



Estructuras de Datos Avanzadas 2024-1
Conjuntos de discos independientes máximos totalmente dinámicos en tiempo poli-logarítmico.

Adrián Aguilera Moreno

Profesora: Adriana Ramírez Vigueras

Contenido

1 Introducción

- Resumen del problema.

2 Algoritmo MIX

- Ejecución.
 - Restricciones de uso.

3 Estructuras de datos empleadas

- Vecino más cercano/lejano.
 - Uso de un árbol conocido ...
 - Detalles con el algoritmo MIX.
 - Generalización a dimensiones superiores.

1. Introducción

Intersecciones y el problema MIS

Descripción:

- Intersección de objetos geométricos en \mathbb{R}^n suele ser un problema muy estudiado en Geometría Computacional.



Podemos encontrar el número de intersecciones para segmentos en $\mathcal{O}(n \log n)$ siendo sensible a su salida.

Intersecciones y el problema MIS

Descripción:

- Intersección de objetos geométricos en \mathbb{R}^n suele ser un problema muy estudiado en Geometría Computacional.



Podemos encontrar el número de intersecciones para segmentos en $\mathcal{O}(n \log n)$ siendo sensible a su salida.

- ¿Podemos encontrar el máximo conjunto de objetos geométricos que no se intersecten?
 - Encontrar el Conjunto Independiente Máximo (en adelante MIS) en una gráfica es uno de los problemas NP-Completo clásicos según Karp.

Intersecciones y el problema MIS

Descripción:

- Intersecciar objetos geométricos en \mathbb{R}^n suele ser un problema muy estudiado en Geometría Computacional.



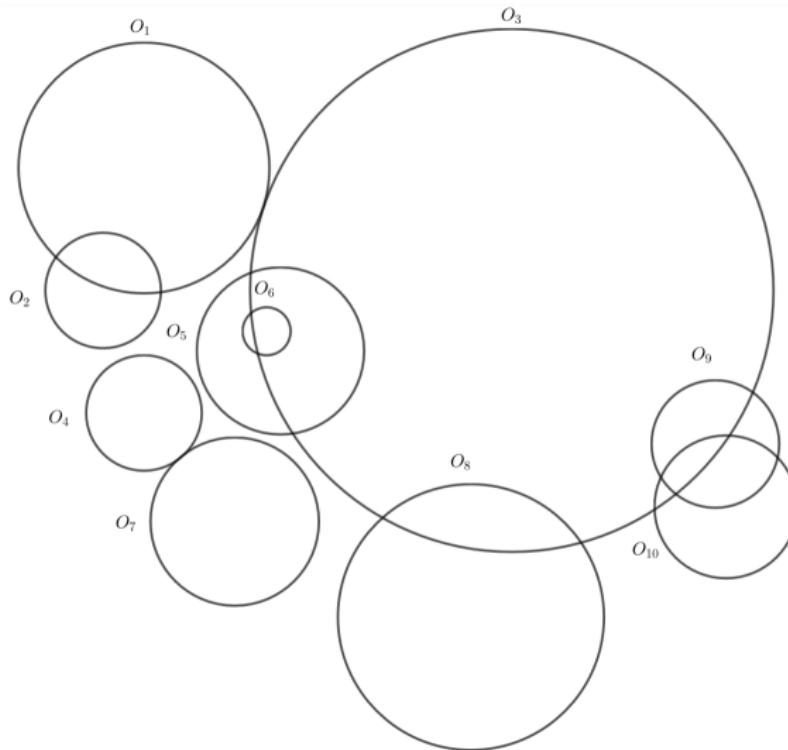
Podemos encontrar el número de intersecciones para segmentos en $\mathcal{O}(n \log n)$ siendo sensible a su salida.

- ¿Podemos encontrar el máximo conjunto de objetos geométricos que no se intersecten?
- Encontrar el Conjunto Independiente Máximo (en adelante MIS) en una gráfica es uno de los problemas NP-Completo clásicos según Karp.

Colección de objetos geométricos.

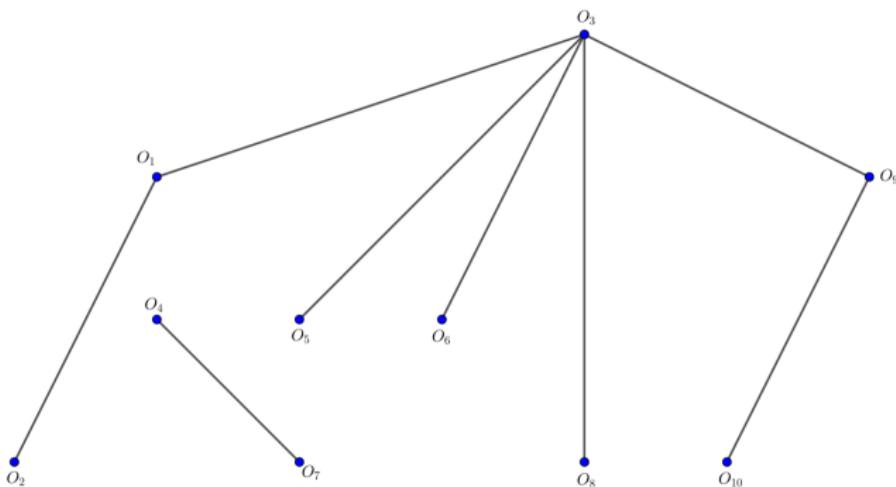
Descripción:

- Para una colección $\mathcal{L} = \{\ell_1, \dots, \ell_n\}$ de objetos geométricos.



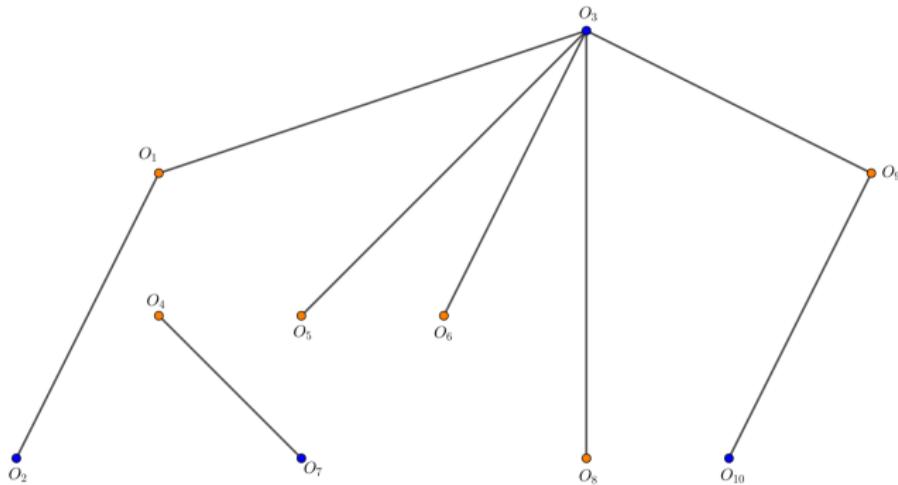
Dualidad.

- La gráfica de intersección $G_{\mathcal{L}} = (E_{\mathcal{L}}, V_{\mathcal{L}})$ de \mathcal{L} , cada objeto $\ell_i \in \mathcal{L}$ es representado por un vértice $v_i \in V_{\mathcal{L}}$ cualquier par $\ell_i \cap \ell_j \neq \emptyset$ se intersecta **SII** $(v_i, v_j) \in E_{\mathcal{L}}$.



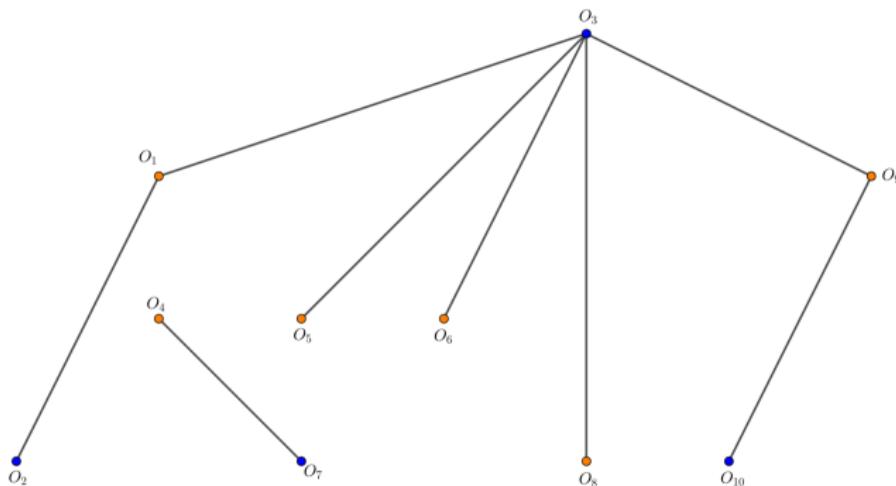
Conjuntos de discos independientes.

- Un conjunto independiente de $G_{\mathcal{L}}$ corresponde a un conjunto de objetos disjuntos por pares en \mathcal{L} .



