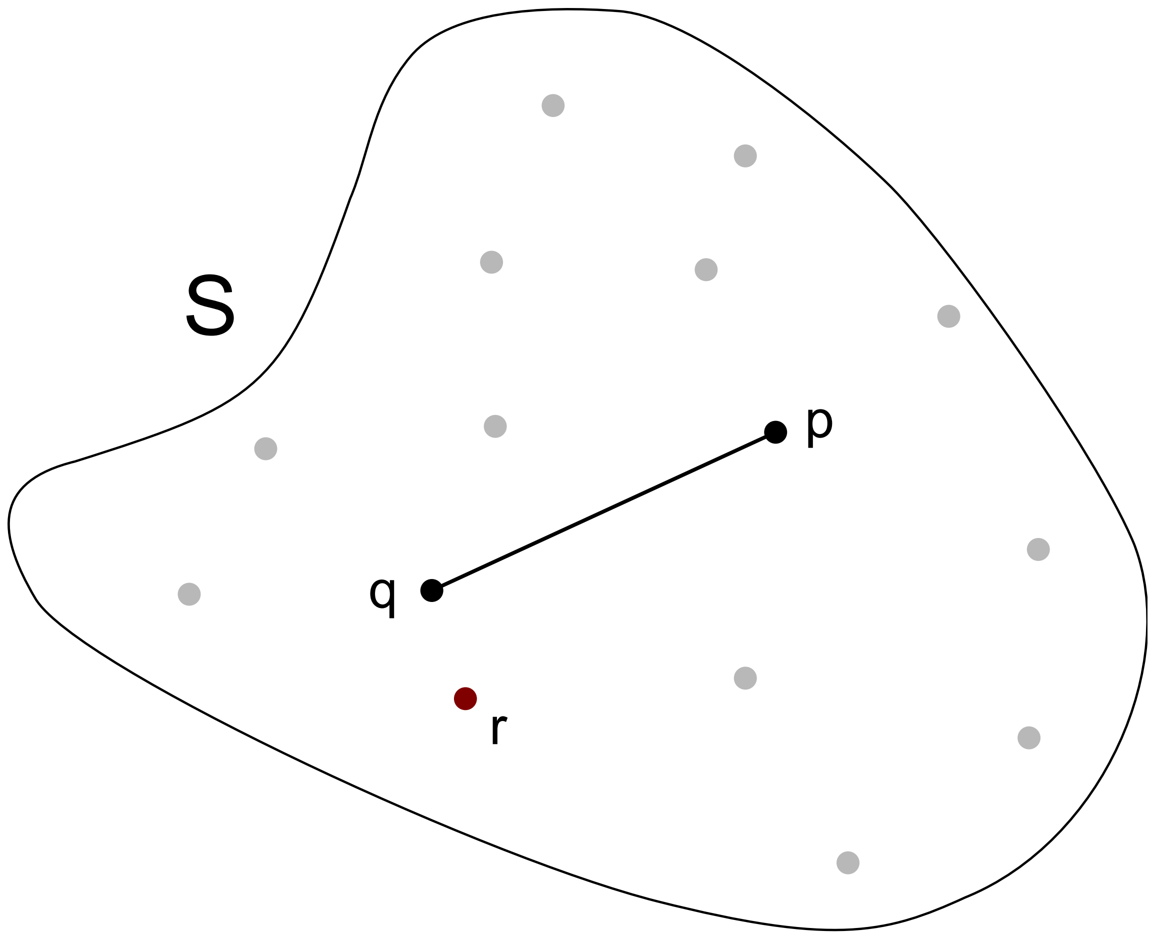
Sesión 12.2: Distance-Based Indexing Methods

CS3102 EDA

Índice

1. Approximate Voronoi Diagram (AVD)

ϵ-nearest neighbor *(ϵ-NN)*



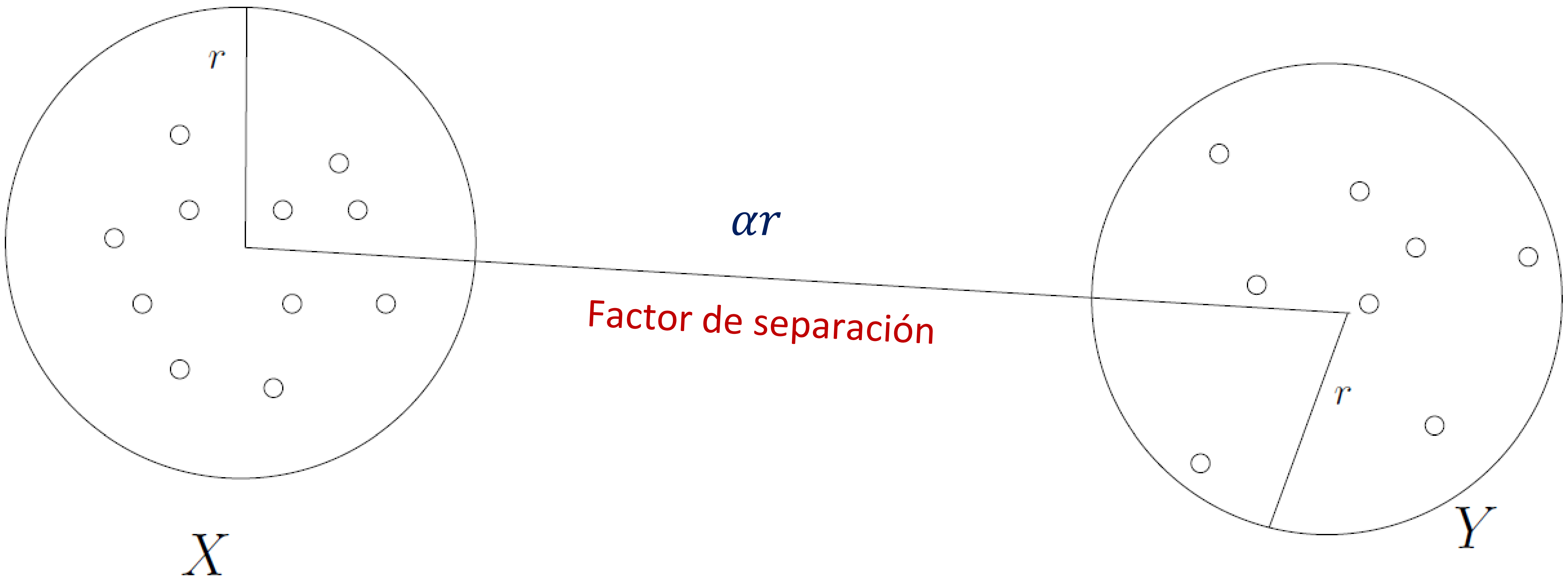
Candidato a vecino más cercano $\boldsymbol{o'}$

Vecino más cercano real: $\boldsymbol{o}$

El punto p es ϵ-NN de q si:

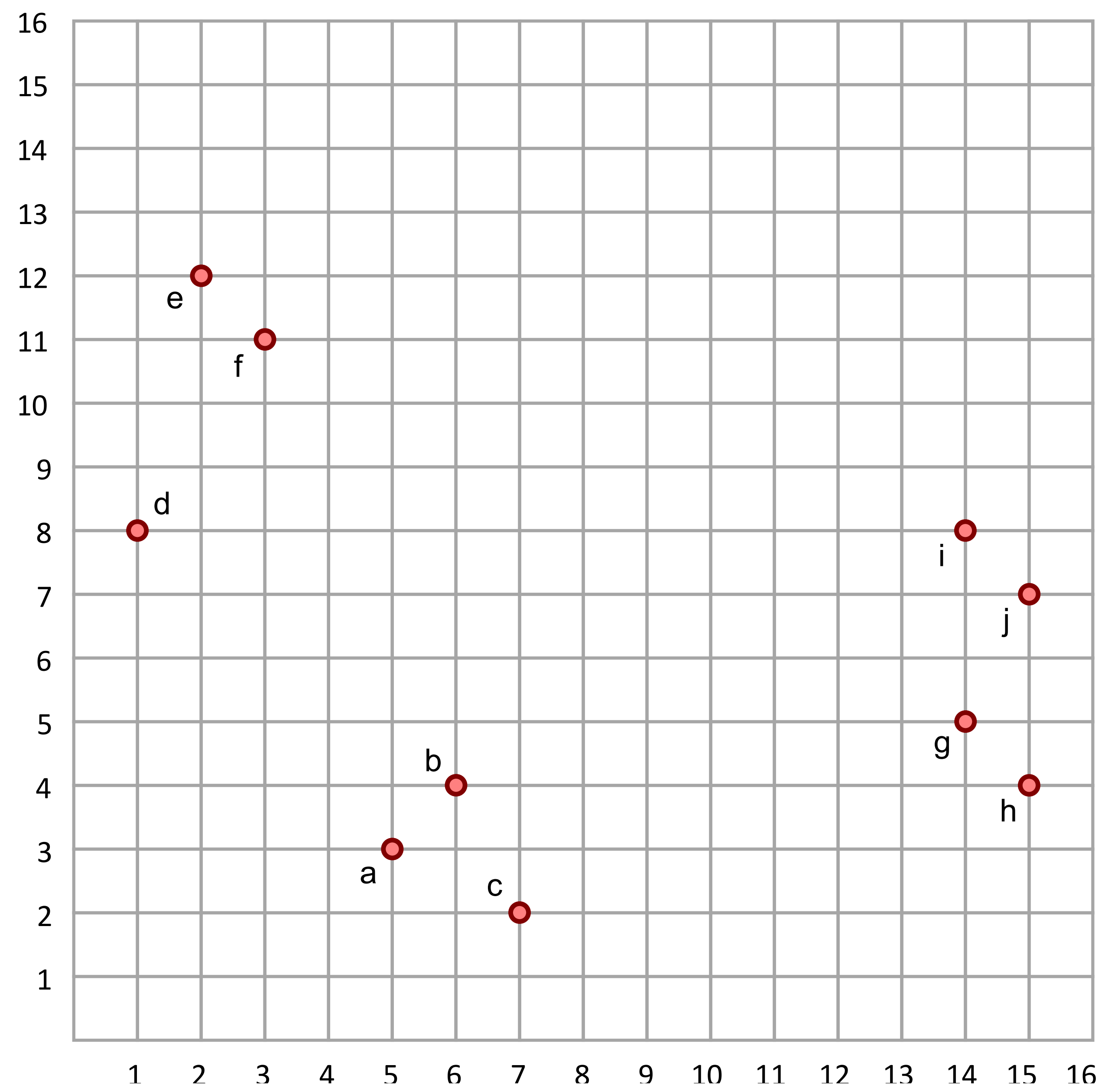
$$d(q, o') \leq (1 + \epsilon) \cdot d(q, o)$$

Well-Separated Pair *Decomposition (WSPD)*



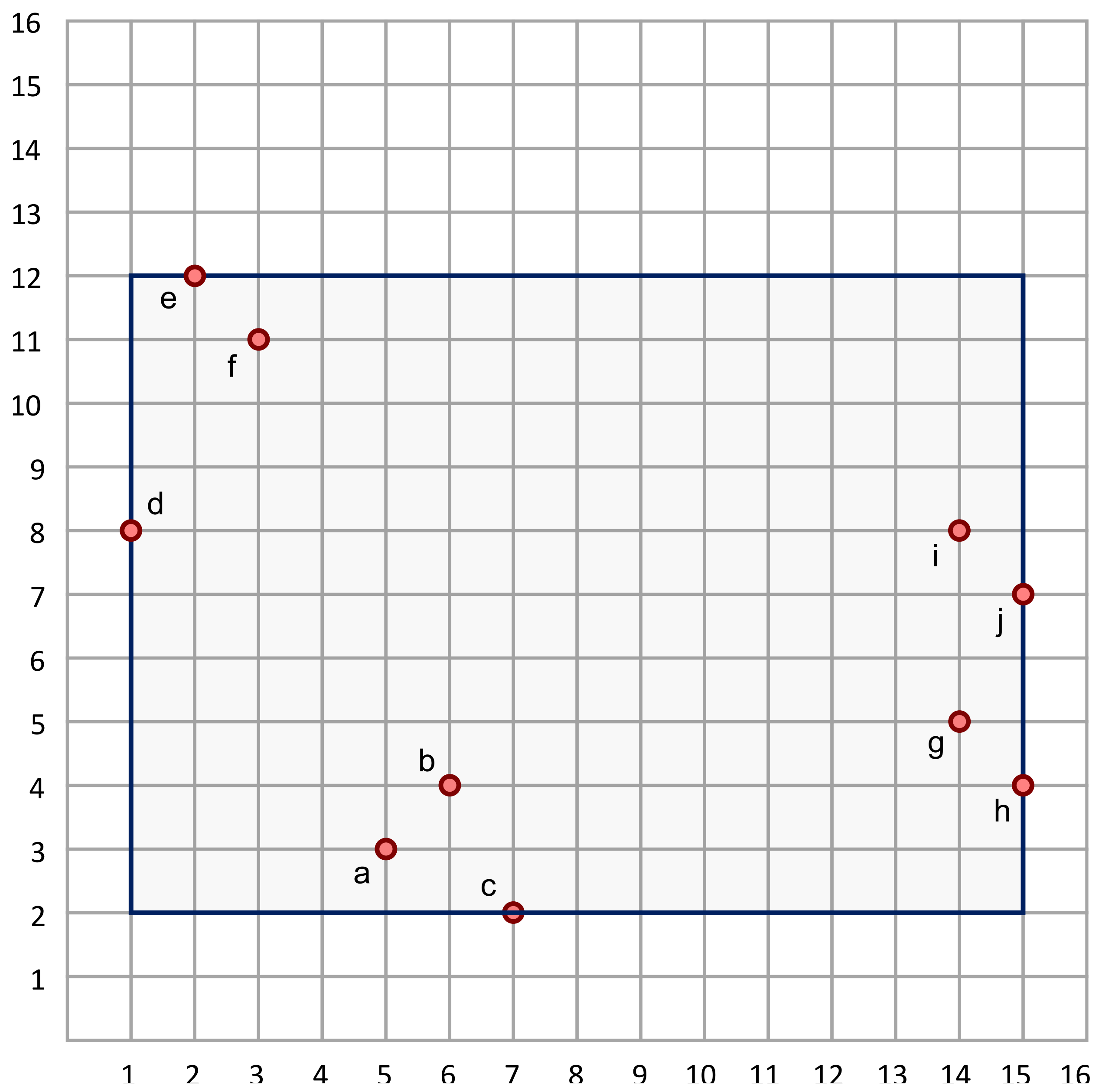
Fair split *tree*

Dividir el espacio recursivamente por la dimensión más grande del MBB.
Construir un árbol binario a partir de las particiones.



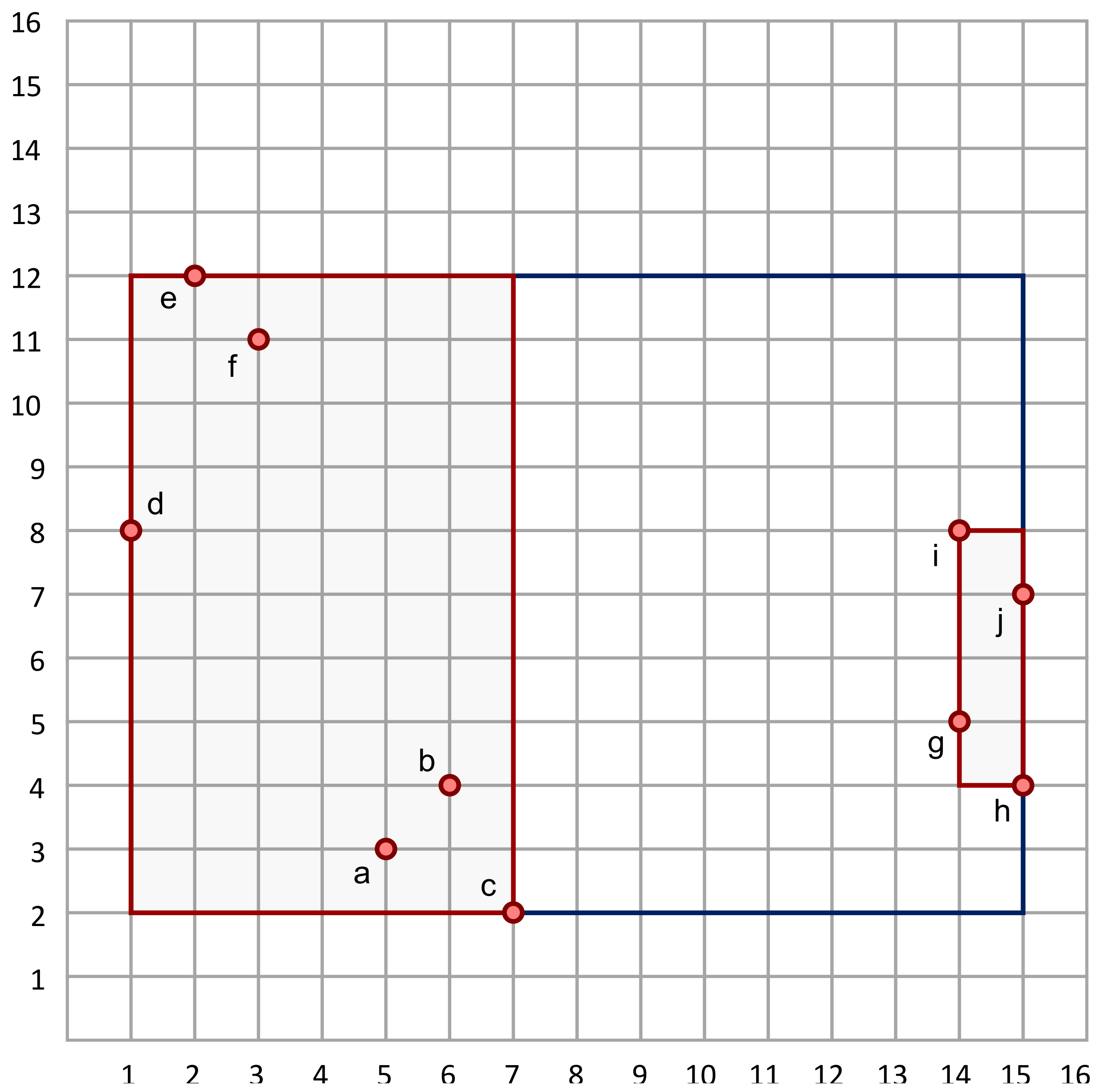
Fair split *tree*

Dividir el espacio recursivamente por la dimensión más grande del MBB.
Construir un árbol binario a partir de las particiones.



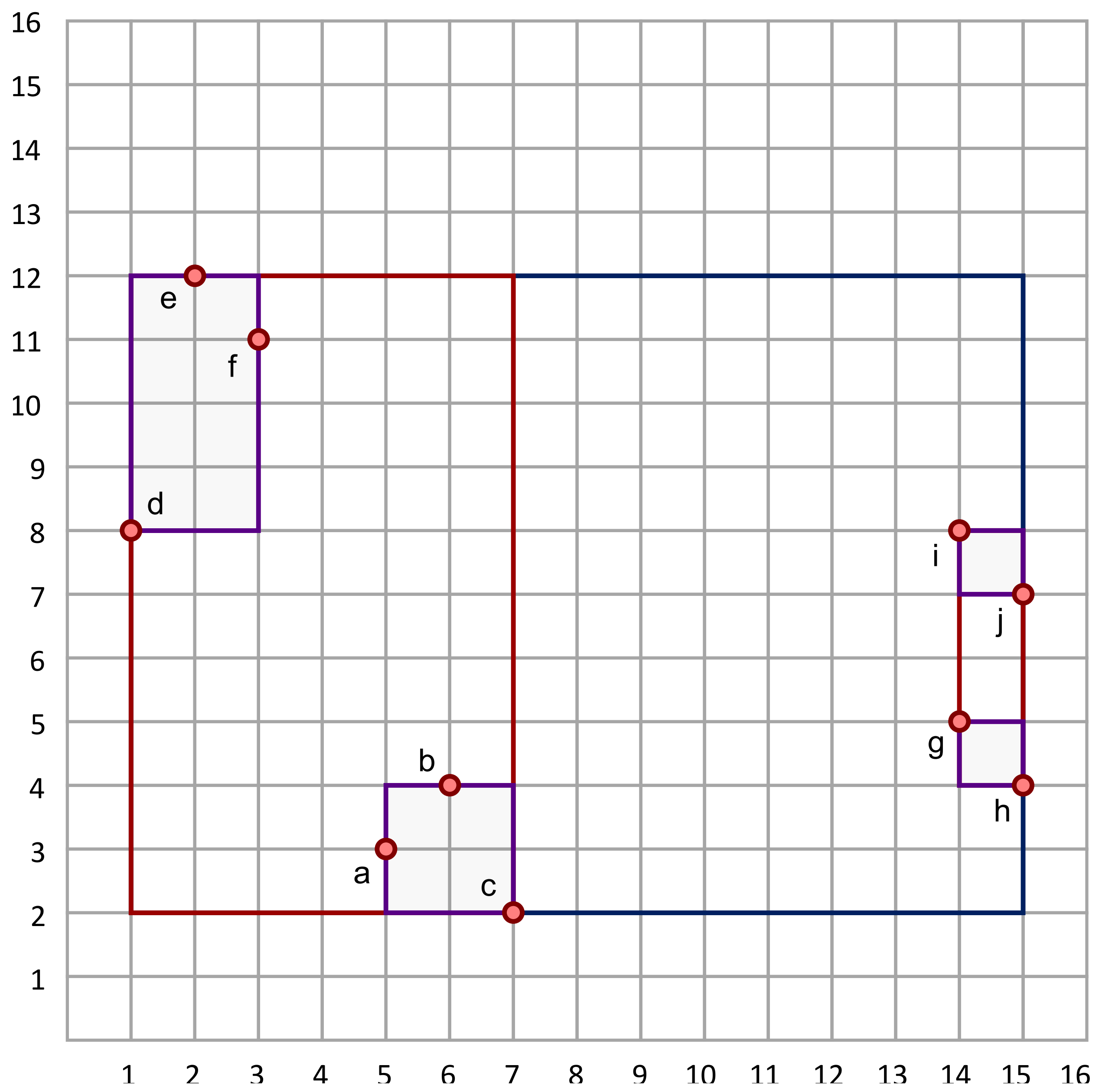
Fair split *tree*

Dividir el espacio recursivamente por la dimensión más grande del MBB.
Construir un árbol binario a partir de las particiones.



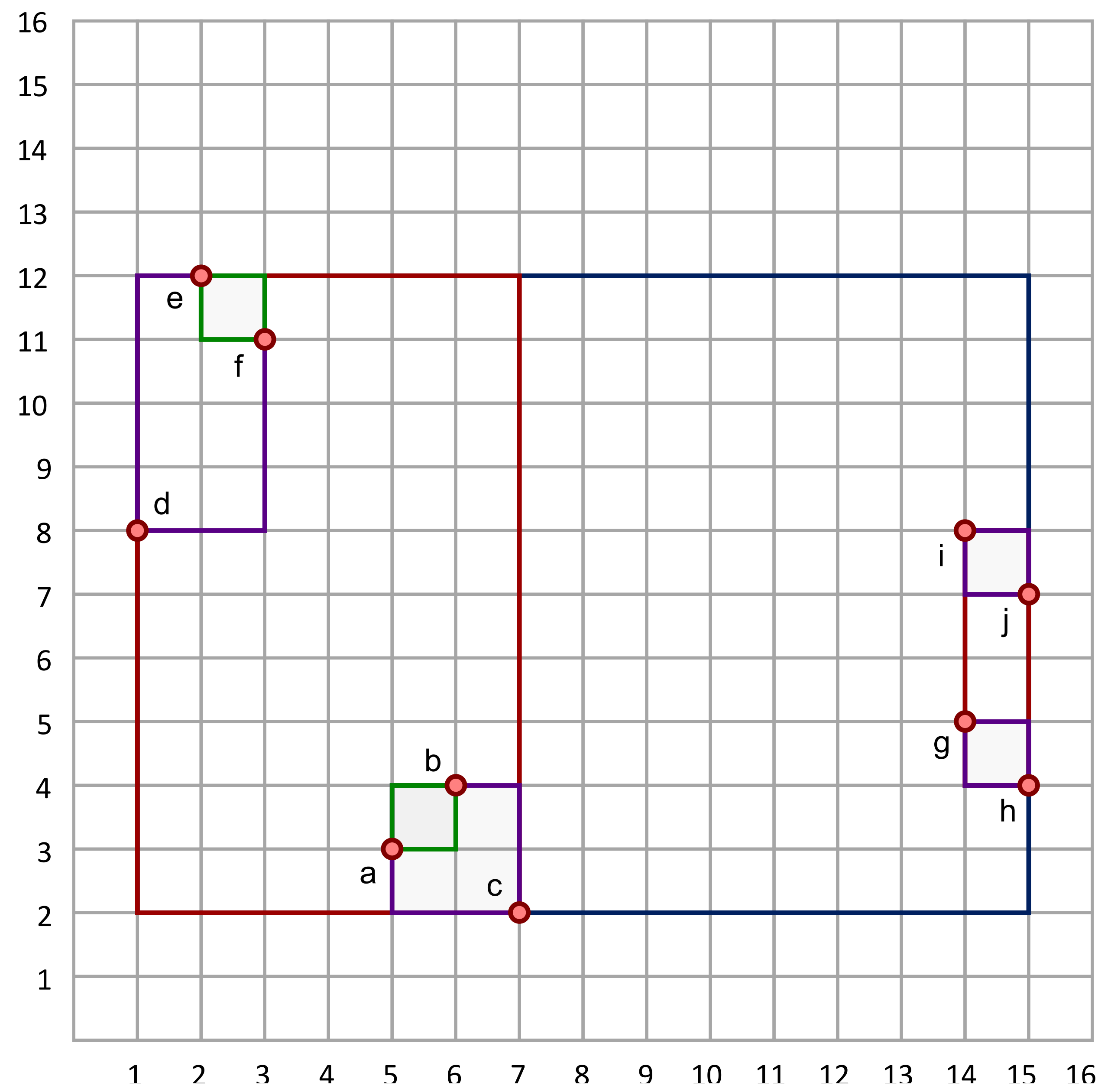
Fair split *tree*

Dividir el espacio recursivamente por la dimensión más grande del MBB.
Construir un árbol binario a partir de las particiones.

