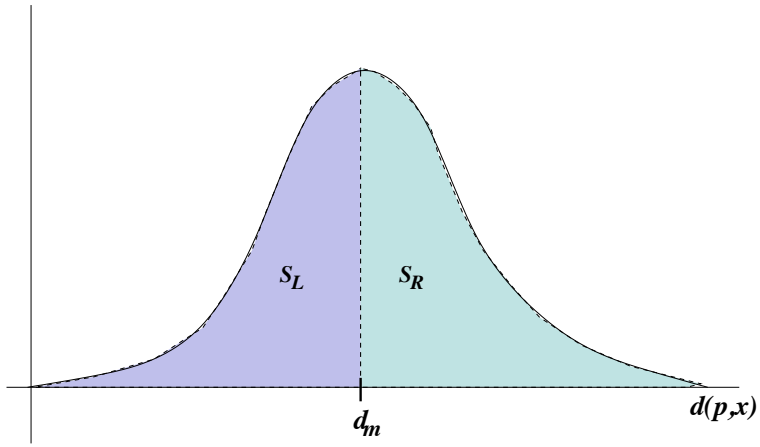


VPT

- ¿Cuán estático es?

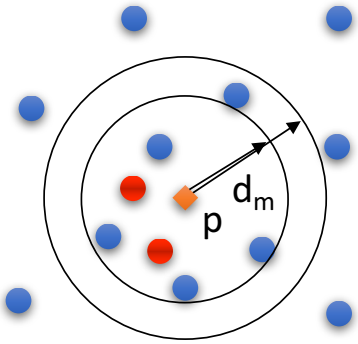


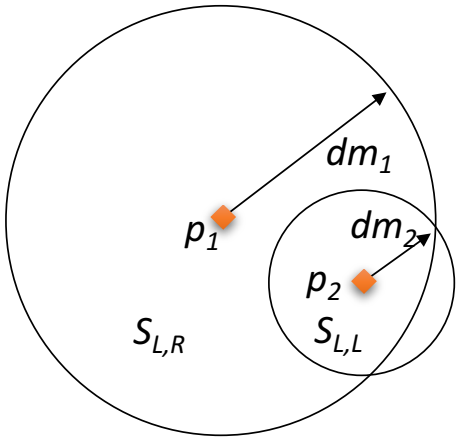


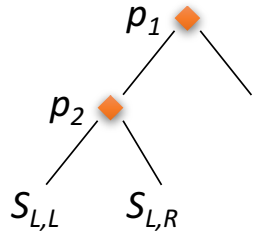


VPT

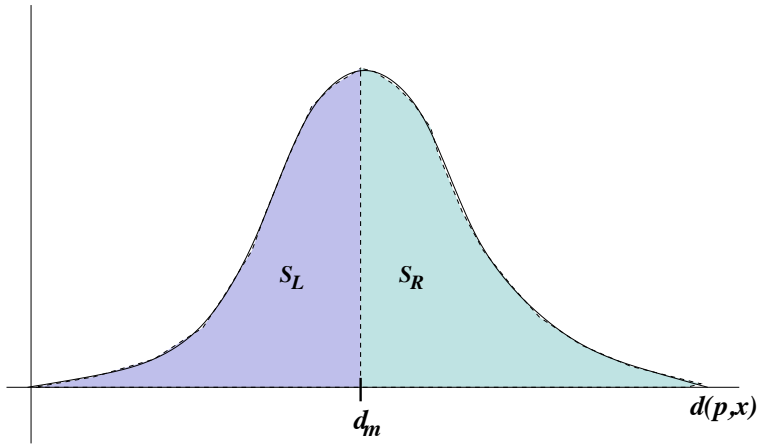
- ¿Cuán estático es?





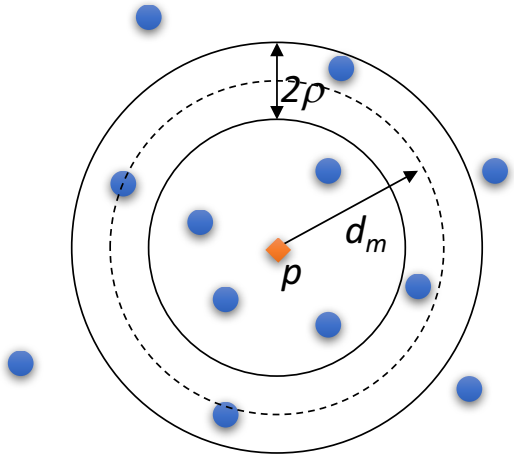


- ¿Cómo serán las eliminaciones?



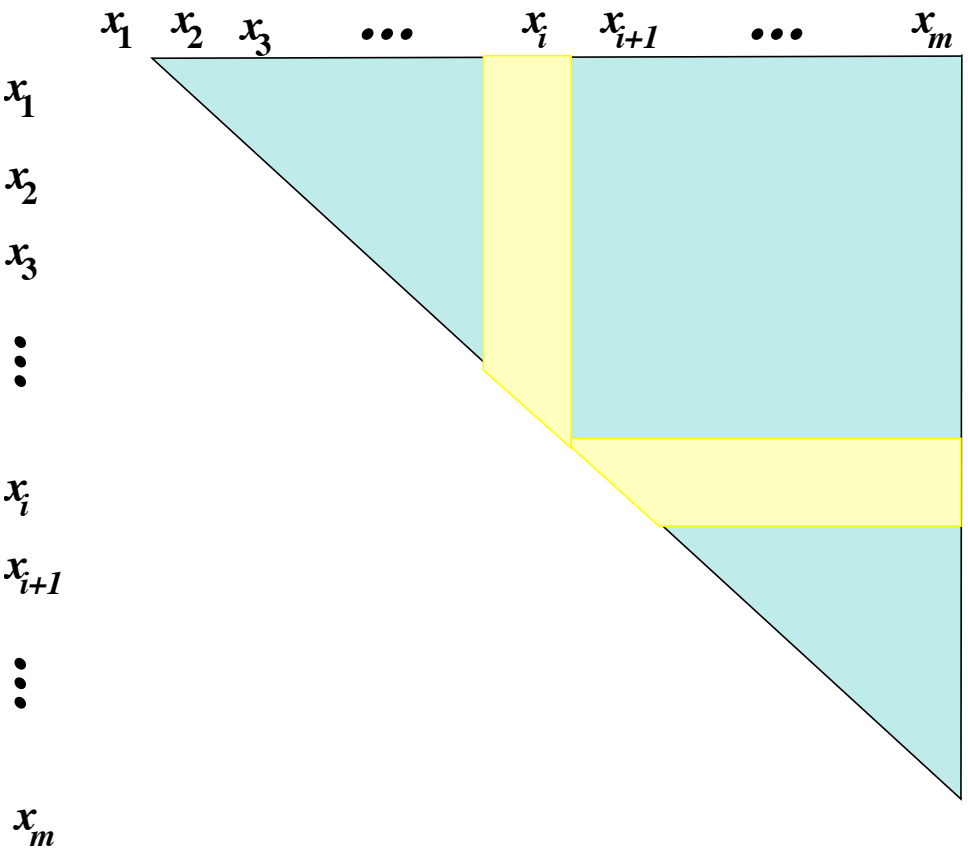
VPF

- ¿Qué pasará con VPF si se admiten inserciones y eliminaciones?



AESA

- ¿Inserciones?
- ¿Eliminaciones?

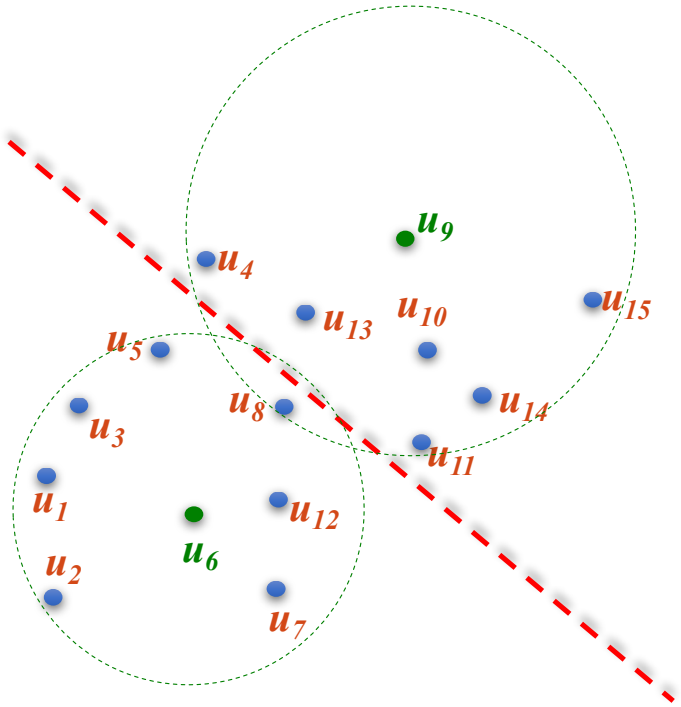


Índices basados en particiones compactas

- Utilizan *centros* y definen las zonas con radios de cobertura, con hiperplanos o con una combinación de ambos.

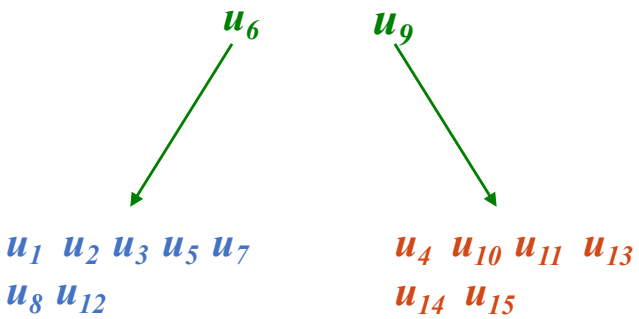
  - BST: particiona por hiperplano y almacena radios de cobertura
  - GHT: particiona por hiperplano y no almacena radios
  - LC: particiona por radios de cobertura y los almacena
  - SAT: particiona por hiperplanos y almacena radios de cobertura

BST



¿Inserciones?

¿Eliminaciones?



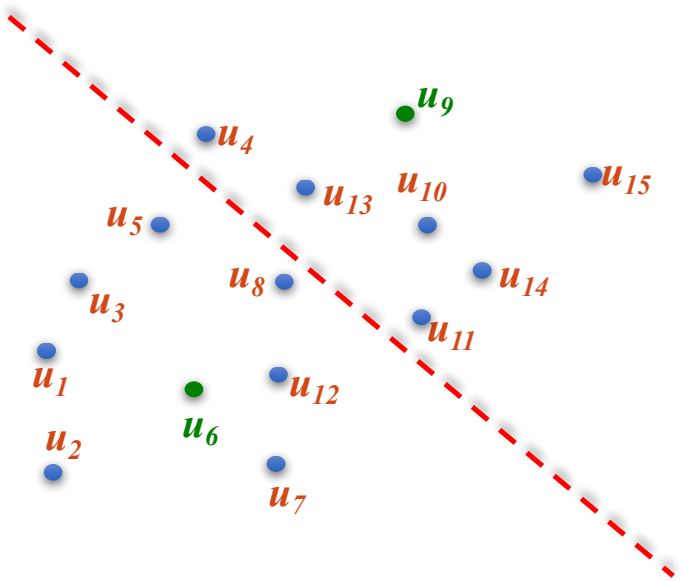
$$rc\ (u_6) = máx\ \{d(u_6, u_i) / u_i \in U_{u_6}\}$$

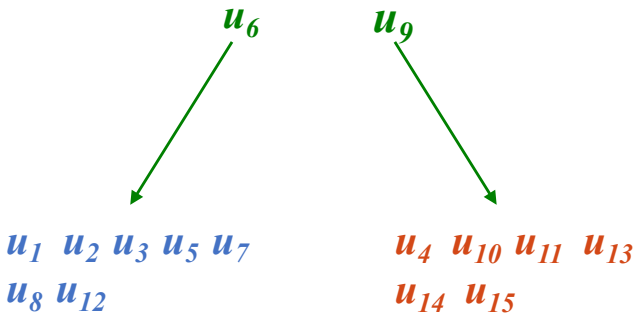
$$rc\ (u_9) = máx\ \{d(u_9, u_i) / u_i \in U_{u_9}\}$$

GHT

¿Inserciones?

¿Eliminaciones?

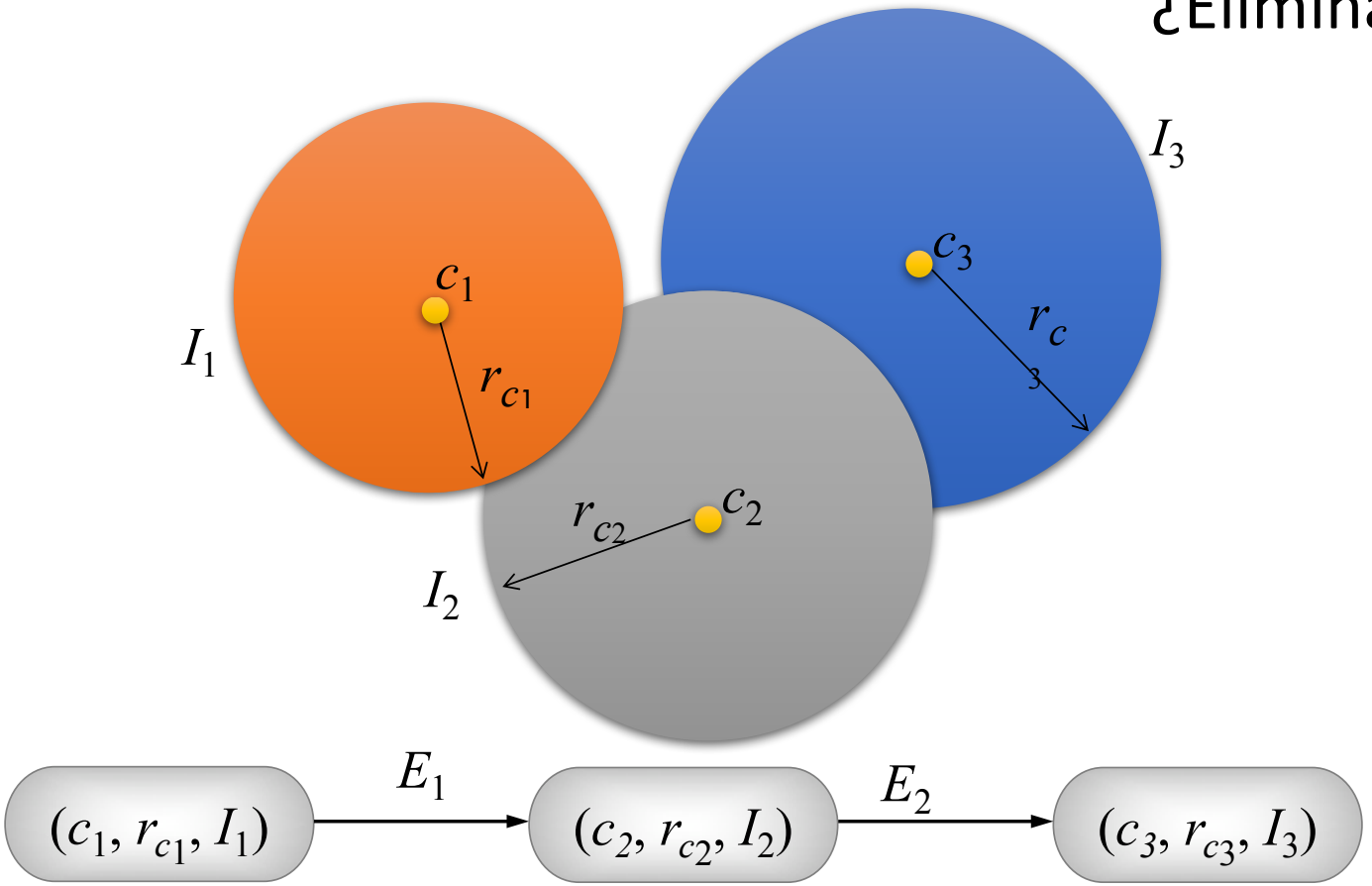




LC

¿Inserciones?

¿Eliminaciones?



M-tree

[Ciaccia, Patella, and Zezula 1997] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: an efficient access method for similarity search in metric spaces. In *Proc. of the 23rd Conference on Very Large Data- bases (VLDB'97)*, pages 426–435, 1997.

M-tree

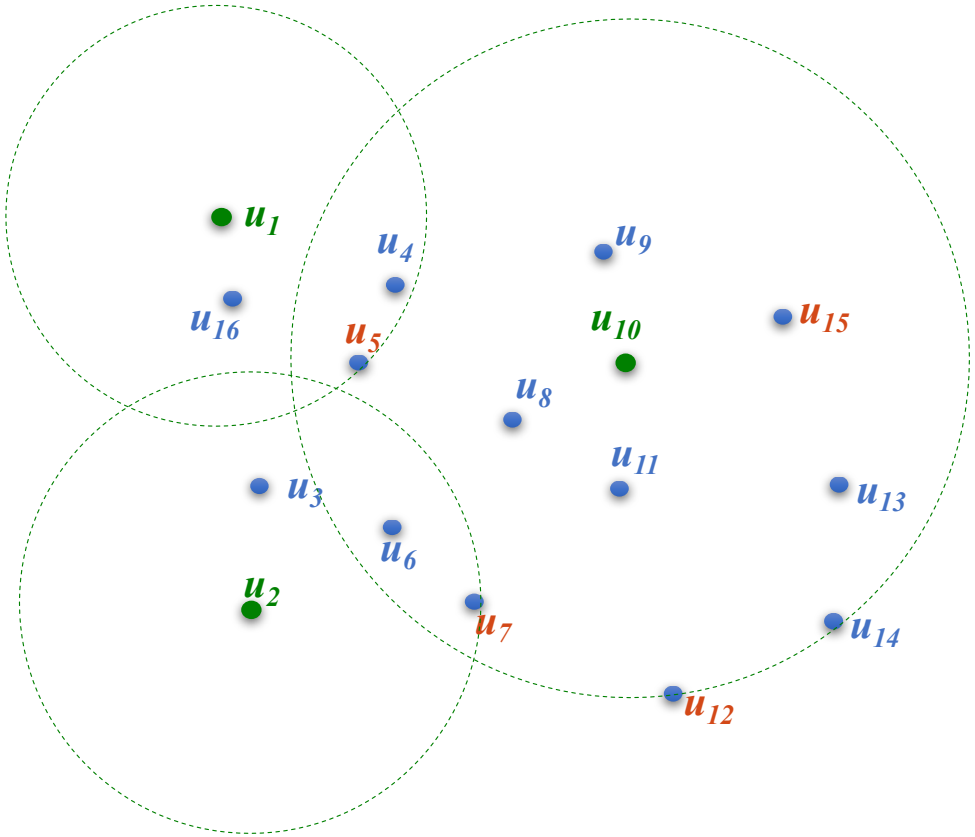
- Admite inserciones y posee buen desempeño en E/S y cálculos de distancia.

- Cada nodo consiste de *m* entradas, cada entrada de un nodo interno es una tupla:
  - *p* es un centro
  - *rc* es el radio de cobertura del subárbol de *p*
  - *d*(*p, P*) es la distancia entre *p* y su centro padre *P*
  - *ptr* es la dirección del nodo hijo de *p*

$$(p, rc, d(p, P), ptr)$$

- Una entrada en un nodo hoja es: $(x, d(x, P))$.

M-tree



Como no trabaja con hiperplanos, por ejemplo u_6 podría ir en la bola de u_2 o en la bola de u_{10}.

M-tree

- Si se conocen de antemano los elementos se puede construir el árbol desde las hojas.

- También se puede crear el árbol incrementalmente (por inserciones).

- Los elementos realmente se encuentran en las hojas del árbol, en los nodos internos los elementos actúan como ***objetos de ruteo***.

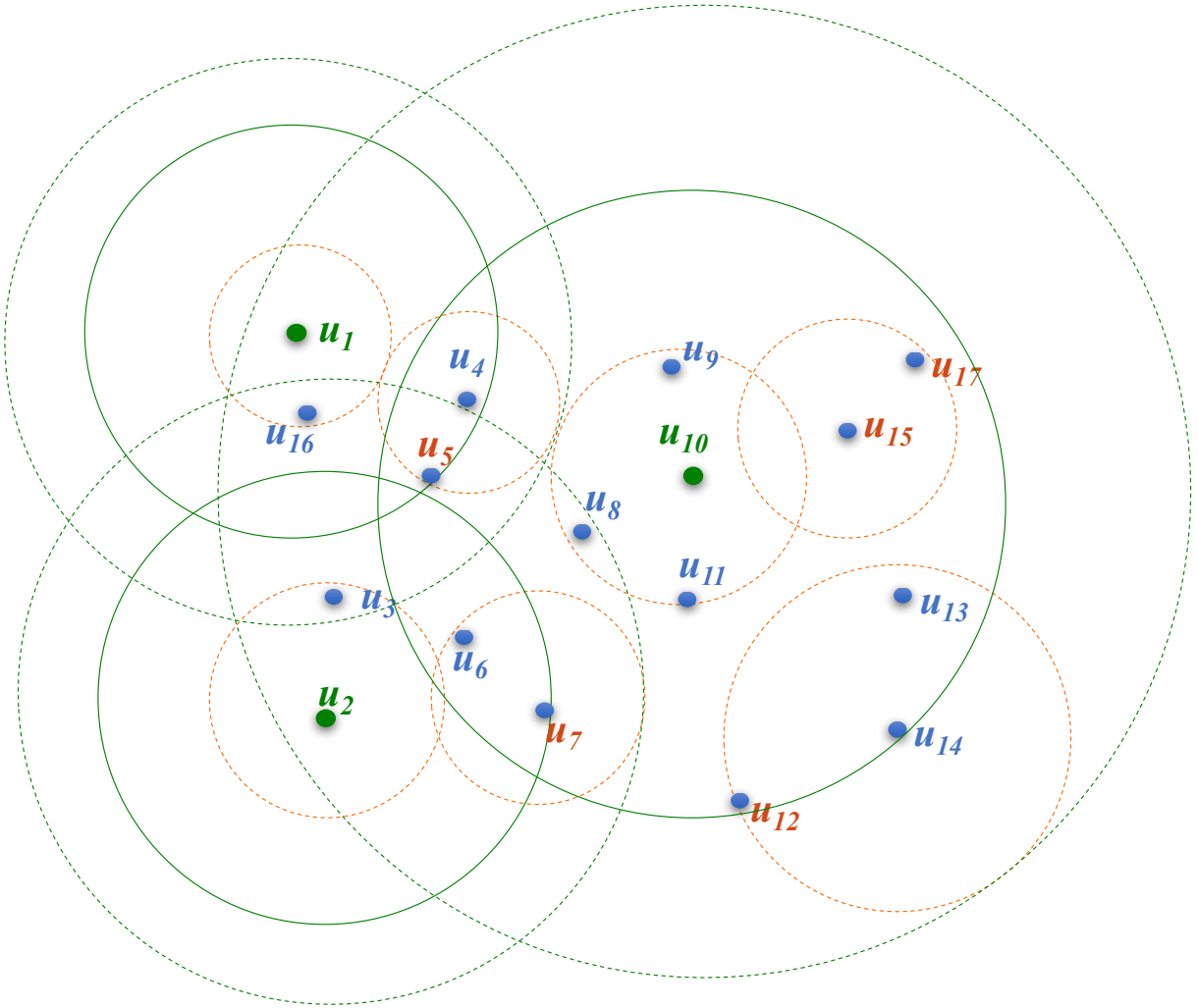
M-tree

- Se inserta un elemento en el “mejor” subárbol y se lo agrega en una hoja; si la hoja rebalsa, se la divide en dos (split) y se promueve un elemento al nodo padre (B-tree y R-tree).

