Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Convex Hull

Gift Wrapping vs Incremental

Integrantes: Sebastián Mira Pacheco
Profesor: Nancy Hitschfeld K.
Auxiliar: Sergio P. Salinas
Ayudantes: Andrés Abarca M.
Fecha de realización: 29 de mayo de 2024
Fecha de entrega: 31 de mayo de 2024
Santiago de Chile

Índice de Contenidos

Índice de Figuras

1. Introducción

En este proyecto se deseaba poner a prueba 2 algoritmos que solucionan uno de los problemas más famosos en geometría computacional: la cerradura convexa.

La cerradura convexa es el polígono convexo de menor área y perímetro que rodea un conjunto de puntos (cuenta con otras múltiples condiciones que se consideran en este momento irrelevantes para la introducción). Tiene múltiples aplicaciones como reconocimiento de patrones, procesamiento de imágenes, información geográfica e incluso teoría de juegos.

Durante la investigaciónr realizada se implementaron 2 soluciones algorítmicas: el Gift Wrapping y el algoritmo Incremental, ambas con tiempos de computación muy diferentes y sensibles a distintos casos. Luego se realizaron múltiples test, se analizaron y se concluyeron diferentes cosas en relación al comportamiento de los mismos.

2. Implementaciones y Supuestos en cada Una

Si bien en la sección de cada algoritmo se mencionarán los supuestos tomados en cuenta al momento de seguir con el experimento, se menciona de antemano que se utilizó el tipo de datos "float" para los diversos cálculos.

2.1. Algoritmo de Gift Wrapping

Dentro de los archivos adjuntos para este proyecto se encuentra dentro de la carpeta *src* la implementación del algoritmo de nombre "GiftWrapping.cpp". Incluye múltiples funciones auxiliares, entre ellas para obtener la orientación entre 3 puntos (contrarreloj o sentido horario) y para saber si un punto está en un segmento.

En la firma de la función del algoritmo principal se realizó el primero cambio:
Poligono<float> giftWrapping(Punto<float> puntos[], int n) se decidió entregar la cantidad de puntos n que tiene la nube (con la "nube" se hace referencia al conjunto de *puntos* del cual se obtendrá la cerradura).

La estructura es bastante estándar: primero se recorren todos los puntos encontrando aquel de menor coordenada x (y de menor y en caso de tener igual x). Luego de encontrar ese punto se decide encontrar los siguientes que generen el menor ángulo que recorra los puntos en sentido antihorario, haciendo pruebas de "todos con todos".

De clases sabemos que el algoritmo es sensible a la salida, con un costo general de $\mathcal{O}(nh)$, donde n es la cantidad de puntos en la nube y h es la cantidad de puntos en la cerradura. Es fácil ver que el peor caso entonces es de costo $\mathcal{O}(n^2)$.

2.2. Algoritmo Incremental (upper y lower hull)

Para realizar la estructura de este algoritmo (ordenar, upper-hull y lower-hull) se tuvo que dar una noción de orden a la estructura "Punto" de la tarea 1, con el operador $<$, para luego aplicar una función de sorting parte de la librería estándar de c++.

Esta implementación también cuenta con una función auxiliar para obtener la orientación entre 3 puntos, con la cuál se procedió a construir la cerradura superior e inferior.

Este algoritmo tiene un costo de $\mathcal{O}(n \cdot log(n))$.

A ambos algoritmos se les creó una versión que registra los tiempos de sus partes principales para una determinada nube. En el caso del Gift Wrapping esta versión registra el tiempo de encontrar el punto con x mínimo y el de las operaciones de orientación, y en el caso del algoritmo incremental se registra el tiempo de ordenamiento, el de construcción de upper-hull y el de construcción de lower-hull.

3. Experimentos y Resultados

Las implementaciones fueron probadas en 2 partes de los archivos del proyecto. La primera son aquellos pertenecientes a la carpeta de "test", los cuales fueron hechos con GTest y principalmente fueron utilizados para comprobar el buen funcionamiento de las funciones de las cerraduras. Además se prueba el funcionamiento de las funciones auxiliares.

En la carpeta "experiment" se utilizó el ejecutable del archivo main.cpp en el directorio principal del proyecto para realizar mayores pruebas en un archivo jupyter con las que se obtuvieron los resultados y conclusiones principales. Se muestran los resultados de estos procedimientos.

A continuación se presentan los resultados de las ejecuciones realizadas en la carpeta experiment (se recomienda revisar además el archivo exp.ipynb, que muestra resultados que no se les encontró una visualización). Para estudiar los algoritmos se hicieron 2 secuencias de ejecuciones: para nubes de puntos aleatorias y para nubes de puntos con un porcentaje de ellos en una circunferencia que rodearía al resto de puntos (esto último para controlar la cantidad de puntos en la cerradura).

Los siguientes experimentos fueron llevados a cabo con las siguientes preguntas en mente:

- ¿Cuál es mejor con puntos generados aleatoriamente?
- ¿Cuál es mejor si un porcentaje de puntos pertenece a la cerradura convexa?
- ¿Existen diferencias en cuanto a tiempo de cálculo?

3.1. Nubes de puntos Aleatorias

Para ver cómo se comportan los algoritmos en este caso se realizaron ejecuciones de encontrar la nube de puntos con cada algoritmo en nubes de tamaño $500 \cdot i$, con $i = 1, 2, ..., 2000$. Cada una de estas ejecuciones se realizó 5 veces. Los puntos de la nube obtienen sus componentes como un valor aleatorio entre -60.000 y 60.000. El rango fue elegido al azar, pero con cuidado de que los valores no fueran lo suficientemente grandes como para causar un overflow del tipo de dato en posteriores cálculos.

