IMPLEMENTACIÓN DE QUADTREES PARA LA DETECCIÓN DE POSIBLES COLISIONES.

Agustín Rico Piedrahita
Universidad EAFIT
Colombia
aricop@eafit.edu.co

Santiago Cano Franco
Universidad EAFIT
Colombia
scanof@eafit.edu.co

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

RESUMEN
El objetivo de este proyecto es desarrollar una estructura de datos que, de manera eficaz, sea capaz de identificar y resolver posibles colisiones entre grandes cantidades de objetos en un plano bidimensional. La importancia del desarrollo de esta estructura de datos gira en torno a la posible futura implementación de abejas robóticas capaces de realizar procesos de polinización en todo tipo de plantas. Para resolver dicha problemática utilizamos una implementación de Quadtree, con unos resultados significativamente

Palabras clave
Detección de colisiones
Quadtree
División Recursiva
Espacio 2D

Palabras clave de la clasificación de la ACM

Mathematics of Computing → Discrethe Mathematics → Graph Theory → Trees

1. INTRODUCCIÓN

Las abejas son responsables de una de las habilidades más únicas e importantes de la naturaleza: la reproducción y polinización de la mayoría de los cultivos esenciales para el hombre. Sin embargo, debido a excesos de fumigación y cambios climáticos, se estima que su población va a desaparecer. Para resolver este problema en el peor de los casos (extinción de las abejas), la industria está forzada a reemplazar el rol de las abejas en la polinización. La solución más factible hasta el momento es la de construir abejas robóticas capaces de llevar polen y moverlo a lo largo de todo el cultivo; pero esto conlleva a otros problemas. Por ejemplo, el de manejar de alguna manera el tráfico o desplazamiento de las abejas en el espacio de tal manera que se minimicen las colisiones entre ellas. Dicho problema será el abordado en este proyecto.

2. PROBLEMA

El problema por resolver consiste en el desarrollo de una estructura de datos la cual pueda ser utilizada para detectar y evitar, de manera rápida y eficiente las colisiones de múltiples objetos en espacios 2D. Para ello, se tendrán en cuenta dos funciones principales que la solución en cuestión debe tener. La primera es la de identificar las colisiones y limitar las evaluaciones a casos realmente alarmantes y cercanos espacialmente entre dos objetos, en este caso, abejas; pues sería poco ineficiente e innecesario calcular una posible solución a una colisión muy poco probable entre dos abejas, por ejemplo, con quinientos metros de distancia entre ellas. La segunda función para tener en cuenta es la eficiencia con la que el algoritmo proporcione soluciones. Puesto que estamos hablando del reemplazo de una población de insectos normalmente numerosa, en la cual puede haber miles de colisiones a solucionar en cuestión de segundos, el algoritmo debe ser especialmente rápido y tener una complejidad temporal lo más baja posible.

3. TRABAJOS RELACIONADOS

3.1 Quadtrees

Un Queadtree es un tipo de estructura de datos jerárquica, cuyo funcionamiento se debe a la descomposición recursiva del espacio. Dicha descomposición se realiza mediante particiones de a cuatro unidades. Es decir, inicialmente, el plano está dividido en cuatro nodos, si en un mismo nodo existen conflictos entre dos objetos, entonces este generará otros cuatros nodos, denominados nodos hijos. De esta manera, se puede determinar en qué nodos puede haber posibles colisiones. Pues si hay un nodo cuyas divisiones en cuatro resultarían siendo menores que alguna de las dimensiones de los objetos, entonces allí es muy posible que ocurra una colisión. La solución que proponen los Quadtrees está más orientada a la rápida identificación de posibles colisiones que a la solución de las mismas.

3.2 Método de Sean Quinlan

Utiliza representaciones envolventes de los objetos basadas en esferas, que están especificadas con datos de posición y de radio. Se rige bajo dos propiedades; la primera es que la unión de todas las esferas cubre completamente el plano, y la segunda es que la esfera de cada nodo contiene las esferas de sus nodos hijos. La generación de nodos es similar al de los Quadtrees, pues se basa en la descomposición recursiva, en este caso, a partir de esferas. Así, cada objeto en el espacio tiene su esfera envolvente, y cuando haya intersecciones entre dichas esferas, es que puede ocurrir una colisión. Para determinar esto, la forma de definir cuando dos objetos van a chocar es cuando la distancia entre las dos esferas envolventes sea menor a la suma de sus radios.

