UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Una primera aproximación a la Computación Cuántica

Informe de Proyecto de Grado presentado por

Diego Fernández, Pablo Palou y Uriel Radzyminski

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisor

Omar Viera

Montevideo, 20 de diciembre de 2023

Agradecimientos

Primero, queremos mencionar al Prof. MSc. Ing. Omar Viera, quien fue nuestro tutor y nos proporcionó dirección en este proyecto. Valoramos su disposición para ayudar y transmitirnos su conocimiento, así como para facilitar nuestra asistencia a charlas sobre Computación Cuántica que hicieron un diferencial en nuestros conocimientos.

Además, queremos agradecer a nuestros familiares, quienes han estado presentes a lo largo de nuestras carreras. Su confianza en nuestras habilidades y su respaldo nos han permitido continuar y esforzarnos por cumplir nuestras metas académicas y personales.

A nuestros amigos, compañeros de estudios y colegas, gracias por compartir con nosotros estos años. La carrera hubiera sido muy diferente sin el apoyo mutuo.

Queremos mencionar también a la institución educativa y a todo el personal docente y administrativo que nos ha proporcionado las herramientas y el entorno para desarrollar nuestras habilidades y conocimientos. Su respaldo a los estudiantes ha sido relevante para nuestro desarrollo personal y profesional.

Agradecemos a todos aquellos que, de alguna manera, han contribuido a este Proyecto de Grado. A aquellos que nos han dado su tiempo, conocimientos y recursos, y a quienes nos han acompañado en este proceso académico, les estamos agradecidos.

Motivación

La Computación Cuántica es un campo emergente y prometedor en el ámbito de la Ingeniería en Computación. Su potencial para revolucionar la manera en que resolvemos problemas computacionales ha capturado el interés de la comunidad científica y tecnológica en todo el mundo. En particular, su capacidad para abordar problemas altamente combinatorios, que son intrínsecos a muchas áreas de aplicación, ha sido objeto de numerosas investigaciones y desarrollos.

Dada la relevancia de la Computación Cuántica en la actualidad y su potencial impacto en el futuro de la Ingeniería en Computación, hemos decidido elegir la propuesta de Proyecto de Grado presentada por nuestro tutor. A continuación, se detallan los motivos que nos llevaron a tomar esta decisión.

Avance tecnológico y científico

La Computación Cuántica representa un avance tecnológico y científico. A medida que los límites de la Computación Clásica se vuelven cada vez más evidentes, la necesidad de explorar nuevos paradigmas computacionales se vuelve cada vez más fundamental. La Computación Cuántica ofrece la posibilidad de abordar problemas que, hasta ahora, eran inaccesibles para la Computación Clásica, lo que impulsa el avance del conocimiento en diversas áreas.

Aplicabilidad en problemas combinatorios

Los problemas combinatorios son comunes en la Ingeniería en Computación y en muchas otras disciplinas. Estos problemas suelen ser difíciles de resolver mediante algoritmos clásicos, ya que su complejidad crece exponencialmente con el tamaño del problema. La Computación Cuántica, en cambio, tiene el potencial de abordar estos problemas de manera más eficiente, lo que podría tener un impacto significativo en la solución de problemas reales y en la mejora de procesos en diversos campos.

Formación académica y profesional

Dado que es un área en constante crecimiento y con un alto potencial de impacto en el futuro, consideramos que adquirir conocimientos y habilidades en este campo es fundamental para nuestra formación académica y profesional. Al elegir esta propuesta de Proyecto de Grado, buscamos desarrollar una sólida base teórica y práctica que nos permita contribuir al avance de la disciplina y enfrentar los desafíos que se presenten en nuestra carrera profesional.

Contribución a la universidad y a la comunidad científica

Una de las razones adicionales para elegir esta propuesta de Proyecto de Grado es nuestra intención de contribuir al conocimiento y al avance de la investigación en la universidad y en la comunidad científica en general. Al abordar un tema tan relevante y en constante evolución como la Computación Cuántica, esperamos que los resultados de esta investigación puedan servir como base para futuras investigaciones y desarrollos en este ámbito.

Este trabajo podría inspirar a otros estudiantes y profesionales a adentrarse en este tema, fomentando la colaboración y el intercambio de ideas entre investigadores y expertos en el campo.

Interés en la mejora del rendimiento de algoritmos actuales

Otro factor que influyó en la decisión es nuestro interés en comprender si los algoritmos actuales podrían ser más eficientes mediante el uso de computadoras cuánticas, ya que se podría tener un impacto considerable en cuanto a la performance de los distintos algoritmos, lo cual puede tener grandes consecuencias a futuro.

Dado que la Ingeniería en Computación se centra en gran medida en el diseño y análisis de algoritmos, nos parece fundamental investigar cómo la Computación Cuántica podría mejorar el rendimiento de los algoritmos existentes y, en última instancia, llevar a soluciones más eficientes y efectivas. Al explorar esta cuestión en el contexto de problemas combinatorios, nos permitirá obtener una comprensión más profunda de las posibilidades y limitaciones del área en cuestión y su aplicabilidad en la mejora del rendimiento de algoritmos.

Abstract

Este trabajo se desarrolla bajo un Proyecto de Grado, en el cual se lleva a cabo un estudio del Estado del Arte en dos áreas clave: la Computación Cuántica y los problemas altamente combinatorios, en particular Clustering. El objetivo principal es hacer una primera aproximación a la Computación Cuántica y un primer análisis sobre si la misma puede o no mejorar el rendimiento de los algoritmos de Clustering en la actualidad.

En primer lugar, se realiza un estudio del Estado del Arte respecto de la Computación Cuántica y el Clustering. En el mismo, se brinda una explicación del funcionamiento de la Computación Cuántica, desde sus fundamentos teóricos hasta sus aplicaciones prácticas. Se exploran conceptos como qubits, superposición y entrelazamiento, y se presenta una revisión de las tecnologías y plataformas cuánticas disponibles en la actualidad. Respecto a Clustering se abordaron los diferentes algoritmos que se han desarrollado, sus características distintivas y aplicaciones. En particular, se analizan algoritmos clásicos como k-means y DBSCAN, mostrando distintas optimizaciones posibles con resultados prácticos.

Seguido a esto, se profundiza en la investigación sobre la utilización de la Computación Cuántica en el ámbito del Clustering. Se analizan los avances más recientes y las propuestas existentes para utilizar la potencia de la Computación Cuántica en el procesamiento de algoritmos de Clustering. Se exploran posibles mejoras en la eficiencia y precisión de los algoritmos de Clustering mediante la explotación de las características únicas de la Computación Cuántica. Para evaluar el impacto de la Computación Cuántica sobre estos algoritmos, se implementan ambas versiones, la clásica y la cuántica. Además, se buscan alternativas para optimizar estos algoritmos utilizando Computación Clásica y así tener un análisis de ambas soluciones en la actualidad.

Como resultado de este trabajo de investigación se llega a la conclusión, avalada por algunos expertos, que a pesar de los avances obtenidos en los últimos años, la Computación Cuántica aún no se presenta como una alternativa factible para la resolución de algoritmos de Clustering. Técnicas como paralelización, son ampliamente utilizadas por las empresas que brindan soluciones en la nube, debido a que son más escalables, confiables y abundantes en la actualidad respecto a la Computación Cuántica.

Palabras clave: Computación Cuántica, Computación Clásica, Clustering, Algoritmos de Clustering, Qubits, Superposición, Entrelazamiento, k-means, DBSCAN

Índice general

Introducción

En las últimas décadas, la computación ha experimentado una evolución notable, expandiendo sus fronteras más allá de los límites de la Computación Clásica [TechTarget,] hacia nuevos paradigmas que prometen resolver problemas que hoy en día resultan inabordables debido a su alta complejidad computacional. Uno de estos paradigmas emergentes es la Computación Cuántica [Amazon Web Services,], que ofrece un enfoque diferente para el procesamiento de información basado en los principios de la Mecánica Cuántica [Live Science,]. A pesar de su potencial, la aplicabilidad práctica de la Computación Cuántica en problemas específicos y su viabilidad económica sigue siendo un área de intensa investigación y debate.

Dentro de los problemas altamente combinatorios, se eligió Clustering [Analytics Vidhya,] debido a la vital importancia en campos como el Análisis de Datos [Coursera,], la Inteligencia Artificial [IBM,] y la Bioinformática [YourGenome,], campos con un gran impacto en la sociedad. Se destaca por ser un problema altamente combinatorio cuya complejidad crece superlinealmente con el tamaño del conjunto de datos. Los algoritmos clásicos, como k-means [DeepAI,] y DBSCAN [KDnuggets, 2020], han sido optimizados hasta cierto punto mediante técnicas de paralelización y optimización algorítmica; sin embargo, el aumento masivo en la generación de datos ha motivado la búsqueda de enfoques más radicales para su optimización.

El presente trabajo se centra en el estudio del Estado del Arte de la Computación Cuántica, evaluando su arquitectura, funcionamiento y los servicios y herramientas actuales que brindan soporte a este tipo de computación. Asimismo, se examina el problema de Clustering y las metodologías de optimización de algoritmos clásicos, contrastándolas con las posibilidades de optimización que ofrece la Computación Cuántica.

El objetivo general de este Proyecto de Grado es hacer una primera aproximación a la Computación Cuántica y un primer análisis sobre si la misma puede o no mejorar el rendimiento de los algoritmos de Clustering en la actualidad. Para lograr esto se detalla técnicamente el funcionamiento de la Computación Cuántica y su infraestructura; se revisa el estado actual de los servicios y herramientas de programación cuántica; se analizan los algoritmos de Clustering más utilizados y sus optimizaciones en el ámbito clásico; y se explora la factibilidad y eficacia de aplicar la Computación Cuántica en la optimización de estos algoritmos.

Los resultados esperados del estudio apuntan a establecer, a través de un análisis, si la Computación Cuántica representa una ventaja tangible para el problema de Clustering en su estado actual de desarrollo. A través de experimentos limitados, se busca obtener evidencia concreta de la efectividad de la Computación Cuántica en este campo.

Como resultado de este trabajo de investigación se llega a la conclusión (avalada por algunos

expertos), que a pesar de los avances obtenidos en los últimos años, la Computación Cuántica no constituye una solución práctica ni económica para la optimización de algoritmos de Clustering, cuando se compara con las metodologías de optimización clásica como la paralelización y la mejora algorítmica.

El presente informe está organizado de la siguiente manera:

En la sección 3, se encuentra el Marco Teórico. En el mismo se introducen los conceptos necesarios para la lectura de este trabajo. Se explican conceptos básicos relacionados a Computación Cuántica. Se desarrolla sobre Clustering y los algoritmos seleccionados para este trabajo. Una vez que se habla sobre Computación Cuántica y Clustering, se explica la forma con la que se combinan estas dos áreas en el presente trabajo.

Luego, en la sección 4 se encuentra el Desarrollo. Esta sección establece la fundamentación de las decisiones tomadas, junto con un análisis de costo actual tanto para el uso como también para la construcción de sistemas cuánticos. Además, se establecen las prácticas de desarrollo de software utilizadas.

Posteriormente, se encuentra la sección 5, en donde se encuentra la Experimentación. Esta sección contiene el análisis y los puntos claves que explican las conclusiones alcanzadas. Se establecen los principales impedimentos encontrados para la implementación de algoritmos cuánticos, se presenta un análisis de los algoritmos de Clustering seleccionados, se compara la performance al agregar técnicas de paralelización y se evalúa la performance que se podría alcanzar con Computación Cuántica.

A continuación está la sección 6, en la cual se presenta una charla con expertos que tuvo lugar el día 8 de noviembre de 2023 en la Universidad de Montevideo. Este espacio fue muy valioso para el presente trabajo, ya que ayudó a validar las conclusiones presentadas.

En la sección 7, se presentan las conclusiones obtenidas en esta investigación, así como también los aportes brindados y una autocrítica de lo que se hizo.

En la sección 8 se establecen líneas de trabajo futuro, en estas se identifican: realizar un trabajo de investigación diseñando algoritmos exclusivamente cuánticos, análisis en profundidad de las plataformas existentes para realizar experimentos cuánticos y desarrollo de APIs para Qiskit.

Por último se encuentran las Referencias y el Anexo, en donde se encuentran las implementaciones realizadas.

Marco Teórico

Aquí, se presenta una revisión de antecedentes en el tema del trabajo, incluyendo productos, procesos, publicaciones y desarrollos a nivel académico y comercial relacionados con la Computación Cuántica y Clustering. La información presentada en este capítulo es extraída del Estado del Arte que se realizó previamente, y proporciona una introducción a los conceptos necesarios para comprender el trabajo desarrollado en este Proyecto de Grado.

3.1. Complejidad Computacional

La Complejidad Computacional es un campo de la Ciencia de la Computación que se encarga de comparar y clasificar la dificultad de resolver problemas.

Se citarán algunos problemas para analizar su dificultad y comprender las diferentes categorías existentes. [Dean, 2021]

Dados dos números naturales n y m donde se busca averiguar si son relativamente primos. Dada una fórmula proposicional ϕ ¿tiene una asignación que la cumple? Si se jugara al ajedrez en un tablero de tamaño nn ¿tienen las blancas una estrategia ganadora a partir de una posición inicial dada? La característica en común de estos problemas es que son igualmente difíciles desde el punto de vista de la teoría clásica de la computabilidad, en el sentido de que todos son efectivamente decidibles. Sin embargo, parecen diferir significativamente en dificultad práctica. Si se dispone de un par de números $m > n > 0$ es posible determinar su primalidad relativa mediante un método (algoritmo de Euclides) que requiere un número de pasos proporcional a $\log n$. Por otra parte, todos los métodos conocidos para resolver los dos últimos problemas requieren una búsqueda de ''fuerza bruta" a través de una gran clase de casos que aumentan al menos exponencialmente en el tamaño de la instancia del problema. [Dean, 2021]

La teoría de la complejidad intenta definir o clasificar cada problema de acuerdo a la Complejidad Computacional para resolverlo. La clase de problemas P o tiempo polinómico, son aquellos problemas que pueden ser resueltos por una máquina de Turing convencional en un número de pasos proporcional a una función polinómica del tamaño de su entrada, el primer problema presentado entra en esta categoría. Luego el segundo problema se considera NP o tiempo polinómico no determinista, esta clase consiste de problemas que pueden resolverse correctamente mediante el cálculo de una máquina de Turing no determinista en un número de pasos que es una función polinómica del tamaño de su entrada. La pregunta de si P = NP es uno de los problemas más importantes y no resueltos en la Teoría de la Computación [Fortnow, 2009]. Por último, el tercer problema se entra dentro de

la categoría EXP, los problemas que su dificultad aumenta exponencialmente con el tamaño de la entrada.

El estudio de la Complejidad Computacional en el contexto de la Computación Cuántica ha llevado a la definición de nuevas clases de complejidad, como BQP, que es la clase de problemas solubles en tiempo polinomial por una máquina de Turing Cuántica. Aunque todavía hay muchas preguntas abiertas en la teoría de la Complejidad Computacional Cuántica, estos avances han proporcionado una base sólida para el desarrollo de algoritmos y hardware cuántico [Aaronson, 2013].

Dentro de la clase de problemas NP se encuentran problemas NP-Completos y NP-Duros. Los problemas NP-Completos son todos aquellos problemas X pertenecientes a NP tal que existe otro problema Y en NP tal que Y puede ser transformado a X en tiempo polinómico. Mientras que los problemas NP-Duros X son aquellos para los cuales existe un problema NP-Completo Y que puede ser transformado en tiempo polinomial al problema X. Lo que quiere decir que no tiene por qué existir una forma polinomial de verificar la solución a este tipo de problemas. [MathWorld, 2021] Estos problemas son al menos tan difíciles como los problemas NP-Completos. [Wikipedia, 2023]

3.2. Computación Cuántica

En esta sección se estudiarán los fundamentos de la Computación Cuántica. Para esto se partirá desde la unidad mínima de memoria hasta llegar a comprender algoritmos cuánticos. En este desarrollo se harán analogías con la Computación Clásica para marcar las principales diferencias y comprender la importancia de esta nueva área de investigación.

A pesar de ser un campo relativamente joven, la Computación Cuántica ha progresado significativamente en los últimos años, con importantes empresas como IBM [Jay Gambetta, 2023], Google [Castelvecchi, 2023] y Microsoft [Microsoft, 2023] invirtiendo fuertemente en el desarrollo de hardware y software cuántico. Esto ha llevado a la creación de potentes Computadoras Cuánticas que pueden realizar tareas a una velocidad superior respecto a las computadoras clásicas en determinado tipo de problemas de interés.

Esta sección se basará fuertemente en la siguiente fuente [Yanofsky and Mannucci, 2008]. En este libro, se aborda de manera técnica los principales conceptos de la Computación Cuántica, a lo largo del trabajo se utilizarán otras fuentes las cuales se encontrarán debidamente citadas.

3.2.1. Arquitectura

3.2.1.1. Bit

Desde la creación de las computadoras clásicas existe un concepto que resulta intuitivo y es el del bit.

El bit es una unidad de información que describe un sistema clásico de dos dimensiones.

En la vida cotidiana fácilmente se encuentran ejemplos de aplicaciones que se pueden representar mediante el uso de bits, por ejemplo indicar si una luz está prendida o apagada, representar la victoria o derrota en un juego o incluso representaciones booleanas de verdadero y falso.

Pero al pasar a la cuántica, estos conceptos que los seres humanos tienen tan fácilmente interiorizados, cambian. Aquí es donde entran en juego los qubits, los cuales aplicando propiedades de la Física Cuántica se aprovechan de sus características peculiares para sacar provecho en el rendimiento de algunas operaciones matemáticas, una característica fundamental de las computadoras cuánticas.

Es importante notar que en sistemas clásicos, el bit puede representar como máximo a dos estados, este aspecto es fundamental para comprender la diferencia con los sistemas cuánticos.

3.2.1.2. Qubit

La unidad básica de información cuántica es el bit cuántico (qubit) [Schumacher, 1995], el cual describe un sistema cuántico en dos dimensiones. Este es el análogo cuántico al clásico bit de la Computación Clásica.

Los qubits son representados por un vector de dos dimensiones, $\binom{c_0}{c_1}$, donde c_0, $c_1 \in \mathbb{C}$ tal que, $c_o^2 + c_1^2 = 1$.

De acuerdo con la Regla de Born [Condon and Morse, 1931], c_0^2 se puede interpretar como la probabilidad de que luego de medir un qubit el valor de este sea cero. Usando esta regla también se concluye que c_1^2 se puede interpretar de igual forma como la probabilidad de que la medición resulte en un uno.

La diferencia fundamental entre bits y qubits es la cantidad de información que contienen.

Para esto se necesita comprender cómo se representa la combinación de dos sistemas cuánticos. Aquí es donde entra en juego el producto tensorial.

El producto tensorial de dos espacios vectoriales V y W sobre un campo $\mathbb{F}$ se denota por $V \otimes_{\mathbb{F}} W$ y se define como el espacio vectorial generado por todos los posibles productos de elementos en V y elementos en W, sujetos a las relaciones bilineales:

$$
\begin{aligned}
(v_1 + v_2) \otimes w &= v_1 \otimes w + v_2 \otimes w \\
v \otimes (w_1 + w_2) &= v \otimes w_1 + v \otimes w_2 \\
(\alpha v) \otimes w &= \alpha(v \otimes w) \\
v \otimes (\beta w) &= \beta(v \otimes w)
\end{aligned}
$$

para todo $v, v_1, v_2 \in V$, $w, w_1, w_2 \in W$, y $\alpha, \beta \in \mathbb{F}$.

Dado que un qubit se representa por un vector de dos dimensiones perteneciente a los números complejos. La representación de dos qubits de acuerdo al producto tensorial será la siguiente:

Sean v_1, $v_2 \in \mathbb{C}^2$, entonces la combinación de estos dos sistemas está dada por el siguiente producto tensorial:

$v_1 \otimes v_2 \in \mathbb{C}^4$

Por esta razón, para la representación de dos qubits se necesitan cuatro números. Siguiendo esta lógica para la representación de n qubits se necesitan 2^n números.

Se puede observar un ejemplo de esta situación; para simplificar los cálculos, los qubits se representan mediante números reales. Se podría considerar que se disponen de los siguientes qubits, $v_1 = \binom{1/\sqrt{2}}{1/\sqrt{2}}$, $v_2 = \binom{1/\sqrt{2}}{1/\sqrt{2}}$,

entonces $v_1 \otimes v_2 = \binom{1/2}{1/2}{1/2}{1/2}$, con esto se observa que para representar el estado de dos qubits se necesitan 4 números.

A su vez, aplicando la Regla de Born, este sistema cuántico tomará cada valor posible con probabilidad de 1/4.

Esto es una de las bondades de las Computadoras Cuánticas, poder almacenar más información por cada unidad básica que la computadora clásica. La contrapartida de esto es que para representar el estado de un sistema cuántico, la cantidad de memoria requerida crece exponencialmente con la cantidad de qubits. Si se desea simular una computadora cuántica de 64 qubits, la cantidad de memoria para almacenar el estado es $2^{64} = 18,446,744,073,709,551,616$ números complejos. Esto supera ampliamente la capacidad de almacenamiento actual.

3.2.1.3. Operaciones sobre qubits

Las puertas lógicas cuánticas son elementos fundamentales de un circuito cuántico. Estas operan sobre un grupo de qubits, y matemáticamente, los qubits experimentan una transformación unitaria reversible al multiplicar su matriz unitaria con el vector de estado cuántico. Como resultado de esta operación, se obtiene un nuevo estado cuántico.

La medida cuántica es una operación irreversible, la cual es usada para medir el estado de un qubit. Luego de realizar esta medición, el qubit medirá 0 o 1 de acuerdo con las probabilidades definidas por la Regla de Born. El resultado de la medición de un solo qubit con el estado $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ será o bien $|0\rangle$ con una probabilidad de $|\alpha|^2$, o $|1\rangle$ con una probabilidad de $|\beta|^2$.

La inicialización o reinicialización a un valor conocido, a menudo $|0\rangle$. Esta operación colapsa el estado cuántico (exactamente como ocurre con la medición). La inicialización a $|0\rangle$ se puede implementar lógica o físicamente: Lógicamente, como una medición seguida de la aplicación de la compuerta Pauli-X si el resultado de la medición fue $|1\rangle$. Físicamente, por ejemplo, si se trata de un qubit de fase superconductora, reduciendo la energía del sistema cuántico a su estado base.

Enviar un qubit a través de un canal cuántico es un proceso crítico en la comunicación cuántica, ya que puede estar expuesto a errores y a la decoherencia cuántica. Sin embargo, esto es fundamental para la construcción de una red cuántica, que tiene el potencial de transformar la forma en que se procesa y se transfiere la información en el futuro.

3.2.1.4. Compuertas Clásicas

Para el funcionamiento de las computadoras clásicas [Thornton, 2006] se necesita manipular bits para la realización de diversas operaciones, las Compuertas Clásicas son la forma de manipular bits en este tipo de computadoras. Estas manipulaciones pueden ser de un bit, como por ejemplo el NOT que recibe un único bit o compuertas como AND, OR, las cuales realizan operaciones entre pares de bits.

Para que el pasaje a las Compuertas Cuánticas sea más sencillo, se tratarán a las compuertas como matrices, lo que permitirá realizar operaciones de matrices para conocer la salida de una compuerta dado una matriz que representa la entrada.

Como ejemplo, la compuerta NOT recibe un único bit, entonces la entrada se puede representar por un vector de dos dimensiones, esto debido a que la cantidad de posibles estados con un bit es dos, 0 o 1. Luego como salida también se espera un vector de dos dimensiones, ya que realizar la operación NOT solo modifica este bit. Con esto se concluye que se necesita una matriz de 2 filas y 2 columnas para la representación de la compuerta NOT.

Se define entonces la matriz NOT $= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Se puede verificar que esta matriz cumple con las salidas esperadas.

El bit 0 se encuentra representado por el vector $\begin{bmatrix}1\\0\end{bmatrix}$ mientras que el bit 1 se representa mediante el vector $\begin{bmatrix}0\\1\end{bmatrix}$.

Por ende, $\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix}$ y $\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$ las salidas esperadas.

De igual forma se puede deducir la matriz para el resto de Compuertas Clásicas, pero lo principal no es conocer la forma matricial de todas las Compuertas Clásicas, sino generar las intuiciones necesarias para el manejo de compuertas con matrices, esto será de ayuda para comprender el funcionamiento de las Compuertas Cuánticas.

3.2.1.5. Compuertas Reversibles

No todas las compuertas de las que se presentaron para la Computación Clásica funcionan en Computación Cuántica. En 1960 Rolf Landauer, llegó a la conclusión de que en algunas operaciones las compuertas lógicas generaban calor. El denominador común de estas operaciones es que todas eliminaban memoria.

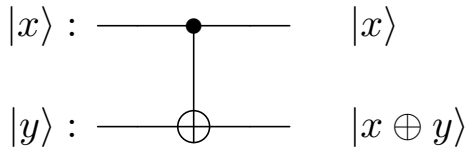
Por esta razón se generaron Compuertas Reversibles, para prevenir la eliminación de memoria y por ende mantener la coherencia del sistema, dado que no se genera pérdida de energía que impacta en el estado de los qubits.