Conjuntos de discos independientes.

- Un conjunto independiente de $G_{\mathcal{L}}$ corresponde a un conjunto de objetos disjuntos por pares en \mathcal{L} .



- Nuestro objetivo es mantener una estructura de datos que pueda mantener eficientemente un conjunto independiente $S \subseteq L$ cuyo tamaño sea una aproximación de factor constante del MIS de L en cualquier momento, con tiempos de actualización polilogarítmicos para inserciones y eliminaciones.

Antecedentes

Antecedentes:

- En el 2022 Jean Cardinal et. al. crean el algoritmo *offline* MIX para aproximar el conjunto independiente máximo de objetos gruesos en \mathbb{R}^d con $d \in o(1)$ que se ejecuta en tiempo $\mathcal{O}(\alpha \log \alpha)$ con α el valor del máximo candidato a solución (en adelante OPT).
- La función MIX utiliza dos estructuras de datos, una de reciente creación llamada *Estructura del vecino más cercano/lejano*, que se basa en la idea del diagrama de Voronoi.
- Chan et. al. logran generalizar la estructura anterior a dimensiones superiores, siempre que sean constante respecto al número de objetos geométricos.

Antecedentes

Antecedentes:

- En el 2022 Jean Cardinal et. al. crean el algoritmo *offline* MIX para aproximar el conjunto independiente máximo de objetos gruesos en \mathbb{R}^d con $d \in o(1)$ que se ejecuta en tiempo $\mathcal{O}(\alpha \log \alpha)$ con α el valor del máximo candidato a solución (en adelante OPT).
- La función MIX utiliza dos estructuras de datos, una de reciente creación llamada *Estructura del vecino más cercano/lejano*, que se basa en la idea del diagrama de Voronoi.
- Chan et. al. logran generalizar la estructura anterior a dimensiones superiores, siempre que sean constante respecto al número de objetos geométricos.

Antecedentes

Antecedentes:

- En el 2022 Jean Cardinal et. al. crean el algoritmo *offline* MIX para aproximar el conjunto independiente máximo de objetos gruesos en \mathbb{R}^d con $d \in o(1)$ que se ejecuta en tiempo $\mathcal{O}(\alpha \log \alpha)$ con α el valor del máximo candidato a solución (en adelante OPT).
 - La función MIX utiliza dos estructuras de datos, una de reciente creación llamada *Estructura del vecino más cercano/lejano*, que se basa en la idea del diagrama de Voronoi.
 - Chan et. al. logran generalizar la estructura anterior a dimensiones superiores, siempre que sean constante respecto al número de objetos geométricos.

2. Algoritmo MIX

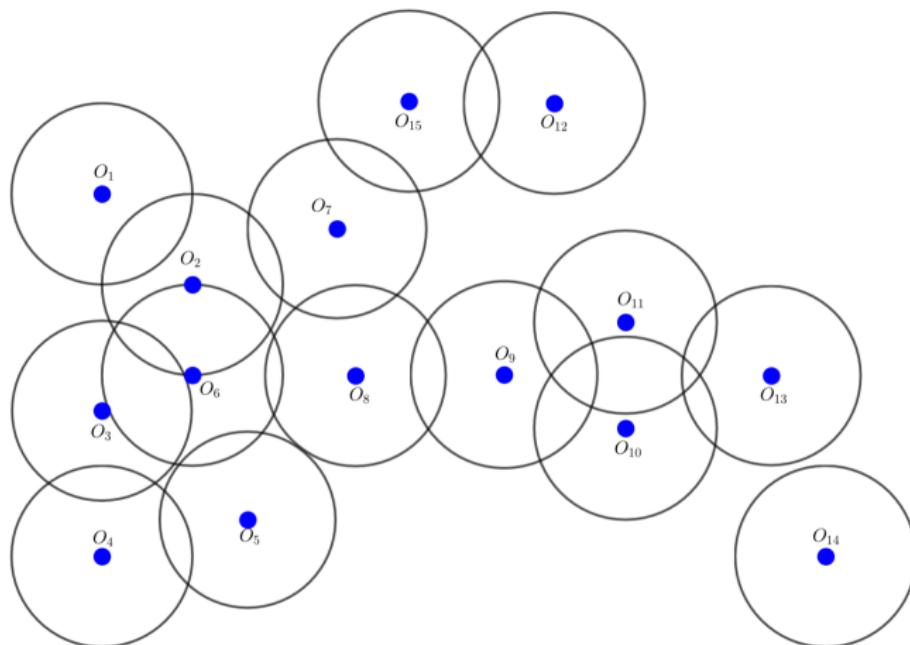
Cuestión a abordar

¿Existe un algoritmo que para un conjunto dinámico, dado, de discos en el plano mantenga un conjunto independiente máximo aproximado de factor constante en tiempo de actualización poli-logarítmico?



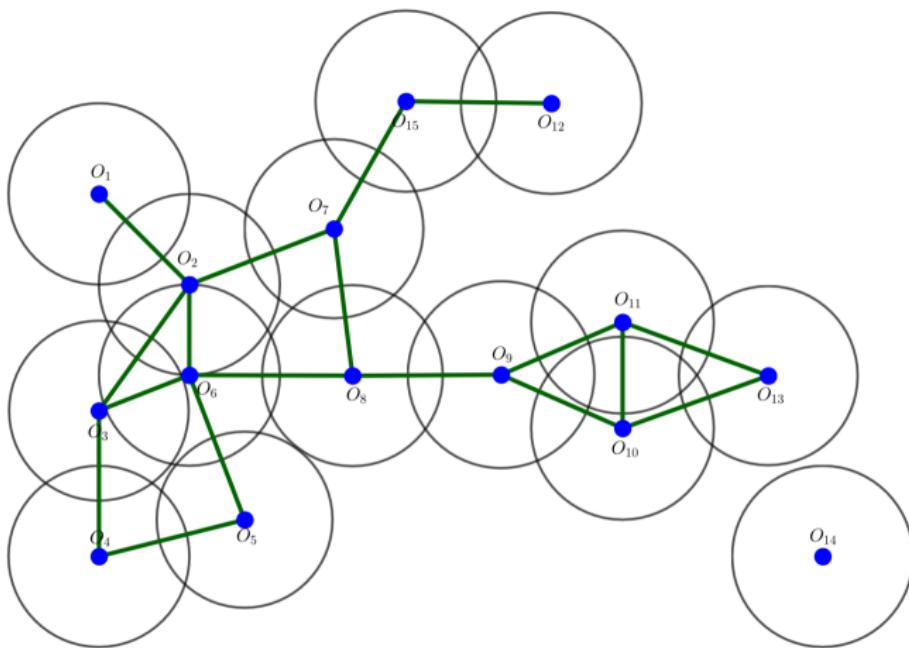
Ejecución

Supongamos el siguiente conjunto de discos unitarios en el plano



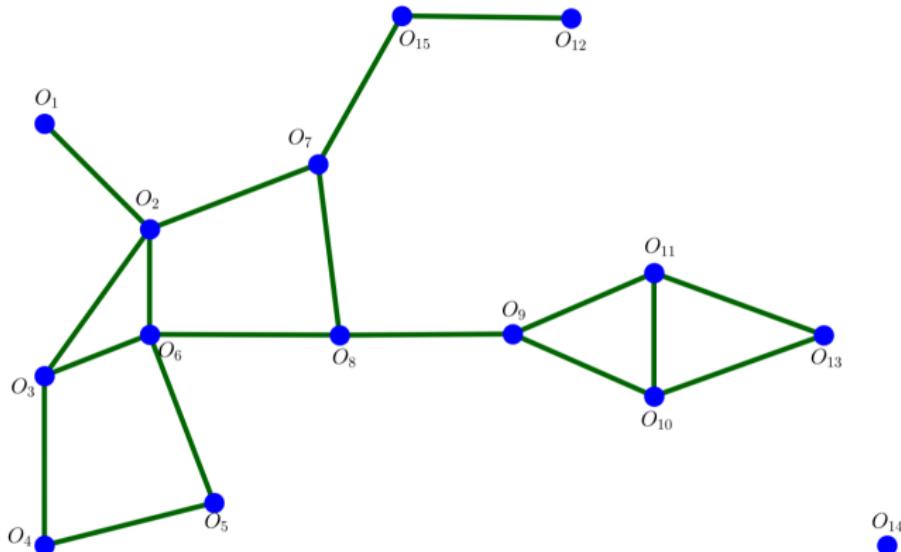
1

Obtengamos su dualidad



...