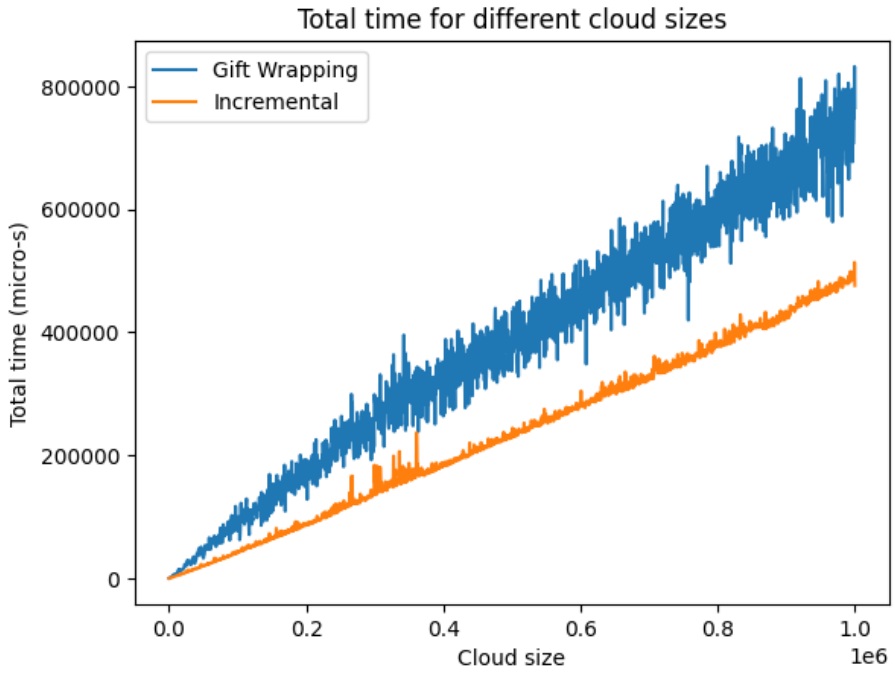

Figura 1: Gráfico de líneas. Muestra el desempeño promedio de las simulaciones con nubes de tamaño de 500 a 500 entre 500 y 1.000.000 puntos, con ambos algoritmos.

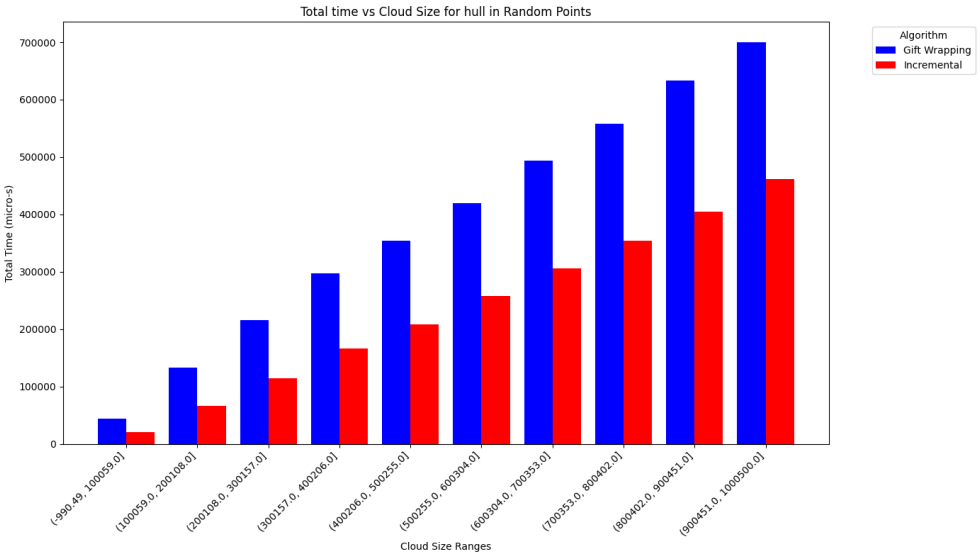

Figura 2: Gráfico de barras. Muestra el desempeño promedio de las simulaciones con nubes de tamaño de 500 a 500 entre 500 y 1.000.000 puntos, con ambos algoritmos. Las barras agrupan los resultados obtenidos en intervalos de tamaño 100.000.

3.2. Nubes de puntos con porcentaje de ellos en la cerradura

En este caso se realizaron ejecuciones de encontrar la nube de puntos con cada algoritmo en nubes de tamaño 10^i, con $i = 1, 2, ..., 6$. Cada una de estas ejecuciones se realizó 10 veces, y cada una de ellas con un porcentaje de puntos igual al $10 \cdot j\,\%$, con $j = 1, 2, 3, ..., 10$.

Los puntos de la nube obtienen sus componentes de dos manera. La primera son los puntos pertenecientes a una circunferencia de radio 60.000. Se generan tantos puntos sobre esta circunferencia como se indica según el total de ellos y el porcentaje indicado. El resto de puntos se encuentra aleatoriamente dentro de ella con componentes en un rango de $(-60.000 \cdot cos(pi/4) - 0.1, 60.000 \cdot cos(pi/4) - 0.1)$.