Ahora bien, ¿qué implica que una compuerta sea reversible? Para responder a esta pregunta se analiza la compuerta OR. Si se obtiene que la salida de la compuerta OR es 0, ante este resultado se puede predecir de que ambas entradas eran 0. Pero qué pasa si la salida de esta compuerta es 1, ante este caso se tienen tres posibles entradas para las cuales la salida es 1. Esto se identifica a una pérdida de información, el sistema tuvo que eliminar información, generando calor.

Ante este problema se presentan las compuertas reversibles como una solución para prevenir la disipación de energía y por ende la generación de calor.

Un ejemplo de compuerta reversible es el NOT, se sabe que si la salida es el bit 1, entonces la entrada fue el bit 0 y si la salida es el bit 0, entonces la entrada fue el bit 1. Otro ejemplo de compuerta reversible es la compuerta identidad.

Dentro de las Compuertas Reversibles se destaca la compuerta NOT controlada (CNOT).



Esta compuerta toma el bit superior de la entrada (bits de la izquierda) como un bit de control. Si el bit de control $|x\rangle$ vale 0 entonces la salida tomará el valor de la entrada $|y\rangle$. Mientras que si el bit de control vale 1, entonces la salida será el opuesto de la entrada.

La matriz correspondiente a esta compuerta es la siguiente:

$$\begin{array}{c} \\ \mathbf{00} \\ \mathbf{01} \\ \mathbf{10} \\ \mathbf{11} \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{00} & \mathbf{01} & \mathbf{10} & \mathbf{11} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

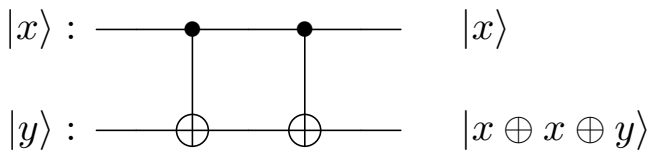
Se puede verificar el funcionamiento adecuado de la compuerta CNOT a través de su matriz correspondiente.

La entrada $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, es decir $|x\rangle = 1$, $|y\rangle = 0$, se representa por el vector $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, utilizando la matriz construida anteriormente para la compuerta CNOT. Se tiene el siguiente producto:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Que representa la salida buscada $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Es decir, $|x\rangle = 1$ ya que retorna el mismo valor que recibió como entrada e $|y\rangle = 1$ ya que retorna $|x\rangle \oplus |y\rangle = 1 \oplus 0 = 1$

La compuerta NOT controlada es fácilmente reversible, en la siguiente figura se observa la forma de obtener su reverso.



Esta compuerta presenta diversos usos, entre ellos el desarrollo de algoritmos, corrección de errores y circuitos cuánticos.

Algunos ejemplos de usos son los siguientes [Nielsen and Chuang, 2010]:

Entrelazamiento Cuántico: Al enviar dos qubits a una compuerta NOT controlada, un qubit tomará la posición de control y el qubit de destino será invertido o no según el estado del qubit de control, como se observó anteriormente, si el estado es 1 entonces el qubit de destino será invertido.

Compuertas lógicas cuánticas: La combinación de esta compuerta con otras permite la generación de circuitos cuánticos más complejos para la implementación de distintos algoritmos.

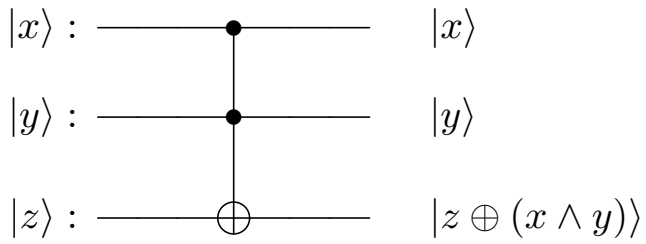
Algoritmos cuánticos: Las puertas CNOT desempeñan un papel crucial en muchos Algoritmos cuánticos, como el algoritmo de factorización de Shor, el algoritmo de búsqueda de Grover y la Transformada Cuántica de Fourier (QFT). Estos algoritmos explotan el paralelismo y el Entrelazamiento Cuántico para resolver problemas más rápidamente que los ordenadores clásicos.

Corrección cuántica de errores: Los códigos cuánticos de corrección de errores, se basan en puertas CNOT para corregir los errores que se producen durante el cálculo cuántico. Estas puertas ayudan a mantener la coherencia del sistema cuántico y lo protegen de la decoherencia causada por el entorno.

Computación Cuántica universal: Las puertas CNOT, junto con un conjunto suficiente de puertas de un solo qubit, forman un conjunto de puertas universal. Esto significa que cualquier cálculo cuántico puede realizarse utilizando únicamente estas puertas. Esta propiedad es esencial para el desarrollo de ordenadores cuánticos de propósito general.

Teletransporte cuántico: Las puertas CNOT se utilizan en el protocolo de teletransporte cuántico, que permite transmitir información cuántica de un qubit a otro sin mover físicamente los qubits. Este protocolo es esencial para la comunicación cuántica y el desarrollo de redes cuánticas.

Se pueden examinar otras Compuertas Reversibles que desempeñan un papel significativo en el ámbito de la Computación Cuántica. Un ejemplo interesante de compuerta reversible es la compuerta de Toffoli.



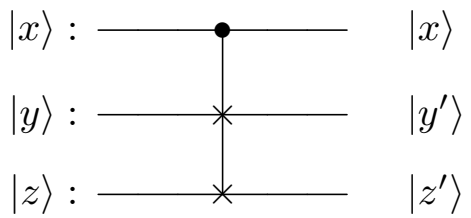
Los primeros dos bits de entrada (x, y) se denominan bits de control, los valores que estos tomen afectarán la salida.

Esta compuerta es interesante, ya que al igual que la compuerta NOT controlada es una compuerta universal, esto quiere decir que utilizando copias de la compuerta de Toffoli se pueden construir diferentes tipos de compuertas como lo son el AND y el NOT. Esto implica que se podría cumplir el requerimiento de no generar calor para mantener el estado de los sistemas cuánticos que integren la computadora en la cual se desea hacer uso de diferentes operaciones lógicas.

Esta compuerta se representa mediante la siguiente matriz:

	000	**001**	**010**	**011**	**100**	**101**	**110**	**111**
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	0	1
111	0	0	0	0	0	0	1	0

Otra compuerta interesante es la de Fredkin. Es una compuerta de tres qubits que intercambia los estados de dos qubits condicionados por el estado de un qubit de control. Es representada con el siguiente circuito.

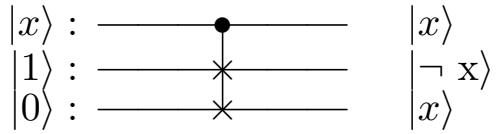


Entonces si el estado de $|x\rangle = 0$ entonces la salida no se verá afectada y $|y'\rangle = |y\rangle$ y $|z'\rangle = |z\rangle$. Por lo contrario, si el valor del qubit de control es $|x\rangle = 1$, entonces la salida será invertida de tal modo que $|y'\rangle = |z\rangle$ y $|z'\rangle = |y\rangle$.

La matriz que corresponde a esta compuerta es la siguiente:

	000	**001**	**010**	**011**	**100**	**101**	**110**	**111**
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	1	0	0	0	0
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	0	1	0
110	0	0	0	0	0	1	0	0
111	0	0	0	0	0	0	0	1

Se puede afirmar que esta compuerta, similar a la de Toffoli, es universal. Se procederá a demostrar cómo se puede construir la compuerta NOT utilizando los valores propuestos, $|y\rangle = 1$ y $|z\rangle = 0$.



3.2.1.6. Compuertas Cuánticas

Una compuerta cuántica[Deutsch, 1985] es un operador que toma qubits como entrada y los manipula de acuerdo a su propósito. Estos operadores son representados por una matriz unitaria.

Se han examinado algunas Compuertas Cuánticas de relevancia. A continuación, se introducirán otras igualmente fundamentales en el campo de la Computación Cuántica.

Existen unas matrices que se utilizan ampliamente en la Computación Cuántica, estas son las matrices de Pauli, son usadas principalmente para el desarrollo de algoritmos. Se definen las siguientes matrices.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Las matrices de Pauli tienen diversas aplicaciones en la Computación Cuántica, entre ellas [Nielsen and Chuang, 2010]:

Compuertas de un solo qubit: Las compuertas Pauli-X, Pauli-Y y Pauli-Z corresponden a rotaciones alrededor de los ejes X, Y y Z de la esfera de Bloch (es una representación geométrica de los estados monocubitales puros como un punto de la unidad), respectivamente. Estas puertas pueden utilizarse para manipular y transformar estados cuánticos.

Corrección de errores: Las compuertas Pauli se utilizan en los códigos estabilizadores, una clase de códigos cuánticos de corrección de errores ampliamente estudiada. Ayudan a detectar y corregir errores que pueden producirse durante la computación o la comunicación cuántica debido a lo que se conoce como la decoherencia de un sistema cuántico.

Algoritmos cuánticos: Las matrices de Pauli aparecen en varios algoritmos cuánticos, como el algoritmo cuántico de estimación de fase y la Transformada Cuántica de Fourier. Se utilizan para aplicar transformaciones y operaciones específicas en los qubits.

Medición cuántica: Los operadores de Pauli se utilizan para describir mediciones cuánticas en la base computacional y en otras bases. Ayudan a comprender las probabilidades de medir diferentes resultados y los estados posteriores a la medición.

Dinámica cuántica: Las matrices de Pauli son esenciales para describir la evolución temporal de los sistemas cuánticos, especialmente en el contexto de los sistemas cuánticos abiertos y el estudio de la decoherencia.

Entrelazamiento Cuántico: Las matrices de Pauli se utilizan para analizar y cuantificar el Entrelazamiento Cuántico. En general, las matrices de Pauli son herramientas fundamentales en Computación Cuántica, ya que proporcionan una base para comprender y manipular estados cuánticos, implementar Compuertas Cuánticas y diseñar algoritmos cuánticos.

3.2.2. Backends cuánticos

3.2.2.1. IBM

IBM [IBM, 2019] es una de las entidades pioneras en la implementación y desarrollo de procesadores cuánticos, y ha establecido un referente en la industria con su plataforma IBM Quantum. La compañía ha adoptado el uso de transmones, una forma de qubits superconductores, como el fundamento de su hardware cuántico. Estos transmones están diseñados para operar en temperaturas extremadamente bajas, cercanas al cero absoluto, lo cual es esencial para minimizar las interacciones no deseadas con el entorno y preservar la coherencia cuántica. La tecnología de IBM se centra en la escalabilidad y en la mejora continua de la calidad de los qubits, lo que es crucial para el aumento del volumen cuántico, una métrica que considera tanto el número de qubits como la calidad de las operaciones que se pueden realizar con ellos.

Un aspecto distintivo de la propuesta de IBM en Computación Cuántica es su plataforma abierta a la comunidad, IBM Quantum Experience, que permite a usuarios y desarrolladores acceder y experimentar con sus procesadores cuánticos a través de la nube. Esto no solo promueve la educación

y la investigación en el campo, sino que también acelera el desarrollo de algoritmos y aplicaciones cuánticas. Adicionalmente, IBM ha hecho significativos avances en la creación de una infraestructura completa de software que incluye Qiskit, un framework de código abierto para la programación cuántica.

3.2.2.2. Rigetti

Rigetti Computing [Rigetti Computing, a] se ha consolidado como una entidad innovadora en el ámbito de la Computación Cuántica, caracterizándose por su enfoque en la integración vertical de sus sistemas cuánticos. La arquitectura de sus procesadores cuánticos se basa también en qubits superconductores, particularmente en la utilización de transmones, similares a los de IBM, pero con una aproximación única en su diseño y ensamblaje. El compromiso de Rigetti hacia la construcción de un ordenador cuántico práctico se refleja en su estrategia de fabricación de chips cuánticos, donde la compañía controla directamente tanto el desarrollo del hardware cuántico como del software asociado, con el objetivo de optimizar el rendimiento y la integración del sistema completo.

Rigetti ha ganado atención por su plataforma de Computación Cuántica como servicio, denominada Quantum Cloud Services (QCS), que permite a los usuarios ejecutar aplicaciones cuánticas en la nube, con un modelo híbrido clásico-cuántico que busca maximizar las ventajas de los procesadores cuánticos en problemas específicos. Además, Rigetti se ha enfocado en la creación de entornos de desarrollo que facilitan la implementación de circuitos cuánticos y fomentan el descubrimiento de algoritmos eficientes para su hardware específico. Este enfoque orientado al servicio y al ecosistema de desarrollo sitúa a Rigetti como un actor clave en la evolución de la tecnología cuántica y en su posible adopción en la industria y la investigación científica.

3.2.2.3. IonQ

IonQ [IonQ,] se distingue en el panorama de la Computación Cuántica por su enfoque en el desarrollo de procesadores cuánticos basados en iones atrapados, una alternativa a los qubits superconductores utilizados por empresas como IBM y Rigetti. Los procesadores de IonQ operan mediante el atrapamiento de iones individuales mediante campos electromagnéticos y utilizan láseres para realizar operaciones cuánticas entre ellos. Esta técnica aprovecha las propiedades naturales de los átomos y ofrece tiempos de coherencia relativamente largos, así como altas fidelidades en las puertas cuánticas, aspectos cruciales para el procesamiento de información cuántica.

IonQ ha tomado la iniciativa en el campo con el lanzamiento de sistemas cuánticos comercialmente disponibles que prometen escalabilidad y alta conectividad entre qubits, dos retos persistentes en la Computación Cuántica. La capacidad de estos procesadores para interactuar con cualquier qubit con cualquier otro proporciona una gran flexibilidad en la implementación de algoritmos cuánticos. Además, la empresa se ha enfocado en la accesibilidad de su tecnología, ofreciendo su hardware a través de plataformas en la nube líderes en la industria, lo que permite a los investigadores y desarrolladores de todo el mundo explorar las capacidades de esta tecnología emergente. La promesa de IonQ de contribuir a la próxima generación de Computación Cuántica reside en su capacidad para ofrecer una plataforma con errores corregibles y escalable, sentando las bases para lo que podría ser una ventaja significativa en la consecución de una computadora cuántica plenamente funcional y de uso general.

3.2.2.4. QuEra

QuEra Computing Inc. [QuEra,] se está posicionando en el vanguardista sector de la Computación Cuántica con una propuesta distintiva: la utilización de átomos neutros en redes ópticas como qubits para sus procesadores cuánticos. Esta tecnología emplea átomos fríos atrapados en una matriz de puntos de luz generados por láseres, que pueden ser manipulados individualmente o en conjunto para realizar operaciones cuánticas. Esta aproximación permite a QuEra ofrecer una alta fidelidad en las operaciones cuánticas y la capacidad de reconfigurar el sistema de qubits de manera dinámica, una característica que potencia la flexibilidad en la ejecución de algoritmos cuánticos complejos.

El enfoque de QuEra destaca por su potencial para escalar a un gran número de qubits, dada la naturaleza intrínsecamente ordenada y reconfigurable de sus redes ópticas. Los procesadores cuánticos de QuEra son prometedores por su capacidad de inicializar y leer qubits con alta precisión, además de la notable coherencia de su sistema cuántico. Esta tecnología coloca a QuEra como un jugador innovador en la carrera hacia la realización de cálculos cuánticos prácticos y el desarrollo de nuevas clases de algoritmos cuánticos. La empresa también está enfocada en resolver problemas específicos que serían intratables para las supercomputadoras clásicas, apuntando a un nicho de aplicaciones cuánticas con impacto directo en sectores como la optimización, la química cuántica y los materiales avanzados.

3.2.2.5. Oxford Quantum Circuits

Oxford Quantum Circuits (OQC) [Oxford Quantum Circuits,] es una compañía británica que se está forjando una reputación en el ecosistema de la Computación Cuántica a través de su enfoque innovador en el diseño y fabricación de procesadores cuánticos. La empresa se basa en la tecnología de qubits superconductores, empleando un diseño patentado conocido como el 'Coaxmon', el cual es una evolución de los tradicionales transmones utilizados en la industria. La arquitectura del Coaxmon se centra en una implementación tridimensional de los qubits, que ofrece ventajas en términos de escalabilidad y coherencia, además de una mejor accesibilidad para el control y la medición, aspectos fundamentales para el avance en la construcción de ordenadores cuánticos más complejos y funcionales.

OQC también se distingue por su infraestructura de Computación Cuántica en la nube, denominada Lucy, que brinda a los usuarios acceso remoto a sus procesadores cuánticos. Esto permite a investigadores y desarrolladores de diversas industrias explorar aplicaciones prácticas y algoritmos cuánticos sin la necesidad de poseer una infraestructura Física Cuántica propia. Con un firme enfoque en la accesibilidad y la colaboración con socios estratégicos en múltiples sectores, OQC está expandiendo las fronteras de lo que es posible con la tecnología cuántica actual. Esta aproximación no solo impulsa la investigación y la innovación en el campo, sino que también acelera el desarrollo hacia la solución de problemas complejos a través de la Computación Cuántica.

3.2.2.6. D-Wave Systems

D-Wave Systems [D-Wave Systems, a] es un actor único en el escenario de la Computación Cuántica, especializándose en el desarrollo de computadoras cuánticas de tipo annealer, o recocido cuántico. A diferencia de los enfoques universales que persiguen empresas como IBM o Rigetti, D-Wave se concentra exclusivamente en producir sistemas cuánticos diseñados para resolver problemas de optimización. La tecnología de recocido cuántico de D-Wave explota las propiedades de los qubits

superconductores para buscar el mínimo global de una función de energía, un proceso análogo al enfriamiento lento de materiales en la Física.

Los procesadores de D-Wave, como el D-Wave 2000Q y el más reciente Advantage, contienen miles de qubits, lo que les permite abordar problemas de optimización y muestreo que son computacionalmente demandantes para los sistemas clásicos. D-Wave ha llevado a cabo mejoras continuas en la conectividad de sus qubits y en la eficiencia de su enfoque de enfriamiento, lo que se traduce en una mayor capacidad para encontrar soluciones a problemas complejos de optimización combinatoria y de búsqueda de energía mínima en diversas aplicaciones prácticas, desde la logística hasta el aprendizaje automático.

Además, D-Wave proporciona un conjunto de herramientas de software intuitivas y potentes que permiten a los usuarios implementar y ejecutar algoritmos cuánticos adaptados a sus necesidades específicas. La compañía promueve una aproximación práctica a la Computación Cuántica con un modelo de acceso a través de la nube, facilitando a las empresas y a los investigadores la exploración de las capacidades de esta tecnología emergente sin inversiones significativas en infraestructura. Con una estrategia claramente diferenciada, D-Wave se posiciona como un pionero en la aplicación de la Computación Cuántica a problemas del mundo real.

3.2.2.7. Honeywell

Honeywell, [Honeywell,] una compañía con una extensa trayectoria en diversos sectores industriales, ha emergido como un innovador prominente en el ámbito de la Computación Cuántica con su división Honeywell Quantum Solutions. El enfoque de Honeywell en esta tecnología disruptiva se basa en el uso de iones atrapados como qubits, similar a IonQ, pero Honeywell ha implementado su propia versión de la tecnología de trampas de iones con algunas diferencias clave. La plataforma de Honeywell se caracteriza por sus elevados niveles de precisión en las operaciones cuánticas, lo que se atribuye al uso de qubits ultrafríos y una interacción altamente controlada entre iones mediante láseres de precisión.

Uno de los principales logros de Honeywell ha sido el desarrollo del concepto de "Volumen Cuántico", una métrica que combina el número de qubits, la conectividad de los qubits, las tasas de error de las puertas y la coherencia de los qubits, para ofrecer una medida integral del poder y la utilidad de un procesador cuántico. Esto refleja la dedicación de Honeywell a construir no solo máquinas cuánticas con un alto número de qubits, sino sistemas con la capacidad para realizar cálculos cuánticos complejos y fiables.

La ambición de Honeywell de avanzar en el campo de la Computación Cuántica se materializa también en su plataforma de acceso en la nube, que democratiza la accesibilidad a sus sistemas cuánticos. Esta plataforma permite a científicos y empresas desarrollar algoritmos y explorar el potencial de la Computación Cuántica en aplicaciones prácticas sin la necesidad de un equipo físico. Con un enfoque que equilibra la innovación técnica y la colaboración estratégica, Honeywell busca acelerar la adopción de la Computación Cuántica en la industria y la ciencia, ofreciendo soluciones que podrían transformar sectores como la química de materiales, la optimización financiera y la ciberseguridad.

3.2.2.8. Xanadu

Xanadu [Xandau,] es una compañía pionera en el campo de la fotónica cuántica y se ha establecido como un actor disruptivo en la Computación Cuántica al desarrollar procesadores basados en qubits fotónicos. A diferencia de las plataformas basadas en qubits superconductores o iones atrapados,

Xanadu utiliza estados de la luz para realizar cálculos cuánticos, lo que constituye una aproximación radicalmente diferente y que ofrece ventajas inherentes como la ausencia de enfriamiento criogénico y la potencialidad para operar a temperatura ambiente.

La tecnología de Xanadu destaca por su plataforma Strawberry Fields, una suite de software dedicada a la fotónica cuántica, y por su lenguaje de programación Blackbird, diseñado para construir, simular y ejecutar programas cuánticos a través de su procesador cuántico en la nube, conocido como Xanadu Quantum Cloud. Este sistema permite a los usuarios acceder a circuitos fotónicos cuánticos, que son esenciales para aplicaciones en el muestreo cuántico, la optimización y el aprendizaje automático. Además, la empresa se ha concentrado en la construcción de un ecosistema de software robusto que facilita la integración con otros sistemas y tecnologías de Inteligencia Artificial.

El compromiso de Xanadu con la escalabilidad se refleja en su enfoque continuo en la investigación y desarrollo, buscando constantemente aumentar el número de qubits fotónicos y la complejidad de las interacciones entre ellos. Con su plataforma basada en la nube y su fuerte enfoque en las tecnologías fotónicas cuánticas, Xanadu apunta a liderar una transición hacia algoritmos cuánticos más prácticos y accesibles, promoviendo una visión de Computación Cuántica que podría ser integrada en el tejido de la computación convencional en un futuro cercano.

3.2.3. Frameworks y librerías

El campo de la Computación Cuántica ha experimentado un rápido crecimiento en los últimos años, y con él, ha surgido la necesidad de lenguajes de programación y marcos de trabajo específicos para diseñar, simular e implementar algoritmos cuánticos. Estos lenguajes y marcos permiten a investigadores y desarrolladores trabajar con diversas plataformas de hardware cuántico y simuladores. A continuación, se presenta una descripción de algunos de los principales lenguajes de programación y marcos de trabajo en Computación Cuántica, junto con enlaces a sus páginas web principales.

3.2.3.1. Qiskit

Qiskit (Quantum Information Science Kit)[Qiskit Developers, 2017]: Es un kit de herramientas de código abierto desarrollado por IBM, Qiskit es una biblioteca de Python de código abierto que permite a los usuarios crear, manipular y ejecutar circuitos cuánticos en hardware cuántico real o simuladores. También proporciona herramientas para la investigación de algoritmos cuánticos, optimización y corrección de errores.

La historia de Qiskit se remonta a 2016, cuando IBM lanzó el proyecto denominado IBM Quantum Experience. Fue una plataforma en línea que permitía a los usuarios y desarrolladores acceder y experimentar Compuertas Cuánticas reales. Fue una de las pioneras en el área y aportó gran valor en las primeras épocas del desarrollo de computadoras cuánticas.

El siguiente año, IBM anunció la creación de Qiskit, como un componente para trabajar con la plataforma IBM Quantum Experience. Desde ese momento esta plataforma se encuentra en desarrollo y ha aumentado la cantidad de funcionalidades que ofrece.

Esta plataforma está construida en base a 4 componentes principales, Qiskit Terra, Qiskit Aer, Qiskit Ignis y Qiskit Aqua.

En Qiskit Terra se puede construir, optimizar y ejecutar circuitos cuánticos en diferentes arquitecturas, (simuladores o computadores cuánticos reales). En esta plataforma también se encuentra un compilador que puede convertir circuitos cuánticos en código ejecutable por estas computadoras. Luego, con Qiskit Aer se tiene acceso a una variedad de simuladores de alto rendimiento para validar el correcto funcionamiento de circuitos cuánticos. Estos simuladores también permiten simular el ruido que se presenta en estos sistemas para construir algoritmos más robustos. En Ignis se cuenta con una cantidad de herramientas diseñadas para disminuir el ruido y errores en dispositivos cuánticos. Esta plataforma es esencial para la corrección de errores. Por último, dentro de Aqua se cuenta con la implementación de diversos algoritmos cuánticos que pueden correr sobre el hardware cuántico.

3.2.3.2. Cirq

Cirq [Google Quantum AI, 2018]: Cirq es un proyecto de código abierto desarrollado por Google, el cual tiene como objetivo la programación de computadoras cuánticas. La plataforma permite a los usuarios diseñar, simular y ejecutar algoritmos cuánticos en computadoras cuánticas y simuladores. Cirq fue creado para facilitar la investigación y el desarrollo en el campo de la Computación Cuántica, especialmente en el contexto de las arquitecturas de computadoras cuánticas superconductoras, que es el enfoque que Google ha adoptado para su hardware cuántico, en la siguiente sección se introducirán los diferentes enfoques existentes para el hardware cuántico.

