BEEBOTS, ESTRUCTURA DE DATOS PARA PREDECIR COLISIONES ENTRE DOS O MÁS OBEJETOS

Santiago Hincapié Murillo
Universidad EAFIT
Medellín, Colombia
shincapiem@eafit.edu.co

Santiago Valencia Arango
Universidad EAFIT
Medellín, Colombia
svalenciaa@eafit.edu.co

Andrés Almánzar Restrepo
Universidad EAFIT
Medellín, Colombia
aalmanzarr@eafit.edu.co

Mauricio Toro
Universidad EAFIT
Medellín, Colombia
mtorobe@eafit.edu.co

RESUMEN
Se tiene como objetivo poder detectar la colisión entre diferentes objetos. Para hacerlo tomaremos como objetos abejas robóticas. La razón de esta elección se debe a una simulación, siendo las abejas robóticas una posible solución a la colisión entre dos o más objetos.

La magnitud que recae sobre este problema es de vital importancia porque las abejas son uno de los principales actores en la polinización.

Es de suma importancia encontrar una solución puesto que, con el paso del tiempo, crece la demanda de alimentos, pero como respuesta al aumento de la población se tiene mayor uso de pesticidas, incremento de la contaminación, cambio climático, etc. Las abejas se ven afectadas y con ello la producción de alimento también.

A lo largo de este informe, se expondrán distintos problemas algorítmicos, de baja complejidad y alta velocidad. El objetivo de estos problemas algorítmicos será su aplicación para detectar colisión entre objetos simulando situaciones de la vida real.

Palabras clave
Organización de sistemas informáticos → Estructuras de Datos → Algoritmos → Robótica → Notación O → Complejidad

Palabras clave de la clasificación de la ACM
Theory of computation → Computer systems organization → Design and analysis of algorithms → Graphs → Shortest paths →Robotic

1. INTRODUCCIÓN

Un factor imprescindible en la polinización es la presencia de las abejas. Desafortunadamente el humano y su constante cambio en la naturaleza ha provocado que las abejas estén en peligro de extinción.

Organizaciones como FAO advierten que hay 100 especies de cultivos que proporcionan el 90% de alimentos a nivel mundial y de ese centenar, unas 71 especies son polinizadas por abejas.

El año 2012 se registró como el peor año de la historia para a la apicultura. Esto según estadísticas en Estados Unidos.

En Oregón murieron 50,000 abejas a causa de los efectos causados por los pesticidas.

En Europa, la comisión para el control de seguridad alimentaria de la unión europea (EFSA), confirmó una masiva baja en la población de abejas a causa de un fertilizante llamado neonicotinoides.

Estos informes, como muchos otros nos exponen sin lugar a dudad la gran pérdida de agentes polinizadores.

Múltiples grupos, organizaciones e instituciones han dado con frecuencia "soluciones" a este problema. ¿Son estas respuestas efectivas?

2. PROBLEMA

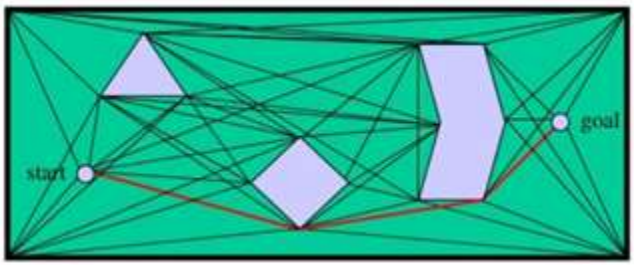
Consideramos útil la implementación de estructuras de datos para la prevención del choque de objetos. Gracias al uso de una estructura de datos podremos consultar la posición y si se está dando alguna colisión entre las abejas robóticas permitiendo así un mayor control de la labor de polinizar cultivos. Ya con el control sobre la colisión de las abejas robóticas, múltiples aplicaciones a futuro serán más fáciles de manejar.

3. TRABAJOS RELACIONADOS

A continuación, se exponen 4 problemas algorítmicos similares.

3.1 Método en espacio cartesiano

Suponiendo que se trabaja en un plano 2D, en el cual el objeto es un punto y los obstáculos son convexos, se elige el camino más corto formado por segmentos que conectan los vértices de los objetos.