- El árbol crece desde las hojas hacia la raíz.

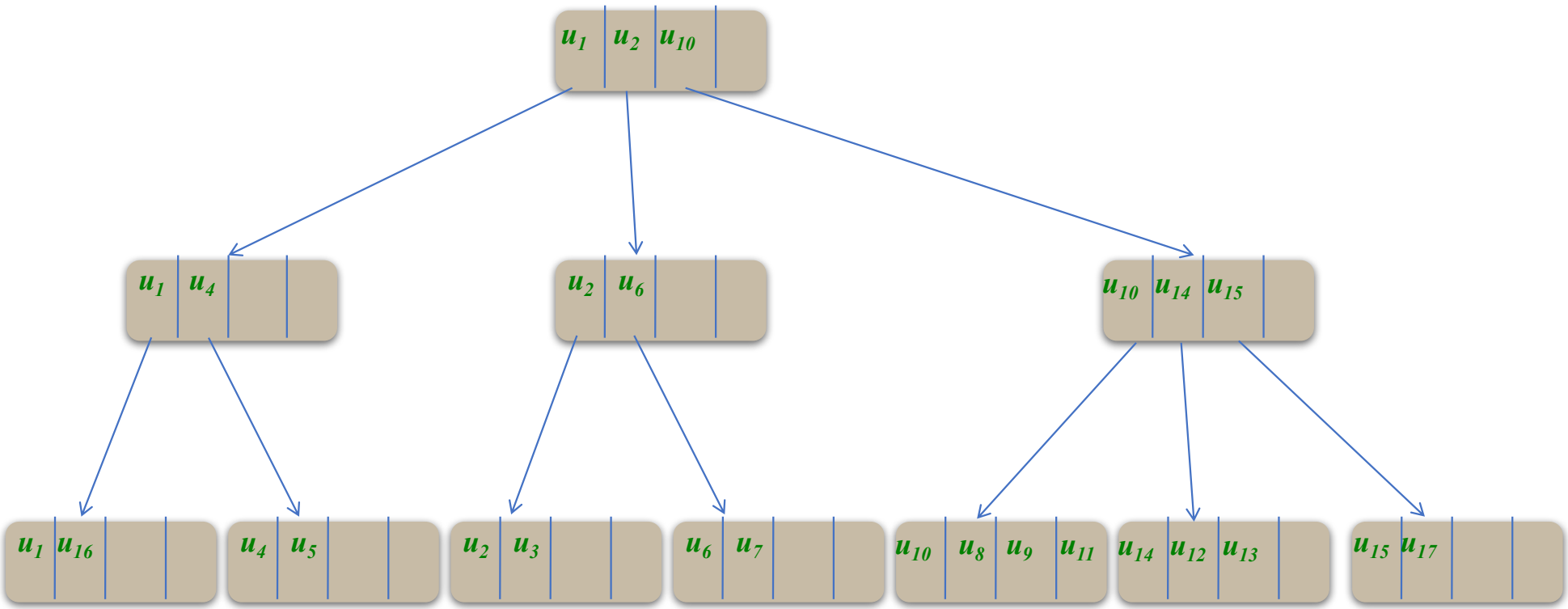
- Es un árbol balanceado.

M-tree

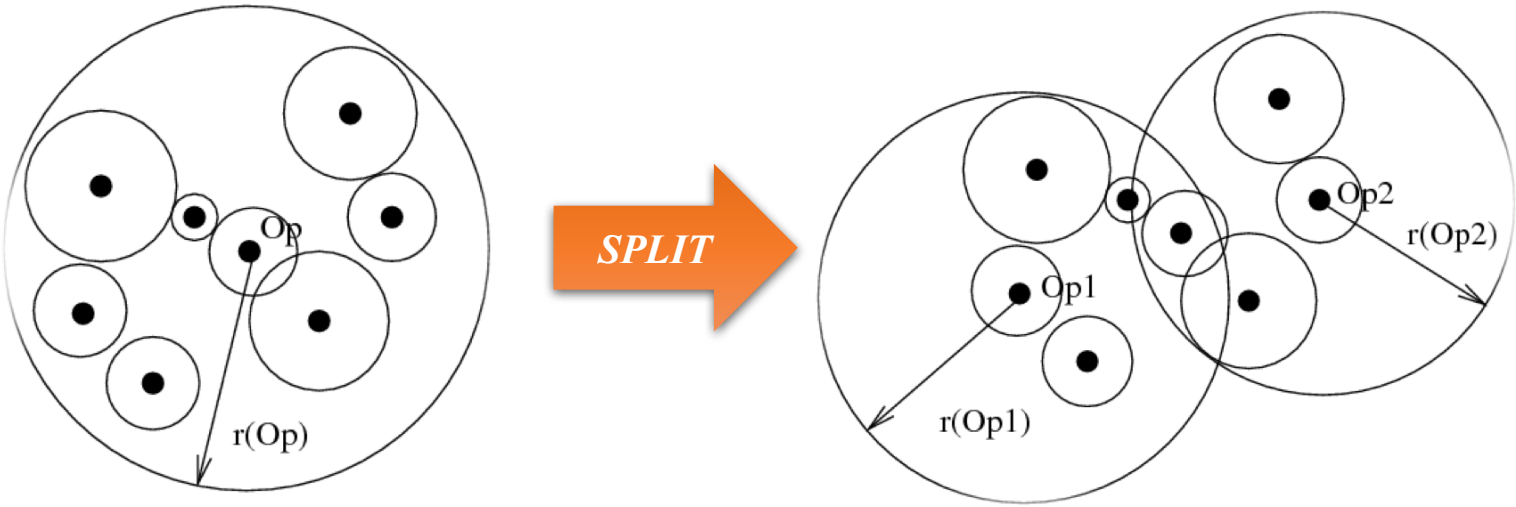


$m = 4$

M-tree


u_1 u_2 u_{10}
u_1 u_4
u_2 u_6
u_{10} u_{14} u_{15}
u_1 u_{16}
u_4 u_5
u_2 u_3
u_6 u_7
u_{10} u_8 u_9 u_{11}
u_{14} u_{12} u_{13}
u_{15} u_{17}

M-tree



Existen distintas políticas de división y distintas heurísticas para definir los nuevos centros.

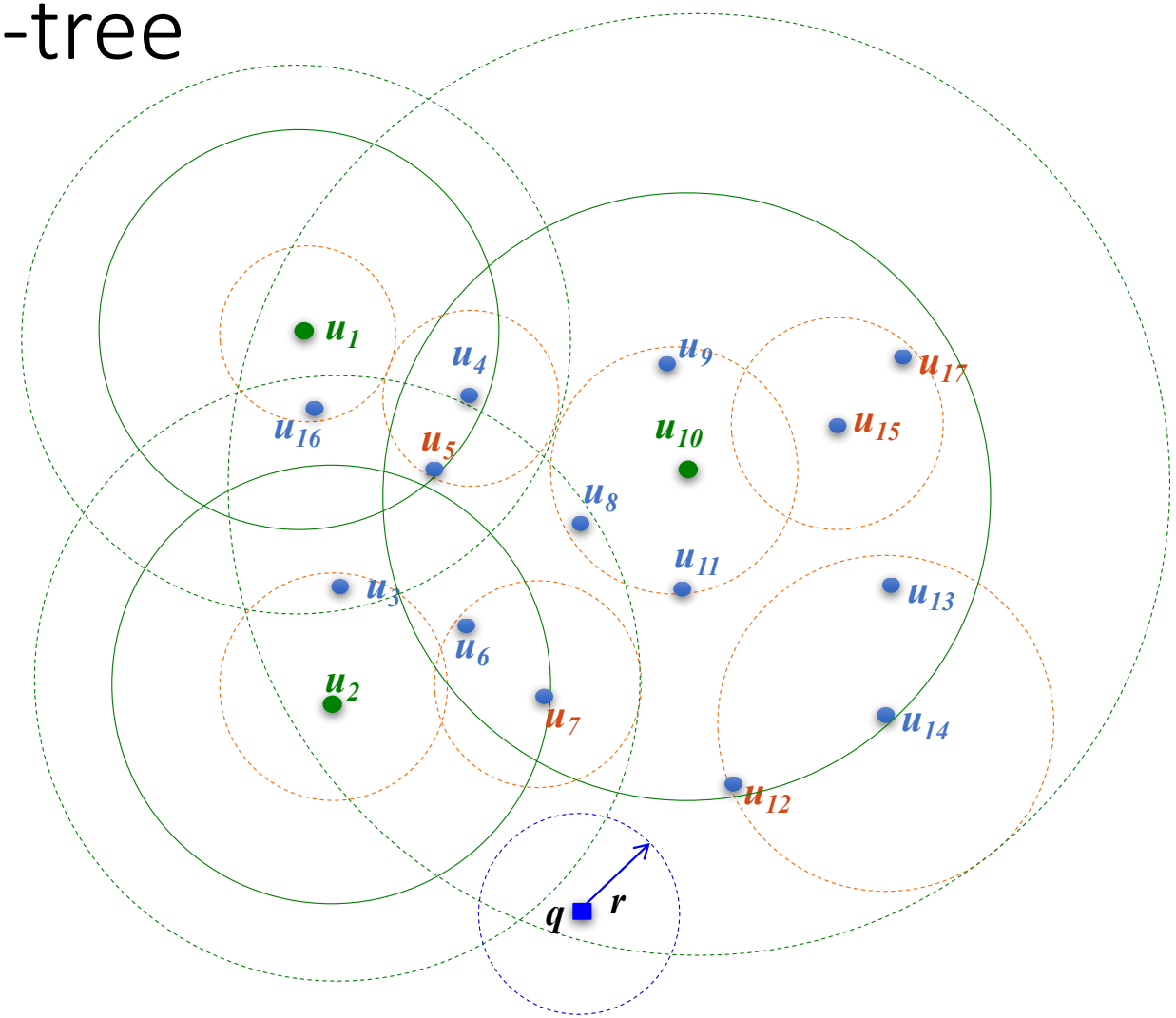
M-tree

- Para una consulta (*q*, *r*) se compara *q* contra los representantes de cada nodo y se ingresa en los que no se descartan con el criterio de radio de cobertura.

- No es necesario entrar al subárbol de *p* si: $\mid d(q,P) - d(p,P) \mid -r > rc$

- Si $\mid d(q,P) - d(p,P) \mid -r \leq rc$ se calcula *d*(*q*, *p*) y entonces se poda usando: $d(q,p) - r > rc$

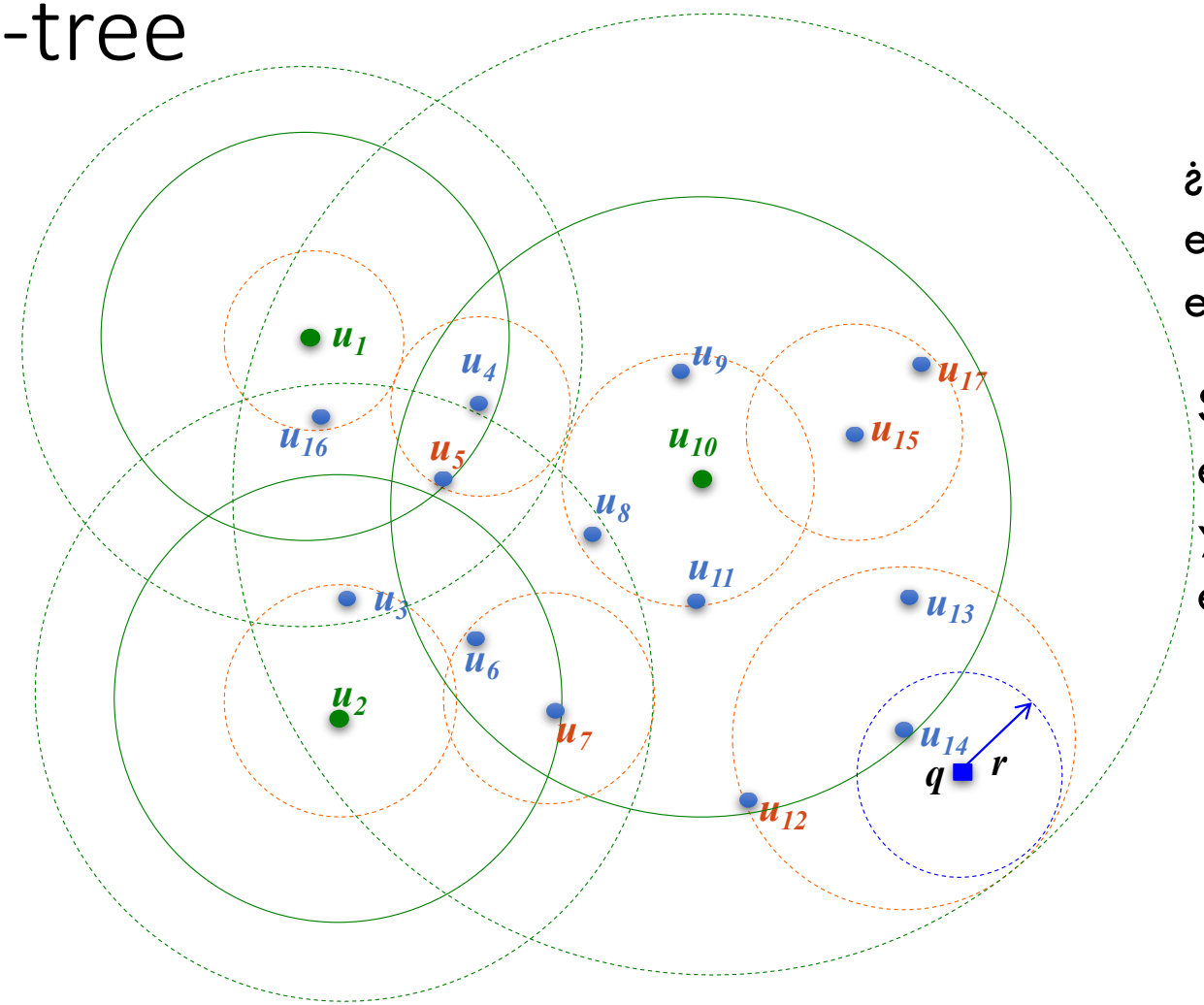
M-tree



¿En dónde debe entrar a buscar en este caso?

Se debe entrar en los subárboles de u_{10} y u_2, aunque de haber registrado sus reales radios de cobertura se hubieran evitado.

M-tree



¿En dónde debe entrar a buscar en este caso?

Se debe entrar en el subárbol de u_{10} y luego solamente en el de u_{14}.

D-Index

[Dohnal, Gennaro, Savino, Zezula 2003] Vlatislav Dohnal, Claudio Gennaro, Pasquale Savino, and Pavel Zezula. D-index: Distan- ce searching index for metric data sets. *Multimedia Tools and Applications*, 21(1):9–33, 2003.

D-Index

- Es una estructura multinivel, con “buckets” de búsqueda separable en cada nivel.

- Intenta que en las búsquedas se acceda a lo sumo a un bucket en cada nivel para las consultas por rango con $r < \rho$.

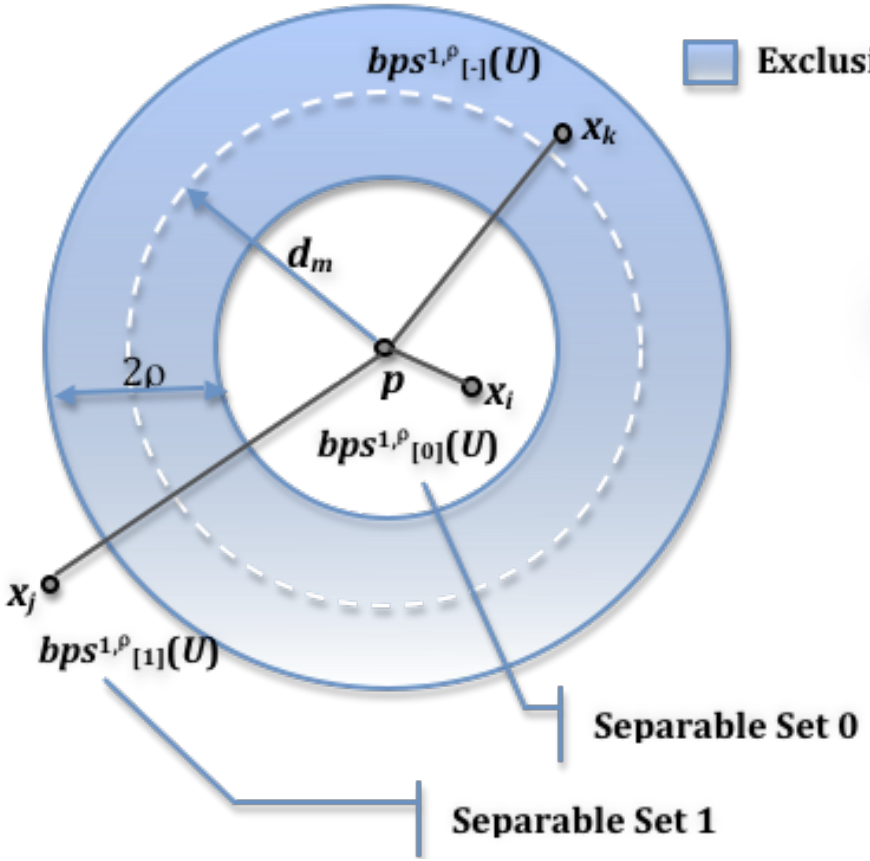
- Aplica pivotes para reducir trabajo en los buckets accedidos.

D-Index

- Particiona *X* usando una función ρ-split *bps*, similar a la que usa *VP-forest*.
- Cada función *bps* usa un objeto de referencia *p* y la distancia media d_m para particionar:

$$bps(x) = \begin{cases} 0 & \text{si } d(p,x) \leq d_m - \rho \\ 1 & \text{si } d(p,x) > d_m + \rho \\ - & \text{en otro caso} \end{cases}$$

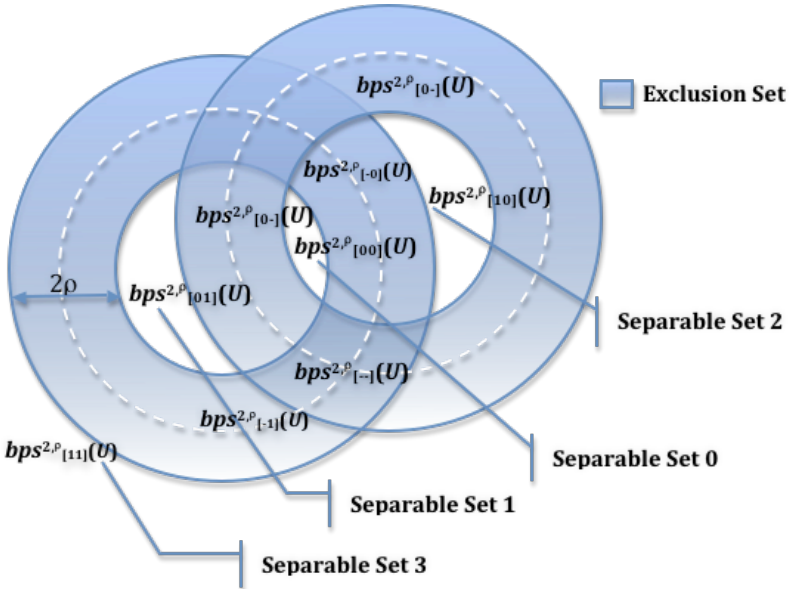
D-Index

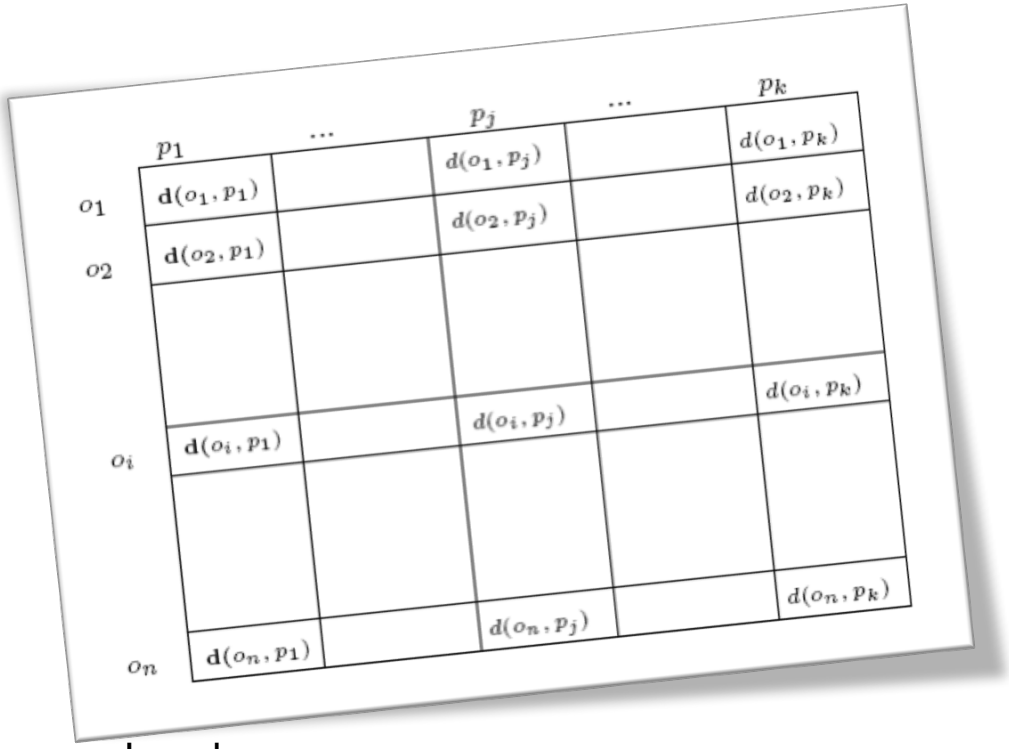


$$bps(x) = \begin{cases} 0 & \text{si } d(p,x) \leq d_m - \rho \\ 1 & \text{si } d(p,x) > d_m + \rho \\ - & \text{en otro caso} \end{cases}$$

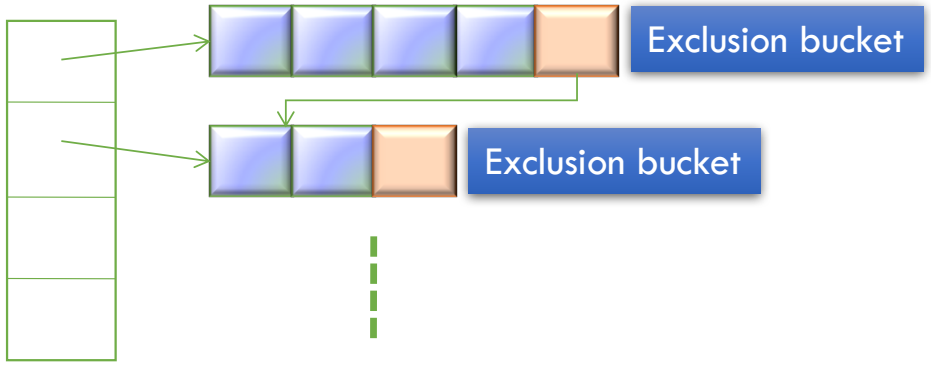
Se obtienen más conjuntos separables por combinación de funciones *bps*.
La exclusión es la unión de las zonas de exclusión.