Luego de IBM, Google lanzó Cirq a comienzos de 2018 como parte de su iniciativa de Investigación Cuántica. El objetivo principal de Cirq era proporcionar a los investigadores y desarrolladores una herramienta para experimentar con algoritmos cuánticos y optimizarlos para el hardware cuántico específico de Google.

Cirq se centra en la optimización de circuitos cuánticos y la asignación de qubits, dos aspectos cruciales para la ejecución eficiente de algoritmos cuánticos en hardware real. La plataforma también ofrece herramientas para simular ruido y errores en el hardware cuántico, lo cual es esencial para desarrollar técnicas de corrección de errores y comprender mejor el comportamiento de los algoritmos cuánticos en entornos reales.

Desde su lanzamiento, Cirq ha sido adoptado por una amplia comunidad de investigadores y desarrolladores, tanto en la academia como en la industria. La plataforma ha sido utilizada en numerosos proyectos de investigación y desarrollo, incluido el hito de Supremacía Cuántica alcanzado por Google en 2019 utilizando su procesador cuántico de 53 qubits llamado Sycamore.

3.2.3.3. QuTiP

QuTiP (Quantum Toolbox in Python)[QuTiP Developers, 2011]: es una biblioteca de código abierto desarrollada en Python para simular y analizar sistemas cuánticos y algoritmos cuánticos. QuTiP fue creado inicialmente por Robert J. Johansson y Paul D. Nation en 2011 y hasta el momento ha continuado en desarrollo y mantenido activo.

La plataforma QuTiP se centra en la simulación y el análisis de sistemas cuánticos abiertos y cerrados, lo que la distingue de otros marcos de trabajo y bibliotecas cuánticas que se centran principalmente en la programación de circuitos cuánticos y computadoras cuánticas.

QuTiP es ampliamente utilizado en la investigación académica y en la industria para estudiar una variedad de sistemas cuánticos. La biblioteca proporciona una amplia gama de funcionalidades, como la generación de operadores y estados cuánticos, la resolución de ecuaciones diferenciales

parciales, la simulación de sistemas cuánticos abiertos y cerrados y la visualización de resultados.

3.2.3.4. Q-sharp

$Q\#$ (Q-sharp)[Microsoft Quantum, 2018]: es un lenguaje de programación cuántica de alto nivel desarrollado por Microsoft como parte de su iniciativa Quantum Development Kit (QDK). $Q\#$ fue anunciado por primera vez en 2017 y se lanzó oficialmente en 2018. Está diseñado específicamente para la programación de computadoras cuánticas y la implementación de algoritmos cuánticos.

El enfoque principal de $Q\#$ es proporcionar un lenguaje de programación cuántica que sea familiar para los desarrolladores que ya trabajan con lenguajes de programación clásicos, como $C\#$ y Python. $Q\#$ utiliza una sintaxis similar a la de estos lenguajes, lo que facilita la adopción y el aprendizaje del lenguaje por parte de los desarrolladores.

$Q\#$ forma parte del Quantum Development Kit (QDK) de Microsoft, que también incluye herramientas de desarrollo, bibliotecas y simuladores cuánticos. El QDK permite a los desarrolladores escribir programas cuánticos utilizando $Q\#$ y ejecutarlos en simuladores cuánticos locales o en la nube. Además, el QDK proporciona integración con lenguajes de programación clásicos como Python y $C\#$, lo que permite a los desarrolladores combinar cómputos cuánticos y clásicos en sus aplicaciones.

Microsoft ha estado invirtiendo en investigación y desarrollo en Computación Cuántica durante varios años, y $Q\#$ es un componente clave de su estrategia en este campo. La compañía ha estado trabajando en el desarrollo de hardware cuántico basado en qubits topológicos, que teóricamente ofrecen una mayor tolerancia a errores y estabilidad en comparación con otros tipos de qubits.

3.2.3.5. pyQuil

pyQuil [Rigetti Computing, b], desarrollada por Rigetti Computing [Rigetti Computing, a], está diseñada específicamente para interactuar con los ordenadores cuánticos de Rigetti. La singularidad de pyQuil yace en su lenguaje de instrucciones cuánticas, Quil (Quantum Instruction Language) [Farhi and Neven, 2016], el cual permite una especificación detallada y flexible de los algoritmos cuánticos para su ejecución en procesadores cuánticos.

Quil es un lenguaje ensamblador con capacidad de control de flujo clásico, permitiendo operaciones cuánticas que pueden ser condicionales a estados clásicos. Además, se apoya en simuladores como el QVM (Quantum Virtual Machine) y compiladores que ayudan en la ejecución eficiente de estos programas en el hardware de Rigetti.

pyQuil dispone de una API de alto nivel que simplifica la creación de programas cuánticos. El soporte para herramientas de simulación local y la integración con bibliotecas de Python para ciencia de datos y machine learning, como NumPy y SciPy, le confieren una gran versatilidad.

3.2.4. Otros frameworks y librerías

3.2.4.1. Strawberry fields

Strawberry Fields [Killoran et al., 2019] es un framework de software creado por Xanadu Quantum Technologies Inc., destinado a la Computación Cuántica fotónica. Este framework se distingue por su enfoque en la simulación y ejecución de programas cuánticos que se basan en el Estado

del Arte de la óptica cuántica. La plataforma ofrece herramientas especializadas para diseñar, simular y optimizar estados de luz cuántica, y se integra con sistemas de fotónica cuántica reales, proporcionando así un puente entre la teoría y la implementación práctica.

Strawberry Fields utiliza el lenguaje de programación Python y se apoya en la biblioteca TensorFlow de Google para permitir la simulación clásica de sistemas cuánticos fotónicos y la optimización de circuitos cuánticos mediante técnicas de aprendizaje automático. Su capacidad de integración con TensorFlow no solo habilita la simulación de estados cuánticos, sino que también aprovecha los algoritmos de aprendizaje profundo para el diseño de experimentos y la mejora de protocolos en la Computación Cuántica.

3.2.4.2. Pennylane

PennyLane [Bergholm et al., 2022] es un framework de software de código abierto, específicamente diseñado para la Computación Cuántica y optimizado para el aprendizaje automático (machine learning). Desarrollado por Xanadu Quantum Technologies Inc., se destaca por permitir la construcción y optimización de circuitos cuánticos en un entorno que promueve la integración con herramientas clásicas de aprendizaje automático, ofreciendo así una plataforma híbrida cuántico-clásica. La elección de este framework para el desarrollo de aplicaciones cuánticas se justifica por su capacidad para calcular gradientes de forma analítica utilizando la técnica del "parameter-shift rule", lo que lo hace especialmente útil en algoritmos de optimización y aprendizaje.

Su arquitectura se distingue por su enfoque modular, lo que permite una integración sencilla con otras bibliotecas de Computación Cuántica y clásica. Con compatibilidad con múltiples backends, brinda a los usuarios la flexibilidad de ejecutar sus circuitos en simuladores cuánticos o en hardware cuántico real de diversos proveedores. Además, PennyLane introduce un conjunto de herramientas específicas para el aprendizaje automático cuántico (QML), lo que incluye una variedad de templates de circuitos, funciones de costo y optimizadores cuánticos, facilitando así el diseño y experimentación con modelos de QML.

Una de las contribuciones más significativas de PennyLane es su capacidad para diferenciar automáticamente circuitos cuánticos. Esta funcionalidad es crítica para aplicaciones en las que se requiere la optimización de parámetros, siendo esencial en la implementación de algoritmos de aprendizaje profundo cuántico. La diferenciación se realiza mediante el uso de la regla de cambio de parámetros (parameter-shift rule), que permite calcular el gradiente de la expectativa de un observable con respecto a los parámetros del circuito. El framework aprovecha estos gradientes para aplicar técnicas de optimización robustas en la búsqueda de mínimos en funciones de coste, una pieza clave en la convergencia de algoritmos de aprendizaje.

La documentación extensa y los tutoriales interactivos facilitan la entrada de nuevos usuarios y desarrolladores en el campo de la Computación Cuántica. Además, el equipo de PennyLane colabora estrechamente con investigadores y desarrolladores para asegurar que el framework se mantenga a la vanguardia de las necesidades científicas y tecnológicas, añadiendo constantemente nuevas funcionalidades y mejorando su rendimiento y usabilidad.

Mirando hacia el futuro, PennyLane está posicionado para ser un catalizador en la investigación y desarrollo de aplicaciones cuántico-clásicas, especialmente en la interfaz con aprendizaje automático. Los desafíos que enfrenta incluyen mantener su relevancia ante la rápida evolución de la tecnología cuántica y la emergencia de nuevos paradigmas computacionales. Además, existe el desafío constante de optimizar la escalabilidad y eficiencia de los algoritmos en hardware cuántico, que aún está en sus fases de desarrollo.

3.2.4.3. D-Wave Ocean

D-Wave Ocean [D-Wave Systems, c] es un conjunto integral de herramientas de software diseñadas para facilitar el desarrollo de aplicaciones que se ejecutan en los sistemas de Computación Cuántica de D-Wave Systems Inc [D-Wave Systems, a]. Este ecosistema de herramientas permite a los usuarios formular problemas de optimización y muestreo para ser procesados en los procesadores cuánticos de D-Wave, los cuales son especialmente aptos para resolver problemas complejos de optimización combinatoria y de búsqueda de mínimo global. La accesibilidad y capacidad de este conjunto de herramientas han catalizado la adopción de la tecnología de D-Wave en sectores industriales y de investigación que buscan explotar el potencial de la optimización cuántica [D-Wave Systems, b].

Ocean se destaca por su arquitectura modular, que comprende una serie de paquetes de Python que abarcan desde la construcción de modelos matemáticos hasta la comunicación directa con el hardware cuántico. La interoperabilidad de Ocean con Python permite a los desarrolladores e investigadores incorporar soluciones cuánticas en sus aplicaciones existentes, aprovechando el amplio ecosistema de paquetes científicos y de análisis de datos de Python.

La característica principal de Ocean es su enfoque en la resolución de problemas de optimización. Los usuarios pueden construir modelos matemáticos correspondientes a su problema de optimización y utilizar las herramientas de Ocean para traducir estos modelos en formatos que sean nativos a la computadora cuántica de D-Wave. La capacidad de resolver estos problemas utilizando Computación Cuántica ofrece una alternativa prometedora a los métodos clásicos de optimización, particularmente en instancias donde la complejidad del problema escala exponencialmente con el tamaño del input.

3.2.5. Tabla de compatibilidad

Cuadro 3.1: Compatibilidad de frameworks y backends cuánticos

Framework / Backend	**IBM Quantum**	**IonQ**	**Rigetti**	**D-Wave**	**Honeywell**	**Oxford Quantum Circuits**	**Xanadu**	**QuEra**
Qiskit	X	X	X		X			
Cirq	X	X						
PyQuil			X					
Ocean				X				
Q#	X	X	X	X	X	X	X	X
Strawberry Fields							X	
ProjectQ	X	X	X	X				

Continúa en la siguiente página

Cuadro 3.1 – continúa de la página anterior

Framework / Backend	IBM Quantum	IonQ	Rigetti	D-Wave	Honeywell	Oxford Quantum Circuits	Xanadu	QuEra
PennyLane	X	X	X		X		X	X

3.2.6. Corrección de errores

Al igual que sucede en paradigmas clásicos, en Computación Cuántica también son utilizados los conceptos de corrección y detección de errores [Yanofsky and Mannucci, 2008]. Mediante redundancia se logra agregar información a un mensaje de tal forma que de producirse una alteración en algunos de los qubits, esto sea identificado y hasta solucionado con algunos de los métodos.

Se analiza un ejemplo sencillo para introducir este concepto. Si se desea mandar un bit desde un lugar a otro por un canal con ruido que cambia el estado del bit con una probabilidad p y por ende lo transfiere sin cambios con una probabilidad $1 - p$. Siguiendo la estrategia de redundancia, lo que se puede hacer es agregar más bits en este mensaje. Si se quiere mandar un bit con el valor 1 por este medio, entonces se triplica la información. De esta forma al enviar un 1 se estará enviando 111 y al enviar un 0 se estará enviando 000. Esta estrategia agrega robustez a los mensajes, ya que al recibir este mensaje en el destino se decidirá por mayoría cuál fue el bit enviado originalmente. De esta forma se transforma la probabilidad de cometer errores de p a $3p^2 - 2p^3$. Siendo un canal más fiable cuando $p < 1/2$.

Esta estrategia de corrección de errores es conocida como códigos de repetición, ya que al construir el mensaje se repite múltiples veces el valor para lograr la redundancia. Es una de las estrategias más sencillas, pero ayuda a ver la intuición bajo las técnicas que se verán en esta sección.

Cuando se observan estos fenómenos en el área de la cuántica, se ve que no se comportan de igual manera y las estrategias similares a las definidas previamente ya no funcionan en este sector debido a los siguientes problemas:

No existe un mecanismo de clonado: en este caso no se puede duplicar un qubit como se hizo en la estrategia de repetición, aunque fuera posible se tendría otra limitante, no se puede medir todos los qubits y compararlos en la salida del canal.

Errores son continuos: a diferencia de lo que pasaba en el mundo clásico, que los errores son discretos (cambio de estado o no). En la Computación Cuántica los errores se consideran continuos y pueden ocurrir varios en cada qubit. Esto hace que se requiera unos niveles de precisión infinitos para su corrección.

Mediciones destruyen información: en el mundo clásico se puede medir el estado de cada bit y en base a estas mediciones decidir si hubo errores o no. En el mundo cuántico, la medición hace que un qubit colapse, transformándose en 0 o 1, sin la capacidad de volver a recuperar la información del qubit.

Si se analiza el problema introducido anteriormente de transmisión de tres bits sobre un canal con ruido. En este caso se observa que pasa en la cuántica si se tiene un canal que puede alterar a lo sumo un qubit y se quiere identificar y corregir este error.

En el caso en que se tenga el siguiente qubit a transmitir $a\,|0\rangle + b\,|1\rangle$ que se transforma en $a\,|000\rangle + b\,|111\rangle$ antes de ser transmitido.

Ahora si por el canal donde se transfiere la información se modifica uno de los qubits. Mediante el siguiente procedimiento de dos etapas se va a detectar y corregir este error.

Suponiendo que hubo un flip en el primer bit, al realizar una medición conocida como diagnóstico de síndrome se obtiene la posición que se modificó, en este caso retorna 1 luego de procesar el siguiente mensaje: $a\,|100\rangle + b\,|011\rangle$.

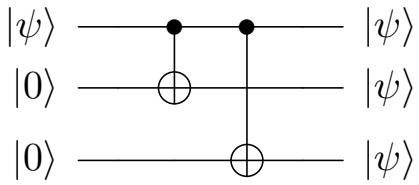
La ventaja de esta medición es que no causó ninguna modificación al estado interno del qubit contenido en a y b.

Para recuperar el estado original y corregir este error, luego de la medición se aplica un cambio de estado al bit que indica la medición, con esto se logra corregir errores de hasta un bit con 100 % de precisión.

Al igual que el método de corrección de errores con repetición para la Computación Clásica, en cuántica se tiene que la probabilidad de que se produzcan errores bajó de p a $3p^2 - 2p^3$. Siendo un canal más fiable cuando $p < 1/2$.

Se puede observar la construcción de un circuito para la implementación de este mecanismo de corrección de errores.

Primero se comienza con el mecanismo de repetición, dado el qubit a ser transmitido por el canal se necesita repetir su estado tres veces. Para esto se utilizan compuertas CNOT de la siguiente manera.



Como vimos, cuando se introdujo la compuerta CNOT, si el bit de control vale 1 entonces el bit restante cambiará de estado y, por lo contrario, si el bit de control vale 0 se mantiene el mismo estado en el bit restante.

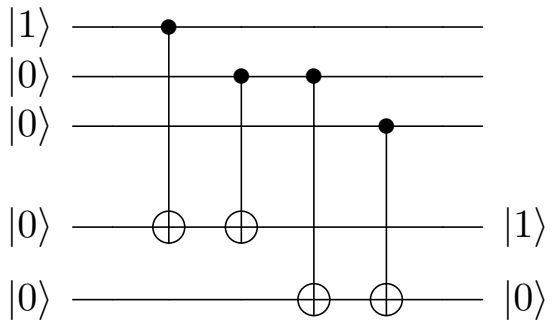
En este caso el bit de control es el bit que se desea repetir, por ende si este vale 1 se quiere repetir tres veces este valor. Mediante la entrada de 0 en la otra entrada de la compuerta se cambiará este estado consiguiendo como salida el valor 1 buscado, con esto se consigue la salida 111 buscada.

Análogamente, ocurre lo mismo con la entrada 0, los bits restantes se mantienen intactos obteniendo como salida el valor 000.

Luego de codificar la información que se quiere transmitir, se procede a enviar por el canal con ruido que modificará con probabilidad p uno de los bits.

En este momento se tiene un mensaje potencialmente con fallos.

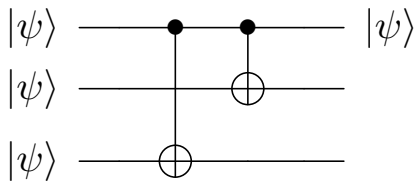
En caso de que llegue el mensaje con el siguiente valor 100, aquí se busca encontrar que el primer bit fue el que se modificó y corregirlo. Para esto se debe de pasar este mensaje por un circuito llamado detección de síndromes, el cual se representa en la siguiente figura.



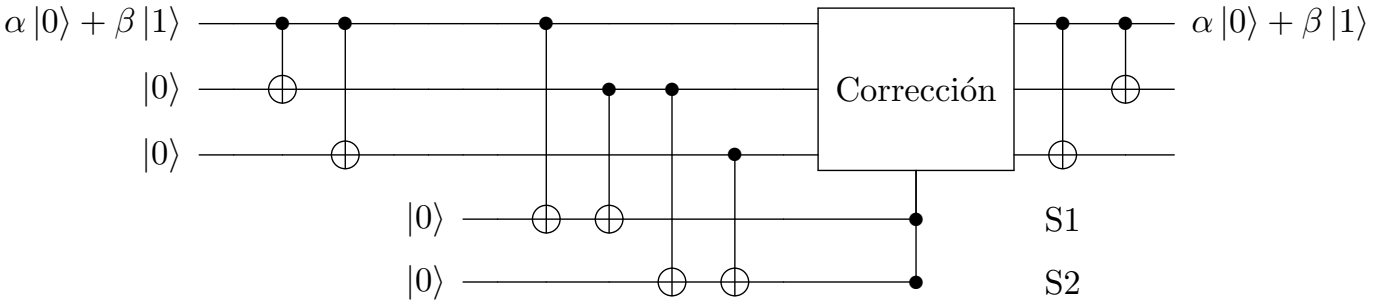
Con esto se logra un circuito que codifica el bit que está alterado de un mensaje, con este código se puede generar una tabla para cada uno de los cuatro posibles resultados y con esto corregir el bit que falló.

S1	S2	Corrección
0	0	$1 \oplus 1 \oplus 1$
0	1	$1 \oplus 1 \oplus X$
1	0	$X \oplus 1 \oplus 1$
1	1	$1 \oplus X \oplus 1$

Con el resultado del circuito de detección de síndromes, en este caso 10, la tabla indica que se debe de cambiar el estado del primer bit. Con esto se logra recuperar el mensaje que inicialmente fue transmitido por el canal 000, ahora solo falta el circuito para decodificar este mensaje y quitarle la repetición. Para esta tarea solo se deben aplicar los operadores que se utilizaron para generar la repetición en orden inverso.



Juntando todas las partes, se logra el siguiente circuito:



Donde la tabla de corrección es la que se construyó anteriormente.

S1	S2	Corrección
0	0	$1 \oplus 1 \oplus 1$
0	1	$1 \oplus 1 \oplus X$
1	0	$X \oplus 1 \oplus 1$
1	1	$1 \oplus X \oplus 1$

Con este sencillo circuito se logra implementar un mecanismo de corrección de errores cuántico.

3.3. Aprendizaje no supervisado

Dentro del área del aprendizaje automático, se pueden encontrar dos grandes grupos de algoritmos. Por un lado, se encuentran los algoritmos supervisados, que son entrenados con datos ya etiquetados. Es decir, datos cuya clasificación es conocida. Por ejemplo, un algoritmo que es entrenado con un conjunto de textos clasificados y el sentimiento asociado a estos para eventualmente poder predecir el sentimiento de textos sin clasificar.

Por otro lado, se podría entrenar un algoritmo únicamente con textos e intentar obtener diferentes agrupaciones, utilizando alguna "distancia" definida entre ellos. El resultado de la clasificación/agrupación dependerá de la representación de los textos y la distancia definida. Por ejemplo, si la representación es la cantidad de palabras de un texto, y la distancia es definida como la diferencia entre estas cantidades, entonces el algoritmo agrupará los textos por largo de palabras. Por otro lado, si se tuvieran dos diccionarios de palabras, uno de positivas y uno de negativas, se podría definir la representación del texto como un vector de dos dimensiones, siendo la primera la cantidad de palabras del diccionario positivo que se encuentran en el texto, y la segunda, análogamente, las del diccionario negativo. Definiendo la distancia entre vectores como la distancia euclídea, se podría suponer (capaz erróneamente) que los textos serán agrupados por el sentimiento general que expresan. A este tipo de algoritmos se los conoce como algoritmos no supervisados.

En particular, el ejemplo anterior es un ejemplo de un problema de "Clustering", considerado como la cuestión más importante del aprendizaje no supervisado [Xu and Wunsch, 2005].

3.4. Clustering

Clustering consiste en agrupar objetos en grupos con características similares (clústeres). Generalmente, se tiene el objetivo de identificar patrones, aunque se utilizan para diversas tareas. Algunas de ellas son:

- Segmentación del mercado.
- Análisis de redes sociales.
- Agrupación de resultados de búsqueda.
- Segmentación de imágenes.
- Diagnósticos médicos.
- Detección de anomalías.

Después de agrupar, se asigna a cada clúster un ID. Representar todo el clúster mediante simplemente un ID hace que la agrupación sea muy valiosa para unificar grandes conjuntos de datos. Los sistemas de Inteligencia Artificial pueden usar los IDs de los clústeres para simplificar el procesamiento de grandes datasets.

En Google, la agrupación se utiliza con distintos motivos como la generalización, la compresión de datos y la preservación de la privacidad en variados productos. En cuanto a la generalización, si faltan determinados datos sobre un ejemplo, se pueden inferir a través de otros ejemplos pertenecientes al mismo clúster. Siguiendo por la compresión de datos, se puede ver que al tener los datos agrupados e identificados mediante el ID del clúster, se está reduciendo la complejidad de los datos de entrada y, por lo tanto, el modelo de Inteligencia Artificial va a ser más sencillo y rápido de entrenar. También se preserva la privacidad agrupando a los usuarios y asociando los datos de usuario a los ID de clúster en vez de usuarios en específico. [Google Team, 2022]

3.4.1. Clustering particional

En este caso, se busca analizar la agrupación de particiones. Las mismas se basan en construir el primer grupo y reasignar repetidamente los objetos hasta la convergencia. Por lo general, en este tipo de agrupación, todos los clústeres se determinan al mismo tiempo.

3.4.2. k-means

k-means es un problema de optimización que dado un conjunto de puntos de datos en un espacio vectorial, hay que intentar ubicar k puntos (centroides) en lugares que minimicen la distancia al cuadrado entre cada punto y el centro más cercano.

Cada clúster está representado por un punto en el espacio (llamado centroide). Un punto va a pertenecer al clúster C_i si el centroide c_i es el centroide más cercano.

Pseudocódigo:

Algorithm 1 k-means

Require: S (dataset), K (número de clústeres)
Ensure: *clústeres*
  Inicializar K centroides de forma aleatoria
  while no se de la condición de fin **do**
    Asignar cada instancia al centroide más cercano
    Recalcular los centroides
  end while
  return *clústeres*

Primero se inicializa y luego se va reduciendo el error. Se finaliza cuando la distancia entre los centroides nuevos y los anteriores es menor a un cierto ϵ.

La convergencia de k-means está garantizada (Selim and Ismail, 1984). [Docentes de Aprendizaje Automatico - Fin

Para probar la convergencia en una cantidad finita de pasos hay que ver primero que hay k^N maneras de particionar N puntos en k clústeres, ya que cada punto tiene k posibilidades. Esto puede ser un número grande, pero es finito. Para cada iteración del algoritmo se produce un nuevo Clustering basado solamente en el anterior. Acá hay 2 cosas para notar. La primera (1) es que si el nuevo clúster es el mismo que el anterior, entonces el siguiente Clustering va a ser otra vez el mismo.

La segunda (2), es que si el nuevo Clustering es diferente que el anterior, entonces este último va a tener un menor costo. Como el algoritmo itera sobre una función cuyo dominio es un conjunto finito, entonces la iteración eventualmente entra en un ciclo. El ciclo no puede tener un largo mayor a 1 porque sino por (2) se tendría un Clustering cuyo costo es más bajo que el mismo, lo cual es imposible. Por lo tanto, el ciclo debe tener largo exactamente 1. Por esto se deduce que k-means converge en un número de iteraciones finito.