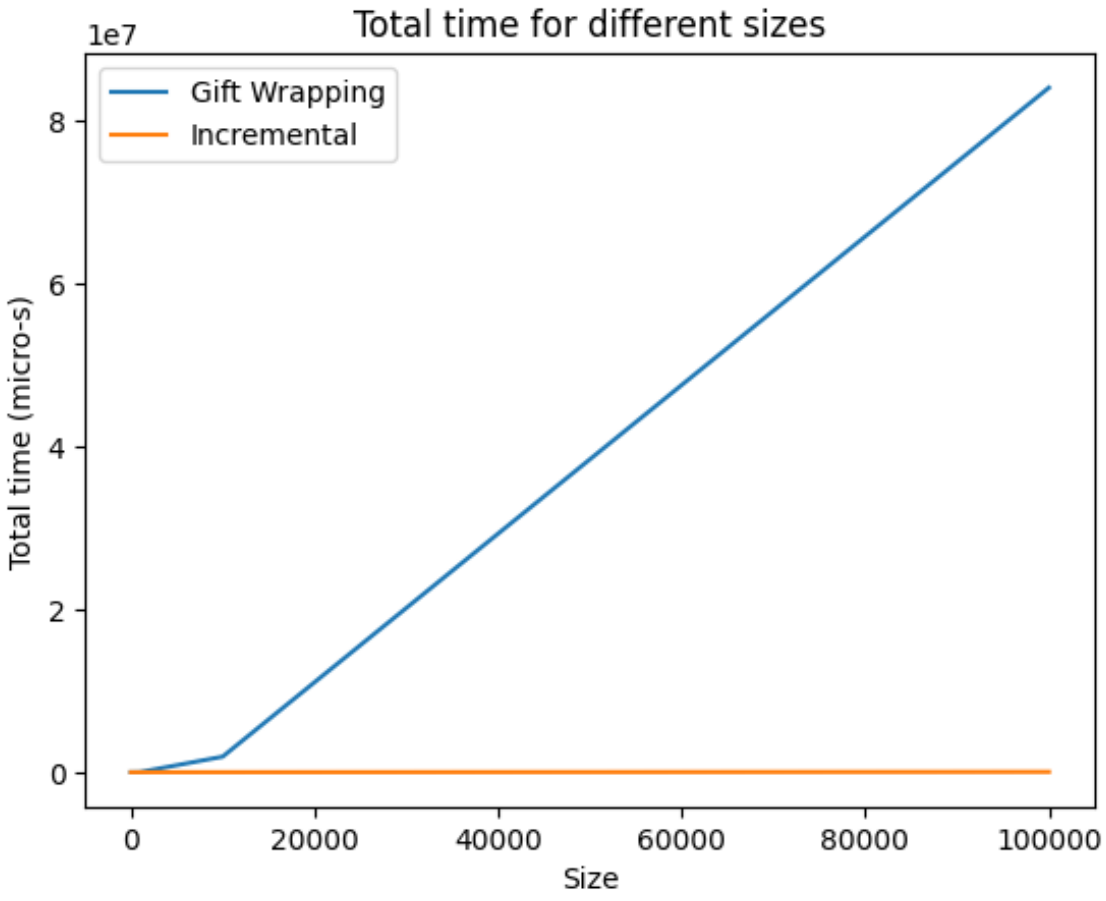

Figura 3: Gráfico de lineas. Muestra el desempeño promedio de las simulaciones con ambos algoritmos en los que se colocó un porcentaje de puntos en la cerradura, sin embargo, no se estudiaron estos por separado.

Se muestra primero el desempeño de los experimentos por porcentaje para el algoritmo

de Gift Wrapping en un gráfico de líneas y en uno de barras. Al de barras se le tuvo que aplicar una escala logarítmica para que se pudieran apreciar los resultados entre un size y otro.

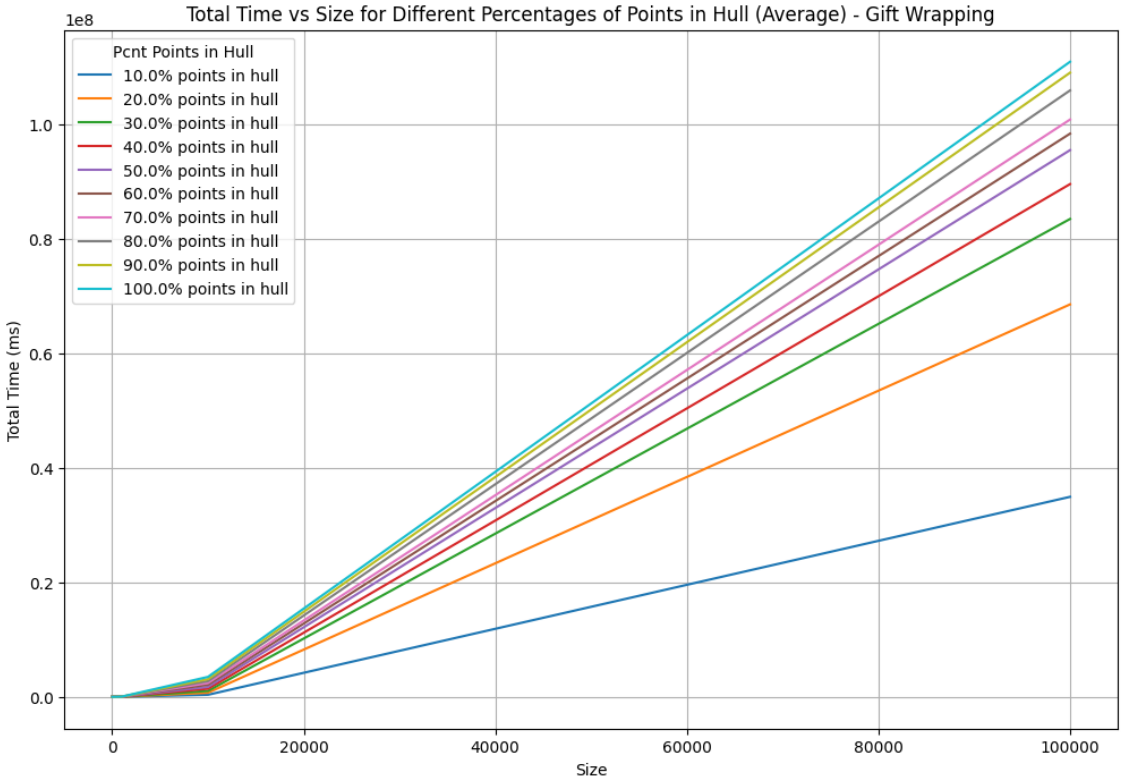

Figura 4: Gráfico de lineas de algoritmo Gift Wrapping. Muestra el desempeño promedio de las simulaciones con un porcentaje de puntos en la cerradura. Cada línea indica un porcentaje utilizado.