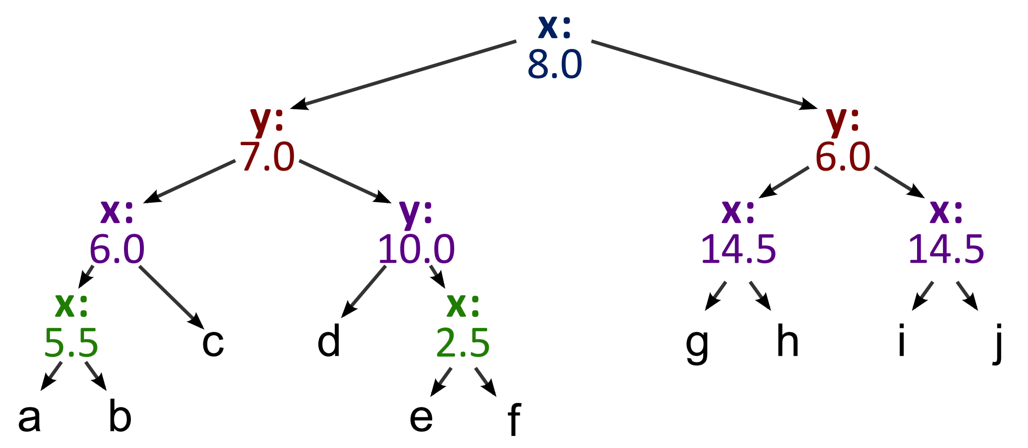


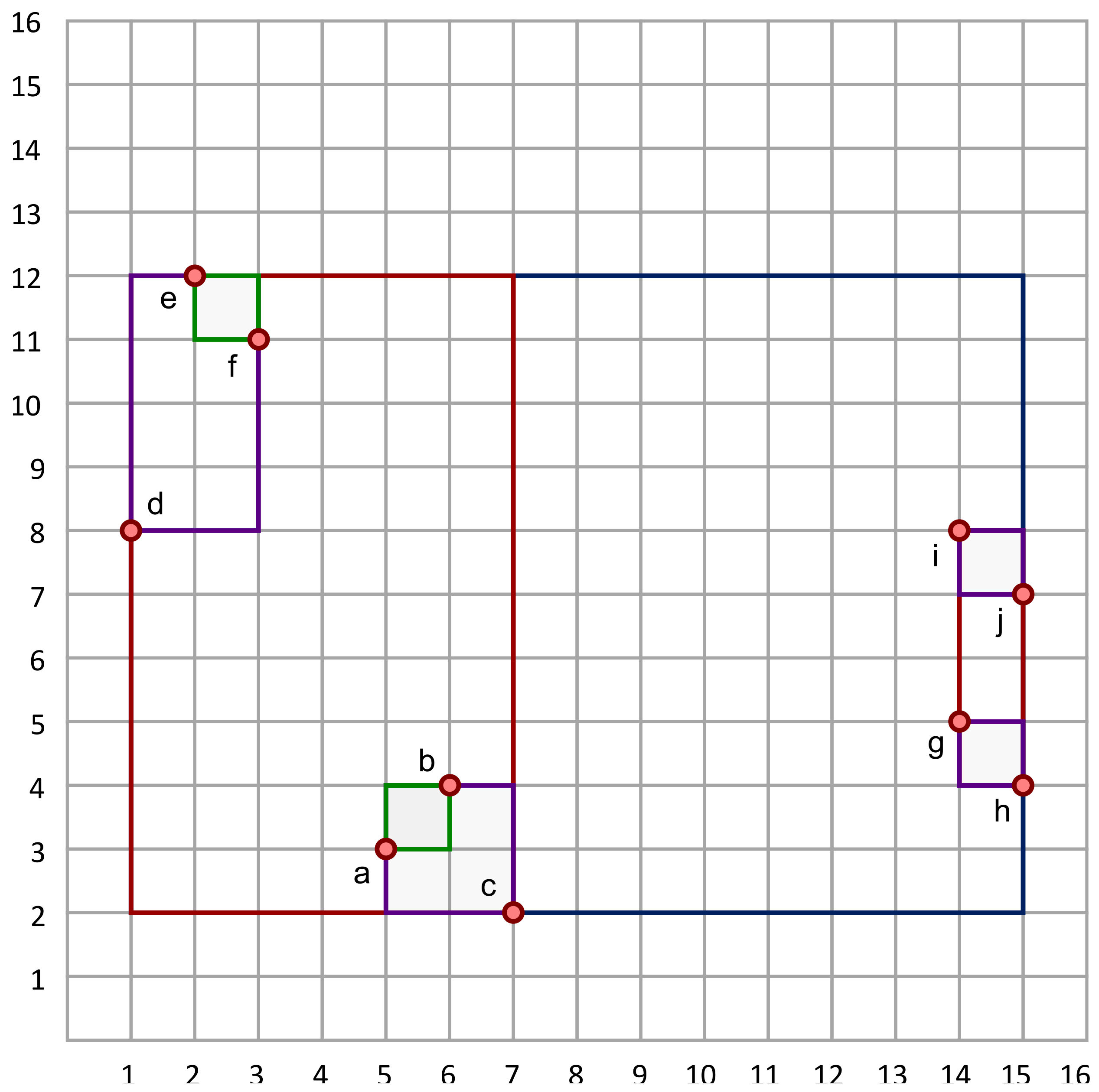
Fair split *tree*

Dividir el espacio recursivamente por la dimensión más grande del MBB.
Construir un árbol binario a partir de las particiones.

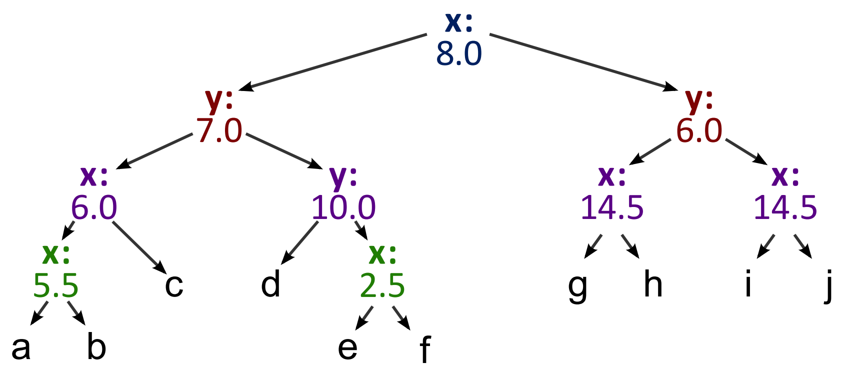


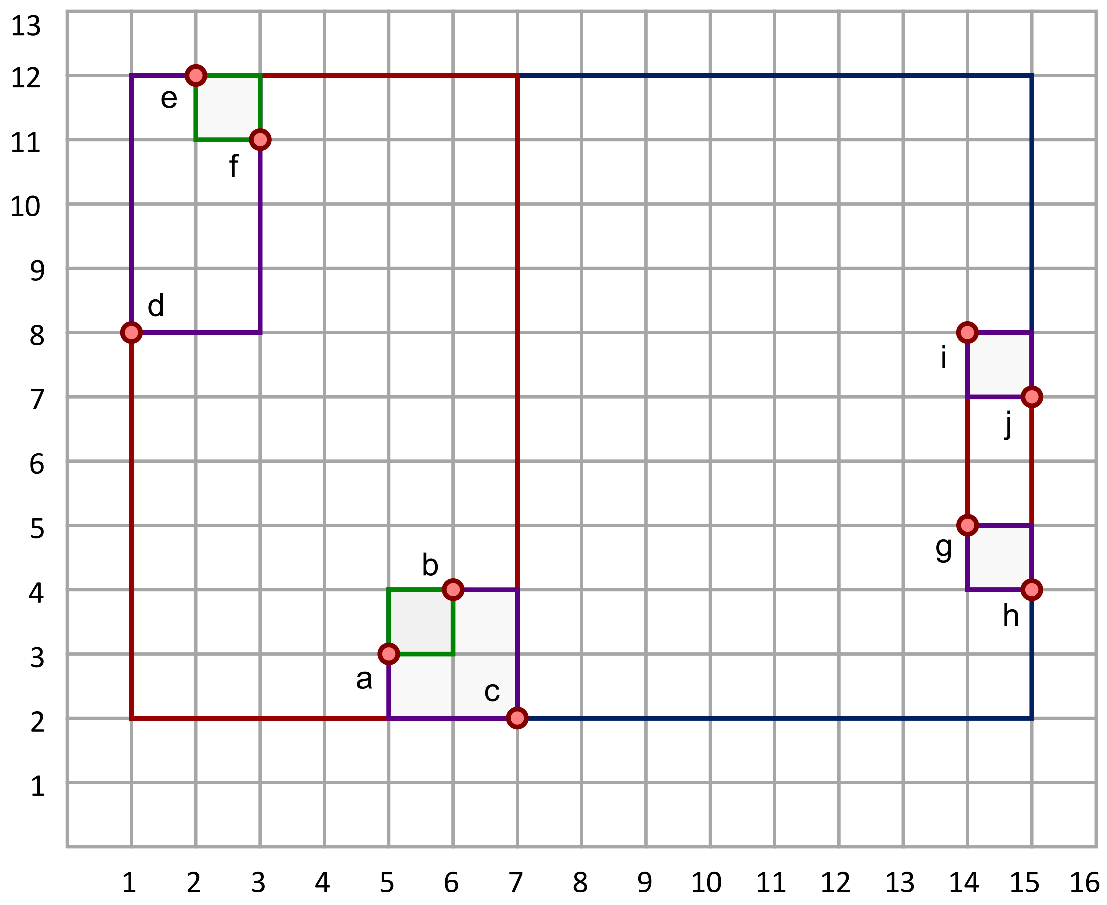
Fair split *tree*





Well-Separated Pair *Decomposition (WSPD)*

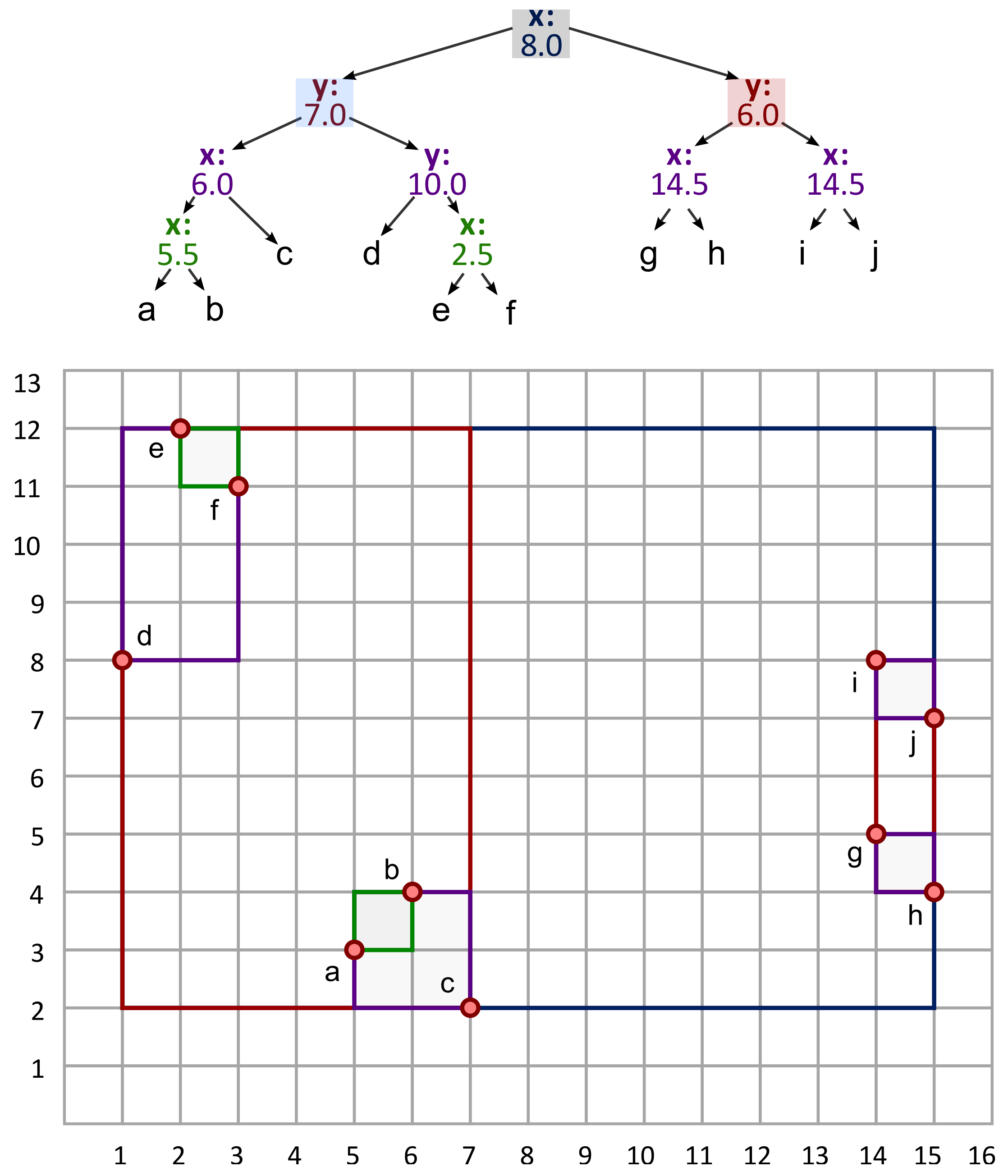




Revisar los hijos de todos los nodos internos

Verificar si son WSP

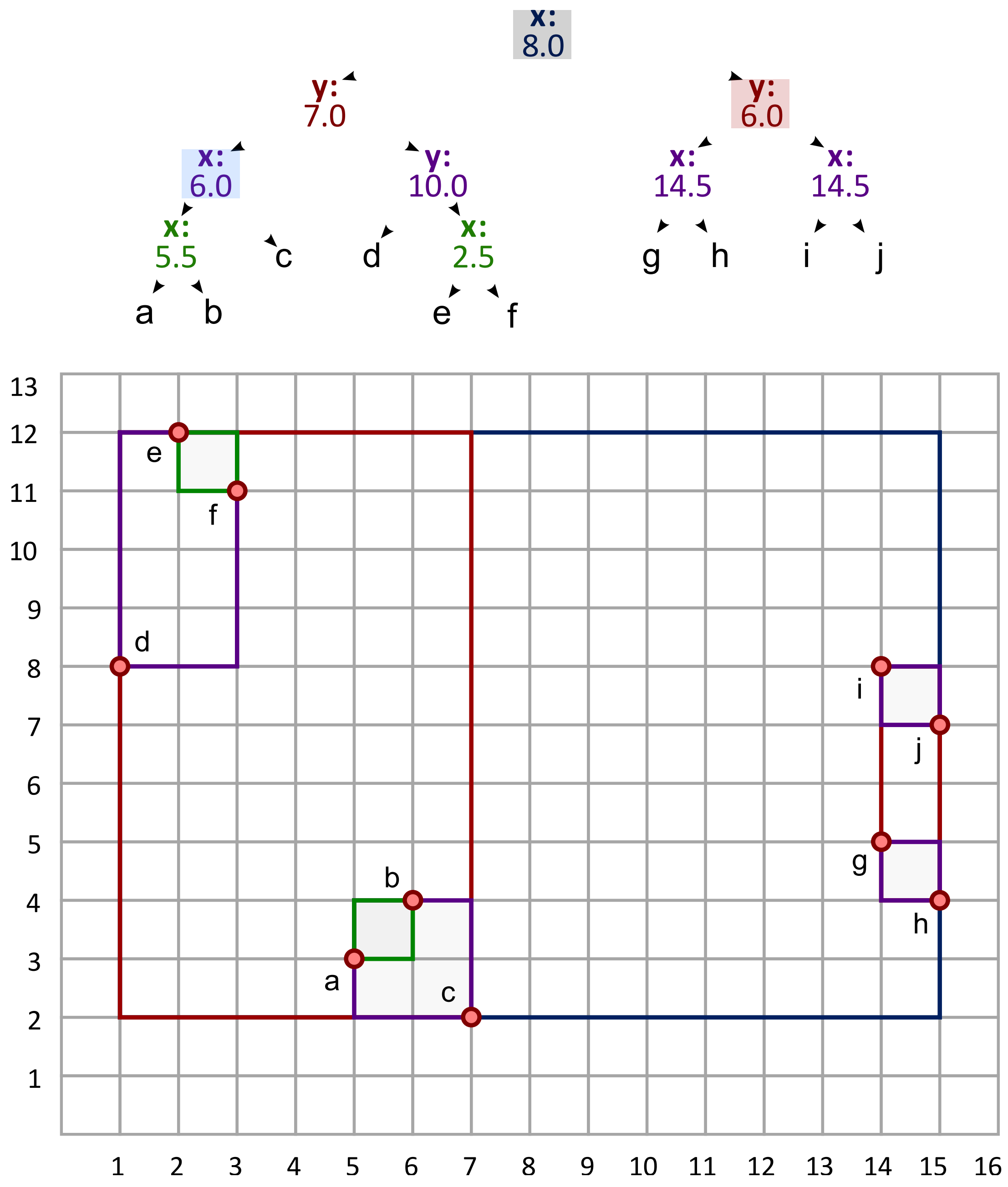
Well-Separated Pair *Decomposition (WSPD)*



Revisar los hijos de todos los nodos internos

Verificar si son WSP

Well-Separated Pair *Decomposition (WSPD)*

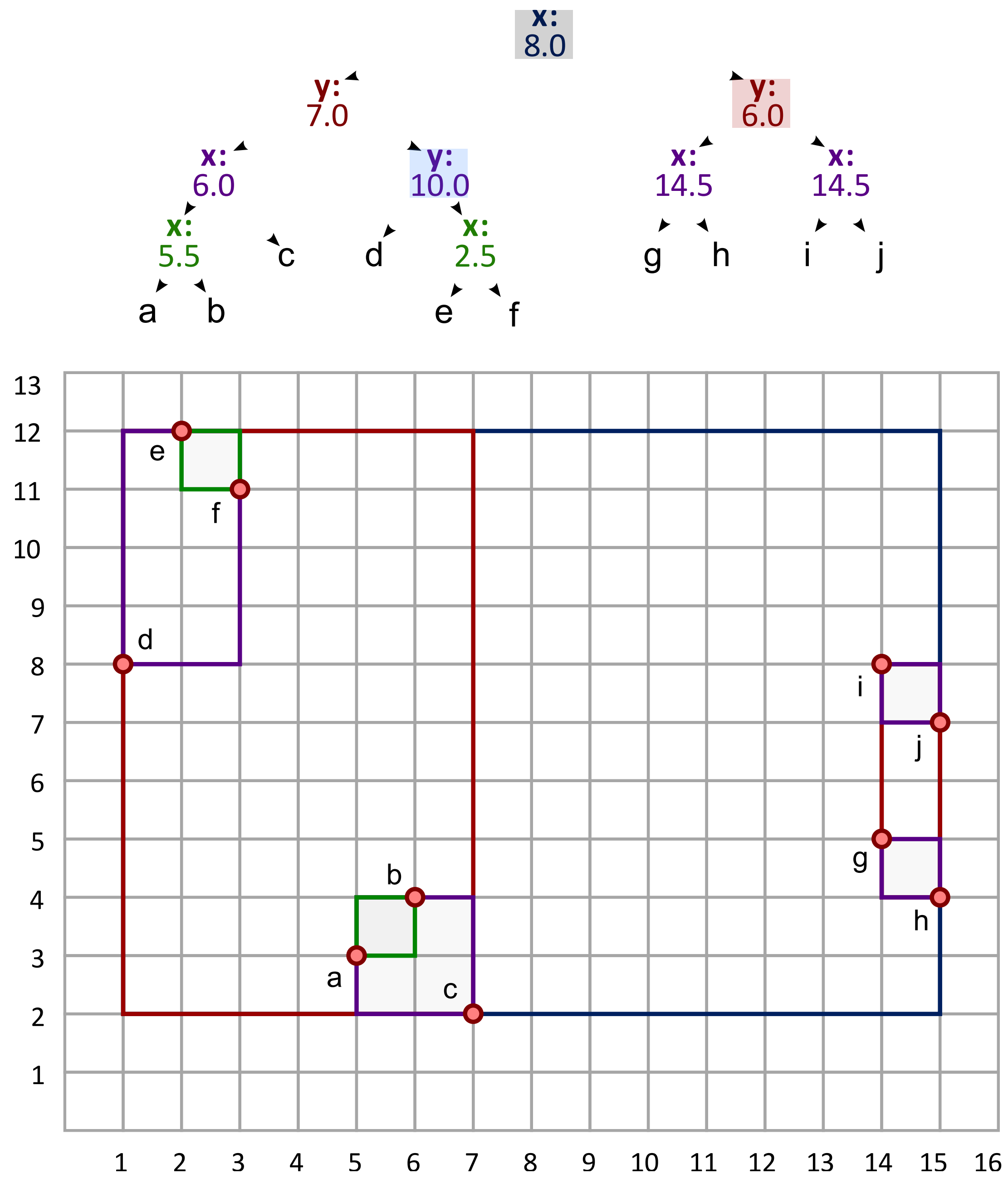


Revisar los hijos de todos los nodos internos

Verificar si son WSP

$$\{a, b, c\}, \{g, h, i, j\}$$

Well-Separated Pair *Decomposition (WSPD)*



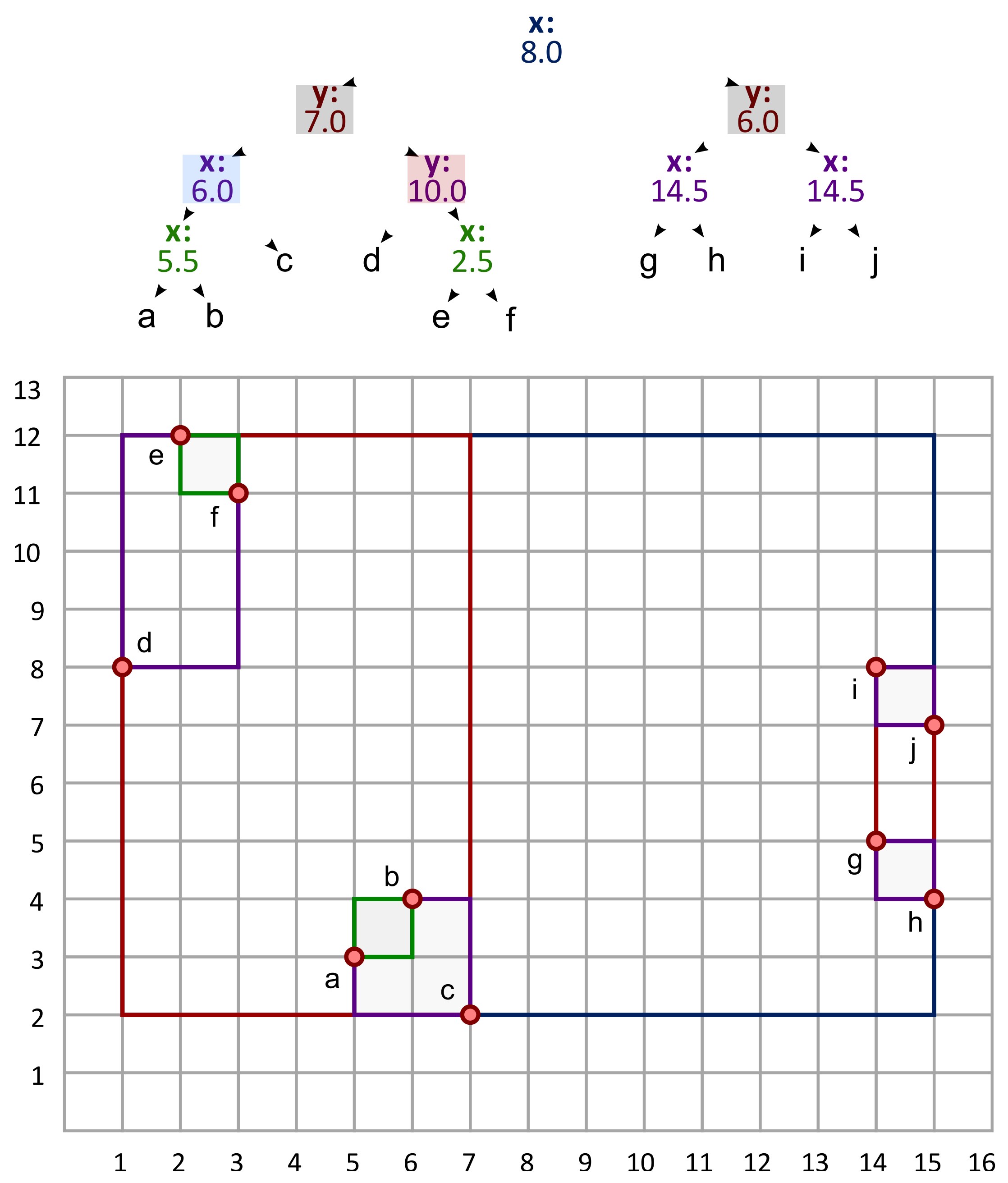
Revisar los hijos de todos los nodos internos

Verificar si son WSP

$$\{a, b, c\}, \{g, h, i, j\}$$

$$\{d, e, f\}, \{g, h, i, j\}$$

Well-Separated Pair *Decomposition (WSPD)*

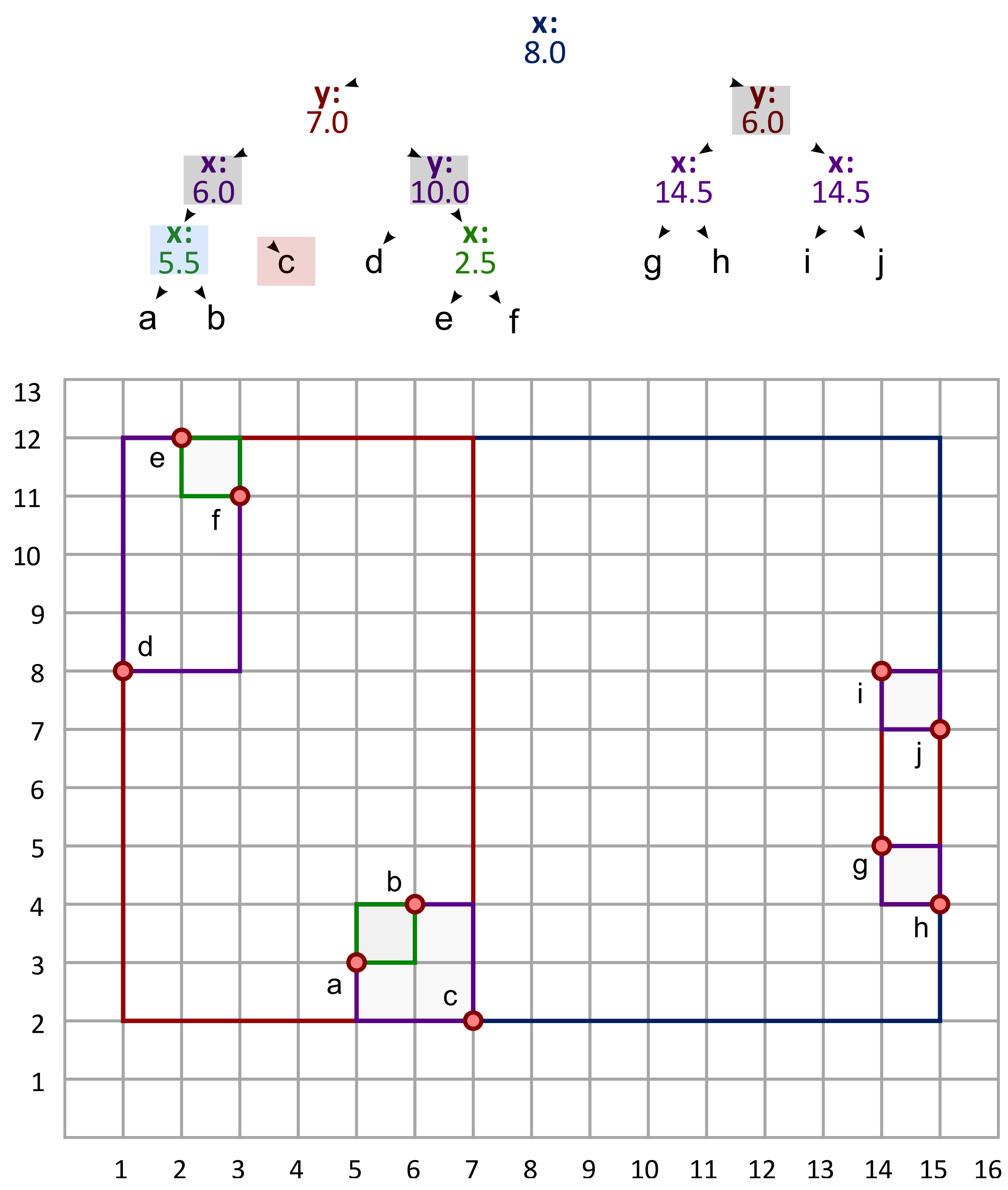


Revisar los hijos de todos los nodos internos

Verificar si son WSP

$$\{a, b, c\}, \{g, h, i, j\}$$
$$\{d, e, f\}, \{g, h, i, j\}$$
$$\{a, b, c\}, \{d, e, f\}$$