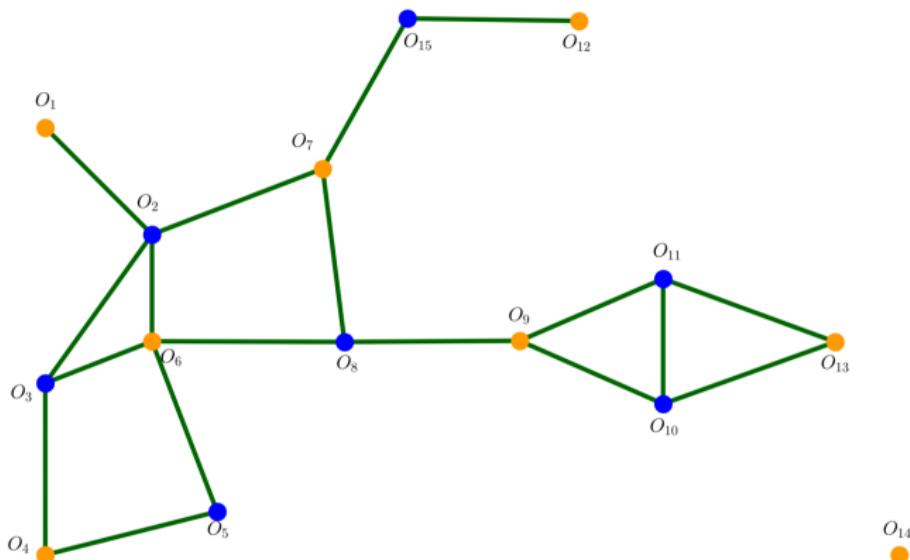
Observemos su gráfica dual



Obs. Existe un preprocessamiento que se encarga de exhibir candidatos a OPT.

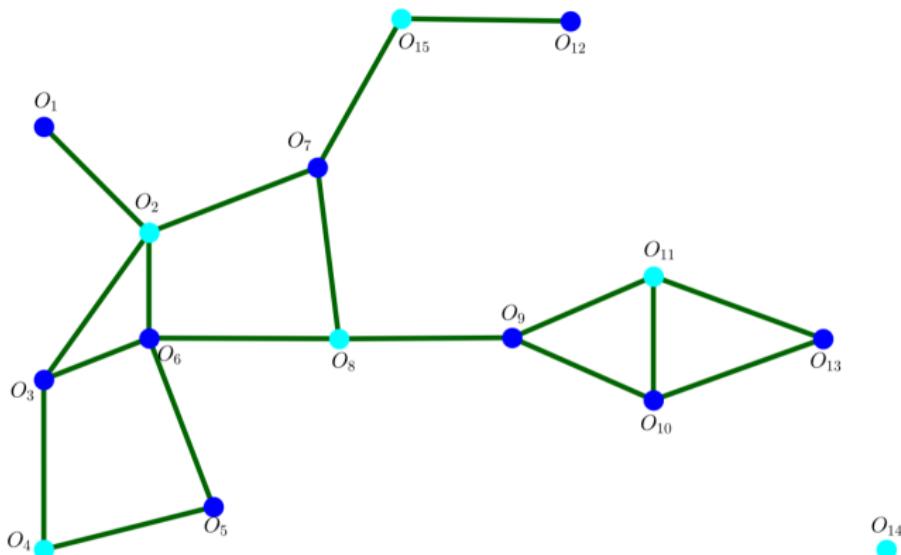
...

Sea $S_1 = \{O_1, O_6, O_4, O_7, O_{12}, O_9, O_{13}, O_{14}\}$



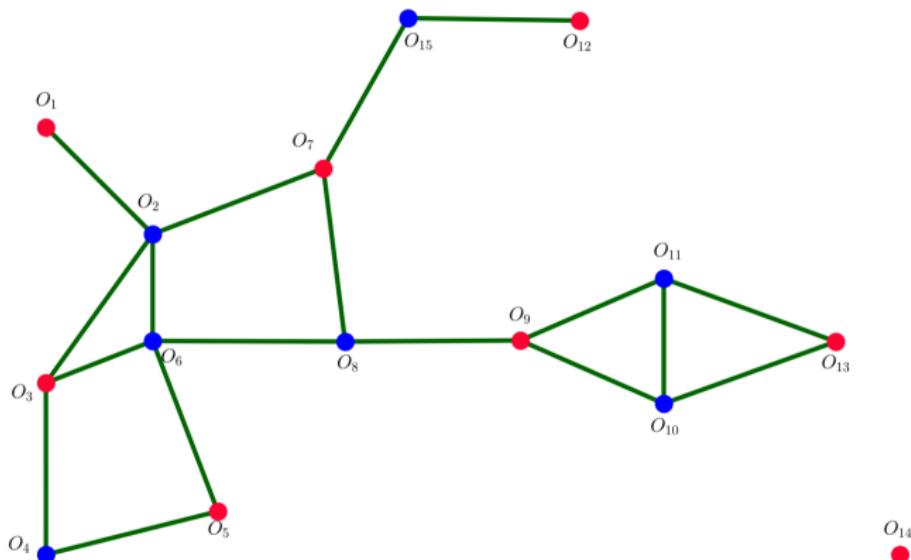
...

Sea $S_2 = \{O_2, O_4, O_8, O_{15}, O_{11}, O_{14}\}$



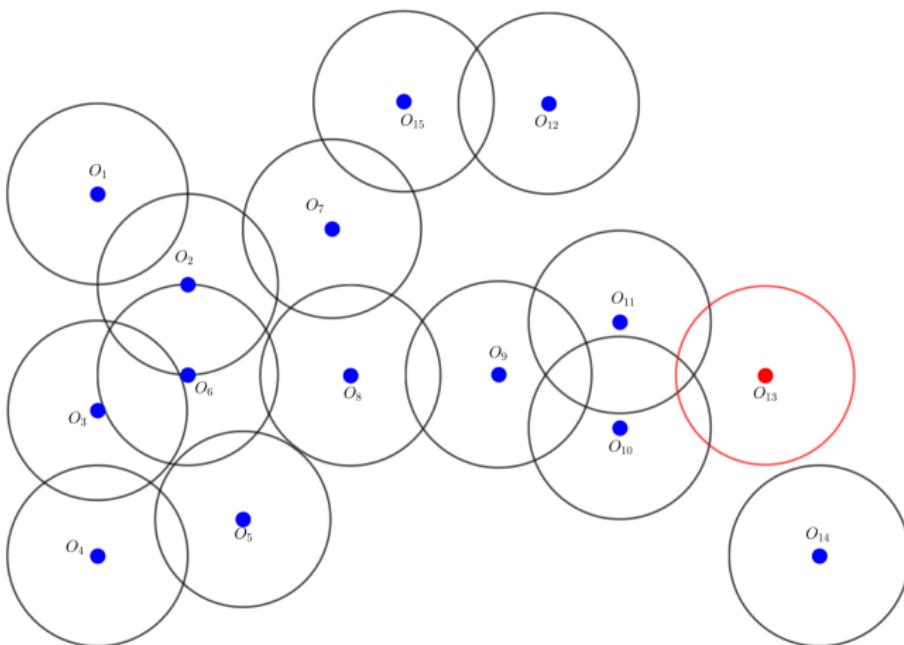
...

Sea $S_3 = \{O_1, O_3, O_5, O_7, O_{12}, O_9, O_{13}, O_{14}\}$



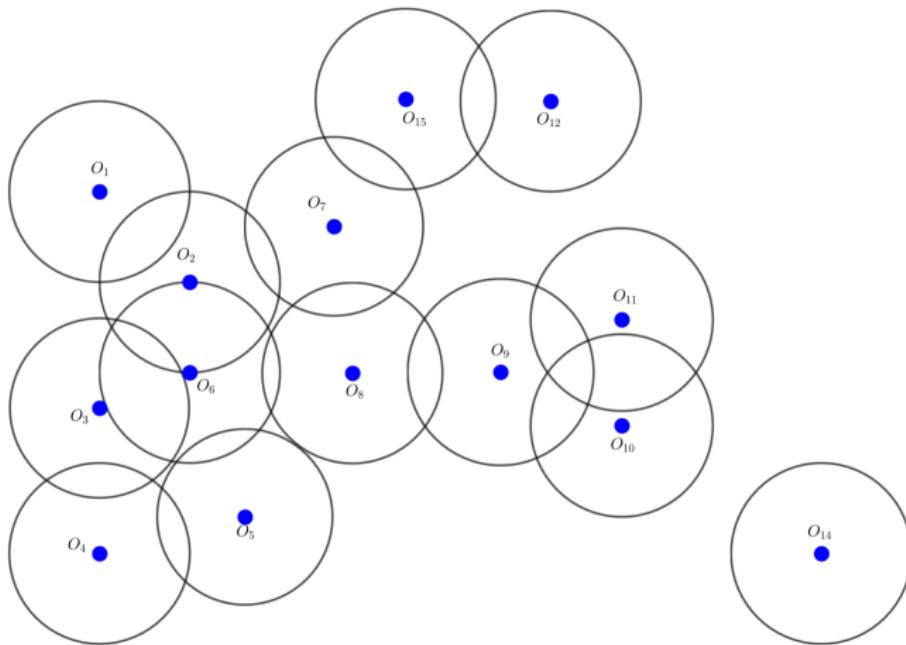
...