3.4.2.1. Aceleración del algoritmo de Lloyd para k-means

El algoritmo de k-means, que es un elemento básico de la minería de datos y el aprendizaje no supervisado [Celebi, 2014], es popular porque es fácil de implementar, rápido, fácilmente paralelizable y ofrece resultados intuitivos. El algoritmo de Lloyd es el enfoque estándar de minimización de la función de optimización k-means. Este algoritmo dedica la mayor parte de su tiempo a calcular las distancias entre los k centros de clúster y los n puntos de datos. Sin embargo, se ha descubierto que gran parte de este trabajo es innecesario, ya que los puntos suelen permanecer en los mismos clústeres después de las primeras iteraciones. En la última década, los investigadores han desarrollado una serie de optimizaciones para acelerar el algoritmo de Lloyd tanto para datos de baja como de alta dimensión.

A continuación, se verán algunas optimizaciones y se presentarán otras nuevas, basándose en el libro Partitional Clustering Algorithms [Celebi, 2014]. Se van a ver aquellas optimizaciones nuevas que evitan los cálculos de distancia mediante la desigualdad del triángulo. Al almacenar en caché las distancias conocidas y actualizarlas eficientemente con la desigualdad del triángulo, estos algoritmos pueden evitar muchos cálculos de distancia innecesarios. Todas las optimizaciones examinadas producen los mismos resultados que el algoritmo de Lloyd dado el mismo conjunto de datos y la misma inicialización, por lo que son adecuados como reemplazos. Estos nuevos algoritmos pueden ser mucho más rápidos y calcular muchas menos distancias que la implementación estándar sin optimizar. En los experimentos que muestra el libro, se ven aumentos de velocidad de más de 30-50 veces en comparación con el algoritmo de Lloyd. Se estudiarán los trade-offs de estos métodos respecto a la cantidad de puntos a agrupar (n), dimensiones (d), clústeres (k) y la estructura de datos.

En Lloyd se desperdicia mucho tiempo procesando cálculos innecesarios y redundantes y, por lo tanto, se verán algoritmos más eficientes que dan la misma salida que el algoritmo estándar de Lloyd.

La razón básica por la que los métodos de lotes estándar para optimizar k-means son ineficientes es porque en cada iteración deben identificar el centro más cercano para cada punto agrupado y para esto, se calculan absolutamente todas las distancias (nk) entre cada uno de los n puntos agrupados y cada uno de los k centros. Después de cada iteración, los centros se mueven y estas distancias pueden cambiar, lo que requiere un nuevo cálculo. Pero normalmente, los centros no se mueven mucho, especialmente después de las primeras iteraciones. La mayor parte del tiempo, el centro más cercano en la iteración anterior sigue siendo el centro más cercano. Por lo tanto, llevar un registro del centro más cercano para cada punto agrupado es mucho más eficiente con algún almacenamiento en caché. Cuando el centro más cercano no cambia, en el caso ideal, no se debería necesitar calcular la distancia entre ese punto y ningún centro del clúster. Si el centro más cercano para un punto cambia, también puede ser posible evitar calcular la distancia desde ese punto a todos los centros, mirando solamente algunos centros que se garantiza que están más cerca de ese punto que todos los demás centros.

Hay varios métodos para acelerar el algoritmo, y los principales son: mejoras algorítmicas, paralelización (incluyendo threading, multiprocessing y cómputo distribuido) y aproximación. A continuación se tratarán mejoras algorítmicas, ya que dan respuestas exactas, mejorando el algoritmo en sí mismo. Luego, en la parte experimental se usarán también las técnicas de paralelización y aproximación.

3.4.2.2. Distorsión de clústeres y algoritmos de Lloyd.

En el Clustering con k-means, se busca agrupar los puntos en clústeres y minimizar la "distorsión", que es la suma de los errores al cuadrado entre los puntos y sus centros asignados. Se puede mejorar la distorsión de dos formas: cambiando la asignación de clústeres de los puntos y moviendo los centros de los clústeres. Es decir, con un conjunto fijo de puntos X, se intenta encontrar la mejor forma de agruparlos en clústeres para minimizar la función de distorsión.

Nombre y tipo	Descripción
$d \in \mathbb{N}$	Dimensión de los puntos
$n \in \mathbb{N}$	Cantidad de puntos a agrupar
$k \in \mathbb{N}$	Cantidad de centroides
$X \subset \mathbb{R}^{n*d}$	Conjunto de puntos a agrupar
$C \subset \mathbb{R}^{n*d}$	Conjunto de centroides
$n(j) \in \mathbb{R}$	Cantidad de puntos que están actualmente asignados al clúster j
$N = \{i \in \mathbb{N} \mid 1 <= i <= n\}$	Índices de los puntos en X
$K = \{j \in \mathbb{N} \mid 1 <= j <= k\}$	Índices de los puntos en C
$a : N \to K$	Índice del centroide asignado para cada punto
$u : N \to \mathbb{R}$	Límite superior de la distancia entre cada punto y su centro asignado.
$l : N$ x $K \to \mathbb{R}$	Límite inferior de la distancia entre cada punto y su centro asignado.
$s : K \to \mathbb{R}$	La mitad de la distancia entre un centro y su actual centro más cercano.

$J(X, C) = \sum_{i \epsilon N} \|x(i) - c(a(i))\|^2$

El algoritmo de Lloyd [2] para minimizar la distorsión consta de tres pasos básicos, que se indican aquí (luego se explicarán con más detalle):

1. Inicializar los centros.
2. Hasta que el algoritmo converja:
   a. Asignar cada punto al centro de clúster más cercano en ese momento.
   b. Mover cada centro a la media de sus centros asignados en ese momento.

El Paso 1 ocurre solo una vez, mientras que los Pasos 2(a) y 2(b) alternan hasta que el algoritmo converge. La convergencia está garantizada debido a que los Pasos 2(a) y 2(b) reducen J(X, C) y hay un número finito de formas de dividir los n puntos entre k clústeres. El paso 2(a) de encontrar

el centro más cercano para cada punto es el que se puede optimizar mediante caché y el uso de geometría.

En cuanto a la inicialización de los centros para el algoritmo k-means, el método más efectivo actualmente tanto en la teoría como en la práctica, es la inicialización k-means++. [Arthur and Vassilvitskii, 2007]

Algorithm 2 Algoritmo Lloyd para k-means (algoritmo estándar para minimizar J(X,C))

```
procedure LLOYD(X, C)
    while No converge do
        for all i ∈ N do                          ▷ Encontrar el centro más cercano a cada x(i)
            a(i) ← 1
            for all j ∈ K do
                if || x(i) − c(j)|| < || x(i) − c(a(i)|| then
                    a(i) ← j
                end if
            end for
        end for
        for all j ∈ K do                                          ▷ Mover los centros
            Mover c(j) a la media de {x(i)|a(i) = j}
        end for
    end while
end procedure
```

El paso 2(b) que corresponde al último for, puede optimizarse mediante el almacenamiento en caché de estadísticas suficientes para cada clúster: la suma de vectores de los puntos asignados al clúster y el número de puntos asignados al clúster. Mantener estos datos es económico y evita una suma sobre todos los puntos en cada iteración. Cada vez que un punto cambia su pertenencia a un clúster, se actualizan las estadísticas correspondientes. Después de las primeras iteraciones, la mayoría de los puntos permanecen en el mismo clúster durante muchas iteraciones. Por lo tanto, estas actualizaciones de estadísticas suficientes se vuelven mucho más económicas que una suma sobre todos los puntos.

3.4.2.3. Análisis del algoritmo de Lloyd

El tiempo de ejecución del algoritmo de Lloyd para k-means es $O(wnkd)$ para w iteraciones, k centros y n puntos en d dimensiones. Para un conjunto de datos fijo y k, el número de iteraciones w variará dependiendo de la inicialización. De hecho, w puede ser superlineal con respecto a n, incluso exponencial en el peor caso. Sin embargo, al considerar casos menos extremos, el algoritmo de Lloyd tiene una complejidad polinómica suavizada [Arthur et al., 2011]. Es decir, w es polinómico en n y $1/\sigma$ (donde σ es la cantidad de perturbación permitida en el conjunto de datos).

3.4.2.4. Algoritmo MacQueen

Otro método estándar, pero menos usado [Celebi, 2014], para resolver los problemas de optimización del k-means es el algoritmo MacQueen. Este algoritmo es parecido al de Lloyd, pero actualiza la ubicación de cada centro de clúster afectado cada vez que un punto cambia de pertenencia al clúster.

Algorithm	1	2	3	4	5	6
Lloyd						
Compare-means	X					
Soft-means		X			X	
Elkan	X	X	X	k		
Hamerly	X		X	1		
Drake	X		X	b <k		
Annular	X		X	1		X
Heap	X		X	0		

Cuadro 3.2: Esta tabla muestra diferentes métodos de aceleración que utilizan límites de distancia, ordenamiento y la desigualdad triangular en varios algoritmos k-means.

El algoritmo de Lloyd mueve los centros una vez por pasada sobre el conjunto de datos completo, mientras que el algoritmo de MacQueen moverá los centros (en cantidades más pequeñas) con mucha más frecuencia.

El algoritmo de Lloyd es más fácil de acelerar, ya que los centros se mueven con menos frecuencia.

3.4.2.5. Enfoques de la desigualdad triangular

La desigualdad triangular es una herramienta simple pero muy poderosa de la geometría. Si $a, b, c \in \mathbb{R}$ entonces la desigualdad triangular establece que $||\mathbf{a} - \mathbf{c}|| \leq ||\mathbf{a} - \mathbf{b}|| + ||\mathbf{b} - \mathbf{c}||$ para la norma de vector euclidiana $||\sqrt{a^T a}||$.

De manera intuitiva, esto significa que la longitud del segmento de línea (a,c) es como máximo la suma de las longitudes de los segmentos de línea (a,b) y (b,c). En otras palabras, la ruta más corta entre dos puntos a y c es una línea recta (tomar una ruta que pase por un punto intermedio b no puede reducir la distancia).

La desigualdad triangular es aplicable en el algoritmo k-means de múltiples maneras.

Generalmente, se desea usarla para demostrar que algún centro debe estar más cerca de un punto que todos los demás centros. Idealmente, se busca hacer esto con la menor cantidad de cálculo posible. Para un punto x y dos centros c y c', aquí hay algunas de las diferentes formas en que se puede utilizar la desigualdad triangular en k-means:

1) Para probar que c' está más cerca de x que de c, dado solo las distancias $||\mathbf{c}' - \mathbf{x}||$ y $||\mathbf{c}' - \mathbf{c}||$

2) Para probar que c' está más cerca de x que de c, dado solo las normas $||x||$ y $||c||$ y la distancia $||\mathbf{x} - \mathbf{c}'||$

3) Mantener un límite superior en $||\mathbf{x} - \mathbf{c}||$ cuando c se mueve.

4) Mantener un límite inferior en $||\mathbf{x} - \mathbf{c}||$ cuando c se mueve.

La Tabla 3.2 muestra diferentes métodos de aceleración que utilizan límites de distancia, ordenamiento y la desigualdad triangular en varios algoritmos de k-means.

Significados de las columnas:

1. Distancia del centro a su otro centro más cercano
2. Distancia desde el centro a todos los otros centros
3. Límite superior en las distancias centro-punto.
4. Límite inferior en las distancias centro-punto.
5. Para cada centro, ordenar los centros según distancias.

6. Ordenar los centros según su norma.

Estas alternativas tienen como objetivo proporcionar exactamente la misma respuesta que el de Lloyd (dada la misma inicialización), pero de manera más rápida. Todos ellos se basan en la desigualdad geométrica del triángulo para evitar cálculos de distancia innecesarios y costosos. Esto puede proporcionar aceleraciones dramáticas de hasta 40x, según las pruebas que se ven en el informe del Estado del Arte. Allí se pueden encontrar las alternativas con más detalle.

3.4.3. DBSCAN

DBSCAN es posiblemente el algoritmo de Clustering basado en densidad más importante en la actualidad [Ester et al., 1996]. Los puntos se categorizan en puntos centrales, también conocidos como core points, que tienen muchos puntos vecinos en su vecindario directo, puntos frontera o border points, que se encuentran en el vecindario de al menos un punto central, y puntos de ruido o noise points, que no pertenecen a ninguno de los dos primeros grupos.

La siguiente parte describe el algoritmo DBSCAN con más detalle, siguiendo las definiciones y estructura originales de [Ester et al., 1996]. Primero, se necesita definir formalmente cuándo un punto debe convertirse en un punto central o core point de un grupo. Para ello, se necesita definir el término de ϵ-vecindario.

Definición 3. El ϵ-vecindario de un punto x_i es el conjunto de puntos que tienen una distancia de como máximo ϵ a ese punto:

$$\mathscr{N}_\epsilon(x_i) = \{x \in \mathscr{D} | d(x_i, x) \leq \epsilon\}$$

Luego se puede definir puntos núcleo con respecto a una medida de densidad local, es decir, el número de puntos dentro del ϵ-vecindario de un punto.

Definición 4. El conjunto de puntos centrales (core points) incluye todos los puntos cuyo ϵ-vecindario contiene al menos $minPts$ puntos adicionales:

$$\mathscr{P} = \{x \in \mathscr{D} \mid ||\mathscr{N}_\epsilon(x)|| \geq minPts\}$$

Ahora, un punto que se encuentra dentro del ϵ-vecindario de un punto central puede que no sea en sí mismo un punto central porque se encuentra en el borde de un clúster y, por lo tanto, no cumple con el criterio de densidad establecido por el parámetro $minPts$.

Definición 5. Un punto p se llama directamente alcanzable por la densidad desde un punto central x si $p \in \mathscr{N}_\epsilon(x)$.

Definición 6. Un punto p se llama alcanzable por la densidad desde un punto central x si hay una serie de puntos $p_1, p_2, ..., p_n$ con $p = p_n$ y $x = p_1$, y para cada par (p_i, p_{i+1}) se cumple que p_{i+1} es directamente alcanzable por la densidad desde p_i.

Definición 7. Dos puntos p y q se llaman densidad-conectados si hay un punto o tal que tanto p como q son alcanzables por la densidad desde o.

Con estas nociones de densidad, finalmente se pueden definir los clústeres que DBSCAN encontrará:

Definición 8. Un conjunto $C \subseteq \mathscr{D}$ se llama clúster si para todo $p, q \in C$ se cumple que p y q están densidad-conectados y no existe ningún conjunto superconjunto de C que también sea un clúster.

Cada punto que no está densamente conectado a ningún otro punto (y por lo tanto no pertenece a ningún clúster) se considera como ruido. La Figura 3.1 visualiza los diferentes conceptos de conectividad explicados anteriormente. En este ejemplo, el conjunto de puntos $C = \{p, x_1, x_2, x_3, q\}$ forma un clúster.

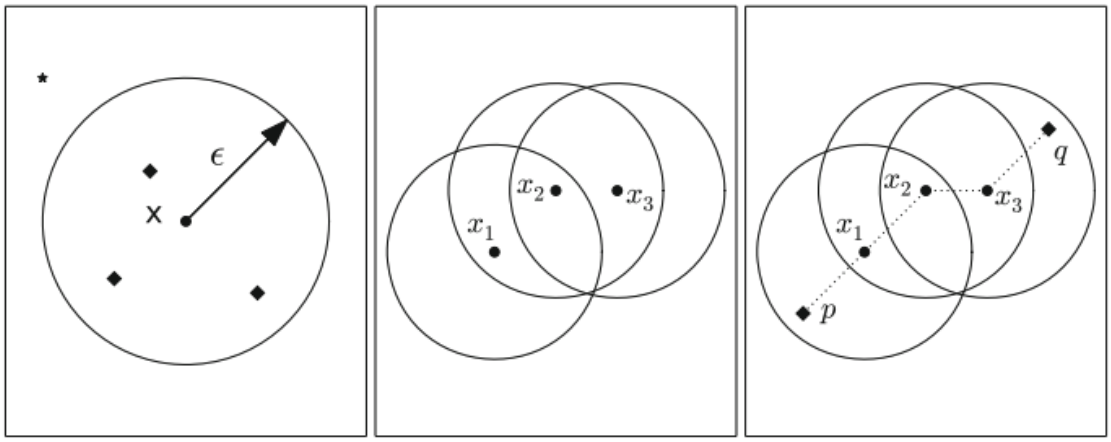


Figura 3.1: El gráfico de la izquierda muestra un punto x y su ϵ-vecindario. El punto con forma de estrella no se encuentra en $\mathscr{N}_\epsilon(x)$, mientras que los puntos con forma de diamante sí lo están. El gráfico central muestra tres puntos x_1, x_2, x_3. x_1 y x_3 son densamente alcanzables directamente desde x_2 y, por lo tanto, x_3 es densamente alcanzable desde x_1. El gráfico de la derecha muestra dos puntos adicionales p y q que están densamente conectados por x_1, x_2 y x_3.

Con esta noción de lo que realmente es un clúster, se pueden encontrar clústeres de forma casi arbitraria en datos también de alta dimensión. Incluso se pueden distinguir clústeres cuyas envolturas convexas se intersectan (que no son separables linealmente) con este algoritmo. Con la ayuda de estructuras de datos adecuadas, las consultas de región (solicitar los puntos en un cierto vecindario de un punto) pueden ser calculadas bastante rápido [Katayama and Satoh, 1997], lo que lleva a un cálculo rápido del resultado del Clustering (Figura 3.2).

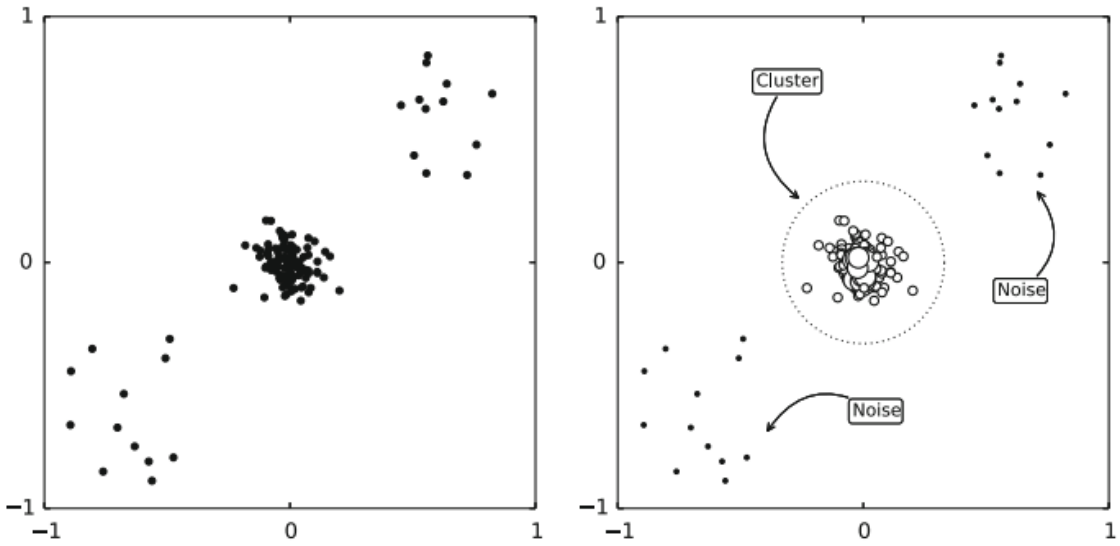


Figura 3.2: Un solo clúster con dos parches de ruido en la esquina inferior izquierda y superior derecha en la imagen de la izquierda. En la imagen de la derecha, los puntos núcleo se muestran como círculos más grandes, y los puntos no núcleo, pero conectados por densidad, se muestran como círculos más pequeños.

Sin embargo, todavía hay algunas desventajas que no deberían pasar desapercibidas en este algoritmo. En primer lugar, la elección de $minPts$ influye en gran medida en el resultado del algoritmo. Un valor demasiado bajo producirá un resultado cercano a lo que se podría haber logrado mediante Clustering Jerárquico. De hecho, el Clustering resultante es igual al del Clustering Jerárquico con un umbral de corte de ϵ si se elige $minPts = 2$. Sin embargo, si el valor de $minPts$ se elige demasiado grande, solo unos pocos puntos pueden cumplir la condición del punto central y no se formarán clústeres en absoluto. Este efecto se ilustra en la Figura 3.3.

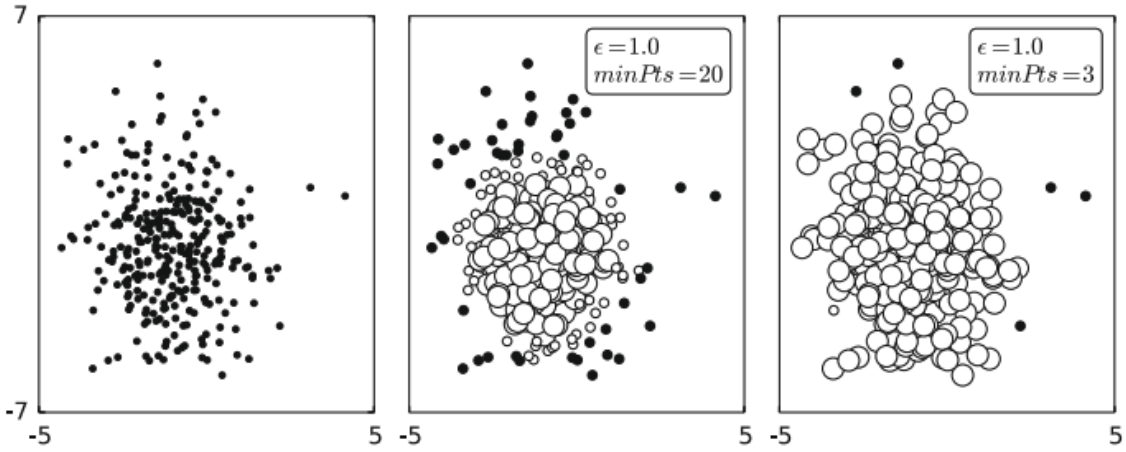


Figura 3.3: El gráfico izquierdo muestra el conjunto de datos de 300 puntos, originalmente dibujados todos desde la misma distribución gaussiana multivariada. Los otros gráficos muestran el agrupamiento con los parámetros dados. Se ha elegido un valor demasiado grande para $minPts$. Los puntos blancos pertenecen al mismo clúster, mientras que los puntos negros se consideran ruido.

El segundo parámetro - ϵ - también influye en el resultado [Esmaelnejad et al., 2010]. Debido a que se utiliza para calcular la vecindad de un punto, es crucial para determinar si un punto debe convertirse en un punto central o no. Los valores más pequeños de ϵ requieren también valores más pequeños de $minPts$, si se va a utilizar la misma densidad, pero esto puede llevar a clústeres más pequeños en general, ya que los puntos fronterizos de un clúster (aquellos que están conectados por densidad pero no son puntos centrales) pueden no ser capturados por ningún ϵ-vecindario. Estos puntos se convertirían entonces en ruido.

Por último, DBSCAN tiene una gran dificultad para diferenciar entre clústeres con diferentes densidades [Ester et al., 1996] o clústeres con densidad localmente variable. Los clústeres con densidades más altas requieren un ϵ más pequeño o un $minPts$ más pequeño, mientras que los clústeres de baja densidad necesitan que estos valores se incrementen. Si en los datos ocurren clústeres dispersos y estrechos al mismo tiempo, DBSCAN se enfrenta a una situación en la que los clústeres dispersos no serían reconocidos en absoluto o podrían fusionarse con el clúster estrecho, lo que hace imposible distinguirlos. La Figura 3.4 muestra esto para un conjunto de datos de muestra de dos clústeres generados con diferentes densidades.

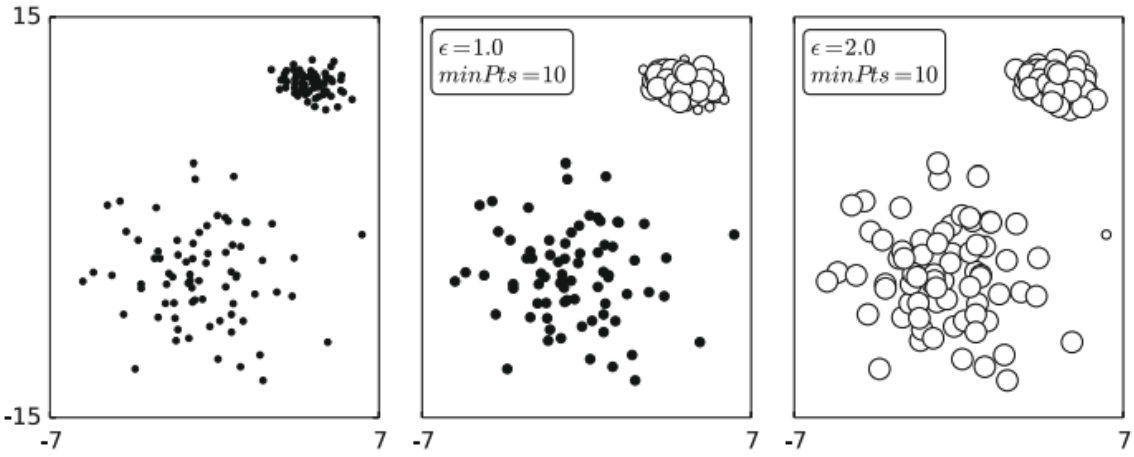


Figura 3.4: Efectos de diferentes elecciones para ϵ si están presentes agrupaciones de diferentes densidades. El gráfico de la izquierda muestra el conjunto de datos de 150 puntos, distribuidos en dos agrupaciones de diferentes densidades. Los otros gráficos muestran la agrupación con los parámetros dados. Los puntos blancos pertenecen a la misma agrupación, mientras que los puntos negros se consideran ruido.