D-Index





Levels



D-Index

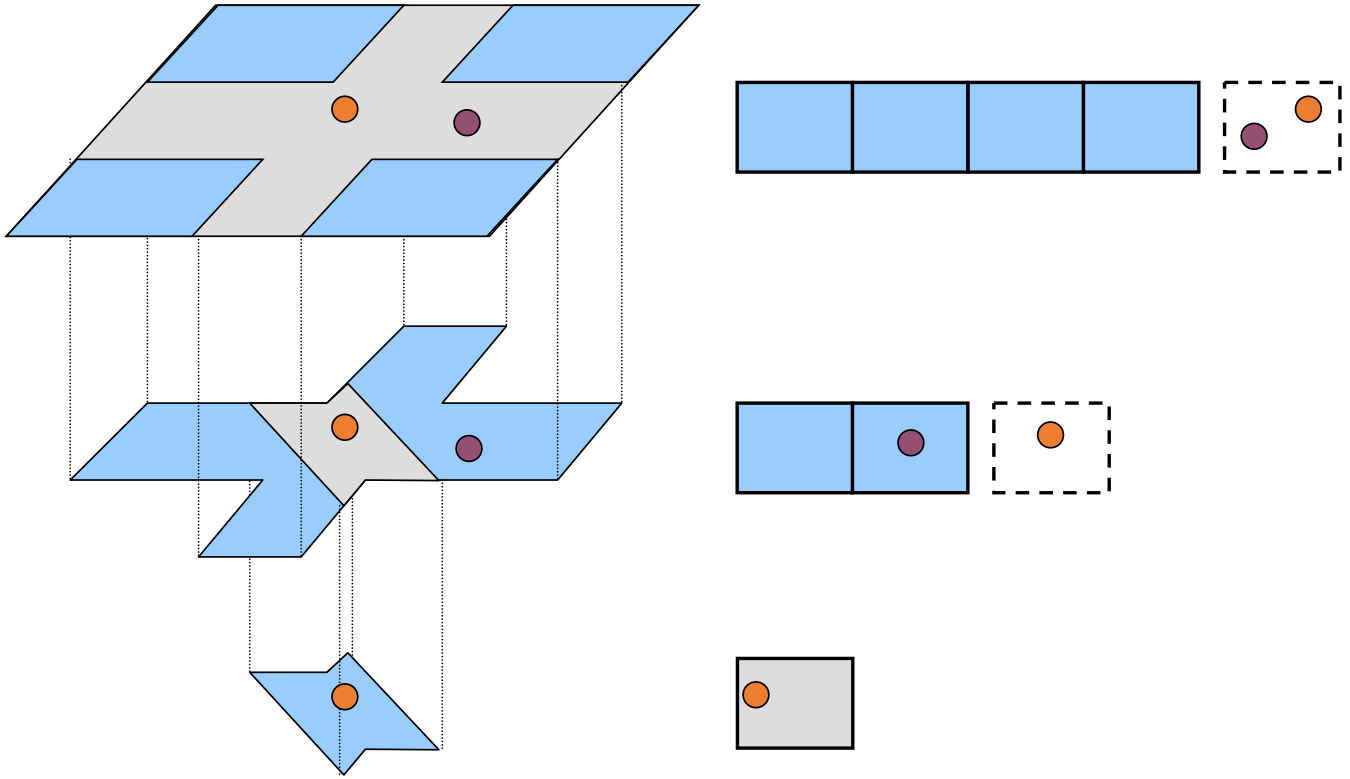
- Los conjuntos separables y el bucket de exclusión forman los buckets de un nivel.

- Para el bucket de exclusión se aplica un nuevo conjunto de funciones split, haciendo así otro nivel.

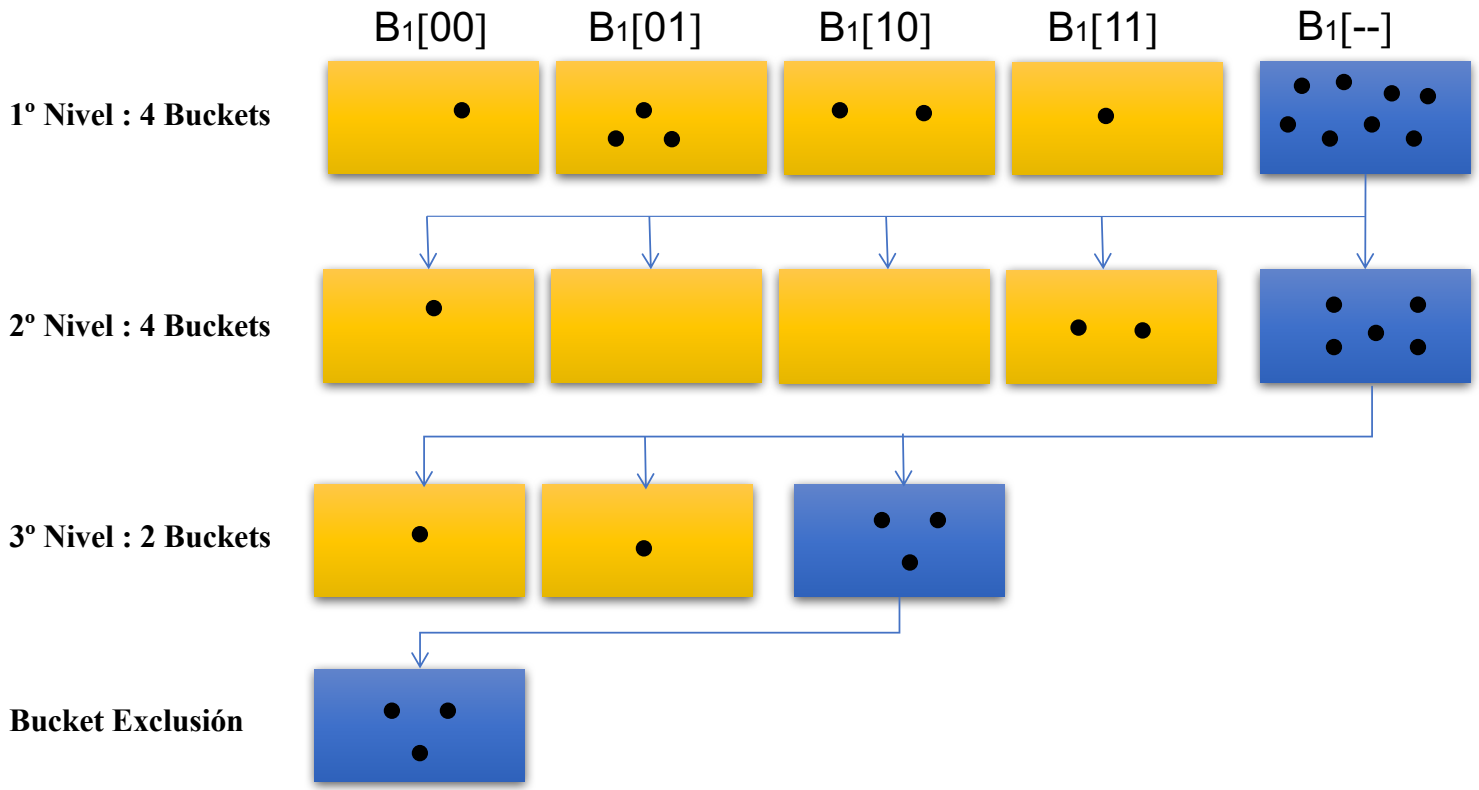
- El bucket de exclusión del último nivel es la exclusión de la estructura completa.

P. Zezula, G. Amato, V. Dohnal, M. Batko:
Similarity Search: The Metric Space Approach

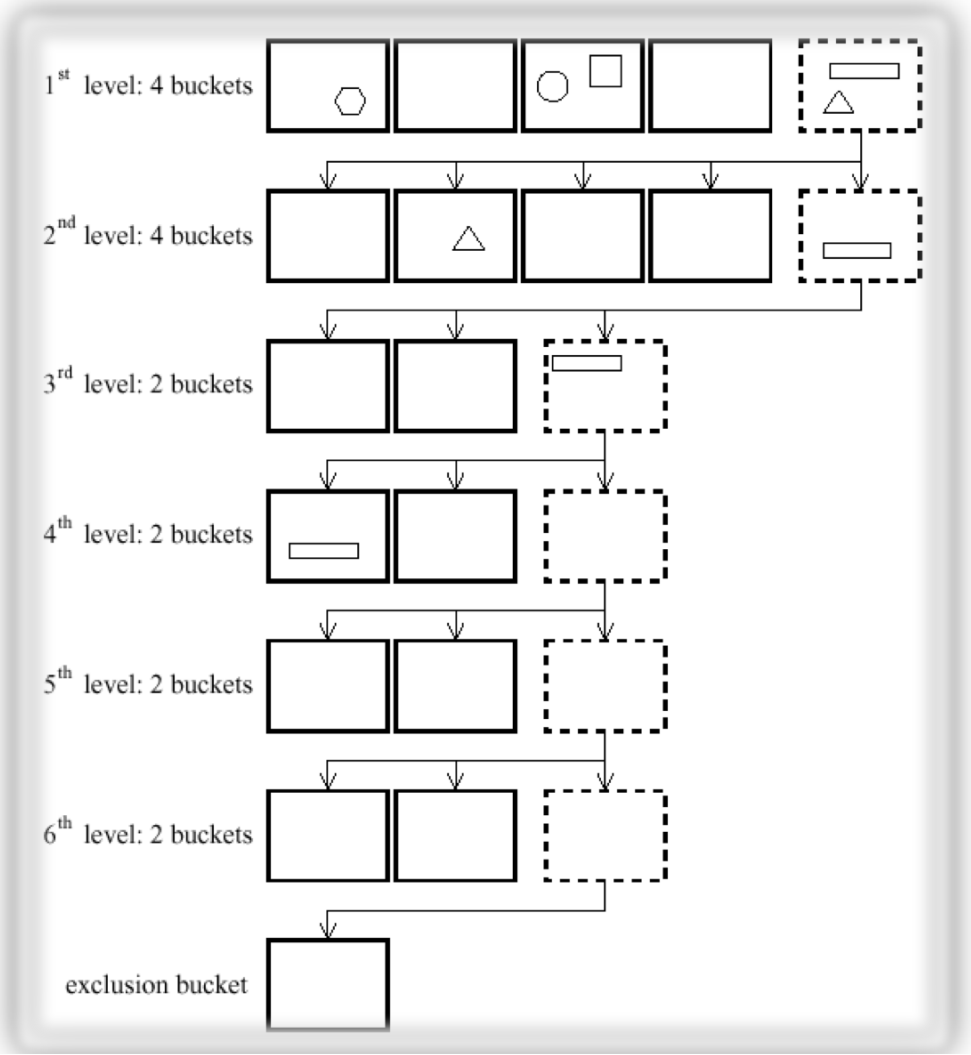
D-Index



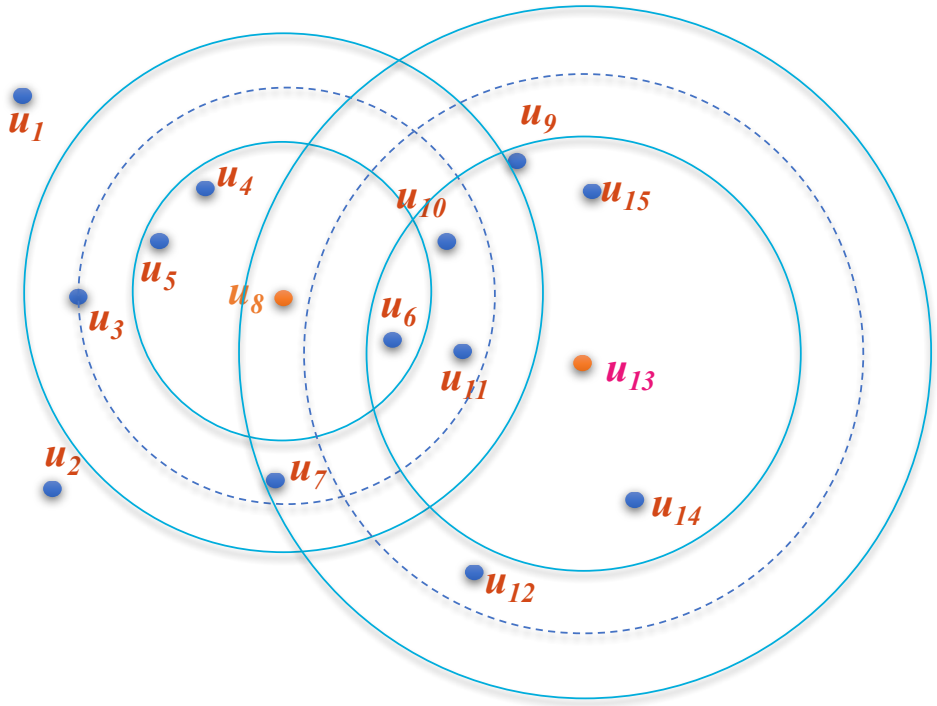
D-Index


B1[00]
B1[01]
B1[10]
B1[11]
B1[--]
1º Nivel : 4 Buckets
2º Nivel : 4 Buckets
3º Nivel : 2 Buckets
Bucket Exclusión

D-Index

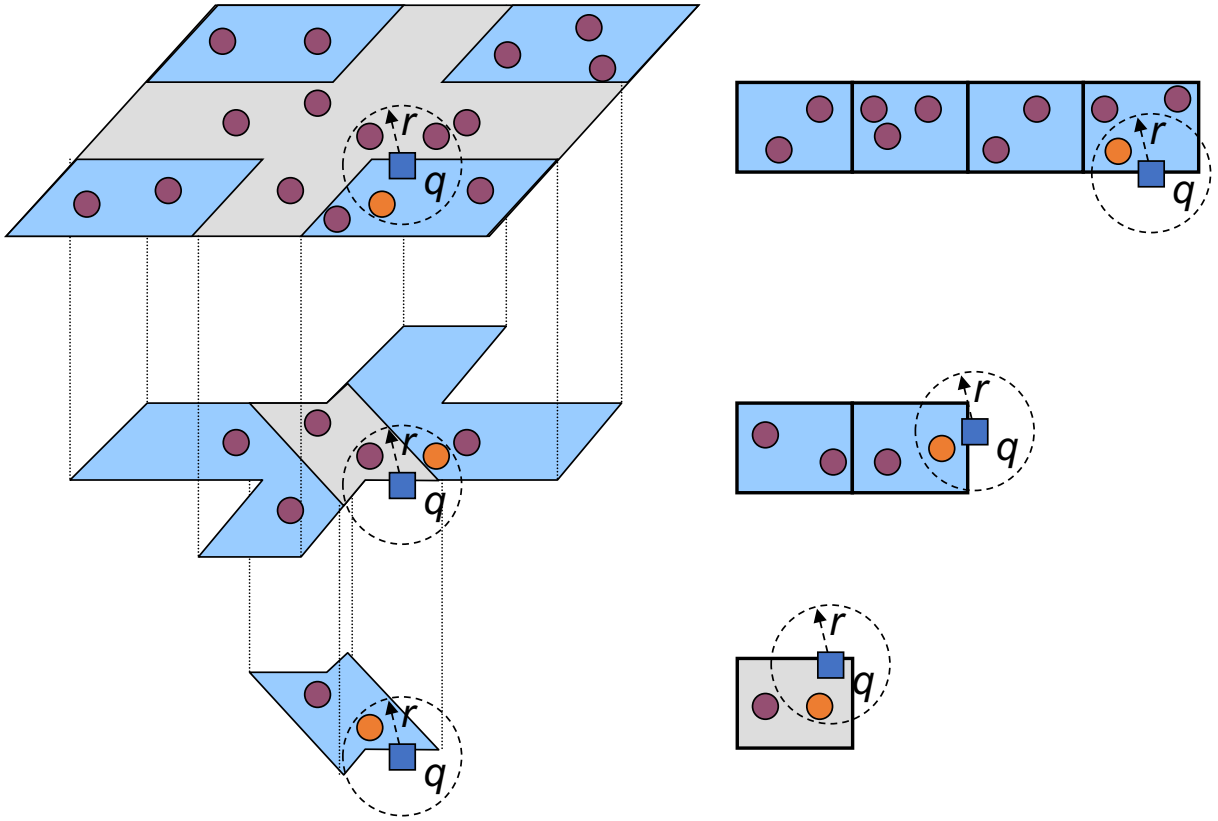

1st level: 4 buckets
2nd level: 4 buckets
3rd level: 2 buckets
4th level: 2 buckets
5th level: 2 buckets
6th level: 2 buckets
exclusion bucket

D-Index

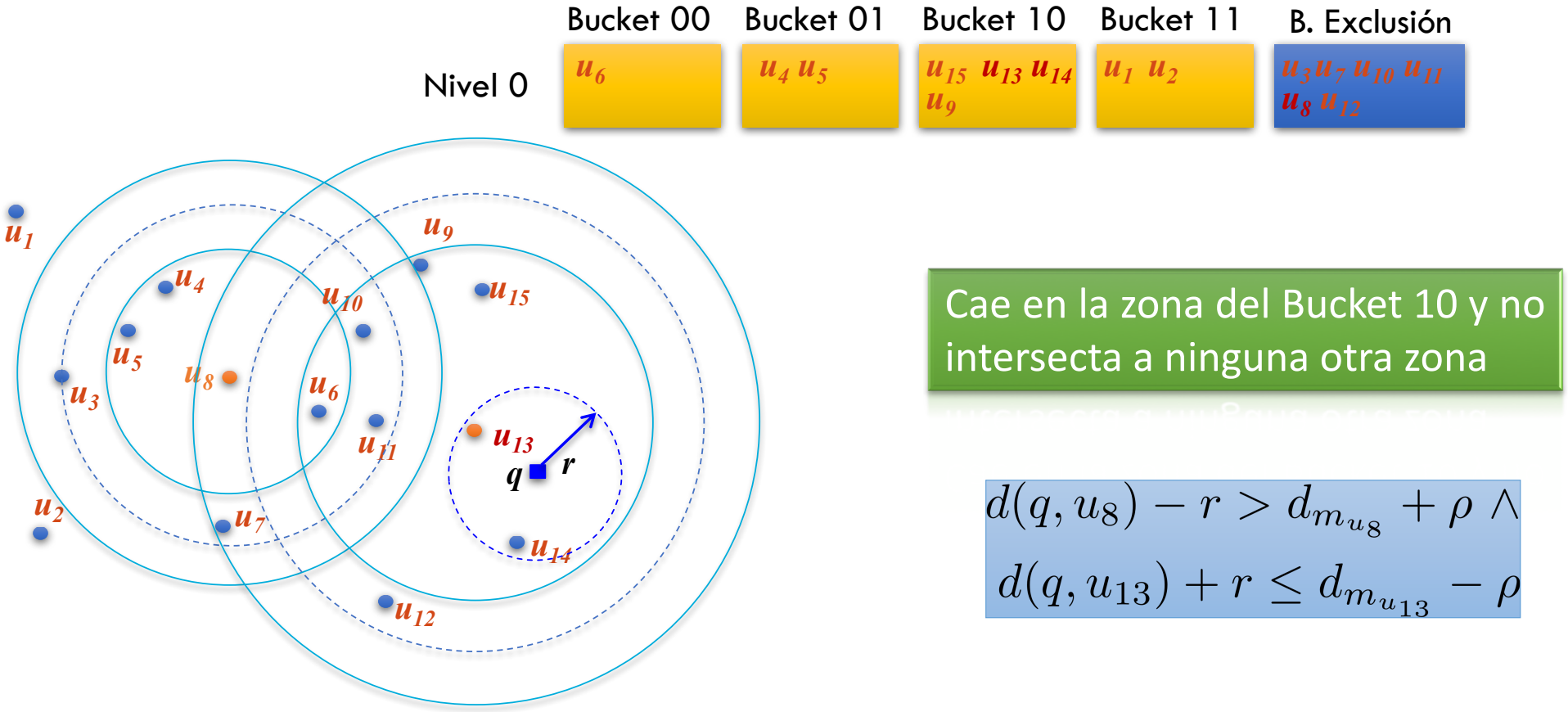


Con una nueva combinación de funciones *bps* se indexan los elementos del bucket de exclusión.

