Tarea 1

M-Tree

Profesores: Benjamín Bustos y Gonzalo Navarro
Auxiliares: Sergio Rojas y Diego Salas

El M-Tree[1][2] (wikipedia, paper original) es un método de acceso métrico dinámico, que consiste en un árbol balanceado. Al igual que muchos métodos de acceso espacial, el rendimiento del M-Tree depende en gran medida del grado de superposición entre las regiones espaciales representadas por los nodos del árbol, y la minimización de la superposición es clave para muchas de las características de diseño del M-Tree y las estructuras relacionadas. La idea de este árbol es poder almacenar en disco cualquier set de objetos que se relacionen mediante una métrica que cumpla con la desigualdad triangular. Para efectos de esta tarea, nuestros objetos serán puntos en un plano bidimensional, y nuestra métrica será la distancia euclidiana.

1. Composición de un M-Tree

Un M-Tree es un árbol que está compuesto de nodos que contiene entradas (p, c_r, a), donde p es un punto, c_r es el radio cobertor (covering radius) de este subárbol (la máxima distancia que hay entre p y cualquier punto del subárbol relacionado a su entrada) y a una dirección en disco a la página de su hijo identificado por la entrada de su nodo interno. Si el nodo corresponde a una hoja, por simplicidad asumiremos c_r y a nulos. Para contexto de esta tarea utilizaremos las coordenadas de p y c_r como reales de doble precisión (double).

De acuerdo a esto, diremos que cada nodo tendrá como capacidad B entradas en disco, es decir, que el tamaño de un nodo es a lo más $B \cdot sizeof(entry)$.

2. Búsqueda

La búsqueda tiene como input el M-Tree ($\mathcal{T}$) donde se buscará una query $Q = (q, r)$, donde q es un punto y r es el radio de búsqueda. Es decir, (q, r) definen una bola. El objetivo es encontrar todos los puntos de $\mathcal{T}$ que residen dentro de esta (ver Figura 1).

Para realizar la búsqueda, se verifica desde la raíz cada nodo hijo de esta:

- Si el nodo es una hoja, se verifica para cada entrada si p cumple con $dist(p, q) \leq r$. Si es así, se agrega p a la respuesta.
- Si el nodo es interno (ver Figura 2), se verifica para cada entrada (p, c_r, a) si $dist(p, q) \leq c_r + r$. Si es así, se busca en su hijo a posibles respuestas. Si no se cumple esa desigualdad, se descarta.

[1] https://en.wikipedia.org/wiki/M-tree
[2] https://cs-web.bu.edu/faculty/gkollios/ada17/LectNotes/Mtree.PDF

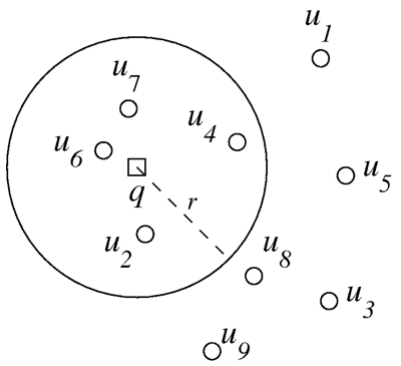

Figura 1: Búsqueda con $Q = (q, r)$, en este caso $\{u_2, u_4, u_6, u_7\}$ son la respuesta a la consulta

