UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Computación Híbrida para Encontrar Soluciones a Problemas de Optimización Utilizando Algoritmos de Optimización Inspirados en la Biología

Alejandra Victoria Ospina Gracia

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, 14 de diciembre de 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN

DE TRABAJO DE FIN DE CARRERA

Computación Híbrida para Encontrar Soluciones a Problemas de Optimización Utilizando Algoritmos de Optimización Inspirados en la Biología

Alejandra Victoria Ospina Gracia

Daniel Riofrío, PhD

Quito, 14 de diciembre de 2023

© DERECHOS DE AUTOR

Por medio del presente documento certifico que he leído todas las Políticas y Manualesde la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en la Ley Orgánica de EducaciónSuperior del Ecuador.

Nombres y apellidos: Alejandra Victoria Ospina Gracia

Código: 00212243

Cédula de identidad: 1723520449

Lugar y fecha: Quito, 14 de diciembre de 2023

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 recomendacionespresentadas 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UNPUBLISHED DOCUMENT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 QuitoUSFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by theCommittee 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

En este estudio, se abordaron dos problemas fundamentales en ingeniería estructural y aprendizaje automático: la optimización de parámetros en el modelo de comportamiento sísmico EBP-EAB y la selección de características en conjuntos de datos de alta dimensionalidad. Para el primer problema, se empleó el algoritmo ABC, destacando su eficacia en ajustar el modelo EBP-EAB y mejorar la correspondencia con comportamientos observados. La utilización de números cuánticos aleatorios, generados mediante computadoras cuánticas de IBM, demostró convergencia más rápida en la optimización.

En cuanto a la selección de características, se implementó un algoritmo genético basado en la puntuación ReliefF, utilizando tanto números cuánticos aleatorios como pseudoaleatorios. Aunque los resultados mostraron rendimientos similares en términos de métricas de clasificación, la convergencia del algoritmo genético fue más eficiente al emplear números cuánticos aleatorios.

En la comparación directa entre números cuánticos y pseudoaleatorios, se encontró que, en el ajuste del modelo EBP-EAB, ambos condujeron a resultados prácticamente idénticos, pero los números cuánticos favorecieron la convergencia más rápida. Para la selección de características, no se observaron diferencias significativas en el rendimiento final de los clasificadores, aunque la eficiencia en términos de generaciones fue superior con números cuánticos.

Los resultados sugieren que los números cuánticos aleatorios pueden desempeñar un papel crucial en la mejora de la eficiencia de algoritmos de optimización, especialmente en términos de convergencia. La computación híbrida, combinando recursos cuánticos y clásicos, se

presenta como una estrategia prometedora para abordar problemas complejos en la ingeniería y el aprendizaje automático. No obstante, se destaca la necesidad de pruebas adicionales y análisis estadísticos detallados para validar y comprender completamente el impacto de los números cuánticos en diversos contextos y problemas.

ABSTRACT

This study delves into optimizing parameters for the seismic behavior model EBP-EAB and conducting feature selection in high-dimensional datasets, addressing critical challenges in structural engineering and machine learning. We present a robust implementation of the ABC algorithm to optimize the EBP-EAB model, markedly enhancing its alignment with observed behaviors. Additionally, genetic algorithms (GAs) are employed for effective feature selection. Notably, the incorporation of quantum random numbers generated by IBM quantum computers demonstrates accelerated convergence during optimization.

In the domain of feature selection, a genetic algorithm utilizing the ReliefF score is applied with both quantum random and pseudorandom numbers. While classification metrics show similar performances, the genetic algorithm exhibits more efficient convergence with quantum random numbers.

Direct comparisons between quantum and pseudorandom numbers in EBP-EAB model fitting yield nearly identical results, with quantum numbers demonstrating expedited convergence. Although feature selection does not show significant differences in classifier performance, quantum numbers display superior efficiency in terms of generations.

These findings underscore the pivotal role of quantum random numbers in optimizing algorithms, particularly in enhancing convergence efficiency. The concept of hybrid computing, integrating quantum and classical resources, emerges as a promising strategy for addressing intricate problems in engineering and machine learning. Nevertheless, a comprehensive exploration, coupled with detailed statistical analyses, is imperative to fully validate and

comprehend the impact of quantum numbers across diverse contexts and problems. Subsequent refinements to the abstract will be made after a thorough review of the entire document.

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la rápida evolución del mundo de la computación, la computación clásica está empezando a alcanzar sus limitaciones teóricas. Por ejemplo, problemas complejos como el Problema del Viajante de Comercio (TSP), que implica encontrar la ruta más eficiente que visita un conjunto de ciudades y regresa a la ciudad de inicio, han representado desafíos duraderos para las computadoras clásicas durante décadas, remontándose a la década de 1950 (Appleagate et al., 2007). Sin embargo, una nueva ola de innovación nos está impulsando de nuevo al dominio analógico, donde la computación cuántica está tomando el centro del escenario. En particular, al ofrecer mecánicas para superar las limitaciones de la computación clásica. La computación cuántica representa una fusión de la mecánica cuántica y la ciencia de la computación, ofreciendo un alejamiento de la computación clásica con la que estamos familiarizados. En los últimos años, ha surgido como una promesa para superar desafíos que han representado desafíos duraderos para las computadoras clásicas (Feyman, 1982).

La discusión de la computación cuántica comenzó hace décadas, con pioneros como Richard P. Feynman, quien visualizó el potencial de que las computadoras cuánticas pudieran simular sistemas físicos de manera más eficiente que las computadoras convencionales (Feynman, 1982). Esto marcó el fundamento teórico para la computación cuántica, particularmente en campos como la química cuántica y la ciencia de materiales. A medida que la investigación en este campo avanzaba, la atención se centró en el desarrollo de algoritmos diseñados específicamente para las computadoras cuánticas. Los algoritmos de Shor y Grover surgieron como las primeras demostraciones de que la computación cuántica no solo era teórica, sino una realidad práctica. El algoritmo de Shor, por ejemplo, mostró la eficiencia de las computadoras cuánticas en la factorización de grandes enteros, un desarrollo que tiene

implicaciones para los métodos de cifrado como RSA (Shor, 1994). Mientras tanto, el algoritmo de Grover se centró en optimizar la búsqueda de elementos en una lista, logrando mejoras notables en la velocidad en comparación con las contrapartes clásicas (Grover, 1996).