D-Index



D-Index

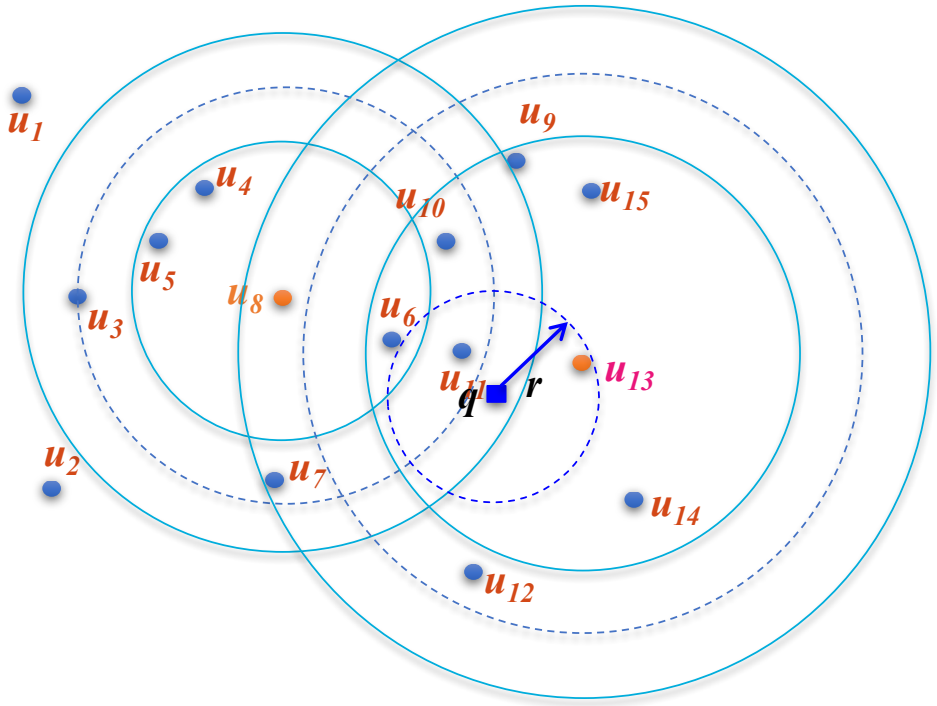


Cae en la zona del Bucket 10 y no intersecta a ninguna otra zona

$$d(q, u_8) - r > d_{m_{u_8}} + \rho \wedge$$
$$d(q, u_{13}) + r \leq d_{m_{u_{13}}} - \rho$$

D-Index

	Bucket 00	Bucket 01	Bucket 10	Bucket 11	B. Exclusión
Nivel 0	u_6	$u_4\ u_5$	$u_{15}\ u_{13}\ u_{14}\ u_9$	$u_1\ u_2$	$u_3\ u_7\ u_{10}\ u_{11}\ u_8\ u_{12}$



Intersecta la zona del Bucket 00, del Bucket 10 y también la zona de exclusión

$$d(q, u_8) - r \leq d_{m_{u_8}} - \rho \wedge$$
$$d(q, u_8) + r > d_{m_{u_8}} + \rho \wedge$$
$$d(q, u_{13}) + r \leq d_{m_{u_{13}}} - \rho$$

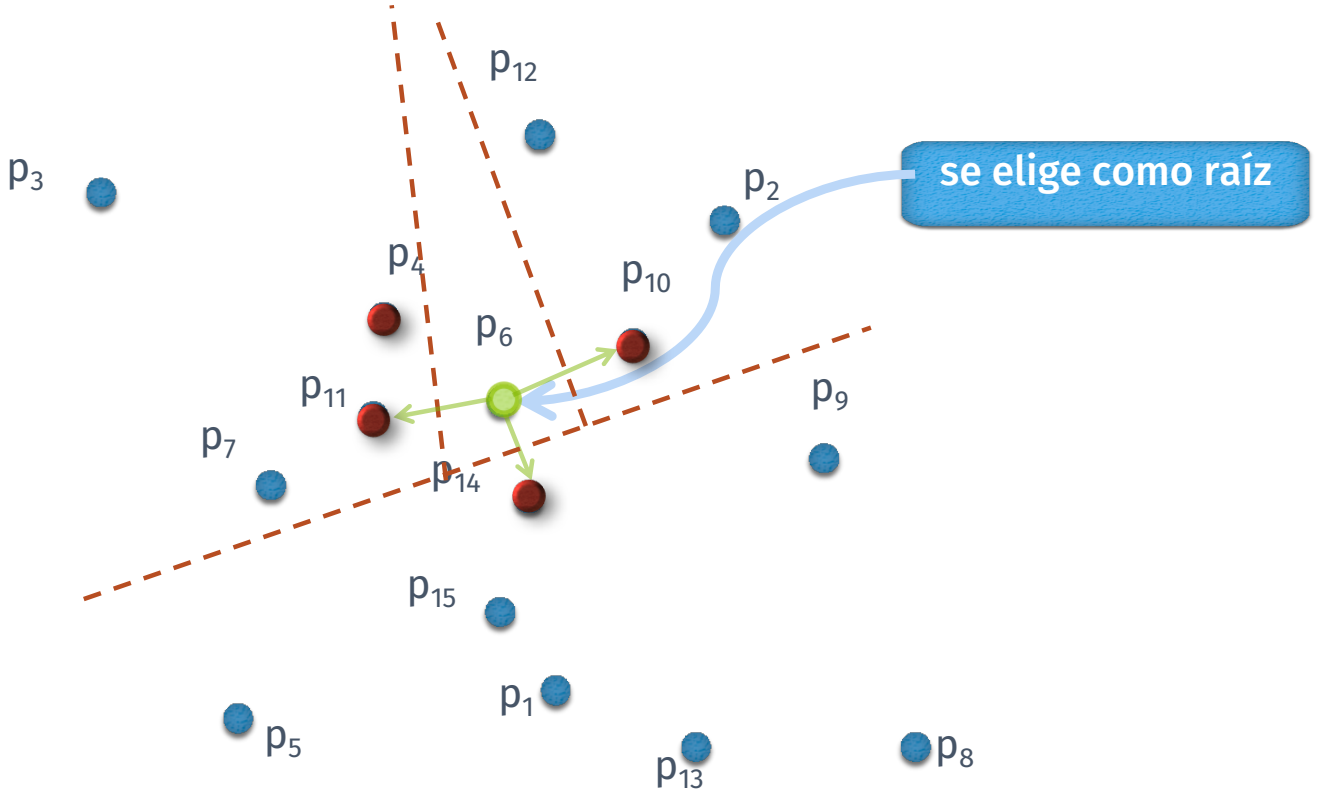
SAT. Spatial Approximation Tree

[Navarro 2002] Gonzalo Navarro. Searching in metric spaces by spatial approximation. *The Very Large DatabasesJournal (VLDBJ)*, 11(1):28–46, 2002.

SAT

- Se selecciona un elemento a como la raíz del árbol y se conecta a sus vecinos $N(a)$.
- Un elemento se convierte en vecino si está más cerca de la raíz que de los otros vecinos.
- El conjunto de vecinos depende del orden en que se consideran los elementos.
- Los demás elementos se asignan a su elemento más cercano en $N(a)$.
- Cada elemento en $N(a)$ recursivamente es la raíz de un nuevo subárbol que contiene los elementos que se le asignaron.

SAT

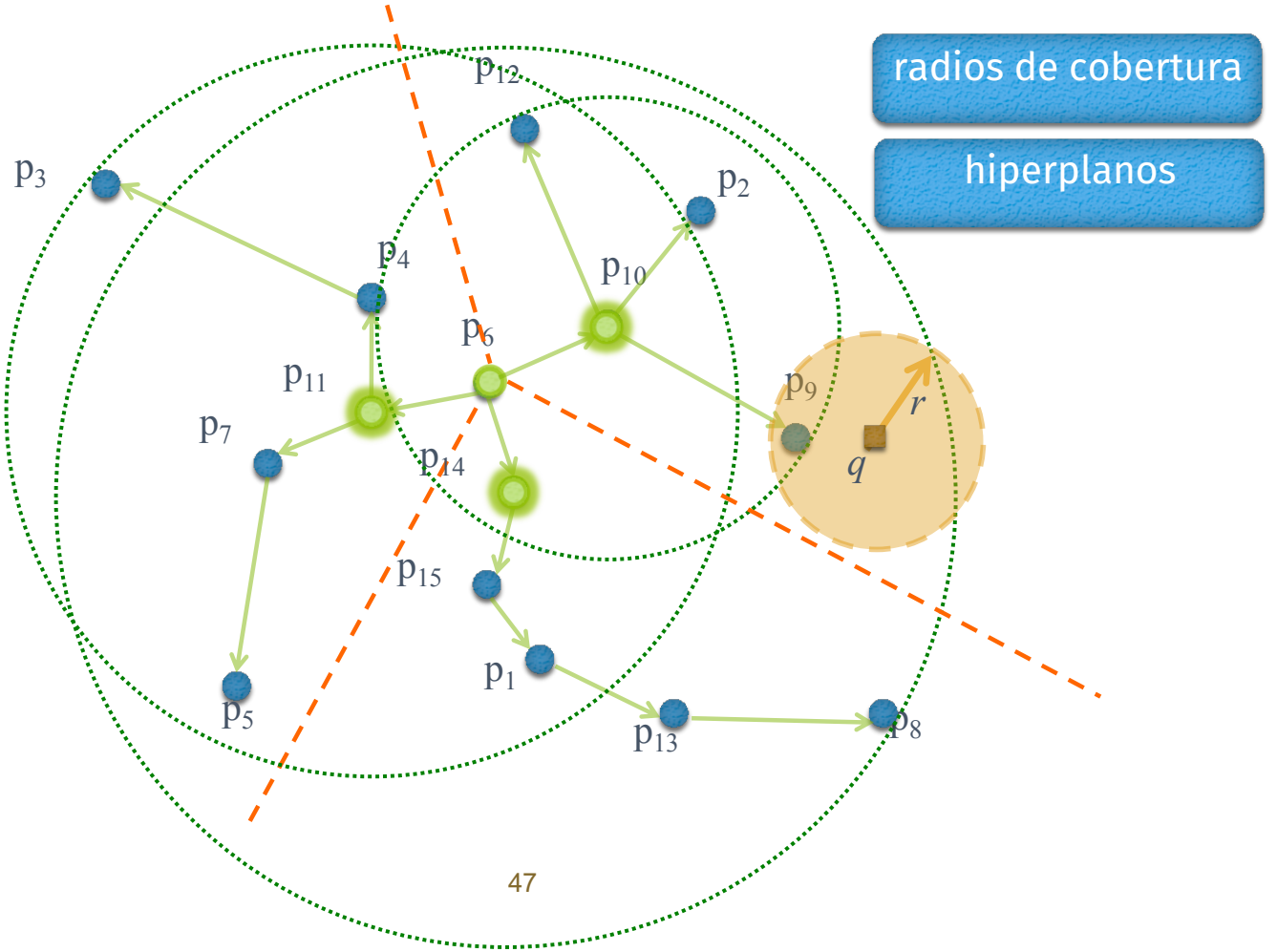


*Heurística: Se consideran los elementos en **orden creciente** de distancia a la raíz.*

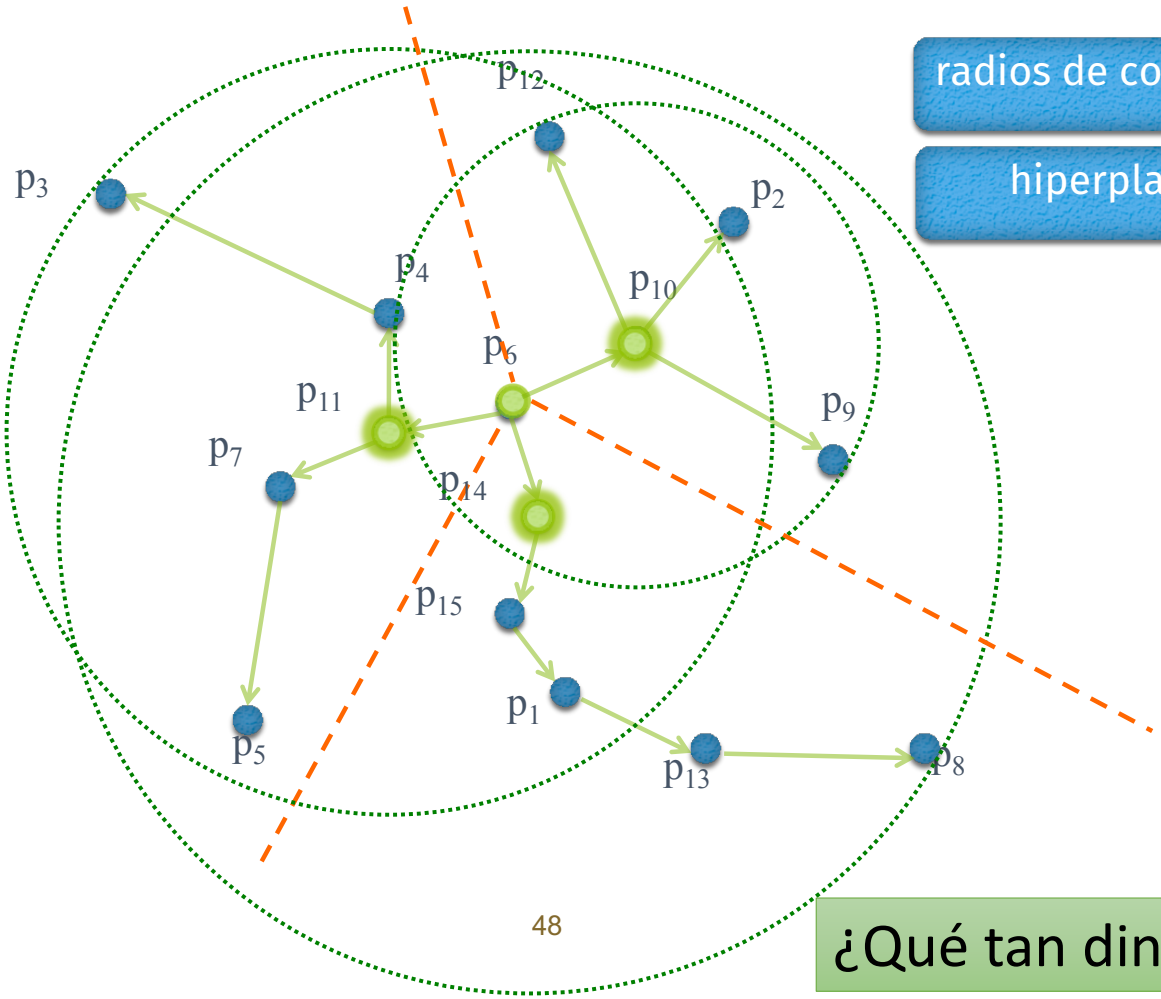
SAT

- Cada elemento mantiene su ***radio de cobertura***.
- El conjunto de vecinos determina ***hiperplanos*** entre los vecinos.
- En las búsquedas se usan a ambos para podar subárboles.
- Se entra a un subárbol si la “bola de consulta” intersecta a la zona determinada *por el radio de cobertura y también por sus hiperplanos*.

SAT


radios de cobertura
hiperplanos
p12
p3
p2
p4
p10
p6
p11
p9
r
p7
q
p14
p15
p1
p5
p13
p8

SAT



radios de cobertura

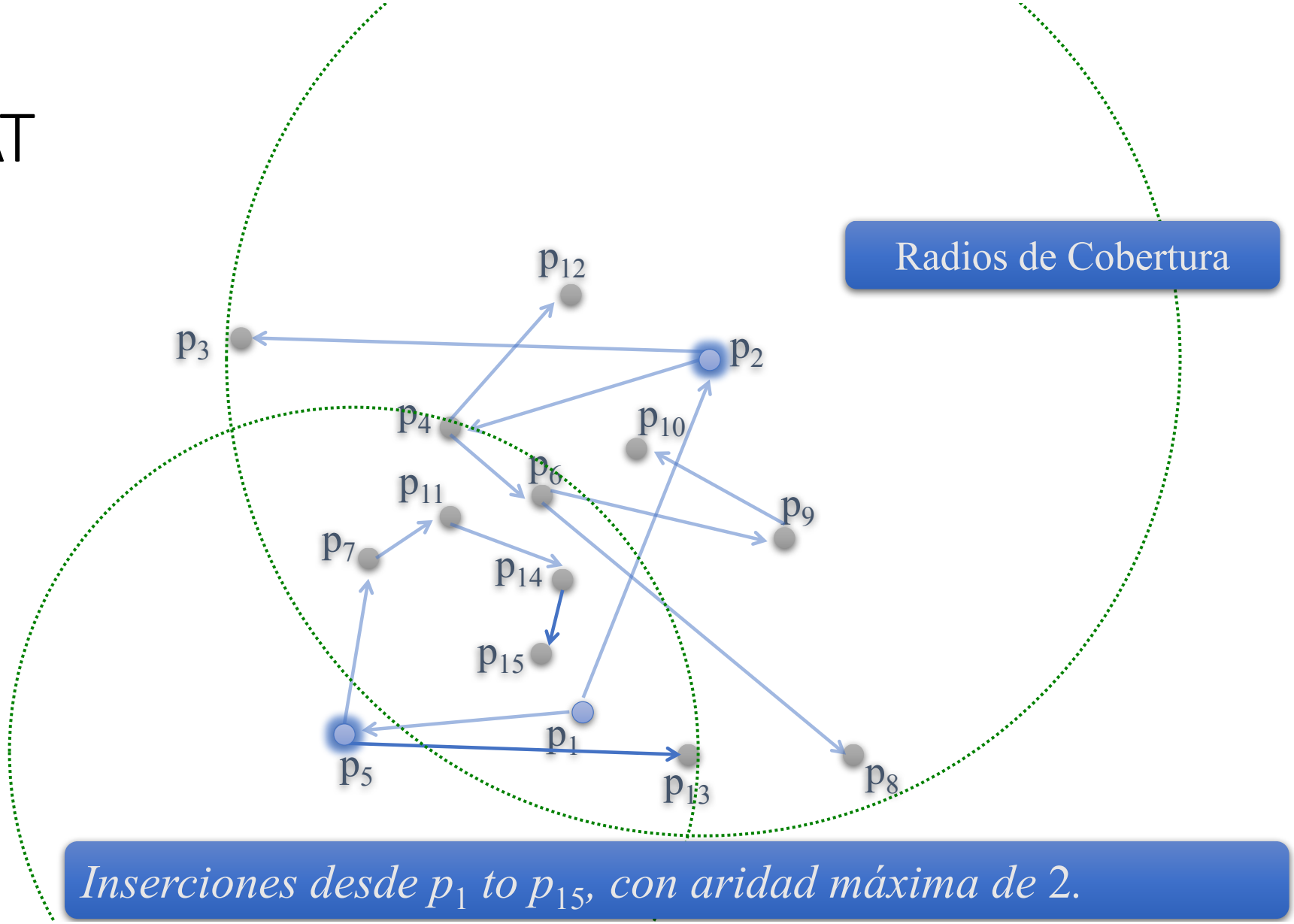
hiperplanos

¿Qué tan dinámico puede ser?

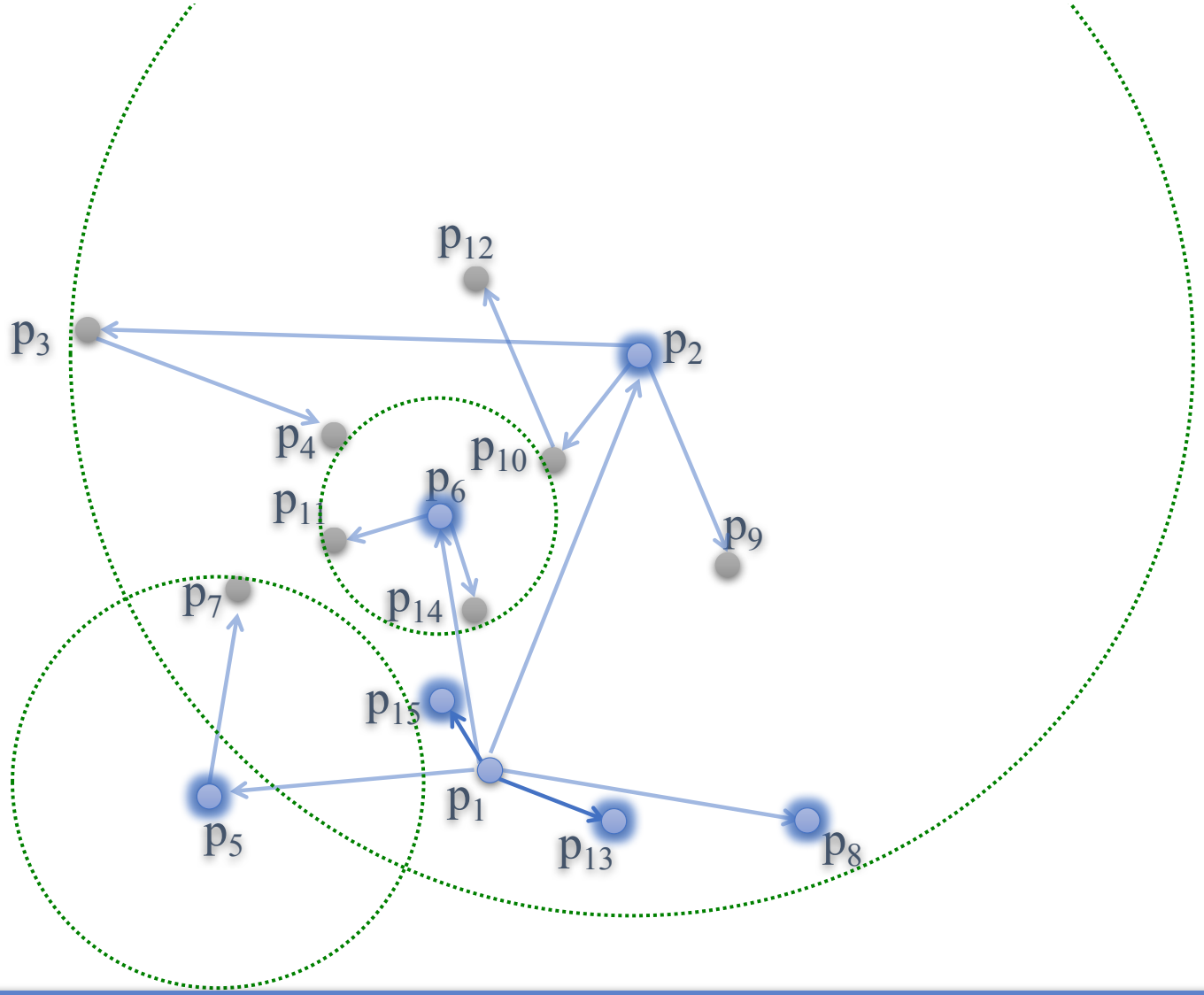
DSAT. Dynamic Spatial Approximation Tree

[Navarro and Reyes 2008] Gonzalo Navarro and Nora Reyes. Dynamic spatial approximation trees. *Journal of Experimental Algorithmics*, 12:1.5:1–1.5:68, June 2008.