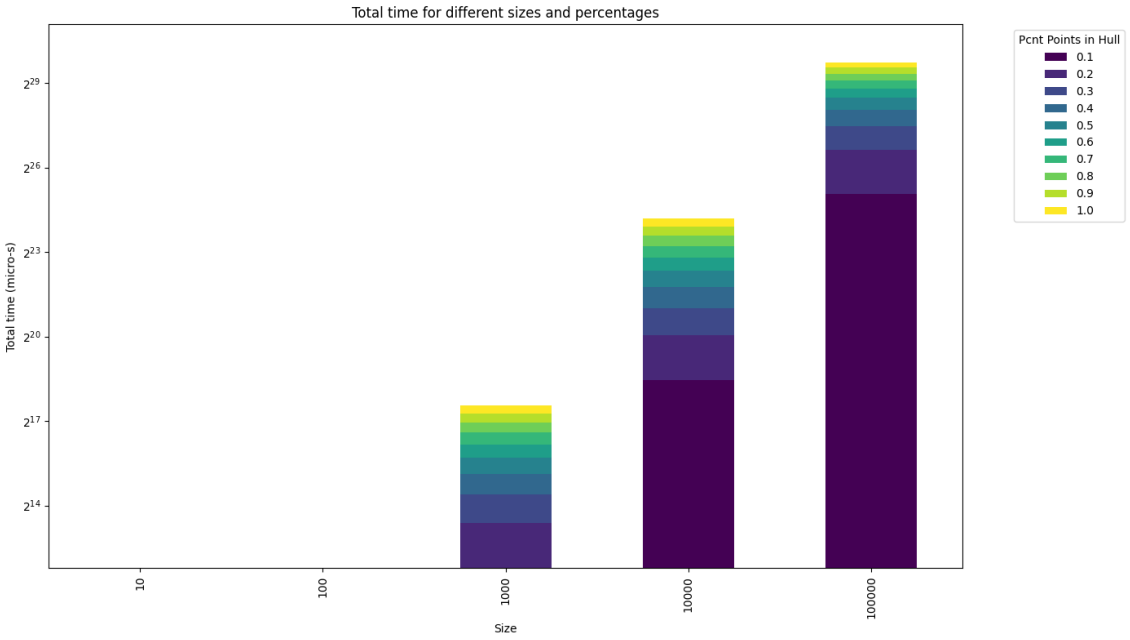

Figura 5: Gráfico de barra de algoritmo Gift Wrapping. Muestra el desempeño promedio de las simulaciones con un porcentaje de puntos en la cerradura. Cada línea indica un porcentaje utilizado. Eje y en escala log base 2

Se muestra a continuación lo mismo para el algoritmo Incremental.

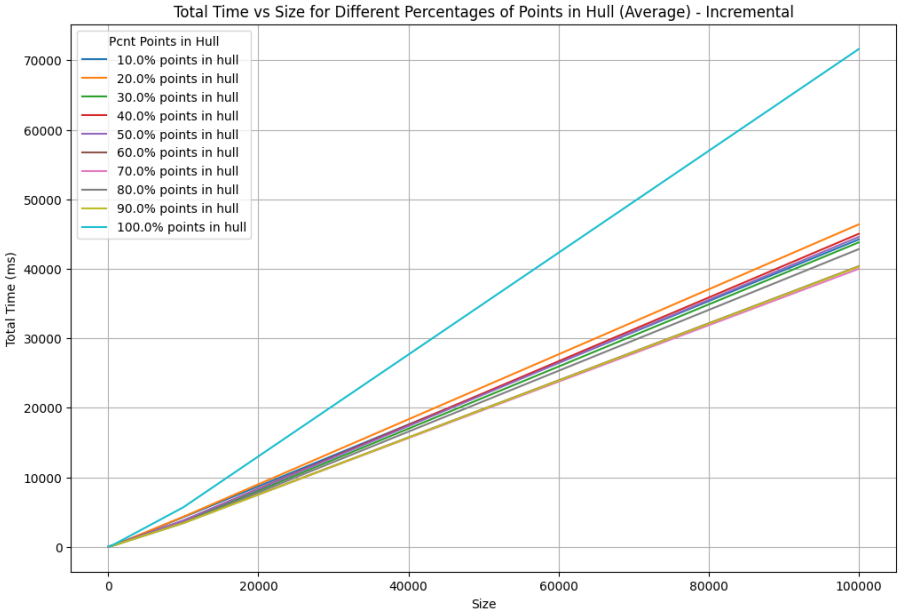

Figura 6: Gráfico de lineas de algoritmo Incremental. Muestra el desempeño promedio de las simulaciones con un porcentaje de puntos en la cerradura. Cada línea indica un porcentaje utilizado.