Well-Separated Pair *Decomposition (WSPD)*



Revisar los hijos de todos los nodos internos

Verificar si son WSP

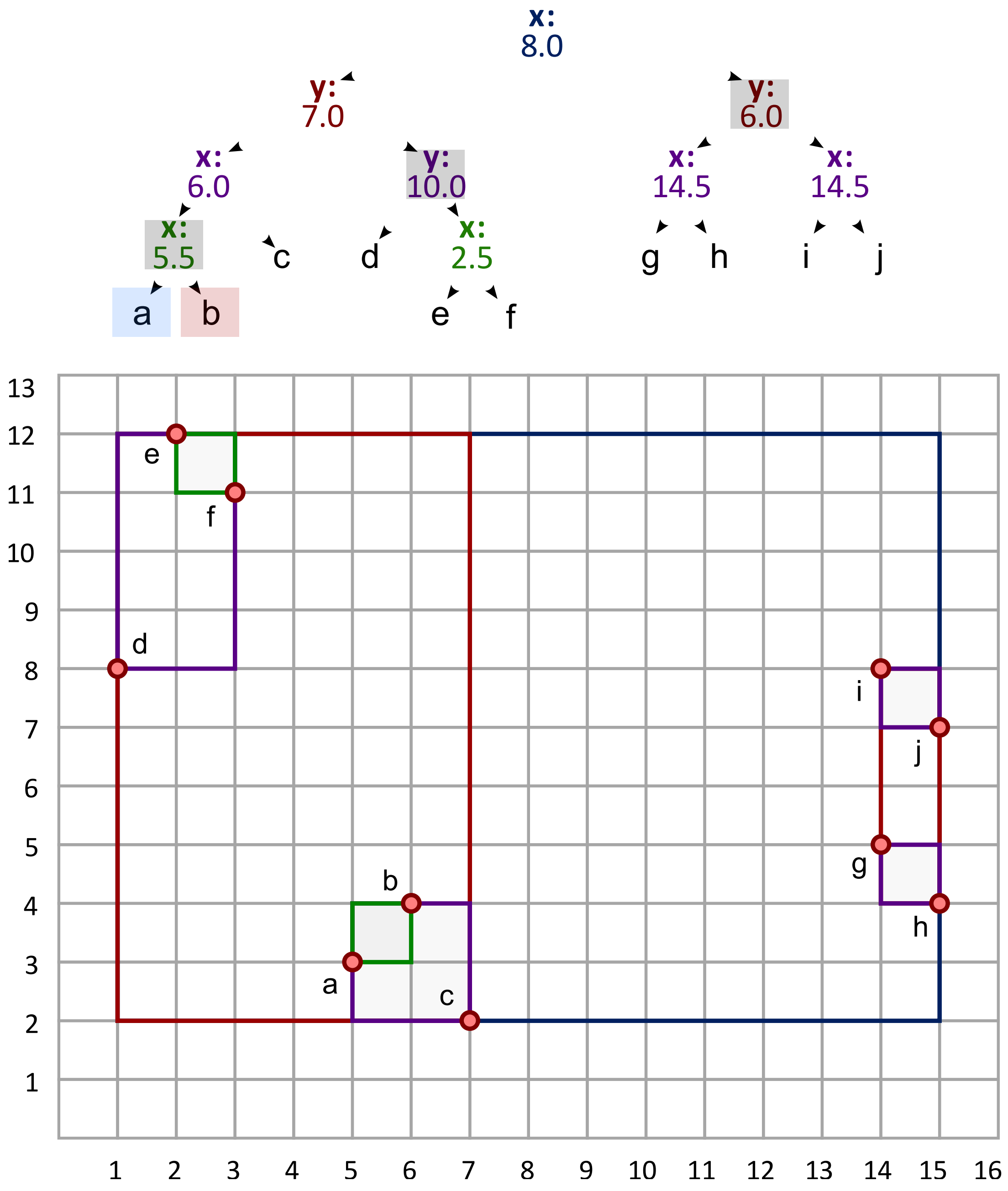
$$\{a, b, c\}, \{g, h, i, j\}$$

$$\{d, e, f\}, \{g, h, i, j\}$$

$$\{a, b, c\}, \{d, e, f\}$$

$$\{a, b\}, \{c\}$$

Well-Separated Pair *Decomposition (WSPD)*



Revisar los hijos de todos los nodos internos

Verificar si son WSP

$$\{a, b, c\}, \{g, h, i, j\}$$

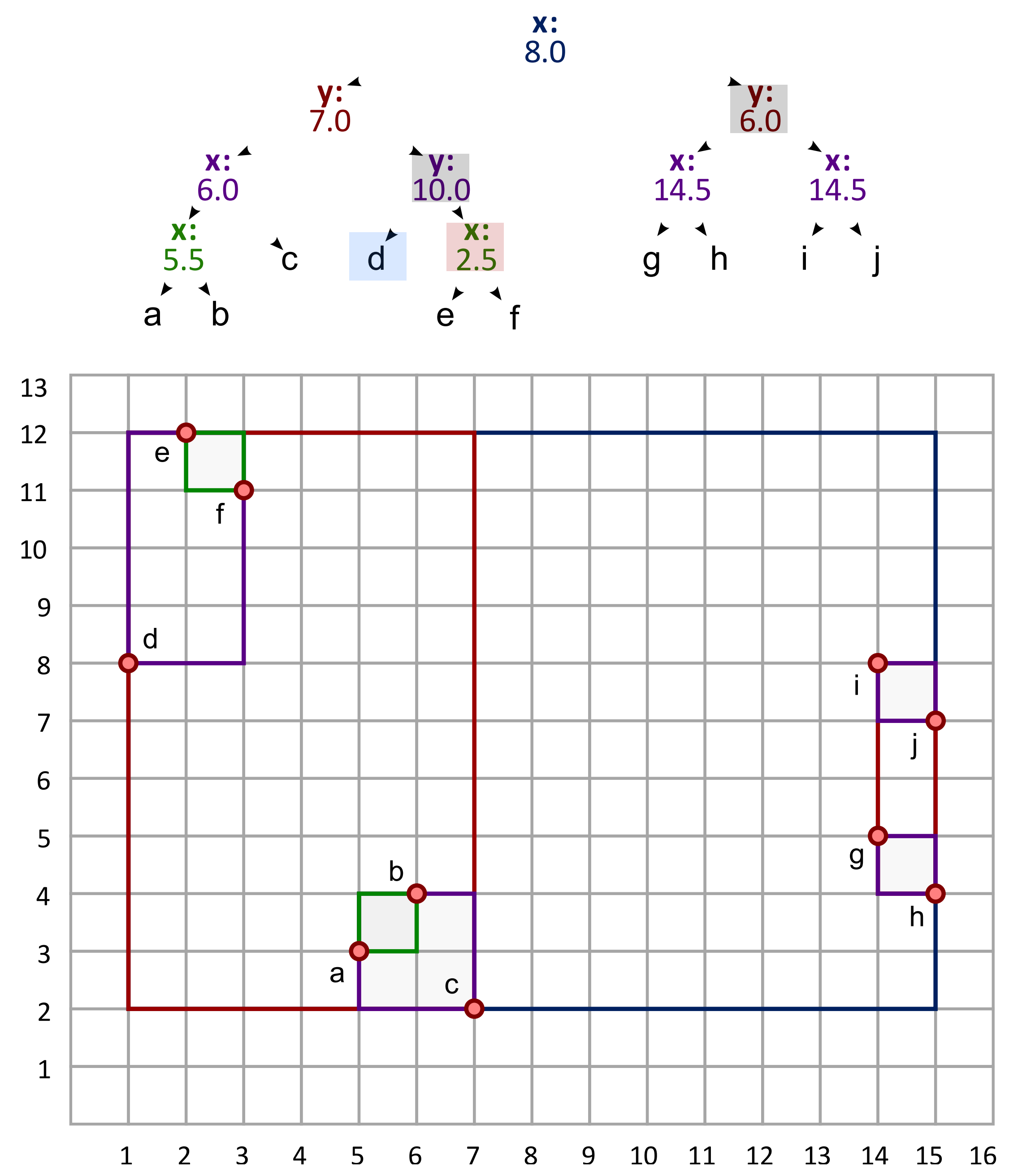
$$\{d, e, f\}, \{g, h, i, j\}$$

$$\{a, b, c\}, \{d, e, f\}$$

$$\{a, b\}, \{c\}$$

$$\{a\}, \{b\}$$

Well-Separated Pair *Decomposition (WSPD)*

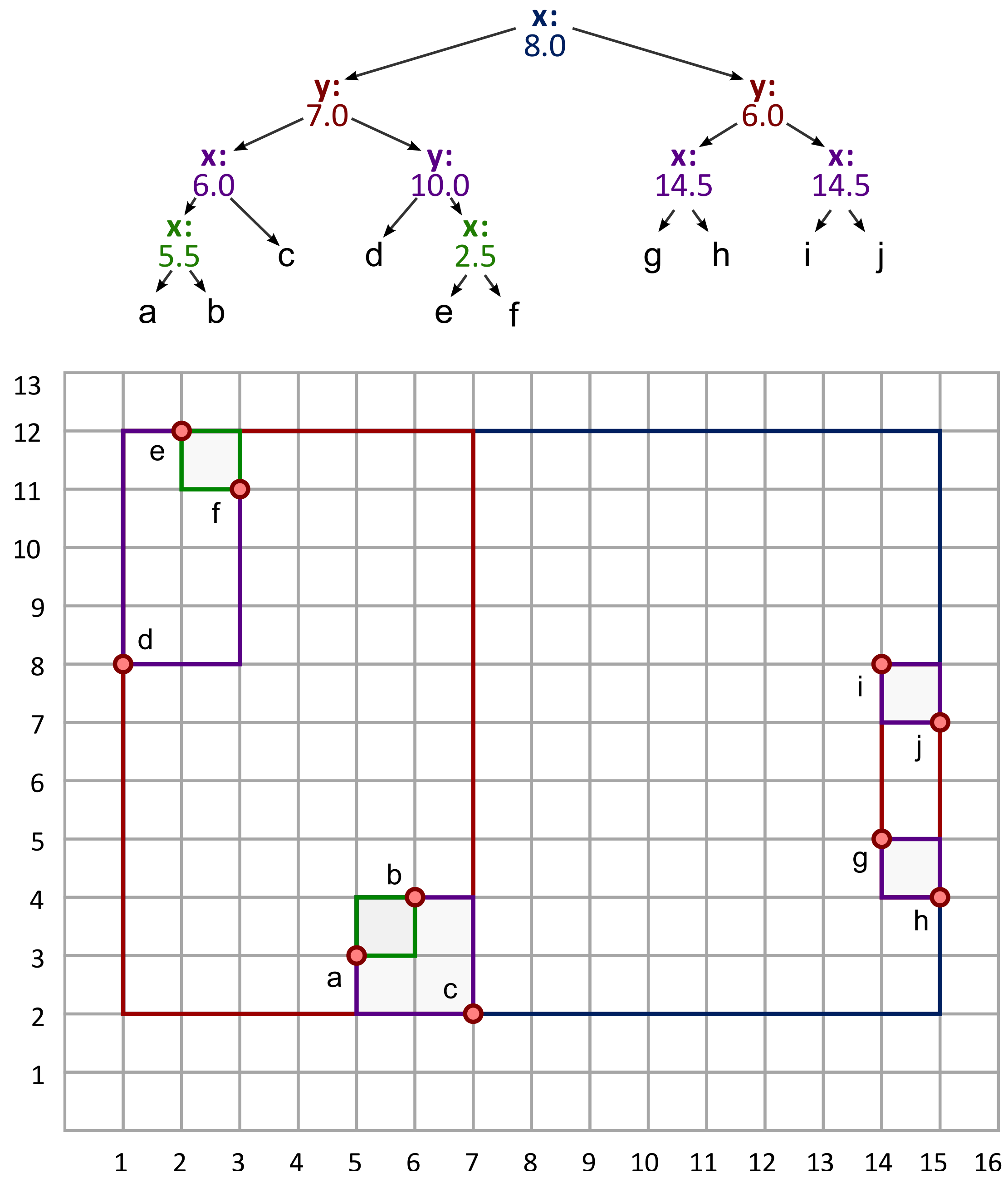


Revisar los hijos de todos los nodos internos

Verificar si son WSP

$$\{a,b,c\},\{g,h,i,j\}$$
$$\{d,e,f\},\{g,h,i,j\}$$
$$\{a,b,c\},\{d,e,f\}$$
$$\{a,b\},\{c\}$$
$$\{a\},\{b\}$$
$$\{d\},\{e,f\}$$