DSAT


Radios de Cobertura
p12
p3
p2
p4
p10
p6
p11
p9
p7
p14
p15
p1
p5
p13
p8
Inserciones desde p1 to p15, con aridad máxima de 2.

DSAT



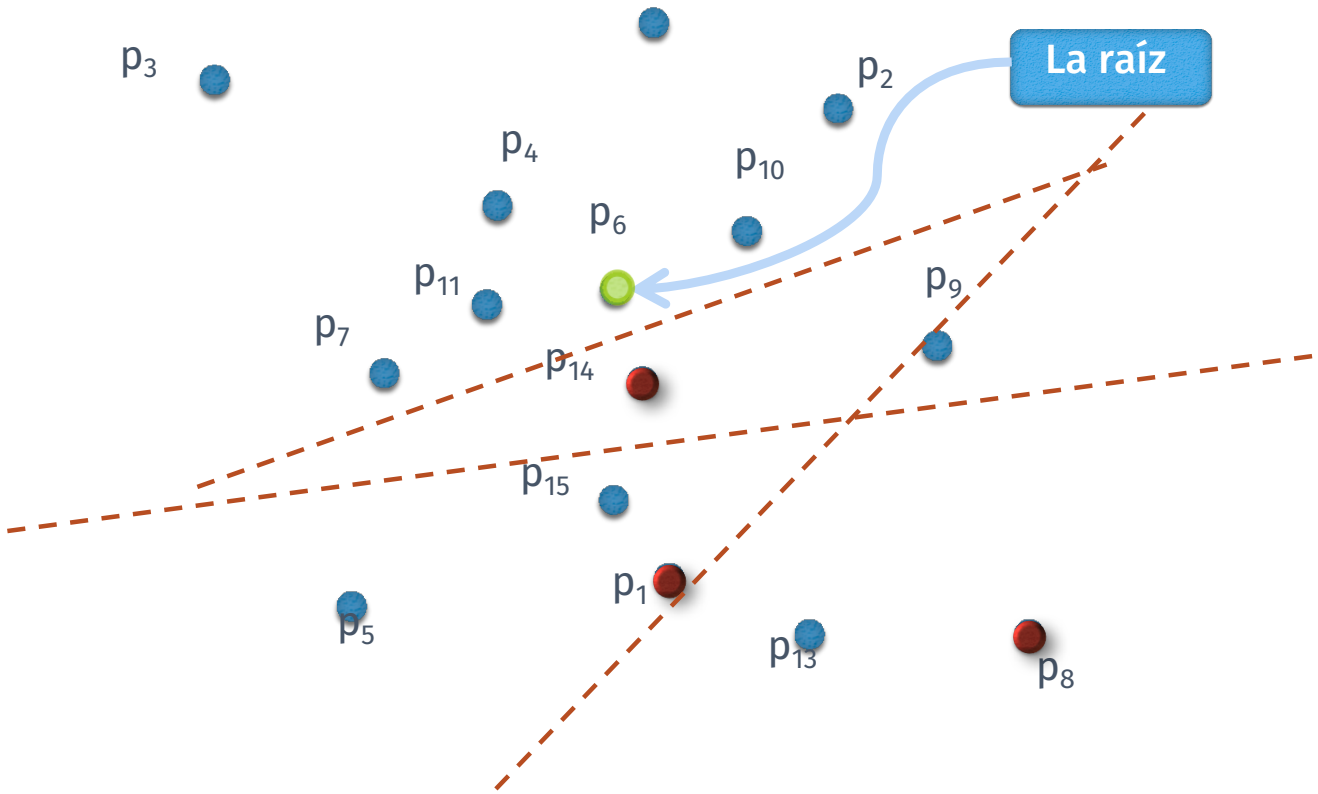
Inserciones desde p_1 to p_{15}, con aridad máxima de 6.

DSAT

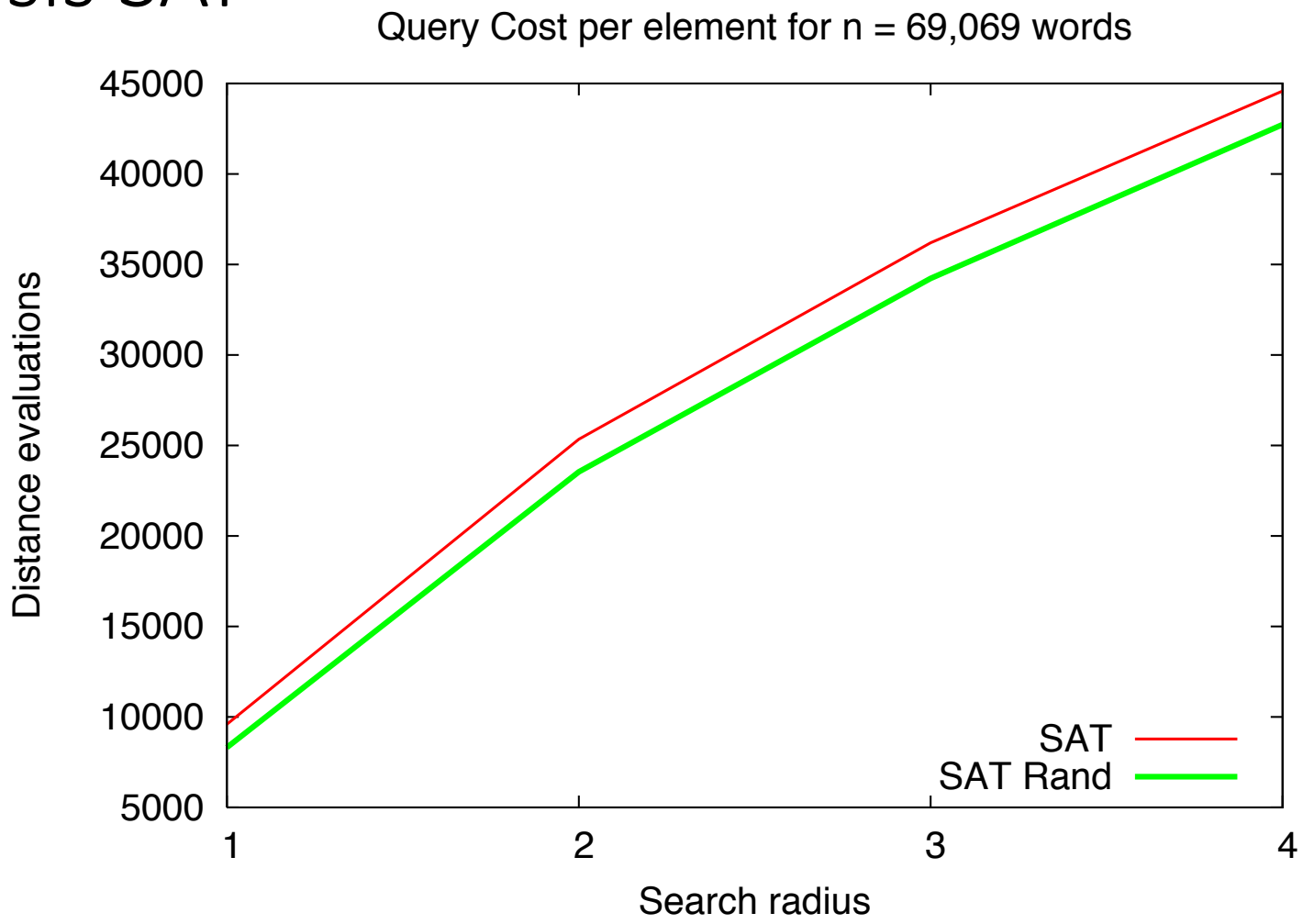
- Es una versión dinámica del SAT, admite *inserciones* y *eliminaciones* y se construye por inserciones sucesivas.
- Se diseñaron y evaluaron muchas estrategias de inserción y eliminación.
- En muchos espacios supera al SAT en costos de creación y de búsqueda.

¿No dispone de toda la información que tiene el SAT y aún así lo supera en costos de construcción y en las búsquedas?

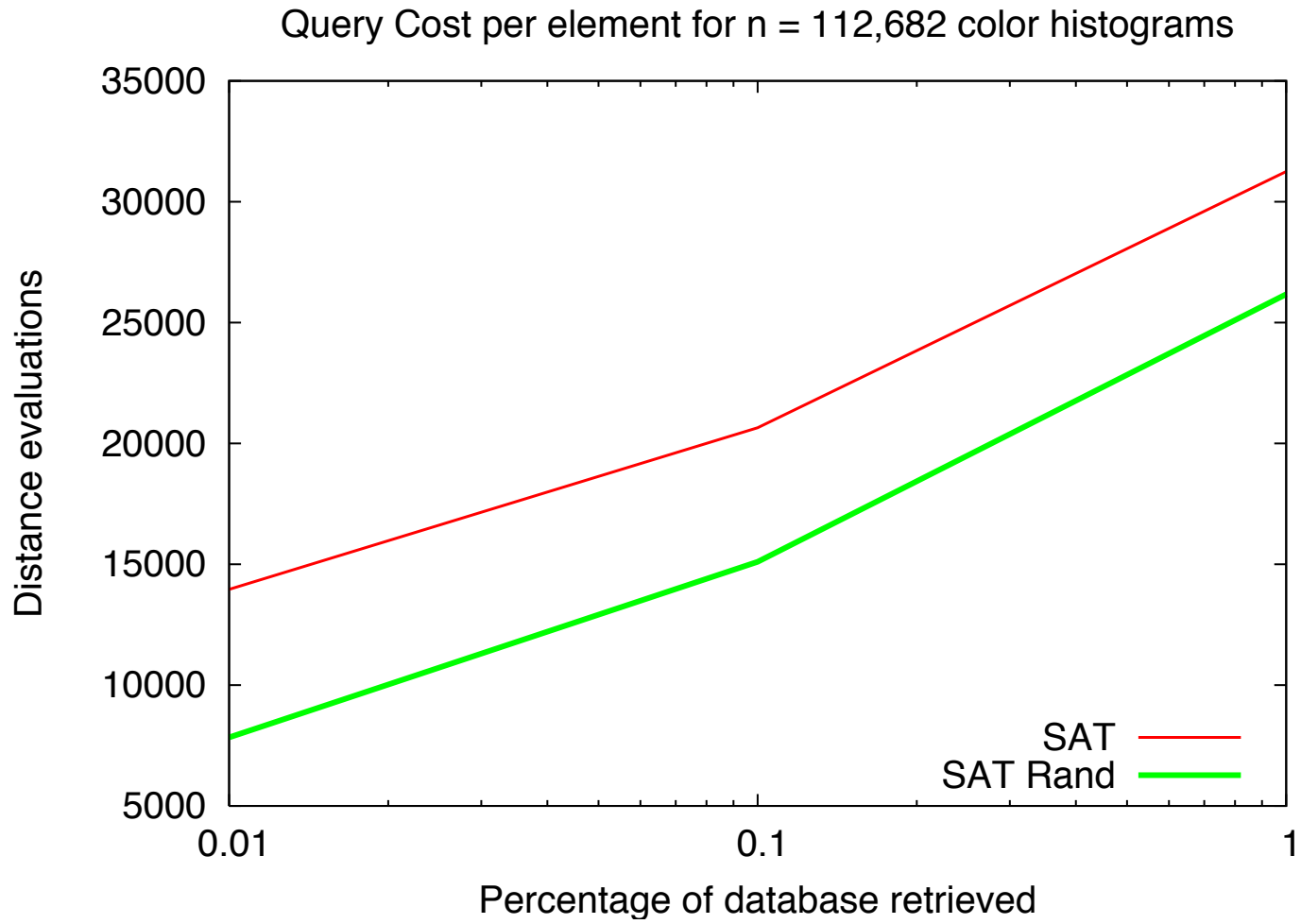
Análisis SAT



Análisis SAT



Análisis SAT

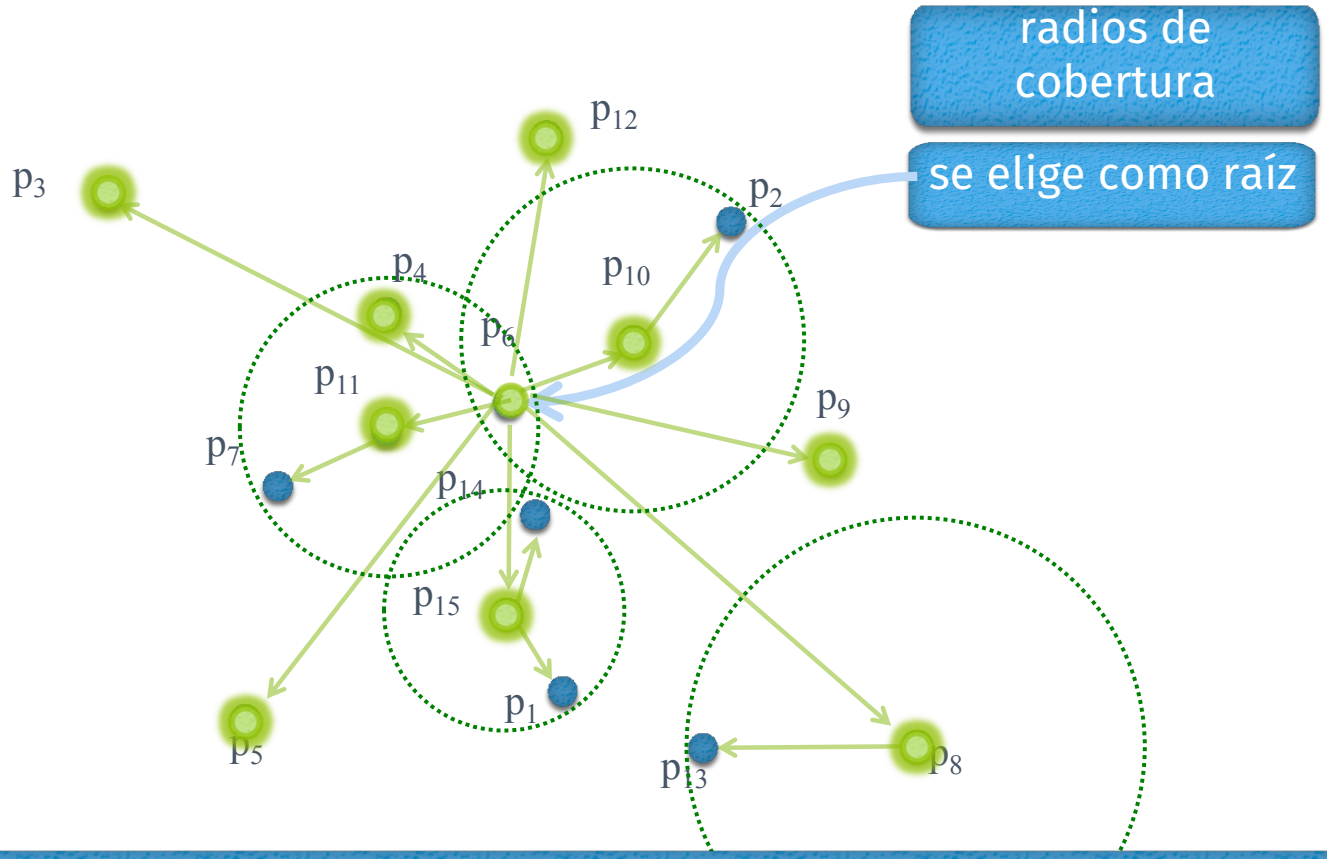


DiSAT. Distal Spatial Approximation Tree

[Chávez, Ludueña, Reyes, Roggero 2016] Edgar Chávez, Verónica Ludeña, Nora Reyes, and Patricia Roggero. Faster proximity searching with the distal sat. *Information Systems*, 2016.

DiSAT

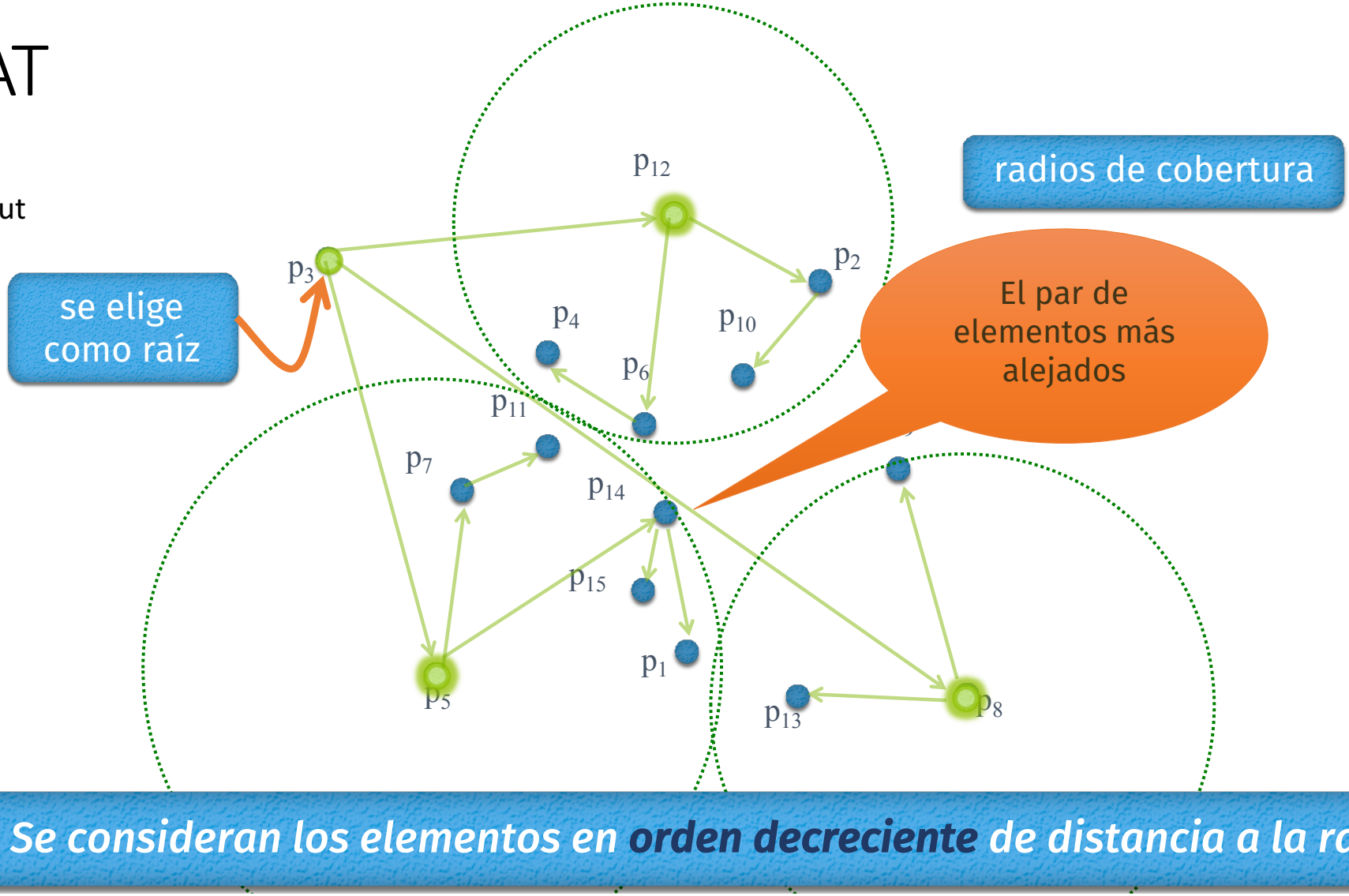
- SAT^{+} y SAT^{Glob}



Heurística: Primer nivel orden creciente. Otros niveles igual o sin orden.