Si bien estos avances apuntaban al vasto potencial de la computación cuántica, el camino hacia las computadoras y los algoritmos cuánticos prácticos sigue siendo desafiante. Los principios de la mecánica cuántica, como la superposición, el entrelazamiento y la interferencia cuántica, dotan a los sistemas cuánticos de una potencia y fragilidad únicas. Estas propiedades hacen que los sistemas cuánticos sean altamente sensibles a las influencias externas, incluidas las fluctuaciones de temperatura incluso menores que pueden introducir errores. Además, la necesidad de corrección de errores introduce complejidad, ya que mantener qubits con tasas de error bajas a menudo requiere una multitud de qubits físicos. El hardware para las computadoras cuánticas también debe diseñarse cuidadosamente, ya que operan en entornos que requieren un control y una gestión de la temperatura precisos (Steane, 1998).

El diseño de algoritmos en el ámbito cuántico presenta otro desafío formidable. Requiere una profunda comprensión de la física y las matemáticas para traducir problemas clásicos en algoritmos cuánticos eficientes. Sin embargo, la escasez de recursos para probar y refinar estos algoritmos complica su implementación práctica (Braun, 2002). Los lenguajes de programación actuales diseñados para la computación cuántica no son de alto nivel, lo que significa que no son tan amigables o abstractos como los lenguajes de programación tradicionales. En cambio, estos lenguajes se centran más en describir los circuitos cuánticos, que son los bloques de construcción fundamentales de la computación cuántica.

A pesar de estos obstáculos, una parte del campo cuántico ha adquirido una importancia considerable, que son los generadores de números aleatorios cuánticos (QRNGs). Los QRNGs son significativos debido a su capacidad para producir números verdaderamente aleatorios, derivados de la impredecibilidad intrínseca de la mecánica cuántica. Estos números aleatorios desempeñan un papel vital en el aumento de la seguridad de los sistemas criptográficos, permitiendo la distribución cuántica de claves (QKD) y asegurando un muestreo aleatorio auténtico en las simulaciones científicas (Herrero-Collantes & Garcia-Escartin, 2016). Los generadores de números aleatorios cuánticos pueden ofrecer una seguridad y aleatoriedad superiores en comparación con los generadores de números pseudoaleatorios clásicos, cuyas salidas son influenciadas por algoritmos o valores semilla, lo que los hace menos confiables para tareas que requieren una verdadera aleatoriedad y seguridad incrementada.

La importancia de los QRNGs se extiende más allá de la seguridad, ya que también son valiosos en el campo de la optimización. Los problemas de optimización y selección de características son desafíos inherentemente complejos que requieren soluciones innovadoras. Los investigadores ya han progresado al mejorar algoritmos como la búsqueda de enjambre de partículas y el optimizador del lobo gris con modificaciones inspiradas en la mecánica cuántica, lo que ha llevado a un mejor rendimiento (Fang et al., 2017). Sin embargo, no ha habido mucha investigación sobre la aplicación práctica de la combinación de enfoques clásicos y cuánticos. La investigación previa ha sido en su mayoría teórica (Imada, 2007). Se han realizado pocos intentos de explorar esta área, por lo que nuestro trabajo es importante para cerrar la brecha entre los algoritmos teóricos y la implementación real (Maucher et al., 2011).

Nuestra exploración se adentra en el ámbito de los algoritmos de optimización inspirados en la biología, destacando su importancia en la resolución eficiente de problemas complejos. Además, investigamos métodos de generación de números aleatorios cuánticos (QRNG). Nuestro estudio tiene como objetivo proporcionar ideas valiosas sobre el impacto de la aleatoriedad cuántica en la optimización, allanando el camino para algoritmos más robustos y eficientes. Esta exploración tiene el potencial de revolucionar el campo de la optimización asistida por computación cuántica. Los QRNG desempeñan un papel crucial en mejorar la calidad de la aleatoriedad en algoritmos de optimización y simulaciones. Esta mejora conduce a soluciones más efectivas en una amplia gama de problemas de optimización, que abarcan desde escenarios con parámetros complejos e inciertos hasta la búsqueda de óptimos globales en espacios de solución de alta dimensionalidad. En consecuencia, los QRNG contribuyen al descubrimiento de soluciones mejoradas y optimizadas para problemas del mundo real a gran escala. Estas aplicaciones van desde la selección de características en aprendizaje automático un diverso espectro de desafíos de optimización encontrados en varias industrias.

ALGORITMOS DE OPTIMIZACIÓN INSPIRADOS EN LA BIOLOGÍA

Algoritmos Genéticos

La implementación de algoritmos genéticos (AG) ha demostrado ser extraordinaria en problemas de optimización. Este algoritmo emula la selección natural y la recombinación genética. Los AG operan en dos fases fundamentales: la selección de individuos de una población para reproducirse y la recombinación de genes entre individuos para producir descendientes nuevos y diversos, con potencial para mutar (Holland, 1992).

Los AG utilizan pruebas de idoneidad para determinar qué individuos avanzarán a la siguiente generación. Aunque en la década de 1950, los AG enfocaban excesivamente la mutación en lugar de la reproducción y la generación de nuevas combinaciones de resultados potenciales, se descubrió que la recombinación genética desempeña un papel vital en la reproducción, lo que impulsa el progreso evolutivo. Al basar un algoritmo en la combinación y mutación de genes, la eficacia de los AG se potencia significativamente (Goldberg, 1989).

En el pseudocódigo siguiente, se representa cómo operan los AG. Los AG representan características como bits binarios (donde 1 indica la presencia de una característica y 0 indica su ausencia). La población inicial se genera aleatoriamente, y una función de idoneidad evalúa la aptitud de cada individuo. Los individuos más aptos son seleccionados, y el proceso de reproducción combina cadenas de manera innovadora, generando soluciones más sofisticadas.

```
Inicializar población
Repetir por un número de generaciones o épocas:
     Evaluar la aptitud de cada individuo en la población
     seleccionar individuos para la reproducción (mediante
     métodos como torneo, ruleta, etc.)
     Aplicar operadores genéticos (cruce y mutación) para
     generar nuevos individuos
     Reemplazar la población anterior con la nueva generación de
     individuos
     Actualizar el contador de generaciones hasta que se cumpla
     un criterio de parada (por ejemplo, un número máximo de
     generaciones o una convergencia deseada).
```

Los AG sobresalen en la exploración de problemas complejos para identificar regiones de mayor oportunidad. Sin embargo, hay que recalcar la importancia de encontrar un equilibrio entre la explotación y la exploración. Mientras que la explotación se centra en refinar soluciones existentes, la exploración implica tomar riesgos para descubrir una solución potencialmente superior (Holland, 1992).

Colonia Artificial de Abejas

El algoritmo de la Colonia Artificial de Abejas (ABC) emula el comportamiento de las abejas. Hay tres grupos de abejas: abejas empleadas, abejas observadoras y abejas exploradoras. Las abejas observadoras actúan como espectadoras, esperando tomar decisiones sobre qué fuente de alimento elegir. Las abejas empleadas representan a las abejas que visitan fuentes de alimento que han descubierto previamente, mientras que las abejas exploradoras realizan búsquedas aleatorias.