Well-Separated Pair *Decomposition (WSPD)*



Revisar los hijos de todos los nodos internos

Verificar si son WSP

$\{a, b, c\}, \{g, h, i, j\}$

$\{d, e, f\}, \{g, h, i, j\}$

$\{a, b, c\}, \{d, e, f\}$

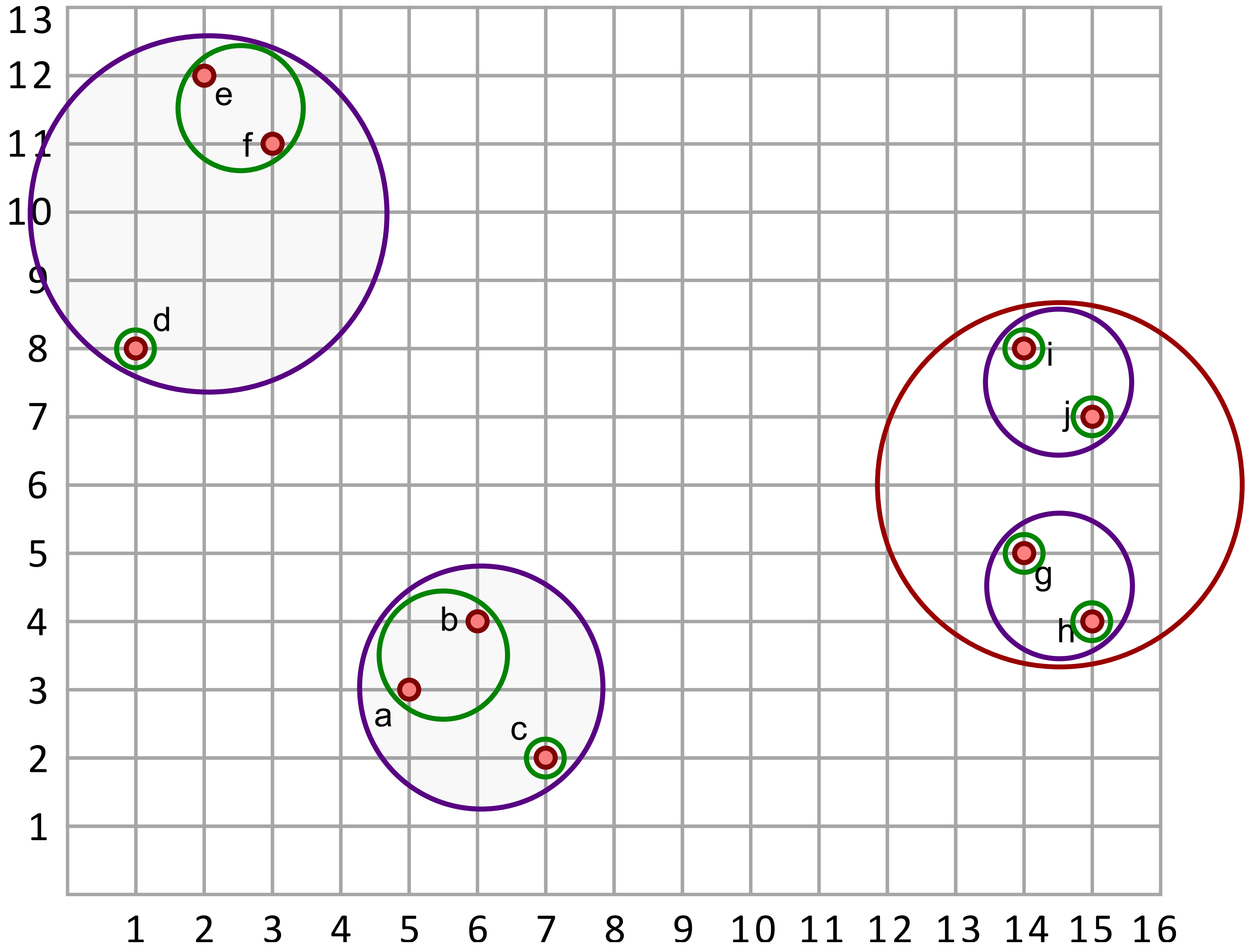
$\{a, b\}, \{c\}$

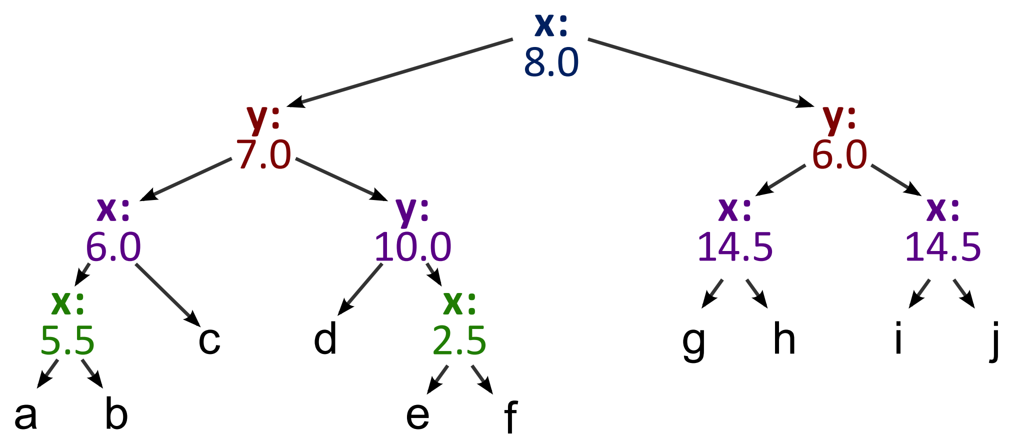
$\{a\}, \{b\}$

$\{d\}, \{e, f\}$

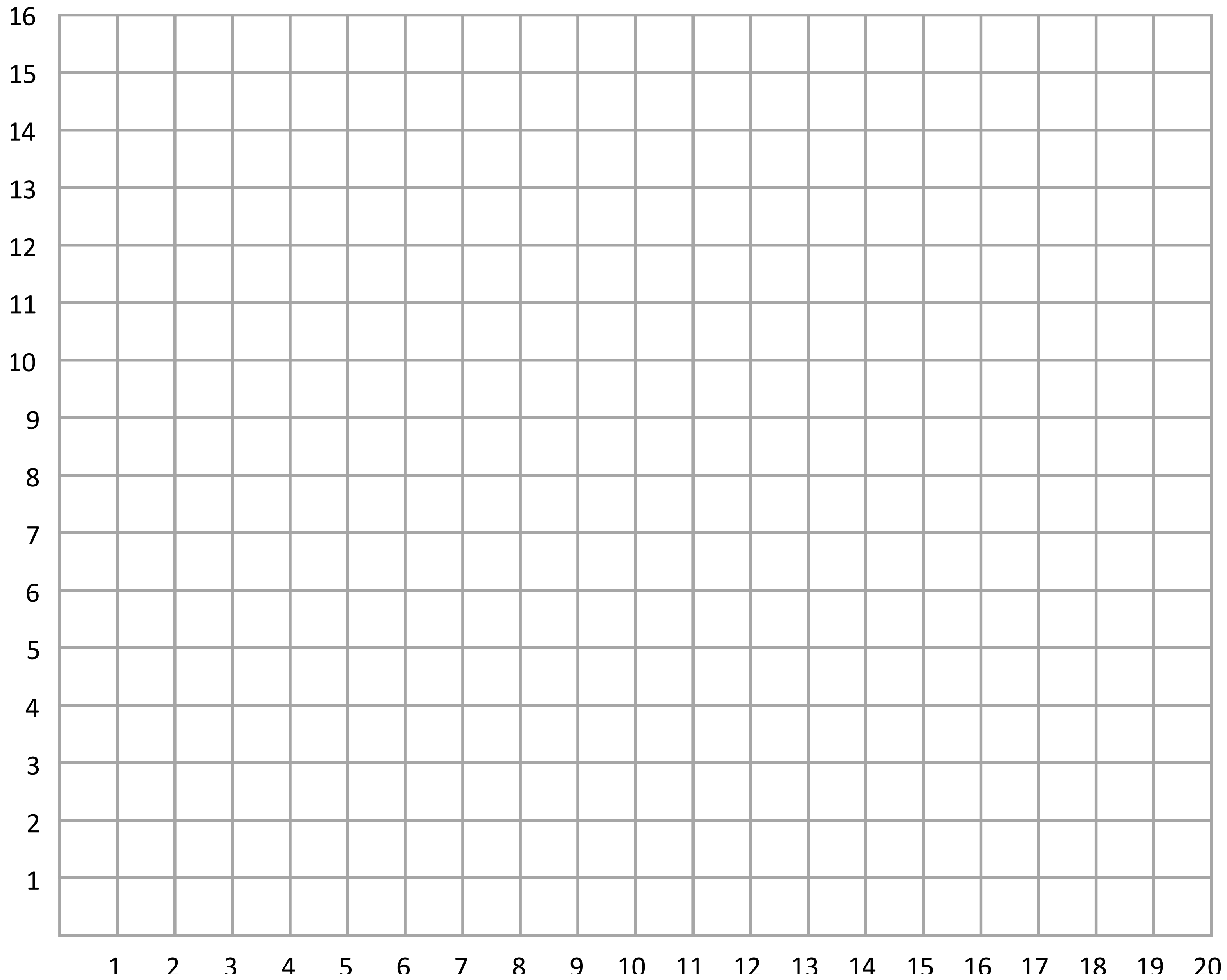
Continúe…

Well-Separated Pair *Decomposition (WSPD)*

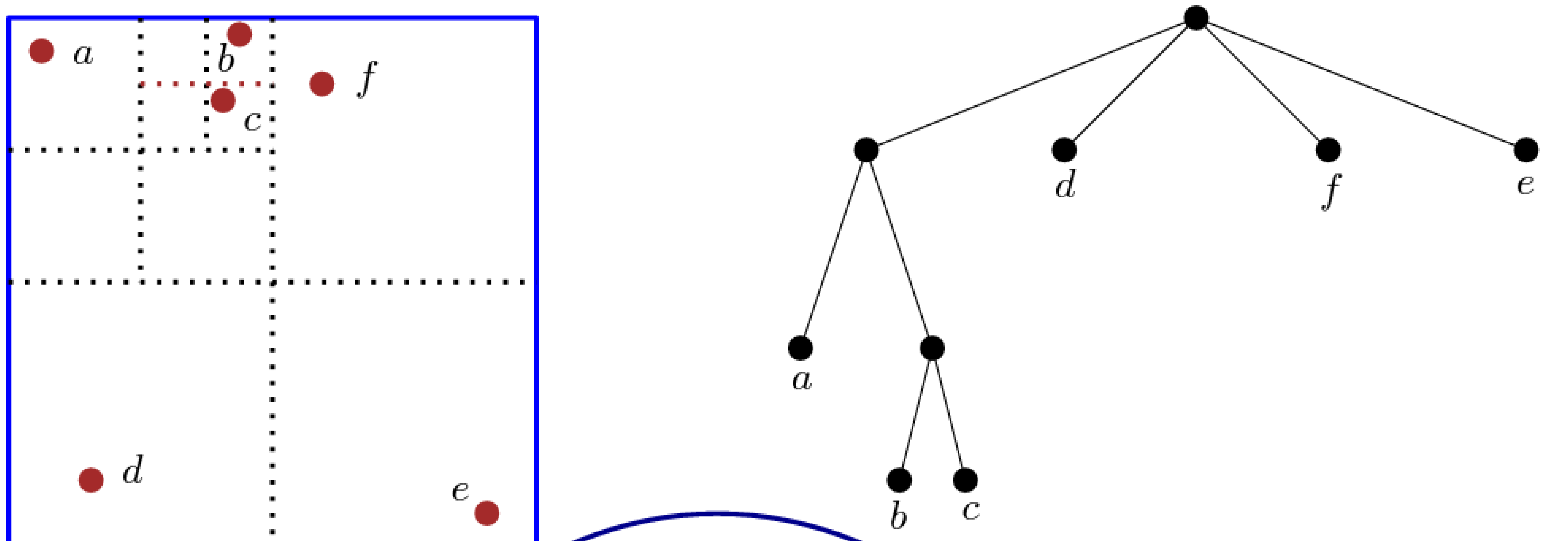


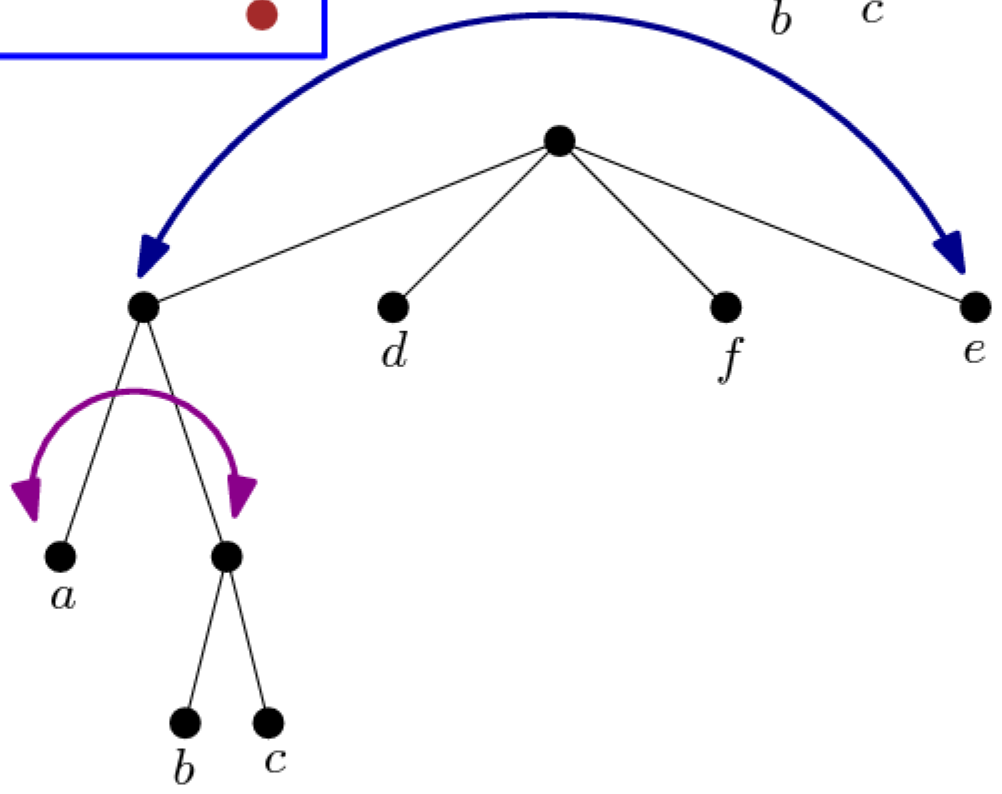


Well-Separated Pair *Decomposition (WSPD)*



Well-Separated Pair *Decomposition (WSPD)*





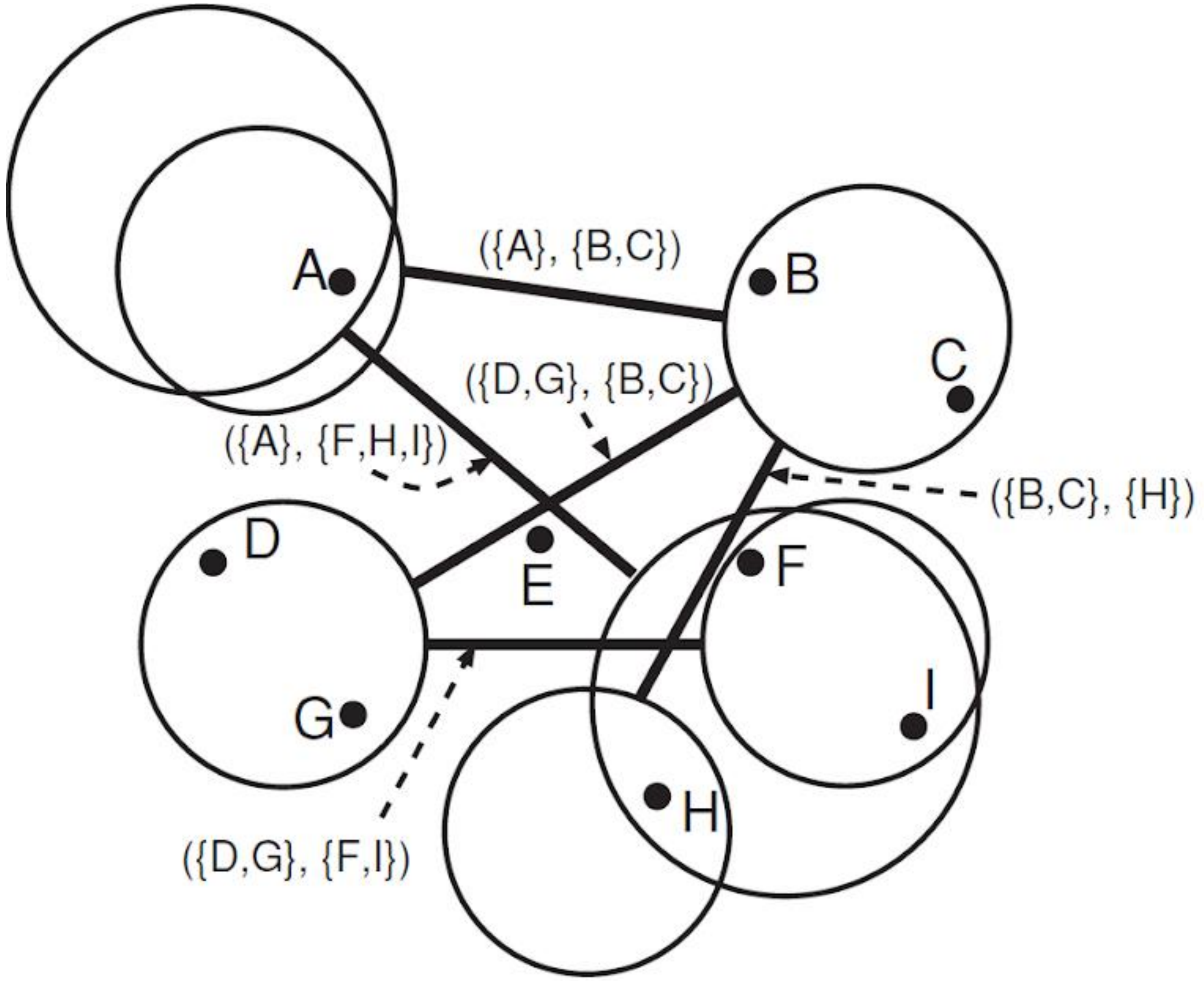
$A_1 = \{a, b, c\}, B_1 = \{e\}$

$A_1 = \{a\}, B_1 = \{b, c\}$

$\vdots$

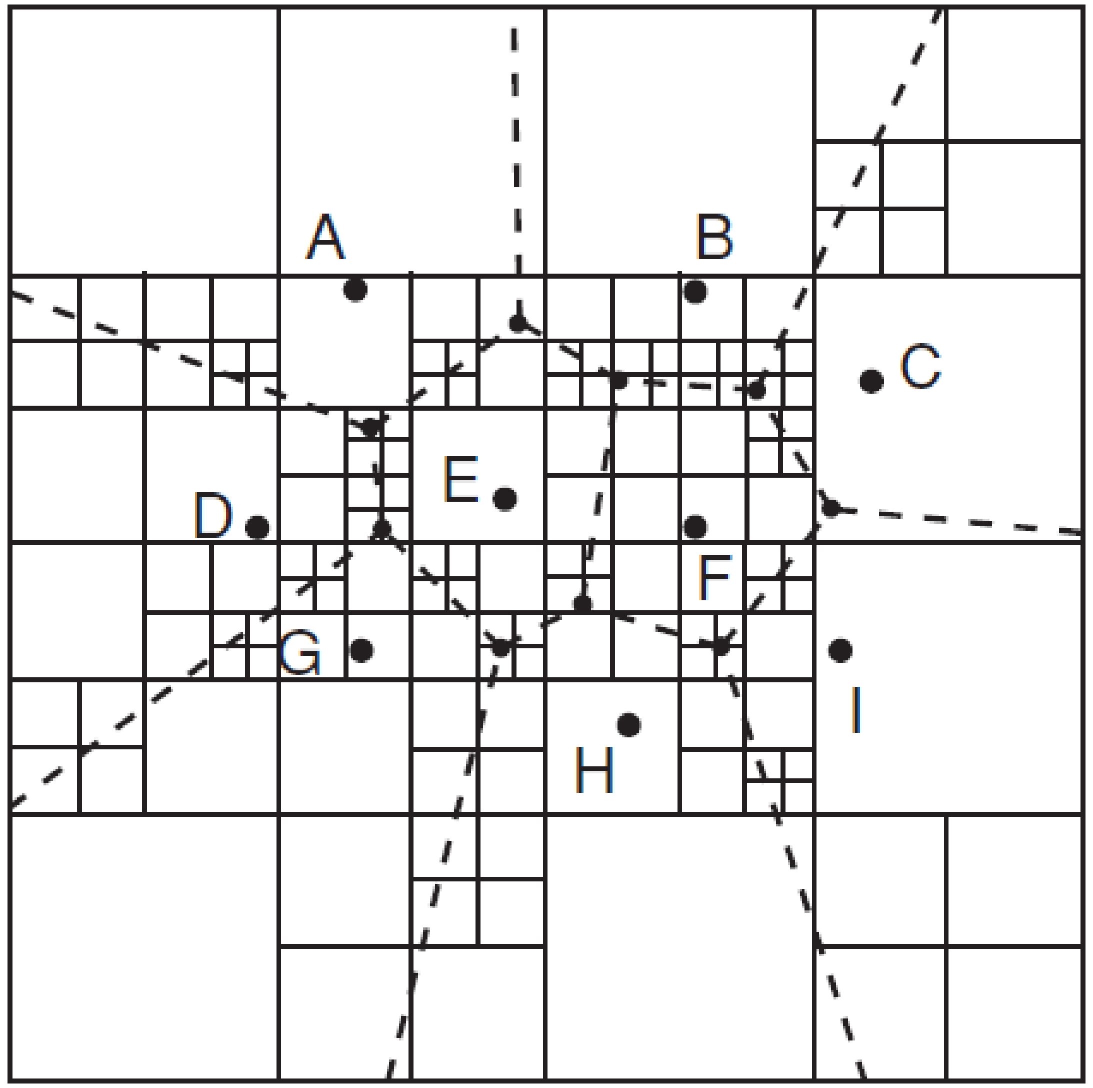
AVD

(Approximate Voronoi Diagram)

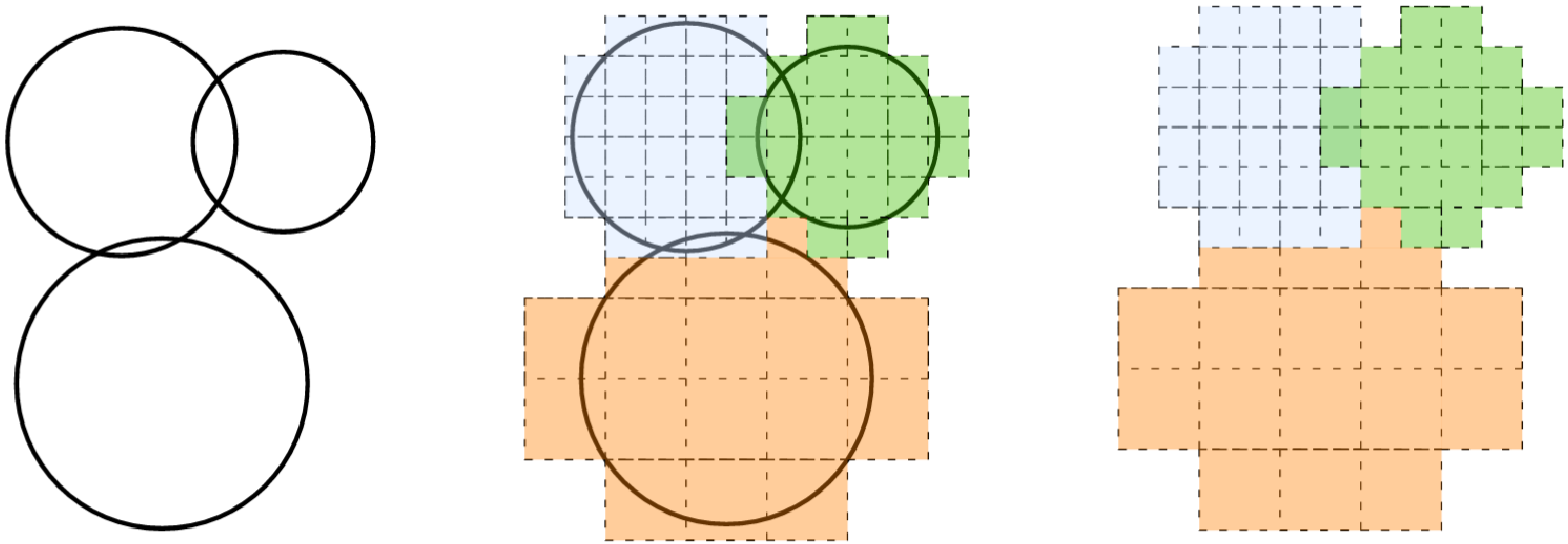


Sariel Har-Peled (2001) “A practical approach for computing the diameter of a point set”. Proceedings of the seventeenth annual symposium on Computational geometry (pp. 177-186).

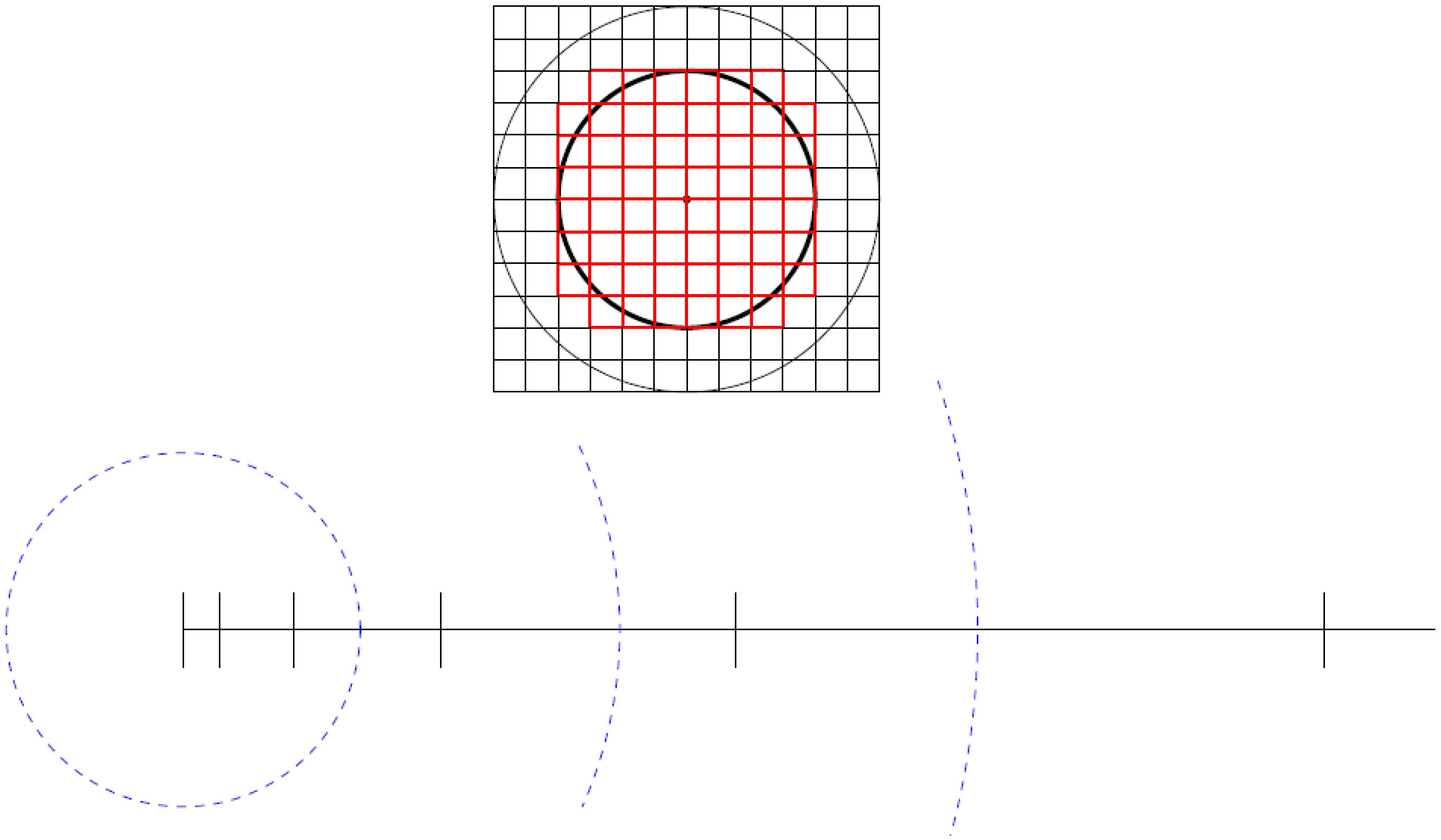
(t,ϵ)-Approximate *Voronoi Diagram*



(t,ϵ)-Approximate *Voronoi Diagram*

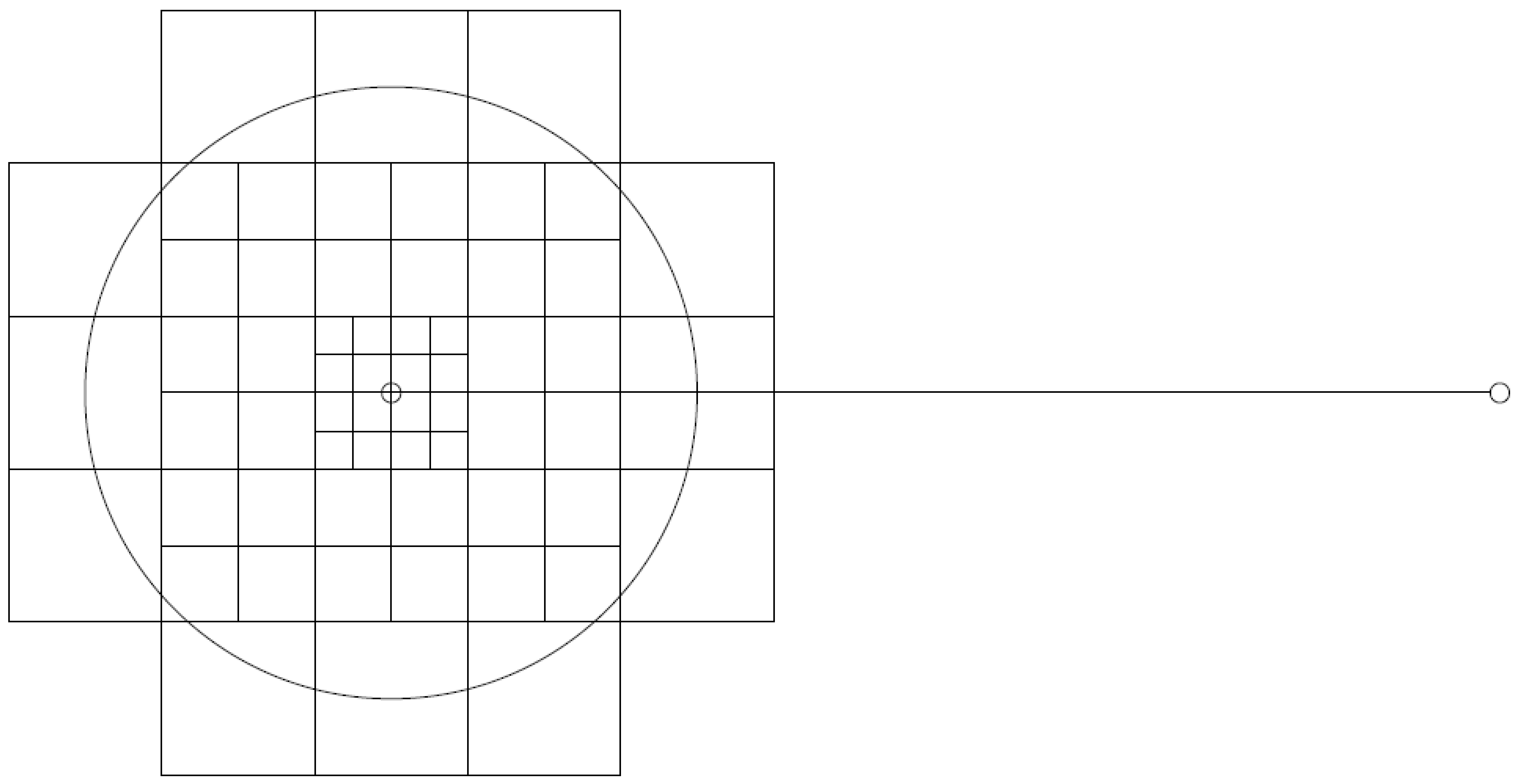


(t,ϵ)-Approximate *Voronoi Diagram*

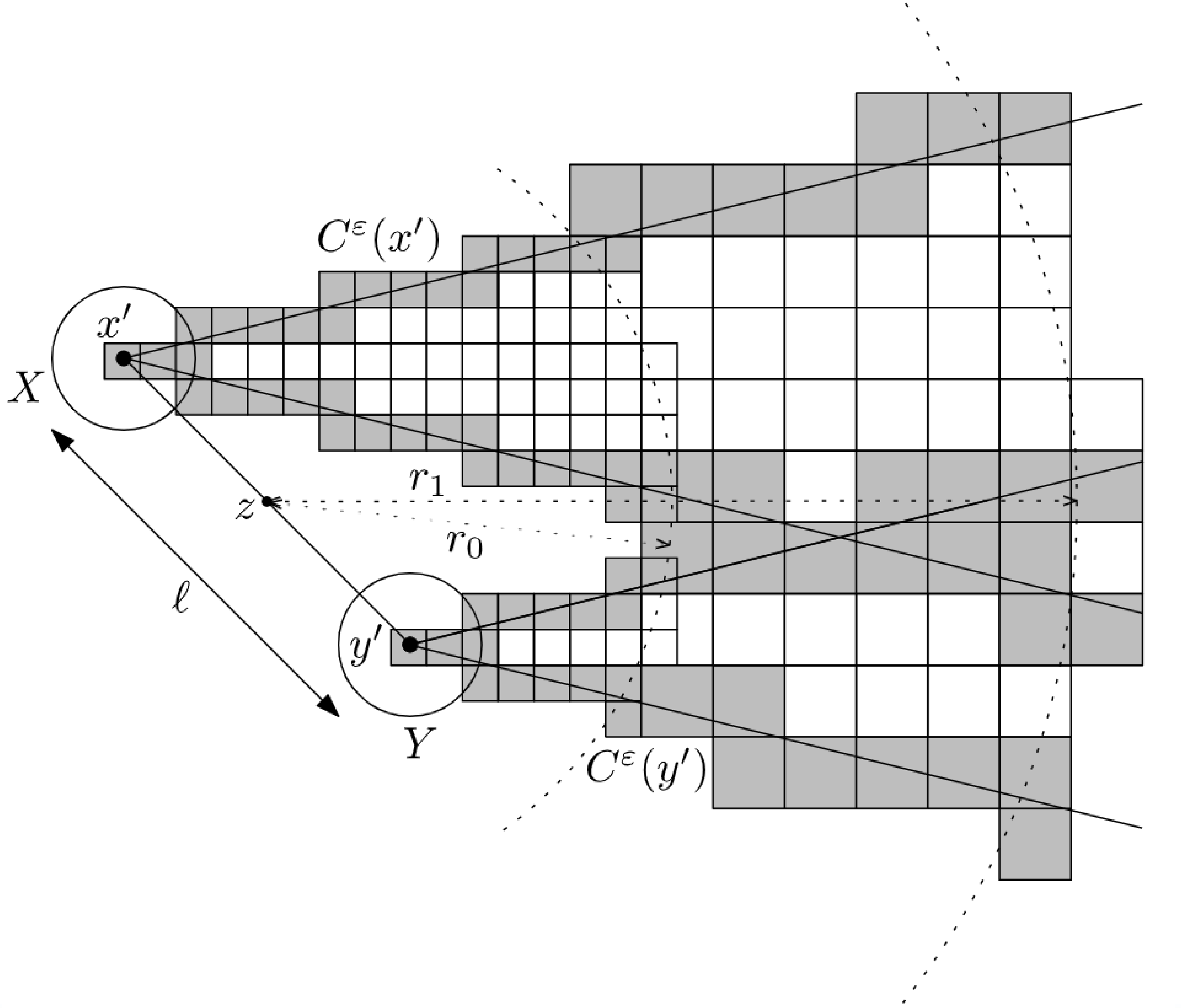


Tamaño de las cajas: $(1 + \epsilon)$, $(1 + \epsilon)^2$, $(1 + \epsilon)^3$, …

(t,ϵ)-Approximate *Voronoi Diagram*

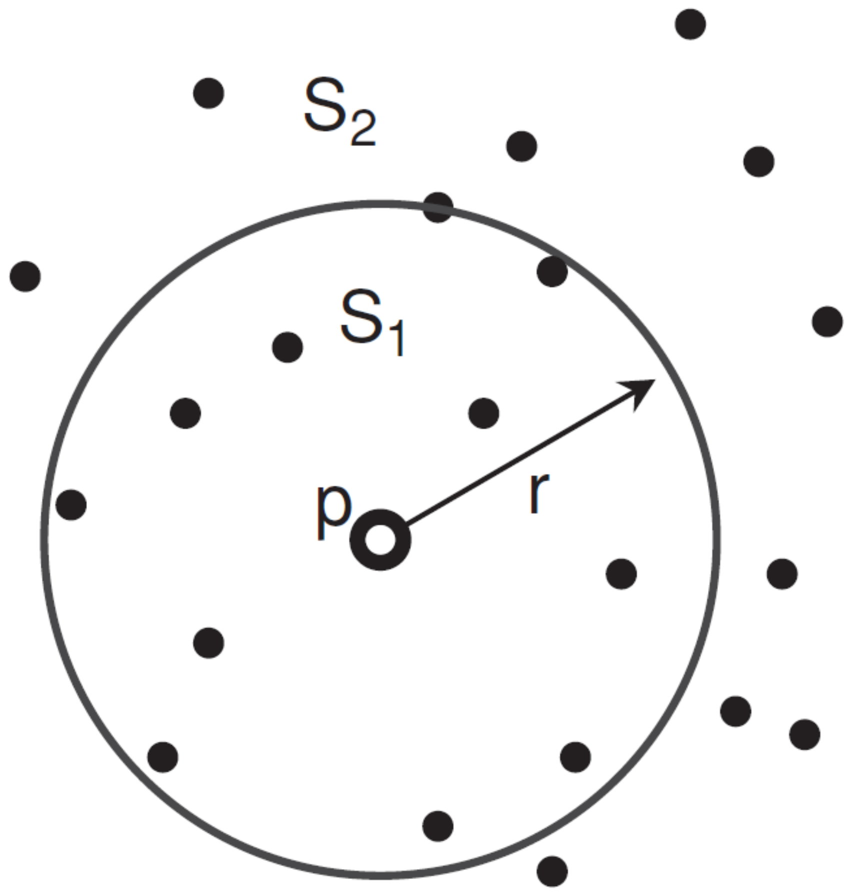


(t,ϵ)-Approximate *Voronoi Diagram*

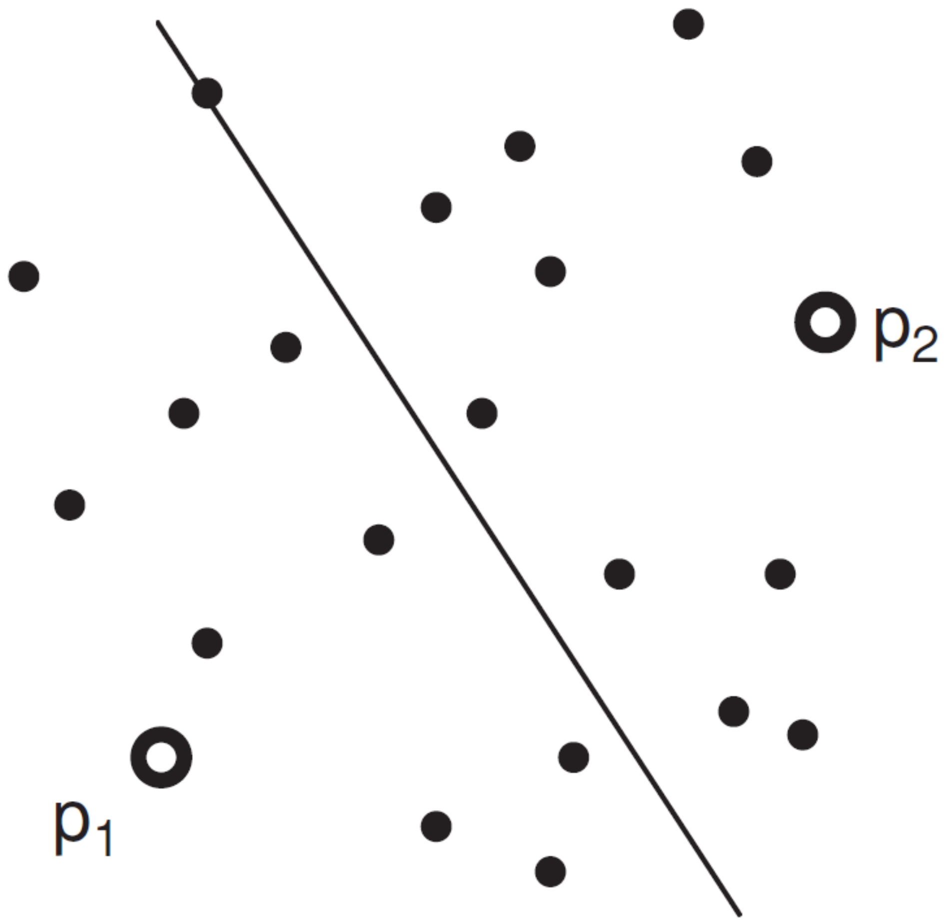


2. Distance-Based Indexing Methods

Distance-Based *Indexing Methods*



Ball partition

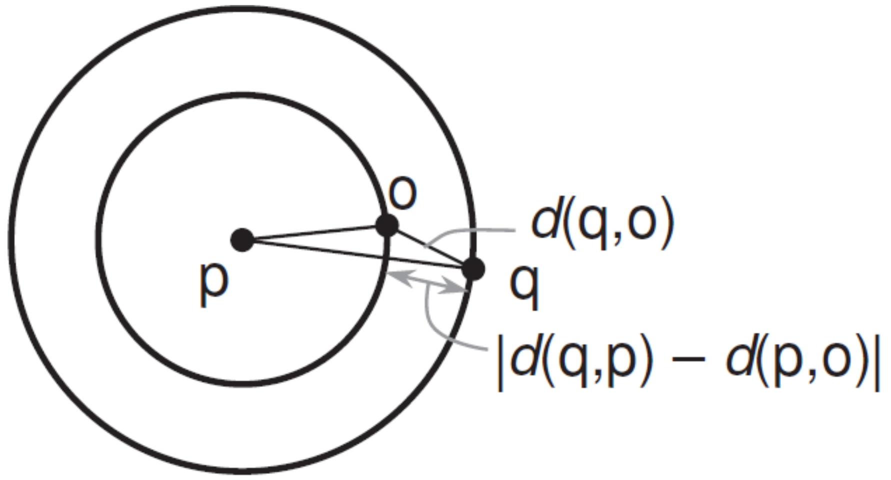


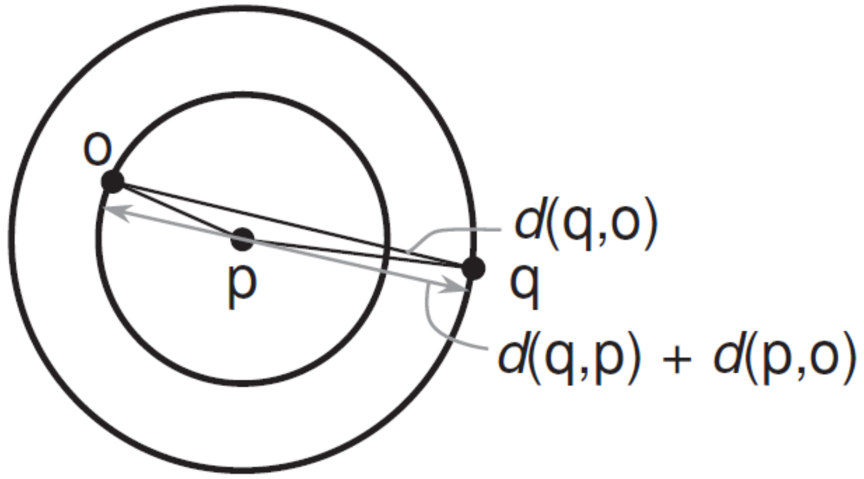
Generalized hyperplane partition

Distance-Based *Indexing Methods*

Lema 1

Sean los objetos p, q, o





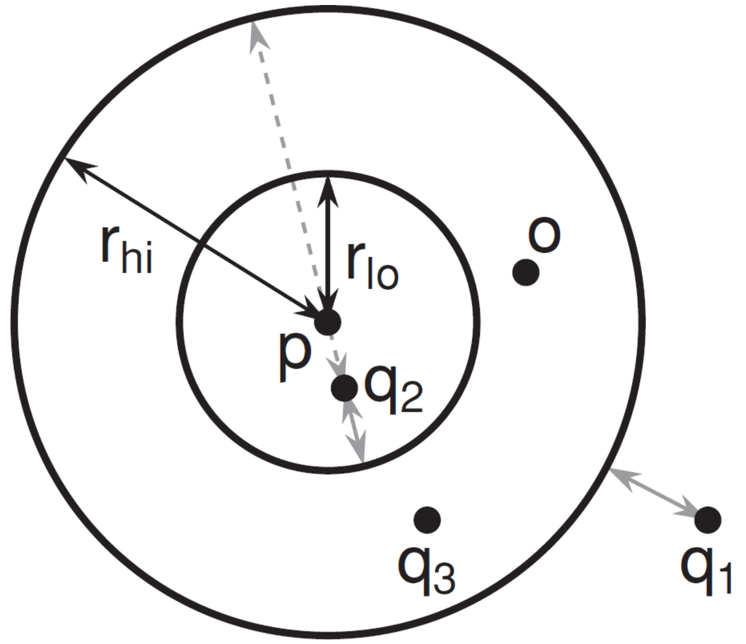
$$|d(q,p) - d(p,o)| \leq d(q,o) \leq d(q,p) + d(p,o)$$