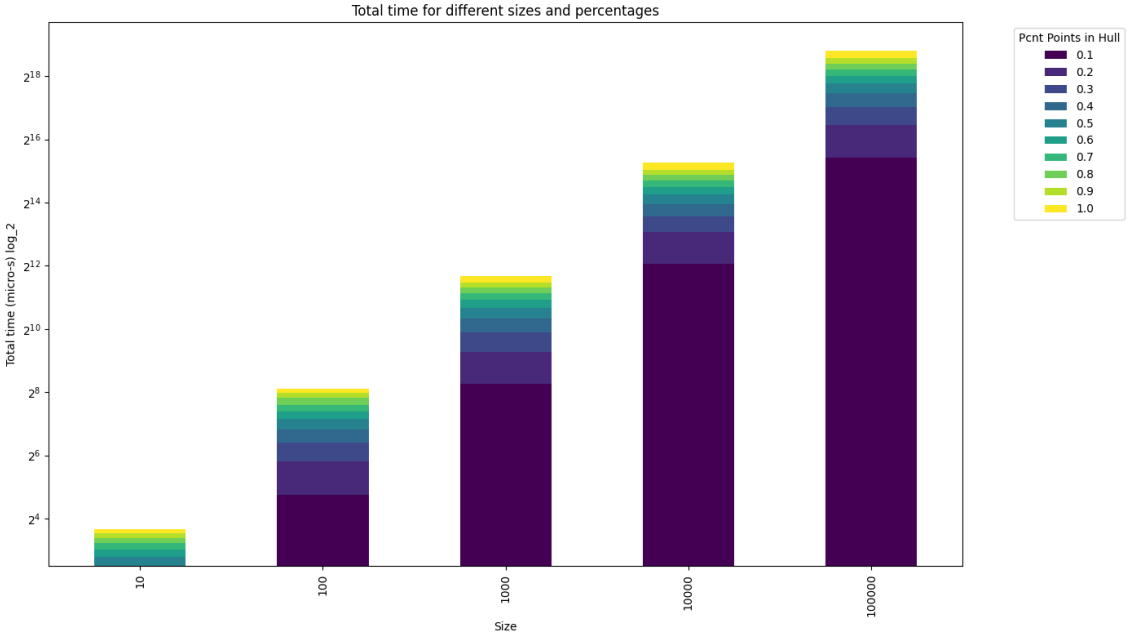

Figura 7: Gráfico de barra de algoritmo Incremental. Muestra el desempeño promedio de las simulaciones con un porcentaje de puntos en la cerradura. Cada línea indica un porcentaje utilizado. Eje y en escala log base 2

Lo siguiente es una comparativa en gráficos de barra, cada uno representando un porcentaje de puntos en particula, con el eje y en escala logarítmica para que se logre apreciar los resultados entre distintos valores de "size". En el eje x los size $0, 1, 2, 3, 4$ representan los valores $10^1, 10^2, 10^3, 10^4, 10^5$.

