UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías "Politécnico"

Benchmarking de Hamiltonian Path Problem y Travelling Salesman Problem mediante Ant Colony Optimization con un generador de números pseudo aleatorios y un generador cuántico de números aleatorios

Proyecto de Investigación

Valery Alexandra Villarruel Mosquera

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniera en Ciencias de la Computación

Quito, 20 de diciembre de 2022

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías "Politécnico"

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Benchmarking de Hamiltonian Path y Travelling Salesman Problem mediante Ant Colony Optimization con un generador de números pseudo aleatorios y un generador cuántico de números aleatorios.

Valery Alexandra Villarruel Mosquera

Calificación:

Nombre del profesor, Título académico **Daniel Riofrío, PhD**

Firma del profesor:

Quito, 20 de diciembre de 2022

© DERECHOS DE AUTOR

Nombres y apellidos: Valery Alexandra Villarruel Mosquera

Código: 207042

Cédula de identidad: 1723268858

Lugar y fecha: Quito, 20 de diciembre de 2022

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

Encontrar soluciones a problemas NP-Completos se relaciona coloquialmente con encontrar una aguja en un pajar debido a su complejidad que, en consecuencia, produce algoritmos de tiempo exponencial. En particular, una estrategia para encontrar "buenas soluciones" a estos problemas es evaluar las posibles soluciones generadas al azar y medir la calidad de cada una de ellas en cada intento. Pero, la aleatoriedad no se puede implementar en las computadoras clásicas, por lo que se emula a través de algoritmos que crean números pseudoaleatorios. Para estos problemas, utilizamos un algoritmo de optimización metaheurística, es decir, Ant Colony Optimization, con diferentes generadores de números aleatorios: pseudoaleatorios y cuánticos aleatorios. Comparamos el tiempo de convergencia y el costo total de diferentes configuraciones de gráficos (diferentes niveles de complejidad, es decir, 50, 100, 150 y 200 nodos) bajo ambos generadores de números aleatorios. Observamos que, en general, cuando se utilizan generadores de números cuánticos-aleatorios se logra una convergencia más rápida y se obtienen mejores resultados.

Palabras clave: TSP, ACO, HPP, NP-Complete, Números aleatorios, Números pseudo aleatorios, Computación cuántica.

ABSTRACT

Finding solutions to NP-Complete problems are colloquially related to finding a needle in a haystack due to their complexity which in consequence yield exponential time algorithms. One strategy to find “good solutions” to these problems is to evaluate potential solutions generated at random and measure the quality of each in every attempt. But randomness cannot be implemented in classical computers, hence it is emulated through algorithms that create pseudo-random numbers. In this study, we analyze two NP-complete problems in graphs, the Traveling Salesman and the Hamiltonian Path Problems.For finding solutions to these problems, we use a metaheuristic optimization algorithm, i.e. Ant Colony Optimization, with Different random number generators: pseudo-random and quantum-random.We compare convergence time and overall cost of different graph set-ups (different complexity levels, i.e. 50, 100, 150 and 200 nodes) under both random number generators.We observed that , in general, when quantum-random number generators are used faster convergence is achieved and better results are obtained.

Key words: ACO, TSP, HPP, Random numbers, Pseudo random numbers, Quantum computing.

Contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Los problemas NP-Complete se consideran de orden exponencial al ser difíciles de resolver con computación clásica. Estos problemas se basan en transformaciones polinomiales reducibles que pueden ser solucionados mediante algoritmos que tomen decisiones aleatorias convirtiéndolos en verificables en tiempo exponencial. Al encontrar una solución para un problema de esta clase, el mismo método servirá para solucionar todos los problemas que pertenezcan a NP-Complete. Los 2 problemas para analizar son Travelling Salesman, que permite encontrar el camino óptimo y de menor costo dado un grafo G y Hamiltonian Path, que buscará la mejor ruta en un grafo G. La estrategia metaheurística que se empleará para solucionar ambos problemas NP-Complete, será Ant Colony Optimization, quien se encargará de tomar decisiones aleatorias en base a generaciones numéricas aleatorias y pseudo aleatorias. Las decisiones serán tomadas mediante la construcción de hormigas artificiales, quienes construirán un camino incremental en base a la probabilidad de elegir un nodo mediante un modelo de rastro de feromonas. El uso de diferentes generadores de números aleatorios permitirá medir el tiempo de convergencia y el costo total de encontrar un camino óptimo para un grafo G. Se optó por Ant Colony Optimization, como algoritmo metaheurístico de solución, debido a que se basa en suposiciones aleatorias entre el conjunto de posibles soluciones.

Por otro lado, para la toma aleatoria de decisiones se plantea un generador de números aleatorios cuánticos mediante el uso de qrng, librería pública de IBM, y pseudoaleatorios usando la librería de Python random. Además, al mencionar la verdadera aleatoriedad se resalta el cumplimiento de todas las propiedades que poseen los números aleatorios.

OBJETIVO GENERAL

Encontrar soluciones a problemas NP-Complete mediante la aplicación de algoritmos metaheurísticos.