El algoritmo comienza inicializando las abejas y colocando las abejas empleadas en fuentes de alimento en la memoria y las abejas observadoras en las mismas fuentes de alimento. Este proceso se repite hasta encontrar la solución deseada. Las posiciones de las fuentes de alimento se inicializan de manera aleatoria por las abejas. Las abejas empleadas regresan a la

colmena y comparten información, lo que provoca que otras abejas empleadas se muevan hacia fuentes de alimento mejores. Si una abeja exploradora encuentra una fuente de alimento mejor, la mayoría de las abejas empleadas cambian a esa fuente.

Matemáticamente, las fuentes de alimento representan posiciones de solución, y las abejas comienzan con posiciones aleatorias. El néctar corresponde a la función de aptitud, y se generan nuevas fuentes de alimento de manera aleatoria. Después de que las abejas empleadas completan su búsqueda, comparten información con las abejas observadoras, quienes evalúan la información. Si una abeja empleada encuentra una fuente de alimento con más néctar, cambia su posición en la memoria y olvida la anterior (selección codiciosa). Las abejas observadoras eligen fuentes de alimento según valores probabilísticos asociados a cada fuente, p_i, calculados mediante la siguiente expresión,

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}$$

donde fit_i es el valor de aptitud de la solución i evaluada por su abeja empleada, que es proporcional a la cantidad de néctar de la fuente de alimento en la posición *i* y *SN* es el número de fuentes de alimento, que es igual al número de abejas empleadas.

Para producir una posición de fuente de alimento candidata a partir de la antigua, el ABC utiliza la siguiente expresión:

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj})$$

donde $k \in \{1, 2, \ldots, BN\}$ y $j \in \{1, 2, \ldots, D\}$ son índices elegidos al azar. Aunque k se determina al azar, debe ser diferente de *i*. φ_{ij} es un número aleatorio entre [−1, 1]. Controla la producción de una posición de fuente de alimento vecina alrededor de x_{ij} y la modificación representa la comparación de las posiciones de fuente de alimento vecinas visualmente por la abeja. Muestra que a medida que disminuye la diferencia entre los parámetros de x_{ij} y x_{kj}, la

perturbación en la posición x_{ij} también disminuye.

El número de abejas es igual al número de fuentes de alimento en el algoritmo, con esto el algoritmo ABC se ha comparado con algoritmos genéticos, PS-EA y PSO utilizando diversas funciones de aptitud como Griewank, Rastrigin, Rosenbrock, Ackley y Schwefel.

Los resultados mostraron que el algoritmo ABC tuvo un rendimiento significativamente mejor que los algoritmos genéticos y otros algoritmos para funciones como Griewank y Ackley, especialmente para dimensiones más bajas. Sin embargo, a medida que aumentaba la dimensionalidad, los algoritmos genéticos funcionaban mejor. El algoritmo ABC demostró su eficiencia en la optimización de problemas multivariables y multimodales al converger hacia el mínimo global y escapar de los mínimos locales (Karaboga y Basturk, 2007).

GENERACIÓN DE NÚMEROS ALEATORIOS Y PSEUDOALEATORIOS

Generadores cuánticos de números aleatorios

En los últimos años, el campo de la computación cuántica ha presenciado avances notables, lo que ha llevado a la exploración de fenómenos cuánticos para una multitud de aplicaciones. Entre estas aplicaciones, los Generadores Cuánticos de Números Aleatorios (QRNG, por sus siglas en inglés) han surgido como un área destacada de investigación y desarrollo. Los QRNG aprovechan la imprevisibilidad inherente de la mecánica cuántica para generar números verdaderamente aleatorios, con implicaciones potenciales en una amplia gama de dominios (Abbott, Calude y Svozil, 2010).

Técnicas

Existen varias técnicas para implementar QRNGs, cada una basada en fenómenos cuánticos únicos. Los QRNG basados en fotones explotan las propiedades cuánticas de los fotones, como la polarización y los tiempos de llegada (Stanco et al., 2020). Los QRNG basados en polarización utilizan divisores de haces, filtros de polarización y detectores para medir los estados de polarización de los fotones individuales. Los QRNG basados en estados de vacío miden los intervalos de tiempo entre la detección de fotones sucesivos, aprovechando la incertidumbre cuántica en los tiempos de llegada de los fotones (Gabriel et al., 2010).

Los QRNGs de entrelazamiento cuántico aprovechan el fenómeno del entrelazamiento cuántico, donde las propiedades de dos o más partículas se correlacionan. Estos dispositivos utilizan pares de partículas entrelazadas, miden una propiedad de una de las partículas (por

ejemplo, el espín) y generan resultados aleatorios para la medición de la otra partícula. El entrelazamiento garantiza que los resultados de la medición estén correlacionados de una manera que no se puede predecir de antemano (Herrero-Collantes y Garcia-Escartin, 2016).

Los QRNGs de túnel cuántico se basan en el principio del túnel cuántico, donde las partículas pueden pasar a través de barreras energéticas que son clásicamente insuperables (Bernardo-Gavito et al., 2017, p. 1). Estos QRNGs utilizan el túnel de electrones a través de una barrera potencial para generar números aleatorios. La imprevisibilidad surge de la naturaleza inherentemente probabilística de los eventos de túnel cuántico (Herrero-Collantes y Garcia-Escartin, 2016).

Los QRNGs de ruido de fase cuántica explotan las fluctuaciones en la fase de la luz láser. Estos dispositivos miden el ruido de fase de la luz, que es inherentemente incierto debido a las fluctuaciones cuánticas en el campo electromagnético. Las variaciones en la fase de la luz se traducen en números aleatorios (Lei et al., 2020).

IBM quantum

IBM Quantum, una división dedicada al avance de la computación cuántica, explora las complejidades de la física cuántica para desarrollar computadoras cuánticas de vanguardia. Estas máquinas operan a temperaturas extremadamente bajas, acercándose al cero absoluto, para aprovechar los principios de superposición y entrelazamiento. A estas temperaturas tan bajas, ciertos materiales se convierten en superconductores, permitiendo el flujo de corriente eléctrica sin resistencia, un requisito crucial para mantener los delicados estados cuánticos de los qubits (Nielsen & Chuang, 2010).

Qiskit, una potente biblioteca de código abierto desarrollada por IBM, sirve como puente entre los usuarios y las computadoras cuánticas, simplificando el proceso de programación y manipulación de circuitos cuánticos. La accesibilidad basada en la nube de las máquinas cuánticas de IBM a través de Qiskit facilita la experimentación práctica con conceptos cuánticos, haciendo que la computación cuántica sea más accesible para investigadores, desarrolladores y organizaciones (IBM Qiskit Runtime, 2023).