Cota inferior **Cota superior**

Distance-Based *Indexing Methods*

Lema 2

Sean los objetos o y p tal que $r_{\mathrm{lo}} \leq d(o,p) \leq r_{\mathrm{hi}}$



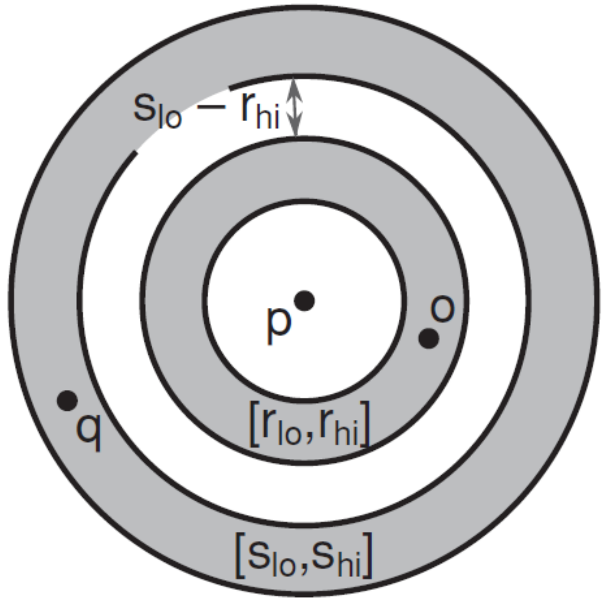
La distancia de o hacia algún otro objeto q está limitado por:

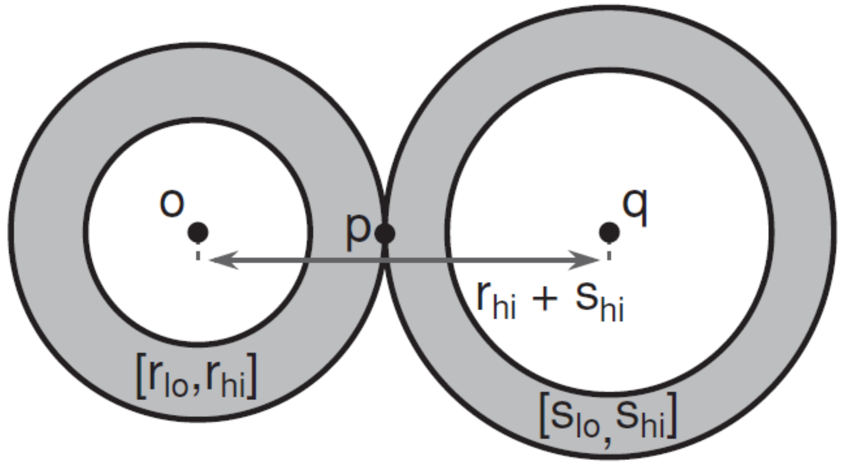
$$\max\{d(q,p) - r_{\mathrm{hi}}, r_{\mathrm{lo}} - d(q,p), 0\} \leq d(q,o) \leq d(q,p) + r_{\mathrm{hi}}$$

Distance-Based *Indexing Methods*

Lema 3

Sean los objetos o, p y q, donde $d(o,p) \in [r_{\mathrm{lo}}, r_{\mathrm{hi}}]$ y $d(q,p) \in [s_{\mathrm{lo}}, s_{\mathrm{hi}}]$





La distancia de q hacia o está limitado por:

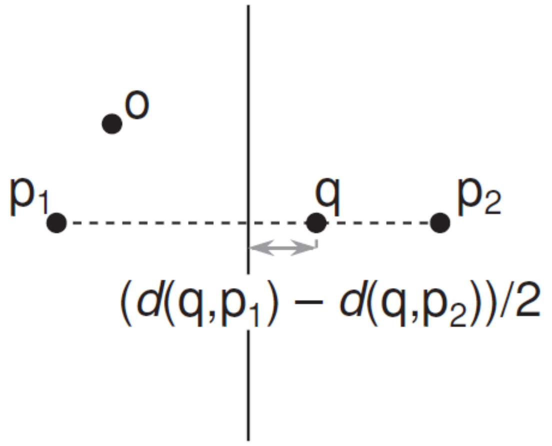
$$\max\{s_{\mathrm{lo}} - r_{\mathrm{hi}}, r_{\mathrm{lo}} - s_{\mathrm{hi}}, 0\} \leq d(q,o) \leq r_{\mathrm{hi}} + s_{\mathrm{hi}}$$

Distance-Based *Indexing Methods*

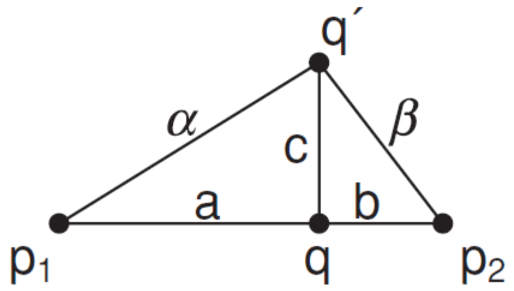
Lema 4

Sean o un objeto más cercano a p_1 que a p_2 o equidistante de ambos (i.e., $d(p_1, o) \leq d(p_2, o)$). Dados $d(q, p_1)$ y $d(q, p_2)$, podemos establecer un límite inferior para $d(q, o)$:

$$\max\left\{\frac{d(q,p_1) - d(q,p_2)}{2}, 0\right\} \leq d(q,o)$$



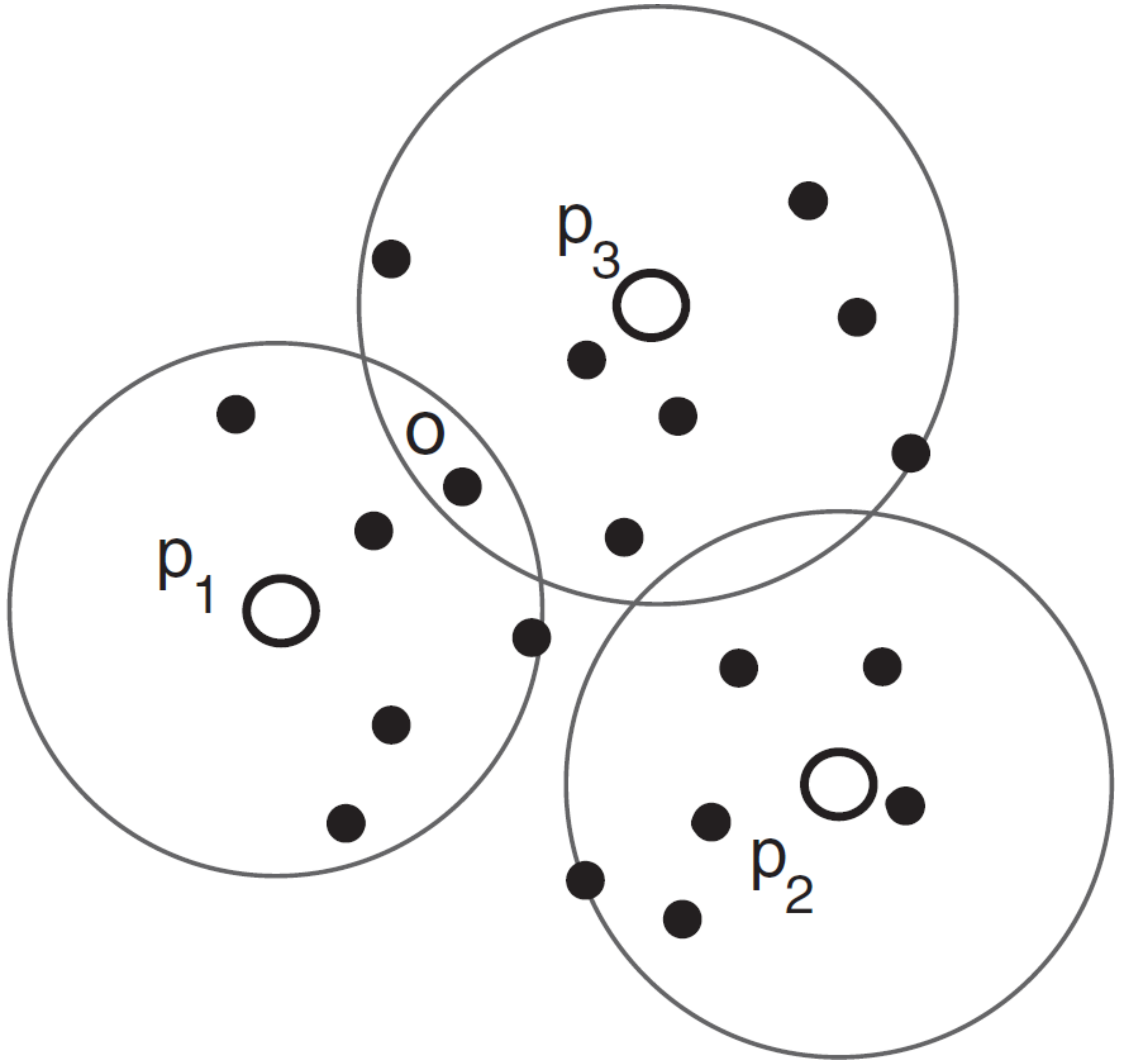
Límite inferior de $d(q, o)$ ilustrado en un espacio euclidiano bidimensional cuando q está en la línea entre p_1 y p_2, más cerca de p_2, mientras que o está más cerca de p_1.



El límite inferior disminuye cuando q se desplaza fuera de la línea (por ejemplo, a q').

3. M-Tree

M-*Tree*



Nodo:

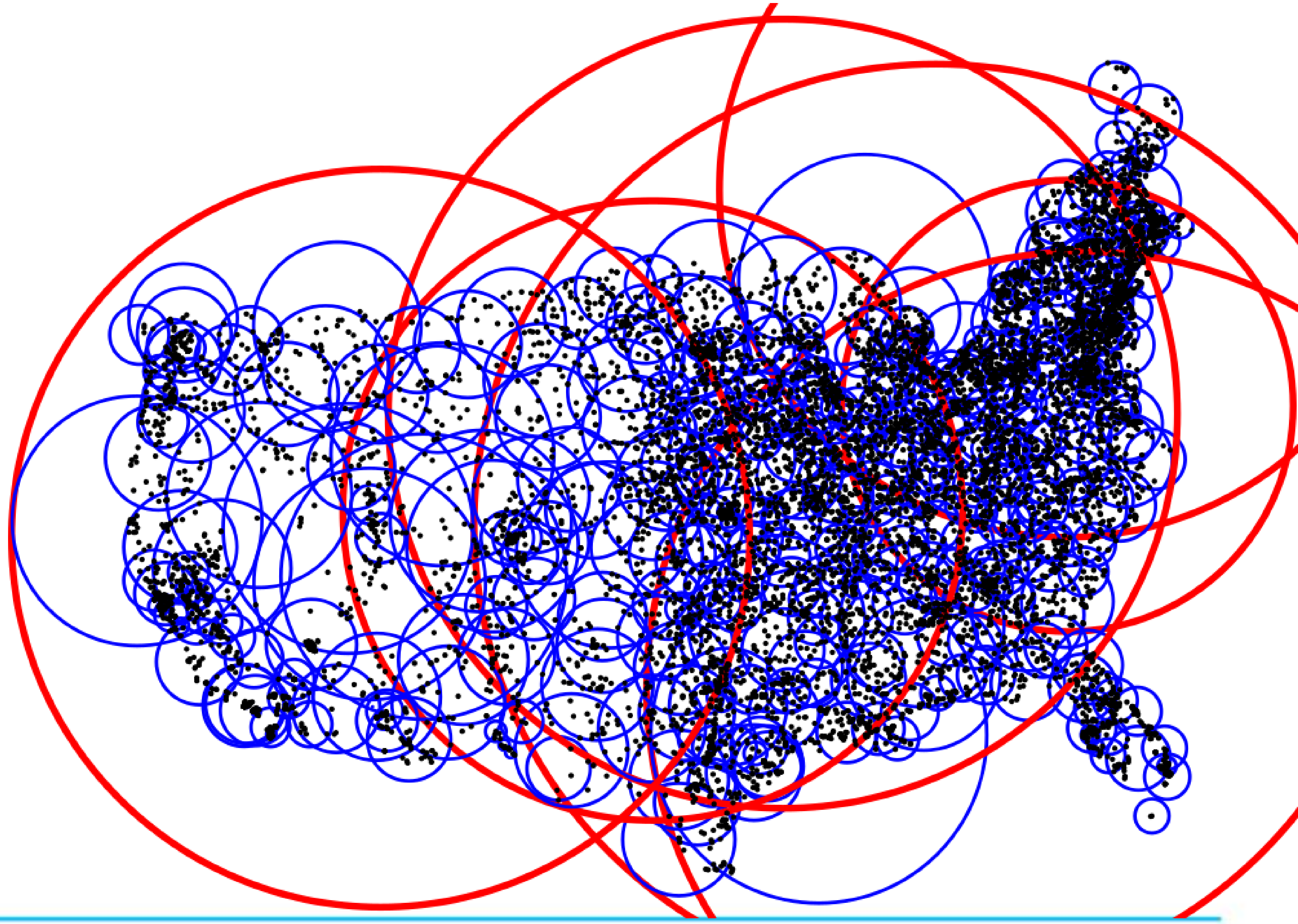
$$[p, r, d(p, p'), T]$$

- p : Pivote (centro de la hiper-esfera)
- r : Radio
- $d(p, p')$: Distancia del pivote al pivote del padre
- T : Lista de nodos hijos

M-*Tree*

Objetivo:

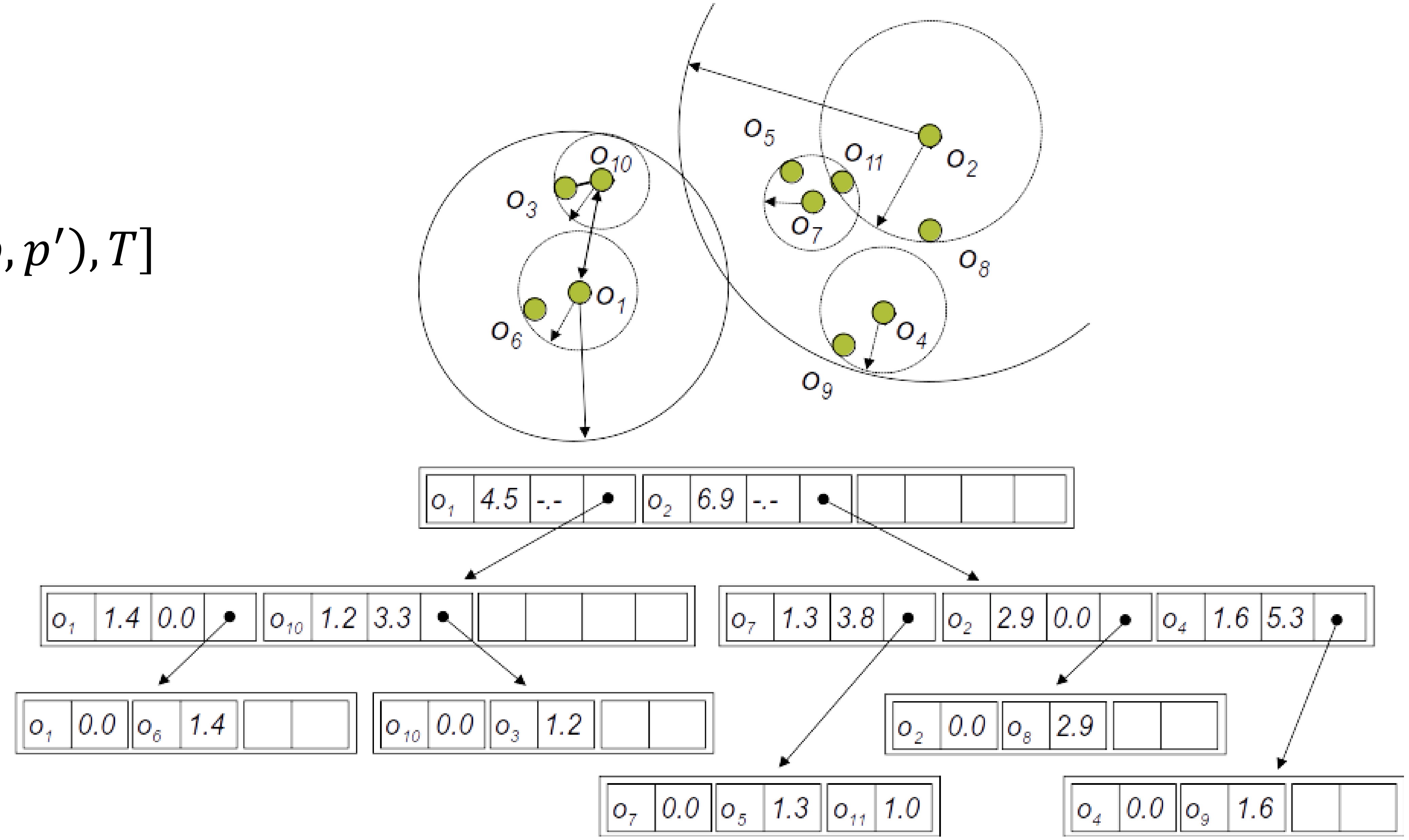
- Minimizar volumen
- Minimizar *overlap*



M-*Tree*

Nodo:

$[p, r, d(p, p'), T]$



M-*Tree*

Inserción

Insertar un nuevo objeto O_n:

- Desciende recursivamente por el árbol para localizar la hoja más adecuada para O_n
- En cada paso, acceda al subárbol con pivote p para el cual:
  - No se necesita ampliar el radio r_c, es decir, $d(O_n, p) \leq r_c$
    En caso de empate, elegir el que tenga el p más cercano a O_n
  - Minimizar la ampliación de r_c
- Al llegar al nodo hoja N entonces
  - **Si N no está lleno:** guardar O_n en N
  - **Si no:** split(N,O_n).

M-*Tree*

Split

- Sea S el conjunto que contiene todas las entradas de N y O_n
- Seleccionar los pivotes p_1 y p_2 de S
- Particionar S en S_1 y S_2 según p_1 y p_2
- Almacenar S_1 en N y S_2 en un nuevo nodo asignado N'

  - **Si N es raíz:**
    - Asignar una nueva raíz y almacenar allí las entradas de p_1, p_2
  - **Si no:** (deja que N_p y p_p sean el nodo y pivote padre de N)
    - Sustituir la entrada p_p por p_1
    - Si N_p está lleno, entonces **split**(N_p,p_2)
    - Sino, almacenar p_2 en el nodo N_p

M-*Tree*

Pivot selection policies

Aleatorio — Completamente aleatorio

Mínima suma de radios

$$\min(r_1 + r_2)$$

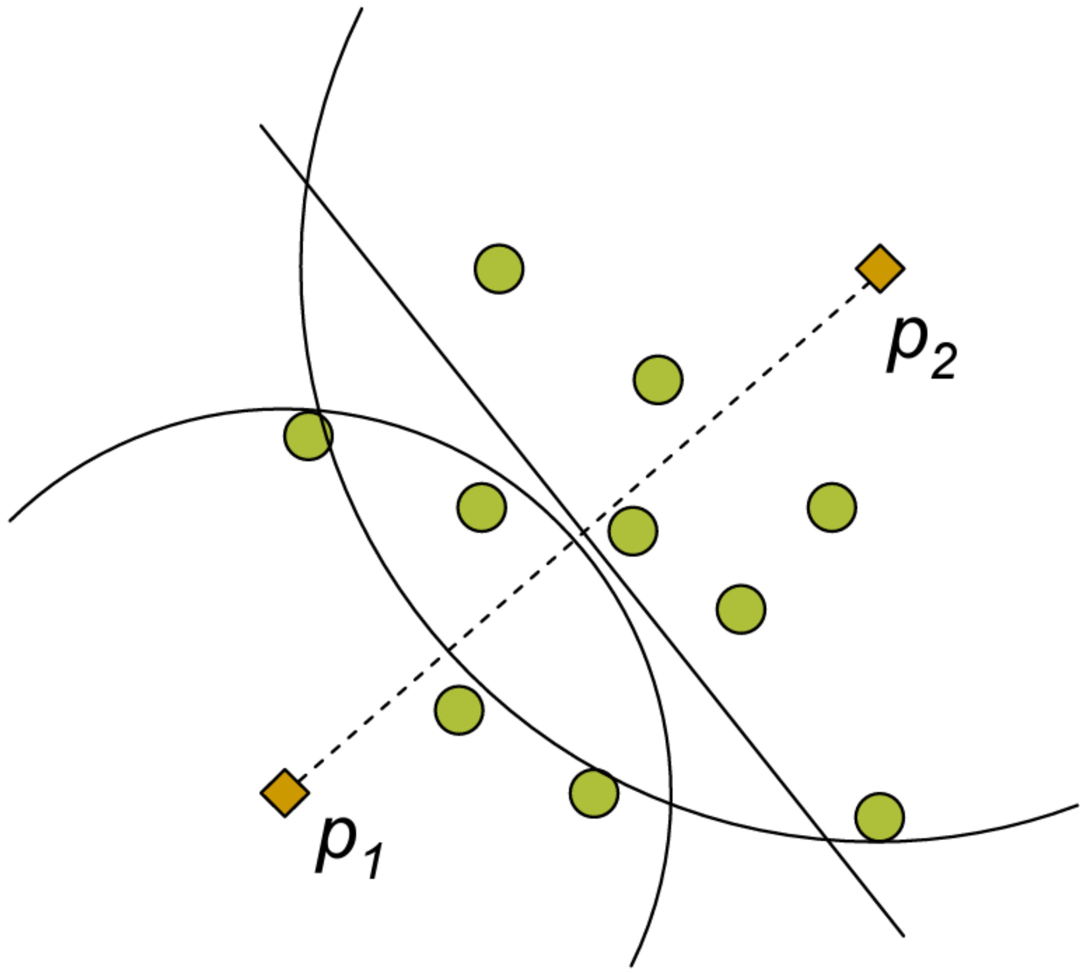
Muestreo — **10%** aleatorio

$$\min(r_1 + r_2)$$

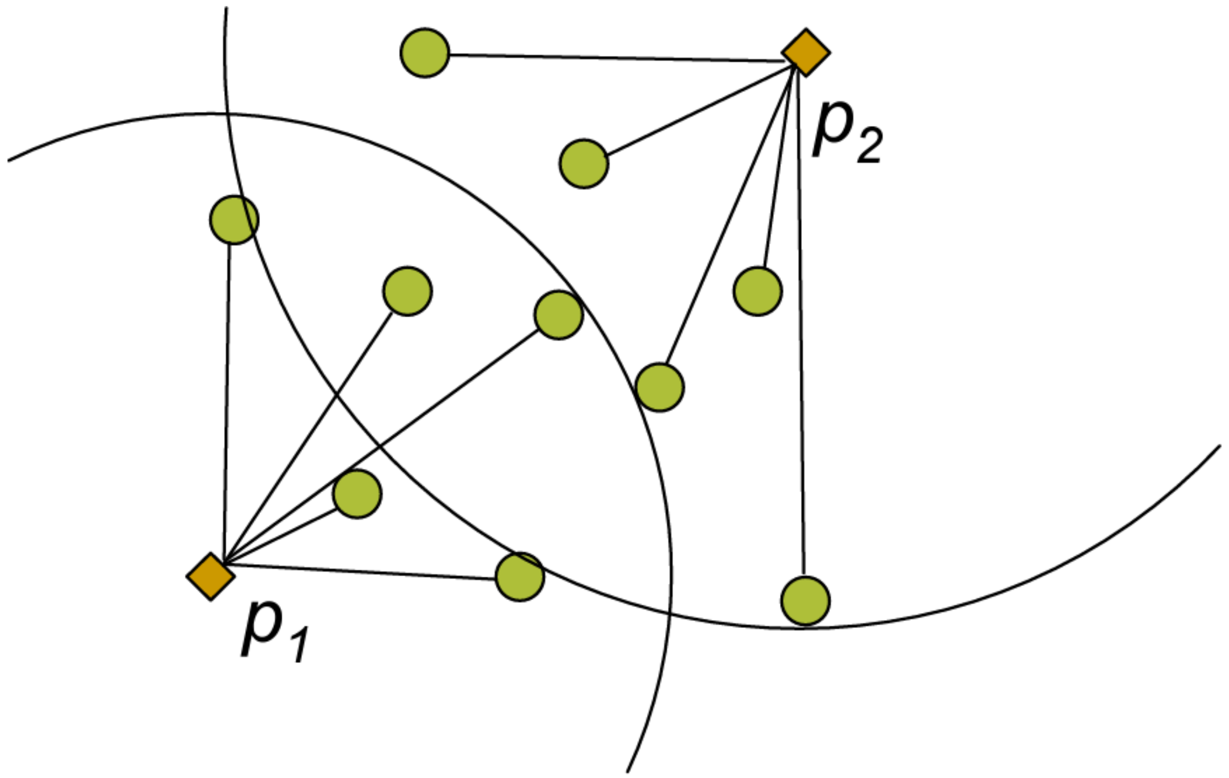
M-*Tree*

Split policies

Unbalanced Hiperplano generalizado



Balanced Radios de cobertura más grandes



Mejor es *unbalanced*

M-*Tree*

Range Query

Nodo interno

Se omite el nodo si $|d(q,p') - \underset{d(p,p')}{D}| - r > \epsilon$ Lema 1, 3

Sino, se omite el nodo si $d(q,p) - r > \epsilon$ Lema 2

Nodo hoja

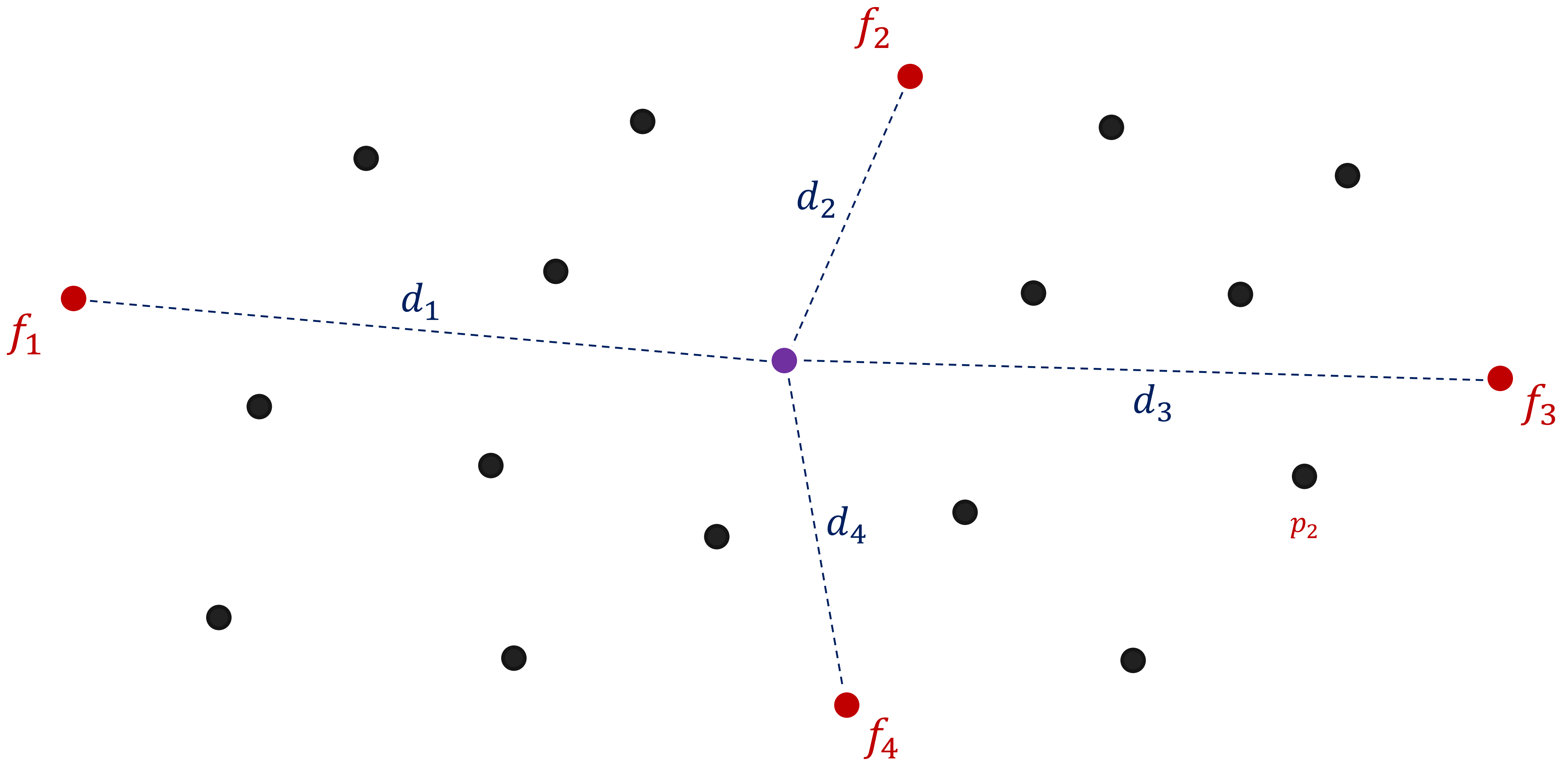
Se omite el objeto si $|d(q,p') - D| - r \leq \epsilon$