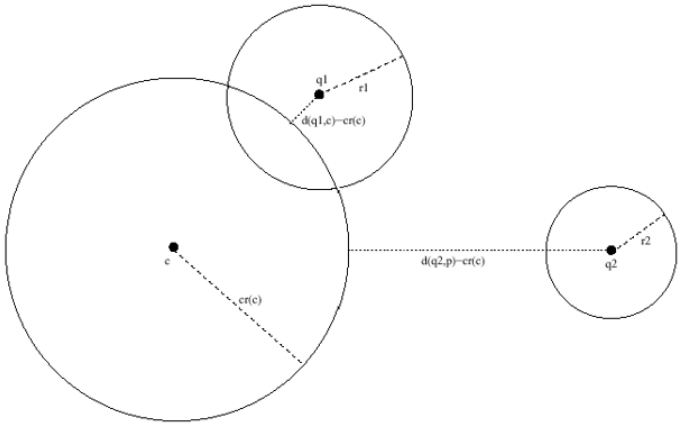

Figura 2: Búsqueda en una entrada de un nodo interno del M-Tree . Podemos notar que la consulta $Q_1 = (q_1, r_1)$ cumple con la desigualdad pedida, por lo que se proseguirá a buscar respuestas en el hijo correspondiente a esa entrada. Por otro lado, la consulta $Q_2 = (q_2, r_2)$ no cumple con la desigualdad, por lo que se descarta la entrada en esa búsqueda.

3. Métodos de construcción

Como se señaló anteriormente, el rendimiento de estas búsquedas dependen en gran medida de la superposición entre las regiones espaciales representadas por los nodos del árbol. A continuación señalaremos los dos métodos de construcción que se utilizarán para esta tarea que buscan minimizar el solapamiento. Estos se basan en algoritmos bulk-loading, es decir, insertan datos en grandes cantidades a la vez en el árbol. Para simplificar las implementaciones, diremos que cada nodo tendrá que tener una capacidad mínima de $b = 0.5 \cdot B$ y capacidad máxima de B.

3.1. Método Ciaccia-Patella (CP)

El algoritmo[3] (paper) que se propone en este método realiza un clustering de n puntos $P = \{p_1, ..., p_n\}$, obteniendo un M-tree balanceado $\mathcal{T}$ que cumple con tener cada nodo dentro de las capacidades permitidas. Se realizan los siguientes pasos:

Algoritmo CP:

Input: Un set de puntos P

1. Si $|P| \leq B$, se crea un árbol $\mathcal{T}$, se insertan todos los puntos a $\mathcal{T}$ y se retorna $\mathcal{T}$.
2. De manera aleatoria se eligen $k = min(B, \frac{n}{B})$ puntos de P, que los llamaremos samples $p_{f_1}, \ldots, p_{f_k}$. Se insertan en un conjunto $\mathcal{F}$ de samples.
3. Se le asigna a cada punto en P su sample más cercano. Con eso se puede construir k conjuntos $\mathcal{F}_1, \ldots, \mathcal{F}_k$.
4. Etapa de redistribución: Si algún $\mathcal{F}_j$ es tal que $|\mathcal{F}_j| < b$:
   - 4.1 Quitamos p_{f_j} de $\mathcal{F}$
   - 4.2 Por cada $p \in \mathcal{F}_j$, le buscamos el sample p_{f_l} más cercano de $\mathcal{F}$ y lo añadimos a su conjunto $\mathcal{F}_l$.
5. Si $|\mathcal{F}| = 1$, volver al paso 2.
6. Se realiza recursivamente el algoritmo CP en cada $\mathcal{F}_j$, obteniendo el árbol $\mathcal{T}_j$
7. Si la raíz del árbol es de un tamaño menor a b, se quita esa raíz, se elimina p_{f_j} de $\mathcal{F}$ y se trabaja con sus subárboles como nuevos $\mathcal{T}_j, \ldots, \mathcal{T}_{j+p-1}$, se añaden los puntos pertinentes a $\mathcal{F}$.
8. Etapa de balanceamiento: Se define h como la altura mínima de los árboles $\mathcal{T}_j$. Se define $\mathcal{T}'$ inicialmente como un conjunto vacío.
9. Por cada $\mathcal{T}_j$, si su altura es igual a h, se añade a $\mathcal{T}'$. Si no se cumple:
   - 9.1 Se borra el punto pertinente en $\mathcal{F}$.
   - 9.2 Se hace una búsqueda exhaustiva en $\mathcal{T}_j$ de todos los subárboles $\mathcal{T}'_1, \ldots, \mathcal{T}'_p$ de altura igual a h. Se insertan estos árboles a $\mathcal{T}'$
   - 9.3 Se insertan los puntos raíz de $\mathcal{T}'_1, \ldots, \mathcal{T}'_p$, $p'_{f_1}, \ldots, p'_{f_p}$ en $\mathcal{F}$
10. Se define $\mathcal{T}_{sup}$ como el resultado de la llamada al algoritmo CP aplicado a $\mathcal{F}$.
11. Se une cada $\mathcal{T}_j \in \mathcal{T}'$ a su hoja en $\mathcal{T}_{sup}$ correspondiente al punto $p_{f_j} \in \mathcal{F}$, obteniendo un nuevo árbol $\mathcal{T}$.
12. Se setean los radios cobertores resultantes para cada entrada en este árbol.
13. Se retorna $\mathcal{T}$.