Una característica destacada de las computadoras cuánticas de IBM Quantum es su capacidad única para generar números aleatorios, una tarea con implicaciones críticas en varios campos. Esta capacidad se aprovecha mediante procesos cuánticos fundamentales, como el túnel cuántico y el ruido de fase cuántica, proporcionando una sólida fuente de verdadera aleatoriedad. Un circuito cuántico básico, como se describe en investigaciones anteriores (Villaruel-Mosquera, 2023), desempeña un papel fundamental en este enfoque de generación de números aleatorios. Las computadoras cuánticas de IBM, impulsadas por Qiskit, demuestran una eficiencia igual en comparación con los generadores cuánticos tradicionales (Savvas et al., 2020).

Este enfoque innovador elimina la necesidad de que los usuarios configuren y mantengan hardware cuántico especializado, una tarea engorrosa y costosa. Además, las computadoras cuánticas de IBM ofrecen una plataforma en la nube, asegurando un acceso eficiente y la generación de números aleatorios sin necesidad de un conocimiento profundo de la programación cuántica (IBM Quantum Platform, 2023). En esencia, la integración de Qiskit y los recursos cuánticos de IBM no solo mejora la accesibilidad, sino que también demuestra ser una solución más efectiva para la generación de números aleatorios, marcando un avance significativo en el ámbito de la computación cuántica.

Generadores de números pseudoaleatorios

Los generadores de números pseudoaleatorios (PRNG, por sus siglas en inglés) son algoritmos que se utilizan en la computación actual para generar secuencias de números que parecen ser aleatorios, pero en realidad son deterministas y generados a partir de un valor inicial llamado "semilla". Estos números se utilizan en una amplia variedad de aplicaciones, desde simulaciones y juegos hasta criptografía y estadísticas (Press et al., 2007). Uno de los generadores de números pseudoaleatorios más conocidos y ampliamente utilizados es el generador Mersenne Twister, este es el generador principal de la mayoría de librerias que generan números pseudoaleatorios en la mayoría de los lenguajes de programación modernos.

El generador Mersenne Twister es apreciado por su larga longitud de período, que es la cantidad de números distintos que puede generar antes de que la secuencia se repita. La generación de números pseudoaleatorios en el Mersenne Twister, y en otros PRNG, se basa en una fórmula matemática que transforma la semilla inicial en una secuencia de números aparentemente aleatorios. Cada vez que se solicita un número aleatorio, el algoritmo calcula el siguiente número en la secuencia utilizando la siguiente función matemática y actualiza internamente el estado del generador

$$X_n = X_{n-(n-2)} \otimes ((X_{n-1} \,\&\, M) \otimes X_n) \gg 1 \otimes A$$

X_n es el número pseudoaleatorio generado en el paso n. X_{n-1}es el número generador en el paso anterior $(n-1)$. $X_{n-(n-2)}$ se refiere al número generado dos pasos atrás $(n-2)$. & representa la operación de AND bit a bit. $\oplus$ representa la operación de XOR bit a bit. M generalmente se refiere a una máscara que se utiliza para mezclar los bits del número generado. A es una

constante que también se utiliza en la fórmula para ayudar a mezclar y generar los números pseudoaleatorios. Y al operador >> denota la operación de desplazamiento a la derecha, que desplaza los bits de la expresión hacia la derecha en 1 posición.

La semilla inicial es crucial, ya que determina la secuencia completa de números generados. Si se conoce la semilla, se puede predecir la secuencia completa, por lo que la elección de una semilla inicial de alta entropía es importante para mantener la aleatoriedad aparente. Asimismo, aunque son periodos muy extensos hay muchas fallas, por ejemplo, ciertas subsecuencias pueden estar repitiéndose durante ciertos periodos (Vigna, 2019).

Los generadores de números pseudoaleatorios tienen limitaciones y problemas. Uno de los principales problemas es que, a pesar de su apariencia de aleatoriedad, los números generados son completamente predecibles si se conoce la semilla. Además, las secuencias de números generadas por PRNG no son realmente aleatorias en el sentido de que carecen de entropía real y no son adecuadas para aplicaciones de seguridad crítica, como la criptografía (Knuth, 1981).

PROBLEMAS DE OPTIMIZACIÓN

Optimización de parámetros en un Modelo de Comportamiento Sísmico EBP-EAB

El problema en cuestión involucra el estudio del comportamiento cíclico de las conexiones EBP-EAB (Placa Base Extendida - Pernos de Anclaje Empotrados), un aspecto crítico en la ingeniería estructural. Estas conexiones desempeñan un papel fundamental en garantizar la estabilidad y el rendimiento de las estructuras, especialmente durante eventos sísmicos. Para comprender y predecir mejor el comportamiento de estas conexiones, se ha desarrollado un modelo matemático, el modelo EBP-EAB.

Uno de los desafíos al estudiar las conexiones EBP-EAB es capturar con precisión su respuesta histérica, que incluye elementos como el comportamiento de pellizco y los modos de deterioro cíclico. Estos aspectos son difíciles de cuantificar a través de métodos analíticos tradicionales debido a la naturaleza compleja y no lineal de la respuesta.

Aquí es donde entran en juego los algoritmos de optimización inspirados en la biología. Estos algoritmos se inspiran en el mundo natural para abordar problemas de optimización complejos. En el caso de las conexiones EBP-EAB, estos algoritmos se utilizan para determinar parámetros auxiliares que describen el comportamiento de pellizco y los modos de deterioro cíclico, que no pueden estimarse de manera significativa mediante principios mecánicos convencionales (Torres-Rodas et al., 2023).

El papel destacado del algoritmo de ABC se evidencia especialmente en su aplicación para la estimación de parámetros que describen el comportamiento de pellizco y deterioro en las

conexiones EBP-EAB. Su utilidad radica en lograr un equilibrio óptimo al minimizar el error entre las predicciones del modelo y el comportamiento observado en simulaciones de elementos finitos, como se expresa en la fórmula:

$$\in = \frac{\int |M^{Abaqus} - M^{EBP-EAB}| \cdot |d\theta|}{\int |M^{Abaqus}| \cdot |d\theta|}$$

Ajustar el modelo EBP-EAB con el uso del algoritmo ABC posibilita una mejor correspondencia con el comportamiento observado, resultando en una mejora significativa en la precisión de sus predicciones. Mediante el ABC, las abejas exploran soluciones para cada variable dentro de un espacio proporcionado por los investigadores. Este espacio se diseña de manera suficientemente amplia para encontrar una solución óptima, pero sin exceder límites que pudieran generar demoras significativas en el rendimiento del algoritmo (Torres-Rodas et al., 2023).