4. Omni-family

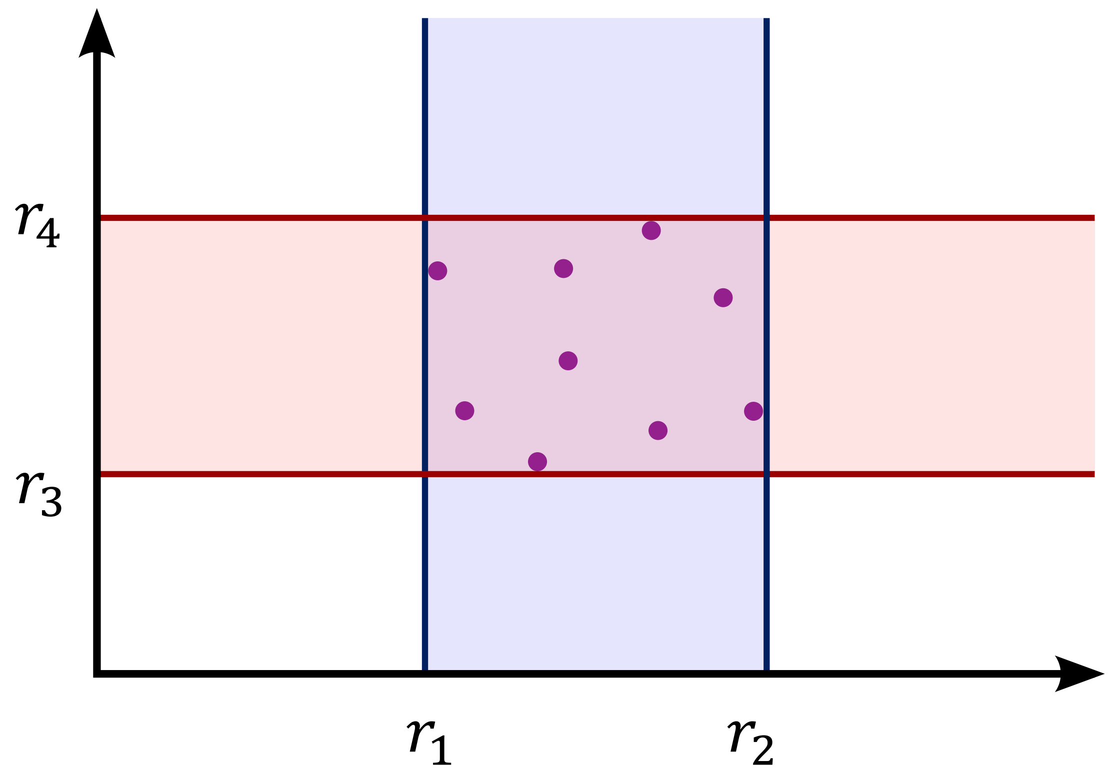
Omni-*family*

Omni-foci base: $\mathcal{F} = \{f_1, f_2, \ldots, f_h\}$

Omni-coordinate: (d_1, d_2, d_3, d_4)

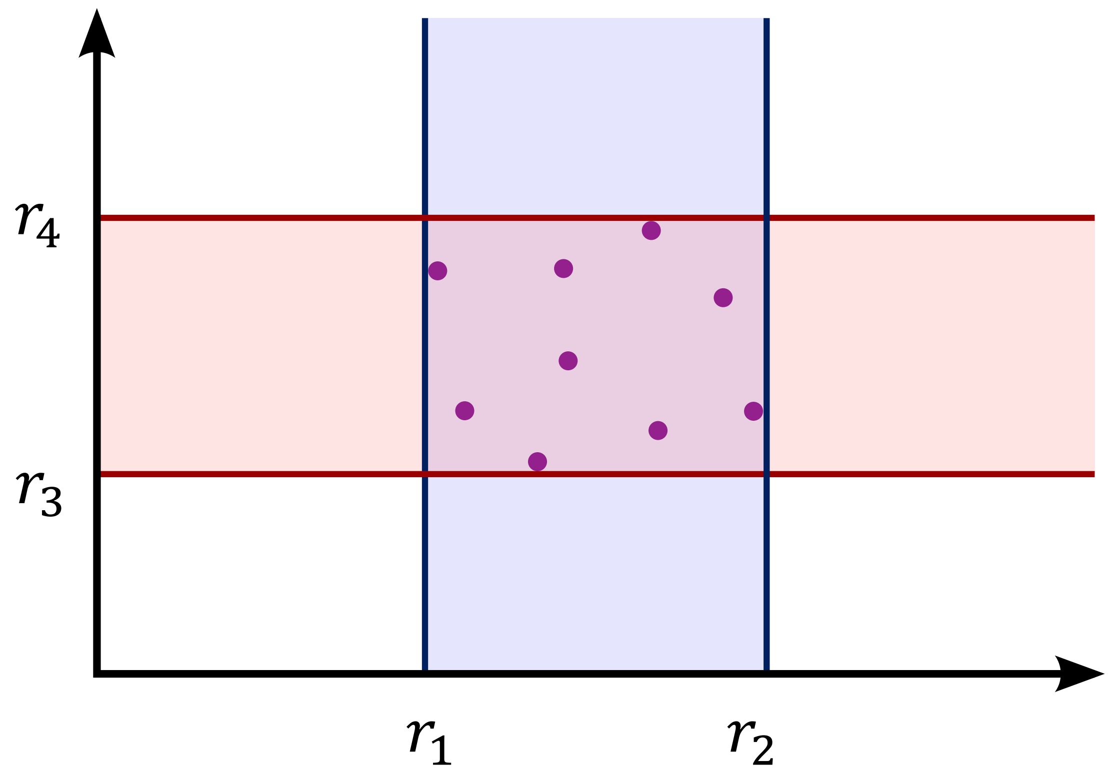


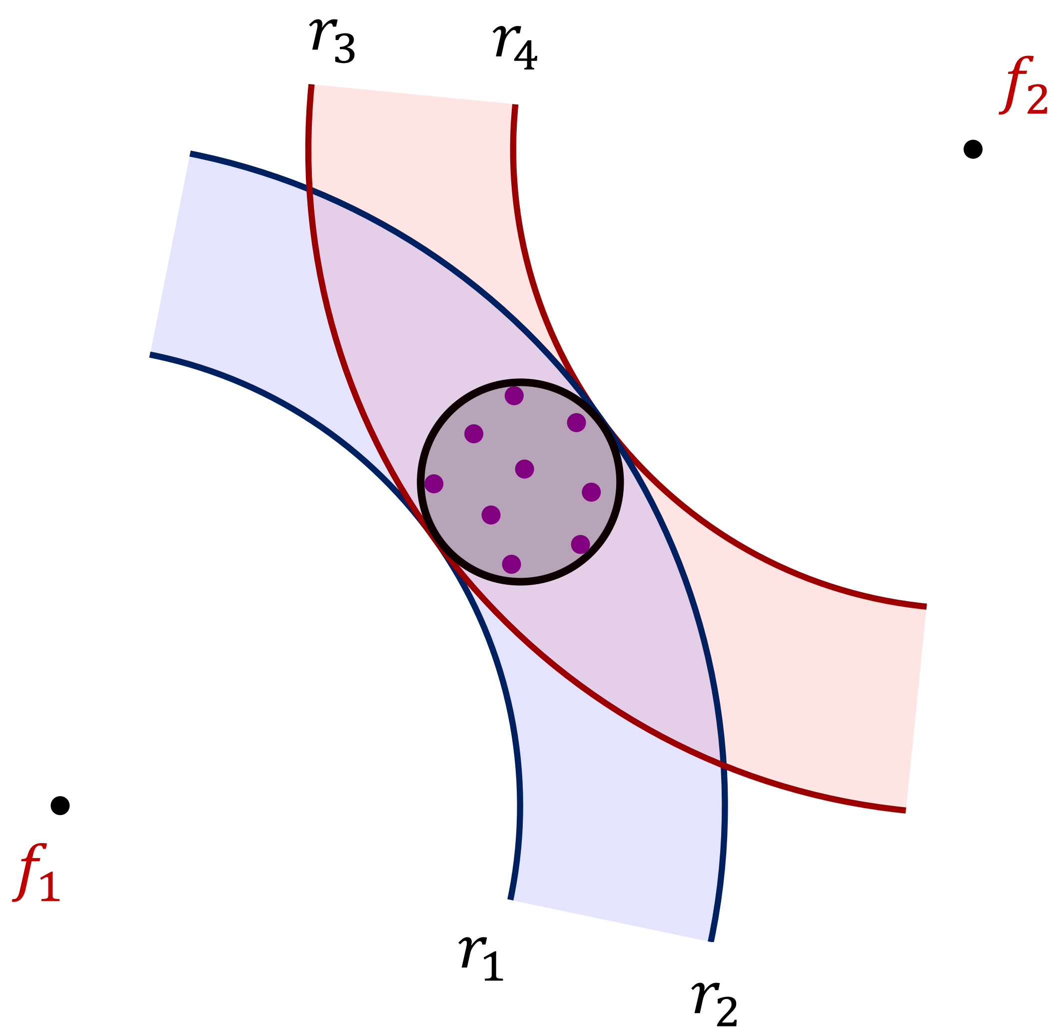
Minimum-bounding-*Omni-region (mbOr)*



Región: $r_1 < x < r_2$
$r_3 < y < r_4$

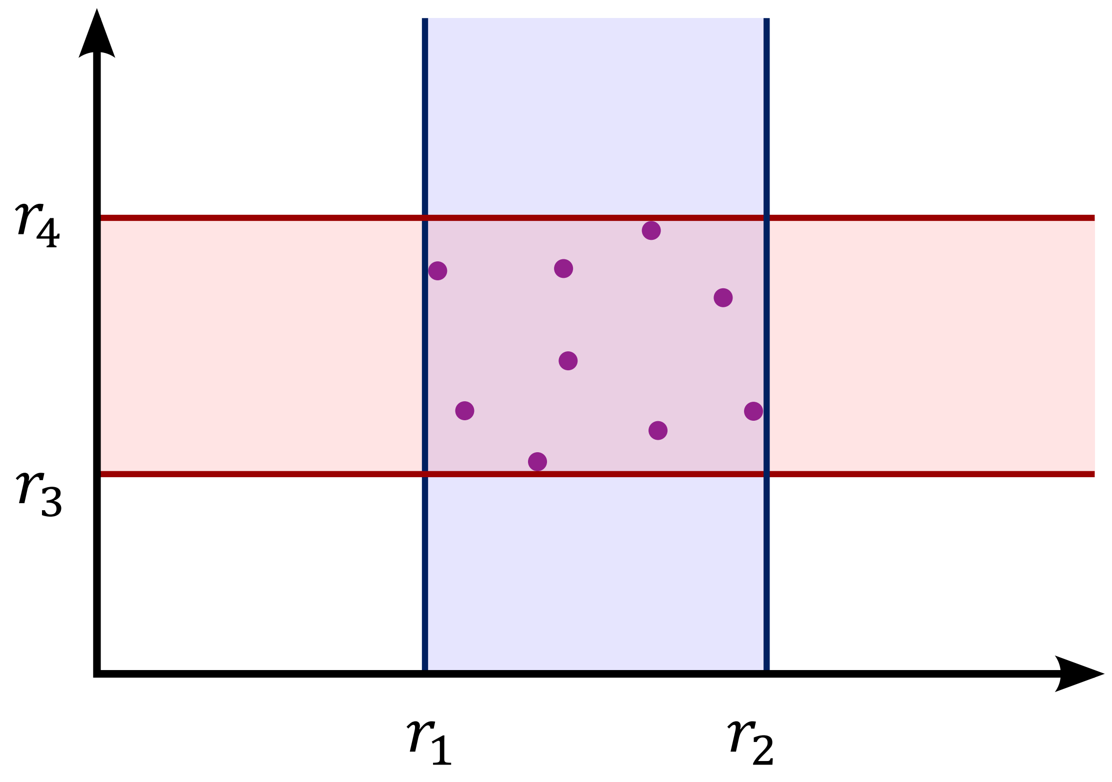
Minimum-bounding-*Omni-region (mbOr)*



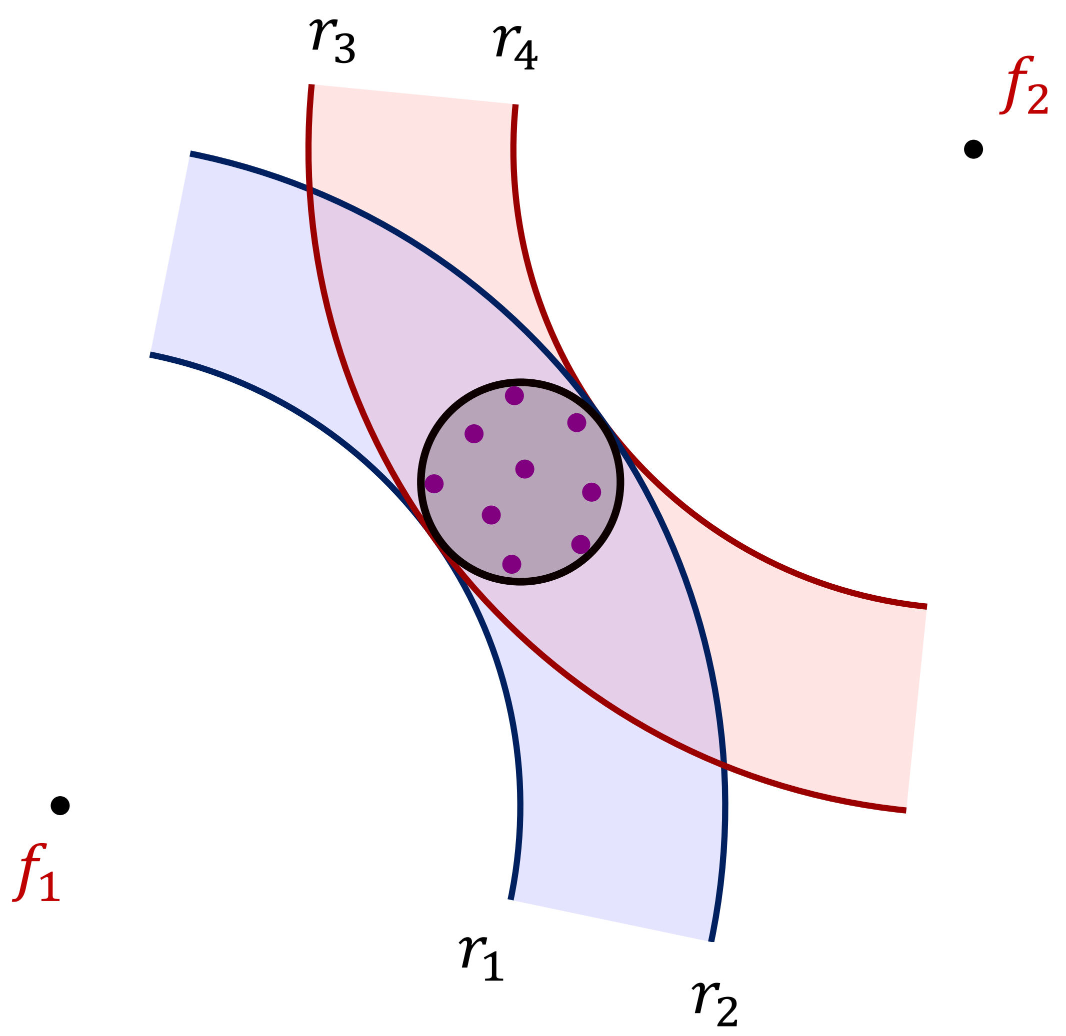


Región: $r_1 < x < r_2$
$r_3 < y < r_4$

Minimum-bounding-*Omni-region (mbOr)*

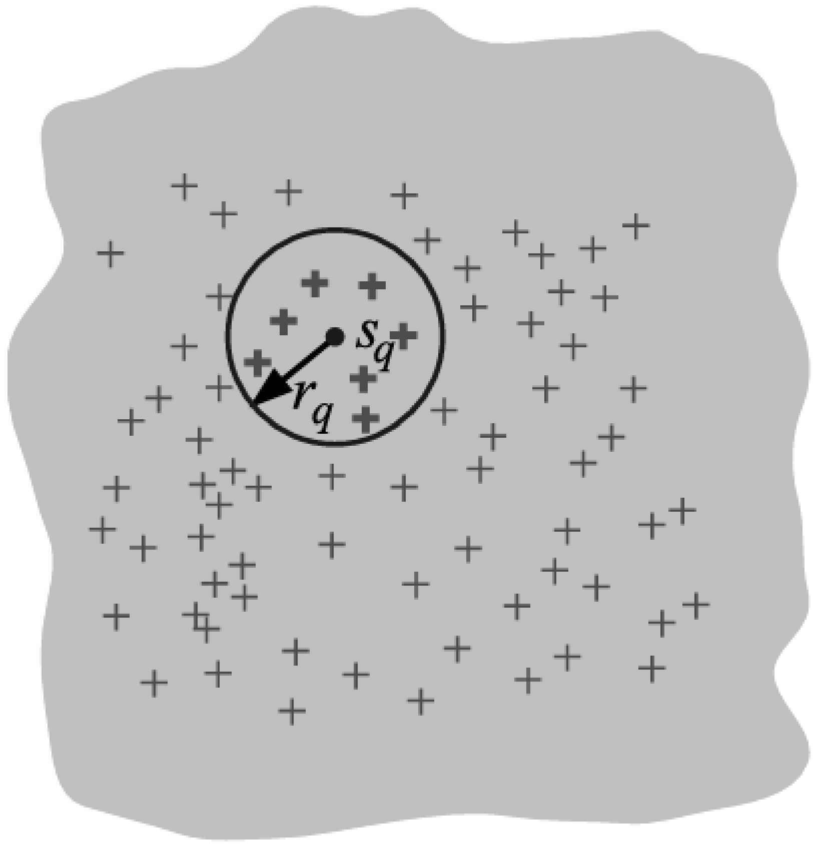


Región: $r_1 < x < r_2$
$r_3 < y < r_4$

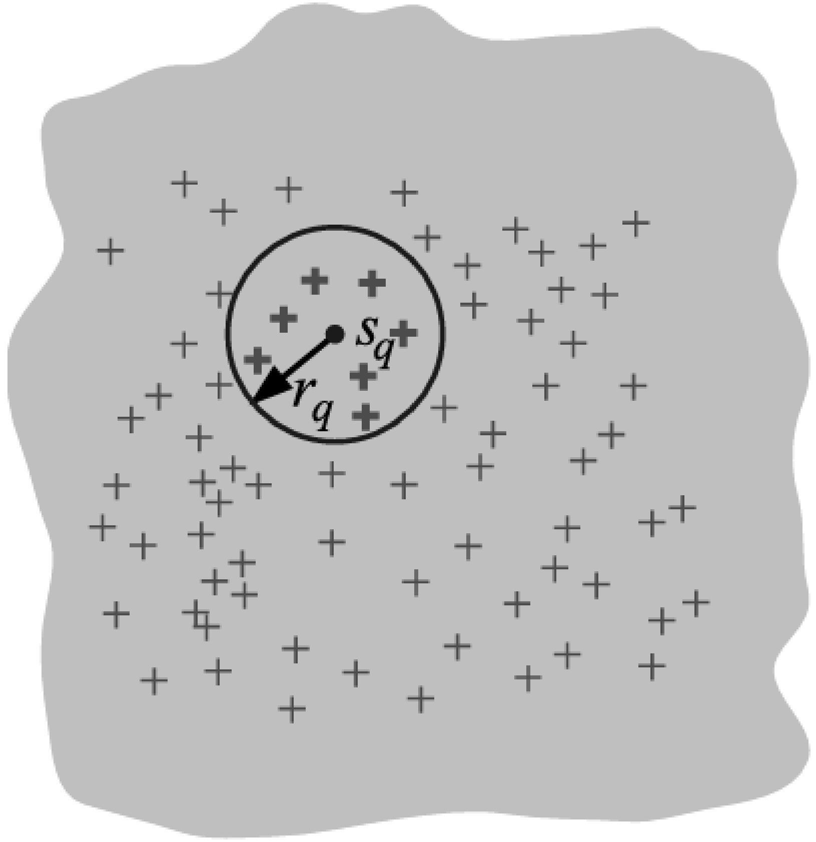


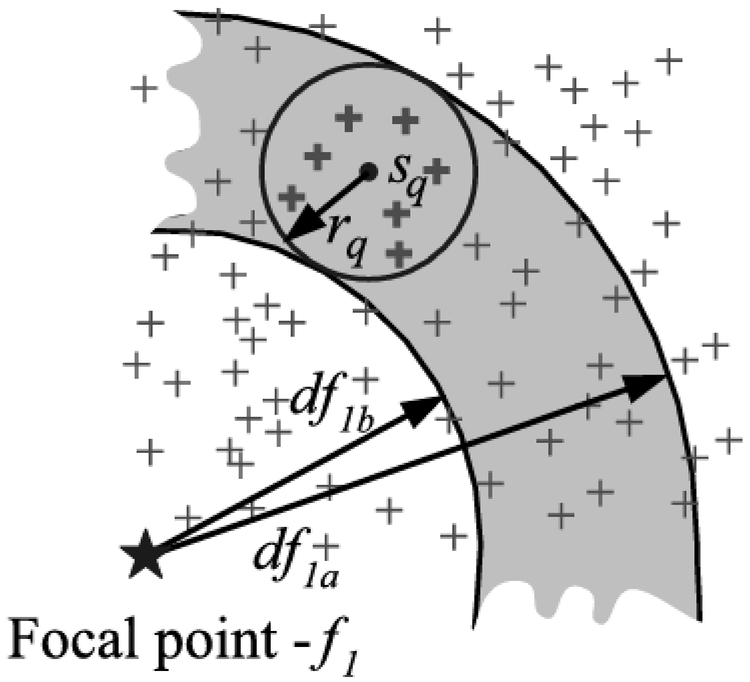
Región: $r_1 < \|x - f_1\| < r_2$
$r_3 < \|x - f_2\| < r_4$