En cuanto a todos los algoritmos de Clustering que se basan en medidas de distancia, DBSCAN no se adapta muy bien al aumento de la dimensionalidad del conjunto de datos. La maldición de la dimensionalidad es aquí solo un problema. La combinación de parámetros $minPts - \epsilon$ no escala de manera muy intuitiva con el aumento del número de atributos del conjunto de datos. Mientras que una configuración de parámetros de $(minPts, \epsilon) = (20, 2)$ significa que se espera ver aproximadamente $1,59$ puntos de datos por unidad de hipervolumen en un conjunto de datos bidimensional, la misma configuración conduce a $0,25$ puntos por unidad de hipervolumen en un conjunto de datos tan simple como aquel que solo contiene cuatro atributos.

3.5. Subrutinas cuánticas

Para la implementación de algoritmos cuánticos de Clustering, se utilizarán los algoritmos ya existentes, reemplazando subrutinas clásicas con sus equivalentes cuánticos que permitan una mejora en la complejidad del algoritmo. Muchas de las subrutinas están basadas en el algoritmo de Grover, el cual se explica en el Estado del Arte.

3.5.1. Diferencia de dos vectores

3.5.1.1. Amplitude Encoding

[Schuld and Petruccione, 2018] Al tener esta visión "vectorial" de los qubits y los estados cuánticos, sería natural pensar que se puede encontrar una representación de vectores clásicos con estados cuánticos. Esta representación de hecho existe, y se le denomina "Amplitude encoding", o "Codificación por amplitud". Esta es exactamente lo que uno se imaginaría, se podría representar un vector

clásico normalizado $x \in \mathbb{C}^{2^n}, \sum_{k=1}^{2^n} |x_k|^2 = 1$ con las amplitudes de un estado cuántico $|\psi\rangle$ como

$$y = \begin{pmatrix} x_1 \\ \vdots \\ x_{2^n} \end{pmatrix} \longleftrightarrow |\psi_x\rangle = \sum_{j=1}^{2^n} x_j |j\rangle \tag{3.1}$$

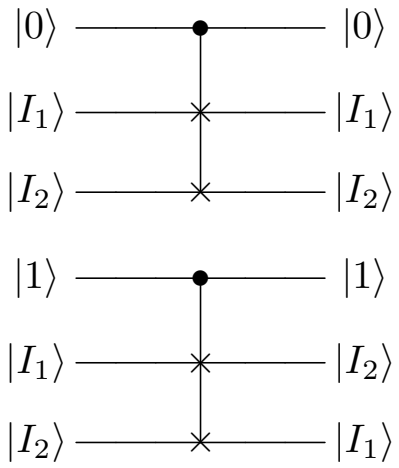
Por ejemplo, se puede suponer que se dispone del siguiente vector $x = (1, 2, 3)$ que normalizado pasaría a $(0{,}267, 0{,}535, 0{,}802)$. Luego, se rellena el vector con 0(s) para que tenga una dimensión potencia de 2: $x' = (0{,}267, 0{,}535, 0{,}802, 0)$

Este vector puede ser representado como un estado cuántico dado por dos qubits:

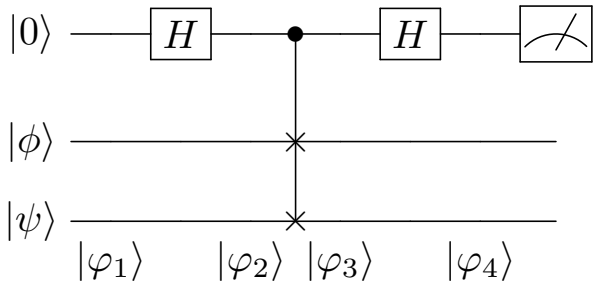
$$0{,}267\,|\mathbf{00}\rangle + 0{,}535\,|\mathbf{01}\rangle + 0{,}802\,|\mathbf{10}\rangle + 0\,|\mathbf{11}\rangle$$

3.5.1.2. Swap test

Se puede recordar la compuerta de Fredkin, también conocida como intercambio controlado. (abreviado CSWAP por el inglés "controlled swap", abreviación que utilizaremos), mencionada anteriormente. Esta consiste de 3 entradas, (C, I_1, I_2) y 3 salidas, cuyo resultado es mantener el bit de control, e invertir las compuertas I_1 y I_2 si el bit de control es 1.



Descubierta inicialmente por [Barenco et al., 1996], la SWAP Test es una simple pero inteligente prueba para medir la similitud de dos estados $|\phi\rangle$ $|\psi\rangle$ utilizando una compuerta CSWAP y dos compuertas de Hadamard de la siguiente manera:



El estado $|\varphi_1\rangle$ es naturalmente $|0,\phi,\psi\rangle$. Luego de la primera compuerta de Hadamard, el estado pasa a

$$|\varphi_2\rangle = \frac{1}{\sqrt{2}}(|0,\phi,\psi\rangle) + \frac{1}{\sqrt{2}}(|1,\phi,\psi\rangle)$$

Al aplicarse CSWAP, se obtiene

$$|\varphi_3\rangle = \frac{1}{\sqrt{2}}(|0,\phi,\psi\rangle) + \frac{1}{\sqrt{2}}(|1,\psi,\phi\rangle)$$

Para finalmente obtener luego de la segunda compuerta de Hadamard

$$|\varphi_4\rangle = \frac{1}{2}|0\rangle(|\phi,\psi\rangle + |\psi,\phi\rangle) + \frac{1}{2}|1\rangle(|\phi,\psi\rangle - |\psi,\phi\rangle)$$

Entonces, la medición en el qubit de control dará 0 con una probabilidad de

$$P(qubitcontrol = 0) = \frac{1}{2}(\langle\phi|\langle\psi| + \langle\psi|\langle\phi|)\frac{1}{2}(|\phi\rangle|\psi\rangle + |\psi\rangle|\phi\rangle) = \frac{1}{2} + \frac{1}{2}|\langle\phi|\psi\rangle|^2$$

lo que devuelve el valor absoluto del producto interno como

$$|\langle\phi|\psi\rangle| = \sqrt{1 - 2P(qc = 0)}$$

3.5.1.3. Algoritmo

[Kopczyk, 2018] Teniendo entonces una representación para vectores clásicos con estados cuánticos y una fórmula para su producto interno, se procederá al cálculo de su diferencia. Sean dos vectores a y b, y $Z = |a|^2 + |b|^2$, entonces se inicializarán los siguientes vectores:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0,a\rangle + |1,b\rangle)$$

$$|\phi\rangle = \frac{1}{\sqrt{Z}}(|a|\,|0\rangle + |b|\,|1\rangle)$$

Se observa que

$$\langle\phi|\psi\rangle = \frac{1}{\sqrt{2Z}}(|a|\,|a\rangle + |b|\,|b\rangle) \rightarrow |a-b|^2 = 2Z|\langle\phi|\psi\rangle|^2$$

3.5.1.4. Tiempo computacional

Según [Lloyd et al., 2013] el tiempo requerido para armar los estados es $logN$, siendo N la dimensión de los vectores. Se puede ver que es la cantidad de qubits necesarios para representar estos vectores. Dado que el tiempo requerido para calcular la distancia en un algoritmo clásico es $O(N)$, este algoritmo logra una mejora exponencial.

Desarrollo

4.1. Decisiones tomadas

4.1.1. Elección del objeto de estudio

El presente trabajo de investigación se centra en la exploración y análisis comparativo de la eficiencia de los algoritmos k-means y DBSCAN, tanto en el contexto de la Computación Clásica como en el emergente campo de la Computación Cuántica. La elección de estos dos algoritmos se basa en una serie de factores que incluyen su relevancia en el campo de la ciencia de datos, su aplicabilidad en una variedad de contextos y su potencial para beneficiarse de las ventajas inherentes a la Computación Cuántica.

La razón principal por la cual se elige el algoritmo k-means para estudiar es su popularidad, la cual viene dada por las siguiente razones:

- La complejidad temporal es $O(nkld)$ donde n es el número de puntos, k es el número de clústeres, l es el número de iteraciones que le lleva al algoritmo converger y d las dimensiones de los puntos.
- La complejidad espacial es $O(k + n)$. Requiere espacio adicional para almacenar la matriz de datos.
- Es independiente del orden. Dado un conjunto inicial de centroides, genera la misma partición de datos sin importar el orden en que los patrones sean presentados al algoritmo.

Los algoritmos k-means y DBSCAN son dos de los algoritmos de agrupamiento más utilizados en el campo del aprendizaje automático [Hashemifar,]. k-means es un algoritmo de agrupamiento basado en centroides que se utiliza ampliamente en una variedad de aplicaciones que van desde la segmentación de imágenes hasta el análisis de redes sociales. DBSCAN, por otro lado, es un algoritmo de agrupamiento basado en densidad que ha demostrado ser particularmente útil en aplicaciones que requieren la identificación de agrupaciones de formas arbitrarias en los datos. Ambos algoritmos, sin embargo, pueden ser computacionalmente intensivos, especialmente cuando se aplican a grandes conjuntos de datos [HDBSCAN Developers,]. En el caso de k-means, el cálculo de las distancias entre los puntos de datos y los centroides del clúster puede ser una operación costosa en términos de tiempo de ejecución. Del mismo modo, en DBSCAN, la necesidad de calcular las distancias entre

todos los pares de puntos de datos puede resultar en un alto costo computacional, como se puede ver en los experimentos realizados.

Aquí es donde la Computación Cuántica puede ofrecer ventajas significativas. La Computación Cuántica es una nueva forma de procesamiento de información que se basa en los principios de la Mecánica Cuántica. A diferencia de la Computación Clásica, que se basa en bits que pueden estar en uno de dos estados (0 o 1), la Computación Cuántica utiliza qubits que pueden existir en superposiciones de estados. Esto permite que las computadoras cuánticas realicen múltiples cálculos simultáneamente, lo que potencialmente puede llevar a un aumento exponencial en la velocidad de procesamiento, como se ve en el Estado del Arte.

En el caso de los algoritmos k-means y DBSCAN, se espera que la capacidad de las computadoras cuánticas para realizar cálculos en paralelo pueda ser utilizada para acelerar las operaciones de cálculo de distancia. En particular, la ventaja de estos algoritmos está en la capacidad de poder realizar el cálculo de distancia en una subrutina cuántica para acelerar este posible cuello de botella para k-means y DBSCAN. Si esto resulta ser cierto, podría tener implicaciones significativas para la eficiencia de estos algoritmos cuando se aplican a grandes conjuntos de datos.

Es importante destacar, sin embargo, que este trabajo de investigación no asume a priori que la Computación Cuántica proporcionará necesariamente una ventaja sobre la Computación Clásica. En lugar de ello, se adopta un enfoque neutral y se realizan experimentos para comparar la eficiencia de estos algoritmos en ambos contextos. También se exploran técnicas de paralelismo en la Computación Clásica como una posible alternativa para mejorar la eficiencia de estos algoritmos.

4.1.2. Lenguajes y frameworks

Se decidió utilizar Python y Qiskit para realizar las implementaciones de los algoritmos.

Python es un lenguaje de programación interpretado de alto nivel que ha ganado inmensa popularidad en los últimos años debido a su simplicidad, legibilidad y versatilidad.
Algunas de las razones clave para elegir Python son:

- Legibilidad y facilidad de uso.

  La sintaxis de Python está diseñada para ser fácilmente legible y comprensible, lo que lo convierte en una excelente opción. Esta legibilidad permite un desarrollo y depuración más rápidos, lo cual es crucial al trabajar en problemas complejos de Computación Cuántica. Además, todos los miembros del equipo estaban familiarizados con Python de una u otra manera.

- Amplias bibliotecas y paquetes.

  Python cuenta con un vasto ecosistema de bibliotecas y paquetes que atienden a diversos dominios, incluyendo la computación científica, el aprendizaje automático y el análisis de datos. Este rico ecosistema permite aprovechar las herramientas y bibliotecas existentes para implementar algoritmos de Computación Cuántica de manera eficiente.

- Compatibilidad multiplataforma.

  Python es independiente de la plataforma, lo que significa que puede ejecutarse en varios sistemas operativos, incluidos Windows y macOS, que son los que normalmente utilizamos.

- Fuerte apoyo de la comunidad.

  Python tiene una gran y activa comunidad de desarrolladores que contribuyen a su desarrollo y brindan soporte a través de foros, listas de correo y otros recursos en línea. Este sólido apoyo de la comunidad garantiza que se puedan encontrar rápidamente soluciones a cualquier problema que pueda surgir durante la implementación.

Por otro lado, Qiskit es un marco de Computación Cuántica de código abierto desarrollado por IBM que permite a los usuarios diseñar, simular y ejecutar circuitos cuánticos en hardware cuántico real. Algunas de las razones clave para elegir Qiskit son:

- Abstracción de alto nivel.

  Qiskit proporciona una abstracción de alto nivel para diseñar e implementar circuitos cuánticos, lo que simplifica el proceso de creación y prueba de algoritmos cuánticos. Esta abstracción permite centrarse en los conceptos y la lógica central de la implementación sin perdernos en detalles de bajo nivel.

- Integración con Python

  Qiskit está diseñado para funcionar a la perfección con Python, lo que significa que se puede integrar fácilmente la implementación basada en Python con las capacidades de Computación Cuántica de Qiskit. Esta integración permite aprovechar el poder de las bibliotecas y paquetes de Python mientras se trabaja con circuitos cuánticos.

- Acceso a hardware cuántico real.

  Qiskit proporciona acceso al hardware cuántico real de IBM a través de la plataforma en la nube IBM Quantum Experience. Este acceso permite probar la implementación en computadoras cuánticas reales, proporcionando información valiosa sobre el rendimiento y la viabilidad de los algoritmos.

- Desarrollo activo y apoyo de la comunidad.

  Qiskit es desarrollado y mantenido activamente por IBM y una gran comunidad de colaboradores. Este desarrollo activo garantiza que el marco se mantenga actualizado con los últimos avances en Computación Cuántica. Además, la comunidad de Qiskit brinda soporte a través de foros, tutoriales y documentación, lo cual puede ser invaluable al trabajar en problemas complejos de Computación Cuántica.

4.2. Análisis de costos

Debido a su complejidad, las computadoras cuánticas tienen un costo significativo.

Actualmente, las computadoras cuánticas se encuentran en la etapa inicial de su desarrollo [The Quantum Insider, 2023]. Empresas como Microsoft, Google e IBM (y muchas startups también), así como decenas de universidades e institutos de investigación gubernamentales, están investigando, desarrollando y fabricando modelos NISQ cuyas aplicaciones, se espera, tengan efectos de gran

alcance en la optimización y logística, simulación de sistemas complejos para la química molecular y problemas de criptografía.

El término NISQ es relativamente nuevo (2018). Se refiere a la era de la Computación Cuántica ruidosa a escala intermedia que se está experimentando, y se caracteriza por el estado actual de los procesadores de hardware cuántico. Por ahora, estos procesadores se denominan dispositivos NISQ, ya que no poseen todas las capacidades de una computadora cuántica completamente funcional. Un dispositivo cuántico ruidoso a escala intermedia (NISQ) es una computadora cuántica que no es tan potente como debería ser para resolver los problemas difíciles en la Computación Cuántica, pero aún tiene suficientes capacidades para ser útil en la resolución de problemas menos complejos. [Objectivity,]

Hoy por hoy es posible comprar una computadora cuántica. Sin embargo, las computadoras cuánticas aún no están ampliamente disponibles para el público en general y actualmente son muy costosas y extremadamente difíciles de fabricar. Otro obstáculo a considerar es que la mayoría de estas máquinas son propiedad u operadas por grandes corporaciones, instituciones de investigación y organismos gubernamentales, por lo que el acceso a ellas (sin mencionar la compra de una) depende de los términos de acceso de estas empresas e instituciones.

Sin embargo, si un individuo (o empresa, en ese caso) estuviera interesado en comprar una computadora cuántica, podría contactar directamente a empresas como IBM, Microsoft, Rigetti Computing, D-Wave o Google. Estas compañías pueden ofrecer a los individuos acceso a sus recursos de Computación Cuántica a través de servicios en la nube, permitiendo a investigadores, desarrolladores y empresas experimentar y utilizar la potencia de la Computación Cuántica sin poseer el hardware físico. El costo de utilizar estos servicios puede variar según la cantidad de tiempo y recursos utilizados, pero generalmente oscila entre unos pocos dólares y varios miles de dólares por hora.

Como ya se mencionó, la informática cuántica es una tecnología que avanza rápidamente pero que aún se encuentra en sus primeras etapas. La advertencia es que ponerle precio a una máquina así es más un arte que una ciencia. Las cosas a considerar antes de la hipotética compra de una computadora cuántica podrían ser qué tipo de sistema desea comprar, la cantidad de qubits requeridos y, otro aspecto importante, el nivel de soporte experto requerido por parte del proveedor al utilizar la máquina. Todos estos factores influyen en el costo total. [The Quantum Insider, 2023]

Para dar una estimación, The Verge citó que costaría unos 15 millones de dólares comprar la computadora cuántica D-Wave 2000Q, aunque el artículo es de 2017. [The Verge, Chaim Gartenberg, 2017]

Como cifra aproximada, el costo de comprar una máquina completa en 2023 es actualmente muy alto, con el equipo que se incluye en la mayoría de los sistemas costando cientos de miles de dólares. Los sistemas de Computación Cuántica como IBM se venden por decenas de millones como parte de un contrato de servicio completo durante varios años.

Por ejemplo, el servicio en la nube de Computación Cuántica Azure de Microsoft permite a los usuarios 500 dólares en créditos gratuitos de Azure Quantum.

Muchos expertos en la industria predicen que el costo del hardware de Computación Cuántica continuará disminuyendo con el tiempo a medida que avance la tecnología, lo que lo hará más accesible para una gama más amplia de empresas y organizaciones. En una charla reciente, el CTO de la CIA, Nand Mulchandani, señaló que la industria cuántica todavía está en una etapa temprana y los costos unitarios siguen siendo muy altos.

En general, los precios de las computadoras cuánticas seguramente estarán influenciados por varios factores importantes, como los avances en descubrimientos en el sector, la demanda del mercado de la tecnología y la competencia entre los proveedores de Computación Cuántica.

Teniendo esto en cuenta, las computadoras cuánticas más avanzadas tecnológicamente en la actualidad, como las de IBM y Google, tienen cientos de qubits, lo cual es un gran logro en sí mismo. Sin embargo, para llegar realmente a donde se quiere estar con la tecnología, es posible que la cantidad de qubits deba aumentar significativamente, quizás a miles o incluso millones de qubits, para tener un impacto útil en el ámbito comercial. [The Quantum Insider, 2023]

Además, estas máquinas requieren infraestructuras complejas y sistemas de enfriamiento (en muchos casos, véase la Computación Cuántica superconductora) para mantener las bajas temperaturas requeridas para que los qubits funcionen correctamente, así como sistemas de corrección de errores confiables.

A pesar de estos desafíos, la industria avanza lentamente. Aunque es difícil predecir con precisión el cronograma para cuando las computadoras cuánticas alcancen el uso comercial, muchos expertos creen que las aplicaciones prácticas pueden surgir en las próximas décadas.

Es importante mencionar que cuando se comenzaron a realizar pruebas en las computadoras de IBM, sólo se podía tener acceso a computadoras de 7 qubits de forma gratuita y de 27 qubits de forma paga, a un costo de 1,60 dólares por segundo, lo que serían 960 dólares cada 10 minutos. Hoy en día, se ha limitado el uso de la máquina de 7 qubits a 10 minutos mensuales, pero se brinda acceso a 10 minutos mensuales para el uso de una máquina de 127 qubits, lo que antes costaba 960 dólares. Obviamente se puede observar como aún es extremadamente caro correr los jobs, y además de esto, al haber muy pocas computadoras cuánticas a disposición, los tiempos de espera suelen ser realmente largos, lo que dificulta el proceso de implementación y testeo.

Esta información sobre los precios y el acceso a las computadoras cuánticas de IBM es relevante para comprender cómo ha evolucionado la disponibilidad y el costo de la tecnología cuántica en los últimos años. A medida que la tecnología avance, es probable que se vean cambios adicionales en los precios y el acceso a las computadoras cuánticas, lo que permitirá a más investigadores y empresas aprovechar el poder de la Computación Cuántica.

En resumen, las computadoras cuánticas son una tecnología emergente y en rápido desarrollo con un gran potencial para transformar diversos campos. Aunque actualmente son costosas y difíciles de fabricar, es probable que los precios disminuyan y la tecnología se vuelva más accesible con el tiempo. Al considerar la adquisición de una computadora cuántica, es fundamental tener en cuenta factores como el tipo de sistema, la cantidad de qubits y el soporte experto requerido, así como las opciones de acceso a través de servicios en la nube.

4.3. Desarrollo de software

Para este proyecto, se siguió un enfoque de desarrollo iterativo e incremental, ya que este permite adaptarse a los cambios y mejoras en los algoritmos y tecnologías a medida que se avanzó en la investigación. Además, permitió evaluar y comparar diferentes enfoques y tecnologías en cada iteración.

Se identifican los siguientes requisitos funcionales y no funcionales:

Requisitos funcionales:

- Implementar algoritmos k-means y DBSCAN en Python.
- Realizar experimentos con paralelización para evaluar la mejora en el tiempo de ejecución.
- Implementar la medición de distancia en una computadora cuántica utilizando Qiskit.

Requisitos no funcionales:

- Facilidad de uso y comprensión del código.
- Rendimiento y eficiencia en la ejecución de los algoritmos.
- Portabilidad y compatibilidad con diferentes plataformas y entornos de ejecución.

Este proyecto se centra en la investigación y comparación de algoritmos y tecnologías, por lo que no incluye una interfaz de usuario específica. Sin embargo, se diseñaron implementaciones para ser fácilmente utilizables y comprensibles por otros investigadores y desarrolladores.

Se realizaron pruebas unitarias para garantizar la corrección de las implementaciones, las mismas se pueden encontrar en el Anexo.

Se trabajó de manera colaborativa utilizando herramientas de comunicación como Slack para mantenernos informados y coordinados en todo momento. Además, se realizaron reuniones periódicas para discutir el progreso, los desafíos y las soluciones encontradas.

Experimentación

5.1. Principales impedimentos para la implementación de un algoritmo de Clustering en Computación Cuántica

Aunque esta emergente tecnología promete una capacidad computacional extraordinariamente poderosa, su aplicabilidad práctica enfrenta varios obstáculos significativos. Este análisis se dedica a discernir los principales impedimentos inherentes a la implementación de un algoritmo de Clustering en un contexto cuántico.

5.1.1. Acceso a procesadores cuánticos

En el contexto de este estudio académico, es esencial subrayar un aspecto crítico de la metodología de investigación: la utilización de procesadores cuánticos proporcionados por IBM. La relevancia de este detalle no debe ser subestimada, ya que tiene implicaciones significativas para la interpretación y el análisis de los resultados.

La Computación Cuántica promete avances revolucionarios en la eficiencia de muchos algoritmos. Sin embargo, es crucial entender que la implementación práctica de estos avances teóricos puede enfrentarse a una serie de desafíos y limitaciones, particularmente para usuarios finales o empresas que deben de acceder a estos dispositivos provistos por otras empresas debido a la escasez actual de esta tecnología. En particular, el uso de procesadores cuánticos provistos por IBM introduce una serie de factores adicionales que deben ser considerados.

Uno de estos factores es el tiempo de espera en la cola. Dado que los recursos de Computación Cuántica son limitados y muy demandados, a menudo hay un retraso significativo debido a la cantidad de usuarios que desean acceder a estos recursos. Este tiempo de espera puede tener un impacto considerable en la eficiencia global de los algoritmos que se están utilizando.

Además, el proceso de armado de circuitos cuánticos puede ser particularmente costoso en términos de tiempo y recursos. Este es un aspecto especialmente relevante en este caso, ya que cada cálculo de distancia requiere el envío de un proceso a IBM para obtener una única distancia. Dado que estos algoritmos se utilizan para el procesamiento de una gran cantidad de datos, este proceso añade un tiempo considerable que se suma a cada cálculo de distancia.

Por lo tanto, aunque la Computación Cuántica ofrece un potencial teórico significativo, es importante tener en cuenta que la realidad práctica de su implementación puede presentar desafíos

significativos. En particular, los tiempos de espera (como se puede ver en la Figura 5.1) y el coste del armado de circuitos cuánticos pueden ser factores limitantes importantes. Este puede ser uno de los principales puntos negativos que se observan con la Computación Cuántica actual, y es un aspecto que debe ser cuidadosamente considerado en el análisis de los resultados presentados.