Feature Selection

La selección de características es un proceso crucial en el aprendizaje automático y el análisis de datos, donde el objetivo es elegir un subconjunto de características relevantes (variables o atributos) de un conjunto más grande de características disponibles. El propósito es mejorar el rendimiento y la eficiencia de los modelos de aprendizaje automático, reducir el sobreajuste y mejorar la interpretabilidad de los resultados. Este proceso desempeña un papel vital en el manejo de datos de alta dimensionalidad, donde el número de características puede ser significativamente mayor que el número de muestras (Guyon y Elisseeff, 2003).

La selección de características es, en esencia, un problema de optimización, ya que

implica la búsqueda del subconjunto óptimo de características que maximiza ciertos criterios mientras minimiza otros. Los criterios pueden variar según el problema específico e incluir la maximización de la precisión predictiva, la minimización de los recursos computacionales o la mejora de la interpretabilidad del modelo (Liu & Motoda, 1998).

Para resolver eficazmente el problema de selección de características, los investigadores recurren con frecuencia a algoritmos de optimización inspirados en la biología. Los algoritmos genéticos, un tipo específico de algoritmo de optimización inspirado en la biología, han ganado popularidad en tareas de selección de características debido a su capacidad para manejar espacios de búsqueda de alta dimensionalidad y explorar una amplia gama de combinaciones de características (Holland, 1992).

La implementación de algoritmos genéticos en la selección de características ha demostrado ser una estrategia efectiva para abordar la dimensionalidad de datos en problemas de aprendizaje automático. Estos algoritmos buscan optimizar un conjunto de características seleccionando subconjuntos relevantes que mejoren el rendimiento del modelo. En aplicaciones médicas, la selección de genes en perfiles de expresión génica puede ayudar a identificar biomarcadores relevantes para enfermedades (Mollanoori & Meybodi, 2009). En visión por computadora, la selección de características puede mejorar la precisión de la detección de objetos (Yang, 2014). En finanzas, identificar características clave puede mejorar la predicción de rendimientos (Majhi y Dash, 2018). Asimismo, destaca la eficacia en áreas como la agricultura (Lukić et al., 2021).

Algoritmos genéticos para la selección de características, con el puntaje ReliefF como un criterio de aptitud crucial, están destinados a ser utilizados. La importancia de ReliefF radica en

su diseño específico para tareas de clasificación, evaluando la relevancia de las características según su capacidad para discriminar entre instancias de diferentes clases. Su robustez ante datos ruidosos, eficiencia en espacios de alta dimensionalidad y adaptabilidad a conjuntos de datos desequilibrados lo hacen adecuado para algoritmos genéticos que buscan identificar un subconjunto de características que maximice la precisión de la clasificación. La naturaleza no paramétrica de ReliefF y su interpretabilidad intuitiva contribuyen aún más a su efectividad, asegurando una selección enfocada de características informativas sin asumir linealidad o distribuciones específicas de características. Asimismo, es importante destacar que un valor más alto en la puntuación de relevancia indica un mejor desempeño. Esta puntuación para cada característica se calcula mediante la siguiente fórmula:

$$Relief(i) = \sum_{k=1}^{N} \left(\frac{diff(x_i, {x_{ik}}^{+})}{N} - \frac{diff(x_i, {x_{ik}}^{-})}{N} \right)$$

Donde, *N* es el número total de instancias en el conjunto de datos. x_i es el valor de la característica *i* para la instancia actual. ${x_{ik}}^{+}$ es el valor de la característica *i* para la *k*-ésima instancia más cercana con la misma clase. ${x_{ik}}^{-}$ es el valor de la característica *i* para la *k*-ésima instancia más cercana con una clase diferente. $diff(a, b)$ es una función que calcula la diferencia absoluta entre *a* y *b* (Yang & Honovar, 1998).

EXPERIMENTOS

Generación de números aleatorios y pseudoaleatorios

Se emplearon las computadoras cuánticas de IBM en conjunto con su interfaz de programación en Python, Qiskit, para la generación de más de 3 millones de números de 32 bits. Para llevar a cabo este proceso, se diseñaron y ejecutaron circuitos cuánticos, requiriendo que cada qubit pasara por una compuerta de Hadamard. La aplicación de la compuerta de Hadamard creó una superposición en cada qubit, similar a lanzar una moneda con igual probabilidad de obtener 0 o 1 (Nielsen & Chuang, 2010).

Posteriormente, los qubits fueron medidos y se seleccionó el backend deseado de IBM Quantum. En este caso, se eligió 'ibm brisbane' debido a su disponibilidad gratuita y la presencia de 127 qubits. El circuito fue transpilado para adaptarse al backend seleccionado y se ejecutó el trabajo en él, generando 300 circuitos, cada uno repetido 300 veces. Notablemente, con la ejecución de un solo circuito, se obtuvieron 90,000 números aleatorios. Dada la necesidad de un conjunto de datos sustancial que superara los 5 millones de números, se ejecutaron múltiples circuitos para acumular un conjunto completo de números aleatorios, asegurando un suministro amplio para su uso posterior en algoritmos de optimización inspirados en la biología. Todos los números aleatorios generados fueron almacenados meticulosamente en un archivo para su posterior integración en los procesos de optimización.

Por otro lado, los números pseudoaleatorios se obtuvieron mediante bibliotecas de dos lenguajes de programación: Python y MATLAB. En Python, se empleó la biblioteca 'random' con la función random.rand() para el problema de selección de características, mientras que en

MATLAB, se utilizó la función rand() para la Optimización del Comportamiento Cíclico de Conexiones EBP-EAB.

Con el fin de facilitar un análisis comparativo entre los dos tipos de números, se desarrolló una nueva función para simular el comportamiento de las funciones existentes, pero esta vez utilizando los números cuánticos aleatorios obtenidos de las computadoras cuánticas. Este paso garantiza una evaluación exhaustiva de la eficacia y las características distintivas de los números cuánticos en comparación con sus homólogos pseudoaleatorios.

Optimización de parámetros en un Modelo de Comportamiento Sísmico EBP-EAB

La configuración experimental para el modelo de comportamiento sísmico EBP-EAB involucra componentes críticos, siendo las principales herramientas MATLAB, OpenSees y un algoritmo de Optimización de Colonia de Abejas Artificiales (ABC) personalizado.

Iniciando el proceso con MATLAB, un entorno de programación versátil y ampliamente utilizado proporciona un sólido respaldo para simulaciones numéricas y tareas de optimización. Los investigadores suelen definir funciones objetivas y restricciones dentro de MATLAB, estableciendo un marco sistemático para la optimización de parámetros (Lohmann, 2019).

Para simulaciones sísmicas y análisis auténticos, los investigadores recurren a OpenSees, una plataforma de software de código abierto diseñada específicamente para análisis sísmicos y estructurales. Sus capacidades permiten la construcción de modelos complejos, la simulación de eventos sísmicos y el análisis del comportamiento estructural bajo diversos escenarios de terremotos.