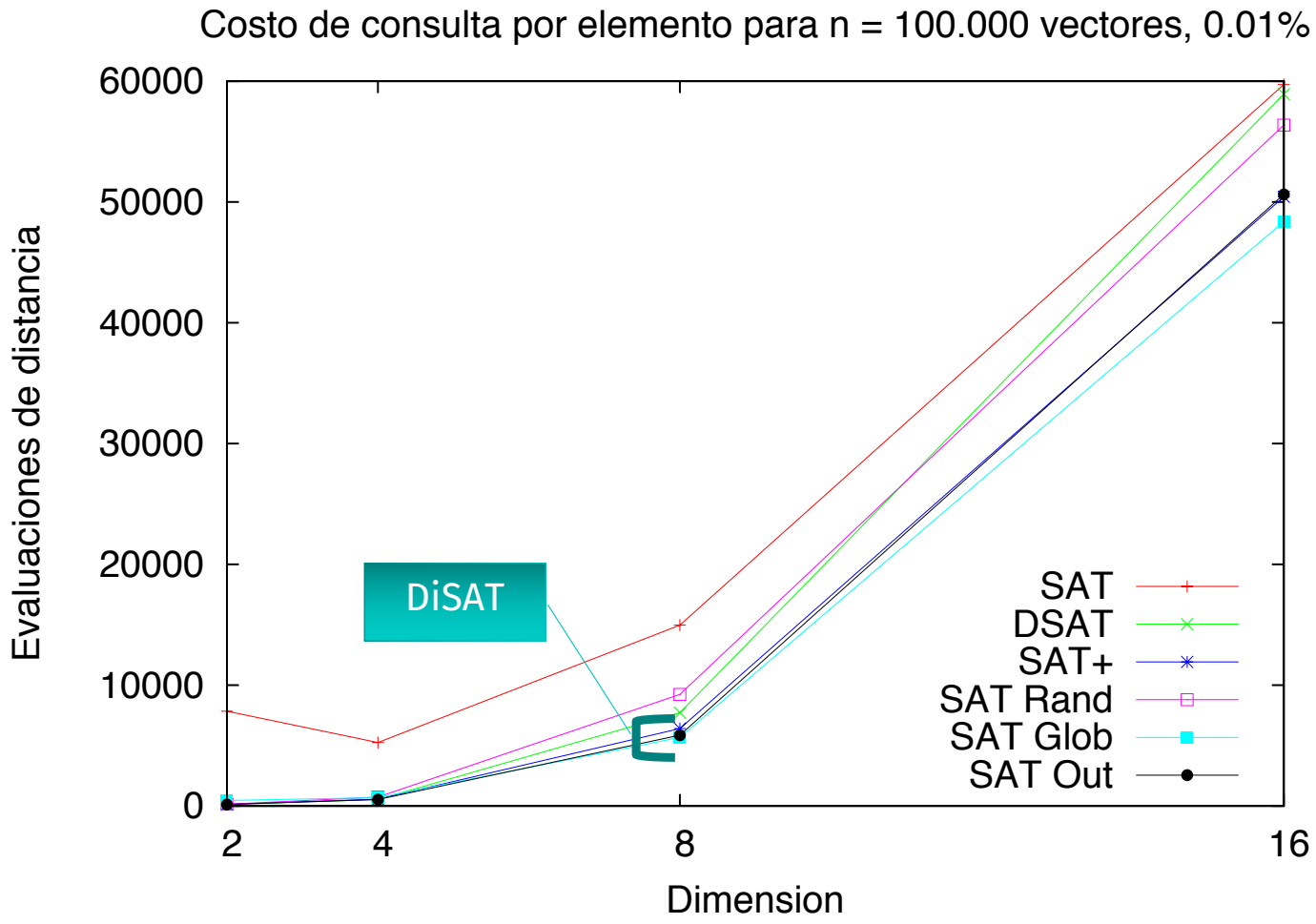
DiSAT

- SAT^{Out}

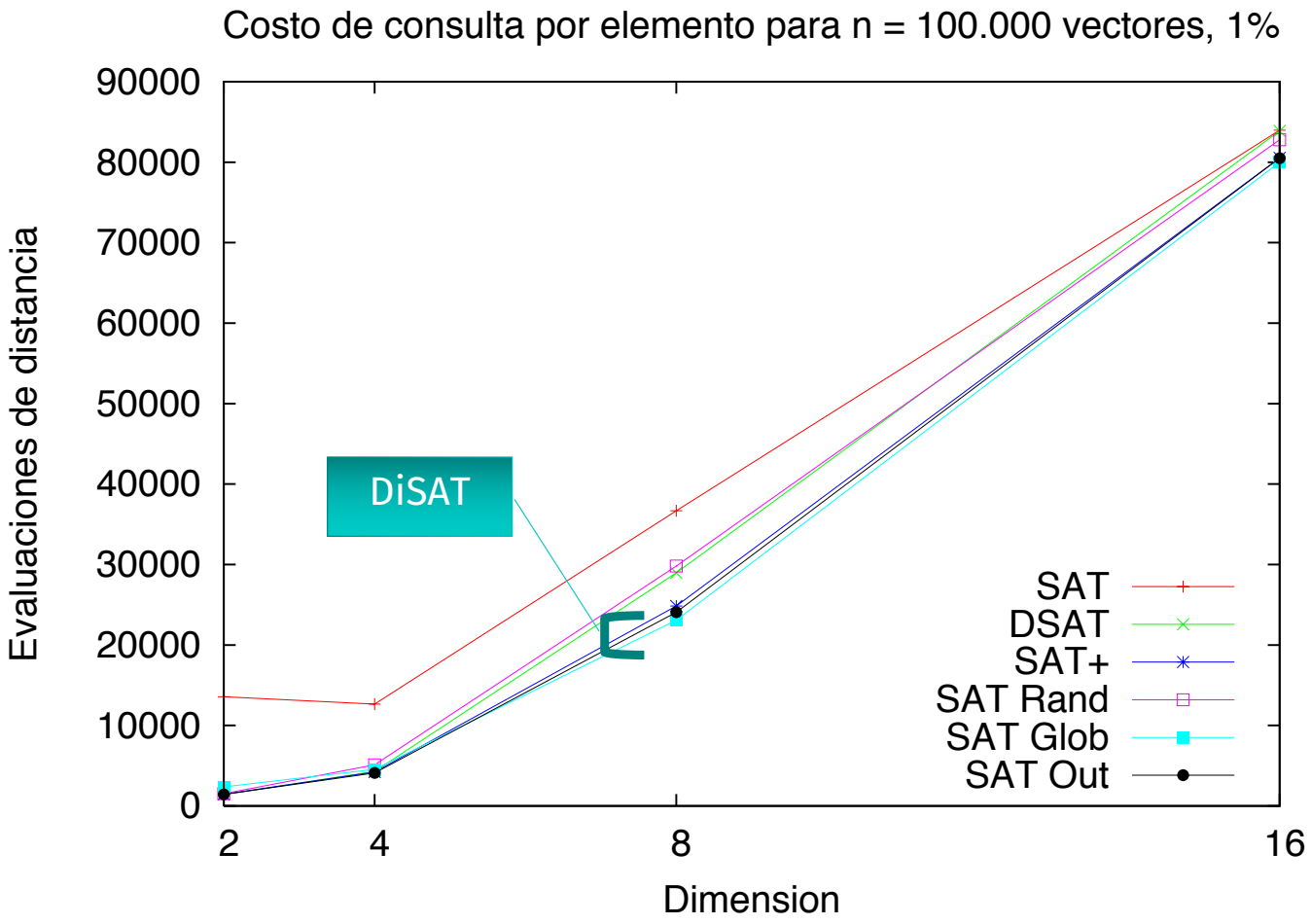


*Heurística: Se consideran los elementos en **orden decreciente** de distancia a la raíz.*

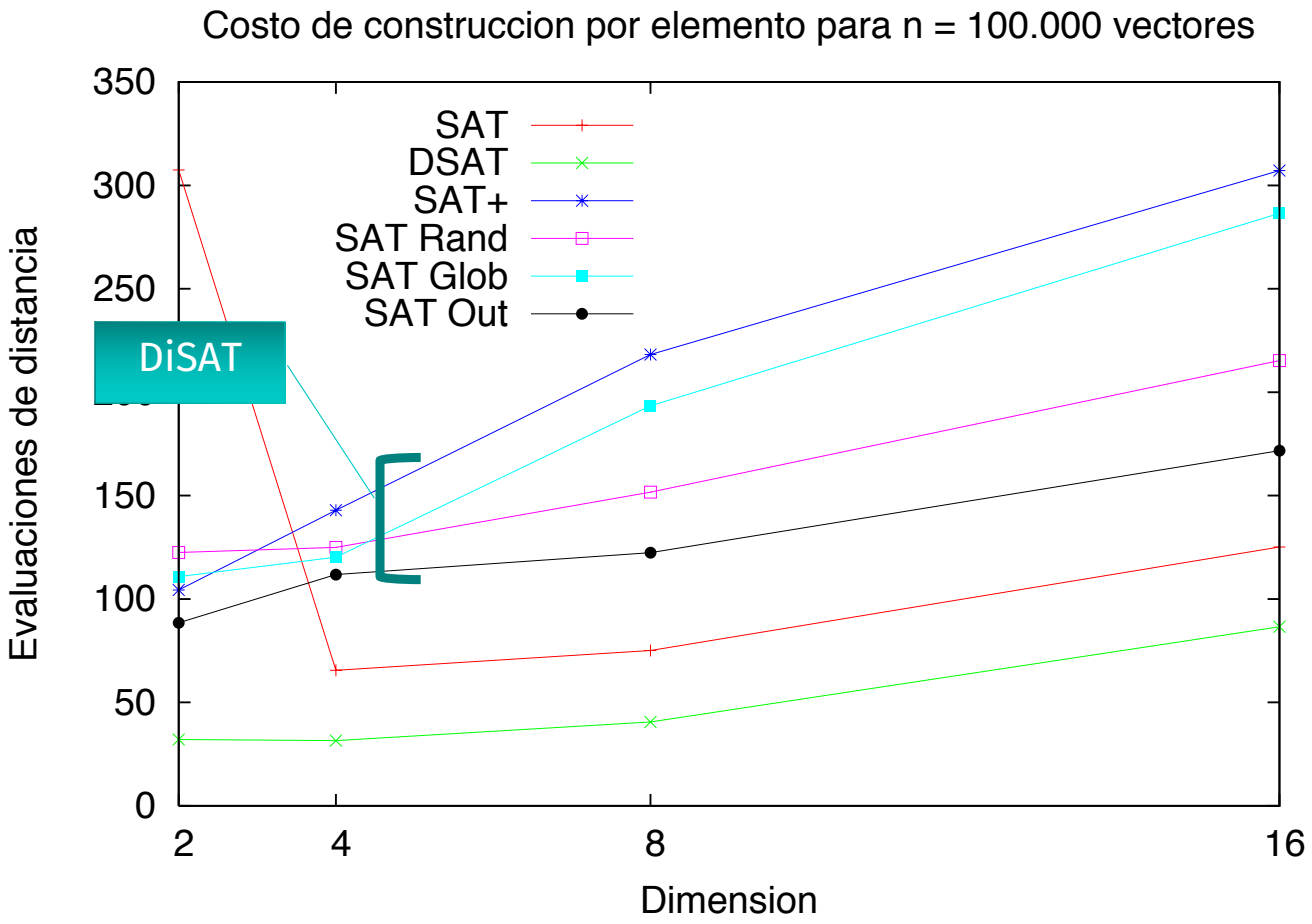
DiSAT


Costo de consulta por elemento para n = 100.000 vectores, 0.01%
Evaluaciones de distancia
60000
50000
40000
30000
20000
10000
0
2
4
8
16
Dimension
DiSAT
SAT
DSAT
SAT+
SAT Rand
SAT Glob
SAT Out

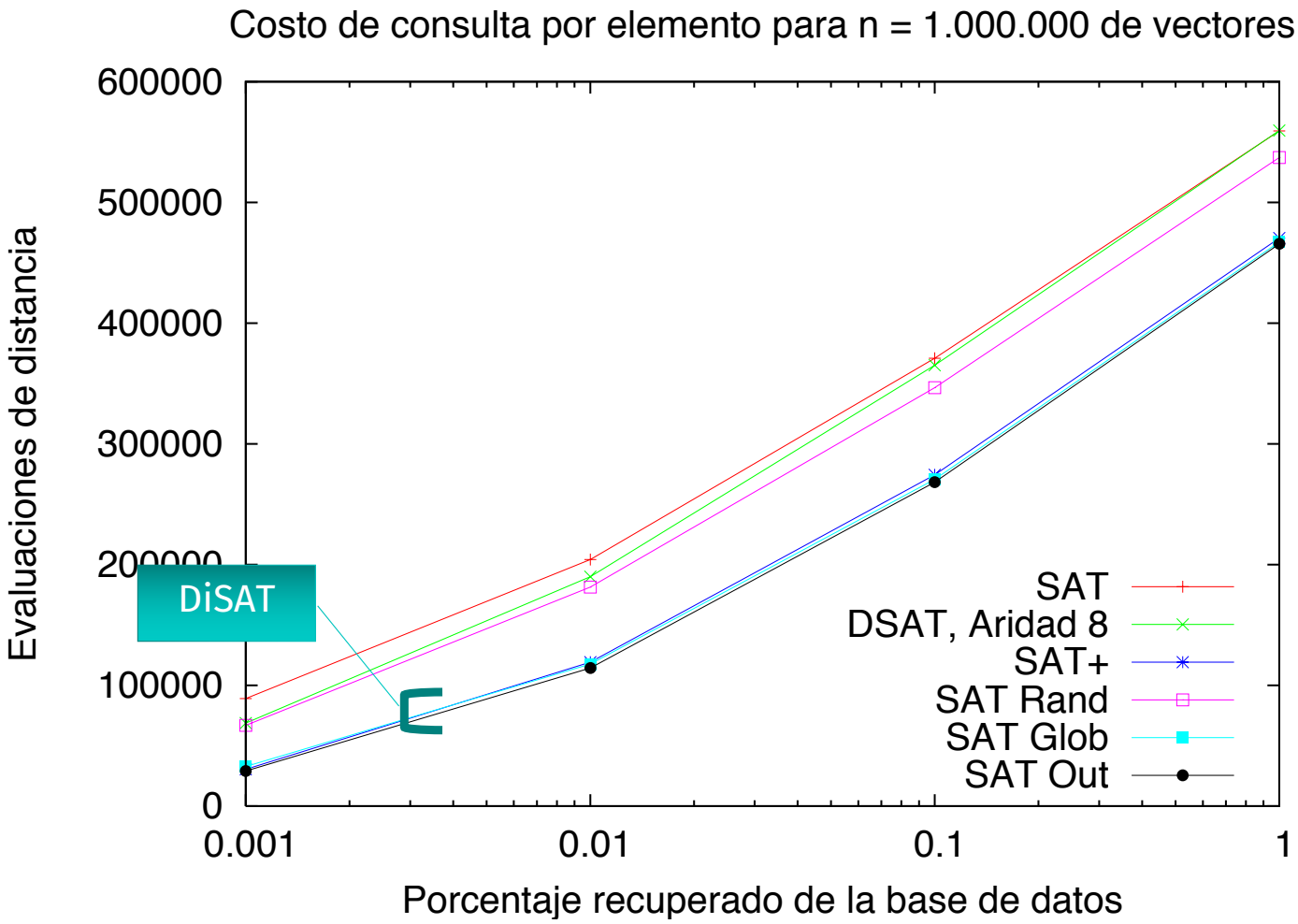
DiSAT


Costo de consulta por elemento para n = 100.000 vectores, 1%
Evaluaciones de distancia
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
2
4
8
16
Dimension
DiSAT
SAT
DSAT
SAT+
SAT Rand
SAT Glob
SAT Out

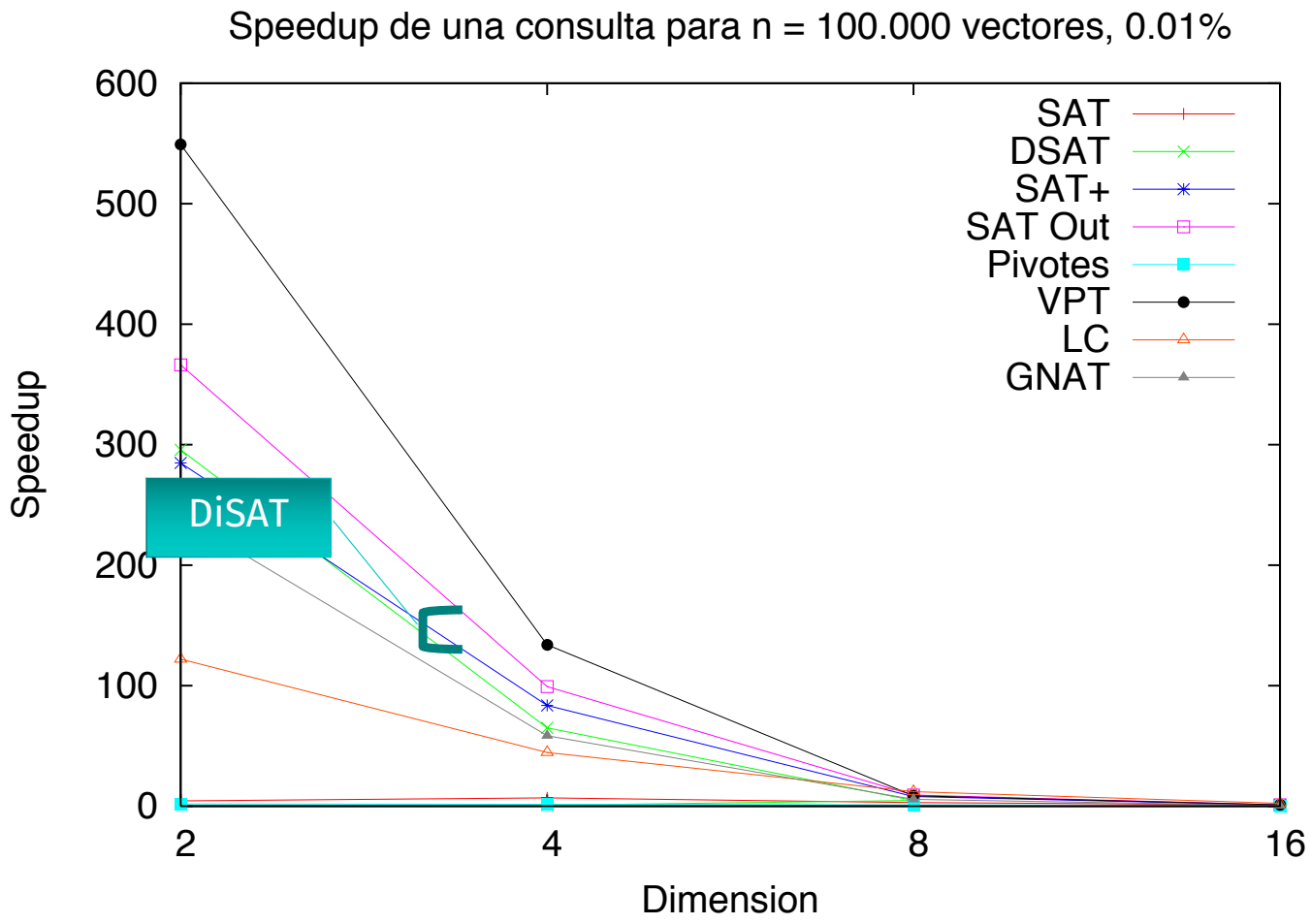
DiSAT


Costo de construccion por elemento para n = 100.000 vectores
Evaluaciones de distancia
Dimension
350
300
250
150
100
50
0
2
4
8
16
SAT
DSAT
SAT+
SAT Rand
SAT Glob
SAT Out
DiSAT

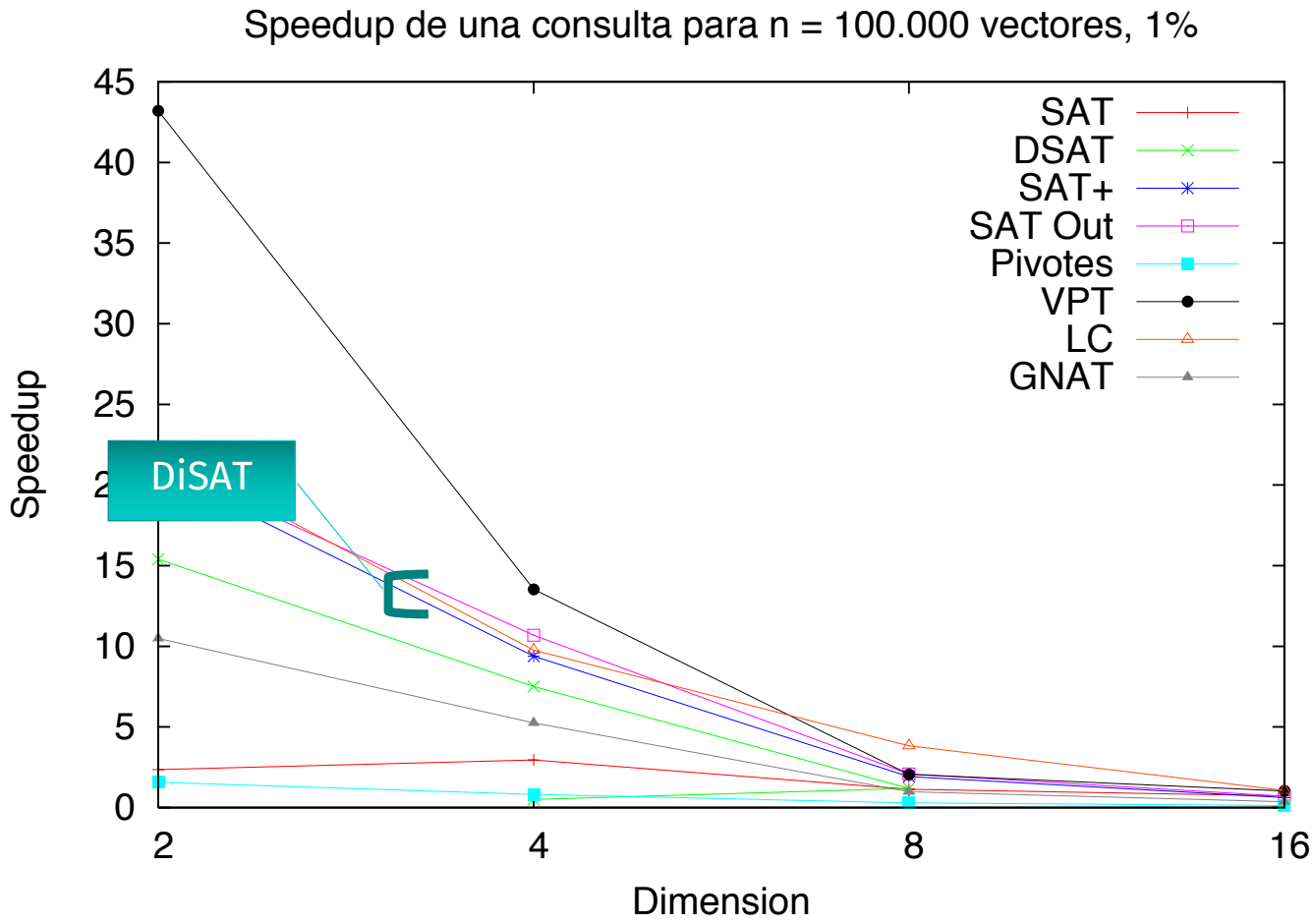
DiSAT


Costo de consulta por elemento para n = 1.000.000 de vectores
Evaluaciones de distancia
600000
500000
400000
300000
200000
100000
0
0.001
0.01
0.1
1
Porcentaje recuperado de la base de datos
DiSAT
SAT
DSAT, Aridad 8
SAT+
SAT Rand
SAT Glob
SAT Out