Explicación de la gráfica: En el método se definen todos los segmentos que conectan los vértices con "Start", con "Goal" y entre sí. eliminando las conexiones que intersecan los obstáculos. En negro los caminos posibles, en rojo el más corto. Notamos que posee cierto parecido con el algoritmo Dijkstra.

3.2 Colisiones 2D en videojuegos

Uno de los problemas más grande a la hora de la creación de un videojuego, es la colisión. Sin colisiones existirían muchos problemas como: atravesar edificios sin restricción, 2 o más personajes no se chocarían, en videojuegos de shooter las balas no harían daño, en videojuegos de carreras no habría choques, etc.

La manera más fácil de detectar colisiones en videojuegos es simplemente envolver los objetos móviles en simples figuras geométricas, ya sea un cuadrado o un circulo.

Colisión rectangulo-rectangulo:

Para que un rectángulo esté colisionando con otro deben cumplirse cuatro cosas, definiendo dos rectángulos como r1 y r2 diremos que colisionan si:

Lado derecho de r1 es mayor que lado izquierdo de r2

Lado izquierdo de r1 es menor que lado derecho de r2

Lado superior de r1 es mayor que lado inferior de r2

Lado inferior de r1 es menor que lado superior de r2

Si se cumplen estas cuatro condiciones es que ambos rectángulos están colisionando.

Colisión círculo-círculo:

Si la suma de sus radios es mayor que la distancia entre sus centros entonces existe colisión. Así que básicamente hay que hacer dos cosas: Calcular la distancia entre sus centros (que es la distancia entre dos puntos) y comprobar si es menor que la suma de sus radios.

¿Cuál es más rápido?

Simple. El cuadrado debido a que su operación es más simple que la del círculo, debido a que la del circulo contiene raíces cuadradas. Pero los cuadrados tienen una desventaja frente a los círculos, debido a que los círculos admiten rotaciones. Ya que por mucho que rotes un circulo siempre será un circulo, pero en el momento que rotes un rectángulo, pierde sus propiedades y por consecuente los cálculos simples no funcionan. En conclusión, es mejor utilizar los rectángulos en situaciones que no se vaya a ejecutar una rotación.

3.3 Búsqueda Heurística en el espacio cartesiano

Este método consta en establecer consta en establecer una función de evaluación que se tenga el objeto en estado x (posición/orientación), se da por:

$$f(x) = g(x) + h(x)$$

siendo:

x el estado (posición/orientación del robot.

f la estimación del coste del camino desde el estado inicial pasando por x hasta el estado final.

g es coste del camino desde el estado inicial a x.

h la estimulación del coste del camino desde x al estado final.

Tomado de: Aníbal Ollero Baturone. ROBÓTICA Manipuladores y robots móviles. Marcombo; Edición: 1 (8 de julio de 2005).

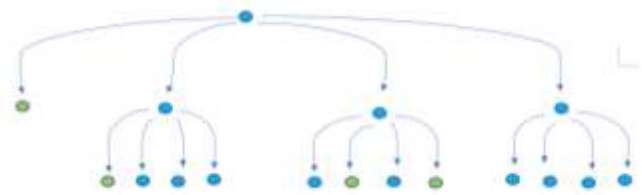
3.4 Teoría de colisiones 2D: QuadTree

QuadTree es un tipo de estructura de datos representado por nodos, en el cual de cada nodo se desprende 4 nodos hijos. Con los 4 nodos hijos se divide el cuadrante donde estén los objetos que podrían colisionar. Dado el caso que en un cuadrante haya demasiados elementos que puedan colisionar, será necesario que ese cuadrante se vuelva a dividir en 4 partes. Esto se hará hasta que haya un número moderado y manejable de objetos.

Resumen hecho de:
http://www.dccia.ua.es/dccia/inf/asignaturas/RG/2002/trabajos/colisiones.pdf

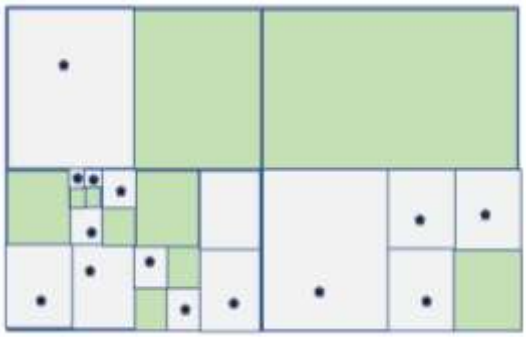
4. BeeBots with Quadtree

Ya conociendo lo que es un QuadTree, a continuación, se muestran dos gráficas para explicar el uso de este para la estructura de datos.