El aspecto distintivo de esta investigación radica en el uso de un algoritmo ABC personalizado, desviándose de bibliotecas o paquetes de optimización preexistentes. Este

algoritmo fue elegido por su eficacia comprobada para converger hacia soluciones óptimas (Torres-Rodas et al., 2023).

Dentro de este algoritmo, establecemos el número máximo de iteraciones en 50 y el número máximo de agentes (abejas) en 30. La población inicial comprende valores aleatorios para las variables $v1, v2, v3, v4$, que representan las variables que rigen el comportamiento de pellizco $pinch\theta, pinchM$ y los modos de deterioro cíclico ($damage1, damage2$). Además, se establecieron límites para cada variable: $[0, 1]$ $[0, 1]$ $[0, 0.0025]$ $[0, 0.50]$. La aptitud de la solución de cada abeja se determina evaluando una simulación externa con OpenSees. La salida de la simulación se compara con datos objetivos para calcular la puntuación de aptitud. El algoritmo consta de tres fases.

La fase de abeja empleada genera nuevas soluciones modificando sus posiciones actuales en el espacio de decisiones, y luego se evalúa la aptitud de estas nuevas soluciones. En la fase de abeja observadora, las abejas seleccionan probabilísticamente soluciones según sus valores de aptitud. Estas abejas generan y evalúan nuevas soluciones, contribuyendo a la exploración del espacio de soluciones. La mejor solución se actualiza considerando los valores de aptitud tanto de las soluciones actuales como de la mejor solución encontrada hasta el momento. Finalmente, la fase de abeja exploradora se inicia si una solución no mejora durante un cierto número de iteraciones; en tales casos, se abandona y se genera una nueva solución al azar para reemplazarla, fomentando la diversidad en el espacio de búsqueda. El proceso concluye cuando se alcanza el número máximo de iteraciones.

Para comparar ambos tipos de números, en el código, se reemplazó la función rand() de MATLAB con una función personalizada para utilizar los números cuánticos aleatorios. Esta función tiene el mismo comportamiento, es decir, regresa una matriz de números entre 0 y 1,

pero con los números cuánticos normalizados.

Feature Selection

La base de datos que se utilizará contiene 84 características relacionadas con la información capturada por dispositivos de medición de sismos en volcanes. Contamos con 5 etiquetas: VT, que representa sismos Tectónicos Volcánicos; LP, que denota sismos de Largo Periodo; REGIONAL, que identifica eventos sísmicos de origen Regional; HB, que corresponde a eventos Híbridos o combinación de fuentes volcánicas y tectónicas; y finalmente, ICEQUAKE, que se refiere a eventos de Icequake (eventos sísmicos asociados al movimiento del hielo).

Las 84 características incluyen, entre otras, f1_t_mean: Valor medio de una característica relacionada con el tiempo (t); f2_t_std: Desviación estándar de una característica relacionada con el tiempo; f3_t_var: Varianza de una característica relacionada con el tiempo; f4_t_entropy: Entropía de una característica relacionada con el tiempo; f5_t_kurtosis: Curtosis de una característica relacionada con el tiempo.

Los datos provienen de sismómetros y otros dispositivos de monitoreo sísmico colocados alrededor del volcán Cotopaxi. Estos instrumentos miden diversas características sísmicas a lo largo del tiempo, proporcionando información sobre la naturaleza y las características de los eventos sísmicos. El conjunto de datos consta de alrededor de 1187 ejemplos, cada uno con 84 características, lo que resulta en un espacio de búsqueda de 2^{84}, que es un espacio extremadamente grande de posibilidades. Además, este conjunto de datos fue seleccionado debido a su comprobada eficacia en la selección de características en varios estudios e investigaciones (Venegas et al., 2019).

Se emplea un algoritmo genético utilizando la biblioteca DEAP para la selección de características, centrándose específicamente en la puntuación ReliefF como función de aptitud. La implementación se realiza en Python e incluye paralelización para una evaluación eficiente de los individuos.

El algoritmo genético utiliza un tamaño de población de 100 individuos durante 100 generaciones, con una probabilidad de cruce del 0.3 y una probabilidad de mutación del 0.4. La evaluación del rendimiento utiliza la puntuación *ReliefF* como función de aptitud, evaluada en paralelo. Los operadores genéticos incluyen *cxTwoPoint* para el cruce, facilitando el intercambio de material genético en dos puntos aleatorios entre individuos progenitores. La estrategia de mutación utiliza la función *mutFlipBit*, introduciendo pequeños cambios aleatorios al invertir bits con una probabilidad especificada. Esto promueve la diversidad genética. Además, la función *selTournament* sirve como operador de selección, implementando una competencia tipo torneo donde grupos de tres individuos compiten y se selecciona el mejor rendimiento. Colectivamente, estos operadores guían la exploración y explotación del espacio de soluciones durante la evolución del algoritmo genético.

El algoritmo genético se involucra de manera iterativa en los procesos de selección, cruce y mutación para los individuos. A lo largo de cada generación, la evaluación captura estadísticas esenciales como el mínimo, máximo, promedio y desviación estándar de las puntuaciones de aptitud. Además, se documenta al mejor individuo y su puntuación de aptitud asociada en cada generación. El algoritmo concluye su ejecución después de 100 generaciones, registrándose el tiempo total de ejecución. Posteriormente, utilizando varios clasificadores como Random Forest, Artificial Neural Network and Support Vector Machine dentro de un stratified 10-fold

crossvalidation se calculan las características seleccionadas del mejor individuo y varias métricas de evaluación, incluyendo precisión, sensibilidad, AUC y F1-score, para probar si la selección de características fue exitosa. Este proceso completo se repite 10 veces utilizando números pseudoaleatorios de la biblioteca random de Python y 10 veces utilizando números cuánticos aleatorios. Para realizar una comparación adecuada, se implementó, aparte de la función random en este lenguaje, las funciones *cxTwoPoint, mutFlipBit* y *selTournament* propias que utilizan los números aleatorios.