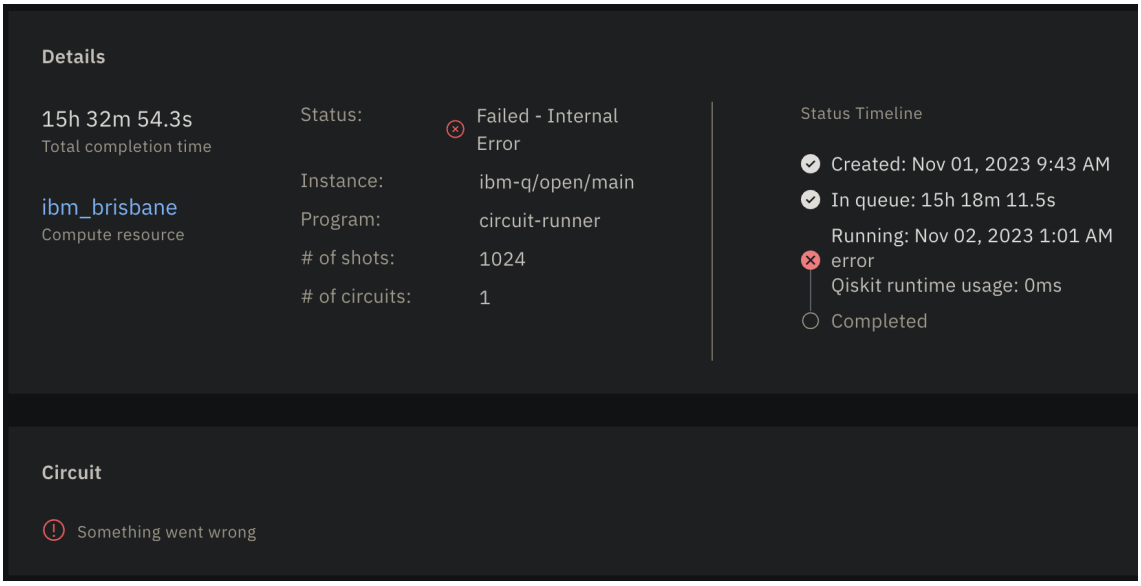


Figura 5.1: Una de las principales dificultades fueron los tiempos de cola. Al existir un único procesador cuántico de 127 qubits de forma "gratuita" en IBM, la demanda del mismo es alta, lo que hace que se den tiempos de espera prolongados. Además de los tiempos de espera, el manejo de errores es otro problema, debido a que ciertos procesos fallaban sin dar ningún tipo de información acerca de cuál fue la falla que produjo que el proceso entero termine, lo que hacía que las iteraciones fueran más lentas.

5.1.2. Dificultades de programación

La primera barrera significativa radica en la complicada naturaleza de la programación cuántica. A diferencia de la programación clásica, la cuántica implica la manipulación de qubits y la aplicación de operaciones cuánticas, como superposición y entrelazamiento. Esto conlleva una curva de aprendizaje empinada y una mayor propensión a errores. Aunque se use una librería como interfaz a la hora de programar el circuito cuántico, se necesitan conocimientos profundos relacionados no solo a la Computación Cuántica, sino también a la parte física que da sustento a las diferentes operaciones, lo que lo hace un estilo de programación difícil para encontrar errores.

Por ejemplo, herramientas como Qiskit ofrecen una plataforma robusta para la codificación de funciones cuánticas, pero presentan limitaciones críticas. Qiskit, pese a su rica documentación y capacidades, está restringido al manejo de bits unitarios (Int1), limitando la complejidad y diversidad de operaciones que se pueden ejecutar [Qiskit Team,].

Durante la realización de experimentos para este trabajo, Qiskit presentó dificultades de programación. Qiskit tiene a disposición en su librería la habilidad de simular una computadora cuántica

para probar el código escrito, pero luego, al cambiar a una computadora cuántica real como las de IBM se presentan múltiples problemas que hacen que la transición desde el simulador a una computadora cuántica real sea más difícil.

5.1.3. Costos elevados

Otro desafío prominente, el cual ya se vio previamente, es el costo asociado con la utilización de recursos cuánticos. Las tarifas por el uso de computadoras cuánticas son elevadas, rondando aproximadamente 1.60 dólares por segundo fuera del plan gratuito. Esto pone un límite significativo en la extensión y la profundidad de los experimentos y aplicaciones que se pueden realizar, especialmente para investigadores o instituciones con presupuestos limitados.

Recientemente, IBM agregó una nueva computadora cuántica de 127 qubits a disposición en el plan gratuito, aunque solamente se pueda acceder a 10 minutos de cómputo por mes en esta computadora, resulta atractivo para estudiantes para poder realizar experimentos más complejos.

5.1.4. Limitaciones de memoria y almacenamiento

La falta de memoria o disco de fácil acceso en el paradigma cuántico es otro obstáculo crucial. En la mayoría de los casos, los datos deben estar "hardcodeados" dentro del circuito cuántico. Esto lleva a una construcción de circuito que es intensiva en tiempo y recursos, especialmente cuando se trata de crear QROMs (Quantum Read-Only Memories) para almacenar y acceder a los datos necesarios para los algoritmos de Clustering.

5.1.5. Overhead en la comunicación entre Computación Clásica y Cuántica

Además, se enfrenta el problema del overhead en la comunicación entre las computaciones clásica y cuántica. En contextos como el cálculo de distancias en algoritmos de Clustering, donde una operación relativamente corta debe ser repetida múltiples veces, este overhead puede ser especialmente perjudicial. Puede llevar a una disminución severa en la eficiencia global del algoritmo debido al tiempo y recursos consumidos en la comunicación constante entre los dos dominios computacionales. Actualmente, con el plan gratuito esto se agrava en la computadora con una mayor cantidad de qubits ya que esta posee una demanda elevada. Alcanzando tiempos de espera de días para la ejecución de un circuito cuántico.

5.1.6. Errores y precisión

La naturaleza propensa a errores de los sistemas cuánticos también es una barrera. Los qubits son susceptibles a errores debido a la decoherencia y otros factores de ruido. Esto puede afectar la precisión y fiabilidad de los resultados obtenidos a través de algoritmos cuánticos de Clustering. Es esencial una robusta corrección de errores y técnicas de mitigación para asegurar la validez de los cálculos cuánticos.

En el video publicado por Microsoft [Research, 2018], se explica que teóricamente, para lograr un qubit lógico, se necesitarían aproximadamente 5 qubits físicos. Sin embargo, en la práctica, parece que para lograr un qubit confiable al 100 %, se necesitarían unos 100 o 200 qubits. Esto es algo a tener en cuenta, ya que por ejemplo si se observa una computadora de 30 qubits, probablemente no significa tanto como se espera, sino que serían unos 2 o 3 qubits de buena calidad.

5.1.7. Escalabilidad

La escalabilidad de los algoritmos cuánticos también es cuestionable. A medida que el tamaño del problema o el conjunto de datos aumenta, la cantidad de recursos cuánticos requeridos también escala significativamente. Esto puede llevar a dificultades con la implementación práctica de algoritmos de Clustering en problemas más grandes y más complejos.

5.1.8. Madurez tecnológica

Finalmente, la Computación Cuántica todavía está en una etapa relativamente temprana de desarrollo. Muchos algoritmos y técnicas aún están en fases experimentales, y la infraestructura cuántica accesible para la investigación y el desarrollo todavía está creciendo y madurando. Esta falta de madurez tecnológica significa que hay una falta de herramientas, bibliotecas y recursos bien establecidos y optimizados para facilitar el desarrollo eficiente de algoritmos de Clustering cuántico.

5.1.9. Conclusión

El potencial de la Computación Cuántica en la mejora de la eficiencia y capacidad de los algoritmos altamente combinatorios es evidente. Sin embargo, a la luz de los impedimentos presentados, es crucial abordar y mitigar estas barreras para avanzar hacia una implementación práctica y efectiva de algoritmos de Clustering en el reino cuántico. Un enfoque deliberado y considerado hacia la superación de estos desafíos permitirá el despliegue pragmático de soluciones cuánticas en el ámbito de los algoritmos combinatorios y más allá.

5.2. Análisis

El objetivo de esta sección es analizar los algoritmos seleccionados, k-means y DBSCAN, para identificar los cuellos de botella en su rendimiento computacional y explorar la posibilidad de mejorar su eficiencia mediante la implementación de algoritmos cuánticos. Esta investigación tiene el potencial de demostrar la relevancia de los algoritmos cuánticos y sus aplicaciones para optimizar el tiempo de ejecución en problemas de alta dimensionalidad, que son de gran importancia en la sociedad actual.

En ambos algoritmos, el cálculo de la distancia entre los puntos es una operación que se repite con frecuencia y, por lo tanto, se espera que sea uno de los principales cuellos de botella en términos de rendimiento computacional. Este cálculo se vuelve cada vez más costoso a medida que aumenta la dimensionalidad de los datos, debido a la "maldición de la dimensionalidad", un fenómeno que ocurre cuando el número de características en un conjunto de datos es tan alto que el espacio de características se vuelve muy disperso.

La Computación Cuántica ofrece una solución prometedora para este problema. Las computadoras cuánticas pueden calcular la distancia entre dos puntos con un orden de complejidad más bajo que las computadoras tradicionales, gracias a su capacidad para realizar cálculos en superposición, lo que permite disminuir el tiempo de ejecución de cada uno de estos cálculos de distancia.

Por lo tanto, se espera que el uso de computadoras cuánticas para el cálculo de distancias en los algoritmos k-means y DBSCAN pueda mejorar significativamente su rendimiento, especialmente en conjuntos de datos de alta dimensionalidad. Para validar esta hipótesis, se realizó un análisis de ambos algoritmos, con un enfoque particular en la identificación de los cuellos de botella en su ejecución en computadoras tradicionales.

Además, se realizaron experimentos utilizando técnicas de paralelismo en computadoras tradicionales para evaluar la velocidad de esta alternativa frente a la Computación Cuántica.

En las siguientes secciones, se realizarán experimentos para demostrar los puntos previamente elaborados. La primera prueba será identificar los cuellos de botella para los algoritmos elegidos. Una vez identificados los cuellos de botella para ambos algoritmos, se procederá a investigar posibles implementaciones cuánticas que permitan reducir estos cuellos de botella con el objetivo de mejorar el tiempo de ejecución de estos algoritmos para problemas que presenten una alta dimensionalidad en los datos. Una vez se cuente con las implementaciones para Computación Clásica y Cuántica, se comparan con la alternativa usando computación paralela.

En función de lo anterior, se llevará a cabo un análisis para determinar si el estado actual de la Computación Cuántica, que abarca no solo los avances tecnológicos, sino también la accesibilidad y la facilidad de uso para los usuarios finales, proporciona alguna ventaja sobre la Computación Clásica en la resolución de problemas de agrupamiento en espacios de alta dimensionalidad.

5.2.1. k-means

En la presente sección, se procederá a realizar un análisis de los tiempos de ejecución de los diversos componentes que conforman el algoritmo k-means. El propósito fundamental de este análisis es identificar los posibles cuellos de botella que puedan estar afectando la eficiencia del algoritmo. Como se destacó en la sección anterior, la identificación precisa de estos cuellos de botella es un paso crucial para la implementación efectiva de algoritmos cuánticos. El objetivo final es aplicar la Computación Cuántica para mitigar estos cuellos de botella, lo que a su vez debería resultar en una reducción significativa del tiempo total de ejecución del algoritmo.

Dado el funcionamiento intrínseco del algoritmo k-means, se anticipa que el cálculo de la distancia entre los puntos de datos podría ser el componente que más tiempo consume. Esta hipótesis se basa en la naturaleza del algoritmo, que requiere repetidos cálculos de distancia para la asignación de puntos de datos a los respectivos centroides.

Para validar esta hipótesis, se ha llevado a cabo un análisis del tiempo de ejecución de cada componente de la implementación desarrollada para el algoritmo k-means. Los resultados de este análisis se presentan en la Figura 5.2.

La interpretación de los datos presentados en la figura confirma la hipótesis inicial. Se observa claramente que, en el contexto del algoritmo k-means, el cálculo de la distancia es, con diferencia, el componente que más tiempo consume, representando el 76.88 % del tiempo total de ejecución.

Esta confirmación empírica de la hipótesis inicial proporciona una base sólida para la exploración de alternativas cuánticas para el cálculo de la distancia. Con la implementación de técnicas cuánticas, se ha logrado una mejora significativa en el tiempo de cómputo. Esta mejora es particularmente relevante para vectores de alta dimensión, donde la reducción en el tiempo de cómputo puede tener

un impacto notable en la eficiencia general del algoritmo.

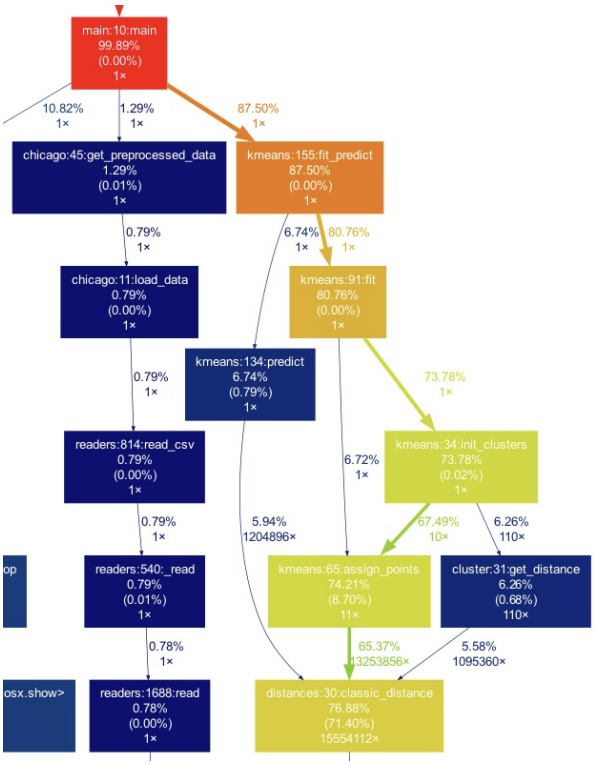


Figura 5.2: Tiempos de ejecución de cada componente del algoritmo k-means.

5.2.2. DBSCAN

El alto costo computacional de la ejecución de DBSCAN se atribuye principalmente al cálculo repetido de las distancias entre puntos cuando se agrupan en clústeres basados en la densidad. En una implementación ingenua o "naive" de DBSCAN, este cálculo de distancia se realiza cada vez que es necesario.

A diferencia de otros algoritmos, como el k-means, DBSCAN ofrece una oportunidad única para reutilizar las distancias previamente calculadas, permitiéndonos así dirigir los esfuerzos hacia la paralelización eficiente de la construcción de la matriz de distancias, buscando de esta manera una mejora sustancial en el rendimiento temporal del algoritmo.

Con una metodología análoga aplicada en análisis previos del algoritmo k-means, se ha desglosado y cuantificado el tiempo de ejecución de diferentes componentes del algoritmo DBSCAN utilizando el profiler cPython.

A partir del análisis (Figura 5.4), es evidente que una proporción predominante, aproximadamente el 99 %, del tiempo de ejecución total se consume en la construcción de la matriz de distancias. Este procedimiento muestra una complejidad que aumenta linealmente con respecto al número de

Figura 5.3: Visualización del resultado de la ejecución de un profiling sobre la ejecución del algoritmo DBSCAN "naive".

características y cuadráticamente en relación con el número de puntos, identificándose así como el principal cuello de botella y el área crítica a la cual deberían dirigirse los esfuerzos de optimización.

En la adaptación y aplicación de técnicas cuánticas para la mejora de este proceso, se propone emular estrategias previamente exploradas en algoritmos como k-means. El enfoque primordial radica en la optimización del cálculo de distancias entre puntos.

5.2.2.1. Implementación tradicional

En esta sección, se abordará la implementación tradicional del algoritmo DBSCAN (Density-Based Spatial Clustering of Applications with Noise) efectuada como parte de la investigación.(Anexo 1: Código - DBSCAN: Implementación tradicional)

El código proporcionado realiza una implementación precisa de DBSCAN, comenzando con la función principal DBSCAN(DB, distMatrix, eps, minPts). Esta función recibe como parámetros una base de datos DB que contiene los puntos a agrupar, una matriz de distancias precalculada distMatrix, un radio especificado eps y un número mínimo de puntos minPts. La función DBSCAN se encarga de organizar y asignar etiquetas a los puntos en clústeres o como ruido, basándose en las propiedades de densidad del espacio de datos.

Inicialmente, se establece un contador de clústeres C en cero y se crea una lista labels para

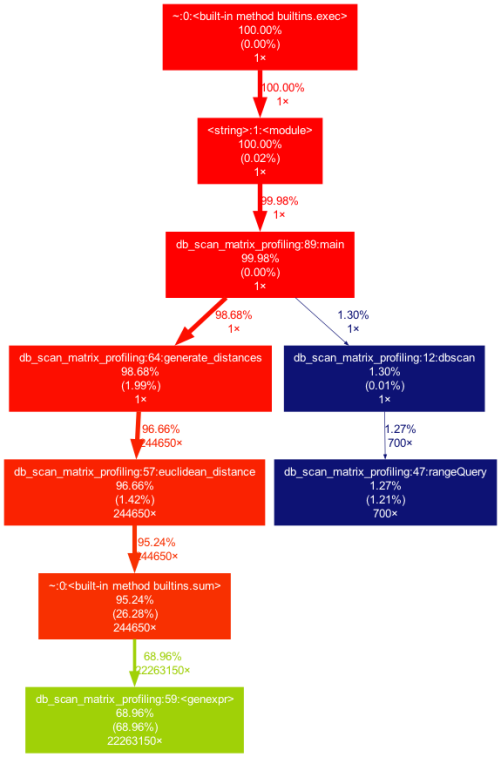


Figura 5.4: Visualización del resultado de la ejecución de un profiling sobre la ejecución del algoritmo DBSCAN con el precálculo de distancias.

almacenar las etiquetas de cada punto en la base de datos. La función procede a iterar sobre cada punto P en la base de datos, verificando y asignando etiquetas conforme avanza. Si un punto ya ha sido etiquetado, la función continúa con el siguiente punto.

Para cada punto sin etiquetar, se realiza una consulta de rango utilizando la función rangeQuery(DB, distMatrix, P_idx, eps), que devuelve todos los puntos dentro de una distancia eps del punto de consulta. Si el número de puntos vecinos, incluyendo el punto mismo, es menor que minPts, se etiqueta el punto como "Ruido"; de lo contrario, se procede a construir un clúster.

Una vez identificado un punto como central para formar un nuevo clúster, se incrementa el contador de clústeres y se asigna la nueva etiqueta de clúster al punto. Los puntos vecinos se agregan a un conjunto de semillas seed_set, que actúa como una frontera para explorar el clúster. A continuación, la función entra en un bucle donde explora y expande el clúster examinando los puntos en el seed_set. Si un punto vecino del conjunto tiene suficientes puntos en su vecindad, se agregan esos puntos al seed_set para futuras exploraciones.

Las funciones auxiliares rangeQuery y generate_distances juegan roles cruciales en la implementación. rangeQuery identifica los puntos dentro del radio eps del punto de consulta, utilizando la matriz de distancias para determinar las proximidades. generate_distances, por otro lado, construye la matriz de distancias entre todos los pares de puntos en la base de datos utilizando una función de distancia especificada, facilitando así las consultas de distancia repetidas durante la ejecución del

algoritmo.

5.3. Implementación paralela

5.3.1. k-means

Paralelizar el procesamiento puede ser una estrategia muy útil en este caso, ya que medir la distancia de todos los puntos con el resto de clústeres es una tarea donde cada distancia es independiente, por lo tanto, cumple con los requisitos para poder ser paralelizable. Es importante notar que paralelizar cada medición de distancia no siempre es buena idea, debido a que generar un nuevo proceso por cada punto del conjunto de datos resulta más costoso que computar la distancia directamente. Lo que se propone como estrategia de paralelización es fraccionar los datos en conjuntos más pequeños de acuerdo a los núcleos que se vayan a usar, por ejemplo, si se cuenta con 200.000 datos y se tienen 8 núcleos, cada núcleo estará encargado de procesar la distancia de 25.000 puntos con cada uno de los clústeres. Este cómputo resulta más eficiente y si se tiene núcleos a disposición puede resultar en una reducción importante del tiempo de cómputo.

5.3.2. DBSCAN

En el esquema avanzado de implementación del algoritmo DBSCAN (Anexo 1 - Código: DBSCAN - Implementación paralela), se exhibe una meticulosa organización y refinamiento algorítmico, el cual se destaca por su estructura modular y la optimización mediante técnicas de programación paralela. Este enfoque está diseñado para efectuar una exploración intensiva y eficiente en el espacio de datos multidimensionales.

Dentro de la implementación, se ha incorporado una estrategia de optimización significativa: la paralelización en el cálculo de las distancias entre puntos. En la función

$$generate\_distances\_mp(DB, distFunc, n\_jobs = -1)$$

se emplea un pool de procesos para distribuir y ejecutar de manera concurrente los cálculos necesarios para completar la matriz de distancias. Este enfoque paralelo permite una utilización más eficiente de los recursos computacionales, logrando una reducción notable en el tiempo requerido para estas operaciones, especialmente en contextos de grandes volúmenes de datos.

5.4. Implementación cuántica

En la sección precedente sobre k-means clásico, se identifica que el cálculo de la distancia constituye el principal obstáculo en términos de rendimiento para el algoritmo k-means. En respuesta a esta limitación, se implementó la subrutina cuántica explicada en el marco teórico específicamente diseñada para calcular la distancia entre dos vectores. Esta implementación cuántica, desde una perspectiva teórica, posee un orden de complejidad inferior al cálculo de distancia realizado en una computadora clásica.

No obstante, es importante destacar que la efectividad de esta implementación cuántica está sujeta a las limitaciones inherentes al estado actual de los procesadores cuánticos disponibles. La primera y más notable de estas limitaciones es que la cantidad de qubits disponibles en el procesador determina la dimensión de los vectores para los cuales se puede calcular la distancia.

Para saber el tamaño máximo de dos vectores para los cuales se desea calcular una distancia en un procesador cuántico se debe conocer el número de qubits. De estos qubits que se tienen disponibles, uno quedará reservado para ser usado como qubit de control y el otro almacenará la norma de cada vector original. Entonces, si se tiene a disposición un procesador cuántico de 7 qubits, se podrá calcular la distancia de dos vectores de tamaño 16, ya que se necesitan 5 qubits para guardar la representación de 32 números.

Además, la necesidad de procesar cada cálculo de distancia en un proceso separado introduce factores adicionales que pueden afectar adversamente el rendimiento del algoritmo. En particular, el tiempo de cola y la generación del circuito para cada cálculo de distancia pueden aumentar significativamente el tiempo total de ejecución.

Por estas razones, se demuestra la correctitud del cálculo de distancia utilizando el simulador que ofrece Qiskit. A pesar de ser ineficiente, es útil para iterar rápido sobre la construcción de algoritmos cuánticos, probando los mismos sobre conjuntos de datos reducidos. En este caso se trabaja con el 30 % de los datos. De esta forma en la siguiente figura se muestra el correcto funcionamiento del algoritmo k-means implementado con distancia cuántica y clásica.

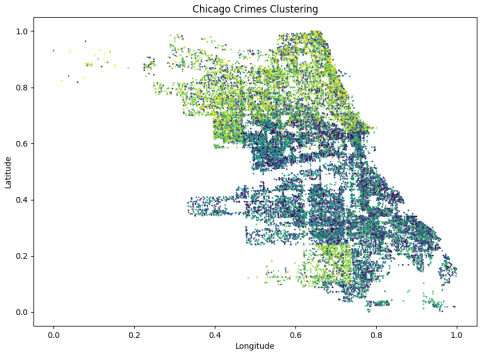


Figura 5.5: K-means utilizando distancia cuántica.

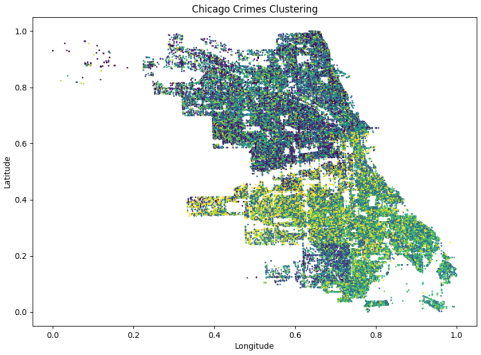


Figura 5.6: K-means utilizando distancia clásica.

Para observar la mejora en los tiempos de ejecución se elaboró el código que se encuentra en el Anexo, en la sección “Cálculo de distancia cuántico”. El mismo, inicializa dos vectores aleatorios normalizados de diferentes tamaños y calcula su distancia. Luego se almacena el tiempo de ejecución en un archivo para poder ejecutar diferentes experimentos con varios tamaños y realizar el análisis que se presentó en la sección anterior.

En resumen, aunque la implementación cuántica del cálculo de la distancia ofrece ventajas teóricas significativas en términos de eficiencia, las limitaciones prácticas asociadas con los recursos de Computación Cuántica disponibles pueden presentar desafíos significativos. Estos desafíos deben ser cuidadosamente considerados al evaluar el rendimiento y la aplicabilidad de un algoritmo k-means cuántico.

5.5. Resultados

5.5.1. k-means

Utilizando el dataset de Chicago [City of Chicago,] el cual contiene registros de los crímenes, desde el 2001 hasta el presente, cometidos en dicha ciudad. Se realizaron experimentos que comparan los tiempos de ejecución en función de la cantidad de puntos. Observando así una mejora del tiempo total del algoritmo si se paraleliza el procesamiento usando diferentes núcleos.