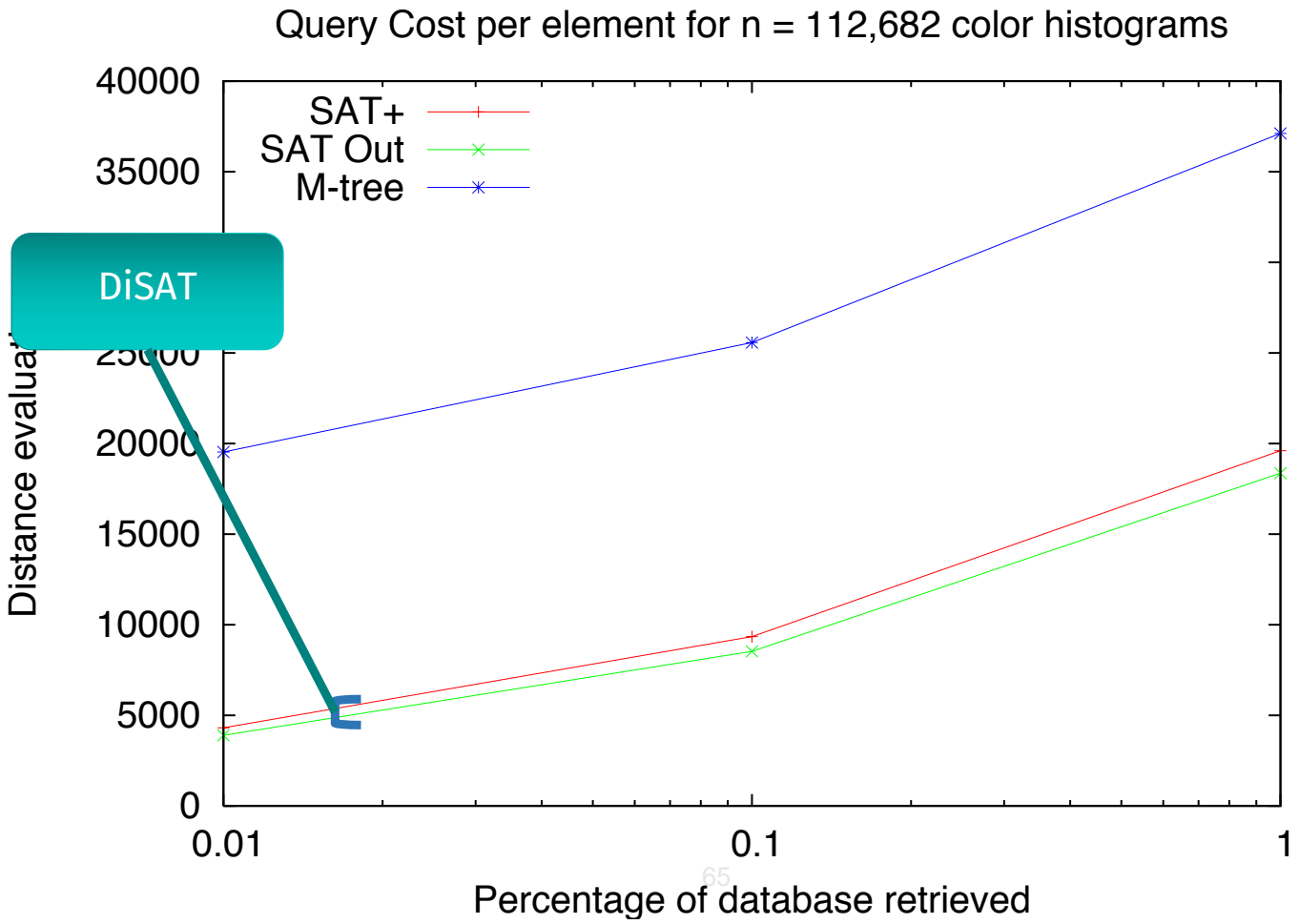
DiSAT


Speedup de una consulta para n = 100.000 vectores, 0.01%
Speedup
Dimension
600
500
400
300
200
100
0
2
4
8
16
SAT
DSAT
SAT+
SAT Out
Pivotes
VPT
LC
GNAT
DiSAT

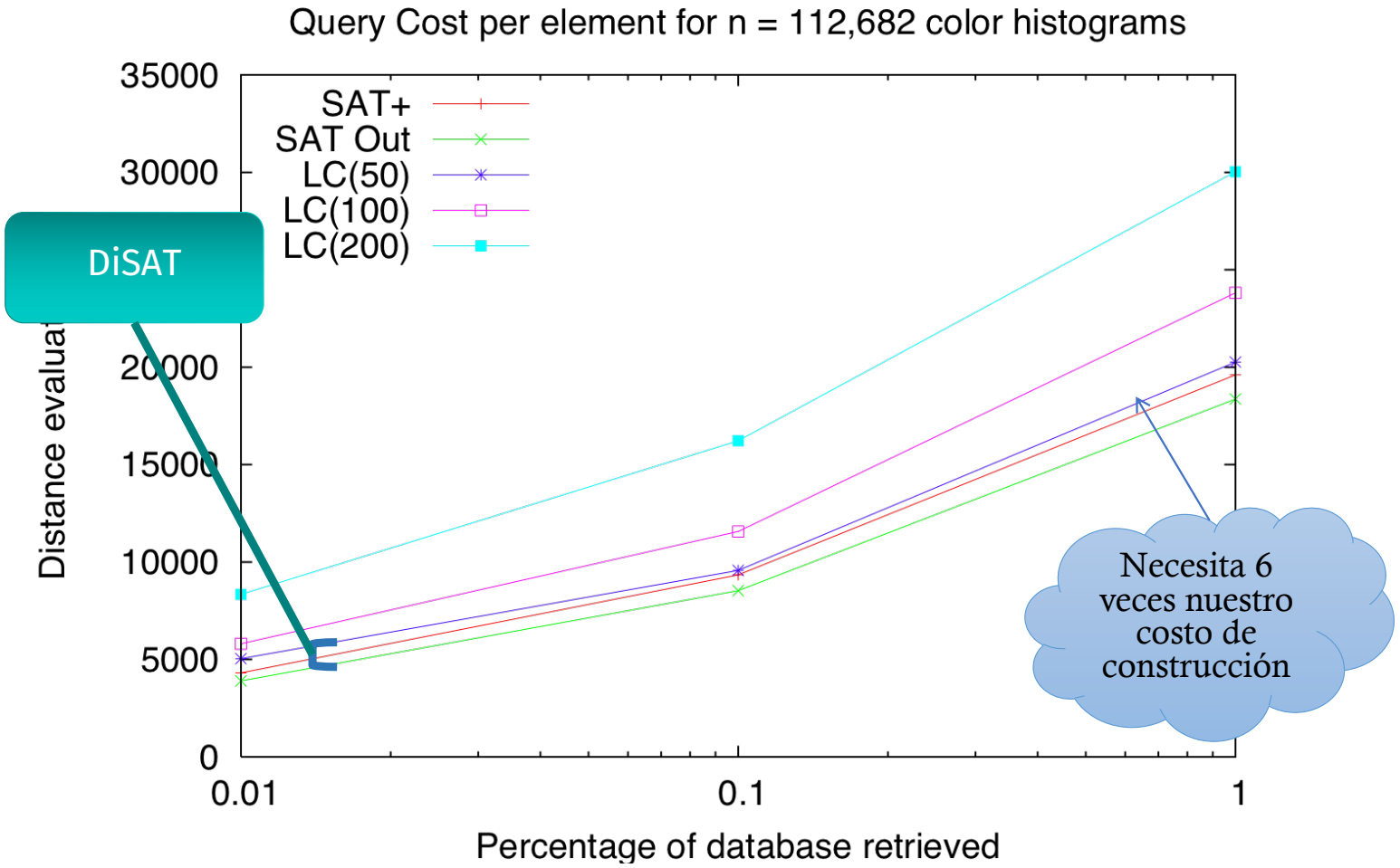
DiSAT


Speedup de una consulta para n = 100.000 vectores, 1%
Speedup
Dimension
45
40
35
30
25
20
15
10
5
0
2
4
8
16
SAT
DSAT
SAT+
SAT Out
Pivotes
VPT
LC
GNAT
DiSAT

DiSAT


Query Cost per element for n = 112,682 color histograms
SAT+
SAT Out
M-tree
DiSAT
40000
35000
20000
15000
10000
5000
0
0.01
0.1
1
Percentage of database retrieved

DiSAT


Query Cost per element for n = 112,682 color histograms
SAT+
SAT Out
LC(50)
LC(100)
LC(200)
DiSAT
35000
30000
20000
15000
10000
5000
0
0.01
0.1
1
Percentage of database retrieved
Necesita 6 veces nuestro costo de construcción

DiSAT

- Buenos hiperplanos mejoran las búsquedas.
- Además, disminuyen los radios de cobertura.

Espacio de Vectores de la NASA

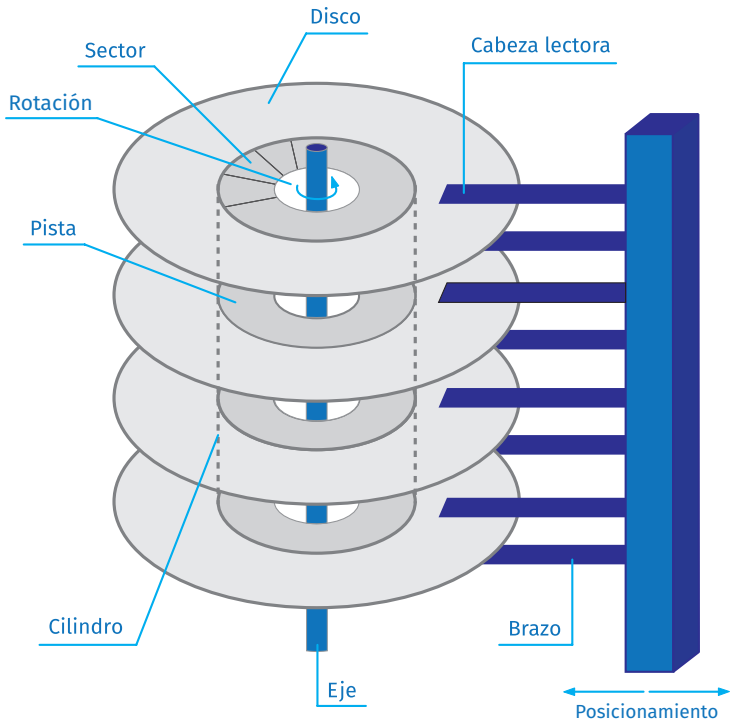
	Ambos	Hiperplano	Radio Cobertura
SAT	344.4	601.7	2369.8
SAT^{+}	266.1	299.9	1127.5
SAT^{Glob}	309.2	446.2	1048.5
SAT^{Out}	238.2	272.0	1025.0

Memoria Secundaria

- Bases de datos masivas y/o objetos grandes:

Memoria Secundaria

- Todavía el dispositivo de almacenamiento masivo más común es el disco.



- El tiempo de acceso a disco (T_{acc}) se calcula como la suma de:
  - ✓ T_{pos}
  - ✓ T_{rot}
  - ✓ T_{transf}

Índices

- ¿Bases de datos pequeñas o masivas?

- Analicemos los índices basados en pivotes:
    - BKT: Burkhard Keller Tree
    - FQT: Fixed Query Tree
    - VPT: Vantage Point Tree
    - AESA: Approximating and Eliminating Searching Algorithm

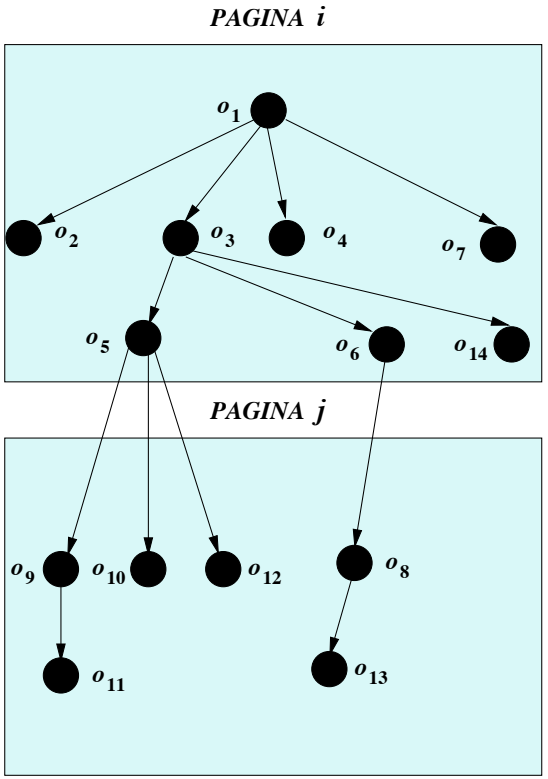
DSAT+ y DSAT*. Dynamic Spatial Approximation Tree for Massive Data

[Navarro and Reyes 2010] Gonzalo Navarro and Nora Reyes. Dynamic spatial approximation trees for massive data. *SISAP 2009*. IEEE Computer Society, pages 81–88.

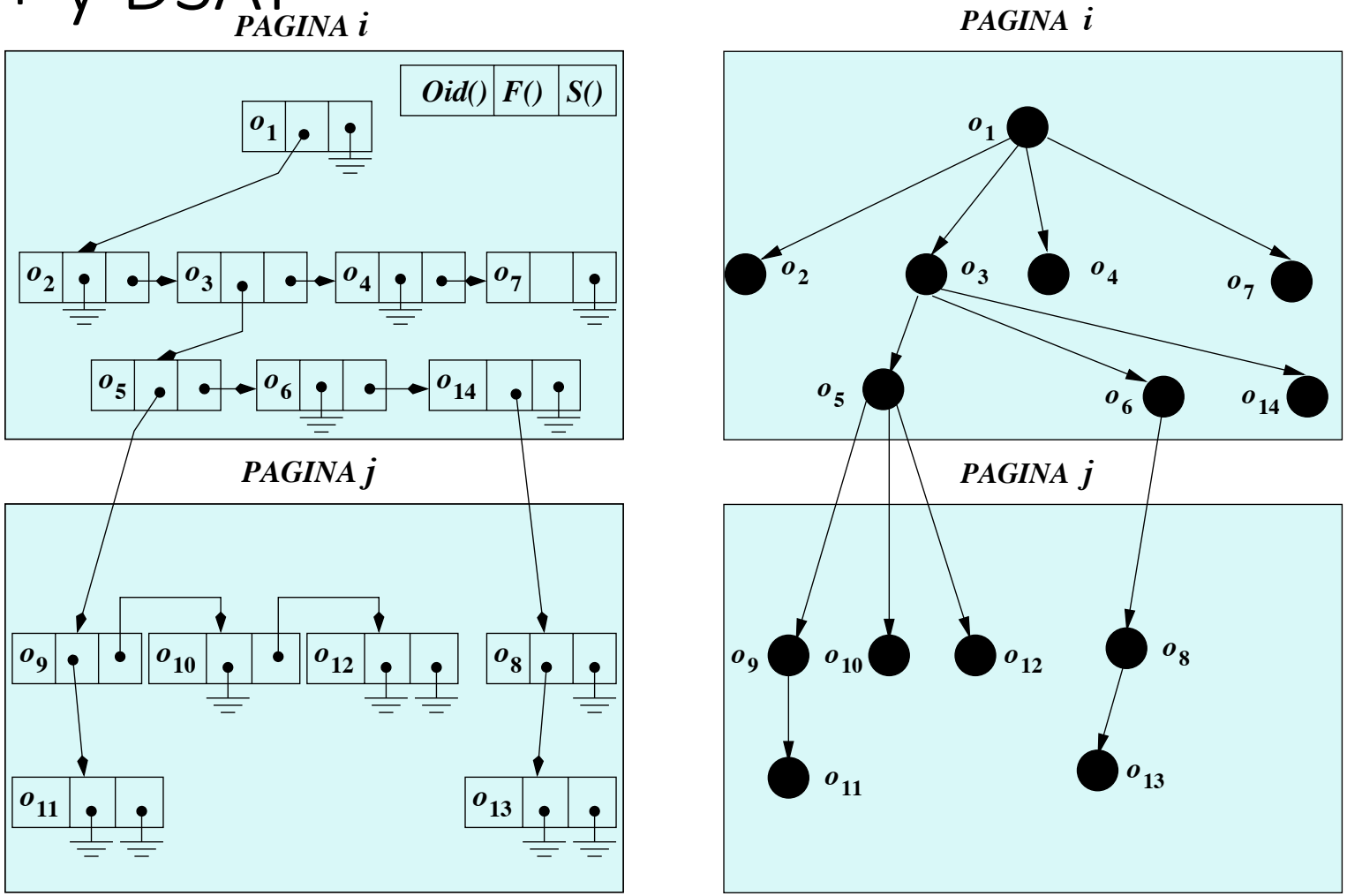
Índices basados en particiones compactas

- *Árbol de Aproximación Espacial Dinámico* DSAT+ y DSAT* son versiones del DSAT para memoria secundaria.

- DSAT+ y DSAT* son DSATs con una estrategia. para volcar los nodos del árbol en páginas de disco.

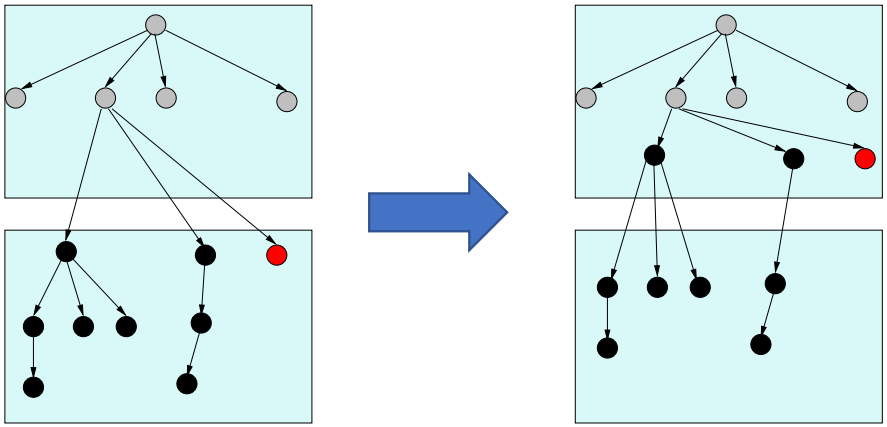


DSAT+ y DSAT*


PAGINA i
Oid()
F()
S()
o1
o2
o3
o4
o7
o5
o6
o14
PAGINA j
o9
o10
o12
o8
o11
o13
PAGINA i
o1
o2
o3
o4
o7
o5
o6
o14
PAGINA j
o9
o10
o12
o8
o11
o13

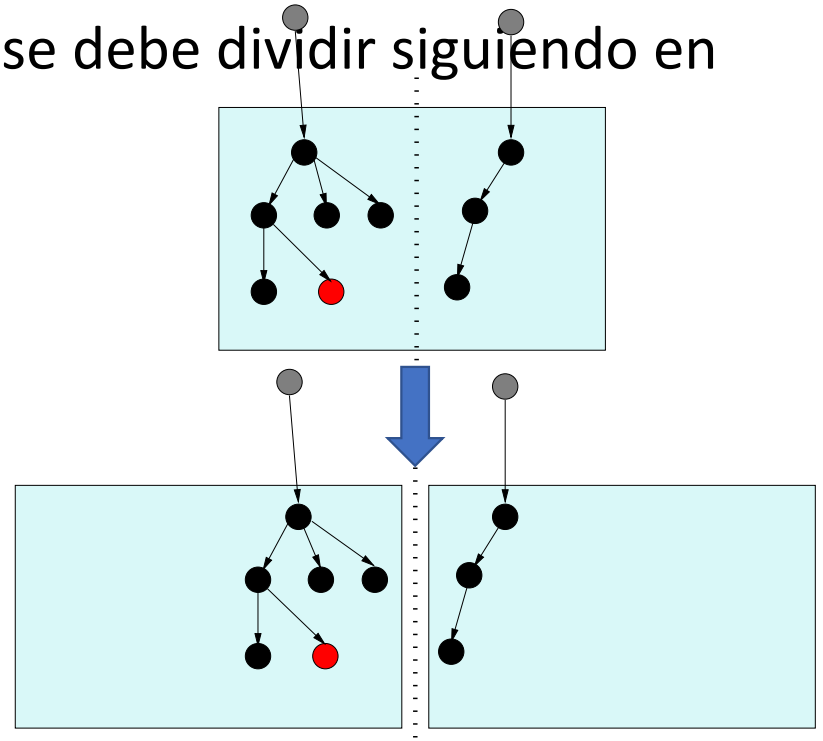
DSAT+ y DSAT*

- Difieren en que DSAT* asegura una ocupación de página del 50%, pero tiene menos localidad de referencia.
- Cuando una página de disco se llena, se debe dividir siguiendo en orden las siguientes soluciones:
  - Mover al padre
  - División Vertical
  - División Horizontal