3.3 Programación Lineal

A pesar de que un sistema anticolisiones naturalmente se aborda geométricamente, también puede manifestarse

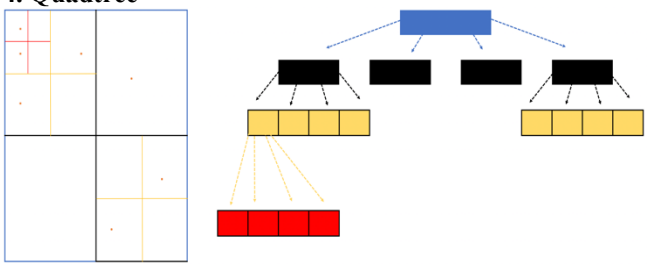
matemáticamente mediante un sistema de inecuaciones. Pues si consideramos cada objeto como el área de intersección de ciertos semiplanos, o inecuaciones, podemos expresar el problema en términos de optimización. El algoritmo determina si hay un punto interior a todos los semiplanos producidos por las inecuaciones lineales de ambos objetos. Además, se puede calcular la distancia de separación y los puntos más cercanos mediante un problema de programación cuadrática.

3.4 Rejilla de Vóxeles

La implementación de una rejilla de Vóxeles consiste en múltiples particiones rectangulares y uniformes del espacio. Así, cada objeto se asocia en las celdas en las que tenga presencia espacial, y de esta manera, dos objetos se pueden chocar si existen en una misma celda. En ese orden de ideas, se deben evaluar y considerar colisiones únicamente en aquellas celdas en las que haya más de un objeto. Parte del éxito del desarrollo del algoritmo consiste en saber determinar el tamaño de las celdas. Asumiendo que todos los objetos (abejas) serán del mismo tamaño y fácilmente representables bajo una figura convexa, puede haber dos complicaciones en cuanto al tamaño de la celda:

1. Si las celdas son muy pequeñas, el número de celdas a actualizar sería alto y por tanto costoso en términos de espacio y tiempo.
2. Si las celdas son muy grandes, entonces muchos objetos podrían habitar a la vez en una de ellas, sin necesariamente tener riesgo de colisiones, produciendo así evaluaciones innecesarias.

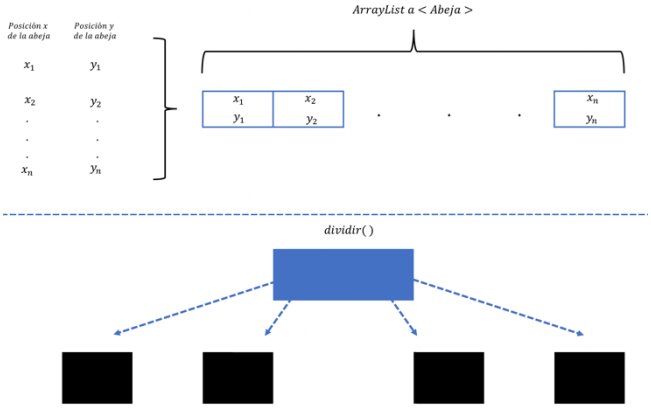
4. Quadtree



Gráfica 1: División recursiva del espacio realizada por un Quadtree.

4.1 Operaciones de la estructura de datos

Gráfica 2: Divisón de un nodo.



Gráfica 3: Creación de un Quadtree con sus respectivos primeros cuatro nodos.

Operación 4: División recursiva del Queadtree hasta alcanzar ciertas condiciones base y calculo del número de colisiones.