La implementación presentada de k-means acepta paralelización, logrando así reducir los tiempos de ejecución.

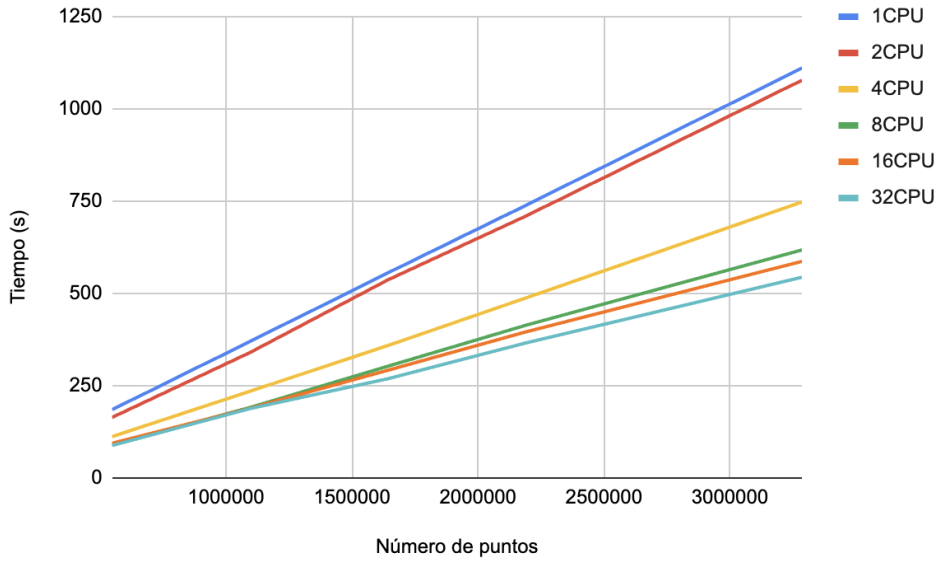


Figura 5.7: Comparación del tiempo de ejecución de k-means utilizando diferentes cantidades de CPUs (1, 2, 4, 8, 16 y 32) en el dataset de Chicago. La gráfica muestra cómo el tiempo de cómputo disminuye a medida que se incrementa el número de núcleos, evidenciando la eficiencia de la estrategia de paralelización propuesta.

5.5.2. DBSCAN

Las experimentaciones correspondientes al algoritmo DBSCAN en su versión tradicional fueron ejecutadas en el Centro Nacional de Supercomputación (ClusterUY). Es preciso mencionar que, debido a las considerables demoras y tiempos de espera en la cola de ejecución de algoritmos cuánticos en las computadoras de IBM, se decidió limitar la ejecución de las pruebas cuánticas en este clúster.

Para garantizar la consistencia y la robustez de los resultados experimentales, se optó por trabajar con un conjunto de datos artificialmente generado mediante la función ‘make_blobs‘ de la librería

scikit-learn [Scikit-learn Developers,]. El conjunto de datos consistió en un total de 1000 puntos, diseñados para ser agrupados en 5 clústeres distintos, facilitando así el proceso de agrupamiento.

Los parámetros esenciales del algoritmo DBSCAN, "eps" y "min_points", fueron configurados a 40 y 20 respectivamente, tras un análisis heurístico preliminar orientado a optimizar los resultados del agrupamiento. Estos valores se seleccionaron basándonos en pruebas exploratorias y revisiones iterativas para asegurar la efectividad del algoritmo en la identificación y asignación correcta de los clústeres.

Se utilizó la librería matplotlib [Matplotlib Developers,] para la visualización gráfica de los clústeres identificados por el algoritmo. A fin de facilitar una representación gráfica más intuitiva y comprensible, se aplicó el método t-distributed stochastic neighbor embedding (t-SNE) de la librería scikit-learn para reducir la dimensionalidad de los datos.

Los resultados obtenidos del proceso de agrupamiento fueron evaluados utilizando el índice Ajustado de Rand (ARI, por sus siglas en inglés) [OECD.AI Policy Observatory,]. Este índice ofreció una medida cuantitativa de la calidad y precisión del agrupamiento, comparando las etiquetas reales retornadas por 'make_blobs' con las etiquetas de clúster asignadas por el algoritmo DBSCAN. Se conservó una semilla constante para la generación de datos entre las diversas pruebas ejecutadas, asegurando así la consistencia y reproducibilidad de los resultados experimentales.

5.5.2.1. Baseline

Se llevó a cabo un análisis baseline para medir el tiempo de ejecución del algoritmo DBSCAN tradicional, ejecutándolo en dos configuraciones distintas: una utilizando un solo procesador (CPU) y otra utilizando 32 procesadores (CPUs). Este análisis se realizó con el objetivo de investigar la existencia de posibles mejoras de eficiencia en la ejecución, atribuibles a factores externos al propio algoritmo.

Los resultados detallan el tiempo de ejecución en función de diferentes cantidades de características (features), proporcionando una vista comparativa del rendimiento del algoritmo en diferentes configuraciones de procesadores.

Features	Tiempo
500	152.7596502
510	152.9794829
520	155.7426789
530	161.8194029
540	161.1337485
550	164.4228561
560	169.1000063
570	169.8414304
580	170.69999
590	179.3110511
600	177.452018

Cuadro 5.1: 1 CPU - Algoritmo tradicional

Uno de los puntos a tener en cuenta es el considerable tiempo de ejecución del algoritmo DBSCAN tradicional. Dado que la complejidad computacional del algoritmo es cuadrática respecto a la cantidad de puntos y lineal en términos de la cantidad de features de los vectores, esto resulta

Features	Tiempo
500	121.3949702
510	120.2926292
520	122.6374366
530	127.6725171
540	127.6683056
550	129.7267559
560	135.9673066
570	134.2256711
580	136.3997064
590	142.2652395

Cuadro 5.2: 32 CPU - Algoritmo tradicional

en tiempos de ejecución significativos, especialmente cuando se trabaja con conjuntos de datos de gran tamaño. Esto podría representar una limitación considerable para la aplicación práctica del algoritmo en escenarios donde se manejan grandes volúmenes de datos o donde la rapidez en el procesamiento es crucial.

Respecto a la paralelización, se observó una ligera diferencia en los tiempos de ejecución entre las configuraciones de 1 y 32 procesadores. Es importante mencionar que, dado que el algoritmo DBSCAN implementado no es inherentemente paralelizable, estas diferencias no son atribuibles a una mayor eficiencia debido a la ejecución paralela del algoritmo mismo. En cambio, estas diferencias podrían explicarse por factores externos a la implementación del algoritmo, tales como la gestión de recursos del sistema operativo, la asignación de cargas de trabajo o la variabilidad en el rendimiento de los recursos de hardware disponibles.

Otras posibles justificaciones para la existencia de diferencias en los tiempos de ejecución podrían ser considerar la interferencia de otras operaciones del sistema, variaciones en la disponibilidad de recursos o incluso diferencias en la gestión del caché y la memoria. Además, podrían existir consideraciones específicas del entorno de ejecución, como la política de planificación de procesos o la presencia de operaciones de entrada/salida que podrían influir en los tiempos de ejecución observados.

A pesar de su eficacia en la identificación precisa de clústeres, la aplicabilidad práctica del algoritmo DBSCAN tradicional puede verse limitada por sus tiempos de ejecución, especialmente en contextos donde se manejan conjuntos de datos extensos.

Como ejemplo, un conjunto de 250 mil registros (lo cual no es una gran cantidad) con una cantidad similar de características, se podría extrapolar a un tiempo de ejecución de aproximadamente 87 días con una capacidad de procesamiento similar.

5.5.2.2. Implementación paralela

Para la implementación paralela del algoritmo DBSCAN, se ha decidido paralelizar la creación de la tabla de distancias precomputadas. Esta decisión está fundamentada en la simplicidad de dicha operación y su independencia respecto a la parte central del algoritmo, garantizando que los resultados obtenidos sean consistentes con la versión no paralelizada del algoritmo.

A continuación, se presentan los resultados obtenidos al ejecutar la implementación paralela utilizando un número variable de CPUs y distintas cantidades de features. Se han llevado a cabo

experimentos utilizando 1, 2, 4, 8, 16 y 32 CPUs, y variando el número de features desde 500 hasta 600.

Features	Tiempo
500	181.2816339
510	168.3276336
520	171.4375029
530	164.4409602
540	164.3033311
550	166.918715
560	173.7001095
570	172.2119291
580	176.3183513
590	181.6901131
600	180.8183506

Cuadro 5.3: 1 CPU - Algoritmo paralelizable

Features	Tiempo
500	82.33624125
510	83.5046103
520	84.63398027
530	87.83015156
540	87.65335035
550	89.05920839
560	91.99756575
570	92.34052348
580	94.30036926
590	97.56543851
600	97.66807175

Cuadro 5.4: 2 CPU - Algoritmo paralelizable

El análisis de los resultados obtenidos sugiere que la paralelización de la precomputación de la tabla de distancias es efectiva y exhibe un comportamiento casi lineal respecto al número de CPUs utilizadas. Esto indica que el problema es altamente paralelizable, lo que se traduce en una mejora significativa en los tiempos de ejecución con el incremento del número de CPUs.

Sin embargo, se debe prestar especial atención al uso de memoria, ya que la tabla de distancias debe mantenerse en memoria de manera permanente durante la ejecución del algoritmo. Este factor podría ser una limitante en casos donde se manejen grandes volúmenes de datos, ya que el consumo de memoria crece cuadráticamente con la cantidad de puntos y linealmente con el número de vectores.

En conclusión, la implementación paralela del algoritmo DBSCAN muestra una mejora significativa en términos de tiempo de ejecución, especialmente cuando se aumenta el número de CPUs. Esto sugiere que las estrategias de paralelización son cruciales para mejorar la eficiencia del algoritmo y hacerlo más aplicable a conjuntos de datos grandes en el ámbito de Clustering.

Features	Tiempo
500	52.77620435
510	46.80118299
520	43.67142034
530	45.88610029
540	45.57037854
550	46.58180237
560	47.92657208
570	48.02668142
580	48.85209298
590	50.50580001
600	50.49035382

Cuadro 5.5: 4 CPU - Algoritmo paralelizable

Features	Tiempo
500	21.74976277
510	22.05914688
520	22.29407501
530	23.18193579
540	22.97108769
550	23.80257273
560	24.27532339
570	24.72277331
580	24.68763256
590	26.06412601
600	25.78064537

Cuadro 5.6: 8 CPU - Algoritmo paralelizable

Features	Tiempo
500	13.49019217
510	13.50927806
520	13.6498735
530	14.2407198
540	14.32641077
550	14.64692235
560	14.98647475
570	15.05640173
580	15.61402845
590	15.60389996
600	15.92114902

Cuadro 5.7: 16 CPU - Algoritmo paralelizable

Features	Tiempo
500	10.1103704
510	8.268512487
520	8.193612337
530	8.378351212
540	8.505381584
550	8.501050234
560	9.087508917
570	8.895025253
580	9.170488119
590	9.32729888
600	9.779723167

Cuadro 5.8: 32 CPU - Algoritmo paralelizable

Processes	600 feat time	Speedup
1	180.8183506	1
2	97.66807175	1.851355794
4	50.49035382	3.581245463
8	25.78064537	7.013724752
16	15.92114902	11.3571169
32	9.779723167	18.48910725

Cuadro 5.9: Tabla de datos de rendimiento

5.5.3. Cálculo de distancia en Computación Cuántica

El cálculo de distancia para vectores de diversas magnitudes confirmó la expectativa teórica de que dicha operación puede ser realizada con una complejidad de O(log n), donde n denota la dimensión de los vectores en cuestión.

Esta optimización podría tener un impacto significativo en una amplia gama de futuras aplicaciones que buscan capitalizar las ventajas de la Computación Cuántica para abordar problemas altamente combinatorios, ya que se identificó en los algoritmos estudiados en el presente trabajo que el cuello de botella es el cálculo de distancias entre puntos.

Sin embargo, las limitaciones actuales de la plataforma IBM plantean desafíos para la implementación de aplicaciones híbridas que buscan beneficiarse tanto de la Computación Clásica como de la cuántica. Específicamente, los tiempos de espera en cola impiden la eficacia de estas aplicaciones. Aunque la posibilidad de implementar un cálculo de distancia cuántico existe, en la práctica, algoritmos como k-means requieren múltiples de estos cálculos durante su ejecución. Por lo tanto, cualquier mejora en el tiempo de ejecución se ve contrarrestada por los tiempos de espera en cola, limitando así la utilidad práctica de esta optimización.

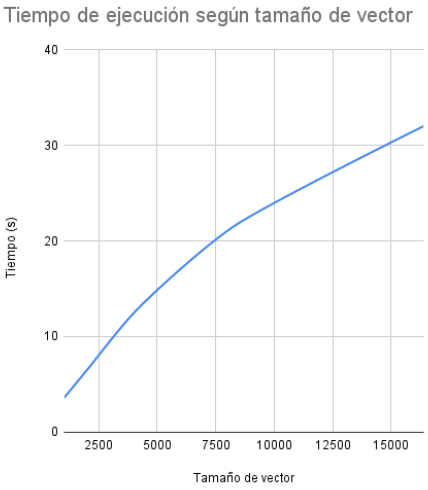


Figura 5.8: Comparación del tiempo de ejecución para diferentes tamaños de vector en un procesador cuántico de 127 qubits.

Charla con expertos

La Facultad de Ingeniería de la Universidad de Montevideo (FIUM), en colaboración con Quantum-South e IEEE Broadcast Technology Society Uruguay, organizó una serie de conferencias impartidas por la Profesora Jingbo Wang de la University of Western Australia. Estas charlas brindaron la oportunidad de conocer más sobre la Computación Cuántica y su estado actual, así como de interactuar con expertos en el campo.

Profesora Jingbo Wang
La Profesora Wang (PhD. Quantum Physics) es Profesora y Jefa del Departamento de Física en la University of Western Australia (UWA). También es Directora del QUISA (Quantum Information, Simulation and Algorithms) Research Hub alojado en la misma universidad, que busca fomentar la colaboración y el espíritu empresarial entre el personal académico, estudiantes de investigación, gobierno y socios industriales para desarrollar soluciones cuánticas innovadoras. En QUISA lidera un grupo activo en investigación de información cuántica, simulación y desarrollo de algoritmos. Su equipo ha sido pionero en algoritmos basados en caminatas cuánticas para resolver problemas de importancia práctica, como el análisis de redes complejas, la optimización combinatoria y el procesamiento de información cuántica.

Wang obtuvo su doctorado en el Departamento de Física y Matemáticas Físicas de la University of Adelaide, Australia. Es autora de los siguientes libros:

- Josh Izaac y Jingbo Wang, Computational Quantum Mechanics, Springer (2018)
- Kia Manouchehri y Jingbo Wang, Physical Implementation of Quantum Walks, Springer (2014)

A lo largo de su carrera académica, la Profesora Wang ha recibido varios premios y reconocimientos, como el Vice-Chancellor's Award for Research Mentorship, la membresía como Fellow en el Australian Institute of Physics y el WA Dennis Moore Award.

Durante las conferencias, la Profesora Wang destacó que la Computación Cuántica aún está "scratching the surface" y que existe un gran "hype" en torno a la misma que debería ser reducido. Explicó que la mayoría de los investigadores, como ella, buscan cómo podría servir la Computación Cuántica asumiendo que se tiene una computadora cuántica perfecta, aunque aún se está lejos de alcanzarla. Sin embargo, Wang mostró que la inversión en el área es muy alta, y comentó que, aunque no hay una computadora cuántica perfecta, igualmente se ha avanzado mucho, especialmente en la cantidad de qubits.

Rafael Sotelo y Laura Gatti, cofundadores de Quantum-South
Quantum-South es la primera startup cuántica en América Latina y se ha convertido en la primera empresa latinoamericana en unirse a la IBM Quantum Network.

Rafael Sotelo es también el Director Regional de América Latina de IEEE Consumer Technology Society - Board of Governors y director de investigación de la Universidad de Montevideo. Durante las charlas, Sotelo mencionó que existen pocos algoritmos cuánticos que actualmente sean mejores que los de la computación clásica.

Laura Gatti, cofundadora y directora de desarrollo de Quantum-South, tiene un doctorado en Computación Cuántica de la Universidad Politécnica de Madrid. En conversaciones con ella, Gatti expresó su acuerdo con la opinión de Wang sobre el "hype" en la Computación Cuántica. Comentó que muchas empresas están publicando papers "sin valor" para obtener recursos e inversión, en lugar de contribuir al avance real de la Computación Cuántica. Gatti también mencionó que la decoherencia en las máquinas cuánticas es demasiado alta y que ha estado esperando durante muchos años un avance significativo en el campo de la Computación Cuántica.

Conclusiones

En el presente Proyecto de Grado, se logra cumplir con la realización del Estado del Arte de Clustering y de Computación Cuántica. Se alcanza una primera aproximación a la Computación Cuántica, evaluando su aplicabilidad en la optimización de los problemas altamente combinatorios, en especial en algoritmos de Clustering, respondiendo a la pregunta de si la Computación Cuántica puede o no mejorar el rendimiento de los algoritmos de Clustering. A lo largo de este estudio, se han destacado los avances significativos en programación paralela, que actualmente se presenta como una solución más accesible y económica para problemas paralelizables en comparación con la Computación Cuántica.

Los resultados obtenidos indican que, aunque la Computación Cuántica ofrece frameworks para la resolución eficiente de ciertos tipos de problemas, para los problemas de Clustering en particular, dichos marcos de trabajo no están suficientemente desarrollados. Además, la programación en Computación Cuántica resultó ser un desafío mayor al anticipado, con una curva de aprendizaje empinada y desafíos técnicos considerables, como la gestión de memoria y las operaciones básicas, que se ven agravados por el alto nivel de abstracción que aún falta en la Computación Cuántica.

Inicialmente, se aspiró a desarrollar una implementación completamente cuántica para la resolución de un algoritmo de Clustering. Sin embargo, esta meta se ajustó a un enfoque más investigativo ante los retos encontrados, centrándose en un análisis teórico más que en una aplicación práctica.

Los aportes de esta investigación incluyen una revisión de dos de los algoritmos de Clustering más prominentes y una evaluación crítica de la Computación Cuántica como herramienta potencial para su optimización. Se destaca la utilidad del Estado del Arte desarrollado como punto de partida para futuras investigaciones en el campo cuántico y se presentan alternativas viables para la mejora del rendimiento de algoritmos de Clustering.

Desde una perspectiva autocrítica, se reconoce la subestimación inicial de las dificultades inherentes a la programación en Computación Cuántica para aplicaciones más complejas que las operaciones básicas. Esto resalta la importancia de establecer expectativas realistas y la necesidad de una mayor familiarización con las herramientas de bajo nivel disponibles.

Por limitaciones de tiempo y recursos, no fue posible explorar en profundidad la implementación de oráculos cuánticos para partes específicas de los algoritmos de Clustering, lo cual podría haber enriquecido la investigación con aplicaciones prácticas y orientado futuros trabajos en el área. La investigación deja un camino abierto para los siguientes estudios que podrían beneficiarse del trabajo teórico realizado.

Las charlas y conversaciones con los expertos en Computación Cuántica, como la Profesora Jingbo Wang, Rafael Sotelo y Laura Gatti, ofrecieron una perspectiva valiosa sobre el estado actual

y los desafíos en el campo, alineándose a las conclusiones presentadas en este Proyecto de Grado.

Si bien la Computación Cuántica ha avanzado en ciertos aspectos, como el aumento en la cantidad de qubits, aún queda mucho por descubrir y mejorar, principalmente en la decoherencia de estos sistemas, antes de poder aprovechar todo su potencial. La inversión en el área es alta, pero es importante mantener un enfoque crítico y realista sobre las expectativas y el progreso en la Investigación Cuántica.

Trabajo futuro

Mediante el estudio en este Proyecto de Grado, se han visto perspectivas únicas y grandes desafíos en la intersección de la Computación Cuántica y los algoritmos de Clustering altamente combinatorios. A pesar de los obstáculos y los resultados inconclusos, la investigación ha dejado en claro que el dominio cuántico posee un potencial sin explotar que podría cambiar la comprensión y aplicación de los algoritmos de Clustering. Esta sección está dedicada a presentar las direcciones prometedoras para el futuro trabajo en este campo.

8.1. Enfoque en algoritmos exclusivamente cuánticos

La emergencia y evolución constante de la Computación Cuántica ha generado muchas posibilidades en el diseño y la aplicación de algoritmos. Un área que aún le queda mucho por desarrollar y promete resultados revolucionarios es el de los algoritmos intrínsecamente cuánticos, diseñados específicamente para aprovechar las propiedades cuánticas y operar de manera óptima en plataformas cuánticas.

Es esencial recalcar la distinción entre los algoritmos que simplemente se han adaptado para funcionar en un contexto cuántico y aquellos que han sido concebidos desde cero con una estructura cuántica en mente. Estos últimos tienen el potencial de ser radicalmente más eficientes y poderosos, ya que están diseñados para aprovechar propiedades como la superposición y el entrelazamiento, fundamentales en el mundo cuántico.

8.2. Análisis, implementación y evaluación en diversas plataformas

Una exploración exhaustiva de diversas plataformas es fundamental para enriquecer y diversificar las perspectivas y estrategias de implementación de algoritmos de Clustering cuánticos.

- Un análisis detallado y una exploración profunda dentro de AWS Braket es esencial. Su arquitectura permite acceder a múltiples tecnologías cuánticas, facilitando una evaluación comprensiva de diferentes enfoques y métodos para implementar y optimizar algoritmos de Clustering cuánticos. La flexibilidad y las capacidades de esta plataforma podrían ser

catalizadores significativos en la búsqueda de soluciones cuánticas más efectivas y robustas para problemas de Clustering.

- Google Cirq, con su enfoque en computadoras cuánticas basadas en puertas, ofrece otra oportunidad para la exploración. Cirq facilita la construcción y el diseño de circuitos cuánticos, lo que podría ser esencial para desarrollar y probar nuevas implementaciones de algoritmos de Clustering.
- Azure Quantum de Microsoft, con su integración fluida con otras tecnologías y servicios en la nube de Microsoft, ofrece un entorno amplio para desarrollar y desplegar soluciones cuánticas. La capacidad de conectar fácilmente los recursos y servicios cuánticos con las capacidades computacionales clásicas y de Inteligencia Artificial podría facilitar un enfoque más híbrido y colaborativo en la resolución de problemas de Clustering.

8.3. Desarrollo de APIs para Qiskit

El desarrollo y refinamiento de APIs específicas para facilitar y mejorar la implementación de algoritmos cuánticos de Clustering en Qiskit es una dirección valiosa. Esto no solo promovería la accesibilidad y la aplicabilidad de los métodos desarrollados, sino también fomentaría una colaboración más amplia y contribuciones de la comunidad científica.

Bibliografía

[Aaronson, 2013] Aaronson, S. (2013). *Quantum Computing since Democritus*. Cambridge University Press, 1st edition.

[Amazon Web Services,] Amazon Web Services. What is quantum computing? https://aws.amazon.com/es/what-is/quantum-computing/. Accessed: 2023-11-2.

[Analytics Vidhya,] Analytics Vidhya. An introduction to clustering and different methods of clustering. https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/. Accessed: 2023-11-17.

[Arthur et al., 2011] Arthur, D., Manthey, B., and Röglin, H. (2011). Smoothed analysis of the k-means method. *J. ACM*, 58(5).

[Arthur and Vassilvitskii, 2007] Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA. Society for Industrial and Applied Mathematics.

[Barenco et al., 1996] Barenco, A., Berthiaume, A., Deutsch, D., Ekert, A., Jozsa, R., and Macchiavello, C. (1996). Stabilisation of quantum computations by symmetrisation.

[Bergholm et al., 2022] Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Ahmed, S., Ajith, V., Alam, M. S., Alonso-Linaje, G., AkashNarayanan, B., Asadi, A., Arrazola, J. M., Azad, U., Banning, S., Blank, C., Bromley, T. R., Cordier, B. A., Ceroni, J., Delgado, A., Matteo, O. D., Dusko, A., Garg, T., Guala, D., Hayes, A., Hill, R., Ijaz, A., Isacsson, T., Ittah, D., Jahangiri, S., Jain, P., Jiang, E., Khandelwal, A., Kottmann, K., Lang, R. A., Lee, C., Loke, T., Lowe, A., McKiernan, K., Meyer, J. J., Montañez-Barrera, J. A., Moyard, R., Niu, Z., O'Riordan, L. J., Oud, S., Panigrahi, A., Park, C.-Y., Polatajko, D., Quesada, N., Roberts, C., Sá, N., Schoch, I., Shi, B., Shu, S., Sim, S., Singh, A., Strandberg, I., Soni, J., Száva, A., Thabet, S., Vargas-Hernández, R. A., Vincent, T., Vitucci, N., Weber, M., Wierichs, D., Wiersema, R., Willmann, M., Wong, V., Zhang, S., and Killoran, N. (2022). Pennylane: Automatic differentiation of hybrid quantum-classical computations.

[Castelvecchi, 2023] Castelvecchi, D. (2023). Google's quantum computer hits key milestone by reducing errors. *Nature*, 1. Accessed: 2023-06-06.