OBJETIVOS ESPECÍFICOS

- Utilizar generadores de números pseudo aleatorios y compararlos con generadores de números aleatorios.
- Solucionar el Travelling Salesman Problem mediante la aplicación del algoritmo Ant Colony Optimization (ACO) para encontrar la ruta mas corta y eficiente que tome llegar a un listado de destinos específicos.
- Solucionar el Hamiltonian Path Problem mediante el empleo del algoritmo Ant Colony Optimizatin (ACO) para determinar si un camino hamiltoniano existe en un grafo específico.
- Comparar las soluciones de los problemas mencionados bajo generadores de números aleatorios cuánticos y pseudo aleatorios.

MARCO TEÓRICO

1. Problemas NP-Complete

El concepto de NP-Complete fue introducido mediante el teorema de Cook-Levin en 1971, donde los científicos informáticos debatían la resolución de problemas de detección en una máquina Turing, a partir de esto, se concluyó que NP-Complete son los problemas más complejos del conjunto NP. Una máquina de Turing o máquina abstracta escribe en su cinta una función o pregunta para un Oracle o caja negra que se encarga de establecer si un elemento certifica o falla una prueba, este analizará problemas de decisión ya que es capaz de recibir un número polinomial de preguntas para generar, comparar y evaluar una solución a un problema dado en tiempo polinomial, en otras palabras, esta es la forma más efectiva de saber si un problema es NP-Complete. Actualmente, esta familia de problemas se delimitan NP-Complete ya que poseen la propiedad de corregir todo el problema en una misma clase de complejidad a mayor tiempo polinomial. Particularmente, si cualquier de estos problemas NP-Complete se puede resolver en tiempo polinomial, entonces, es posible llegar a encontrar una solución para todos los problemas de esta familia. La definición matemática para NP-Complete establece que un problema de decisión X es completo si X ϵ NP-Complete, por lo tanto, si otro problema Y ϵ NP es reducible a X en un tiempo polinómico (Pérez, 2020) (Sipser, 2013). La reducción en tiempo polinomial establece que cuando un problema A se reduce a un problema B, la solución de B puede emplearse para resolver A. La teoría de NP-Complete se basa en transformación polinomial, esta es una función que consiente en modificar la representación de un problema A1 a otro problema A2 empleando un algoritmo de decisión determinístico rápido que comprueba una solución única. Una de las formas de solucionar estos problemas es usar parametrización y metaheurísticas, es decir, averiguar un algoritmo que se base en suposiciones aleatorias entre el conjunto de posibles opciones de solución. Las operaciones aleatorias son la mejor opción cuando no es viable explorar todas las posibilidades o se tiene un alto costo en tiempo computacional, ya que, la aleatoriedad permite que los bits aleatorios se empleen como entradas auxiliares para guiar el comportamiento del algoritmo, de esta forma, se garantiza la reducción en el tiempo de búsqueda de soluciones óptimas, además, permitirá la disminución de la probabilidad al fallo del algoritmo (Pérez, 2020). A continuación, se detallan dos ejemplos de problemas NP-Complete a analizar a lo largo de este documento y se hace énfasis en un algoritmo metaheurístico para solucionar esta clase de problemas.

A. Hamiltonian Path Problem

Es un problema de decisión que consiste en tener un grafo no dirigido del cual se requiere encontrar una ruta desde s a t, donde s es el nodo de inicio y t el nodo final, que garantice que cada nodo se visitará exactamente una vez y se retornará al nodo inicial, es decir, se centra en establecer si existe una trayectoria Hamiltoniana. Un ciclo o viaje hamiltoniano es un proceso simple que contiene a cada vértice de un grafo G, también es considerado como una sucesión de aristas adyacentes, donde no se pasa dos veces por la misma arista. Al ser un problema NP-Complete se conoce que la posible solución sea en tiempo polinomial en base al número de vértices del grafo (Sipser, 2013). Cabe recalcar que, si el grafo es considerado bipartito con un número impar de nodos, además, si dos aristas no son incidentes con un vértice, no será un camino de tipo hamiltoniano (Rose, 2007). Así mismo, mientras más bordes posea un grafo aumentará la probabilidad de tener un ciclo hamiltoniano (Rosen, 2007). No existe un criterio que permita determinar si hay caminos Hamiltonianos en un grafo. Sin embargo, hay dos teoremas importantes que permiten establecer la existencia de un circuito simple:

- Teorema de Dirac: Si G es un grafo simple con n vértices $n \geq 3$ tal que el grado de cada vértice en G es al menos n/2, entonces G tiene un circuito Hamiltoniano (Cormen, 2009).
- Teorema de Ore: Si G es un grafo simple con n vértices $n \geq 3$ tal que $\deg(u) + \deg(v) \geq n$ para cada par de nodos adyacentes u y v en G, entonces G tiene un circuito Hamiltoniano (Cormen, 2009).