RESULTADOS Y DISCUSIÓN

	v1	*v2*	*v3*	*v4*	**Error (∈)**
Números aleatorios	0.9887	0.0602	0.0001	0.0000	4.1271
Numeros pseudoaleatorios	0.9875	0.0822	0.0005	0.0000	4.1165

Tabla 1: Resultados del modelo matemático EBP-EAB

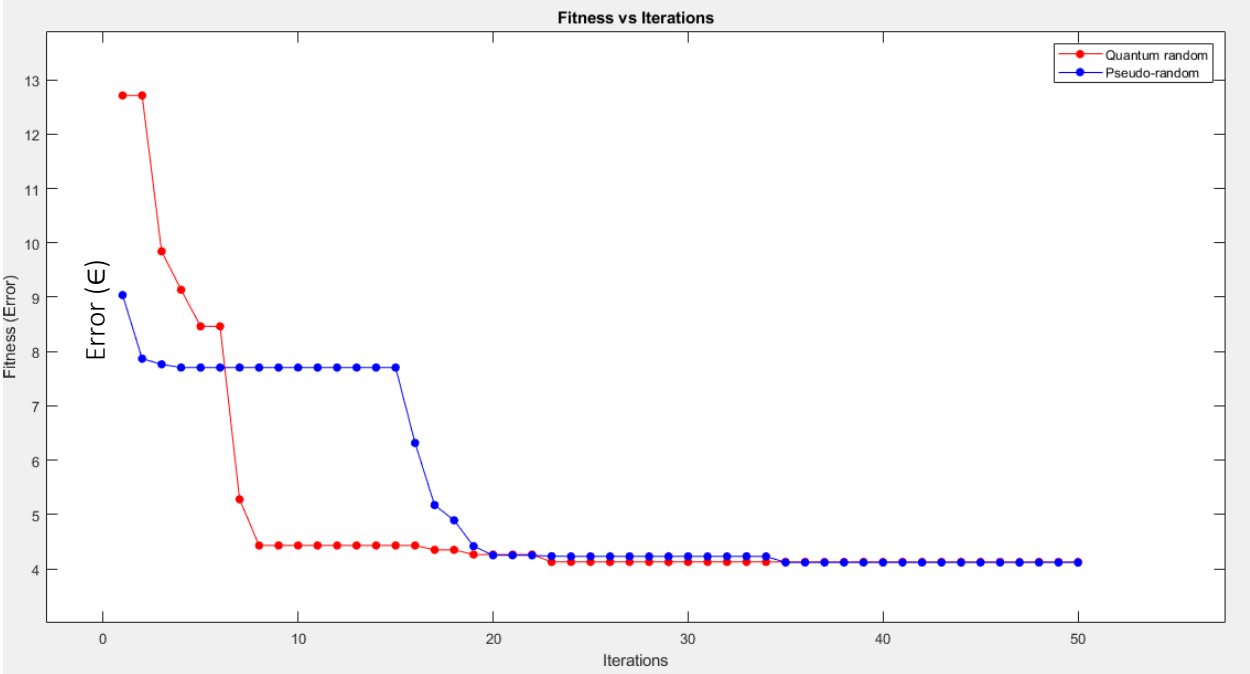

Figura 1: Error vs número de generación del modelo matemático EBP-EAB

Como se observa en la Tabla 1, tanto al emplear números aleatorios como pseudoaleatorios, el modelo matemático EBP-EAB con la aplicación del algoritmo ABC converge a valores casi idénticos; la diferencia es mínima. La Figura 1 ofrece una evaluación adicional, revelando que, utilizando números pseudoaleatorios, el proceso requirió 35 iteraciones para alcanzar la solución óptima, mientras que, en contraste, el uso de números aleatorios logró la convergencia en tan solo 23 iteraciones. En otras palabras, el modelo necesitó 12 generaciones menos al emplear números aleatorios en comparación con los pseudoaleatorios, sugiriendo que la

convergencia fue más rápida en el primer caso.

Número de ejecución	Fitness del mejor individuo	
	Números aleatorios	**Números pseudoaleatorios**
1	1.321	1.298
2	1.284	1.284
3	1.374	1.294
4	1.320	1.418
5	1.425	1.405
6	1.362	1.329
7	1.314	1.342
8	1.327	1.393
9	1.341	1.332
10	1.401	1.405
Promedio	1.3470	1.3499
Desviación estándar	0.0411	0.0487

Tabla 2: Fitness del mejor individuo por cada ejecución del algoritmo genético para feature selection

Número de ejecución	Número de generaciones hasta converger al óptimo	
	Números aleatorios	**Números pseudoaleatorios**
1	93	89
2	92	73
3	98	97
4	79	94
5	89	93
6	89	91
7	88	95
8	78	93
9	86	99
10	97	99
Promedio	89	93
Desviación estándar	6.3316	5.5578

Tabla 3: Número de generaciones hasta converger a la solución óptima por cada ejecución del algoritmo genético para feature selection

	RFC	SVM	ANN
Accuracy	0.9170	0.9127	0.9171
Precision	0.9038	0.9049	0.9027
Recall	0.9170	0.9127	0.9171
AUC	0.8166	0.7645	0.8420
F1 Score	0.9070	0.8848	0.9008

Tabla 4: Métricas de los clasificadores después del feature selection con números aleatorios del mejor individuo

	RFC	SVM	ANN
Accuracy	0.9144	0.9109	0.9170
Precision	0.8924	0.9108	0.9156
Recall	0.9144	0.9199	0.9170
AUC	0.7610	0.6834	0.8165
F1 Score	0.9015	0.8780	0.8904

Tabla 5: Métricas de los clasificadores después del feature selection con números pseudoaletorios del mejor individuo

Además, en la Tabla 2 se observa que prácticamente no existen diferencias significativas en el logro de un rendimiento de aptitud superior al utilizar conjuntos de números pseudoaleatorios o aleatorios. En otras palabras, se obtienen resultados esencialmente similares en promedio al emplear cualquiera de los conjuntos de números. No obstante, en la Tabla 3 se evidencia que, en promedio, se alcanza el resultado óptimo del algoritmo en menos generaciones al utilizar números aleatorios, con una reducción promedio de 4 generaciones. Esta observación sugiere una mejor convergencia al emplear números aleatorios. Finalmente, una vez realizado el proceso de selección de características, se procede a aplicar a varios clasificadores. En las tablas 4 y 5 se puede apreciar las métricas del clasificador después de la reducción en el número de características seleccionadas. Es importante destacar que no se observan diferencias significativas al utilizar números pseudoaleatorios o aleatorios.

CONCLUSIONES Y TRABAJO FUTURO

Aunque se descubren nuevos algoritmos en el ámbito de la computación cuántica, la computación híbrida es un enfoque razonable para aprovechar algunas de las ventajas que ya tiene la computación cuántica. En particular, esta investigación inicial muestra que el uso de Números Cuánticos Aleatorios (números verdaderamente aleatorios) tanto en ABC como en los algoritmos genéticos es tan efectivo como encontrar soluciones utilizando números pseudoaleatorios. Aunque, debido a su naturaleza verdadera, los números cuánticos aleatorios ayudan a que estos algoritmos converjan más rápidamente. Es razonable pensar que los números verdaderamente aleatorios permiten una caminata inicial aleatoria en el conjunto de posibles soluciones, lo que a su vez permite que el algoritmo explore de manera más uniforme el espacio de búsqueda y, por lo tanto, permita a los algoritmos metaheurísticos visitar soluciones mejores más rápidamente.