[Celebi, 2014] Celebi, M. E. (2014). *Partitional Clustering Algorithms*. Springer Publishing Company, Incorporated.

[City of Chicago,] City of Chicago. Crimes - 2001 to present. https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2/data. Accessed: 2023-5-3.

[Condon and Morse, 1931] Condon, E. U. and Morse, P. M. (1931). Quantum mechanics of collision processes i. scattering of particles in a definite force field. *Rev. Mod. Phys.*, 3:43–88.

[Coursera,] Coursera. What is data analysis? definition, types, and examples. https://www.coursera.org/articles/what-is-data-analysis-with-examples. Accessed: 2023-11-17.

[D-Wave Systems, a] D-Wave Systems. About d-wave. https://www.dwavesys.com/company/about-d-wave/. Accessed: 2023-7-1.

[D-Wave Systems, b] D-Wave Systems. Customer success stories. https://www.dwavesys.com/learn/customer-success-stories/. Accessed: 2023-8-3.

[D-Wave Systems, c] D-Wave Systems. Ocean documentation. https://docs.ocean.dwavesys.com/en/stable/index.html. Accessed: 2023-6-3.

[Dean, 2021] Dean, W. (2021). Computational Complexity Theory. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2021 edition.

[DeepAI,] DeepAI. K-means.

[Deutsch, 1985] Deutsch, D. (1985). Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 400:97–117.

[Docentes de Aprendizaje Automatico - Fing,] Docentes de Aprendizaje Automatico - Fing. Aprendizaje no supervisado. https://eva.fing.edu.uy/pluginfile.php/291366/mod_folder/content/0/9%20-%20Aprendizaje%20No%20Supervisado.pdf. Accessed: 2023-3-07.

[Esmaelnejad et al., 2010] Esmaelnejad, J., Habibi, J., and Yeganeh, S. (2010). A novel method to find appropriate for dbscan.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

[Farhi and Neven, 2016] Farhi, E. and Neven, H. (2016). A quantum algorithm for training neural networks using low-depth circuits. *arXiv preprint*.

[Fortnow, 2009] Fortnow, L. (2009). The status of the p versus np problem. *Commun. ACM*, 52(9):78–86.

[Google Quantum AI, 2018] Google Quantum AI (2018). Cirq.

[Google Team, 2022] Google Team (2022). What is clustering? https://developers.google.com/machine-learning/clustering/overview. Accessed: 2023-5-21.

[Hashemifar,] Hashemifar, S. Kmeans vs dbscan.

[HDBSCAN Developers,] HDBSCAN Developers. Performance and scalability. https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html. Accessed: 2023-8-20.

[Honeywell,] Honeywell. Honeywell quantum solutions. https://www.honeywell.com/us/en/company/quantum. Accessed: 2023-11-7.

[IBM,] IBM. Artificial intelligence (ai). https://www.ibm.com/topics/artificial-intelligence. Accessed: 2023-5-8.

[IBM, 2019] IBM (2019). Ibm quantum computing. Online; accessed May 15, 2023.

[IonQ,] IonQ. About ionq. https://ionq.com/company. Accessed: 2023-11-7.

[Jay Gambetta, 2023] Jay Gambetta, M. S. (2023). Charting the course to 100,000 qubits. Accessed: 2023-06-06.

[Katayama and Satoh, 1997] Katayama, N. and Satoh, S. (1997). The sr-tree: An index structure for high-dimensional nearest neighbor queries. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, SIGMOD '97, page 369–380, New York, NY, USA. Association for Computing Machinery.

[KDnuggets, 2020] KDnuggets (2020). Dbscan clustering algorithm in machine learning. https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html. Accessed: 2023-11-2.

[Killoran et al., 2019] Killoran, N., Izaac, J., Quesada, N., Bergholm, V., Amy, M., and Weedbrook, C. (2019). Strawberry fields: A software platform for photonic quantum computing. *Quantum*, 3:129.

[Kopczyk, 2018] Kopczyk, D. (2018). Quantum machine learning for data scientists.

[Live Science,] Live Science. What is quantum mechanics? https://www.livescience.com/33816-quantum-mechanics-explanation.html. Accessed: 2023-11-2.

[Lloyd et al., 2013] Lloyd, S., Mohseni, M., and Rebentrost, P. (2013). Quantum algorithms for supervised and unsupervised machine learning.

[MathWorld, 2021] MathWorld (2021). Np-hard problem. Accessed: 2023-06-11.

[Matplotlib Developers,] Matplotlib Developers. Matplotlib: Visualization with python. https://matplotlib.org/. Accessed: 2023-7-10.

[Microsoft, 2023] Microsoft (2023). Building a quantum-safe future. Accessed: 2023-06-06.

[Microsoft Quantum, 2018] Microsoft Quantum (2018). Q#. https://docs.microsoft.com/en-us/quantum/.

[Nielsen and Chuang, 2010] Nielsen and Chuang (2010). *Quantum Computation and Quantum Information*. Cambridge University Press., Cambridge, United Kingdom.

[Objectivity,] Objectivity. Nisq devices: Are they quantum computers or not quite yet? https://www.objectivity.co.uk/blog/nisq-devices-are-they-quantum-computers-or-not-quite-yet/. Accessed: 2023-10-17.

[OECD.AI Policy Observatory,] OECD.AI Policy Observatory. Adjusted rand index (ari). https://oecd.ai/en/catalogue/metrics/adjusted-rand-index-%28ari%29. Accessed: 2023-8-10.

[Oxford Quantum Circuits,] Oxford Quantum Circuits. About oxford quantum circuits. https://oxfordquantumcircuits.com/story. Accessed: 2023-11-7.

[Qiskit Developers, 2017] Qiskit Developers (2017). Quantum information science kit. https://qiskit.org/.

[Qiskit Team,] Qiskit Team. Qiskit documentation. https://qiskit.org/documentation/. Accessed: 2023-5-10.

[QuEra,] QuEra. About quera. https://www.quera.com/about. Accessed: 2023-11-7.

[QuTiP Developers, 2011] QuTiP Developers (2011). Quantum toolbox in python. http://qutip.org/.

[Research, 2018] Research, M. (2018). Quantum computing for computer scientists. YouTube video.

[Rigetti Computing, a] Rigetti Computing. About rigetti computing. https://www.rigetti.com/about-rigetti-computing. Accessed: 2023-7-12.

[Rigetti Computing, b] Rigetti Computing. Pyquil: A python library for quantum programming using quil. https://github.com/rigetti/pyquil. Accessed: 2023-7-12.

[Schuld and Petruccione, 2018] Schuld, M. and Petruccione, F. (2018). *Supervised Learning with Quantum Computers*. Springer.

[Schumacher, 1995] Schumacher, B. (1995). Quantum coding. *Phys. Rev. A*, 51:2738–2747.

[Scikit-learn Developers,] Scikit-learn Developers. Scikit-learn: Machine learning in python. https://scikit-learn.org/stable/. Accessed: 2023-7-10.

[TechTarget,] TechTarget. What is classical computing? https://www.techtarget.com/whatis/definition/classical-computing. Accessed: 2023-11-1.

[The Quantum Insider, 2023] The Quantum Insider (2023). Price of a quantum computer. https://thequantuminsider.com/2023/04/10/price-of-a-quantum-computer/. Accessed: 2023-10-10.

[The Verge, Chaim Gartenberg, 2017] The Verge, Chaim Gartenberg (2017). D-wave is now shipping its new 15 million, 10-foot tall quantum computer. https://www.theverge.com/circuitbreaker/2017/1/25/14390182/d-wave-q2000-quantum-computer-price-release-date/. Accessed: 2023-10-10.

[Thornton, 2006] Thornton, M. A. (2006). A tutorial on reed-muller logic and its synthesis. *Journal of Multiple-Valued Logic and Soft Computing*, 12(5-6):429–455.

[Wikipedia, 2023] Wikipedia (2023). Np-hard. Accessed: 2023-06-11.

[Xandau,] Xandau. About xanadu. https://www.xanadu.ai/about. Accessed: 2023-11-7.

[Xu and Wunsch, 2005] Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678.

[Yanofsky and Mannucci, 2008] Yanofsky, N. S. and Mannucci, M. A. (2008). *Quantum Computing for Computer Scientists*. Cambridge University Press, USA, 1 edition.

[YourGenome,] YourGenome. What is bioinformatics and how do we use it? https://www.yourgenome.org/facts/what-is-bioinformatics-and-how-do-we-use-it. Accessed: 2023-11-5.

Anexo

8.4. Anexo 1 - Código

8.4.1. Cálculo de distancia cuántico

```
import concurrent.futures
import json
import math
import os
import numpy as np
from dotenv import load_dotenv
from qiskit import ClassicalRegister, QuantumCircuit, QuantumRegister, transpile
from qiskit_ibm_provider import IBMProvider

load_dotenv()

IBM_TOKEN = os.getenv("IBM_TOKEN")

# Load your IBMQ account
provider = IBMProvider(token=IBM_TOKEN)

# Define the backend
backend = provider.get_backend("ibm_brisbane")

# Define the vector sizes
size = 8192
shots = 1024

def normalize_point(point):
    norm = 0
    for coordinate in point:
        norm += coordinate**2
    norm = norm**0.5
    normalized_point = []
    for coordinate in point:
```

```
        normalized_point.append(coordinate / norm)
    return {"vector": normalized_point, "norm": norm}

# Define the quantum circuit
def create_circuit(vector1, vector2):
    A_norm_point = normalize_point(vector1)
    B_norm_point = normalize_point(vector2)

    A_norm = A_norm_point["norm"]
    B_norm = B_norm_point["norm"]

    Z = A_norm*2 + B_norm*2

    reg_size = math.ceil(
        math.log(len(A_norm_point["vector"]) + len(B_norm_point["vector"]), 2)
    )
    print(f"Reg_size: {reg_size}")

    phi = [A_norm / math.sqrt(Z), -B_norm / math.sqrt(Z)]
    psi = []

    for j in range(len(A_norm_point["vector"])):
        psi.append((A_norm_point["vector"][j] / math.sqrt(2)))
        psi.append((B_norm_point["vector"][j] / math.sqrt(2)))

    q1 = QuantumRegister(1, name="q1")
    q2 = QuantumRegister(1, name="q2")
    q3 = QuantumRegister(reg_size, name="q3")
    c = ClassicalRegister(1, name="c")
    qc = QuantumCircuit(q1, q2, q3, c)

    qc.initialize(phi, q2[0])
    qc.initialize(psi, q3[0:reg_size])

    qc.h(q1[0])
    qc.cswap(q1[0], q2[0], q3[0])
    qc.h(q1[0])
    qc.measure(q1, c)

    return qc

# Function to calculate distance
def calculate_distance(vector1, vector2, backend):
    qc = create_circuit(vector1, vector2)
    transpiled_qc = transpile(qc, backend, optimization_level=3)
    job = backend.run(transpiled_qc, shots=shots)
```

```
    result = job.result()
    execution_time = result.time_taken

    # Get the measurement results
    counts = result.get_counts()

    # Calculate the distance
    distance = np.sqrt(counts.get("0", 0) / sum(counts.values()))
    return distance, execution_time

# Create a thread pool
with concurrent.futures.ThreadPoolExecutor() as executor:
    # Store the execution times

    print("Calculating distance for vector size = ", size)
    # Generate random vectors
    vector1 = np.random.rand(size)
    vector2 = np.random.rand(size)
    future = executor.submit(calculate_distance, vector1, vector2, backend)
    distance, execution_time = future.result()

    # Save the result to an existing file called times.json
    with open("times.json", "r") as f:
        data = json.load(f)
        data[f"{size}_{shots}shots"] = execution_time

    with open("times.json", "w") as f:
        json.dump(data, f, indent=4)
```

8.4.2. DBSCAN

8.4.2.1. Implementación tradicional

```
def dbscan(DB, distMatrix, eps, minPts):
    C = 0
    labels = [None] * len(DB)

    for P_idx, _P in enumerate(DB):
        if labels[P_idx] is not None:
            continue

        N = rangeQuery(DB, distMatrix, P_idx, eps)

        if len(N) < minPts:
            labels[P_idx] = "Noise"
            continue
```

```
            C += 1
            labels[P_idx] = C
            seed_set = set(N) - {P_idx}

            while seed_set:
                Q_idx = seed_set.pop()

                if labels[Q_idx] == "Noise":
                    labels[Q_idx] = C

                if labels[Q_idx] is not None:
                    continue

                labels[Q_idx] = C
                N = rangeQuery(DB, distMatrix, Q_idx, eps)

                if len(N) >= minPts:
                    seed_set.update(N)

    return labels

def rangeQuery(DB, distMatrix, Q_idx, eps):
    """
      Devuelve todos los puntos P en DB tales que dist(P, Q) <= eps
    """
    N = []
    for P_idx, P in enumerate(DB):
        if distMatrix[Q_idx][P_idx] <= eps:
            N.append(P_idx)
    return N

def generate_distances(DB, distFunc):
    """
      Genera una matriz de distancias para el conjunto de datos DB
      usando la función de distancia distFunc
    """
    N = len(DB)
    D = [[0] * N for _ in range(N)]
    for i in range(N):
        for j in range(i + 1, N):
            D[i][j] = distFunc(DB[i], DB[j])
            D[j][i] = D[i][j]
    return D
```

8.4.2.2. Implementación paralela

```
def dbscan(DB, distMatrix, eps, minPts):
    C = 0
    labels = [None] * len(DB)

    for P_idx, _P in enumerate(DB):
        if labels[P_idx] is not None:
            continue

        N = rangeQuery(DB, distMatrix, P_idx, eps)

        if len(N) < minPts:
            labels[P_idx] = "Noise"
            continue

        C += 1
        labels[P_idx] = C
        seed_set = set(N) - {P_idx}

        while seed_set:
            Q_idx = seed_set.pop()

            if labels[Q_idx] == "Noise":
                labels[Q_idx] = C

            if labels[Q_idx] is not None:
                continue

            labels[Q_idx] = C
            N = rangeQuery(DB, distMatrix, Q_idx, eps)

            if len(N) >= minPts:
                seed_set.update(N)

    return labels

def rangeQuery(DB, distMatrix, Q_idx, eps):
    N = []
    for P_idx, P in enumerate(DB):
        if distMatrix[Q_idx][P_idx] <= eps:
            N.append(P_idx)
    return N

def generate_distances(DB, distFunc):
    N = len(DB)
    D = [[0] * N for _ in range(N)]
```

```
    for i in range(N):
        for j in range(i + 1, N):
            D[i][j] = distFunc(DB[i], DB[j])
            D[j][i] = D[i][j]
    return D

def distances(DB, distFunc, i):
    N = len(DB)
    D = [0] * N
    for j in range(i + 1, N):
        D[j] = distFunc(DB[i], DB[j])
    return D

def generate_distances_mp(DB, distFunc, n_jobs=-1):
    from multiprocessing import Pool
    from functools import partial

    N = len(DB)
    D = [[0] * N for _ in range(N)]

    with Pool(n_jobs) as p:
        D = p.map(partial(distances, DB, distFunc), range(N))
    return D

# Example usage
def euclidean_distance(a, b):
    #start = datetime.now()
    res = sum((x - y)**2 for x, y in zip(a, b))**0.5
    #end = datetime.now()
    return res

"""
#Sample usage:

data = ...
eps = ...
minPts = ...
processes = ...

distMatrix = generate_distances_mp(
  data,
  euclidean_distance,
  n_jobs=processes
)

labels = dbscan(data, distMatrix, eps, minPts)
```

```
"""
```

8.4.2.3. Test

```
import unittest
from db_scan_multiprocessing import dbscan, rangeQuery, generate_distances, \
  euclidean_distance

class DBScanTests(unittest.TestCase):

    def test_dbscan(self):
        DB = [[0], [1], [4], [10], [11]]
        distMatrix = [[0, 1, 4, 10, 11],
                      [1, 0, 3, 9, 10],
                      [4, 3, 0, 6, 7],
                      [10, 9, 6, 0, 1],
                      [11, 10, 7, 1, 0]]

        labels = dbscan(DB, distMatrix, eps=2, minPts=2)
        self.assertEqual(labels, [1, 1, "Noise", 2, 2])

    def test_rangeQuery(self):
        DB = [[0], [1], [4], [10], [11]]
        distMatrix = [[0, 1, 4, 10, 11],
                      [1, 0, 3, 9, 10],
                      [4, 3, 0, 6, 7],
                      [10, 9, 6, 0, 1],
                      [11, 10, 7, 1, 0]]
        neighbours = rangeQuery(DB, distMatrix, 0, eps=2)
        self.assertEqual(neighbours, [0, 1])

    def test_generate_distances(self):
        DB = [[0], [1], [4]]
        distMatrix = generate_distances(DB, euclidean_distance)
        self.assertEqual(distMatrix, [[0, 1, 4], [1, 0, 3], [4, 3, 0]])

if __name__ == '__main__':
    unittest.main()
```

8.4.3. k-means

8.4.3.1. cluster.py

```
import random
from typing import List
```

```
import matplotlib.cm as cm


class Cluster:
    def __init__(self, id: int, centroid: List) -> None:
        """
        Class to represent a centroid

        Parameters
        ----------
        id: int
            Cluster id
        centroid : List
            Initialization point of the centroid
        """
        self.id = id
        self.centroid = centroid
        self.points_assigned = []
        self.color = cm.rainbow(random.random())

    def update(self) -> None:
        """
        Update the centroid point
        """
        new_center = [sum(sub_list) / len(sub_list) for sub_list in zip(*self.points_assigned)]
        self.centroid = new_center

    def get_distance(self, distance: callable) -> float:
        """
        Get the distance between the centroid and the points assigned to it

        Parameters
        ----------
        distance : callable[[List, List], float]
            Distance function

        Returns
        -------
        float
            Distance
        """
        dist = 0
        for point in self.points_assigned:
            dist += distance(self.centroid, point)
        return dist
```

8.4.3.2. distances.py

```
from typing import List

import numpy as np

from subroutines.quantum_distance_calculator import calculate_distance


def build_distance(distance_name: str):
    """
    Build the distance function

    Parameters
    ----------
    distance_name : str
        Distance name

    Returns
    -------
    function
        Distance function
    """
    if distance_name == "classic":
        return classic_distance
    elif distance_name == "quantum":
        return quantum_distance
    elif distance_name == "vectorized":
        return vectorized_distance
    else:
        raise Exception("Distance not implemented")


def vectorized_distance(point1: np.ndarray, point2: np.ndarray) -> float:
    dist = np.linalg.norm(point1 - point2)
    return dist


def classic_distance(point1: List, point2: List) -> float:
    """
    Get the distance between two n-dimensional points

    Parameters
    ----------
```

```
    point1 : List
        Point 1
    point2 : List
        Point 2

    Returns
    -------
    float
        Distance

    Raises
    ------
    Exception
        The points must be of equal dimension
    """
    if len(point1) != len(point2):
        raise Exception("The points must be of equal dimension")

    dist = 0
    for i in range(len(point1)):
        sq_diff = (point1[i] - point2[i]) ** 2
        dist += sq_diff

    return dist**0.5


def quantum_distance(point1: List, point2: List):
    """
    Get the quantum distance between two n-dimensional points using the quantum distance calcula

    Parameters
    ----------
    point1 : List
        List of n-dimensional coordinates of point 1.
    point2 : List
        List of n-dimensional coordinates of point 2.

    Returns
    -------
    float
        The quantum distance between point1 and point2.

    Raises
    ------
    Exception
        The points must be of equal dimension.
```

```
    """
    if len(point1) != len(point2):
        raise Exception("The points must be of equal dimension")

    distance, _ = calculate_distance(point1, point2)

    return distance
```

8.4.3.3. kmeans.py

```
import random
import sys
from itertools import chain
from multiprocessing import Pool
from typing import List

from kmeans.cluster import Cluster
from kmeans.distances import build_distance


def assign_points_chunks(args) -> List:
    X_train_chunk, clusters, distance = args
    assigned_points = []
    for point in X_train_chunk:
        dist = sys.float_info.max
        for cluster in clusters:
            point_dist = distance(cluster.centroid, point)
            if point_dist < dist:
                dist = point_dist
                new_centroid = cluster
        assigned_points.append((point, new_centroid.id))
    return assigned_points


def point_cluster_distance(point_cluster):
    """Calculate the distance between a point and a cluster

    Parameters
    ----------
    point_cluster : tuple
        Tuple containing a point and a list of clusters

    Returns
    -------
    int
```

```
            The id of the nearest cluster
        """
        points, clusters, distance = point_cluster[0], point_cluster[1], point_cluster[2]
        res = []
        for point in points:
            dist = sys.float_info.max
            for cluster in clusters:
                point_dist = distance(cluster.centroid, point)
                if point_dist < dist:
                    dist = point_dist
                    new_centroid = cluster
            res.append(new_centroid.id)
        return res


    class KMeans:
        def __init__(
            self,
            n_clusters: int,
            distance_name: str,
            max_iter: int = 300,
            n_process: int = 1,
        ) -> None:
            """
            Class to implement the KMeans algorithm

            Parameters
            ----------
            n_clusters : int
                Cluster's number
            distance_name : str
                Name of the distance function defined in src/kmeans/distances.py
            max_iter : int
                Maximum number of iterations
            """
            self.k = n_clusters
            self.max_iterations = max_iter
            self.clusters = []
            self.distance = build_distance(distance_name)
            self.n_processes = n_process

        def init_clusters(self, X_train: List) -> List:
            """Initialize the clusters

            Parameters
            ----------
```

```
    X_train : List
        Data

    Returns
    -------
    List
        List of clusters
    """
    best_sse = sys.float_info.max
    for i in range(10):
        if self.n_processes == 1:
            self.clusters: List[Cluster] = [
                Cluster(id=i, centroid=random.choice(X_train))
                for i in range(self.k)
            ]
        else:
            self.clusters: List[Cluster] = [
                Cluster(id=i, centroid=random.choice(X_train[0]))
                for i in range(self.k)
            ]

        # Assign points to the nearest centroid
        self.clusters = self.assign_points(X_train, self.clusters)

        sse = 0
        for centroid in self.clusters:
            sse += centroid.get_distance(self.distance)
        if sse < best_sse:
            best_sse = sse
            best_centroids = self.clusters

    return best_centroids

def assign_points(self, X_train: List, clusters: List) -> List:
    """Assign points to the nearest centroid

    Parameters
    ----------
    X_train : List
        Data
    clusters : List
        List of clusters

    Returns
    -------
    List
```

```
            List of clusters
        """
        if self.n_processes == 1:
            for cluster in clusters:
                cluster.points_assigned = []
            for point in X_train:
                dist = sys.float_info.max
                for cluster in clusters:
                    point_dist = self.distance(cluster.centroid, point)
                    if point_dist < dist:
                        dist = point_dist
                        new_centroid = cluster
                new_centroid.points_assigned.append(point)

            return clusters
        else:
            with Pool(self.n_processes) as pool:
                # Use starmap to call assign_points_helper on each chunk in parallel
                results = pool.map(
                    assign_points_chunks,
                    [(points, self.clusters, self.distance) for points in X_train],
                )
                results = list(chain.from_iterable(results))

            # Clear points from clusters
            for cluster in clusters:
                cluster.points_assigned = []

            # Assign points to clusters based on the results
            for point, cluster_id in results:
                clusters[cluster_id].points_assigned.append(point)

            return clusters

    def fit(self, X_train: List) -> "KMeans":
        """Fit the model

        Parameters
        ----------
        X_train : List
            Data

        Returns
        -------
        KMeans
            Model
```

```
    """
    # Initialize clusters
    self.clusters = self.init_clusters(X_train)

    for i in range(self.max_iterations):
        previous_centroids = [cluster.centroid for cluster in self.clusters]

        # Clear points from clusters
        for cluster in self.clusters:
            cluster.points_assigned = []

        # Assign points to the nearest centroid
        self.clusters = self.assign_points(X_train, self.clusters)

        # Update centroid position to the new points
        for cluster in self.clusters:
            cluster.update()

        # Check if the centroids have changed
        centroids_changed = False
        for cluster, previous_centroid in zip(self.clusters, previous_centroids):
            # if not np.array_equal(cluster.centroid, previous_centroid):
            if cluster.centroid != previous_centroid:
                centroids_changed = True
                break

        if not centroids_changed:
            print("Centroids not changed after: ", i)
            break

    return self

def predict(self, X_train: List) -> List[Cluster]:
    """Get the cluster of each point

    Returns
    -------
    List[Cluster]
        List of clusters
    """
    if self.n_processes == 1:
        labels = []

        for point in X_train:
            dist = sys.float_info.max
            for cluster in self.clusters:
```

```
                point_dist = self.distance(cluster.centroid, point)
                if point_dist < dist:
                    dist = point_dist
                    new_centroid = cluster
            labels.append(new_centroid.id)

        return labels
    else:
        with Pool(self.n_processes) as pool:
            # Use the map function to distribute the computation across the processes
            labels = pool.map(
                point_cluster_distance,
                [(point, self.clusters, self.distance) for point in X_train],
            )
            labels = list(chain.from_iterable(labels))
        return labels

def fit_predict(self, X_train: List) -> List[Cluster]:
    """Fit the model and get the cluster of each point

    Returns
    -------
    List[Cluster]
        List of clusters
    """
    self.fit(X_train)
    return self.predict(X_train)
```