[3] https://www.researchgate.net/profile/Paolo-Ciaccia/publication/2291756_Bulk_Loading_the_M-tree/links/09e4150a2406d17c3c000000/BulkLoadingtheMtree.pdf

3.2. Método Sexton-Swinbank (SS)

El algoritmo[4] (paper) que se propone utiliza otro tipo de clustering. Primero se arma distintos grupos de clusters, los cuales son utilizados para armar el M-tree resultante. Primero definiremos los siguientes conceptos:

a. El medoide de un cluster es un punto g en el cluster tal que no existe otro punto p en el cluster que, al ser nominado como medoide, genere un radio menor al entregado por g

b. Un cluster puede tener muchos medoides, se define alguna estrategia para definir el medoide primario.

c. La distancia entre dos clusters es la distancia entre sus medoides primarios.

d. Dado un cluster c, su vecino más cercano es otro cluster c' tal que no hay otro cluster que su distancia a c sea menor a la distancia entre c y c' (se puede tener múltiples vecinos más cercanos).

e. El par más cercano es un par de clusters c_1, c_2 tal que $dist(c_1, c_2) \leq dist(c_i, c_j)$ para cualquier otro par c_i, c_j.

Para explicar el algoritmo bulk-loader se definen las siguientes funciones:

- Cluster: retorna un conjunto de clusters de tamaño entre b y B.

Input: Un set de puntos $\mathcal{C}_{in}$ de tamaño mínimo b

1. Se define $\mathcal{C}_{out} = \{\}$ y $\mathcal{C} = \{\}$
2. Por cada punto $p \in \mathcal{C}_{in}$ se añade $\{p\}$ a $\mathcal{C}$.
3. Mientras $|\mathcal{C}| > 1$:
   3.1 Sea c_1, c_2 los pares más cercanos de clusters en $\mathcal{C}$ tal que $|c_1| \geq |c_2|$.
   3.2 Si $|c_1 \cup c_2| \leq B$, se remueve c_1 y c_2 de $\mathcal{C}$ y se añade $c_1 \cup c_2$ a $\mathcal{C}$.
   3.2 Si no, se remueve c_1 de $\mathcal{C}$ y se añade c_1 a $\mathcal{C}_{out}$.
4. Sea c el último elemento de $\mathcal{C}$
5. Si $|\mathcal{C}_{out}| > 0$:
   5.1 definimos c' como el vecino más cercano a c en $\mathcal{C}_{out}$. Removemos c' de $\mathcal{C}_{out}$
   5.2 Si no, se define $c' = \{\}$.
6. Si $|c \cup c'| \leq B$:
   6.1 Añadimos $c \cup c'$ a $\mathcal{C}_{out}$.
   6.2 Si no, dividimos $c \cup c'$ en c_1 y c_2 usando MinMax split policy. Se añaden c_1 y c_2 a $\mathcal{C}_{out}$.
7. Se retorna $\mathcal{C}_{out}$

El MinMax split policy corresponde a lo siguiente: Se considera todos los posibles pares de puntos, y alternadamente se van agregando el punto más cercano a alguno de estos centros (esto garantiza que la división sea balanceada) y se calcula el radio cobertor máximo entre estos dos grupos resultantes. Esto se prueba para todo par de puntos y se elige el par que tenga el mínimo radio cobertor máximo.