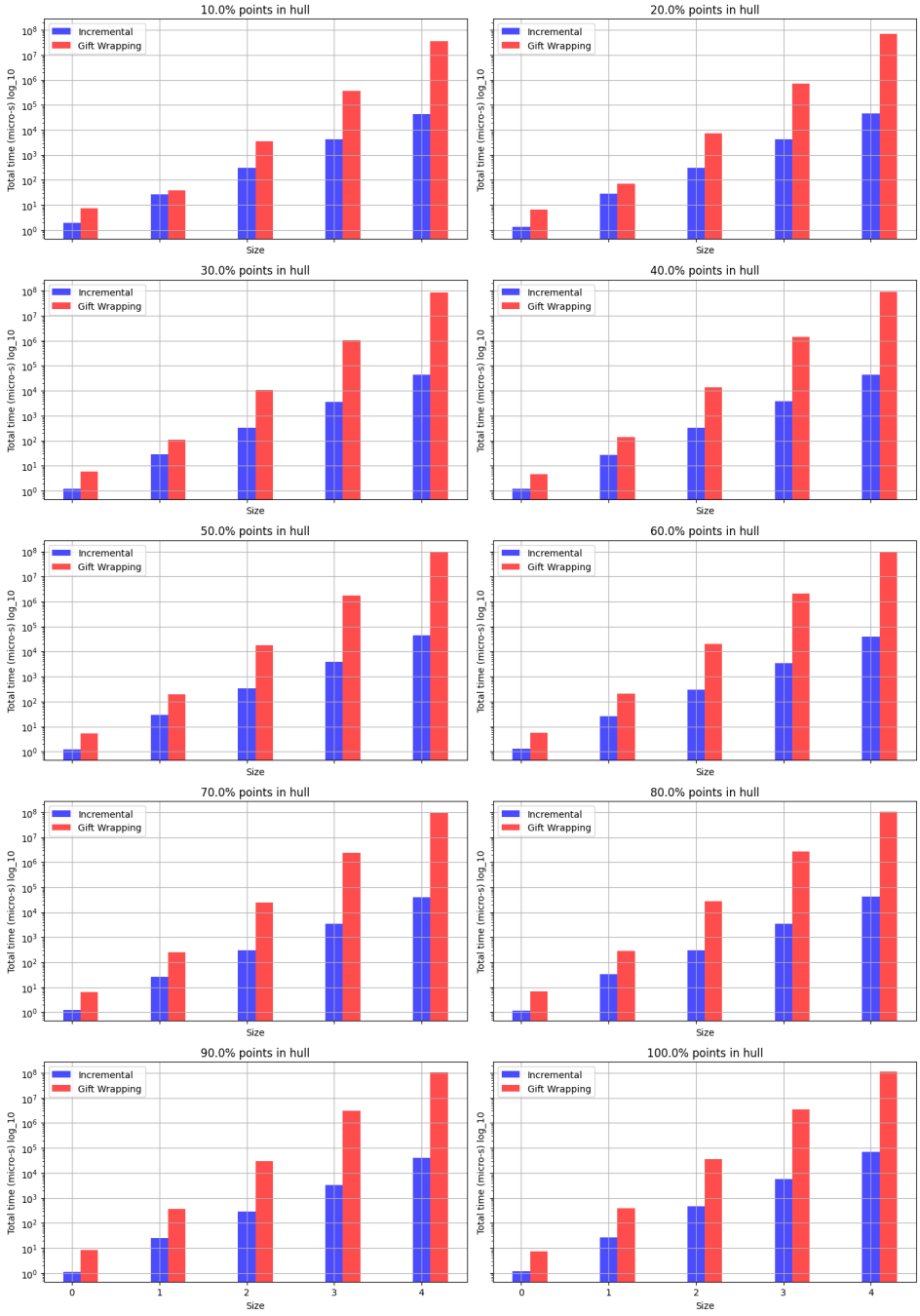

Figura 8: Gráfico de barra. Comparación del desempeño promedio de 2 algoritmos por cada porcentaje.

Se muestra finalmente el speedup alcanzado por el algoritmo Incremental en comparación al de Gift Wrapping, para cada porcentaje y para cierto tamaño. El eje x denota los mismos valores que en la figura 8.

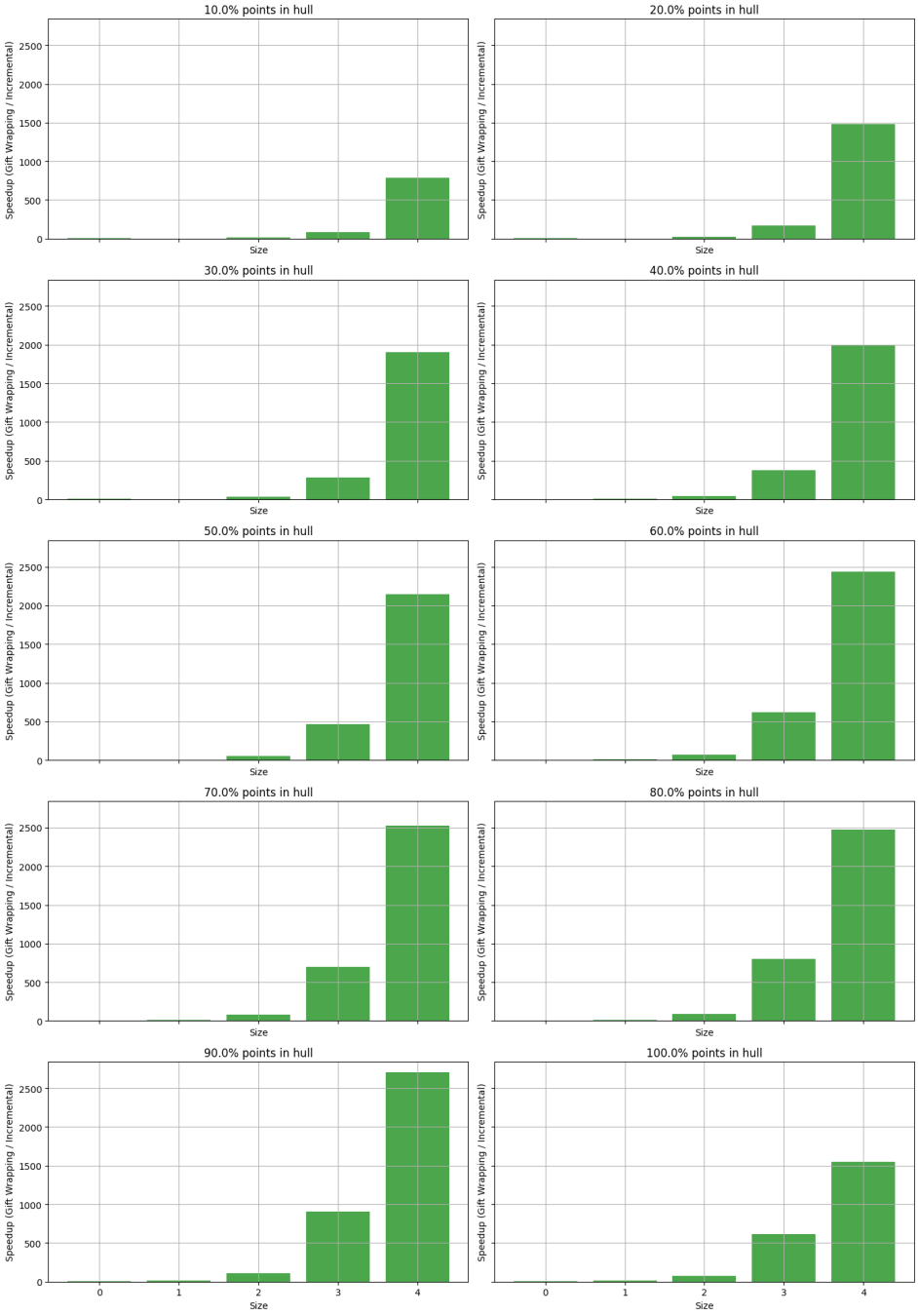

Figura 9: Gráfico de barra. Speedup alcanzado por el algoritmo Incremental vs el de Gift Wrapping. Se muestra para cada porcentaje y entre cada uno de los tamaños de nube.

Por último se quería además visualizar el uso de los tiempos por cada algoritmo.

Primero el Gift Wrapping, en el cual se tomaron los tiempos para encontrar el valor mínimo de componente x entre los puntos y el tiempo que se toma en realizar las operaciones para determinar la orientación entre puntos.