No obstante, estos teoremas resultaron no ser indispensables para establecer un camino hamiltoniano en un grafo o instaurar que no existe un ciclo que tiene una complejidad de tiempo exponencial. Esto se debe a que un grafo puede tener un camino Hamiltoniano que no cumpla con las condiciones establecidas en los teoremas. Por ejemplo, en base al siguiente grafo de la Ilustración 1, se puede trazar su respectivo camino hamiltoniano mostrado en Ilustración 2.

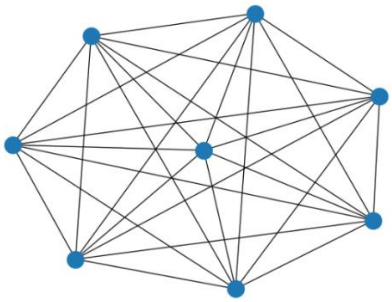

Figura 1: Grafo no dirigido

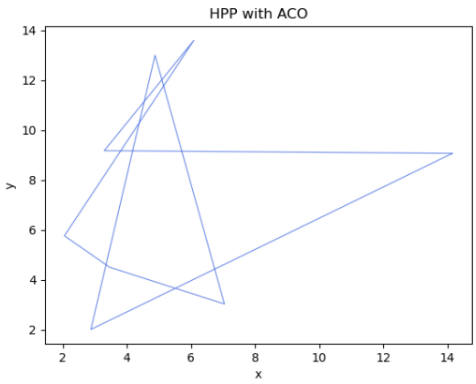

Figura 2: Solución de encontrar un ciclo Hamiltoniano dado un grafo G

Por lo que, al hacer una búsqueda de la mejor ruta pasando por cada nodo de un grafo G. Es necesario garantizar que los nodos no se repitan. Además, se debe tomar en cuenta que el orden de localizar si existe un ciclo o camino Hamiltoniano es $O(1.251^n)$ aproximadamente ya que, en caso de encontrar un camino, se desconoce su comportamiento. Evidentemente se requerirá de gran tiempo computacional, por lo tanto, una solución frente a este problema es aplicar un algoritmo metaheurístico basado en la toma aleatoria de decisiones hasta hallar la respuesta óptima.

B. Travelling Salesman Problem

Este problema es una reducción a un ciclo Hamiltoniano, donde se tiene un grafo no dirigido con bordes ponderados, para el que se desea visitar cada vértice exactamente una vez y retornar al vértice de partida. El grafo se puede definir de manera que el número de vértices correspondan a las ciudades que se quiere visitar, los bordes corresponderán a las rutas y las distancias serán los pesos de la ruta. Sin embargo, este proceso tiene la restricción de que el peso total de sus aristas o costo del viaje sea mínimo. La manera simple de resolver este problema es examinar

todos los caminos Hamiltonianos, es decir, una ruta que pasa por cada nodo del grafo exactamente una vez y termine en su nodo de partida, para elegir el que tenga menor costo. No obstante, al intentar buscar un camino hamiltoniano se origina un problema de minimización que inicia y termina en un vértice específico, después de visitar cada nodo exactamente una vez. Asimismo, si no existe un camino entre dos vértices, se requiere añadir un borde lo adecuadamente largo que ultimará el grafo sin perturbar a la ruta óptima. Sin embargo, al existir $(n-1)!$ posibles circuitos que visitar, además, de la dificultad de encontrar un circuito Hamiltoniano y tomando en cuenta que un camino hamiltoniano también se puede recorrer en orden inverso, se necesitará demasiado tiempo computacional realizar una búsqueda exhaustiva para un camino hamiltoniano con una longitud mínima dentro de las opciones, por lo que, una solución viable será aplicar un algoritmo que tome decisiones aleatorias para explorar todas las posibles rutas.

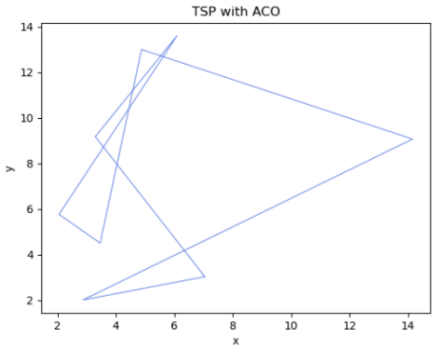

Figura 3: Solución de Travelling Salesman Problem, donde la línea negra representa el camino más corto dado un grafo G

Para los problemas antes mencionados, se requiere encontrar un algoritmo metaheurístico, es decir, un conjunto de conceptos que pueden ser empleados para resolver problemas con una cantidad mínima de transformaciones, mismas que implementen decisiones en función de pruebas aleatorias. Por lo tanto, se ha elegido Ant Colony Optimization, ya que al ser una metaheurística poblacional combinatoria, toma varias decisiones aleatorias que permite explorar las posibles opciones de soluciones.