[4] https://www.researchgate.net/profile/Alan-Sexton/publication/220862641_Bulk_Loading_the_M-Tree_to_Enhance_Query_Performance/links/5421bf2b0cf26120b79fc4cc/BulkLoadingtheMTreetoEnhance-QueryPerformance.pdf

- OutputHoja: Retorna una tupla (g, r, a) donde g es el medoide primario de $\mathcal{C}_{in}$, r es llamado el radio cobertor y a la dirección del hijo respectivo.

Input: $\mathcal{C}_{in}$

1. Sea g el medoide primario de $\mathcal{C}_{in}$. Sea $r = 0$. Sea $\mathcal{C} = \{\}$ (el que corresponderá al nodo hoja).
2. Por cada $p \in \mathcal{C}_{in}$: Añadimos $(p, null, null)$ a $\mathcal{C}$. Seteamos $r = max(r, dist(g, p))$
3. Guardamos el puntero a $\mathcal{C}$ como a
4. Retornamos (g, r, a)

- OutputInterno: Retorna (G, R, A) donde G es el medoide primario del conjunto de puntos $\mathcal{C}_{in} = \{g | \exists (g, r, a) \in \mathcal{C}_{mra}\}$, R el radio cobertor, y A la dirección del hijo respectivo.

input: $\mathcal{C}_{mra}$, un conjunto de tuplas (g, r, a) retornadas por OutputHoja

1. Sea $\mathcal{C}_{in} = \{g | \exists (g, r, a) \in \mathcal{C}_{mra}\}$. G el medoide primario de $\mathcal{C}_{in}$. Sea $R = 0$. Sea $\mathcal{C} = \{\}$ (el que corresponderá a un nodo interno).
2. Por cada $(g, r, a) \in \mathcal{C}_{mra}$: Añadir (g, r, a) a $\mathcal{C}$. Se setea $R = max(R, dist(G, g) + r)$
3. Guardamos el puntero a $\mathcal{C}$ como A.
4. Retornamos (G, R, A)

- Algoritmo SS: retorna la raíz del M-tree construído.

Input: $\mathcal{C}_{in}$, un conjunto de puntos

1. Si $|\mathcal{C}_{in}| \leq B$: Se define $(g, r, a) =$ OutputHoja($\mathcal{C}_{in}$) y se retorna a.
2. Sea $\mathcal{C}_{out} =$ Cluster($\mathcal{C}_{in}$). Sea $\mathcal{C} = \{\}$.
3. Por cada $c \in \mathcal{C}_{out}$: Se añade OutputHoja(c) a $\mathcal{C}$
4. Mientras $|\mathcal{C}| > B$:
   - 4.1 Sea $\mathcal{C}_{in} = \{g | (g, r, a) \in \mathcal{C}\}$. Sea $\mathcal{C}_{out} =$ Cluster($\mathcal{C}_{in}$). Sea $\mathcal{C}_{mra} = \{\}$.
   - 4.2 Por cada $c \in \mathcal{C}_{out}$: Sea $s = \{(g, r, a) | (g, r, a) \in \mathcal{C} \wedge g \in c\}$, se añade s a $\mathcal{C}_{mra}$
   - 4.3 Sea $\mathcal{C} = \{\}$.
   - 4.4 Por cada $s \in \mathcal{C}_{mra}$: Añadir OutputInterno(s) a $\mathcal{C}$
5. Sea $(g, r, a) =$ OutputInterno($\mathcal{C}$)
6. Se retorna a

4. Objetivos

Para esta tarea, se deberá implementar:

- Una función por cada método de construcción, que permita a partir de un conjunto de puntos, construir el árbol.
- El método de búsqueda dentro del árbol.
- Evaluar el costo del método de búsqueda con los distintos métodos de construcción.

De esta forma, la idea es hacer una serie de búsquedas en el M-tree y ver cuál método de construcción funciona mejor.

5. Experimentación

Por simplicidad en la realización de la tarea, haremos todos los procesos en memoria principal y simularemos los accesos a disco (acceder a un nodo del árbol equivale a una lectura de bloque en disco). Se pide comparar la cantidad de accesos (simulados) a bloques de disco de la búsqueda. Definiremos el tamaño de un bloque de disco como 4096 Bytes.