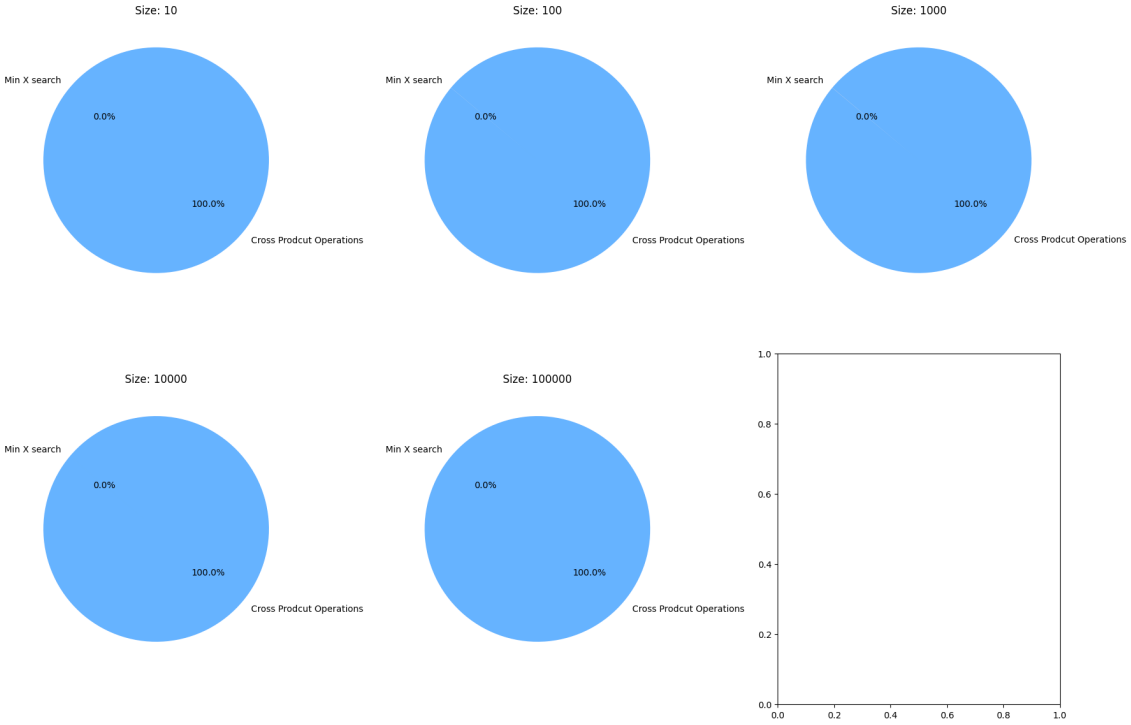

Figura 10: Gráfico de torta. Muestra el porcentaje de tiempo tomado por cada parte cronometrada con respecto al total. Es un gráfico por porcentaje de puntos en el hull. Algoritmo de Gift Wrapping.

Luego el Incremental se tomaron los tiempos para ordenar los puntos según la componente x, el tiempo que se toma en construir el upper-hull y el tiempo necesario para construir el lower-hull.

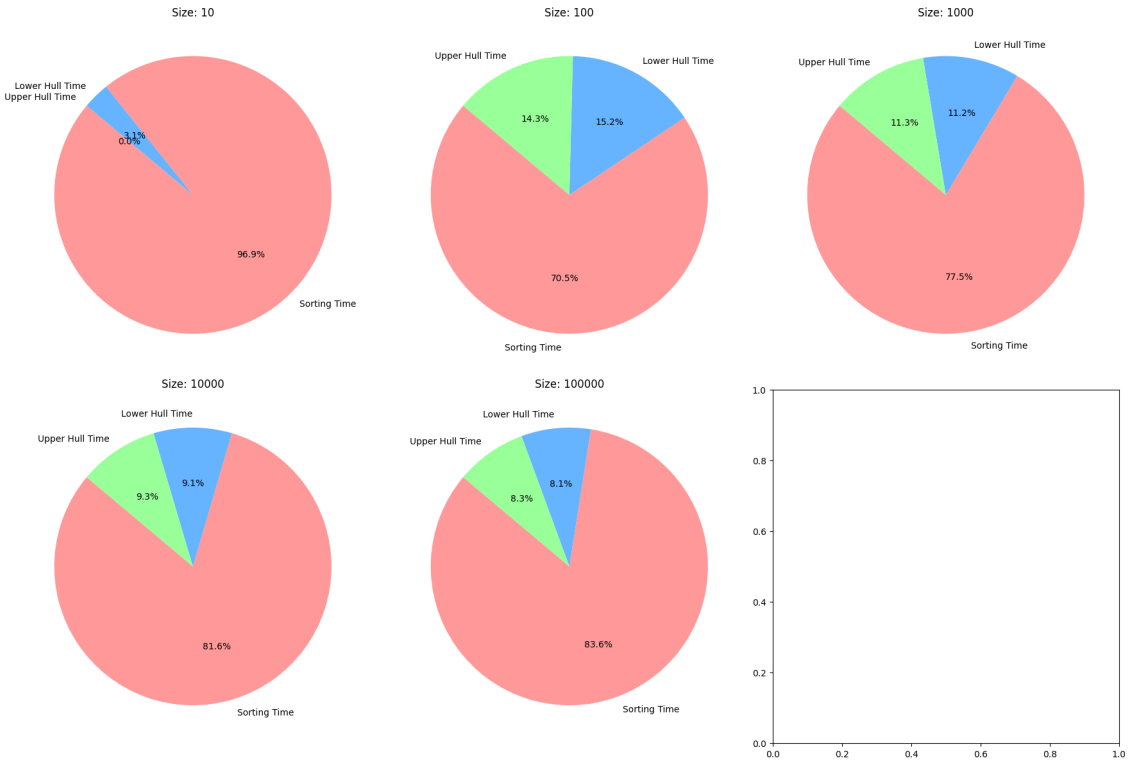

Figura 11: Gráfico de torta. Muestra el porcentaje de tiempo tomado por cada parte cronometrada con respecto al total. Es un gráfico por porcentaje de puntos en el hull. Algoritmo de Incremental.