C. Ant Colony Optimization

Las hormigas reales encuentran el camino más corto desde su nido hasta fuentes de alimento. Así mismo, son capaces de adaptarse al entorno y producir cambios en él

para encontrar la mejor ruta, esto es posible debido a las feromonas que poseen. Existen varios algoritmos ACO, sin embargo, a lo largo de la experimentación se empleará Ant System, ya que este algoritmo consiste en encontrar la mejor ruta en un grafo, en base a hormigas artificiales quienes serán las encargadas de construir un camino de manera incremental lo que aumentará la probabilidad de escoger uno de esos nodos, usando un modelo de rastro de feromonas, asociadas a cada borde del grafo, o información histórica respecto a los viajes previos en el espacio de búsqueda. Dicha probabilidad se calcula en función de:

$$p^k ij = \frac{\tau^\alpha ij \ * \ n^\beta ij}{\sum_{\tau il \in N(s^p)} \tau_{il}^\alpha * n_{il}^\beta}$$

Ecuación 1: Fórmula de la probabilidad de que una hormiga elija un nodo

Donde N (s^p) representa el conjunto de bordes (i, l), en el que *l* es un vértice que no ha sido visitado aún. Por otro lado, los parámetros α y β determinan la probabilidad en función de las feromonas. Finalmente, nij representa la distancia que hay entre la ciudad i a la ciudad j, también conocido como tamaño del edge, por lo tanto, está dada por:

$$nij \ = \frac{1}{dij}$$

Ecuación 2: Fórmula para el cálculo del tamaño de cada edge de un grafo G

Por ende, el camino que eligen las hormigas depende de una variable probabilística asociada a la heurística de cada hormiga controlada por los parámetros α y β, mencionados en la ecuación 1, y el rastro de feromonas, en el que, un camino exitoso tendrá mejor probabilidad (Dorigo, 2006). El siguiente vértice por visitar será elegido de manera iterativa mediante un mecanismo estocástico, sesgado por feromonas, entre un listado de nodos no visitados. Las hormigas artificiales modificarán el valor de cada vértice del grafo en tiempo de ejecución (Bautista, 2000). Los parámetros antes mencionados definen la importancia de la heurística asociada o si se dará preferencia a la información histórica disponible en la colonia. El modelo de feromonas funciona como rastro y mientras más intenso sea, existirán más probabilidades de escoger ese camino, por lo que, se debe garantizar la cantidad necesaria de feromonas para evitar convergencias aceleradas, esto se logra mediante

la evaporación de las feromonas por cada iteración, es decir, la evaporación permite que los caminos más frecuentes se vuelvan candidatos mientras que los que poseen poca repetición son descartados. Dicha actualización disminuirá la concentración de feromonas en los bordes, lo cual, hará que las hormigas alienten a las hormigas posteriores a elegir otro borde, es decir, producir soluciones diferentes (Dorigo, 2006). Por otro lado, es necesario buscar a la mejor hormiga n del conjunto de hormigas artificiales, ya que es la encargada de encontrar el mejor camino, debido a que cada hormiga posee un efecto de velocidad de convergencia. Cabe recalcar que, en las primeras iteraciones se otorgará prioridad a las reglas heurísticas y mientras exista mayor experiencia, las reglas disminuirán en peso frente a las decisiones probabilísticas que dependerán del peso histórico del rastro.

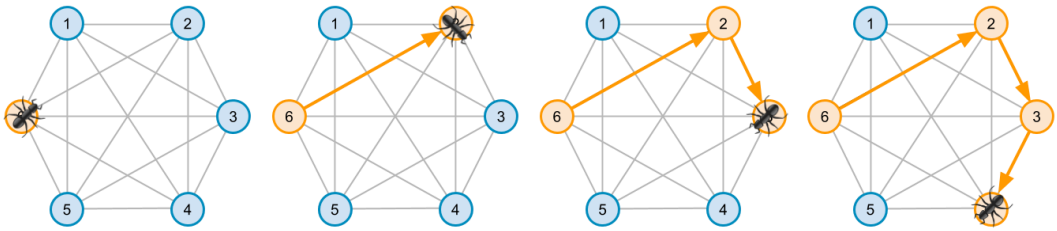

Figura 4: Comportamiento del Ant Colony Optimization

La aplicación de ACO para solucionar Travelling Salesman Problem y Hamiltonian Path Problem será a través de la selección del rastro o camino a seguir a lo largo del grafo en función de las feromonas en los arcos según el orden de solución y empleando reglas heurísticas. Para estos problemas, la secuencia del rastro se colocará en pares de tareas según la posible solución. Las actualizaciones de los rastros permitirán aumentar valores de memoria de la ruta de mejor calidad, por lo que, la eficacia del rastro dependerá del inverso de la calidad de la función objetivo alcanzada en la solución (Bautista, 2000). Al ser ACO, un algoritmo que se basa en suposiciones aleatorias, se pretende garantizar la aleatoriedad de estas mediante la generación de números aleatorios cuánticos y comparar los resultados de usarlos con una generación de números pseudo aleatorios.