Minimum-bounding-*Omni-region (mbOr)*

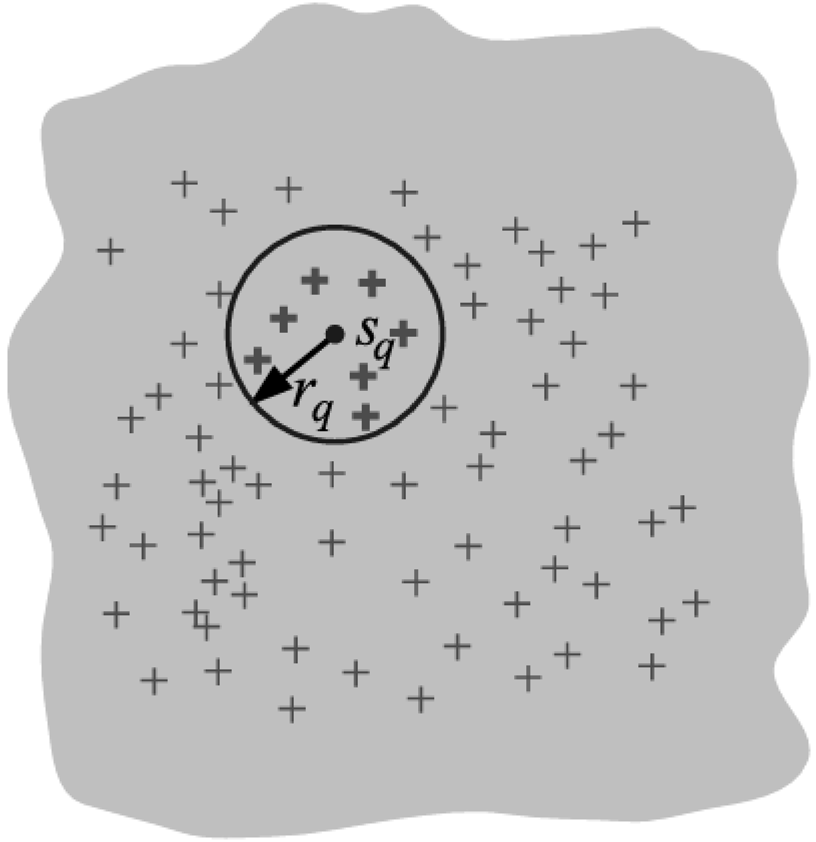


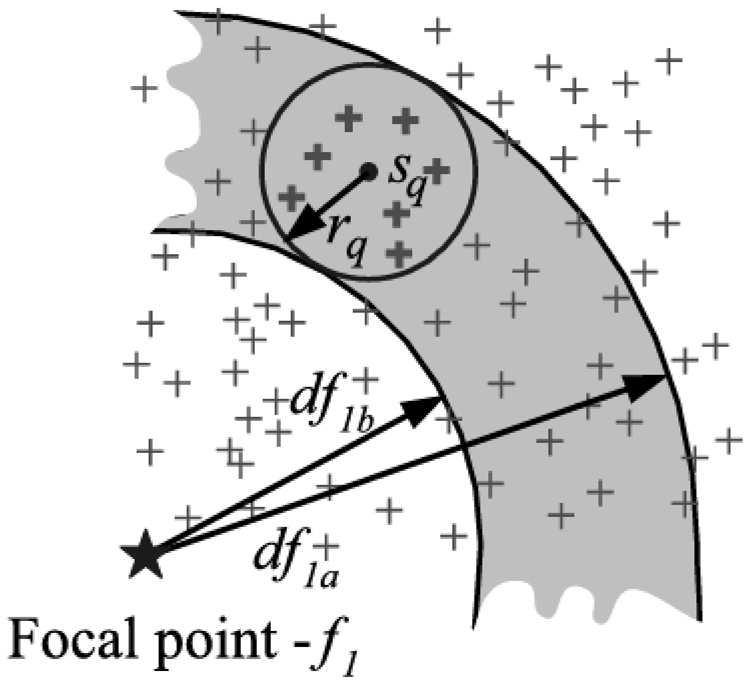
Minimum-bounding-*Omni-region (mbOr)*

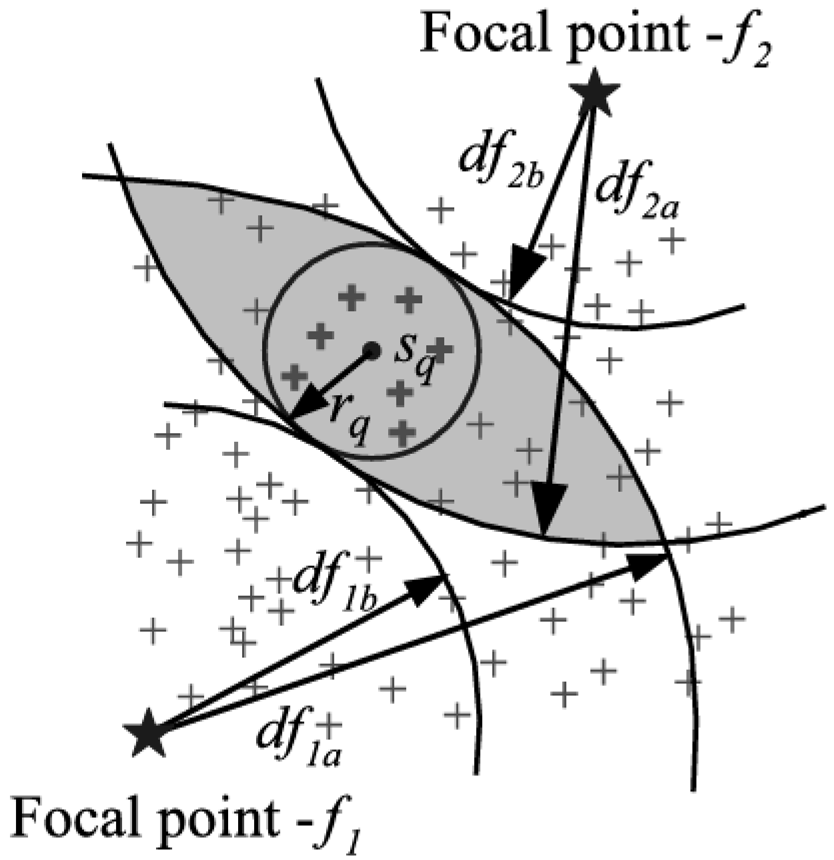




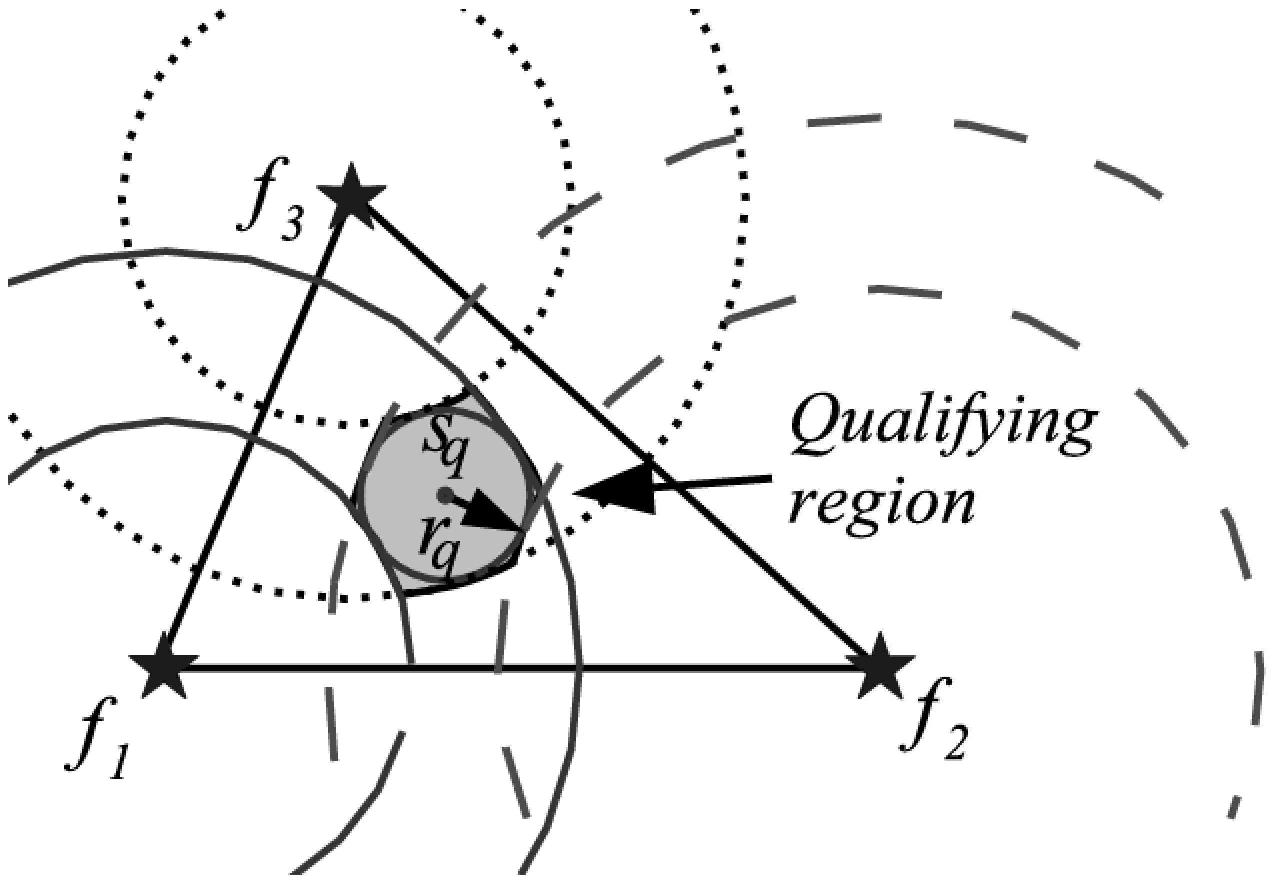
Minimum-bounding-*Omni-region (mbOr)*



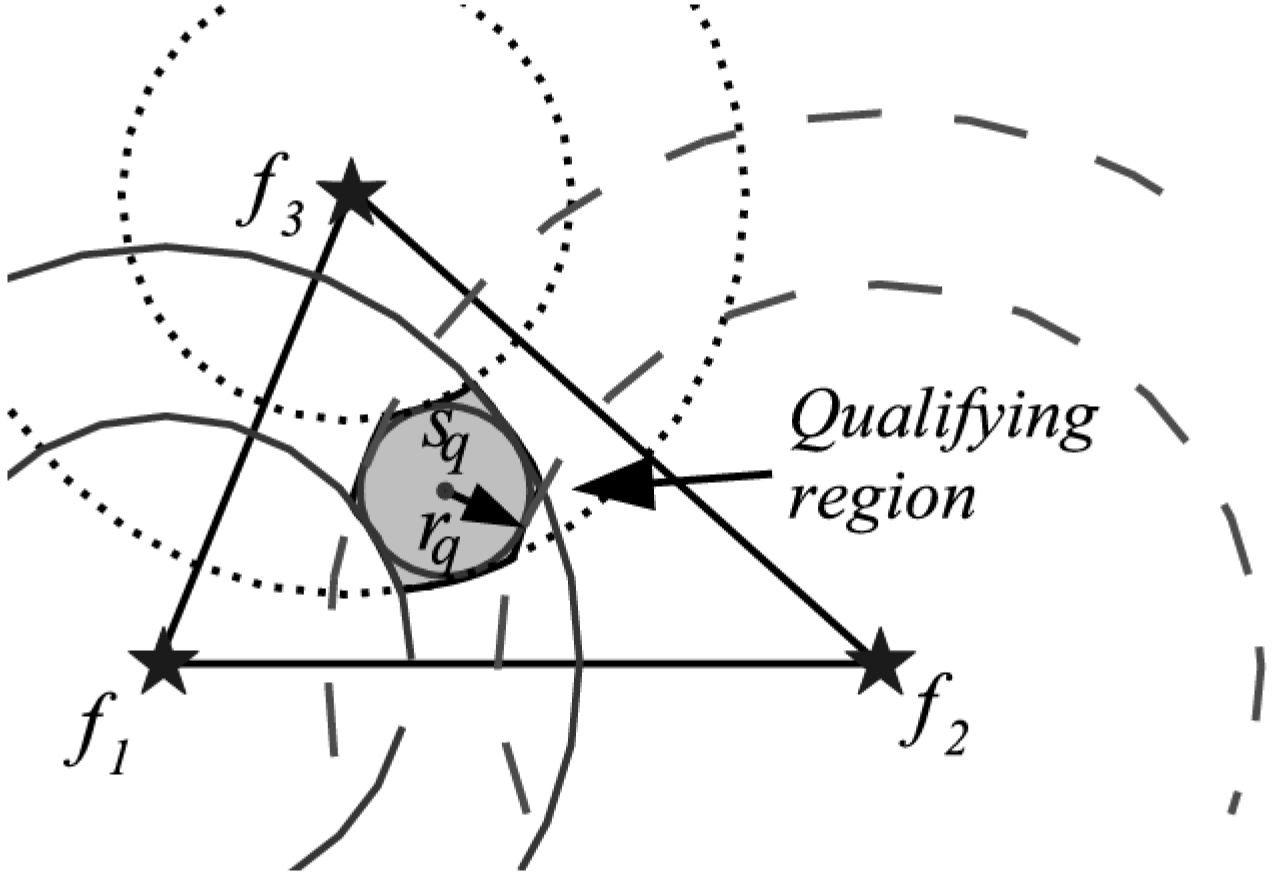


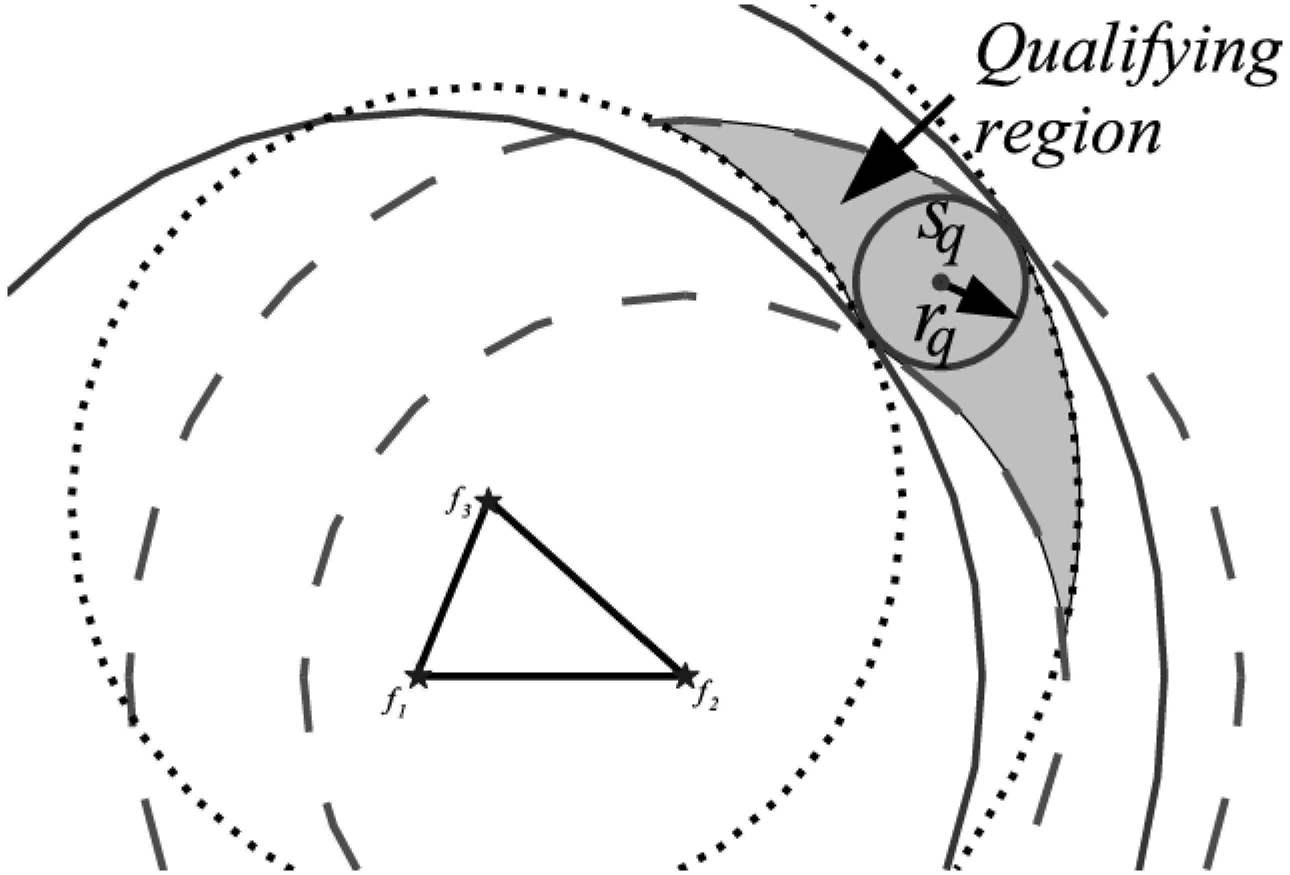


Minimum-bounding-*Omni-region (mbOr)*

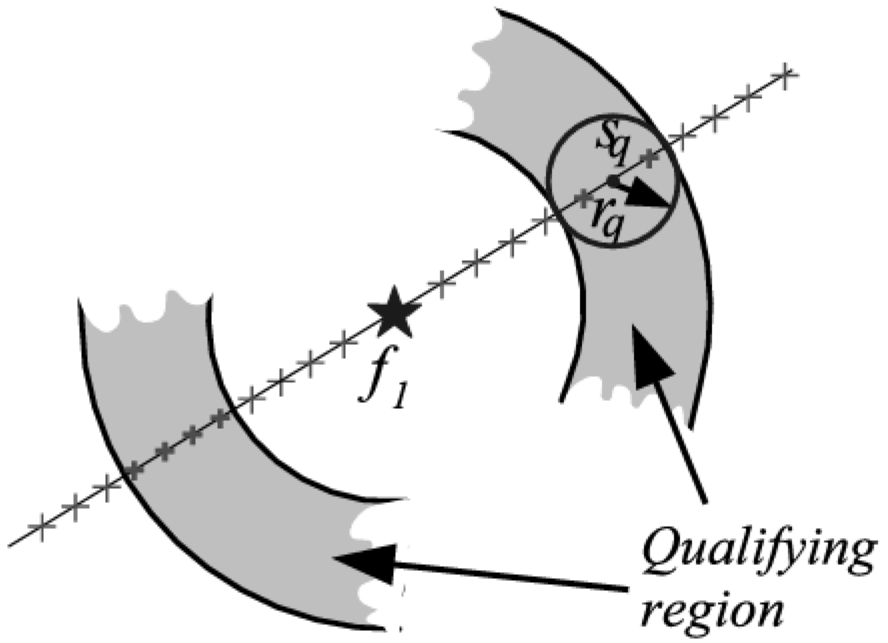


Minimum-bounding-*Omni-region (mbOr)*

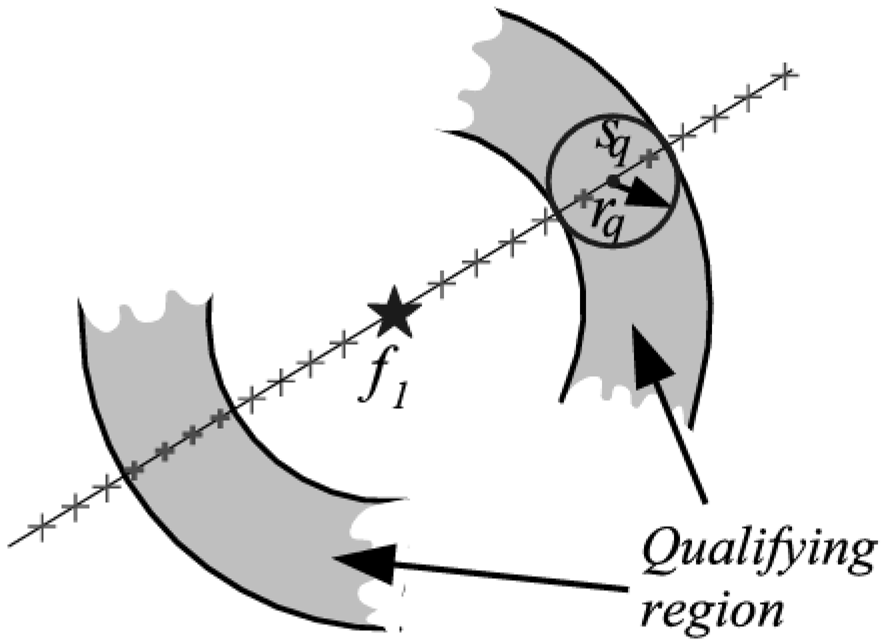


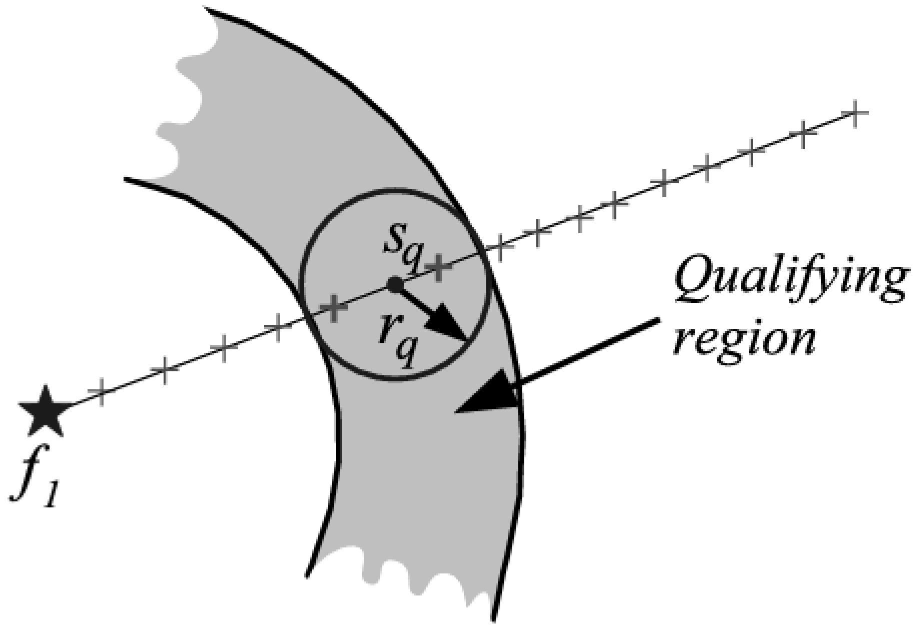


Omni-foci *base*

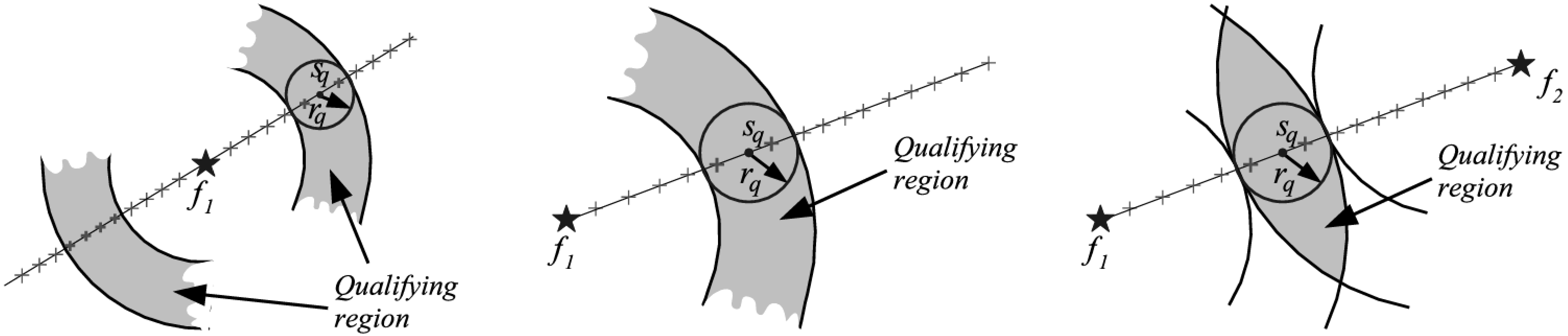


Omni-foci *base*





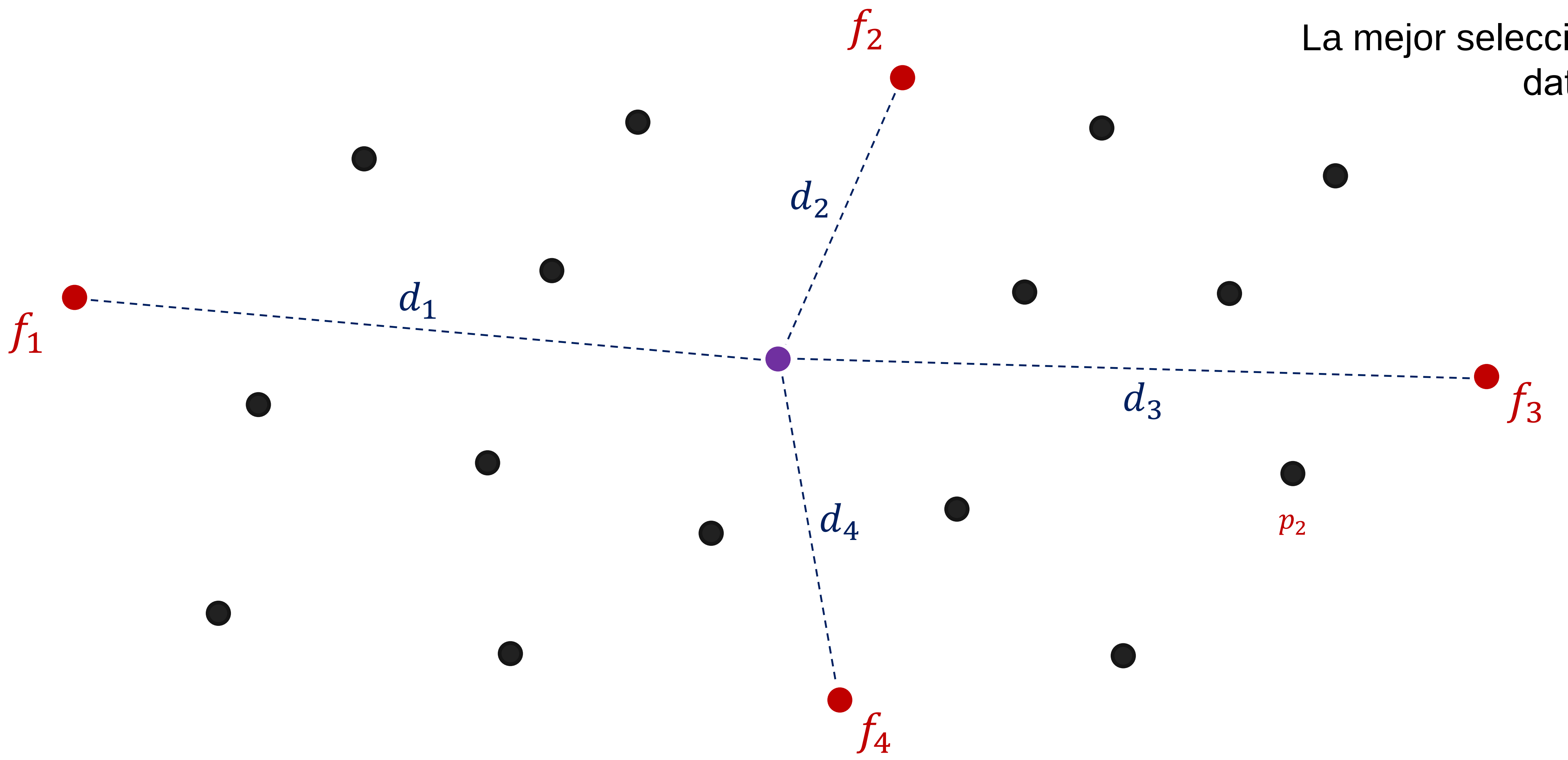
Omni-foci *base*



Omni-foci *base*

¿Como elegimos los foci?

Omni-foci *base*

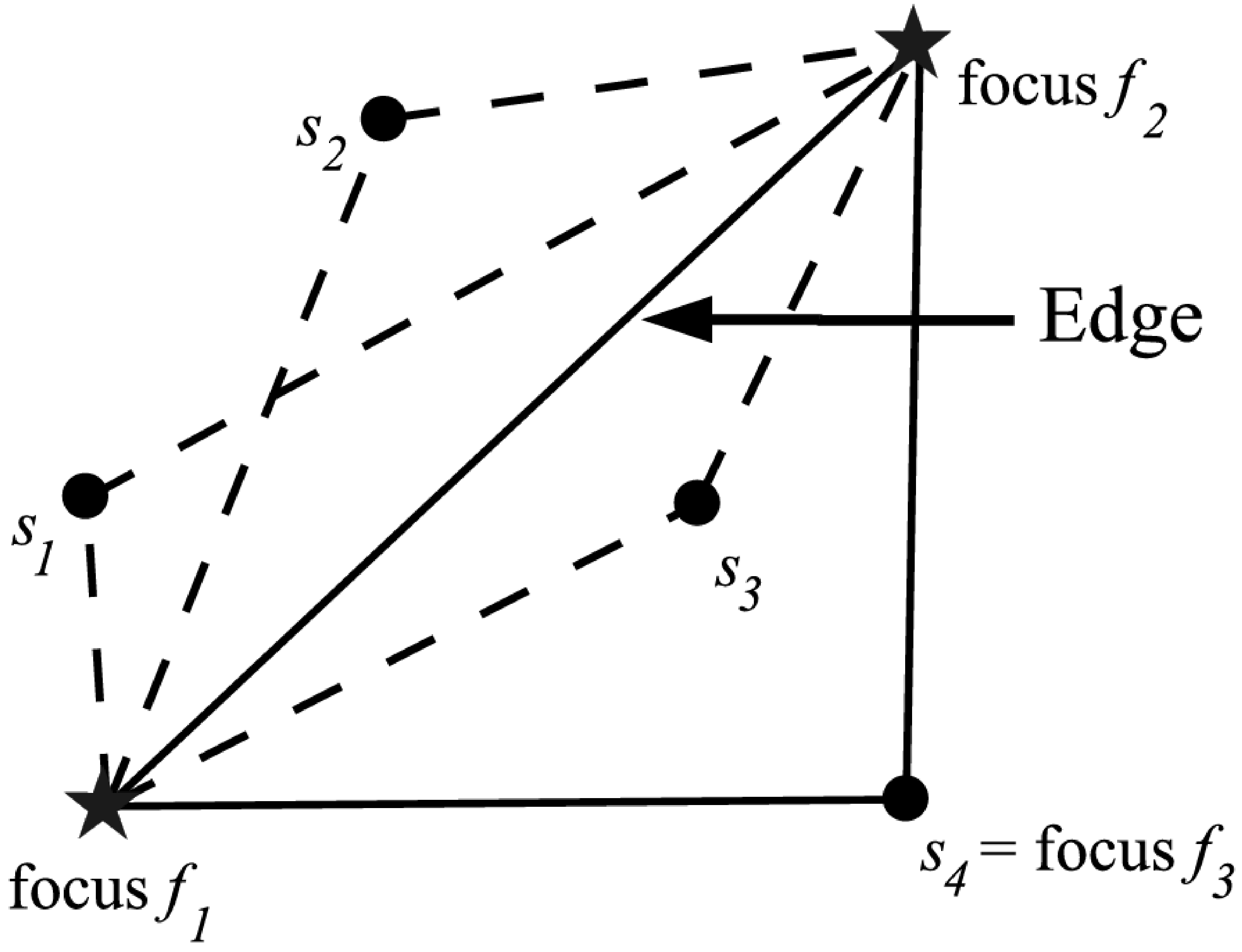


La mejor selección de *focis* son los datos en la "frontera".