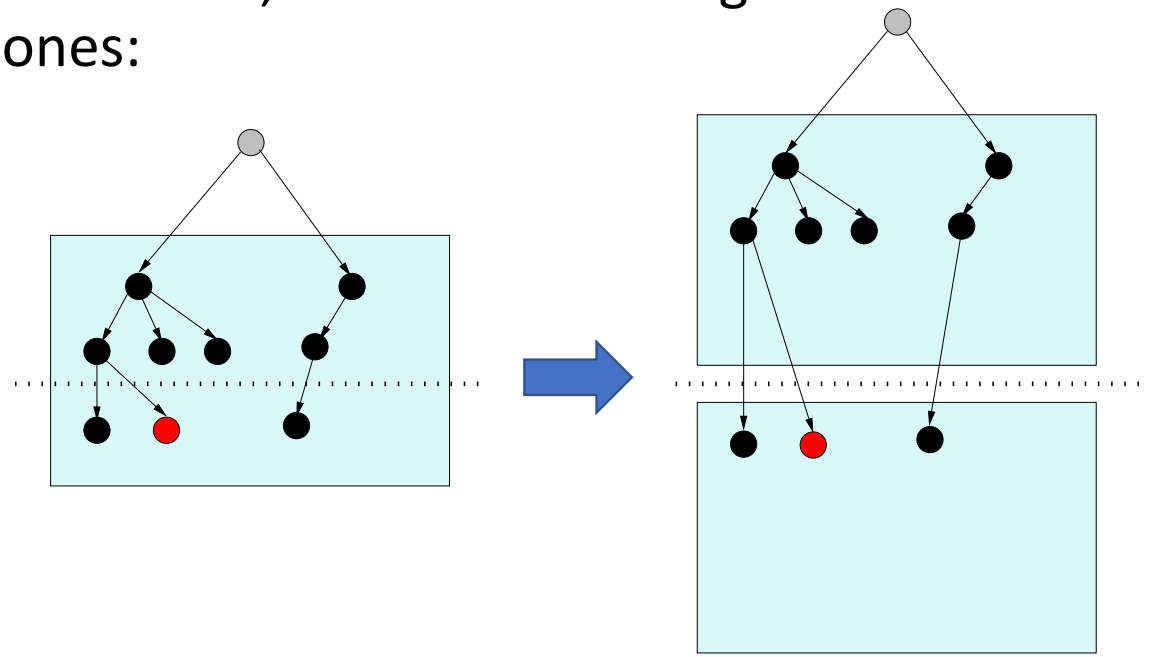
DSAT+ y DSAT*

- Difieren en que DSAT* asegura una ocupación de página del 50%, pero tiene menos localidad de referencia.
- Cuando una página de disco se llena, se debe dividir siguiendo en orden las siguientes soluciones:
  - Mover al padre
  - División Vertical
  - División Horizontal



DSAT+ y DSAT*

- Difieren en que DSAT* asegura una ocupación de página del 50%, pero tiene menos localidad de referencia.
- Cuando una página de disco se llena, se debe dividir siguiendo en orden las siguientes soluciones:
  - Mover al padre
  - División Vertical
  - División Horizontal

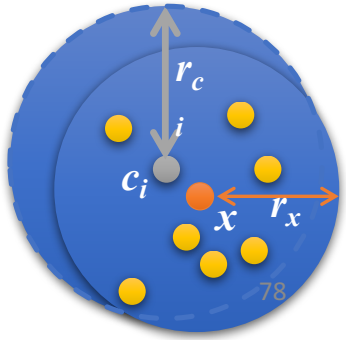


DLC: Dynamic List of Clusters
DSC: Dynamic Set of Clusters

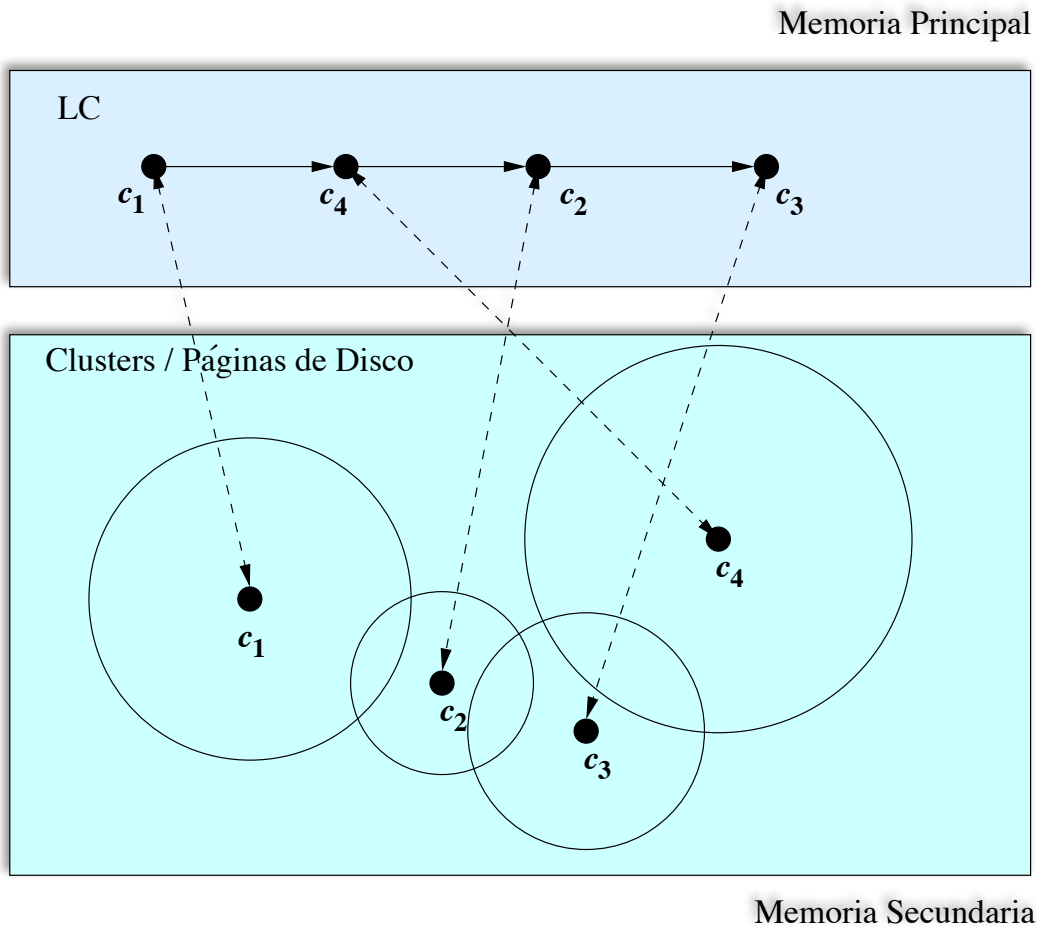
[Navarro and Reyes 2016] Gonzalo Navarro and Nora Reyes. New dynamic metric indices for secondary memory. *Information Systems*, 2016. ISSN: 0306-4379

DLC

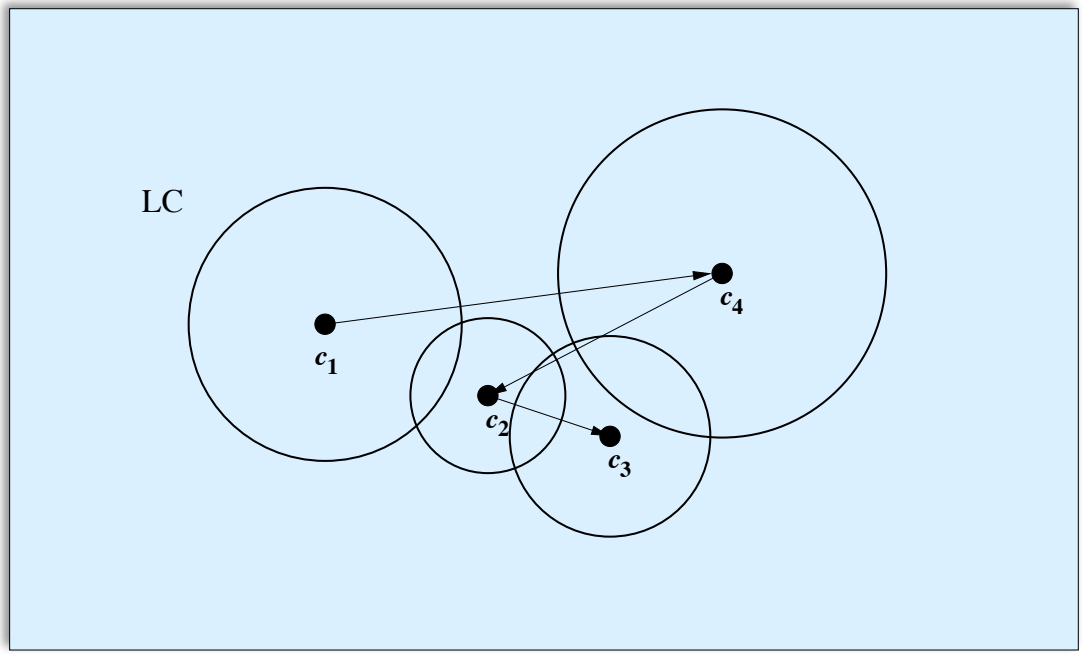
- *Lista de Clusters Dinámica en Memoria Secundaria* (DLC) comienza vacía y se construye por inserciones sucesivas.
- Para insertar un nuevo objeto *x* se ubica el clúster más adecuado para acomodarlo.
- Cada clúster corresponde a una o más páginas de disco.
- La estructura del clúster podría mejorarse por la inserción del nuevo elemento *x*.



DLC


Memoria Principal
LC
c_1
c_4
c_2
c_3
Clusters / Páginas de Disco
c_1
c_2
c_3
c_4
Memoria Secundaria

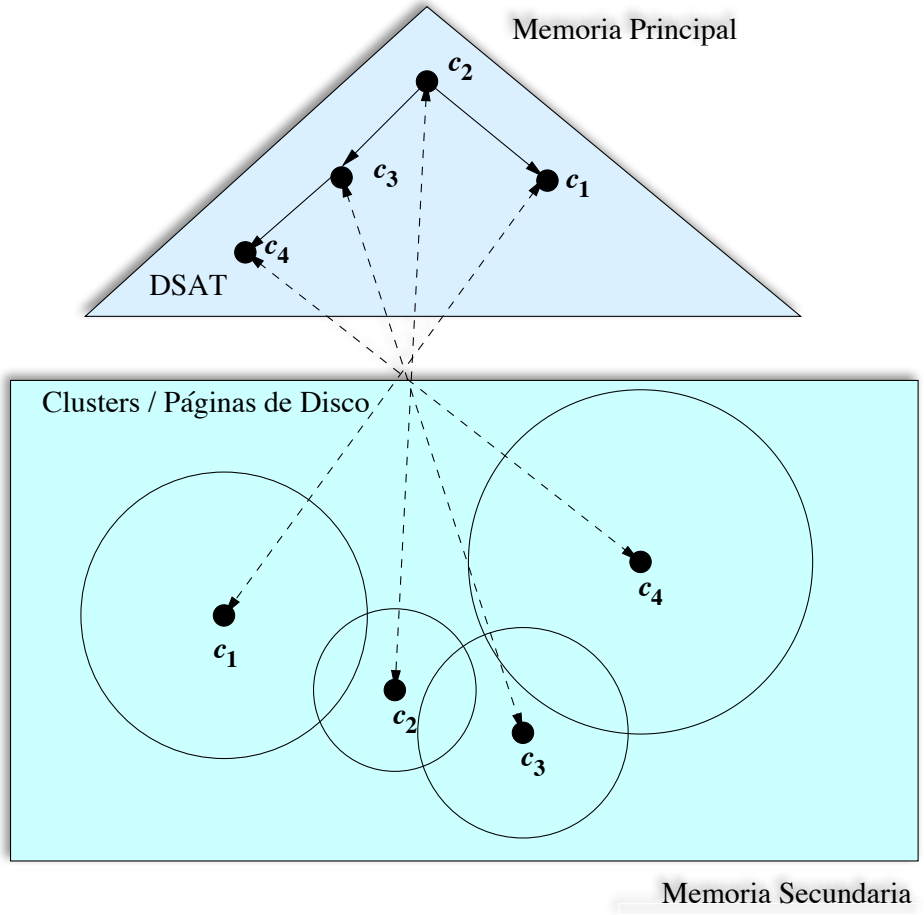
DLC



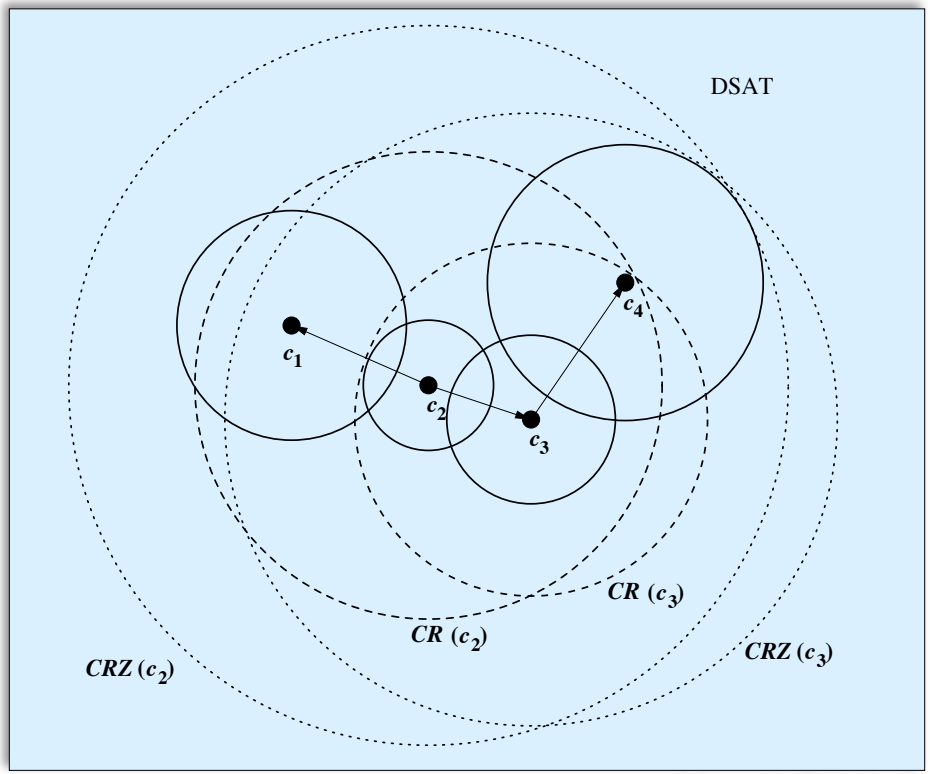
DSC

- El Conjunto Dinámico de Clusters (DSC) en lugar de una lista de centros en memoria principal mantiene un DSAT de centros.
- Se elige como el clúster más adecuado para inserción al 1-NN(x) en el DSAT.
- Admite inserciones y eliminaciones.

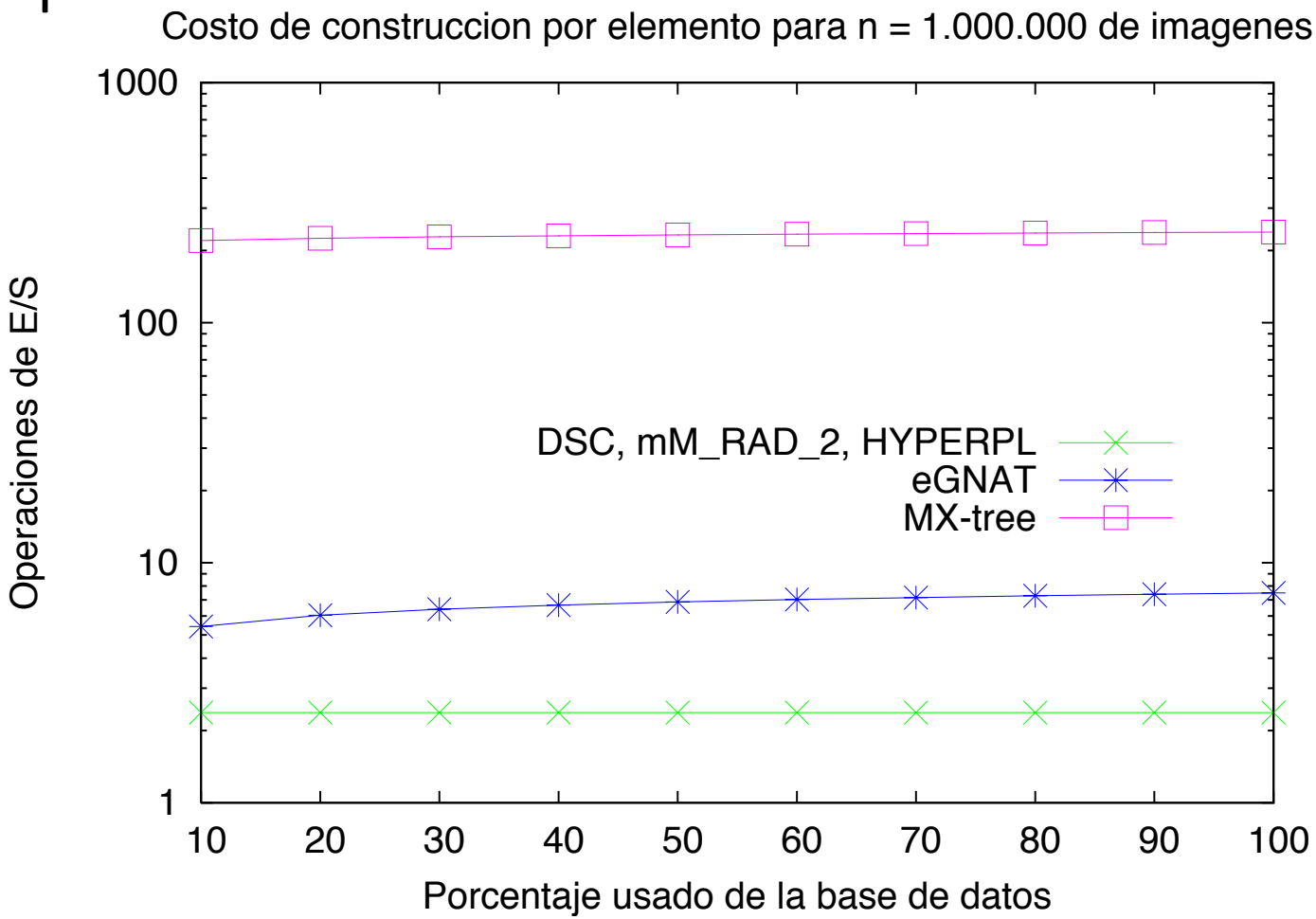
DSC


Memoria Principal
c_2
c_3
c_1
c_4
DSAT
Clusters / Páginas de Disco
c_1
c_4
c_2
c_3
Memoria Secundaria

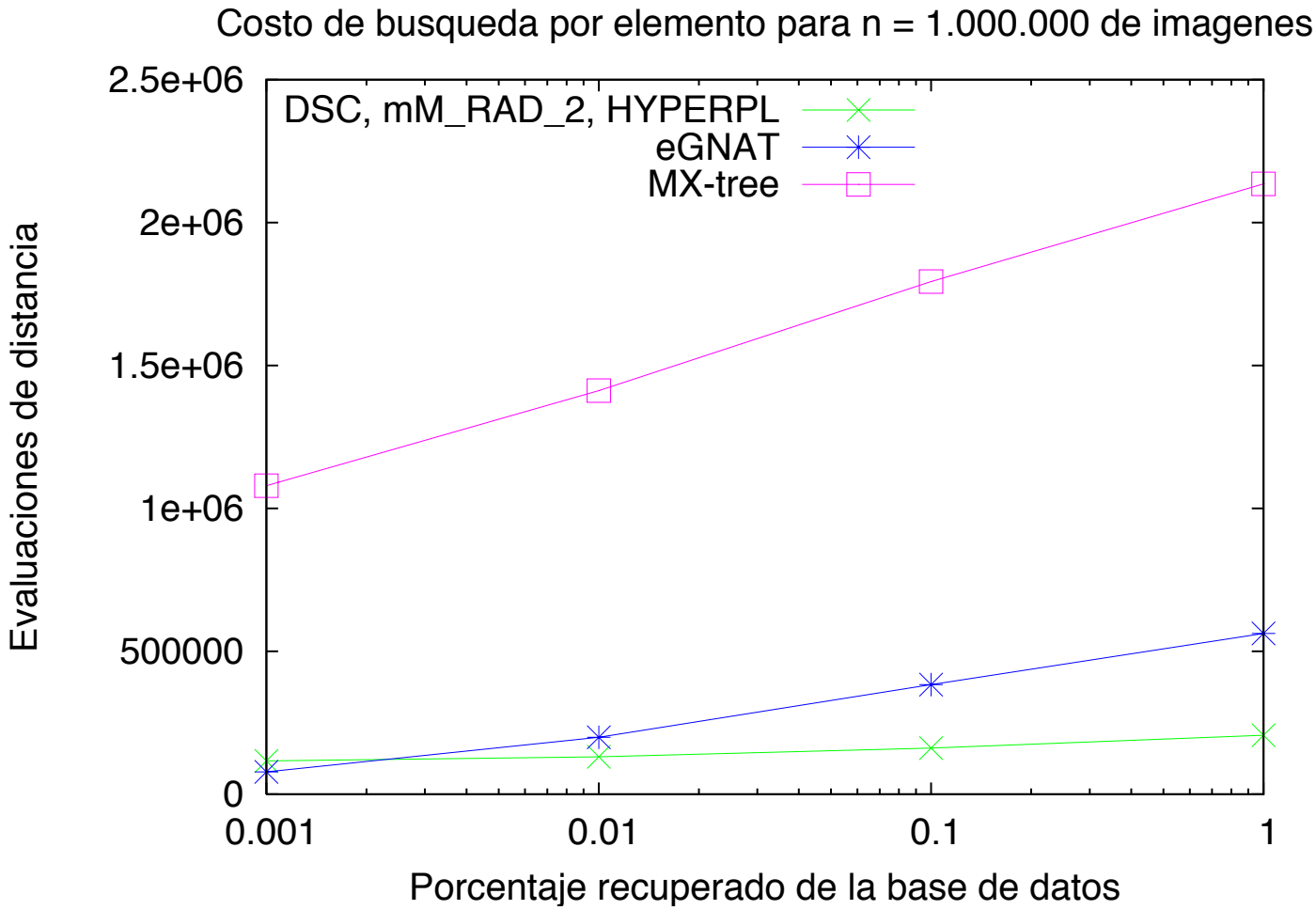
DSC



Comparación



Comparación



Comparación