2. Generación de números aleatorios

Los ordenadores clásicos, de estado finito, generan números aleatorios mediante algoritmos cuyos cálculos están basados en funciones, por ende, la secuencia resultante es pseudoaleatoria (RNG) predecible, la cual expande de forma determinística una semilla

aleatoria. Por otro lado, mediante la teoría de complejidad de Kolmogrov, se supone que una secuencia RNG puede considerarse aleatoria si se adapta al concepto de cuantificar la aleatoriedad del resultado (Ma, 2022). Por ende, solo es posible garantizar la aleatoriedad verdadera mediante procesos que implican aleatoriedad inherente. De igual manera, por el concepto de Born, no es posible predecir un estado cuántico debido a que es intrínsecamente aleatorio. Sin embargo, gracias a la aleatoriedad inherente en cálculos cuánticos es factible generar números aleatorios. Además, dicha generación debe basarse en cualquier proceso cuántico que rompa la superposición coherente de estados. Los generadores cuánticos de números aleatorios (QRNG) incorporan un sistema que cumple con el efecto cuántico aleatorio de reintegrar la configuración inicial inmediatamente de cada proceso de medición de valores, ya que al tener las mismas condiciones iniciales y la misma forma de medición facilitará valores distintos, lo que, el sistema desarrollará un generador de números aleatorios deseable (Kollmitzer, 2022).

A. Algoritmos clásicos

Los RGN son resultado de la implementación de algoritmos determinísticos a los cuales se les pasa una semilla como parámetro inicial. Estas secuencias periódicas poseen propiedades que puede asemejarlas a secuencias aleatorias ya que no siguen un patrón o regla evidente. Por otro lado, la correlación entre cada elemento es relativamente baja por lo cual aparentan ser diferentes entre sí (Murray, 2022). Para la generación de números pseudoaleatorios se utilizará la librería de Python random ya que proporciona una función básica random(), cada vez que se invoca a esta función se devolverá un valor de entre una secuencia de números determinada, esta secuencia se caracterizará por tener un largo periodo, es decir, se obtendrán varios números antes de que una secuencia se repita.

B. Computación cuántica y generación de números aleatorios

Los QRGN se diferencian de los RNG ya que los QRNG contiene una fuente de entropía cuya salida es esencialmente impredecible y el extractor de aleatoriedad puede ser un algoritmo generador de números aleatorios casi perfectos gracias a la salida de su fuente de entropía. Actualmente, existen algunas versiones de QRNG caracterizados por poseer un costo bajo y una velocidad considerable de generación

(Murray, 2022). Para la generación de números aleatorios en una computadora cuántica es esencial calcular la cantidad de bits aleatorios que se pueden extraer de datos sin procesar, además, se debe evitar la presencia de ruido ya que podría afectar al proceso de extracción. Se probaron varias formas para generar números aleatorios cuánticos, de las cuales algunos resultaron exitosos, como, QuantumRandom es un proyecto que facilita la interacción con un generador de números aleatorios cuánticos de ANU mediante comunicación directa con la API, misma que brinda comandos online para la generación de números aleatorios. El laboratorio de ANU usa campos electromagnéticos en un espacio vacío ya que presentan fluctuaciones aleatorias y amplitud de frecuencias, por lo tanto, al medir cada fluctuación se obtienen números aleatorios con un alto ancho de banda (Qrng, 2022). QRand es un algoritmo que toma como parámetro un número natural n, es decir, n qubits y genera un circuito cuyo resultado de ejecución resulta un número aleatorio en un rango de 0 y $2^n - 1$, donde la probabilidad de medir 0 o 1 se mide por la implementación de compuertas lógicas Hadamard a cada qubit (Li, 2022) y qrng es un generador de números aleatorios cuánticos que usa la API QISKit de IBM con el objetivo de establecer un vínculo con las computadoras cuánticas. El algoritmo, de esta librería de código abierto, funciona mediante un conjunto de compuertas ajustables a n número de qubits, de los cuales su estado está basado en probabilidad de ser 1 o 0, el propósito del algoritmo es modificar el estado de los qubits ya que al leerlo será posible generar números aleatorios (Escánez, 2022), esta sección fue desarrollada en base a la norma ISO/IEC AWI TR 18157 Introduction to quantum computing (ISO, 2022) y apoyada por la norma ISO/IEC DIS 4879 Quantum computing- Terminology and vocabulary (ISO, 2022).

Para el proceso de experimentación se utilizará la librería qrng de IBM ya que requiere un proceso de conexión directa, a través de un token, al hardware proporcionado por IBM, de esta forma, es posible garantizar la verdadera aleatoriedad en el conjunto de números resultantes al hacer uso de un computador cuántico.