Hull of *Foci (HF)*

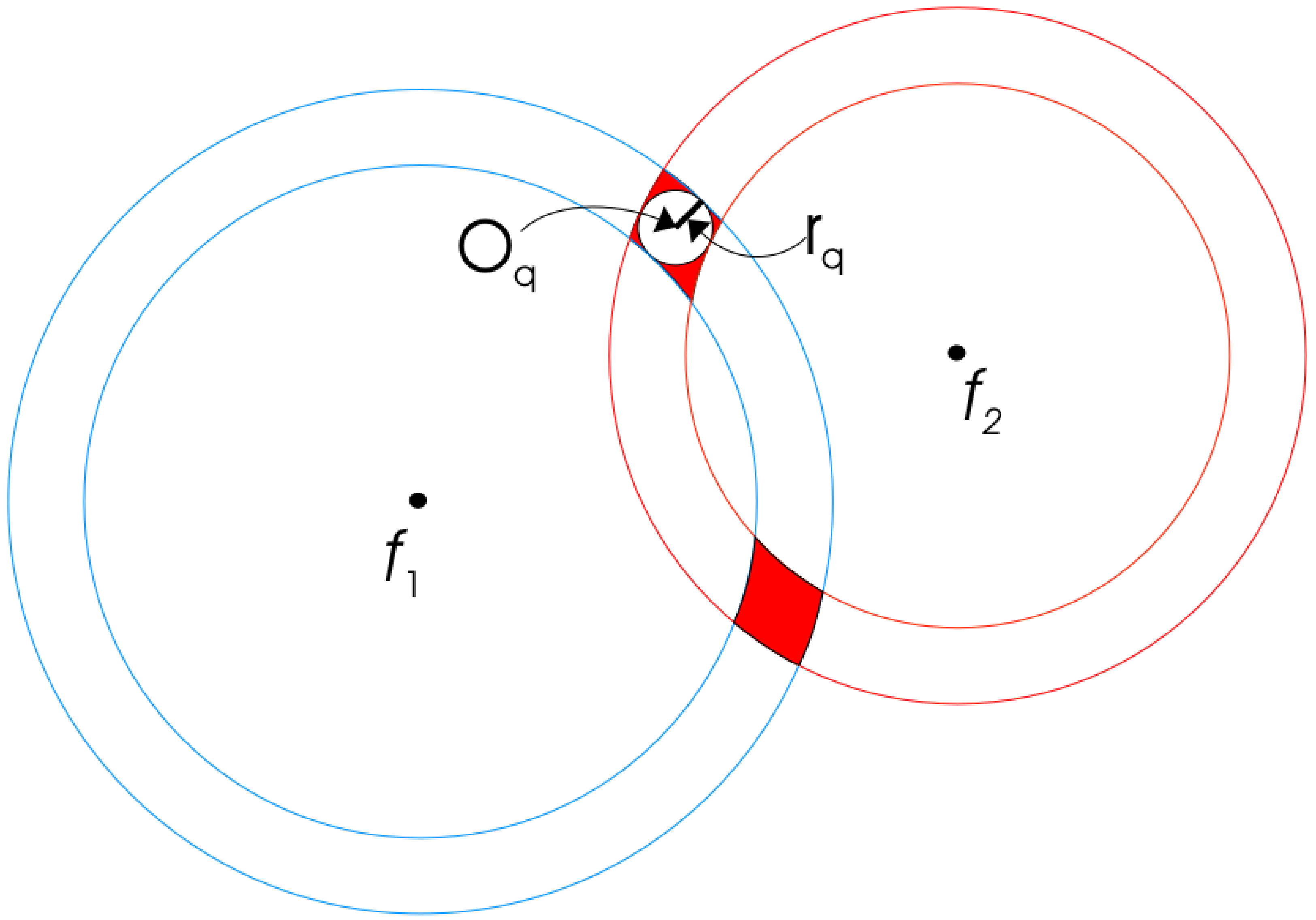
- Un punto cualquiera: p_0
- Buscar el punto más lejano a p_0: f_1
- Buscar el punto más lejano a f_1: f_2
- Los siguientes foci se obtienen con:

$$\min_{s_i} \sum_k |\underbrace{d(f_1, f_2)}_{\text{edge}} - d(f_k, s_i)|$$

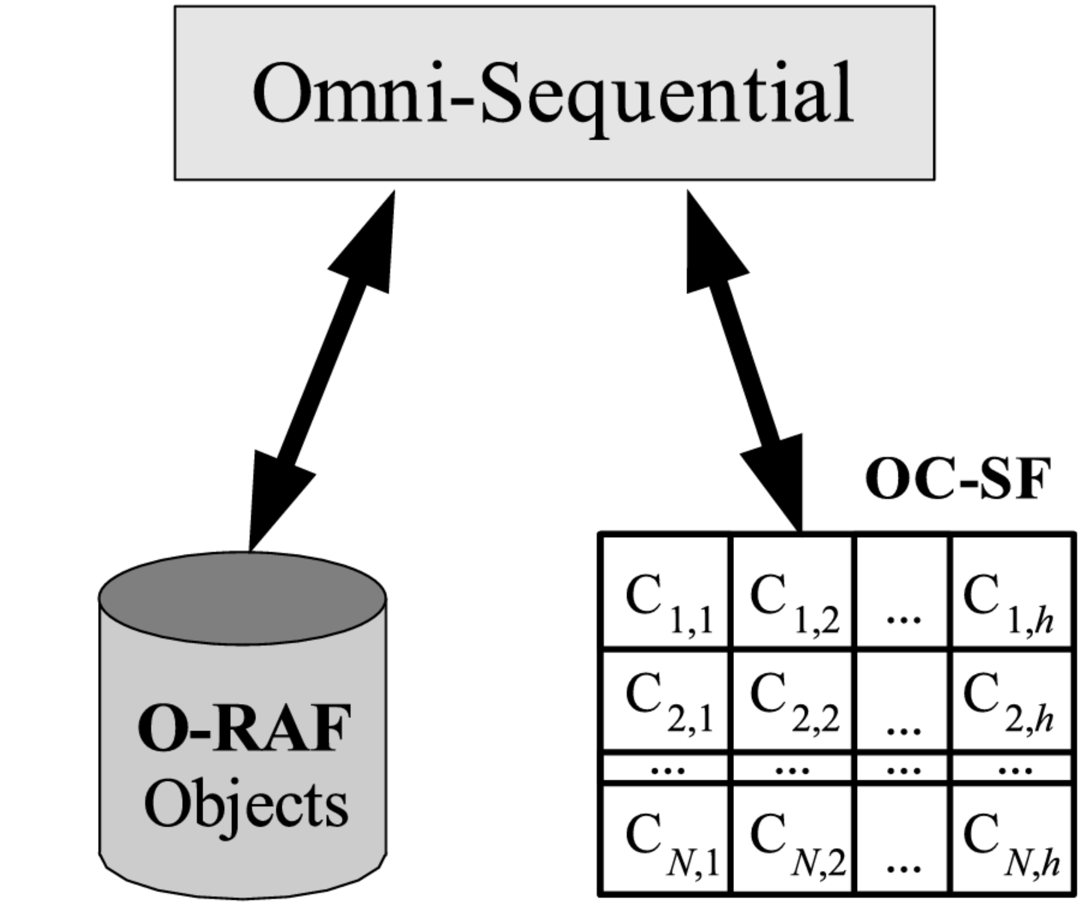


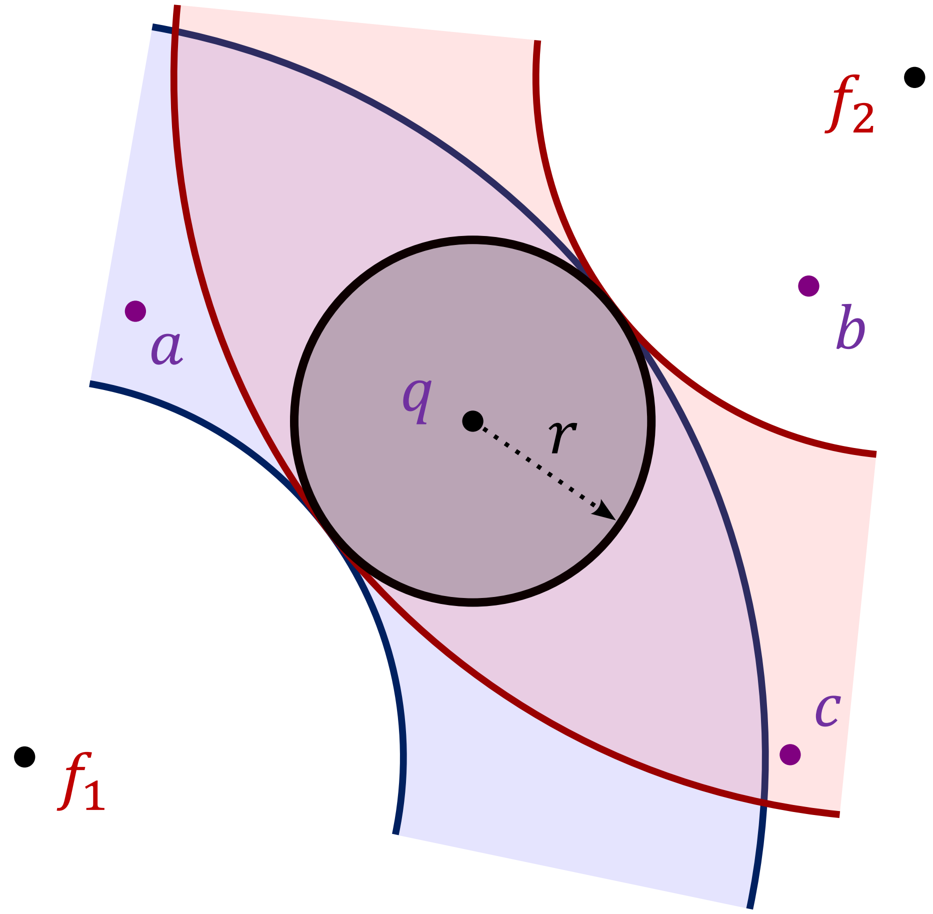
Omni-*family*

Range query



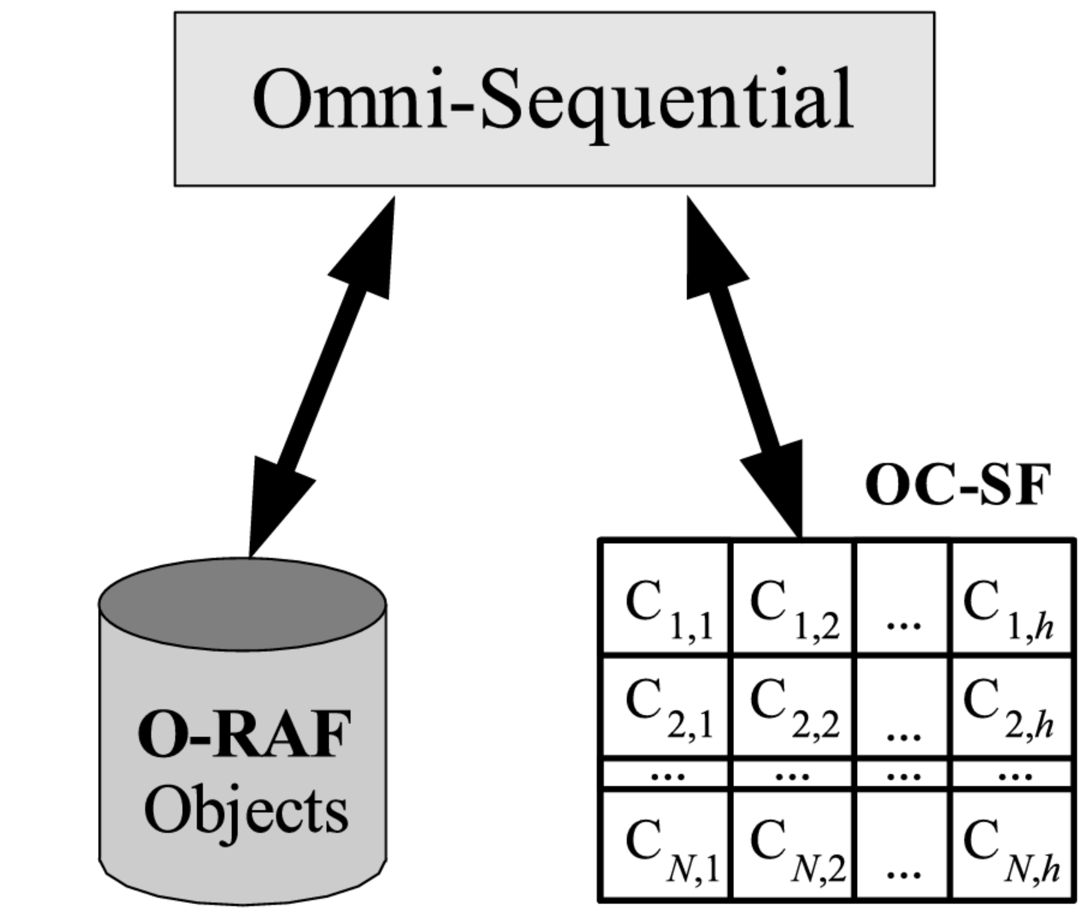
Omni-*Sequential*

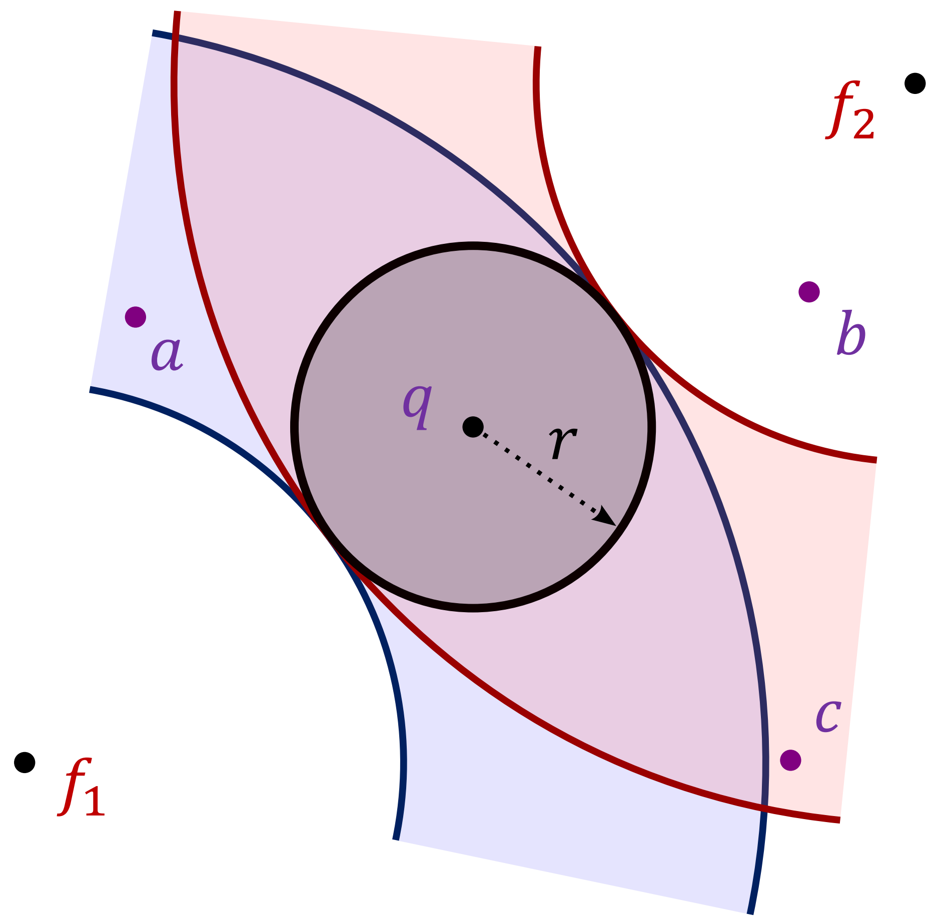




$$f_1 \rightarrow [d(q, f_1) - r, d(q, f_1) + r]$$
$$f_2 \rightarrow [d(q, f_2) - r, d(q, f_2) + r]$$

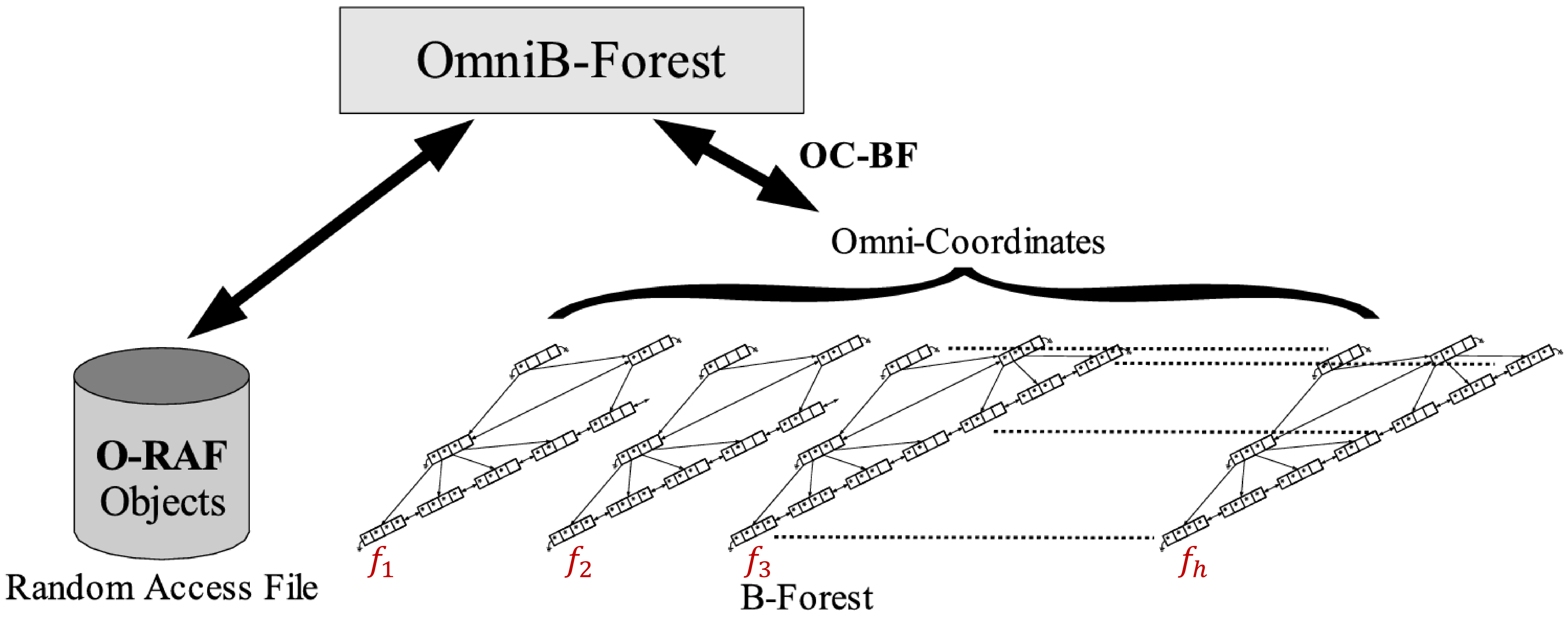
Omni-_Sequential_



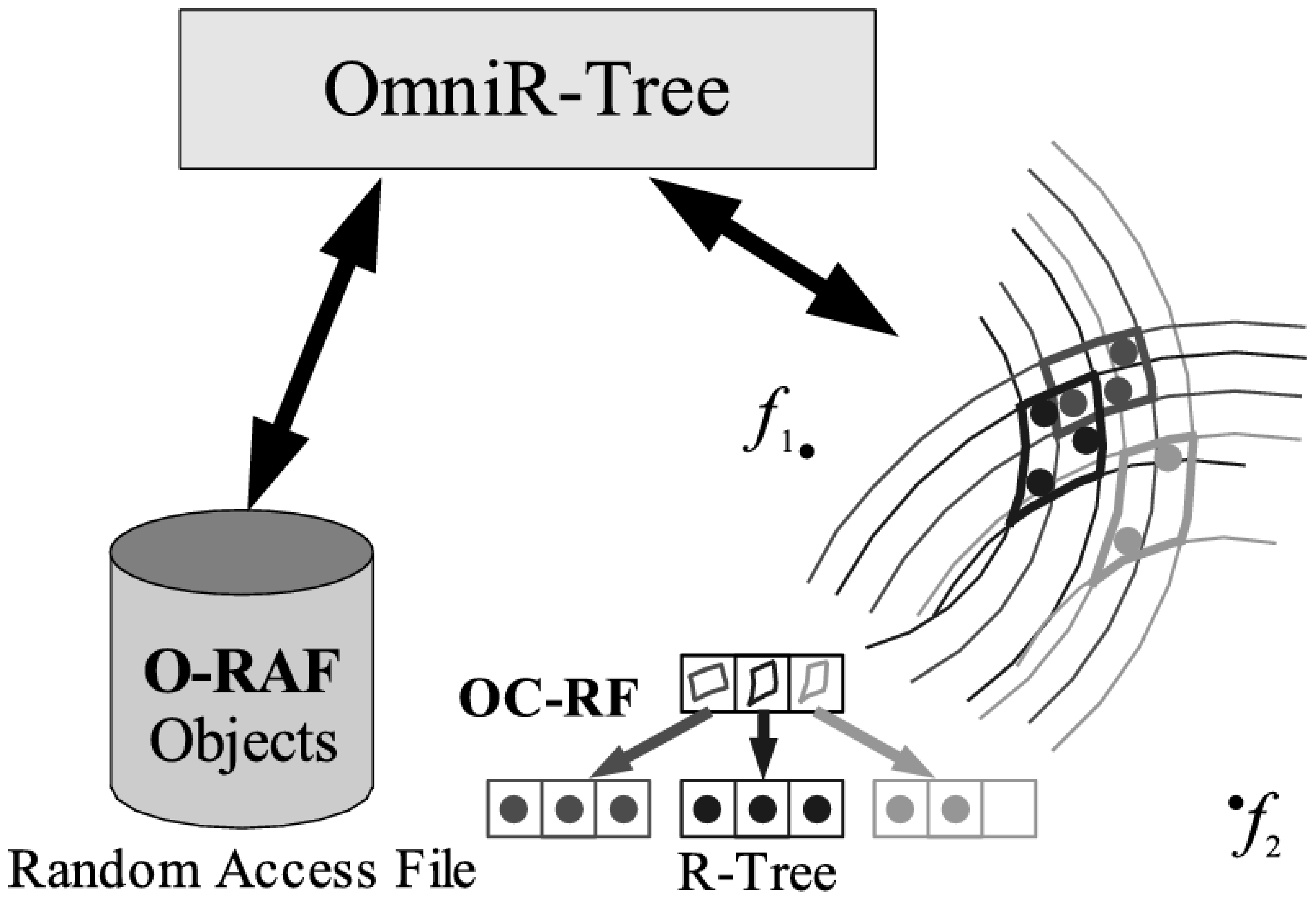


$d(a, f_1)$	$d(a, f_2)$
$d(b, f_1)$	$d(b, f_2)$
$d(c, f_1)$	$d(c, f_2)$

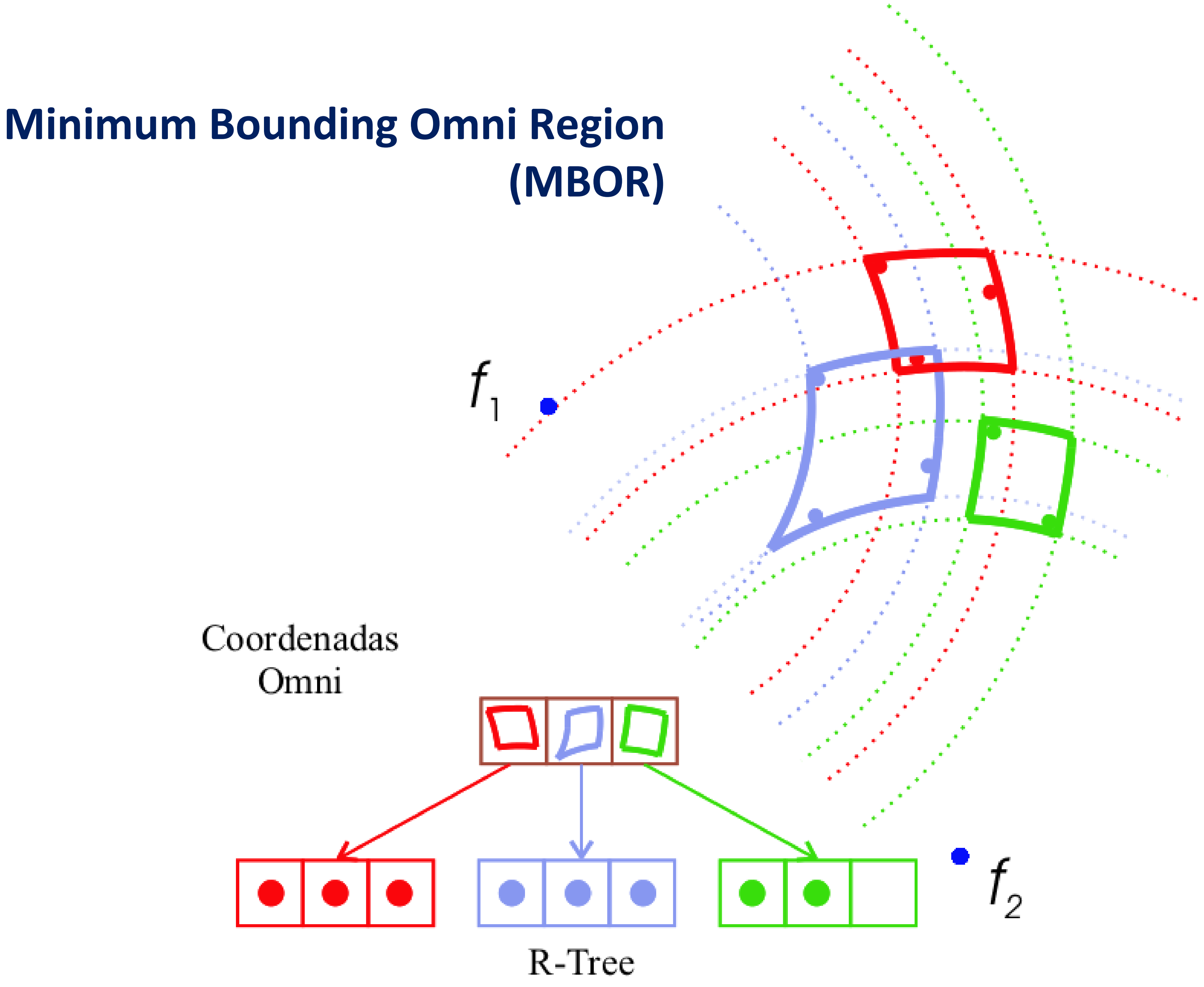
OmniB-_Forest_



OmniR-*Tree*



OmniR-*Tree*


Minimum Bounding Omni Region
(MBOR)
f_1
Coordenadas
Omni
f_2
R-Tree


UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA
MECATRÓNICA
BIOINGENIERÍA
CIENCIA DE
LA COMPUTACIÓN