Sea $S_{MIS} = \{S_1, S_2, S_3\}$. Elijamos un candidato de S_{MIS} tal que sea de mayor tamaño, digamos $S = S_1$. Ahora, eliminemos O_{13} .



...

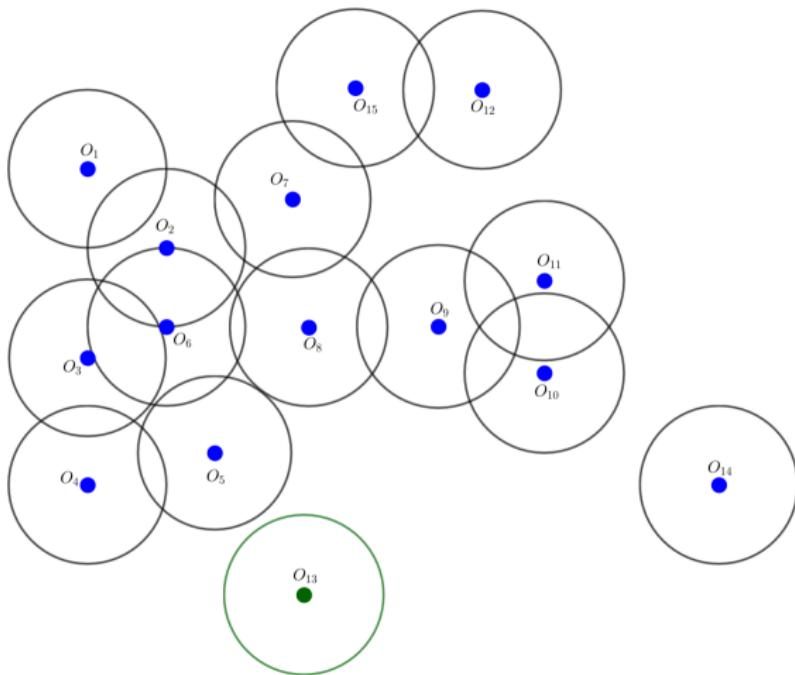
Cómo hemos hecho una eliminación, debemos actualizar $S_1 = S_1/O_{13}$,
 $S_2 = S_2/O_{13}$, $S_3 = S_3/O_{13}$.



$$S = \{O_1, O_6, O_4, O_7, O_{12}, O_9, O_{14}\}.$$

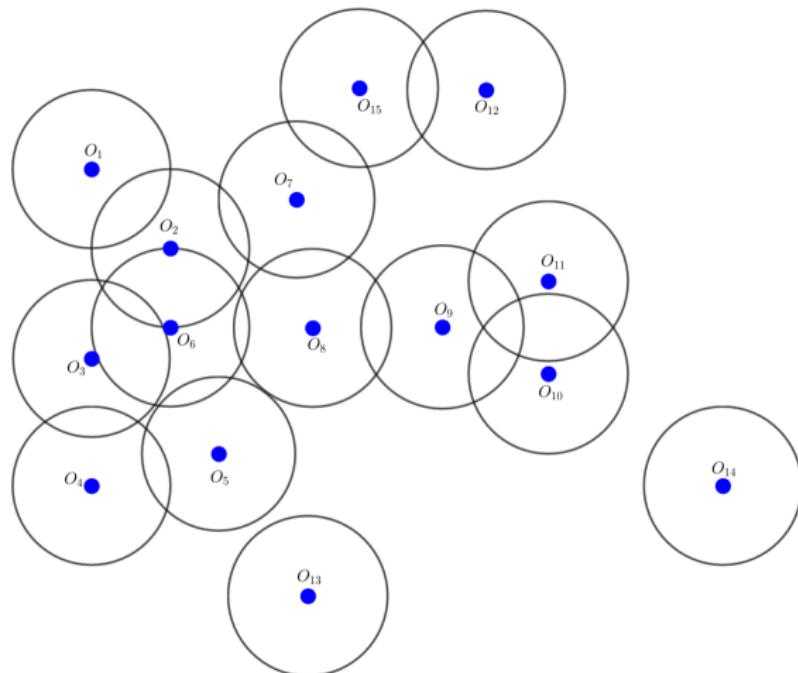
...

Añadamos O_{13} (un nuevo disco)

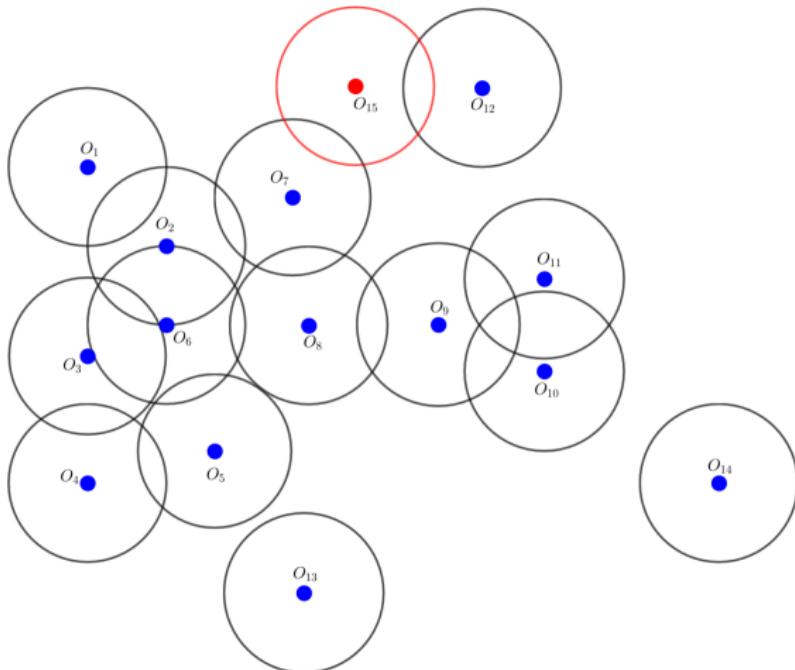


...

Verificamos si este disco debe formar parte de los candidatos, en este caso actualizamos $S_1 = S_1 \cup \{O_{13}\}$, $S_2 = S_2 \cup \{O_{13}\}$, $S_3 = S_3 \cup \{O_{13}\}$. Y $S = \{O_1, O_6, O_4, O_7, O_{12}, O_{13}, O_9, O_{14}\}$.

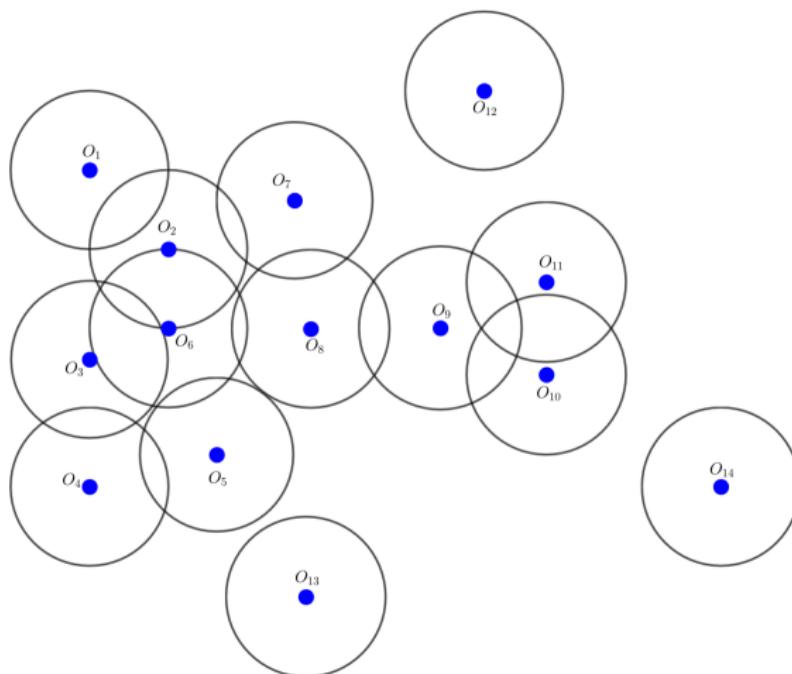


Eliminemos O_{15}



...