MÉTODO EXPERIMENTAL

1. Descripción de la implementación

El código desarrollado para todo el experimento fue escrito en Python versión 3.8. Para la obtención de números pseudo aleatorios se usó la librería random para tener acceso a la configuración de la semilla, para la cual se generó la lista de numérica mediante la función random.uniform, misma a la que se le especifica el rango en el que deben ir los elementos de la lista, en este caso, entre 0 y 1500. Por otro lado, la generación de números aleatorios cuánticos se manejó con la librería de código abierto qrng versión 0.1.3, incluido en el kit de desarrollo de Qiskit creado por IBM. Fue necesaria la instalación de dicha librería mediante el comando "pip install qrng". Para tener acceso a las funciones dentro del kit, primero, es preciso crear una cuenta en la plataforma cuántica de IBM para obtener un token el cual se trazará una conexión desde la máquina principal hacia cualquier computador cuántico público disponible en la plataforma de IBM. Dicha conexión es establecida mediante la función qrng.set_provider_as_IBMQ, la cual recibe como parámetro el token personalizado generado. A continuación, se hace un llamado a la función qrng.set_backend, misma que recibe el nombre de la computadora cuántica a la que se desea realizar la conexión, para este experimento se estableció el enlace con 'imbq_quito' que posee 5 qubits. Finalmente, la generación de la lista aleatoria es factible mediante la función qrng.get_random_float a la cual se le especifica que el rango para el listado numérico irá desde 0 a 1500.

```
def qRng():
    #Qiskit
    rnd = qrng.get-random-int(0, 1500)
    return rnd

def rng():
    num = random.randinit(0, 1500)
    return num

def generateNumber():
    # PSEUDO
    seed = random.seed(10)
        numbers = rng()
    # RANDOM
        numbersQ = qRng()
```

Listing 1: Generación de números aleatorios y pseudo aleatorios

Por otro lado, en cuanto a la implementación del Hamiltonian Path Problem, se debe considerar que, la función de costos es calculada en base a la matriz de adyacencia del grafo, el orden de la matriz está representada por el número de vértices del grafo, se toma en cuenta dos vértices i y j, además, la matriz de adyacencia del grafo $M_{n,n}$, donde si i = j el valor correspondiente a dichos nodos en la matriz M [i, j] será 0, este valor representa el costo de ir desde i hasta j. Mientras que, para cada edge que une 2 vértices se añade 1 al

valor correspondiente de M [i, j] en la matriz. Al implementar grafos no dirigidos la matriz de adyacencia se caracteriza por ser simétrica, es decir, que es una matriz cuadrada cuya forma es igual a su traspuesta. Asimismo, debido a la estructura de los grafos construidos para la experimentación, la matriz adyacente resulta ser binaria, cuya diagonal se encuentra compuesta por valores de 0. Sin embargo, para el desarrollo del Travelling Salesman Problem, la función de costo será calculada en base a la distancia entre cada nodo perteneciente al grafo. Estos nodos al estar representados por coordenadas en un plano facilitan el cálculo de la distancia entre puntos.

De igual manera, para el desarrollo de Ant System, se establecieron funciones generales para el cálculo de la probabilidad de cada nodo en el grafo en función de la feromona, esta probabilidad está dada por la fórmula 1, para la cual, los valores de Alpha, beta y Q fueron asignados como 2, 1 y 10 respectivamente. A continuación, se construyó la función de inicialización para cada hormiga, esta recibe como parámetro el número de hormigas definidas. Para este caso, el número de hormigas será igual a la cantidad de nodos en el grafo. En esta función de inicialización se construirá la lista de nodos iniciales y en base a cada nodo dentro de la lista, se modelan los nodos que no han sido visitados. Después, se emplea la función para seleccionar el camino completo respecto a cada nodo inicial. Posteriormente, se crea la función fitness_estimator para calcular el costo del camino seleccionado previamente. En base a la lista de fitness se actualiza la feromona mediante la fórmula Q/fit. Finalmente, se construye el camino óptimo en base a la lista de fitness ya que esta lista contiene valores correspondientes al costo del camino seleccionado.

HPP función de costo	TSP función de costo

For i in range(len(adjM)):
for j in range (len (adjM [i])):
if (i + 1, j + 1) in edges:
adjM[i][j] = 1
adjM[j][i] = 1
	cost_matr[i,j]=
	math.sqrt((xi -xj) **2 + (yi -yj)**2)

Tabla 1: Diferenciación del cálculo de la función de costo tanto para TSP como HPP

Por último, para el diseño de los grafos, se construyó un grafo no dirigido completamente conectado mediante la librería networkx, donde se puede especificar el número de nodos para los cuales cada uno de sus valores x, y son generados mediante números pseudo aleatorios enteros con un rango de 1.0 a 14.5. Este grafo es pasado a la forma graphml y almacenado en un archivo con extensión ".graphml" para una mejor apreciación de la construcción del grafo.

```
<?xml version='1.0' encoding='utf-8'?>
<graphml xmlns="http://graphml.graphdrawing.org/
    xmlns" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://
    graphml.graphdrawing.org/xmlns http://graphml.
    graphdrawing.org/xmlns/1.0/graphml.xsd"><key id=
    "x" for="node" attr.name="x" attr.type="double"/
    >
<key id="y" for="node" attr.name="y" attr.type="
    double"/>
<graph edgedefault="undirected"><node id="0">
  <data key="y">1.601</data>
  <data key="x">6.735</data>
</node>
<node id="1">
  <data key="y">2.005</data>
  <data key="x">7.303</data>
</node>
<node id="2">
  <data key="y">2.37</data>
  <data key="x">5.828</data>
</node>
<node id="3">
  <data key="y">6.388</data>
  <data key="x">13.317</data>
</node>
<edge source="0" target="1"/>
<edge source="0" target="2"/>
<edge source="0" target="3"/>
<edge source="1" target="2"/>
<edge source="1" target="3"/>
<edge source="2" target="3"/>
</graph></graphml>
```