4.2 Criterios de diseño de la estructura de datos

Se eligió la estructura Quadtree por su facilidad para subdividirse recursivamente e ir realizando validaciones sobre ella. Al lograr subdividir se puede entender como el problema como de tipo Divide & Conquer logra una mayor eficacia en los tiempos de ejecución. Adicionalmente, puesto que el problema radica más en la búsqueda y localización de elementos almacenados muy uniformemente (que se traduce en el árbol como un buen balanceo de elementos) y no son necesarias acciones como agregar o eliminar elementos, utilizamos ventajosamente la baja complejidad que poseen los árboles correctamente balanceados para la búsqueda.

4.3 Análisis de Complejidad

Método	Complejidad
División de un nodo	O(n)
Creación de un Quadtree	O(2n) = O(n)
Cálculo de numero de posibles colisiones.	O(n*log(4)n)

Tabla 1: Tabla para reportar la complejidad de las operaciones.

4.4 Tiempos de Ejecución

	N = 10	N = 100	N = 1000
División de un nodo	0,01µs	0,06µs	0,12µs
Creación de un Quadtree	0,44µs	0,81µs	4,45µs
Cálculo de numero de posibles colisiones.	0,06µs	0,36µs	6,44µs

Tabla 2: Tiempos de ejecución de las operaciones de la estructura de datos con diferentes conjuntos de datos.

4.4 Análisis de los resultados

En la siguiente tabla se puede observar la notable diferencia, en tiempo, con respecto a la efeciencia de la implementación del Quadtree con respecto a la simple implementación de la comparación de ¨todos con todos¨, también conocido ¨fuerza bruta¨. Vemos que los tiempos, para todas las muestras de datos, son menores en la implementación del Quadtree, y siguiendo la lógica del crecimiento de las funciones que su complejidad obedece, es claro que cada vez la diferencia entre los tiempos, a medida que se aumenta N, será mucho mayor.

	N = 10	N = 100	N = 1000
Quadtree	0,06µs	0,36µs	6,44µs
Fuerza bruta	0,3µs	1,0µs	32,0µs

Tabla 3: Resultados, en tiempo, para el cálculo del número de posibles colisiones obtenidos con la implementación de la estructura de datos vs la implementación con ¨fuerza bruta¨

5. CONCLUSIONES

Para concluir, la correcta implementación de una estructura de datos, en este caso para la detección de posibles colisiones, es crucial en situaciones de la vida real, pues resolviendo un problema meramente computacional se llegan a resultados tangibles y necesarios, como eficiencia en tiempo y recursos, y fiabilidad en los resultados.

Con la solución aquí implementada, se lograron tiempos mucho menores que los que se hubiesen obtenido con el mecanismo de comparar todos los puntos (abejas) con todos y calcular su distancia, comunmente conocido como ¨fuerza bruta¨.

Durante el proceso se tuvo un enfoque diferente para la solución del problema, sin utilizar TreeSet ni String; dicha implementación no está incluida en el reporte puesto que, a pesar de constituir gran parte del enfoque para la solución final, retornaba un margen de error muy grande, de más del 50%. Haciéndola una implementación inútil para la solución del problema mas no para el desarrollo de la solución final.

En el proyecto aquí elaborado, como ingenieros matemáticos en desarrollo, vemos grandes aplicaciones en el manejo de datos y estadísticas. Específicamente, dada una muestra muy grande de datos, se puede utilizar la implementación de un Quadtree que retorne, por ejemplo, la distribución a la que los datos más se acoplan, así como demás datos estadísticos cruciales al momento de cierto análisis. En el desarrollo del proyecto nuestro mayor problema fue que el proceso gradual de nuestra implementación fue muy abrupto: inicialmente se tenía un margen de error del 50%, para pasar a una implementación en la que no hubo mejoras significantes.

REFERENCIAS

1. Juan J. Jiménez Delgado. 2006. *Detección de Colisiones mediante Recubrimientos Simpliciales*. Tesis Doctoral. Universidad de Granada, Granada, España.
2. María del Carmen Ramírez Ortega. 2005. *Estudio e Implementación de un Algoritmo de Detección de Colisiones Basado en Esferas*. Master's thesis. Universidad de las Américas, Puebla, México.