4. Análisis de Resultados

En las figuras 1 y 2 se puede observar la diferencia en eficiencia entre las implementaciones hechas y cómo aumenta la brecha entre promedios de tiempo a medida que aumenta el tamaño de la nube de puntos. Además en el primer gráfico mencionado es apreciable lo variables que son los valores para el primer algoritmo en comparación al segundo, el cual tiene una línea de crecimiento más delgada.

En el siguiente gráfico (3) si bien no se puede observar correctamente los valores de tiempo promedio para la implementación de Incremental, pero sí se evidencia la diferencia de eficiencia entre las implementaciones cuando se controla la cantidad de puntos que pertenecen a la cerradura.

Para la imagen 4 se aprecia cómo a medida que se aumenta el porcentaje de puntos en el hull el tiempo que se toma el primer algoritmo aumenta drásticamente. A diferencia de eso en 6 el algoritmo Incremental es parejo en su comportamiento para todos los porcentajes de puntos en la cerradura a excepción de cuando todos los puntos están en la circunferencia exterior.

En 5 se muestra lo mismo, un aumento drástico del tiempo a medida que aumenta el porcentaje de puntos. Se recuerda que el eje y está en escala logarítmica en base 2, y el salto entre el tamaño 100 a 1000 es notable, a diferencia de la imagen 7 para el algoritmo Incremental, donde se ve un crecimiento proporcional.

Para los gráficos en 8 la comparación es más directa aún entre los algoritmos (reiterando que el eje y está en escala logarítmica) junto con 9, alcanzando una ventaja de miles de veces en velocidad.

Por último tomando en cuenta los gráficos de torta de las figuras 10 y 11 se puede decir lo siguiente: en Gift Wrapping el tiempo se dedica casi totalmente a las operaciones de cálculo de orientación entre tríos de puntos. Para el algoritmo incremental la mayoría del tiempo es dedicado al ordenamiento de los puntos, dedicando menos de la mitad del tiempo en la construcción de las cerraduras en la mayoría de los casos.

5. Conclusiones

Ya habiendo revisados los resultados presentados en el informe y en el archivo de experimentación (exp.ipynb) se pueden mencionar conclusiones no contempladas en el proceso y responder las preguntas planteadas en un inicio además de poder hablar acerca del comportamiento de los algoritmos.

No se contemplaba el impacto que tendría el cálculo de valores de tipo flotante el tratar con coordenadas muy cercanas. En la experimentación antes de proceder con la generación de visualizaciones se revisaron que las cerraduras creadas por ambos algoritmos fueran idénticas. La igualdad no se cumplía cuando se trataba de nubes de puntos de tamaño 10^5 y algunos de los porcentajes de puntos probados (0.5, 0.8 y 0.9), y la diferencia era de menos de 10 puntos.

Con respecto a las nubes de puntos aleatorios se puede concluir que ambos algoritmos tienen un comportamiento bastante adecuado a la teoría: para GW (Gift Wrapping) era mayor el costo y este aumentaba su diferencia con el tamaño anterior gradualmente, haciendo caso al $\mathcal{O}(n \cdot h)$ que se estipula en los libros, pensando en que las cerraduras eran de pocos puntos en comparación al tamaño de la nube (revisar los .csv producidos). Para el In (Incremental) es un crecimiento estable, proporcional, nuevamente adaptado a la teoría $\mathcal{O}(n \cdot log(n))$, pues un aumento de 10 veces en los puntos es un valor pequeño al aplicarse el logaritmo. Cabe destacar que el costo es relativamente cercano porque la cantidad de puntos en el hull es muy baja en comparación al total de puntos, algo que beneficia al algoritmo de GW, y que no tiene incidencia para el In en nubes aleatorias.

Con respecto a las nubes controladas (procentaje de puntos en el hull) es mejor el algoritmo In. En los gráficos de linea se evidencia que entre más puntos tenga el hull (mayor porcentaje) el algoritmo GW llega a tener un costo casi cuadrático de ejecución, aún para porcentajes pequeños, pues en comparación al tamaño de la nube siguen siendo mucho vertices en la cerradura. A diferencia del In, el presenta comportamientos parejos y estables, lo cual es lógico al no ser sensible a la salida y teniendo que ordenar sí o sí el conjunto de puntos. Aún así se evidencia que es mejor mientras menos puntos tenga el hull.

Se comprueba por experiencia que el GW y su inestabilidad e ineficiencia en el cálculo no es conveniente en aplicaciones en las que es necesaria rapidez y robustez, a pesar de que tenga buenos resultados generalmente. Se prefiere la constancia de un algoritmo no sensible a la salida como In.

Respondiendo a las preguntas entonces:

- En la máquina en la que fue probado este experimento el In es superior al GW en todos los casos.
- En la teoría el GW debería ser mejor para porcentajes de puntos muy pequeños en comparación al tamaño de la nube, sin embargo, a medida que crece el porcentaje el In será mejor que el GW al tener que llevar el proceso de ordenamiento de todas formas y este último tendrá que efectuar el cálculo de todos contra todos al comprobar la orientación de puntos.
- Existen diferencias en cuanto a tiempo de cálculo, son considerables (visto a lo largo del informe) y se apegan a la teoría.