Listing 2: Diseño del grafo

El código fuente de los algoritmos TSP y HPP resueltos mediante ACO, se puede encontrar en GitHub. https://github.com/alevm569/TSP-HPP-with-ACO.git

2. Configuración experimental

La ejecución del experimento se repitió 10 veces para cada configuración del grafo (diferentes niveles de complejidad, es decir, 50, 100, 150, 200 nodos). En todos los casos se mide el tiempo de convergencia, dado en segundos, a lo largo de la búsqueda del camino óptimo para TSP y HPP, con generadores de números aleatorios y pseudo aleatorios. Se busca comparar los tiempos de convergencia, el menor tiempo y el mejor costo para encontrar una ruta óptima para ambos generadores de números aleatorios.

Los experimentos se realizaron en un solo dispositivo, en este caso un servidor con IP 172.21.67.11 caracterizado por tener 4 cores. Cada núcleo se encargaba de correr un nivel de dificultad diferente. A medida que aumentaba el nivel de dificultad, aumentaba el tiempo de respuesta. Debido al excesivo tiempo requerido para la ejecución, se utilizó un bucle for para repetir la ejecución de la configuración establecida 5 veces. Para este caso, en el nivel de configuración de 200 nodos, el tiempo máximo de respuesta fue de

aproximadamente 27 horas, mientras que, para el nivel más bajo, 50, el tiempo de respuesta fue de aproximadamente 2 horas.

RESULTADOS Y DISCUSIÓN

En este estudio se han logrado solucionar 2 problemas NP-Complete mediante el uso de un algoritmo metaheurístico de decisión, en este caso, Ant Colony Optimization. Este algoritmo es el encargado de elegir la mejor ruta en base a un nodo de inicio y a una lista de todas las combinaciones de nodos que no han sido visitados aún, según el nodo inicial. El mejor camino se elige en base a la probabilidad que posee cada nodo de ser elegido, por cada iteración se acumula la probabilidad de cada nodo, para la cual, si dicha suma supera a un número generado de manera aleatoria o pseudo aleatoria, el nodo al que corresponda la probabilidad que permitió satisfacer la condición es considerado como el nodo siguiente a visitar.

Por otra parte, la configuración mencionada en la sección anterior se puede evidenciar en la Tabla 2. Además, en la Tabla 3 se reporta el promedio, la desviación estándar de la distancia correspondiente al Travelling Salesman Problem, la mejor distancia y el menor tiempo empleando números aleatorios y pseudo aleatorios. Durante la fase de experimentación se han visto las ventajas de utilizar números aleatorios, ya que a los algoritmos les toma menos tiempo encontrar la ruta óptima. A lo largo, de la generación de números, se nota alta correlación numérica (Murray, 2022). Mientras que, para la generación de números aleatorios cuánticos los números poseen baja correlación. Por ende, durante el proceso de decisión aleatoria en el Ant Colony Optimization pasarán varios turnos entre cada uno de los números pseudo aleatorios para que la condición de arquitectura mencionada previamente para este algoritmo se cumpla.

Number of nodes	**Algorithms**	**OverallTime Pseudo Random Numbers**	**Stdev Pseudo Random Numbers**	**Overall Random Numbers**	**Stdev Random Numbers**
50	TSP	1.335,39	205,42	1.324,25	212,53
	HPP	1.351,57	176,25	1.281,78	392,06
100	TSP	4.596,36	588,10	4.479,49	569,47
	HPP	4.620,43	491,85	4.522,48	382,86
150	TSP	10.057,00	683,28	9.570,33	331,90
	HPP	10.177,63	757,48	9.839,70	681,41
200	TSP	1.855.233,82	5.811.493,81	19.074,81	5.355,40
	HPP	17.506,21	581,92	17.455,21	778,57

Tabla 2: Resultados de las 10 iteraciones para algoritmos TSP y HPP mediante ACO. El tiempo fue medido en segundos y se adjunta la desviación estándar

Nodos	Mean PR (s)	Std PR	Mean QR (s)	Std QR	Better d PR	Better d QR
50	331,93	25,73	327,09	28,71	298,06	289,72
100	670,43	27,20	694,00	27,44	667,00	655,09
150	1.029,49	38,76	984,47	39,11	978,98	944,68
200	1.383,92	47,31	1378,44	34,29	1349,76	1320,14

Tabla 3: Reporte del promedio y desviación estándar para la distancia del Travelling Salesman Problem. Además, se añade la mejor distancia y el menor tiempo.