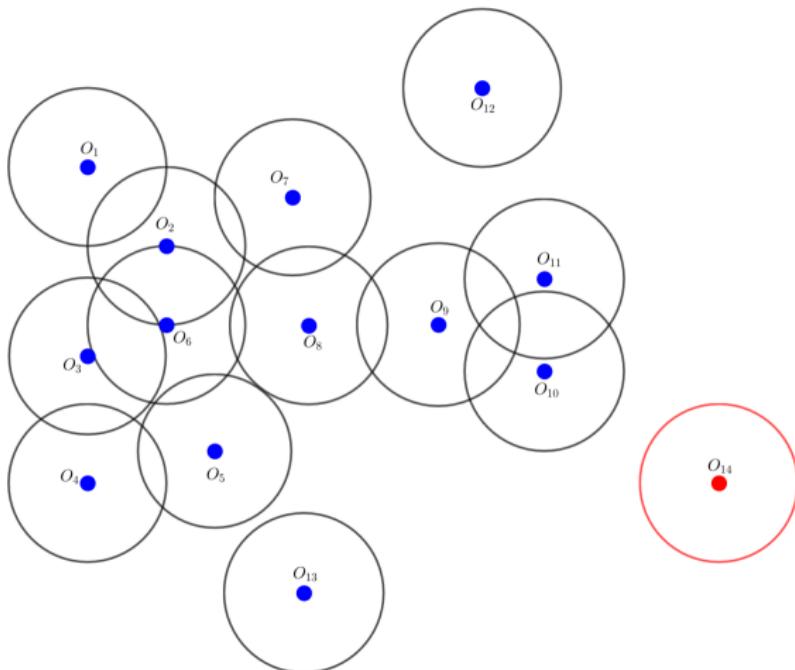
Verificamos si esta eliminación afecta a nuestros candidatos, en este caso no se ven afectados y continuamos



$$S = \{O_1, O_6, O_4, O_7, O_{12}, O_{13}, O_9, O_{14}\}$$

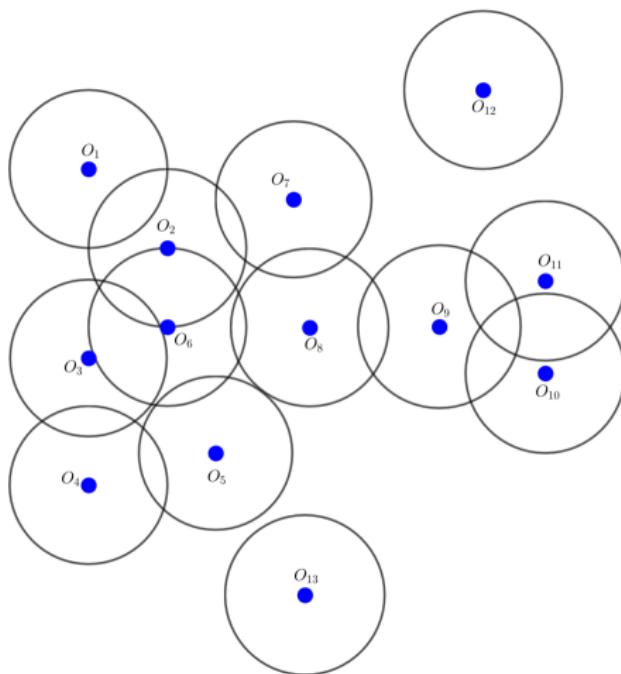
...

Eliminamos O_{14}



...

Actualizamos los estados de nuestros candidatos $S_1 = S_1/O_{14}$, $S_2 = S_2/O_{14}$, $S_3 = S_3/O_{14}$.

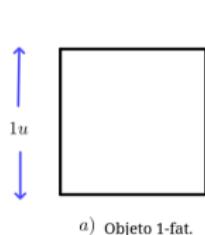


$S = \{O_1, O_6, O_4, O_7, O_{12}, O_{13}, O_9\}$. **FIN.**

Restricciones

Restricciones:

- La función de transición suave MIX sólo funciona en objetos geométricos f -fat con $f \in \mathbb{R}$ y $f > 0$.



a) Objeto 1-fat.



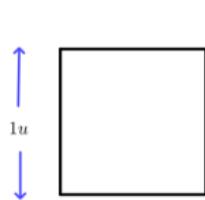
b) Objeto 5-fat.

- La función recibe dos estados S_i, S_f y transita de S_i a S_f en $\mathcal{O}(\alpha \log \alpha)$, pues debe ordenar cada conjunto de entrada.
- Sólo funciona para \mathbb{R}^d con d constante.
- Se pueden realizar, a lo más, $5u$ eliminaciones o adiciones, con $u \in \mathbb{Z}^+$, asumiendo $u > 0$.
- Las eliminaciones se realizan de manera directa, las adiciones se realizan al final de la ejecución del algoritmo. Basta guardar las adiciones en una cola.
- La cantidad de conjuntos candidatos es $K \in \mathcal{O}(1)$.

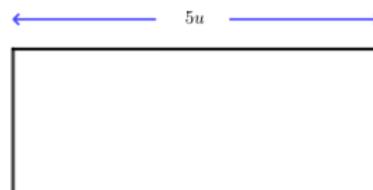
Restricciones

Restricciones:

- La función de transición suave MIX sólo funciona en objetos geométricos f -fat con $f \in \mathbb{R}$ y $f > 0$.



a) Objeto 1-fat.



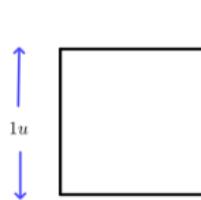
b) Objeto 5-fat.

- La función recibe dos estados S_i, S_f y transita de S_i a S_f en $\mathcal{O}(\alpha \log \alpha)$, pues debe ordenar cada conjunto de entrada.
- Sólo funciona para \mathbb{R}^d con d constante.
- Se pueden realizar, a lo más, $5u$ eliminaciones o adiciones, con $u \in \mathbb{Z}^+$, asumiendo $u > 0$.
- Las eliminaciones se realizan de manera directa, las adiciones se realizan al final de la ejecución del algoritmo. Basta guardar las adiciones en una cola.
- La cantidad de conjuntos candidatos es $K \in \mathcal{O}(1)$.

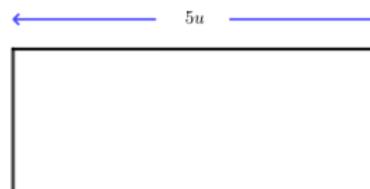
Restricciones

Restricciones:

- La función de transición suave MIX sólo funciona en objetos geométricos f -fat con $f \in \mathbb{R}$ y $f > 0$.



a) Objeto 1-fat.



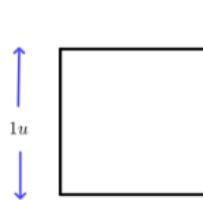
b) Objeto 5-fat.

- La función recibe dos estados S_i, S_f y transita de S_i a S_f en $\mathcal{O}(\alpha \log \alpha)$, pues debe ordenar cada conjunto de entrada.
- Sólo funciona para \mathbb{R}^d con d constante.
- Se pueden realizar, a lo más, $5u$ eliminaciones o adiciones, con $u \in \mathbb{Z}^+$, asumiendo $u > 0$.
- Las eliminaciones se realizan de manera directa, las adiciones se realizan al final de la ejecución del algoritmo. Basta guardar las adiciones en una cola.
- La cantidad de conjuntos candidatos es $K \in \mathcal{O}(1)$.

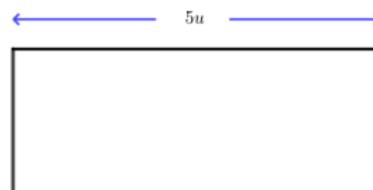
Restricciones

Restricciones:

- La función de transición suave MIX sólo funciona en objetos geométricos f -fat con $f \in \mathbb{R}$ y $f > 0$.



a) Objeto 1-fat.



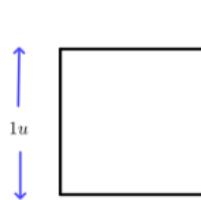
b) Objeto 5-fat.

- La función recibe dos estados S_i, S_f y transita de S_i a S_f en $\mathcal{O}(\alpha \log \alpha)$, pues debe ordenar cada conjunto de entrada.
- Sólo funciona para \mathbb{R}^d con d constante.
- Se pueden realizar, a lo más, $5u$ eliminaciones o adiciones, con $u \in \mathbb{Z}^+$, asumiendo $u > 0$.
- Las eliminaciones se realizan de manera directa, las adiciones se realizan al final de la ejecución del algoritmo. Basta guardar las adiciones en una cola.
- La cantidad de conjuntos candidatos es $K \in \mathcal{O}(1)$.

Restricciones

Restricciones:

- La función de transición suave MIX sólo funciona en objetos geométricos f -fat con $f \in \mathbb{R}$ y $f > 0$.



a) Objeto 1-fat.



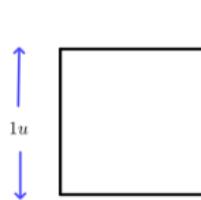
b) Objeto 5-fat.

- La función recibe dos estados S_i, S_f y transita de S_i a S_f en $\mathcal{O}(\alpha \log \alpha)$, pues debe ordenar cada conjunto de entrada.
- Sólo funciona para \mathbb{R}^d con d constante.
- Se pueden realizar, a lo más, $5u$ eliminaciones o adiciones, con $u \in \mathbb{Z}^+$, asumiendo $u > 0$.
- Las eliminaciones se realizan de manera directa, las adiciones se realizan al final de la ejecución del algoritmo. Basta guardar las adiciones en una cola.
- La cantidad de conjuntos candidatos es $K \in \mathcal{O}(1)$.

Restricciones

Restricciones:

- La función de transición suave MIX sólo funciona en objetos geométricos f -fat con $f \in \mathbb{R}$ y $f > 0$.



a) Objeto 1-fat.