Gráfica 1: Se observa un grafo con nodos. Al principio todos los nodos salen de un nodo padre o raíz.

Nótese que hay nodos de color azul y de color verde. Aquellos de color azul se dividen en otros 4 nodos hijos. Los verdes no se dividen más. Esto porque en aquel nodo ya no hay más objetos. En la siguiente gráfica se muestra otra manera de ver el QuadTree.



Gráfica 2: inicia desde un rectángulo padre. Aquellos cuadrantes que se pueden dividir en más nodos son de color azul y aquellos que no lo pueden hacer son verdes. Cada punto que se observa en los distintos cuadrantes, son los objetos, en nuestro proyecto son las abejas robóticas.

4.1 Operaciones de la estructura de datos

Clase Abeja: En ella se retornan las posiciones en los ejes (x,y), esto para ubicarlas en el plano cartesiano.(Getters y setters). De igual forma con el radio de cada abeja.

Además, se retorna el valor de identificación de las abejas.

Clase AgregarAbejas: Se encuentra un constructor que crea una instancia de Quadtree. Lee del fichero las coordenadas de las abejas. Llama dos métodos de la clase Quadtree para detectar las colisiones entre las mismas.

Clase Mostrar: Es la clase encargada de dibujar las abejas en un Jframe(parte visual). En este se encuentran método Mostrar que hace una instancia de AgregarAbejas. Método Paint el cual dibuja el círculo de cada abeja en el Jframe.

ProgramaPrincipla: Clase principal que corre el programa.

Clase QuadTree: Clase donde se encuentran los mínimos y máximos de las posiciones de las coordenadas(x,y).

Boundry almacena dos puntos diagonales del cuadrado.

Hay un **Arraylist** de Abeja que guarda dichas abejas para que sean comparadas con el resto.

split método encargado de dividir el quadtree cuando el cuadrante alcanza el nivel máximo de abejas.

Insert Método encargado de introducir as abejas al quadtree.

Colision Se encarga de recibir dos abejas, recuperar su posiciones y posterior sacar su distancia entre las mismas por medio del teorema de Pitágoras. Obtiene sus radios y comprara si el radio es o no mayor a la distancia máxima. De ser el caso verdadero estarán colisionando.

Chocan Método que recorre el arraylist donde se almacenan las abejas, compara cada una de ellas con el resto.

ComprobarColisiones Método que recorre el quadtree nodo por nodo y así recuperar las abejas y guardarlas en un Arraylist auxiliar para posterior compararlas.

4.2 Criterios de diseño de la estructura de datos

La elección del QuadTree para el desarrollo del proyecto radica en diversos factores:

- Representación de imágenes gracias a la estructura que posee.

- Detección eficiente de la colisión entre objetos en un campo 2D(dos dimensiones)

-Al compararlo con otras estructuras de datos, posee mejor organización del espacio de los objetos. (esto debido a su constante división en cuadrantes).

- Se encarga de descomponer de manera recursiva el espacio.

5.1 Operaciones de la estructura de datos

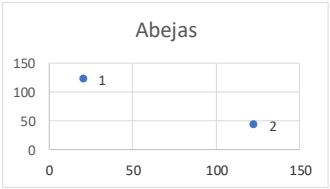
Explicaciones de las operaciones.

Abeja: tiene los valores en los que están las abejas y un valor para identificar la abeja.

	X	Y	VAL
Abeja1	21	123	1
Abeja2	231	43	2
Abeja3	...	...	...

Gráfica 3: Gráfica de abeja.

	X	Y	VAL
Abeja1	21	123	1
Abeja2	231	43	2
Abeja3	...	...	...



Gráfica 4: Gráfica de Agregar abeja.

Insertar: crea una abeja y la agrega a una lista de abejas del quadtree.

Colisiona: mira que abejas chocan y con cuales chocha.