A pesar de que sabemos que se deben realizar más pruebas para confirmar esto en escenarios muy diferentes. También sería beneficioso mejorar la calidad de los números aleatorios con un verdadero Generador Cuántico de Números Aleatorios (QRNG) para realizar pruebas. Se podría realizar un análisis estadístico más detallado para comprender mejor las diferencias, si las hay, entre los conjuntos de números cuánticos y pseudoaleatorios. Esto podría incluir pruebas de significancia estadística y análisis de sensibilidad.

REFERENCIAS

Abbott, A. A., Calude, C. S., & Svozil, K. (2010). A Quantum Random Number Generator Certified by Value Indefiniteness. IEEE.

Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2007). The Traveling Salesman Problem: A Computational Study. Princeton University Press.

Bagci, I. E., Bernardo-Gavito, R., Roberts, J., Sexton, J., Astbury, B., Shokeir, H., ... & Young, R. J. (2017). Extracting random numbers from quantum tunneling through a single diode. Scientific Reports, 7, 1-6.

Braun, D. (2002). Quantum chaos and quantum algorithms. Phys. Rev. A, 65, 1–7.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics - Part B, 26(1), 29–41.

Fang, W., Zhang, L., Zhou, J., Wu, X., & Sun, J. (2017). A novel quantum-behaved particle swarm optimization with random selection for large-scale optimization. In 2017 IEEE Congress on Evolutionary Computation (CEC) (pp. 2746–2751).

Feynman, R. P. (1982). Simulating physics with computers. International Journal of Theoretical Physics, 21(6/7), 467–488.

Gabriel, C., Wittmann, C., Sych, D., Dong, R., Mauerer, W., Andersen, U. L., ... & Leuchs, G. (2010). A generator for unique quantum random numbers based on vacuum states. Nature Photonics, 4, 717-715.

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In 28th Annual ACM Symposium on the Theory of Computing (STOC) (pp. 212–219). Murray Hill, NJ: Bell Labs.

Guyon, I., & Elisseeff, A. (2003). Feature Selection: A Data Perspective. Journal of Machine Learning Research.

Herrero-Collantes, M., & Garcia-Escartin, J. C. (2016). Quantum Random Number Generators. IEEE.

Holland, J. H. (1992). Genetic algorithms. Scientific American, 267(1), 66–73.

IBM Qiskkit Runtime. (2023). IBM Qiskit Runtime. https://www.ibm.com/quantum/qiskit-runtime

IBM Quantum (2023). What is Quantum Computing?. https://www.ibm.com/topics/quantum-computing

IBM Quantum Platform. (2023). IBM Quantum Platform. https://quantum-computing.ibm.com/

Imada, A. (2007). Finding a needle in a haystack: From baldwin effect to quantum computation. In 6th International Conference on Computer Information Systems and Industrial Management Applications (CISIM'07) (pp. 20–25). IEEE.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Springer, 39, 459–471.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. IEEE Xplore.

Knuth, D. E. (1981). The Art of Computer Programming, Volume 2: Seminumerical Algorithms.

Addison-Wesley Professional.

Lei, W., Xie, Z., Li, Y., Fang, J., & Shen, W. (2020). An 8.4 Gbps real-time quantum random number generator based on quantum phase fluctuation. Quantum Information Processing, 19, 405.

Liu, H., & Motoda, H. (Eds.). (1998). Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers.

Lohmann, C., & Böhme, T. (2019). Optimization in MATLAB. Springer.

Ma, X., Yuan, X., Cao, Z., Qi, B., & Zhang, Z. (2016). Quantum random number generation. Quantum Information, 2016.

Majhi, R., & Dash, R. (2018). A Genetic Algorithm-Based Feature Selection for Improved Customer Churn Prediction. In Advances in Computing and Data Sciences: Second International Conference, ICACDS 2018, Dehradun, India, 20-21.

Maucher, M., Schöning, U., & Kestler, H. A. (2011). Search heuristics and the influence of non-perfect randomness: examining Genetic Algorithms and Simulated Annealing. IEEE, February 2011.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. Journal Name, Volume Number, Page Numbers.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in Engineering Software, 69, 46–61.

Mollanoori, H., & Meybodi, M. R. (2009). Feature Selection by Using Genetic Algorithm for the Cancer Classification. International Journal of Advanced Research in Computer Science and Software Engineering, 4(9).

Nielsen, M. A., & Chuang, I. L. (2010). Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). Numerical Recipes: The Art of Scientific Computing. Cambridge University Press.

Savvas, I. K., Chernov, A. V., & Butakova, M. A. (2020). Experiments with IBM Quantum Devices for Random Number Generation and String Matching. In 2020 28th Telecommunications Forum, TELFOR 2020 - Proceedings (pp. 24 November 2020, Article number 9306624).

Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings 35th Annual Symposium on Foundations of Computer Science (pp. 124–134). Santa Fe, NM, USA.

Steane, A. Quantum computing. Reports on Progress in Physics, vol. 61, pp. 117–173, July 1998

Stanco, A., Marangon, D. G., Vallone, G., Burri, S., Charbon, E., & Villoresi, P. (2020). Efficient random number generation techniques for CMOS single-photon avalanche diode array exploiting fast time tagging units. Physicial review research, 2.

Torres-Rodas, P., Medalla-Riquelme, M., Herrera, M., Lopez-Garcia, D., Benitez, D., & Camacho, O. (2023). A Mathematical Model for the Cyclic Behavior of Exposed Base Plates with Extended Anchor Rods.

Venegas, P., Pérez, N., Benítez, D., Lara-Cueva, R., & Ruiz, M. (2019). Combining Filter-Based Feature Selection Methods and Gaussian Mixture Model for the Classification of Seismic Events From Cotopaxi Volcano. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 12(6).

Villarruel-Mosquera, V. A., Zumárraga, M. G., Ospina, A., Riofrío, D., Benítez, D., Pérez, N., Grijalva, F., Flores Moyano, R., & Baldeon-Calisto, M. (2023). Hybrid-Computing for Finding Solutions to NP-Complete Problems in Graphs Using Ant Colony Optimization. In 2023 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC 2023). Ixtapa, Mexico.

Vigna, S. (2019). It Is High Time We Let Go Of The Mersenne Twister.

Yang, J. (2008). An Extended Genetic Algorithm for Feature Selection. IEEE Transactions on Neural Networks, 19(5), 726-732

Yang, H. H., & Honavar, V. (1998). A Genetic Algorithm-Based Approach for Feature Subset Selection. IEEE Transactions on Evolutionary Computation.