CONCLUSIONES

Al finalizar la investigación se concluyó que la computación cuántica permite resolver problemas exponenciales mediante un buen desempeño. Como es el caso de los problemas de la clase NP-Complete para los cuales se ha demostrado que el uso de algoritmos que se basen en la toma de decisiones aleatorias, en este caso Ant Colony Optimization, permiten solucionarlos. Es así como, se ha encontrado una solución para todos los problemas dentro de la clase NP-Complete (Pérez, 2020). Para este caso, se generaron números aleatorios cuánticos y números pseudo aleatorios, mismos que fueron usados para resolver 2 problemas de la clase NP-Complete, donde se probó que el uso de números aleatorios cuánticos es más eficiente en comparación al uso de números pseudo aleatorios, ya que el promedio del tiempo de convergencia para cada configuración del grafo es más pequeño que el promedio del tiempo de convergencia reportado para el uso de números pseudo aleatorios. Es decir, tanto a Travelling Salesman Problem como Hamiltonian Path Problem les toma menos tiempo en encontrar la mejor ruta dentro de un grafo G, debido a que, al garantizar una verdadera aleatoriedad, donde la correlación entre cada número es realmente baja (Murray, 2022), las decisiones aleatorias tomadas por Ant Colony Optimization no están sesgadas por repeticiones numéricas y tampoco siguen patrones de generación predecibles. Esto es útil, ya que para el

caso de Travelling Salesman Problem se requiere encontrar un camino que posea el menor costo y que tanto Travelling Salesman Problem como Hamiltonian Path Problem calculen el mejor camino en el menor tiempo posible. Como trabajo futuro se buscará mejorar los tiempos de respuesta para ambos algoritmos NP-Complete con la ayuda de la computación cuántica.

REFERENCIAS BIBLIOGRÁFICAS

Anu. *Anu QRNG – Quantum random numbers.* https://qrng.anu.edu.au/

Bautista, J. (2000). *Metaheurística ACO (Ant colony optimization) para la resolución de problemas en líneas de producción* [Pregrado, Universidad Politécnica de Cataluña]. https://upcommons.upc.edu/bitstream/handle/2117/90260/2000.CEIODEF.ACO_pon.pdf.

Cook, S. (1971). La complejidad de los procedimientos de demostración de teoremas. Tercer Simposio Anual de ACM sobre Teoría de la Computación. Recuperado el 12 de octubre de 2022 de https://hmong.es/wiki/NP-complete.

Cormen, T. H. Stein, C., Rivest, R. L. y Leiserson, C. E. (2009). *Introduction to algorithms*. MIT Press.

Dorigo, M. Birattari, M. y Stuzle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, 12.

ISO/IEC AWI TR 18157 Information technology — Introduction to quantum computing.

ISO/IEC DIS 4879 Information technology — Quantum computing – Terminology and vocabulary.

Kollmitzer, S. Schaucer, S. Rass, S y Rainer, B. (2020). *Quantum Random Number Generation*. Springer International Publishing. https://doi.org/10.1007/978-3-319-72596-3.

Li, Y., Fei, Y., Wang, W., Meng, X., Wang, H., Duan, Q. y Ma, Z. (2021). Quantum random number generator using a cloud superconducting quantum computer based on source-independent protocol. *Scientific Reports*, *11*(1). https://doi.org/10.1038/s41598-021-03286-9.

Ma, X., Yuan, X., Cao, Z., Qi, B. y Zhang, Z. (2016). Quantum random number generation. *Npj Quantum Information*, *2*(1). https://doi.org/10.1038/npjqi.2016.21

Murray, L. (2022). *Números aleatorios generación de números y variables pseudoaleatorias.* Facultad de Ciencias Exactas, Ingeniería y Agrimesura, Universidad Nacional de Rosario. https://eva.fing.edu.uy/pluginfile.php/380761/mod_resource/content/1/3_num_aleat_2022.pdf

Escánez, D., Caballero, P. y Fernández, F. (2022). Evolución de la librería QuantumSolver para el desarrollo cuántico. *Xvii recsi*, 94. https://books.google.es/books?hl=es&lr=&id=V4qREAAAQBAJ&oi=fnd&pg=PA94&dq=generacion+de+numeros+aleatorios+cuanticos&ots=YVMjLstTZR&sig=yUqGo-AmlIT2YHW90Pqj6Hnszfw#v=onepage&q&f=false

Pedemonte, M. (2007). *Ant colony optimization* [Pregrado, Universidad de la República de Montevideo]. https://www.colibri.udelar.edu.uy/jspui/bitstream/20.500.12008/3554/1/TR0711.pdf.

Pérez, M. (2020). Teoría de la complejidad computacional tema 5: Problemas NP completos. En *Grupo de investigación en computación natural*. Grupo de investigación en Computación Natural. Recuperado el 12 de octubre de 2022 de https://www.cs.us.es/~marper/docencia/TCC-2019-2020/temas/tema-5-trans.pdf

Python. *Random generación de números pseudoaleatorios.* Documentación Python 3.8. https://docs.python.org/es/3/library/random.html

PyPI. *Qrng.* PyPI. https://pypi.org/project/qrng/.

Rosen, K. H. (2007). *Discrete mathematics and its applications* (7ª ed.). McGraw-Hill Science/Engineering/Math.

Sipser, M. (2013). *Introduction to the theory of computation* (3ª ed.). Cengage Learning.