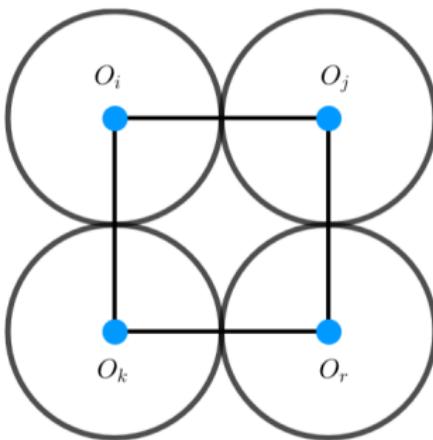
b) Objeto 5-fat.

- La función recibe dos estados S_i, S_f y transita de S_i a S_f en $\mathcal{O}(\alpha \log \alpha)$, pues debe ordenar cada conjunto de entrada.
- Sólo funciona para \mathbb{R}^d con d constante.
- Se pueden realizar, a lo más, $5u$ eliminaciones o adiciones, con $u \in \mathbb{Z}^+$, asumiendo $u > 0$.
- Las eliminaciones se realizan de manera directa, las adiciones se realizan al final de la ejecución del algoritmo. Basta guardar las adiciones en una cola.
- La cantidad de conjuntos candidatos es $K \in \mathcal{O}(1)$.

3. Estructuras de datos empleadas

Observaciones

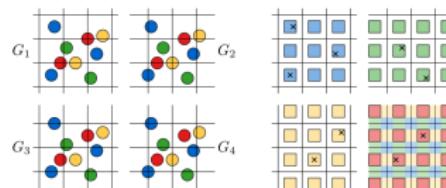
Para un cuadrado 2×2 , puede contener, a lo más, 4 discos unitarios que no se intersecten



Construcción de rejillas desplazadas

Construcción:

- Definimos cuadrículas, por conjunto candidato, G_1, \dots, G_k . Estas pueden ser distintas y en ocasiones iguales

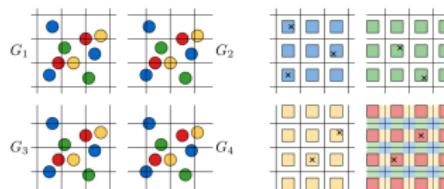


- Cada celda puede ser definida de longitud 4 o trabajar celdas de longitud 2 de manera unificada.
- Para G_1 tenemos las líneas $\{x = 4i\}$ y $\{y = 4i\}$ con $i \in \mathbb{Z}^+$. Análogo para longitud 2.
- Para las siguientes G_2 y G_3 marcamos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$ respectivamente.
- G_4 se desplaza en ambos sentidos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$. Esto se sigue de manera iterativa.
- La celda cumple la propiedad de consultar por intersecciones de discos más cercanos y encontrar el disco más lejano dado un disco. La última propiedad no la utilizaremos en nuestro caso.

Construcción de rejillas desplazadas

Construcción:

- Definimos cuadrículas, por conjunto candidato, G_1, \dots, G_k . Estas pueden ser distintas y en ocasiones iguales

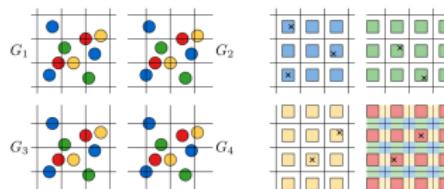


- Cada celda puede ser definida de longitud 4 o trabajar celdas de longitud 2 de manera unificada.
- Para G_1 tenemos las líneas $\{x = 4i\}$ y $\{y = 4i\}$ con $i \in \mathbb{Z}^+$. Análogo para longitud 2.
- Para las siguientes G_2 y G_3 marcamos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$ respectivamente.
- G_4 se desplaza en ambos sentidos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$. Esto se sigue de manera iterativa.
- La celda cumple la propiedad de consultar por intersecciones de discos más cercanos y encontrar el disco más lejano dado un disco. La última propiedad no la utilizaremos en nuestro caso.

Construcción de rejillas desplazadas

Construcción:

- Definimos cuadrículas, por conjunto candidato, G_1, \dots, G_k . Estas pueden ser distintas y en ocasiones iguales

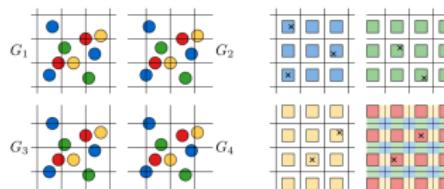


- Cada celda puede ser definida de longitud 4 o trabajar celdas de longitud 2 de manera unificada.
- Para G_1 tenemos las líneas $\{x = 4i\}$ y $\{y = 4i\}$ con $i \in \mathbb{Z}^+$. Análogo para longitud 2.
- Para las siguientes G_2 y G_3 marcamos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$ respectivamente.
- G_4 se desplaza en ambos sentidos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$. Esto se sigue de manera iterativa.
- La celda cumple la propiedad de consultar por intersecciones de discos más cercanos y encontrar el disco más lejano dado un disco. La última propiedad no la utilizaremos en nuestro caso.

Construcción de rejillas desplazadas

Construcción:

- Definimos cuadrículas, por conjunto candidato, G_1, \dots, G_k . Estas pueden ser distintas y en ocasiones iguales

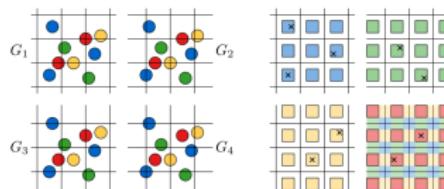


- Cada celda puede ser definida de longitud 4 o trabajar celdas de longitud 2 de manera unificada.
- Para G_1 tenemos las líneas $\{x = 4i\}$ y $\{y = 4i\}$ con $i \in \mathbb{Z}^+$. Análogo para longitud 2.
- Para las siguientes G_2 y G_3 marcamos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$ respectivamente.
- G_4 se desplaza en ambos sentidos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$. Esto se sigue de manera iterativa.
- La celda cumple la propiedad de consultar por intersecciones de discos más cercanos y encontrar el disco más lejano dado un disco. La última propiedad no la utilizaremos en nuestro caso.

Construcción de rejillas desplazadas

Construcción:

- Definimos cuadrículas, por conjunto candidato, G_1, \dots, G_k . Estas pueden ser distintas y en ocasiones iguales

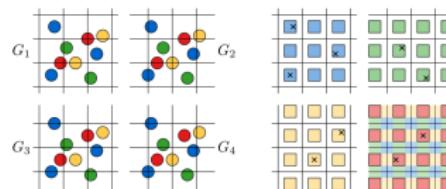


- Cada celda puede ser definida de longitud 4 o trabajar celdas de longitud 2 de manera unificada.
- Para G_1 tenemos las líneas $\{x = 4i\}$ y $\{y = 4i\}$ con $i \in \mathbb{Z}^+$. Análogo para longitud 2.
- Para las siguientes G_2 y G_3 marcamos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$ respectivamente.
- G_4 se desplaza en ambos sentidos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$. Esto se sigue de manera iterativa.
- La celda cumple la propiedad de consultar por intersecciones de discos más cercanos y encontrar el disco más lejano dado un disco. La última propiedad no la utilizaremos en nuestro caso.

Construcción de rejillas desplazadas

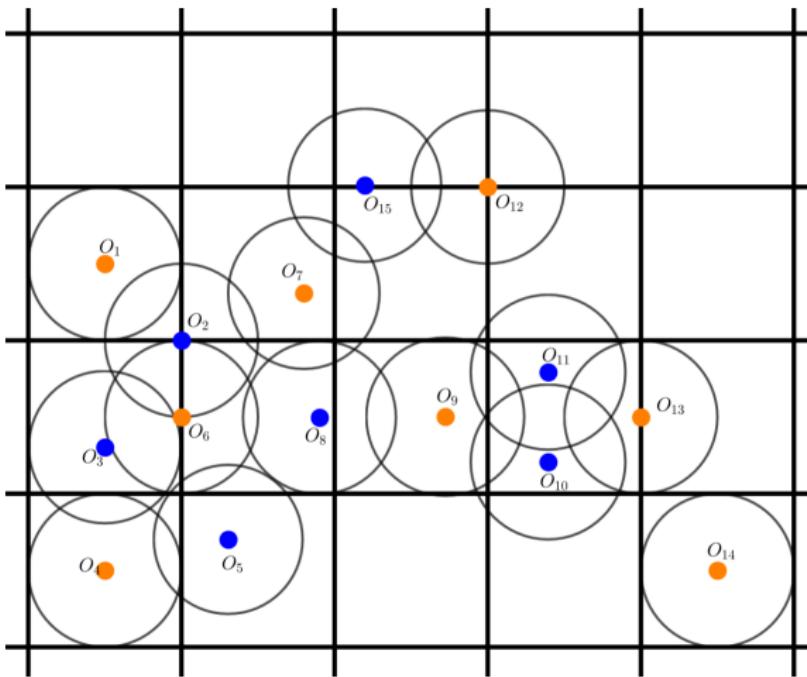
Construcción:

- Definimos cuadrículas, por conjunto candidato, G_1, \dots, G_k . Estas pueden ser distintas y en ocasiones iguales

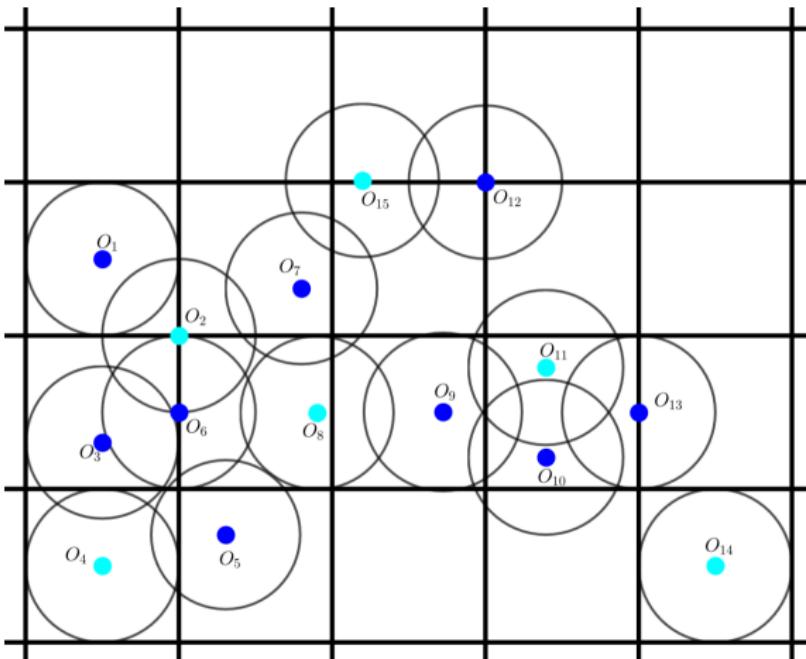


- Cada celda puede ser definida de longitud 4 o trabajar celdas de longitud 2 de manera unificada.
 - Para G_1 tenemos las líneas $\{x = 4i\}$ y $\{y = 4i\}$ con $i \in \mathbb{Z}^+$. Análogo para longitud 2.
 - Para las siguientes G_2 y G_3 marcamos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$ respectivamente.
 - G_4 se desplaza en ambos sentidos $\{x = 4i + 2\}$ y $\{y = 4i + 2\}$. Esto se sigue de manera iterativa.
 - La celda cumple la propiedad de consultar por intersecciones de discos más cercanos y encontrar el disco más lejano dado un disco. La última propiedad no la utilizaremos en nuestro caso.

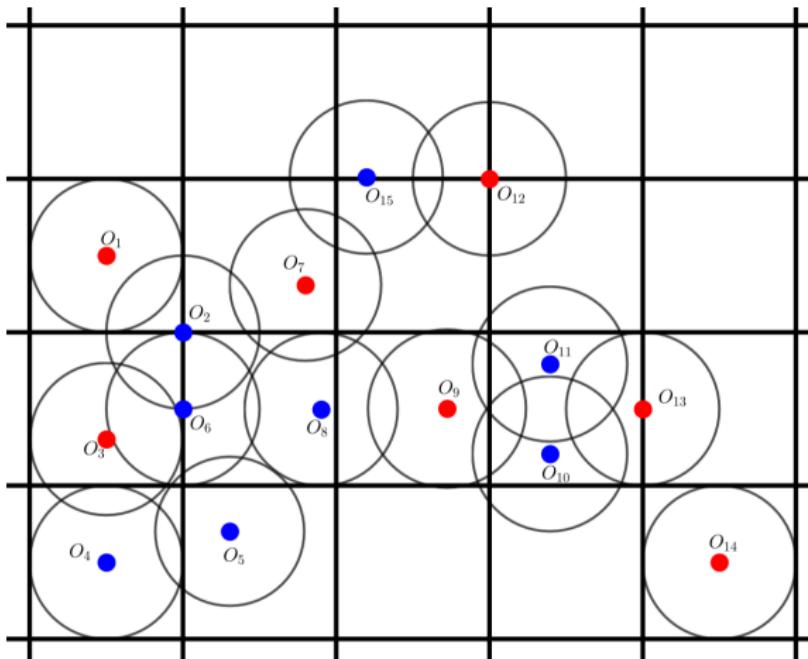
En nuestro caso, usemos la rejilla de tamaño 2, con la consideración de que cada centro de disco se encuentra en una única celda. Así, para S_1 tenemos que



Para S_2 tenemos que



Para S_3 tenemos que



Lema 1.

Cada disco unitario en \mathbb{R}^2 está contenido en una celda de, al menos, una de las cuadrículas desplazadas G_1, \dots, G_k . En consecuencia para un conjunto S de discos unitarios la celda de una de las rejillas contiene conjuntamente

$$\frac{|S|}{4}$$

discos.

Dem. La primera parte se da a partir de su construcción. La segunda parte se sigue del principio de casillas. □

Gracias a este lema sabemos que una cuadrícula contiene al menos una fracción constante ($o(1)$) de la aproximación de OPT.

Lema 2.

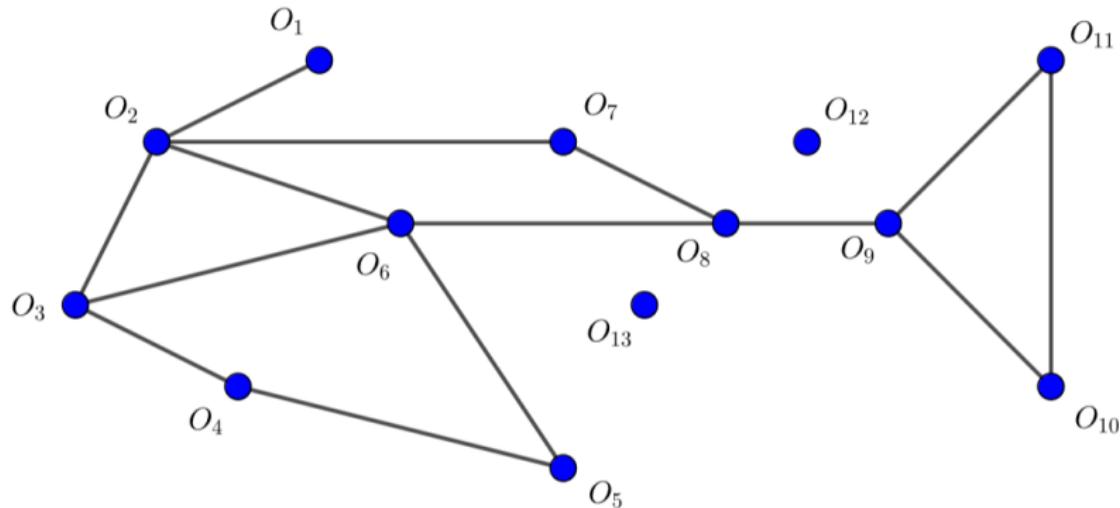
Sean S_1, \dots, S_k conjuntos independientes en el conjunto D de discos unitarios calculados para G_1, \dots, G_k respectivamente. El más grande de S_1, \dots, S_k es una 12-aproximación de MIS, para D .

Dem. Observar el caso en dónde $|S_i| = 1$ para rejillas 4×4 (este caso no es válido, pero da la idea de la prueba).

Obs. Una x -aproximación es una aproximación en $\frac{1}{x}$ que se puede interpretar como $\text{OPT} - x$, pero nunca como $\text{OPT} + x$.