Mostrar: dibuja las abejas en el plano.

Gráfica 4: cantidad de abejas en el mapa de Bello.

Principal: llama a agregar abejas y luego a mostrar

Quadtree: tiene los métodos insertar, dividir, distancia entre abejas, colisionan (boleano).

5.3 Análisis de la Complejidad

Método	Complejidad
insert	O(1)
colision	O(1)
chocan	O(n^2)
agregarAbejas	O(n)
ComprobarColisiones	Log(n)

Tabla 5: Tabla para reportar la complejidad

5.4 Tiempos de Ejecución

Tiempo	N = 10	N = 100	N = 1000	N= 10000	N = 100000	N = 200000
Insertar	3ms	10ms	47ms	130ms	960ms	1573ms
Choque	0Ms	3ms	45ms	250ms	3300 ms	11253ms
Comprobar colisiones	0ms	24ms	41ms	300ms	1500 ms	6325 ms

Tabla 6: Tiempos de ejecución de las operaciones de la estructura de datos con diferentes conjuntos de datos

5.5 Memoria

Memoria que consume el programa para los conjuntos de datos

Memoria	10	100	1000	10000	100000	200000
Insertar	1Mb	1Mb	2Mb	8Mb	16Mb	23Mb
Choque	1Mb	1Mb	2Mb	8Mb	16Mb	20Mb
Comprobar colisiones	1Mb	1Mb	2Mb	9Mb	20Mb	30Mb

Tabla 7: Consumo de memoria de la estructura de datos con diferentes conjuntos de datos

6. CONCLUSIONES

Consideramos que el correcto uso y manejo de colisiones de objetos como lo pueden ser abejas robóticas es esencial para poder llegar a una solución contra distintas problemáticas. Una de ellas puede ser la falta de agentes para la polinización.

Una forma de manejar dichas colisiones es el Quadtree como se presentó en este proyecto.

A diferencia de los pasados trabajos entregados, esta entrega final es más sólida ya que permite leer posiciones de grandes cantidades de abejas, dibujarlas en un mapa (en este caso el mapa de Bello), calcular cual colisiona con cual. Junto a ello pruebas de tiempo de ejecución de los algoritmos, complejidades y cantidad de memoria que se usa para hacer el cálculo.

El mayor reto en este proyecto fue el desarrollo de la parte funcional. Entre ello detectar las colisiones entre los objetos.

6.1 Trabajos futuros

A futuro nuestro grupo de trabajo considera que sería bueno poder modelar el proyecto no solamente como 2D sino 3D. Así se llegaría a un modelo más cercano a la vida real. Visualizando de esta manera no solo la ubicación en X y Y de los objetos, sino logrando mediante 3 dimensiones, la ubicación más cercana de las abejas (Alto Y, ancho Z y largo X).

De ser posible mejorar la implementación del programa. En C++ para así mejorar su velocidad y disminuir lo que ocupa en memoria.

AGRADECIMIENTOS

Nosotros agradecemos por su ayuda con ideas creativas a Cristhian Darío Ceballos Rodríguez, estudiante de segundo Semestre de Ingeniería de Sistemas en EAFIT por sus ideas que ayudaron a ver de otra manera el proyecto para así dar una solución.

REFERENCIAS

1. ¿Qué pasaría si desaparecen las abejas?

https://www.sostenibilidad.com/medio-ambiente/que-pasaria-si-desaparecen-las-abejas/

2. Científicos polacos crean la primera abeja robótica que poliniza como una real. http://www.sinembargo.mx/28-11-2016/3120015

3. Abejas robóticas polinizadoras: ¿realmente son una buena idea?

https://www.ecologiaverde.com/abejas-roboticas-polinizadoras-realmente-son-una-buena-idea-591.html

4. Teoría de colisiones 2D: Conceptos básicos

https://www.genbetadev.com/programacion-de-videojuegos/teoria-de-colisiones-2d-conceptos-basicos

5. Colisiones 2D en los videojuegos

http://www.dccia.ua.es/dccia/inf/asignaturas/RG/2002/trabajos/colisiones.pdf

6. Aníbal Ollero Baturone. ROBÓTICA Manipuladores y robots móviles. Marcombo; Edición: 1 (8 de julio de 2005).