Tanto para el set de puntos P, como para los puntos del set de consultas Q, para las coordenadas se debe utilizar valores aleatorios reales de doble precisión (double) uniformemente distribuidos en el rango [0, 1]. El radio cobertor también corresponde a un valor real de doble precisión. El tamaño del atributo a de las entradas corresponde al tamaño de un puntero en memoria. Con todo esto, más el tamaño del bloque de disco mencionado, se debe estimar los valores de b y B. El radio r de las consultas debe ser de 0.02 (lo que retorna aproximadamente un 0.12 % de los puntos del conjunto).

Evalúe el resultado para cada $n \in \{2^{10}, 2^{11}, ..., 2^{25}\}$, siendo este valor, la cantidad de puntos en P. Ocupe el mismo conjunto P y Q para evaluar los dos métodos de construcción del M-tree. Realice 100 consultas por cada valor de n, y reporte el intervalo de confianza de sus resultados. A partir de lo anterior, construya un gráfico que compare los resultados entre sí.

6. Entregables

Se deberá entregar el código y un informe donde se explique el experimento en estudio. Con esto se obtendrá una nota de código ($NCod$) y nota de informe ($NInf$). La nota de la tarea será:

$$NT_1 = 0.5 \cdot NCod + 0.5 \cdot NInf$$

Fecha de entrega: 9 de mayo a las 23:59.

6.1. Código

La entrega de código debe ser hecha en C, C++ o Java. Tiene que contener:

- (**0.5 pts**) README: Archivo con las instrucciones para ejecutar el código, debe ser lo suficientemente explicativo para que cualquier persona solo leyendo el README pueda ejecutar la totalidad de su código.
- (**0.5 pts**) Experimento: Creación de los puntos y consultas a realizarse en los experimentos, esto basándose en la cantidad pedida en este mismo informe.
- (**1.5 pts**) Método CP: Implementación de este método de construcción.
- (**1.5 pts**) Método SS: Implementación de este método de construcción.
- (**1.5 pts**) Obtención de resultados: La forma en el que se obtienen los resultados pedidos es correcta, es decir, se cuentan todos los accesos y los tiempos de ejecución adecuados.
- (**0.5 pts**) Main: Un archivo o parte del código (función main) que permita ejecutar los distintos métodos de construcción y experimentos.

6.2. Informe

El informe debe ser claro y conciso. Se recomienda hacerlo en LaTeX. Debe contener:

- (**0.8 pts**) Introducción: Presentación del tema en estudio, resumir lo que se dirá en el informe y presentar una hipótesis.
- (**0.8 pts**) Desarrollo: Presentación de algoritmos, estructuras de datos, en lo que se diferencian los algoritmos/estructuras y cómo funcionan y por qué. Recordar que los métodos ya son conocidos por el equipo docente, lo que importa son sus propias implementaciones.
- (**2.4 pts**) Especificación de los datos que se utilizaron para los experimentos, la cantidad de veces que se realizaron los tests, con qué inputs, que tamaño, etc. Se debe mencionar en que sistema operativo y los tamaños de sus cachés y RAM con los que se ejecutaron los experimentos. Se deben mostrar gráficos/tablas y mencionar solo lo que se puede observar de estos, se deben mostrar los valores y parámetros que se están usando.
- (**1.2 pts**) Comentar y concluir sus resultados. Se hacen las inferencias de sus resultados.
- (**0.8 pts**) Recapitulación de lo que se hizo, se concluye lo que se puede decir con respecto a sus resultados. También ven si su hipótesis se cumplió o no y analizan la razón. Por último, se menciona qué se podría mejorar en su desarrollo en una versión futura, qué falta en su documento, qué no se ha resuelto y cómo se podrían extender.

Todo lo mencionado debe estar en sus informes en las secciones en las que se señalan, la falta de algún aspecto o la presencia de algún aspecto en una sección equivocada hará que no se tenga la totalidad del puntaje.