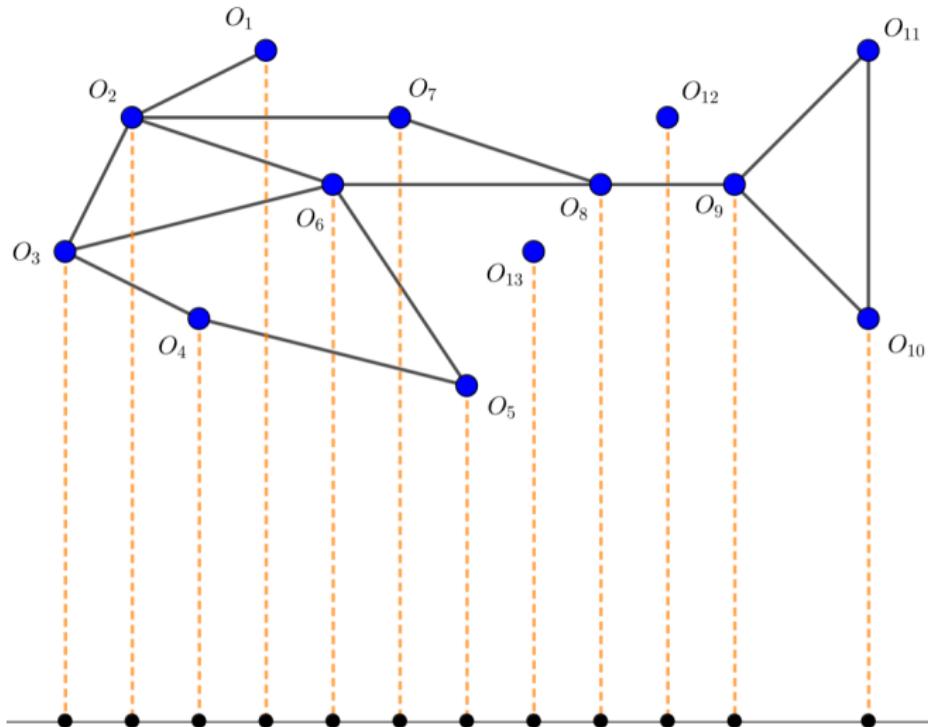
Resultado de MIX

S ha sido el resultado de usar MIX, entonces nos retorna la siguiente gráfica



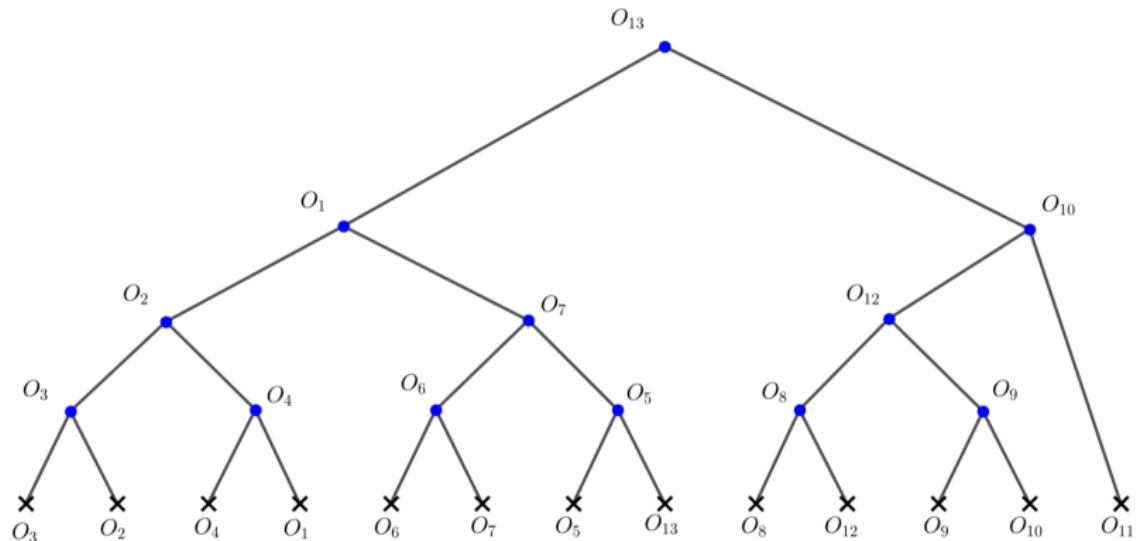
Proyecciones en ℓ -eje

Realicemos las proyecciones correspondientes en X



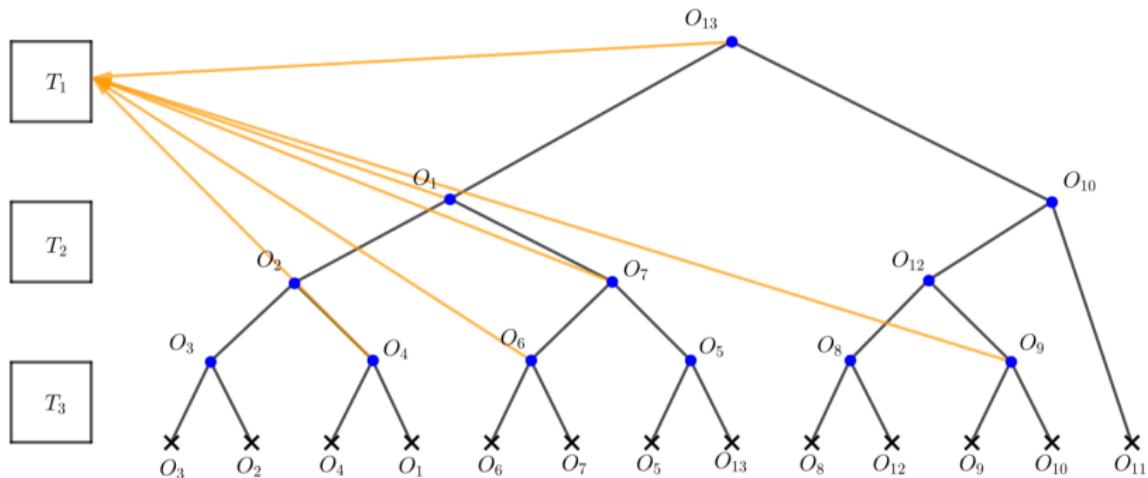
Construcción del árbol de rangos

A continuación realizamos la construcción de un árbol de rangos



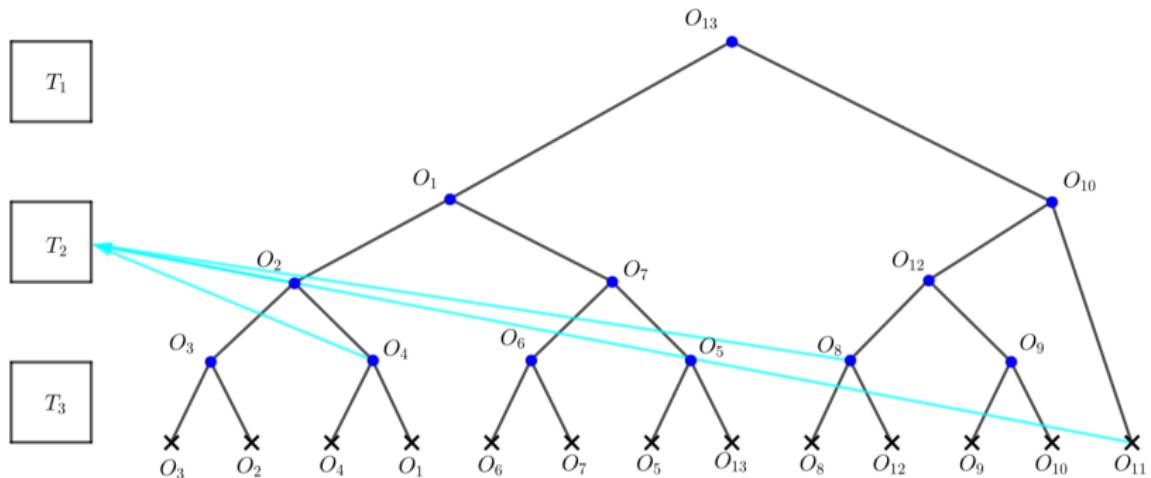
...

Sea T_1 el árbol asociado a los discos en S_1



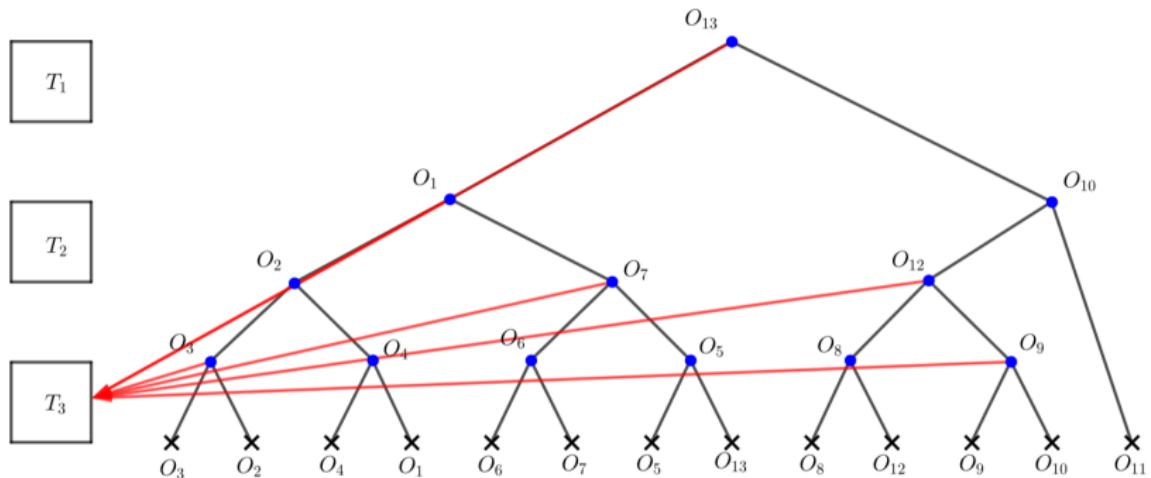
8

Sea T_2 el árbol asociado a los discos en S_2



8

Sea T_3 el árbol asociado a los discos en S_3



Comentarios sobre generalizaciones ...

¡Gracias!